Solving Common-Payoff Games with Approximate Policy Iteration

by

Samuel Sokota

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computing Science

University of Alberta

Abstract

For artificially intelligent learning systems to be deployed widely in real-world settings, it is important that they be able to operate decentrally. Unfortunately, decentralized control is challenging. Even finding approximately optimal joint policies of decentralized partially observable Markov decision processes (Dec-POMDPs), a standard formalism for modeling these interactions, is a NEXP-complete problem. While there has been significant progress in developing strong teams of agents over the last decade, this progress has been fragmented among the engineering community, the Dec-POMDP community, and the deep multi-agent reinforcement learning (MARL) community. This thesis begins by reviewing the literature of each of these communities in unified language and formalizes a connection between recent developments exploiting common knowledge in common-payoff games and two-player zero-sum games. It then experimentally compares a select group of never-before-compared successful training paradigms. Lastly, and most significantly, it identifies and fills an algorithmic gap existing between the engineering and Dec-POMDP communities, which have proposed algorithms that are asymptotically optimal but difficult to scale, and the deep MARL community, which has proposed algorithms that, while scalable, have difficulty solving even small games, by proposing cooperative approximate policy iteration (CAPI), a novel algorithm for computing joint policies in common-payoff games. Experiments demonstrate that CAPI is capable of solving games that are orders of magnitudes larger than those that have been solved in existing literature.

Preface

This thesis is based on work done during my master's degree. Much of this work was done in collaboration with others. Edward Lockhart is especially deserving of attribution. He played played a significant role in the proposal of the cooperative approximate policy iteration algorithm. He also ran CAPI on abstracted tiny bridge. All other experiments, all of the writing, all of the theory, and any mistakes are my own.

Outside of the subject matter of this thesis, I also worked on approximate uncertainty quantification [66] and model-based reinforcement learning [80] during my master's degree.

Acknowledgements

There are many people deserving of acknowledgement for the growth and success I experienced during my master's degree: my parents for their endless support; my supervisors Marc Lanctot and Martha White for their invaluable mentorship and for allowing me the freedom to pursue my own ideas; my peers Ryan D'Orazio, Caleb Ho, Khurram Javed, Alexander Robey, and Muhammad Zaheer for stimulating conversation and patient explanations; and the Alberta Machine Intelligence Institute community, including Michael Bowling, Eric Graves, Russell Greiner, Cam Linke, Martin Müller, Matthew Schlegel, Richard Sutton, Adam White, Yi Wan, and James Wright, and the DeepMind community, including Thomas Anthony, Neil Burch, Edward Lockhart, and Martin Schmid, and many others, for facilitating my intellectual and personal development across a wide range of areas during the time I spent at The University of Alberta.

Contents

List of Figures

Chapter 1

Introduction

In reinforcement learning [72], an agent seeks to learn a policy that extracts a large cumulative reward from an environment, making it a valuable algorithmic tool for many control problems. But the classical reinforcement learning framework assumes that decision making is centralized in a single decision maker. In general, a control problem may require multiple decision makers to act independently. This problem setting is broadly referred to as decentralized control.

Despite involving multiple agents, the decentralized control problem bears relatively little similarity to general game-theoretic settings [55] in which agents possess adversarial incentives and there is not a generally agreed upon notion of optimality. It is also distinct from other problem settings operating under common-payoff game formalisms, such as ad hoc coordination [69], in which some of the agents are externally specified, or emergent communication [34], in which coordination must arise naturally (not from precoordinated learning procedures). Instead, in the decentralized control problem, an entity seeking to maximize cumulative reward specifies all agents and may precoordinate learning procedures.

While the decentralized control problem bears resemblance to the classical reinforcement problem in that both involve maximizing cumulative reward, decentralized control presents challenges that do not arise in classical reinforcement learning. One distinction is that, in decentralized control, agents have imperfect information, meaning that some have information that the oth-

ers do not have (and vice versa). Another is that dynamic programming is not directly applicable to agents in decentralized control problems because the value of an agent's information state depends on its teammates' policies.

One way to circumvent the latter issue is by alternating maximization [46]. Each agent optimizes its policy in turn, holding the policies of its teammates fixed. This procedure guarantees convergence to a local optimum, but can be arbitrarily far away from a global optimality and yields particularly poor results in games in which good performance requires agents to perform simultaneous exploration. A closely related approach is independent reinforcement learning (IRL) [73], a paradigm in which all agents concurrently execute reinforcement learning algorithms. Despite resting on more tenuous theoretical foundations than alternating maximization, IRL is popular among the deep multiagent reinforcement learning (MARL) community and has achieved significant empirical success in large games.

A second family of approaches is centered around the idea of common knowledge [47]. By conditioning on common knowledge, a team of decentralized agents effectively acts as a single agent, allowing for the direct application of dynamic programming. But while conditioning on common knowledge leads to solution methods capable of finding optimal policies for toy games, these methods are not immediately applicable to larger games, as conditioning on common knowledge comes at an exponential cost.

Unfortunately, while a number of important lines of research have emerged out of these approaches, they exist in largely disjointed communities and there has been relatively little interaction between them. This thesis takes a modest step toward addressing this issue by:

1. Offering discussion on the relationship between these lines of work in unified language and formalizing a connection between methods based on common knowledge and public subgame decomposition, a recent insight in the two-player zero-sum imperfect information game community.

2. Benchmarking a collection of successful, yet never-before-compared, training paradigms arising from these lines of work.

Finally, this thesis makes its main contribution by recognizing and filling a gap in existing literature. Lines of research coming out of engineering and Dec-POMDP communities have produced algorithms that, while recovering the optimal joint policy in small games, do not scale. Inversely, the deep MARL community has produced algorithms that, while scaling to very large games, fail to recover the optimal policy, even in very small games. To bridge this gap, this thesis proposes cooperative approximate policy iteration (CAPI), a novel approximate policy iteration algorithm for decentralized control. CAPI scales a framework originating in the engineering community using artificial neural networks, insights from the two-player zero-sum game community, and ideas from the deep MARL community. To demonstrate the efficacy of CAPI, this thesis employs two common-payoff games from OpenSpiel [33]. Having as many as tens of thousands of states and as many as hundreds of actions, these games are orders of magnitudes larger than the common-payoff games in existing literature that have been solved exactly [16]. CAPI achieves strong performance on these games, solving (discovering an optimal joint policy for) both games a majority of the time.

Chapter 2

Preliminaries

This chapter introduces game theory, common-payoff games, temporally-extended games, the decentralized control problem, Markov decision processes, partially observable Markov decision processes, dynamic programming, reinforcement learning, and deep reinforcement learning in language intended to be approachable for a reader who is already familiar with most (but perhaps not all) of the topics. Readers who require a more thorough introduction are referred to Shoham and Leyton-Brown [63] for game theory, common-payoff games, and temporally-extended games; to Oliehoek and Amato [49] for decentralized control; to Sutton and Barto [72] for Markov decision processes, partially observable Markov decision processes, dynamic programming, reinforcement learning; and to François-Lavet *et al.* [20] for deep reinforcement learning. A reader with a good understanding of the basics of all of these subjects may safely skip this chapter.

2.1 Game Theory

A game is a mathematical model describing a relationship between behaviors, technically referred to as joint policies (also known as policy profiles, joint strategies, and strategy profiles), and outcomes. Board games such as chess, checkers, and backgammon are among the most obvious examples. Many real-world applications, such as auctions, biological systems, consumer product pricing, security systems, and war bargaining, can also be formulated as games. But perhaps the most customarily used example of a game is the prisoner's

dilemma.

In the prisoner's dilemma, two criminals, Alice and Bob, are being interrogated for two charges. The major charge has a two year sentence. The lesser charge has a one year sentence. The prosecutor has insufficient evidence to convict Alice and Bob on the major charge, but sufficient evidence to convict them for the lesser. To incentivize Alice and Bob to testify against one another for the major charge, the prosecutor offers the following transpose deals to Alice and Bob, respectively:

- If Alice testifies against Bob for the major charge, the prosecutor will drop the lesser charge against her.
- If Bob testifies against Alice for the major charge, the prosecutor will drop the lesser charge against him.

Each of Alice and Bob must decide independently from the other (without knowledge of the other's decision).

There are four possible outcomes, which are more concisely described by Figure 2.1:

1. Alice and Bob remain loyal to one another and refuse the prosecutors deal. Both serve one year sentences for the lesser crime.
2. Alice betrays Bob and takes the prosecutor's deal. Bob remains loyal to Alice and refuses the prosecutor's deal. Alice gets off while Bob serves three years for the major and the lesser charge.
3. Bob betrays Alice and takes the prosecutor's deal. Alice remains loyal to Bob and refuses the prosecutor's deal. Bob gets off while Alice serves three years for the major and the lesser charge.
4. Both Alice and Bob betray one another and take the prosecutor's deal. Each serves two years for the major charge.

Among the most important questions about games like the prisoner's dilemma are: *What should the players do?* and *What will the players do?* Unfortunately, these questions are difficult to answer with generality because what a

Alice \ Bob	Take Deal	Refuse Deal
Take Deal	2y, 2y	0y, 3y
Refuse Deal	3y, 0y	1y, 1y

Figure 2.1: The prisoner's dilemma.

particular player should do is a function of what the other players will do and because what the players will do depends on who is playing the game.

A less ambitious question is: *Assuming participating agents are perfectly rational, what are the equilibria of the game?* The answer depends on the precise definition of the equilibrium concept, of which there are many. This thesis only requires understanding one—the Nash equilibrium. A joint policy is Nash equilibrium if no agent, after having been informed of the policies of the other agents, would have anything to gain by deviating from its policy. Every game is guaranteed to have at least one Nash equilibrium.

In the prisoner's dilemma, there is only one Nash equilibrium—both Alice and Bob take the deal. The reason this is a Nash equilibrium is that Alice receives a lesser sentence by taking the deal than by refusing if Bob is taking the deal, and vice versa. In contrast, other joint policies are not Nash equilibria because, independent of Bob's policy, Alice minimizes her prison time by taking the deal, and vice versa.

Another question that game theory asks of games like the prisoner's dilemma is: *What are desirable joint policies?* In general, this is a difficult question to answer because agents may have competing interests. But one criteria that is agreed upon is that a desirable joint policy should be Pareto optimal. A joint policy is Pareto optimal if no agent can be made better off without making another worse off. Every game is guaranteed to have at least one deterministic joint policy that is Pareto optimal.

In the prisoner's dilemma, there are three deterministic Pareto optimal joint policies:

- If both Alice and Bob refuse the prosector's deal, each receives one year in prison. For either Alice or Bob to receive less than one year in prison, the other must receive more than one year. Thus, refuse-refuse satisfies the

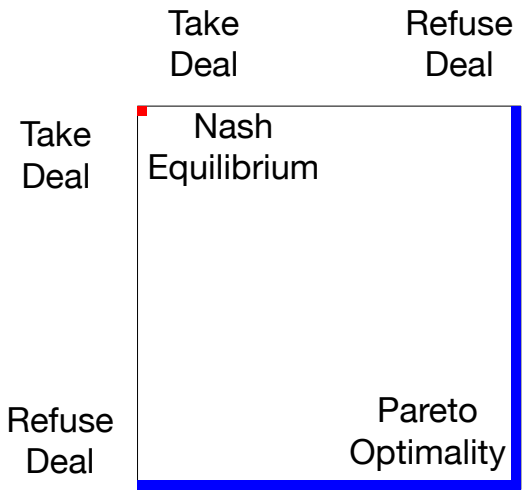

Figure 2.2: Analysis of the prisoner's dilemma.

criteria of Pareto optimality.

- If Alice takes the deal and Bob refuses the deal, Alice gets off with no prison time and Bob serves three years. For Bob to serve less than three years, Alice must serve prison time. Thus, take-refuse (and, by the same logic, refuse-take) are Pareto optimal.

There are also an infinite number of non-deterministic Pareto optimal joint policies. For example, Alice could refuse the deal and Bob could refuse the deal with probability 1/2 and take the deal with probability 1/2. In this example, Alice receives an expected sentence of 2 years and Bob receives an expected sentence of six months. One of them must be made worse off to make the other better off.

On the other hand, the deterministic joint policy take-take is a not a Pareto optimal joint policy because a player (in fact both players) would be better off under refuse-refuse without another player being worse off.

Paradoxically, not only is the Nash equilibrium of the prisoner's dilemma not Pareto optimal, it is the only not Pareto optimal deterministic joint policy in the game, as show in Figure 2.2. This puzzling property of the prisoner's dilemma illustrates a fundamental tension between the incentives of the individual and the welfare of the collective in strategic interactions: Each agent acting in its own best interest can lead to a collectively undesirable outcome.

2.2 Common-Payoff Games

Common-payoff games are a special subclass of games in which all participating actors experience the same outcome. One might think that by forcing participating actors to share the outcome, the tension between the incentive of the individual and the welfare of the collective would be resolved. Unfortunately, this is only true to a limited extent, as is illustrated by the following hiker's quandry.

After two years in prison, Alice and Bob, neither of whom hold grudges, have reunited. To celebrate their freedom, they decide to go backcountry hiking together at a local mountain. But unfortunately, on their way to the summit, they are separated. Having no cell phone service, and little chance of finding one another in the dense forest, Alice and Bob are each left with two options: To continue on to the summit or to head back to the car. Because Alice and Bob must make decisions independently, there are again four possible outcomes for the pair.

1. Alice and Bob both return the car. They are disappointed that they did not get to see the view at the summit but happy that they have found one another.

2. Alice goes to the summit while Bob goes to the car. Both are unhappy that they have not found one another. Alice cannot enjoy the view at the summit without the company of Bob. And Bob cannot drive home without first finding Alice.

3. Alice goes back to the car while Bob goes to the summit. Both are unhappy that they have not found one another. Alice cannot drive home without first finding Bob. And Bob cannot enjoy the view at the summit without the company of Alice.

4. Alice and Bob both go to the summit. They are happy that they are together and enjoying the scenic view.

These joint policies are described concisely in Figure 2.3 in terms of utilities.

Alice \ Bob	Car	Summit
Car	1	0
Summit	0	2

Figure 2.3: The hiker's quandry.

A utility is a real number associated to an agent and a joint policy that reflects that reflects the agent's degree of satisfaction with the joint policy.[1] In general, different agents may associate different utilities to the same joint policy, as is the case in the prisoner's dilemma. However, in common-payoff games, by definition, all agents associate the same utility (a.k.a. payoff) to each outcome. Therefore, it is only necessary to write a single utility for each outcome, as is done in Figure 2.3.

In common-payoff games, all Pareto optimal joint policies receive the same payoff and it is acceptable to simply refer to these joint policies as optimal, as is done hereinafter. In the hiker's quandry, there is only one optimal joint policy—both Alice and Bob proceed to the summit and enjoy the scenic view in each other's company.

On the other hand, there are two deterministic Nash equilibria: Either both Alice and Bob head back to the car or both Alice and Bob proceed to the summit. That the former is a Nash equilibrium may be puzzling at first glance—why wouldn't they both proceed to the summit and receive the higher payoff? The answer lies in a careful inspection of the definition of a Nash equilibrium. For a joint policy to be a Nash equilibrium, it need only be the case that no player stands to gain from unilateral deviation. If Alice is told that Bob is going to the car, she only stands to lose from going to the summit, and vice versa.

There is also one non-deterministic Nash equilibrium in the hiker's quandry: each of Alice and Bob go back to the car with probability 2/3 and go to the summit with probability 1/3. This is a Nash equilibrium because, given Alice's policy, Bob's expected utility for going to the car ($2/3 \times 1 = 2/3$) is the same as his expected utility for going to the summit ($1/3 \times 2 = 2/3$), and vice

[1] The actual values of the utilities are only important up to positive affine transformation.

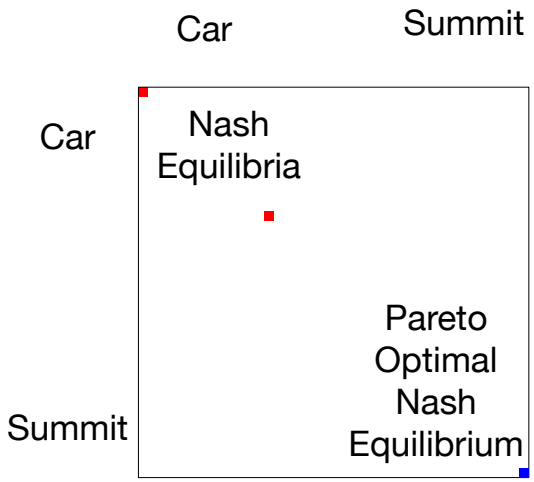

Figure 2.4: Analysis of the hiker's quandry.

versa. Thus, neither Alice nor Bob has any incentive to deviate, despite that this Nash equilibrium is even worse than the one in which they both go to the car.

The status of Nash equilibria in the hiker's quandry, shown in Figure 2.4, is reflective of broader properties of common-payoff games. In common-payoff games, all optimal joint policies are Nash equilibria and at least one of these is guaranteed to be a deterministic joint policy. This contrasts general-sum games (the general case in which agent utilities need not observe any particular structure), in which, as is observed in the prisoner's dilemma, there may not exist any Pareto optimal Nash equilibria. But like general-sum games, common-payoff games may have arbitrarily suboptimal Nash equilibria. For example, if the payoff for car-car were -10^{10} and the payoffs for car-summit and summit-car were $-10^{10} - 1$, car-car would still be a Nash equilibrium. Thus, while the tension between the incentives of the individual and the welfare of the collective is lessened in common-payoff games, it remains a significant challenge.

2.3 Temporally-Extended Games

Despite having prematurely ended their hike, Alice and Bob remain in a celebratory mood and head to the local casino to play guess the hat, their favorite

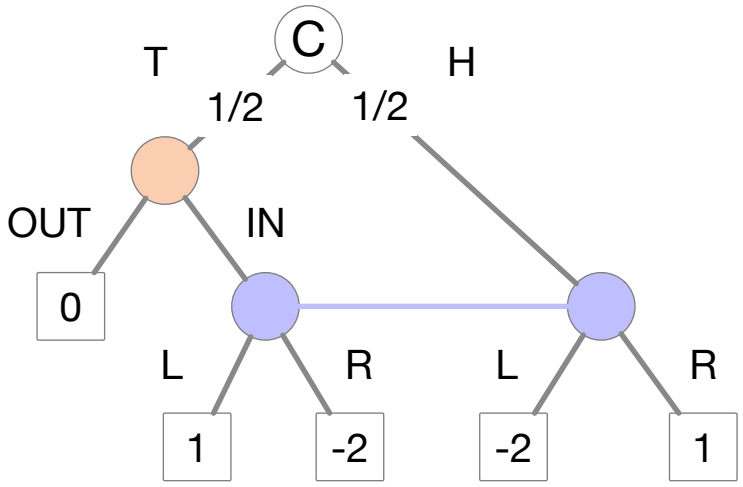

Figure 2.5: Guess the hat.

gambling game. The game has a $2 entrance fee and proceeds as follows.

1. First, the dealer flips a fair coin.

   - If the coin comes up heads, the dealer hides $3 under the right hat.
   - If the coin comes up tails, the dealer hides $3 under the left hat and asks Alice whether she would like to opt out of the game. If Alice opts out of the game, Alice and Bob are refunded their $2 entrance fee and the game ends.

2. If the game has not ended, Bob, who is unaware of both the outcome of the coin-flip and whether Alice has been given the option to opt out, guesses a hat.

   - If Bob guesses the hat with $3 underneath, Alice and Bob get to keep the $3 (netting $1 in aggregate).
   - Otherwise, Alice and Bob do not win any money (and are not refunded their $2 entrance fee).

The guess the hat game described above differs from the games that have been described so far in that it takes place over multiple time steps. While it is possible to express temporally-extended interactions in normal-form, the table representation that was used to express the prisoner's dilemma and the hiker's

quandry, it can be very inconvenient to do so. Instead, it is easier to represent such interactions in formalisms explicitly designed for temporally-extended interaction. Figure 2.5 expresses the guess the hat game in an extensive-form representation, one such formalism.

In extensive-form representation:

- The node labelled C (for chance) represents a probabilistic event.
- The apricot colored nodes represent decisions that Alice makes.
- The blue colored nodes represent decisions that Bob makes.
- The squares represent the end of the game. The numbers inside of the squares represent the utility corresponding to that trajectory.
- The horizontal line between Bob's rightmost nodes means that Bob is unable to distinguish the two nodes. In other words, Bob's policy must dictate a single distribution over actions for both of these nodes. Games in which players have access to different sets of information are called imperfect information games.

Because Bob is unable to distinguish the situation in which the $3 is under the right hat from the situation in which the $3 is under the left hat, it is not possible for Alice and Bob to win $1 every game. The best that can be done is for Alice to always opt out. That way, Bob knows that if the game proceeds, the $3 will be under the right hat. This optimal joint policy yields an expected $0.50.

By first glance, temporally-extended representations, having chance events and imperfect information, might appear more difficult to handle than normal-form games, for which the optimal policy can be found simply by scanning over the payoff matrix. One might ask: *Can we just convert the game to normal-form and then solve it?* Unfortunately, while this works for very small games, it is not a scalable approach. Converting an extensive-form game to normal-form is exponentially expensive. The size of the induced normal-form game quickly reaches astronomical proportions.

A reader may counter: *If we must deal with games in some temporally extended representation, can we at least get rid of imperfect information? It may be necessarily for gambling or board games, but in real-world applications we could equip the agents with communication devices.* This is a fair point, and in many applications it is indeed possible to remove imperfect information by equipping agents with communication devices. Unfortunately, it is not always so. In some applications, communication devices may be too expensive to be worth the cost. In others, they may pose a security risk. And even when agents can be equipped with communication devices, it does not necessarily guarantee perfect information—communication may be delayed or of uncertain reliability. Thus, imperfect information must be accepted as a reality of temporally-extended common-payoff games.

2.4 The Decentralized Control Problem

There are a number of important problem settings taking place within temporally-extended common-payoff games. In the ad hoc coordination setting [69], the objective is to train an agent to spontaneously cooperate with other, externally specified, agents. In the emergent communication setting [34], a team of agents must learn to communicate naturally, meaning there can be no ex ante (before the event) coordination. In the decentralized control setting, the objective is to produce a team of ex ante coordinated agents that generate a large utility.

There are many more specific variants of the above problem settings. This thesis regards a specific variant of the decentralized control problem. In the variant considered in this thesis, the team of agents is given an exact simulator for the common-payoff game of interest and may generate their joint policy by any means. In particular, this setting allows for centralized training, meaning that agents may communicate during training within the common-payoff game in ways that they may not be allowed during execution. Note that this setting does not allow for centralized execution, meaning that, during execution (i.e. test time) agents must obey the constraints of the game, which may not have

been adhered to during training. This variant of the decentralized control problem is henceforth referred to as the decentralized control problem with the understanding that the above qualifications apply.

The decentralized control problem can be thought of as a single entity seeking to maximize its utility in a control problem requiring multiple decision makers. In this respect, the decentralized control problem is not game-theoretic, as it involves no strategic interaction between agents. Instead, the decentralized control problem is better thought of as an optimization problem—a problem in which the objective is to maximize an objective function subject to some constraints. In particular, the decentralized control problem is about optimizing the utility function, subject to the constraint that the input be a joint policy.

2.5 Single Player Games

A special edge case of the decentralized control problem is common-payoff games in which there is only one player. Such games can be expressed as Markov decision processes (MDPs) or partially observable Markov decision processes (POMDPs).

2.5.1 Markov Decision Processes

An MDP is a tuple $\langle \mathcal{S}, \mathcal{A}, p \rangle$ where:

- $\mathcal{S}$ is a set of states.
- $\mathcal{A}$ is a set of actions.
- p is the dynamics function specifying the probability $p(s', r \mid s, a)$ of transitioning to the state $s' \in \mathcal{S}$ and the emitting the reward $r \in \mathbb{R}$ given the state $s \in \mathcal{S}$ and action $a \in \mathcal{A}$.

In an MDP, rather than receiving a utility at the end of the game, an agent may receive incremental rewards throughout the game. The agent's utility is

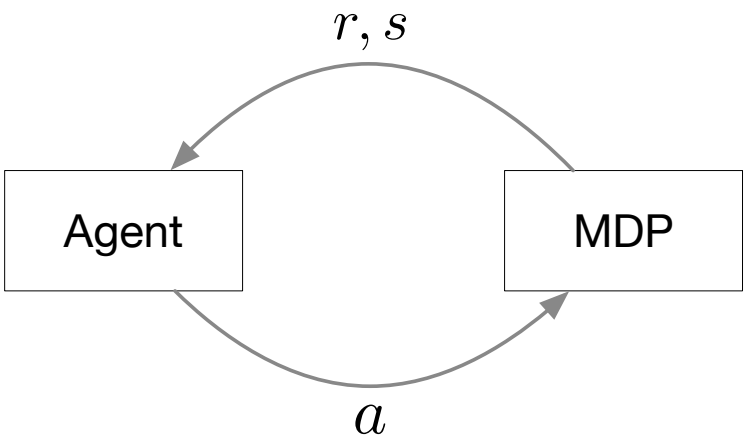

Figure 2.6: An agent interacting with an MDP.

equal to the sum of these rewards.[2]

The agent-MDP interaction loop is shown in Figure 2.6. At each time step, the agent decides an action a based on its current state and passes that action to the MDP. The MDP samples a reward r and a next state s using the dynamics function p, the current state, and the action a. The MDP passes r and s to the agent for its next decision.

2.5.2 Partially Observable Markov Decision Processes

That the next state and reward of an MDP depend only on the current state and action make MDPs easy to work with. But, as has been discussed, the agent may not be able to observe the underlying state of the world. In these cases, it is more natural to express the environment as a POMDP. A POMDP is a tuple $\langle \mathcal{S}, \mathcal{O}, \mathcal{A}, p \rangle$ where:

- $\mathcal{S}$ is a set of states.
- $\mathcal{A}$ is a set of actions.
- $\mathcal{O}$ is a set of observations.
- p is the dynamics function specifying the probability $p(s', o, r \mid s, a)$ of transitioning to the state $s' \in \mathcal{S}$ and emitting observation $o \in \mathcal{O}$ and reward $r \in \mathbb{R}$ given the state $s \in \mathcal{S}$ and action $a \in \mathcal{A}$.

[2]This thesis assumes a finite horizon setting. In an infinite horizon setting the agent's utility is either a discounted sum of rewards or the average reward.

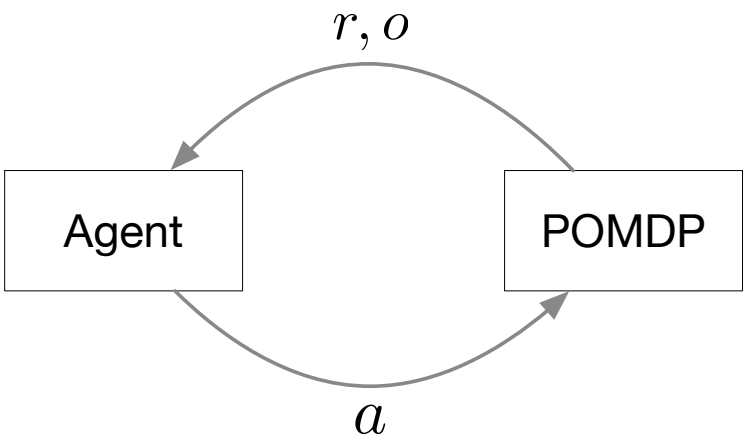

Figure 2.7: An agent interacting with a POMDP.

When interacting with a POMDP, the agent observes observations rather than the underlying state, as shown in Figure 2.7. This is a crucial difference. In an MDP, an optimal policy need only condition on the state most recently emitted by the environment. But in a POMDP, there generally does not exist an optimal policy that only conditions on the the most recent observation. There are two general theoretically sound ways of resolving this problem. Each amounts to converting the problem to an MDP. In the first, called a history MDP:

- Each state is a history $h = (o^0, a^0, r^1, o^1, \ldots, o^t)$ of the agent's interaction with the POMDP, from the perspective of the agent, where the superscripts denote the time step.
- Actions are those of the POMDP.
- For history $h' \equiv (h, a, o, r)$, the transition probability is given by

$$p(h', r \mid h, a) = \mathop{\mathbb{E}}_{s \sim p(S|h)} p(o, r \mid s, a).$$

Because the agent's history contains all information the agent has access to, an optimal policy for history MDP gives an optimal policy for the POMDP.

In the second, called a belief MDP:

- Each state is a distribution b over the states $\mathcal{S}$. These distributions are called belief states.

- Actions are those of the POMDP.
- Given a belief state b, an action a, a reward r and an observation o, the next belief state $b' = \tau(b, a, r, o)$ is defined by

$$b'(s') \propto \mathop{\mathbb{E}}_{s \sim b} p(s', o, r \mid s, a).$$

  The transition probability in the belief MDP is given by

$$p(b', r \mid b, a) = \sum_{o:\tau(b,a,r,o)=b'} \mathop{\mathbb{E}}_{s \sim b} p(o, r \mid s, a).$$

Belief states capture all information relevant for the agent's decision making. Thus, an optimal policy for the belief MDP gives an optimal policy for the POMDP.

2.6 Solving Single Player Games

The reason that single player games merit special attention is that there exist principled and efficient (in the case of MDPs) algorithms for solving them. This section discusses two closely related families of such algorithms for MDPs. The first, dynamic programming methods, assume white box access to the dynamics function p, meaning that the exact function is known. The second, reinforcement learning methods, assume black box access, meaning that p can be sampled but its internal values are not known.

2.6.1 Dynamic Programming

To introduce dynamic programming, it is first necessary to formally introduce the notions of a policy and value function. A policy $\pi \colon \mathcal{S} \to \Delta(\mathcal{A})$ is a function mapping from states to distributions over actions. Given some policy π, the corresponding value function $v_\pi \colon \mathcal{S} \to \mathbb{R}$ maps a state s to the expected reward that π accumulates from state s. The optimal value function v_* maps each state to the maximum expected cumulative reward that can be achieved from that state. Any policy π whose value function $v_\pi = v_*$ is optimal.[3]

[3]Additionally, any policy which achieves the optimal expected return may be informally referred to as optimal. This section uses the term formally. Later parts of the thesis use it informally.

Dynamic programming methods are based on Bellman's equations, which relate the expected value of states to the expected value of their successors.

Value Iteration

Value iteration uses the Bellman optimality equation

$$v_*(s) = \max_a \mathop{\mathbb{E}}_{r,s' \sim p(R,S|s,a)} r + v_*(s').$$

The Bellman optimality equation states that the optimal value of a state is equal to the expected sum of the reward and the optimal value of the next state, for the best action. By induction, it can be observed that the optimal value function satisfies the Bellman optimality equation and that a value function satisfying Bellman's optimality equation must be the optimal value function.

Value iteration works by beginning with an arbitrary real-valued function over the state space $v_0 \colon \mathcal{S} \to \mathbb{R}$ and iteratively applying the Bellman optimality equation as a backup operator

$$v_{k+1}(s) \leftarrow \max_a \mathop{\mathbb{E}}_{r,s' \sim p(R,S|s,a)} r + v_k(s').$$

Assuming finite horizon T, it is easily shown by induction that value iteration converges to the optimal value function after T iterations. Given the optimal value function, an optimal policy can be constructed using the greedification operator

$$\pi_*(s) \leftarrow \arg\max_a \mathop{\mathbb{E}}_{r,s' \sim p(R,S|s,a)} r + v_*(s').$$

The above equation should be understood as an abuse of notation by which it is meant that $\pi_*(s)$ is assigned such that its support is restricted to the arg max actions.

It also possible to do value iteration using state-action values instead of state values. In this case, value iteration uses the Bellman optimality equation

$$q_*(s,a) = \mathop{\mathbb{E}}_{r,s' \sim p(R,S|s,a)} r + \max_{a'} q_*(s',a').$$

Again, it follows from induction both that an action-value function is the optimal action value function if and only if it obeys Bellman's equations and

that applying the Bellman optimality equation as a backup operator

$$q_{k+1}(s,a) \leftarrow \mathop{\mathbb{E}}_{r,s' \sim p(R,S|s,a)} r + \max_{a'} q_k(s',a')$$

converges to the optimal state-action value function after T iterations for games of finite horizon T. Given the optimal state-action value function q_*, an optimal policy can be constructed using the greedification operator

$$\pi_*(s) \leftarrow \arg\max_a q_*(s,a).$$

Policy Iteration

Policy iteration uses the Bellman equation

$$v_\pi(s) = \mathop{\mathbb{E}}_{a \sim \pi(s)} \mathop{\mathbb{E}}_{r,s' \sim p(R,S|s,a)} r + v_\pi(s').$$

The Bellman equation states that the value of a state is equal to the expected sum of the reward and the value of the next state, when the action is sampled according to the policy. By induction, a value function is the value function corresponding to the policy π if and only if it satisfies π's Bellman equations.

Policy iteration works by alternating between evaluating and greedifying the policy. It begins with an arbitrary policy π_0. It evaluates π_k by iteratively applying the Bellman equation as a backup operator to each state. By induction, for a finite horizon T, the sequence converges to v_{π_k} after no more than T iterations. Given v_{π_k}, policy iteration produces π_{k+1} by applying the greedification operator

$$\pi_{k+1}(s) \leftarrow \arg\max_a \mathop{\mathbb{E}}_{r,s' \sim p(R,S|s,a)} r + v_k(s').$$

In aggregate, policy iteration looks like

$$\pi_0 \overset{\text{evaluation}}{\longrightarrow} v_{\pi_0} \overset{\text{greedification}}{\longrightarrow} \pi_1 \overset{\text{evaluation}}{\longrightarrow} \cdots \overset{\text{greedification}}{\longrightarrow} \pi_*.$$

Again, by induction, it can be observed that policy iteration coverges to an optimal after T iterations of evaluation and greedification, for an MDP of finite horizon T.

2.6.2 Reinforcement Learning

While dynamic programming offers an efficient path to an optimal policy, it requires white box access to the dynamics function p, meaning that the exact function must be known. In contrast, reinforcement learning methods only require black box access, meaning that it need only be possible to sample trajectories from p. Intuitively, this means that the agent is able to interact with the MDP as it would at execution time but that it is not privy to p itself. This sections reviews epsilon-greedy Q-learning and policy gradients, two foundational reinforcement learning algorithms.

Epsilon-Greedy Q-learning

The first, epsilon-greedy Q-learning, can be thought of as a form of approximate asynchronous (state-action) value iteration. Given a state s, with probability ϵ, epsilon-greedy Q-learning selects an action at uniform random and, with probability $1 - \epsilon$, selects an action in $\arg\max_a q(s, a)$. In other words, it acts randomly ϵ of the time and greedily the rest, hence the name epsilon-greedy. After acting, it receives $(r, s') \sim p(R, S|s, a)$ from the MDP and updates the state-action value $q(s, a)$ toward the target $r + \max_{a'} q(s, a')$. Note that this update rule is essentially the same as that of state-action value iteration, except that the target is sampled instead of taken in expectation and the update only moves $q(s, a)$ in the direction of the target rather than setting it exactly to the target. This difference is necessary to satisfy the requirement that reinforcement learning only assumes black box access to the MDP.

Epsilon-greedy Q-learning is considered approximate value iteration because it makes updates using samples rather than the full expectation. It is considered asynchronous because it may update some state-action pairs multiple times before updating other state-action pairs. Given a particular learning rate schedule, it can be shown that epsilon-greedy Q-learning converges asymptotically to the optimal state-action value function [7].

Policy Gradients

In contrast to value-based methods, policy gradient methods work by directly optimizing the policy. Observe that the utility of a policy can be written

$$u(\pi) \coloneqq \mathop{\mathbb{E}}_{(s^0,a^0,r^1,s^1,\ldots,r^T)\sim P(S^0,A^0,R^1,S^1,\ldots,R^T|p,\pi)} \sum_{t=1}^{T} r^t$$

where superscripts denote the time step. Taking the gradient of both sides, it follows that

$$\nabla_{\theta_\pi} u(\pi) = \mathop{\mathbb{E}}_{(s^0,a^0,r^1,s^1,\ldots,r^T)\sim P(S^0,A^0,R^1,S^1,\ldots,R^T|p,\pi)} \sum_{t=0}^{T-1} \nabla_{\theta^\pi} \log \pi(a^t \mid s^t) \sum_{t=1}^{T} r^t.$$

This equation is written in terms of an expectation over trajectories, exactly the form required to do updates with black box access to p. The policy gradient algorithm implemented using this equation is called *REINFORCE* [78].

In practice, it is unusual for policy gradient methods to be implemented using the above equation because it exhibits high variance. A lower variance, but also unbiased, equation for the gradient is

$$\nabla_{\theta_\pi} u(\pi) = \mathop{\mathbb{E}}_{(s^0,a^0,r^1,s^1,\ldots,r^T)\sim P(S^0,A^0,R^1,S^1,\ldots,R^T|p,\pi)} \sum_{t=0}^{T-1} \nabla_{\theta_\pi} \log \pi(a^t \mid s^t)(q_\pi(s^t,a^t) - v_\pi(s^t)).$$

This update rule is commonly seen in practice using concurrently learned estimates of q_π and v_π. The value $q_\pi(s^t,a^t) - v_\pi(s^t)$ is called the advantage.

In the tabular case, policy gradient methods can be shown to converge to the optimal policy under certain restrictions [8].

2.7 Deep Reinforcement Learning

While tabular dynamic programming and reinforcement methods are effective tools for solving MDPs, they are nevertheless too slow and memory intensive to be practical for very large MDPs. Deep reinforcement learning is a recent, rapidly expanding, line of research that aims to produce strong, though not necessarily optimal, policies for very large MDPs. The characterizing feature of deep reinforcement learning algorithms is that, rather than keeping tabular functions of state, it parameterizes functions of states with deep learning architectures.

Introducing deep learning into the training loop drastically increases the number of required design choices. There has been significant work on developing best practices regarding what transitions to replay [59]; how to compute learning targets from those transitions [25]; what learning architecture to use [15]; how to do distributed learning [27]; how to do recurrent learning [30]; how to do model-based reinforcement learning [60]; how to do exploration [5]; how to implement policy gradient methods [23], [61]; and many other subjects. While these matters are of crucial importance to the performance of reinforcement learning agents, they are each expansive topics in their own right. The body of this thesis will largely omit details on these subjects, including them only insofar as they are necessary to understand recent progress in decentralized control.

Chapter 3

Background and Notation

Roughly speaking, modern algorithms for decentralized control fall into one of two families:

1. Those based on aspirations of player-by-player optimality.

2. Those based on common knowledge.

This chapter discusses background necessary to understand player-by-player optimality and common knowledge and introduces suitable notation for describing algorithms based on these ideas.

3.1 Player-by-Player Optimality

The first family of approaches is based on notions of player-by-player optimality, also known as person-by-person optimality [40]. A joint policy is player-by-player optimal if and only if it is a Nash equilibrium. Formally, a joint policy π is a Nash equilibrium (or equivalently, player-by-player optimal) if the expected utility of π is greater than the expected utility achieved by a unilateral deviation from π. Symbolically, this is written as

$$u(\pi) = \max_i \max_{\pi'_i} u(\pi_{-i}, \pi'_i),$$

where u denotes the expected utility function and π_{-i} denotes the policies of each player except player i.

3.2 Common Knowledge

The second family of approaches is based on the idea of common knowledge. Common knowledge has long been a subject of investigation in philosophy [37], [76], multiagent systems [24], and (epistemic) game theory [4], [52], [53]. Let K_1^G be the set of information known to all agents in group G. Let K_{i+1}^G be the subset of K_i^G that is known by all agents to be in K_i^G. Then

$$K_{\text{common}}^G := \cap_{i=1}^{\infty} K_i^G$$

is **common knowledge** among G. The significance of common knowledge is that the inclusion status of any proposition is known by every member of the group. In general, the same cannot be said for the set K_i^G for any $i \in \mathbb{N}$. This distinction has important implications regarding the abilities of groups of agents to coordinate their actions. To gain intuition for this, consider the following thought experiment.

> Alice and Bob are participating in a cooperative game show. Partway through the game show, the host comes up behind Alice and Bob and places a red hat on the head of each. Because Alice and Bob are in plain view of one another, it is mutually clear that Alice knows the color of Bob's hat, and vice versa. But they are both unaware of the colors of the hats on their respective heads. After this, one of three events occurs.
>
> 1. The host does nothing.
>
> 2. The host informs Alice, in view of Bob but such that Alice is unaware that Bob is observing, that at least one of the two has a red hat, and vice versa.
>
> 3. The host publicly announces that at least one of Alice and Bob has a red hat.
>
> Following this event, the host asks: "Do you know whether your own hat is red?" Both Alice and Bob, obligated to respond honestly, answer "No". Now Alice and Bob must simultaneously and separately decide whether to

opt in or opt out. If both Alice and Bob opt out, they keep their existing winnings. If both Alice and Bob opt in and have red hats, they double their winnings. Otherwise, they lose their winnings. Assuming that it is common knowledge among Alice and Bob that both are loss-averse, self-interested, and have strong deduction skills, what happens?

In variant (1), the answer is that both Alice and Bob will opt out. Both knows that the other's hat is red but have no information about the color of their own hat. Therefore both of them will reason that the loss-averse decision is for both of them to opt out.

In variant (2), both Alice and Bob can reason that both of them have red hats. Like variant (1), each of Alice and Bob knows that the other's hat is red. Thus, each of Alice and Bob knows that at least one of them has a red hat. Additionally, as a result of event (2), each knows that the other knows that at least one their hats is red. Thus, prior to the host's question, both Alice and Bob know that both Alice and Bob know that at least one their hats is red. Each of Alice and Bob can use this information to reason about the other's answer to the host's question. Alice can reason that, because Bob knows that at least one of them has a red hat, Bob would've answered "yes" to the host's question if he had not seen that Alice had red hat, and vice versa. (If Bob knew that at least one of them had a red hat and also knew that Alice did not have a red hat then he would have deduced that his own hat was red.) Thus, after the host's question, both Alice and Bob can deduce that both Alice and Bob have red hats. However, Alice does not know that Bob was observing when the host informed Alice that at least one of the them had a red hat, and vice versa. Since this information was necessary to make the deduction above, neither Alice nor Bob is aware that the other knows that both have red hats. Thus, each is unwilling to risk only one of them opting in and opts out.

In variant (3), both Alice and Bob can reason that both agents have the correct answer using the same logic as in variant (2). However, unlike in variant (2), both can additionally reason that both of them know that both of them have the correct answer because both knew that they both had access

to the information necessary to make the deduction. In fact, it is common knowledge among the agents that both have the correct answer because it is common knowledge among the agents that both had access to the information necessary to make the deduction. Resultantly, they both reach the conclusion that it is safe to opt in.

The differences in these outcomes are characterized by the differences in the levels of knowledge. In all three variants, that fact ϕ—that at least one of Alice and Bob has a red hat—is a member of K_1^G. However, in variant (1), the fact ψ—that both Alice and Bob have a red hat—is not known.

On the other hand, in variant (2), $\psi \in K_1^G$ as a result of $\phi \in K_2^G$ and deduction about the answers to the host's question. Yet, since neither Alice nor Bob knows that $\psi \in K_1^G$, neither can be sure that the other will opt in. More generally, Alice and Bob would be forced to opt out for similar reasoning were $\psi \in K_i^G \setminus K_{i+1}^G$ for any $i \in \mathbb{N}$. For example, say that they agreed beforehand to opt in if $\psi \in K_i^G$. Then they both must know $\psi \in K_i^G$ to safely opt in, which is equivalent to agreeing to opt in if $\psi \in K_{i+1}^G$, which is in turn subject to the same logic.

This contrasts variant (3), in which each agent can be positive that the other will opt in because they are able to deduce that $\psi \in K_{\text{common}}$ from $\phi \in K_{\text{common}}$. The set K_{common} is special because $\psi \in K_{\text{common}}$ implies that every agent knows that $\psi \in K_{\text{common}}$.

Unfortunately, while common knowledge appears to be essential for coordination in many settings, the infinite regress that defines it can make it expensive to compute [14], [24], [32], [71], [79]. In imperfect information games, a recent effort to circumvent this issue focuses attention on public knowledge, a special subset of common knowledge that is easily computable [32]. Specifically, **public knowledge** is the subset $K_{\text{public}} \subset K_{\text{common}}$ of common knowledge that is immediate, rather than derived. For example, in variant (3), that at least one of Alice and Bob had a red hat is public knowledge (due to the host's public announcement), whereas that both Alice and Bob have red hats is common knowledge, but not public knowledge because it required deduction. Another, more realistic, example contrasting public and common knowledge

is as follows: *Alice shows Bob her driver's license.* Now it is public knowledge among Alice and Bob that Alice has her driver's license. But it is only common knowledge (and not public) that Alice has passed the driving test because it requires some amount of deduction from the fact that Alice has her driver's license.

3.3 Factored-Observation Games

Because this thesis deals both with methods based on player-by-player optimality and based on common knowledge, it is important that the choice of notation have machinery conducive to describing members of both families. Unfortunately, while there are many well-established formalisms for temporally-extended common-payoff games, including extensive-form games (EFGs), random variable notation, partially observable stochastic games (POSGs), and Decentralized POMDPs (Dec-POMDPs), none of them were designed with public knowledge in mind, and are thereby not well-equipped to describe methods that exploit it. Many of the methods discussed in this thesis rely on public knowledge.

Thus, so as to increase the clarity of presentation, this thesis eschews these more well-established notations in favor of a newly introduced formalism known as factored observation games (FOGs) [32]. FOGs are very similar to POSGs, but differ in that they possess machinery for handling public knowledge.

A FOG is a tuple $G = \langle \mathcal{N}, \mathcal{W}, w^0, \mathcal{A}, \mathcal{T}, \mathcal{R}, \mathcal{O} \rangle$ where

- $\mathcal{N} = \{1, \ldots, N\}$ is the **player set**.
- $\mathcal{W}$ is the set of **world states** and w^0 is a designated initial world state.
- $\mathcal{A} = \mathcal{A}_1 \times \cdots \times \mathcal{A}_N$ is the space of **joint actions**.
- $\mathcal{T}$ is the **transition function** mapping $\mathcal{W} \times \mathcal{A} \to \Delta(\mathcal{W})$.
- $\mathcal{R}$=$(\mathcal{R}_1, \ldots, \mathcal{R}_N)$ where $\mathcal{R}_i \colon \mathcal{W} \times \mathcal{A} \to \mathbb{R}$ is player i's **reward function**.
- $\mathcal{O} = (\mathcal{O}_{\text{priv}(1)}, \ldots, \mathcal{O}_{\text{priv}(N)}, \mathcal{O}_{\text{pub}})$ is the **observation function** where

- $\mathcal{O}_{\text{priv}(i)} \colon \mathcal{W} \times \mathcal{A} \times \mathcal{W} \to \mathbb{O}_{\text{priv}(i)}$ specifies the **private observation** that player i receives.
- $\mathcal{O}_{\text{pub}} \colon \mathcal{W} \times \mathcal{A} \times \mathcal{W} \to \mathbb{O}_{\text{pub}}$ specifies the **public observation** that all players receive.
- $O_i = \mathcal{O}_i(w, a, w') = (\mathcal{O}_{\text{priv}(i)}(w, a, w'), \mathcal{O}_{\text{pub}}(w, a, w'))$ is player i's **observation**.

In addition to the fundamental objects described above, there are also a number of important derived objects in FOGs.

- A **history** is a finite sequence $h = (w^0, a^0, \ldots, w^t)$. The notation $g \sqsubseteq h$ means that g is a prefix of h.
- The **set of histories** is denoted by $\mathcal{H}$.
- The **information state** for player i at $h = (w^0, a^0, \ldots, w^t)$ is

$$s_i(h) \coloneqq (O_i^0, a_i^0, \ldots, O_i^t).$$

- The **information state space** for player i is $\mathcal{S}_i \coloneqq \{s_i(h) \mid h \in \mathcal{H}\}$.
- The **legal actions** for player i at s_i is denoted $\mathcal{A}_i(s_i)$.
- A **joint policy** is a tuple $\pi = (\pi_1, \ldots, \pi_N)$, where each **policy** is of the form

$$\pi_i \colon \mathcal{S}_i \to \Delta(\mathcal{A}_i).$$

- The **cumulative reward** for player i for a terminal trajectory z is denoted

$$u(z) \coloneqq \sum_{ha \sqsubset z} \mathcal{R}_i(h, a).$$

- The **expected return** to joint policy π is for player i denoted

$$u_i(\pi) \coloneqq \mathop{\mathbb{E}}_{z \sim P(Z|\mathcal{T}, \pi)} u_i(z).$$

- The **public state** at h is the sequence $s_{\text{pub}}(h) \coloneqq s_{\text{pub}}(s_i(h)) \coloneqq (O_{\text{pub}}^0, \ldots, O_{\text{pub}}^t)$.

- The **public tree** $\mathcal{S}_{\text{pub}}$ is the space of public states.
- The **public set** for $s \in \mathcal{S}_{\text{pub}}$ is $I_{\text{pub}}(s) \coloneqq \{h \mid s_{\text{pub}}(h) = s\}$.
- The **information state set** for player i at $s \in \mathcal{S}_{\text{pub}}$ is

$$\mathcal{S}_i(s) \coloneqq \{s_i \in \mathcal{S}_i \mid s_{\text{pub}}(s_i) = s\}.$$

- The **reach probability** of h under π is $P^\pi(h) = P_{\mathcal{T}}(h) \prod_{i \in \mathcal{N}} P_i^\pi(h)$ where
  - Chance's contribution is $P_{\mathcal{T}}(h) \coloneqq \prod_{h'aw \sqsubseteq h} \mathcal{T}(h', a, w)$.
  - Player i's contribution is $P_i^\pi(h) \coloneqq P_i^\pi(s_i(h)) \coloneqq \prod_{s_i' a \sqsubseteq s_i(h)} \pi_i(s_i', a)$.

There are a number of important details about FOGS deserving of further discussion.

- World states in FOGs are Markov and play an analogous role to states in POMDPs.
- The transition function in FOGs is analogous to the dynamics functions in MDPs and POMDPs. But whereas the dynamics function described in Section 2.5 gives a joint probability distribution over the next Markov state, reward, and observation, the transition function in FOGs gives a distribution over only the next world state, which (in combination with the current world state and joint action) deterministically specifies the players' rewards and observations. This choice does not reduce the expressive power of FOGs, but does mean that a greater number of world states may need to be included.
- In FOGs, a player's information state is its history of actions and observations but not its rewards. FOGs implicitly assume that a player's reward history can be losslessly reconstructed from its action observation history (this constraint can easily be satisfied by including the reward in the observation).
- In common-payoff FOGs, for any terminal history z, and for two players i and j, it must be the case that $u_i(z) = u_j(z)$. The remainder of this thesis deals exclusively with common payoff FOGs.

- FOGs assume timeability, meaning that players can always agree on how much time has passed, and perfect recall, meaning that the game never forces a player to forget specific information that it already knows. POSGs and Dec-POMDPs also assume timeability and perfect recall, whereas EFGs assume neither. This thesis further assumes a finite horizon, meaning that there exists some positive integer T such that every trajectory terminates after no more than T steps.

Chapter 4

The Player-by-Player Approach

Computing an ϵ-optimal joint policy for Dec-POMDPs is a provably NEXP-complete problem [56]. This is a discouraging result. Informally, a NEXP-complete problem is a problem for which it takes an exponential amount of time to verify that a proposed solution is valid. For comparison, it takes only a polynomial amount of time to verify solutions to NP-complete problems, a class of problems which is itself regarded as very hard.

In practical terms, this means that, for any algorithm that is guaranteed to produce an approximately optimal solution, there exists a common-payoff game for which that algorithm takes an intolerably long time to terminate. More straightforwardly, **there is no scalable algorithm with approximate optimality guarantees.**

This dismaying fact leads to valid skepticism regarding the usefulness of optimality guarantees. *If performance in practice is the priority, does it matter whether algorithms are asymptotically (approximately) optimal? What if there exist heuristic algorithms that are not guaranteed to produce approximately optimal solutions but perform better in practice than those for which guarantees exist?*

This chapter takes the perspective of the skeptic. That is, it takes the perspective that to achieve good performance in practice, it may not be important for algorithms to have global optimality guarantees. In particular, this chapter regards algorithms designed based on notions of player-by-player optimality. While some algorithms it discusses do have formal guarantees

Alice \ Bob	left	right
left	0	1
right	1	0

Figure 4.1: Evasion.

of player-by-player optimality, player-by-player optimal joint policies can be arbitrarily globally suboptimal. Others, though inspired by notions of player-by-player optimality, operate with no guarantees at all. Nevertheless, as is detailed hereinafter, these algorithms can achieve good performance in practice.

4.1 Alternating Maximization

This section discusses the theoretical foundations of player-by-player approaches. In particular: *How can a team of agents find a player-by-player optimal joint policy?* Perhaps the most obvious approach would be to have the agents run independent dynamic programming algorithms, each considering the policies of its teammates as part of the dynamics function. This approach does work, but there is a caveat. The problem is that the value of an agent's information state depends on its teammates' policies in addition to its own. If the agents perform simultaneous dynamic programming updates, each (falsely) assuming that its teammates' policies will stay fixed, policy improvement is not guaranteed and the procedure may cycle. To illustrate this problem, consider the following one-shot perfect information mutual evasion game.

Alice and Bob are upset with one another and would like to avoid interacting. They live together in a two room house and each must decide whether to go to the left room or the right room. Both are happy if they select different rooms and both are unhappy if they select the same room, as described in Figure 4.1.

If Alice and Bob independently run dynamic programming with simultaneous updates, they run the risk of cycling, as is shown in Figure 4.2. In narrative form, one might imagine that both Alice and Bob initially go to the left room. Both Alice and Bob are unhappy about this situation, and each,

Iteration	q_{Alice}	q_{Bob}	Greedy Action (Alice)	Greedy Action (Bob)
0	$(\cdot, \cdot)$	$(\cdot, \cdot)$	left (arbitrary tiebreak)	left (arbitrary tiebreak)
1	(0, 1)	(0, 1)	right	right
2	(1, 0)	(1, 0)	left	left
3	(0, 1)	(0, 1)	right	right
⋮	⋮	⋮	⋮	⋮

Figure 4.2: Synchronous independent dynamic programming in evasion.

falsely presuming that the other will not move, moves to the right room. Both repeat this logic again, moving back to the left room, and so forth, never converging to a policy in which they choose different rooms.

The natural way to resolve this issue is to perform alternating dynamic programming updates among the players. By having only one agent update its policy at a time, the presumption that the policies of the other agents is fixed holds true and policy improvement is guaranteed. And since there are a finite number of deterministic joint policies, and there is guaranteed to be at least one deterministic player-by-player optimal joint policy (by virtue of the fact that there is guaranteed to be at least one deterministic optimal joint policy) it is easy to see that this process will converge to a player-by-player optimal solution after a finite amount of time. This approach, referred to as alternating maximization, coordinate ascent, iterated best response, or hill climbing, is detailed in Algorithm 1. The subroutine optimize($\pi_i \mid \pi_{-i}$) optimizes player i's policy as if player i were interacting with a single-agent game with the transitions in part dictated by π_{-i}. Any method that guarantees policy improvement may be used. In small common-payoff games, tabular forms of alternating maximization can easily be implemented that achieve their player-by-player optimality guarantees in practice.

Algorithm 1 Alternating Maximization

```
procedure OPTIMIZE(π)
    while π not converged do
        for i ∈ 𝒩 do
            optimize(π_i | π_{-i})
```

4.2 Scaling Player-by-Player Algorithms

Recently, there has been much interest in scaling learning algorithms to large settings. In the single-agent setting, this has involved equipping reinforcement learning algorithms with artificial neural networks. Reinforcement learning algorithms are preferred over dynamic programming algorithms because they are better equipped to handle unwieldy dynamics functions. And artificial neural networks allow algorithms to be scaled to settings that would be too time and memory intensive to handle tabularly.

In the multi-agent setting, scaling up alternating maximization in an analogous fashion is expensive—each player must collect an entirely new batch of data to do its maximization (because its teammates' policies have changed). In practice, the deep MARL community most often simply ignores the possibility of cycling and operates within an independent reinforcement learning (IRL) framework in which the agents synchronously and independently execute their own reinforcement learning algorithms, as shown in Algorithm 2. While the convergence properties of IRL are tenuous, it is easy to implement and often performs well in practice. Moreover, as the introduction to this chapter discussed, it is not even clear that theoretical guarantees should be a point of emphasis to entities concerned with performance in practice.

Algorithm 2 Independent Reinforcement Learning

procedure RUN EPISODE
  for $t = 1 \dots T$ **do**
    for $i \in \mathcal{N}$ **do**
      $a_i^t \leftarrow i.\text{act}(s_i^t)$
    env.step(a^t)
    for $i \in \mathcal{N}$ **do**
      $i.\text{update}(s_i^t, a_i^t, r_i^{t+1}, s_i^{t+1})$

The remainder of this chapter describes research in the deep MARL community that builds upon the idea of simply running deep reinforcement algorithms concurrently. Roughly speaking, those described fall into three categories (i) those modifying experience replay to better handle the nonstationarity induced by the presence of other agents; (ii) those exploiting centralized

training by means of a shared value function; (iii) those signaling exploration. While these areas are by no means exhaustive of the work that has been done in deep MARL, they do represent some of the more significant recent progress.

4.2.1 Stabilizing Experience Replay

Even in the single player case, introducing deep learning to reinforcement causes significant instability. If updates are made online, meaning that each transition is used to update the network immediately after it occurs and is thereafter discarded, the network's learning progress may be inhibited by the fact that its training examples are highly correlated. One of the important insights made by DQN [43], the seminal breakthrough in deep reinforcement learning, is that learning stability can be drastically improved by updating the network with batches which are randomly sampled from the last `buffer_size` transitions. This idea is commonly referred to as experience replay.

In a single-agent setting, transitions continue to offer useful information even long after they have occurred because the dynamics function is fixed. Thus, in addition to stabilizing network updates, experience replay can improve sample efficiency.

On the other hand, in multi-agent setting, under the IRL paradigm, from the perspective of any given agent, the dynamics function involves both the transition function of the underlying game and the policies of the other players. This means that learning from old transitions may be counterproductive because they are no longer reflective of the policies of the other agents.

Foerster *et al.* propose two ways of remedying this problem [18]. The first is by importance sampling. Importance sampling is a method that, given sampling access to one distribution, reweights the samples of this distribution so that a property of another distribution can be estimated. As an example, take some distribution μ. By reweighting each sample from μ by $\tilde{\mu}/\mu$ the expectation of $\tilde{\mu}$ can be estimated

$$\mathop{\mathbb{E}}_{x_j \sim \mu} \frac{\tilde{\mu}(x_j)}{\mu(x_j)} x_j = \sum_{x_j} \mu(x_j) \frac{\tilde{\mu}(x_j)}{\mu(x_j)} x_j = \sum_{x_j} \tilde{\mu}(x_j) x_j = \mathop{\mathbb{E}}_{x_j \sim \tilde{\mu}} x_j.$$

Foerster *et al.* propose using importance sampling to reweight the loss of old

transitions according to the probability that they would occur with the players' current policies.

The second remedy Foerster *et al.* propose is augmenting the agent state with information about the exploration rate and episode number. This allows agents to learn from old experience while simultaneously differentiating it from contemporaneous experience.

4.2.2 Learning a Central Value Function

Another line of research involves exploiting the centralized training for decentralized execution paradigm. One way in which this paradigm can be exploited is by training with a centralized value function. A centralized value function is a value function that takes as input more information than is accessible to any one agent. As long as each agent's policy is not centralized and the centralized value function is only required for training, the team of agents can still execute decentrally while obeying the constraints of the game.

Lowe *et al.* and Foerster *et al.* were among the first to propose doing this [17], [38]. Though they differ at a detailed level, the basic idea of both papers is to perform independent policy gradient algorithms using a centralized value function. For example, each agent might perform an update using an equation resembling

$$\nabla_{\theta^{\pi_i}} \log \pi_i(a_i \mid s_i)(q_\pi(w, a) - v_\pi(w)).$$

This update differs from independent policy gradients in that the advantage is computed using the world state w and joint action a as input, rather than player i's information state s_i and action a_i.

Sunehag *et al.* were also among the first to propose doing this, but with independent Q-learning rather than policy gradients [70]. In particular they perform Q-learning updates using the loss

$$\text{MSE}\left(\sum_{i\in\mathcal{N}} q_i(s_i^t, a_i^t), \sum_{i\in\mathcal{N}} r_i^{t+1} + \max_{a'} q_i(s_i^{t+1}, a_i')\right).$$

This can be thought of as using a centralized Q-function $q\colon \mathcal{S}_1 \times \cdots \times \mathcal{S}_N \times \mathcal{A}_1 \times \cdots \times \mathcal{A}_N \to \mathbb{R}$ under the assumption that this Q-function decomposes

additively across players. Because the centralized Q-function decomposes additively by construction, each player is able to construct its part of the argmax joint action at execution time.

Rashid *et al.* build on this idea using the insight that the centralized value function need not be constrained to decompose additively [57]. As long the partial gradient of the centralized value function with respect to each player's input is positive, each player can reconstruct its component of the argmax joint action at execution time. Rashid *et al.* propose exploiting this insight by parameterizing the centralized value function using $q_w \colon \mathbb{R}^N \to \mathbb{R}$ under the constraint that the network uses only positive weights (thereby ensuring that the necessary partial derivatives are positive). In total, a centralized value looks like $q_w\left(q_1(s_1, a_1), \ldots, q_N(s_N, a_N)\right)$. Note how not assuming an additive decomposition allows the decomposition to be world state specific.

4.2.3 Signaling Exploration

A third line of research is concerned with multi-agent exploration. It seeks to address the problem that each time an agent explores, its teammates are unaware of its exploration. The reason that this can be problematic is that agents may associate the ensuant outcome with their own actions, rather the exploration of their teammate(s).

Simplified Action Decoding

Hu and Foerster suggest resolving this issue via the centralized training for decentralized execution paradigm [28]. In particular, they suggest that the acting agent announce its greedy action to the other players during training. By doing so, each agent is made to understand whether the acting player is exploring or not (the action that was taken matches the announced greedy action if and only if exploration did not occur). In games with publicly observable actions, the agents can operate on the same input during test time because all agents are known to be acting greedily (i.e., the observed action is also the greedy action). Hu and Foerster refer to this as simplified action decoding (SAD).

While the original version of SAD only works in games with public observable actions, it can be modified to work in games in which actions are not public observable. These modifications are most easily explained using commands. In the original version of SAD, agents command acting players: *Tell me your greedy action.* That this command requires information even if the acting agent is not exploring precludes its applicability to games in which that information is not available at test time. The key modification that makes other SAD variants compatible with games in which actions are not publicly observable is that they use commands of the form: *If you explored, tell me X. If you did not explore, tell me nothing.* Commands of this form are compatible with private actions because they do not require any additional information at test time (since all agents are acting greedily). Three SAD variants that are compatible with private actions, binary SAD, action SAD, and private SAD, are described below.

- Binary SAD (BSAD) was independently proposed by Edward Lockhart and Jakob Foerster through personal communication. BSAD agents make the command: *Tell me whether or not you explored.* BSAD is the simplest form of SAD.

- Action SAD (ASAD) is novel to this thesis. ASAD agents make the command: *If you explored, tell me your greedy action. If you did not explore, tell me nothing.* ASAD is the most faithful way to extend the original version of SAD to games with private actions.

- Private SAD (PSAD) was proposed by Edward Lockhart through personal communication. PSAD agents makes the command: *If you explored, tell me your private information. If you did not explore, tell me nothing.* PSAD is the logical extreme of information sharing within the SAD family.

Chapter 5

The Common Knowledge Approach

The previous chapter took the position that it may not be useful to pursue algorithms with approximate optimality guarantees for the purpose of achieving good performance in practice. To justify its position, it invoked the fact that complexity results lead to the conclusion that there can be no scalable algorithm with approximate optimality guarantees. This chapter takes the opposite perspective. Although there exist games for which algorithms with approximately optimal guarantees take an intolerably long time to terminate, these games may be small in number or may not be representative of games that have practical relevance. Furthermore, algorithms with approximately optimal guarantees may offer useful intuitions on top of which to build heuristic algorithms with greater practical relevance. Thus, this chapter regards algorithms that either themselves have guarantees of global optimality, or are inspired by those that do. As will become clear, the idea of common knowledge is essential to this pursuit.

5.1 Computing Optimal Joint Policies

In order to discuss how it is possible to compute an optimal joint policy, it is helpful to consider, at a high level, why it is that player-by-player approaches suffer from spurious local optima. The answer, in short, is because player-by-player approaches optimize each player's policy independently. They are

susceptible to suboptimal joint policies that can only be improved by multilaterial deviation. Take the hiker's quandry, described in Figure 2.3 as an example. A strictly player-by-player approach beginning in the deterministic joint policy car-car can never escape because joint policy improvement requires Alice and Bob to simultaneously shift their policies. The identification of this deficiency of player-by-player approaches leads to the intuition that, to compute a globally optimally policy, it may be necessary to directly optimize the joint policy, without factoring updates by player.

In common-payoff games with perfect information, there is an obvious way of doing this. At each time step, a coordinator observes the world state w and selects a joint action a. Each player plays its corresponding component of the joint action as instructed by the coordinator. From this perspective, the coordinator is facing an MDP in which the states are the world states of the common-payoff game, the actions are the joint actions of the common-payoff game, and the reward and dynamics function are induced by the common-payoff game. Clearly, there is a bijection between policies in this MDP and the joint policies in the common-payoff game for which identified policies and joint policies receive the same expected return. Thus, to solve a common-payoff game with perfect information, it suffices for the coordinator to solve the corresponding MDP. One can imagine that this perspective could be executed decentrally by having each agent carry around an identical copy of the coordinator. Or alternatively, the coordinator could write out its policy and each player could memorize its part.

Now consider a slightly more general case—games in which all actions and observations are publicly observable but the underlying world state is not. This case can be solved by a similar approach. At each time step, the central coordinator observes the information state $s_1 \equiv \cdots \equiv s_N$ (or the corresponding belief state b) and again acts by selecting a joint action a, each component of which is executed by the corresponding agent. From this perspective, the coordinator is facing a history MDP (or a belief MDP), optimal policies of which are also optimal joint policies of the common-payoff game. Again, this formulation can be executed decentrally by having each player carry around

an identical copy of the coordinator.

Now consider the most general case—that in which agents neither observe the world state nor share information states with their teammates. In this case, it is not immediately clear that the previous approach can be extended. The central coordinator cannot condition its policy on every player's information state $(s_1, \ldots, s_N)$ because, at execution time, each copy of it will only have access to one player's information state. But if coordinator uses less information than s_i to decide a_i, it may be unable to express an optimal joint policy of the common-payoff game.

An alternative, fully general, approach is for the coordinator to specify the entire deterministic joint policy at once, without having received any information. From this perspective, the coordinator faces an MDP in which there is only one state and each deterministic joint policy is an action. Clearly, there is again a bijection between the coordinator's policies and joint policies in the common payoff game in which identified policies and joint policies receive the same expected return.

These two approaches represent two extremes of central coordination. On one extreme, the coordinator decides as little as possible (only the joint action for the current time step) at a time. On the other, it decides everything (the entire deterministic joint policy) all at once. While the former is more easily scalable, it is limited in its applicability. In contrast, the latter is much harder to scale, but fully general. Ideally, there would be an approach that interpolates between the two, prescribing individual actions where it is possible, falling back to joint policy prescriptions where it must, and most typically lying somewhere in between.

As it turns out, such an approach exists. The key insight is that, even when players exist in distinct information states, a central coordinator can still condition on the public knowledge among the players. The logic behind this insight is that, by definition of public knowledge, each player is independently cognizant of public knowledge at execution time.

To gain intuition for how this how this approach works, consider the previous edge cases. In the edge case that all actions and observations are public,

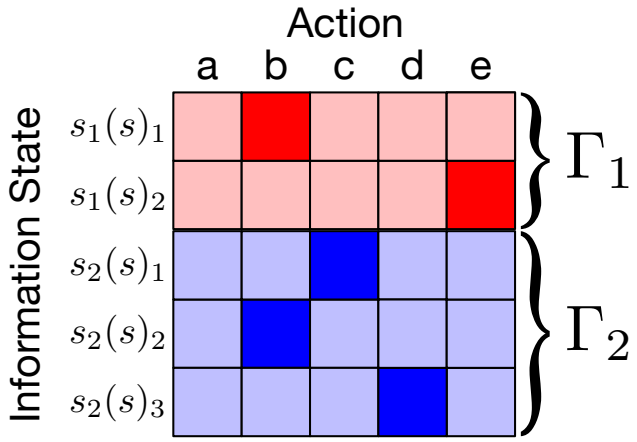

Figure 5.1: An example prescription vector.

the coordinator need only decide the the actions for that particular public state. In the edge case that there is no public knowledge, the coordinator must decide the entire deterministic joint policy at once.

In the more general case in which the players have enough public knowledge to rule out some information states but not others, the coordinator must decide the part of a deterministic joint policy relating to the information states that are deemed possible from public information. The construction in which this coordinator operates, which is called the public POMDP. Given a common-payoff FOG $\langle \mathcal{N}, \mathcal{W}, w^0, \mathcal{A}, \mathcal{T}, \mathcal{R}, \mathcal{O} \rangle$, the entity seeking to maximize return can construct a **public POMDP** $\langle \tilde{\mathcal{W}}, \tilde{w}^0, \tilde{\mathcal{A}}, \tilde{\mathcal{T}}, \tilde{\mathcal{R}}, \tilde{\mathcal{O}} \rangle$ as follows.

- The world states of the public POMDP $\tilde{\mathcal{W}}$ are the histories $\mathcal{H}$ of the common-payoff FOG.
- The initial world state of the public POMDP $\tilde{w}^0$ is the one tuple (w^0).
- The actions of the public POMDP are called **prescription vectors**. A prescription vector is denoted by Γ and has N components. The ith component of a prescription vector Γ_i is the **prescription** for player i. The prescription Γ_i maps s_i to an element of $\mathcal{A}_i(s_i)$ for each $s_i \in \mathcal{S}_i(s_{\text{pub}})$. In words, a prescription instructs a player in the common-payoff FOG how to act as a function of its private information. An example is shown in Figure 5.1. There are two players, with five actions each. Player one (red) has

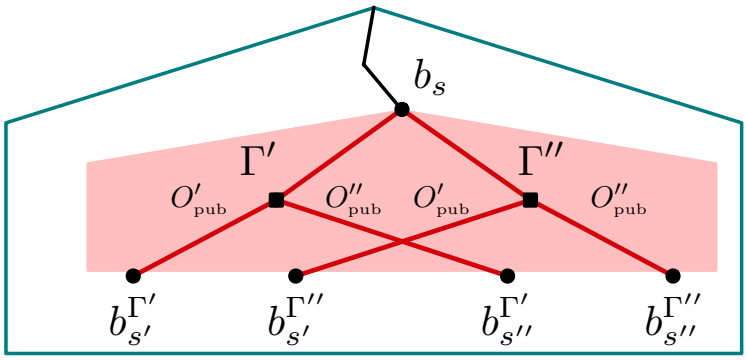

Figure 5.2: A visualization of a decision point in the PuB-MDP.

two possible information states while player two (blue) has three. The prescription vector decides each player's action as a function of its information state, as shown by the darkened squares.

- Given $\tilde{w} \equiv h$ and Γ, the transition distribution $\tilde{\mathcal{T}}(\tilde{w}, \Gamma)$ is induced by $\mathcal{T}(h, a)$, where

$$a \equiv \Gamma(h) \coloneqq (\Gamma_1(s_1(h)), \ldots, \Gamma_N(s_N(h)))\,.$$

- The reward function is such that, for any trajectory, the cumulative reward is equal to that of the players' in the common-payoff game.

- Given $\tilde{w} \equiv h$ and $\tilde{w}' \equiv h'$, the observation $\tilde{\mathcal{O}}(\tilde{w}, \Gamma, \tilde{w}') \equiv \mathcal{O}_{\text{pub}}(h, \Gamma(h), h')$.

In summary, the public POMDP can be informally described as involving a coordinator who only observes public observations and instructs the players how to act as a function of their non-public information. There is a bijection between policies in the public POMDP and joint policies in the common-payoff game for which identified policies and joint policies receive the same expected return. Thus, **to solve a common-payoff game, it is sufficient to solve the corresponding public POMDP.**

As with any with POMDP, the public POMDP can also be considered as a belief MDP, as is exampled in Figure 5.2. The game begins in state b_s, a distribution over the public set $I_{\text{pub}}(s)$. The coordinator is given a choice between two possible prescriptions Γ' and Γ''. Both choices generate observations O'_{pub} and O''_{pub} with positive probabilities (inducing public states s' and

s'' respectively). Accordingly, there are four possible belief states for the next time step. This thesis follows the precedent set by Foerster *et al.*, who refer to this perspective as the **public belief MDP** (PuB-MDP).

As may already be apparent, the public POMDP is not the only possible central coordinator transformation of a common-payoff game—every coarsening of the common knowledge partition of the histories of the game leads to a distinct central coordinator POMDP. For a specific game, there may be a very large number of possible coarsenings. Other than the public POMDP and PuB-MDP, the common POMDP and CB-MDP, which condition on the full set of common knowledge, the temporal POMDP and TB-MDP, which condition on the time step, and the vacuous POMDP and VB-MDP, which condition on nothing, merit names. While the common POMDP is the easiest to solve, it can be difficult to construct. Resultantly, the public POMDP is the most frequently used. The temporal POMDP is more difficult to solve than the public POMDP but is important for historical reasons. The vacuous POMDP is the most difficult to solve and is essentially equivalent to converting the game to normal form.

5.2 Common Knowledge in Games

The significance of common knowledge in imperfect information games has been rediscovered numerous times across different communities.

5.2.1 The Engineering Community

Nayyar *et al.* [47] were the first to formalize the general importance of public knowledge for coordinating teams of agents in common-payoff games. They introduced *the partial history sharing information structure*, a model for decentralized stochastic control. Nayyar *et al.* show that this structure can be converted into a central coordinator POMDP by conditioning on any subset of common knowldge. Nayyar *et al.*'s insights have been applied extensively in control literature [1], [3], [21], [22], [29], [36], [50], [51], [74], [75], [77], [81].

5.2.2 The Dec-POMDP Community

Independently from Nayyar *et al.*, Dibangoye *et al.* [16] showed that a common-payoff game can be converted into the temporal POMDP and the temporal belief MDP (TB-MDP),[1]. Dibangoye *et al.* also introduce feature-based heuristic search value iteration (FB-HSVI), a novel planning algorithm with optimality guarantees. In a large-scale study, Dibangoye *et al.* show that combining the TB-MDP, FB-HSVI, and equivalence relations over information states solves games with hundreds of states and outperforms contemporaneously existing methods.

MacDermed and Isbell build on the TB-MDP approach, showing that it extends to infinite horizon problems in games with a finite number of beliefs. They also show how belief compression can be combined with point-based value iteration [68] to achieve good empirical results.

5.2.3 The Two-Player Zero-Sum Game Community

Public knowledge has also proven to be of significant importance in two-player zero-sum imperfect information games. Independently from the communities studying common-payoff games, the two-player zero-sum game community recently showed that public knowledge can be used to decompose a game into independently analyzable subgames [9], [12], [13], [44], [45]. This idea, known as **subgame decomposition**, is the engine behind recent superhuman performances in no-limit hold'em [10], [11], [45]. The centralized POMDPs in common-payoff games can also be viewed as a form of subgame decomposition, as this thesis shows in Appendix A.

Like the public POMDP, public subgame decomposition requires maintaining a public belief state. Maintaining a public belief state is problematic because the size of the public set (the domain of a public belief state) grows exponentially as a function of time.[2] Even when the public set is small enough

[1]Dibangoye *et al.* call it the occupancy state MDP.

[2]Though in practice, there are many cases in which the public set does not grow exponentially. One example is in games in which all private observations occur at the beginning of the game, such as Texas Hold'em or Stratego. Another example is in games in which old private observations are revealed when new private observations are introduced, such as

that an analytical belief state is tractable, the belief state may still be too unwieldy for practical use as input to a neural network, as is required by DeepStack. DeepStack addresses this problem by losslessly compressing the representation of the public belief state using playerwise factorization [31], [45].

To understand this compression, it is helpful to examine the playerwise factorization of the public belief state

$$P^\pi(h \mid s) = \frac{P_{\mathcal{T}}(h) \prod_{i=1}^{N} P_i^\pi(s_i(h))}{\sum_{h' \in I_{\text{pub}}(s)} P_{\mathcal{T}}(h') \prod_{i=1}^{N} P_i^\pi(s_i(h'))}.$$

Observe that it follows from this factorization that $P^\pi(h \mid s)$ can be losslessly reconstructed from the rules of the game, the public state s, and each player's functional contribution to the public belief state

$$\left[P_1^\pi|_{\mathcal{S}_1(s)}(\cdot \mid s), \ldots, P_N^\pi|_{\mathcal{S}_N(s)}(\cdot \mid s)\right]$$

where $f|_X$ denotes the restriction of f to the domain X. Therefore, the public state and each player's functional contribution to the public belief state is a sufficient statistic for the public belief state. While this sufficient statistic still grows exponentially in the length of the game, it is a much smaller dimensionality than the public belief state and is used by DeepStack to play heads-up no-limit hold'em.

5.2.4 The Deep MARL Community

Unfortunately, even using DeepStack's belief state compression trick, the public POMDP is so massive (there are roughly $|\mathcal{A}_i|^{N \cdot |\mathcal{S}_i|}$ actions at each decision point) that it is infeasible to apply POMDP solution methods [26], [54], [58], [62], [64], [65], [67], [68], out-of-the-box, to common-payoff games of non-trivial size. Nevertheless, the deep MARL community has proposed a number of approaches for scaling common knowledge approaches.

Hanabi. A third is settings in which imperfect recall is sound, as is true in some graphical security games.

BAD

The Bayesian action decoder (BAD) [19] scales an approximate form of the PuB-MDP to two-player Hanabi [6], a large imperfect information common-payoff card game. There are two problems that make the PuB-MDP difficult to scale that BAD overcomes. The first is that even a compressed form of the belief state in Hanabi is too unwieldy to use as input to a network. The second is that there are an astronomical number of prescription vectors in Hanabi. To address the first problem, BAD uses a Hanabi-specific independence assumption on top of belief compression and a belief update correction procedure resembling expectation propagation [41]. To address the second problem, BAD parameterizes its policy as a distribution over prescription vectors that factorizes by information state, as shown below

$$\pi(b, \Gamma) = \prod_{i \in \mathcal{N}} \prod_{s_i \in \mathcal{S}_i(\text{supp}(b))} \tilde{\pi}(b, s_i, \Gamma_i(s_i))$$

where $\text{supp}(b)$ denotes the support of b. Pictorially, one can imagine that this parameterization amounts to keeping an independent distribution for each row of the prescription vector shown in Figure 5.1, where $\tilde{\pi}(b, s_i, \cdot)$ is the distribution corresponding to the row for information state s_i. BAD trains its policy network, parameterized as described above, using policy gradients.

Impressively, when combined with population based training, Foerster *et al.* show that BAD can be scaled to two-player Hanabi, yielding contemporaneous state of the art performance. Yet, BAD's scalability comes at a significant cost. The independence assumption that it makes about belief state and the way in which it parameterizes its policy leave it vulnerable to local optima. This vulnerability is illustrated by BAD's inability solve toy games that are small enough to be easily solved by brute force.

MACKRL

Witt *et al.* propose multi-agent common knowledge reinforcement learning (MACKRL), a hierarchical policy gradient algorithm for decentralized control [79]. MACKRL uses one central coordinator and $\binom{N}{2}$ pairwise coordinators.

The central coordinator only observes common knowledge among all agents and partitions the team of agents into pairs. For each pair that the central coordinator selects, the corresponding pairwise coordinator either selects a joint action (not a prescription vector) based on common knowledge among the pair of agents or instructs each agent to select its own action using its full information state.

MACKRL has vulnerabilities both to local optima requiring trilateral (or more) deviation to improve upon and to local optima in which a pair of players must act according to nontrivial functions of their private information to escape. Nevertheless, MACKRL empirically shows strong performance on challenging problems in StarCraft II unit micromanagement compared to player-by-player approaches.

SPARTA

Lerer *et al.* propose search for partially observing teams of agents (SPARTA), a decision-time policy improvement algorithm for common-payoff games that operates within the PuB-MDP. Decision-time policy improvement algorithms are decision-time planning algorithms that improve upon an externally specified policy, which Lerer *et al.* call the blueprint policy. SPARTA improves its blueprint by doing a one-ply search over the actions of the acting player, assuming that the remainder of the game will be played according to the blueprint.

SPARTA's approach has the advantage of scalability but the drawback of weak theoretical guarantees. It is scalable because, by virtue of not requiring a function approximation to take public belief states as input, it is able to maintain an exact belief state in very large games (e.g. five-player Hanabi). On the other hand, because SPARTA assumes that the rest of the game will be played according to its blueprint, it only guarantees weak asymptotic improvement, meaning that in the limit of the number of search rollouts, SPARTA's policy is only guaranteed to be no worse than its blueprint. In practice, despite this weak guarantee, empirical results on Hanabi suggest that SPARTA tends to significantly outperform its blueprint policy.

Chapter 6

Benchmarking Tabular Value-Based Approaches

Having completed the previous two chapters, each of which introduced a large number of algorithms, a reader might wonder: *How do these algorithms compare?* Unfortunately, because the communities that introduced these algorithms are largely disjointed and because there are multiple popular benchmarks within each community, many of the algorithms that previous sections discussed have never been directly compared. This chapter takes a modest step toward addressing the issue by benchmarking a large number of these algorithms on The Tiny Hanabi Suite.

6.1 The Tiny Hanabi Suite

The Tiny Hanabi Suite is a collection of six toy common-payoff games created by Neil Burch and Nolan Bard. Each game is structured in three steps.

1. A dealer samples two cards from separate piles of `num_cards` cards with uniform probability and gives the first card to player one and the second player two.

2. Player one chooses one of `num_actions` actions. Player two observes player one's action.

3. Player two chooses one of `num_actions` actions.

After player two's action, the game ends and the common payoff is determined as a function of the cards dealt and the actions taken.

The games, which are labelled by the letters A-F, are described in detail in Appendix B. Foerster *et al.* [19] introduced game E of The Tiny Hanabi Suite. The remainder of the games were unreleased prior to this thesis.

6.2 Experiments

This section describes the experiments performed on The Tiny Hanabi Suite. Documented code for all of the experiments in this chapter is available at `https://github.com/ssokota/tiny-hanabi`.

To make the comparison between approaches straightforward, the experiments restrict their attention to approaches that can be implemented with tabular epsilon-greedy Q-learning. This includes the player-by-player approaches: IRL, SAD, BSAD, ASAD, and PSAD, each of which can optionally be combined with additive value decomposition (AVD)—the loss function used by value decomposition networks. This also includes the common knowledge approaches: PuB-MDP, TB-MDP, and VB-MDP.

The experimenter performed two sets of experiments. The first tested the hyperparameter sensitivity of each algorithm. The second tested the performance of algorithms after tuning the hyperparameters. These sets of experiments offer complementary value. Experiments using tuned hyperparameters give a sense of what an algorithm's performance might be in settings in which extensive tuning is possible, whereas hyperparameter sensitivity experiments give a sense of what an algorithm's performance might be in settings in which it is not.

6.2.1 Setup

For each experiment:

- The experimenter normalized the payoffs of each game to $[0, 1]$.
- The experimenter used tabular epsilon-greedy Q-learning.

- The experimenter monitored performed over one million episodes.
- The experimenter linearly decayed the learning rate and exploration rate from their initial values to zero over the course of one million episodes.
- The experimenter ran 30 independent runs. (Graphs display the average of these runs.)

An algorithm is said to solve the game only if its joint policy after one million episodes is optimal for each of the 30 runs.

For the hyperparameter sensitivity experiments, the experimenter ran each algorithm over the same set of nine hyperparameter settings. For the tuned hyperparameter experiments, for each algorithm, the experimenter selected the hyperparameter setting having the highest average final performance among the hyperparameter settings solving the largest number of games for that algorithm. For visual clarity, the tuned hyperparameter experiments exclude the algorithms that performed most poorly in the hyperparameter sensitivity experiments.

6.2.2 Hyperparameter Sensitivity Results

This section discusses the high level takeaways from the hyperparameter sensitivity experiments. See Appendix C for graphs for each experiment.

A summary of the results is presented in Figure 6.1. In the table, per-game-max refers to the number of games for which the best performing hyperparameter setting for that game solved the game; max refers to the number of games for which the best performing hyperparameter setting across the suite solved the game; per-game-median refers the number of games for which the median performing hyperparameter setting for that game solved the game.

As expected, the common knowledge approaches perform best, each solving all six games under both the per-game-max and max criteria. The PuB-MDP also solves six games under the per-game-media criteria. The TB-MDP and VB-MDP do a bit worse under the per-game-median criteria but only due to have having an insufficient number of episodes—not as a result of local optima.

Method	Per-Game-Max	Max	Per-Game-Median
IRL	4	4	3
IRL with AVD	5	5	3
SAD	5	5	5
SAD with AVD	4	3	1
BSAD	4	4	4
BSAD with AVD	3	2	1
ASAD	5	5	5
ASAD with AVD	3	2	1
PSAD	4	4	4
PSAD with AVD	4	4	3
PuB-MDP	6	6	6
TB-MDP	6	6	4
VB-MDP	6	6	5

Figure 6.1: Tiny Hanabi hyperparameter sensitivity results summary.

It is worth keeping in mind that Q-learning is not a particularly sensible choice of algorithm for the TB-MDP or the VB-MDP because they are deterministic. And if an entity were to use Q-learning for the TB-MDP or VB-MDP, it would make sense to do so with high learning and exploration rates, as is evidenced by the fact that the TB-MDP and VB-MDP runs with the highest learning and exploration converge quickly and reliably.

Of the player-by-player methods, SAD and ASAD perform best, reliably solving all games except game C. That ASAD performs comparably to SAD is encouraging, given that, as discussion in Section 4.2.3, ASAD possesses wider applicability. IRL with AVD also performs well, but is more hyperparameter sensitive than SAD and ASAD. BSAD and PSAD perform next best, reliably solving four of the six games. IRL and PSAD with AVD also solve four of the six games but are more hyperparameter sensitive. The other approaches, which combine variants of SAD with AVD, appear to be very hyperparameter sensitive and solve the fewest games.

Overall, it appears the AVD is only helpful in conjunction with IRL and is actually harmful when combined SAD variants. One possible explanation is that it is harmful for an agent, to whom exploration has been signaled, to help set the target value for that exploration.

Regarding the difficulty of the games, it appears that game A is the easiest. Almost all hyperparameter settings for all methods discovered an optimal joint policy within the allotted horizon. On the other hand, it appears that game C is hardest for methods in the player-by-player family. There was not a single hyperparameter setting for any of the player-by-player methods that solved game C on average within one million episodes. Game E appears to be the second hardest for player-by-player methods—of those that solved only four games, none solved game E. Game D is the hardest for the PuB-MDP and is the only game for which not all hyperparameter settings solve the game. Games E and F are hardest for the TB-MDP and VB-MDP, presumably because they are the largest.

6.2.3 Tuned Hyperparameter Results

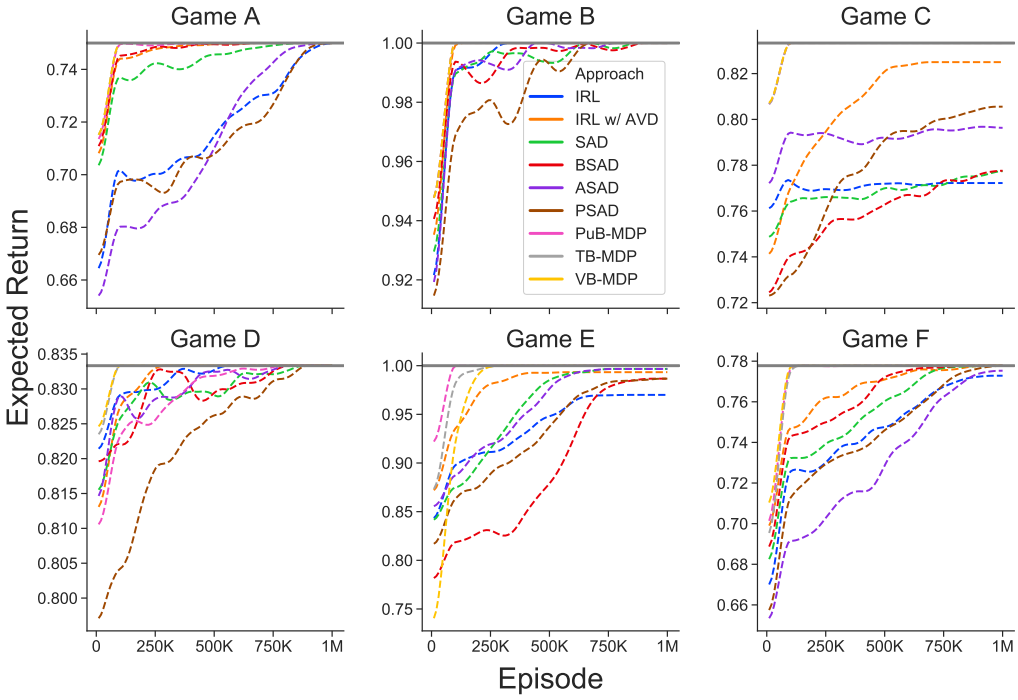

Figure 6.2: Tiny Hanabi tuned hyperparameter results.

Method	Number of Games Solved
IRL	3
IRL with AVD	4
SAD	4
BSAD	4
ASAD	3
PSAD	4
PuB-MDP	6
TB-MDP	6
VB-MDP	6

Figure 6.3: Tiny Hanabi tuned hyperparameter results summary.

Figure 6.2 shows the results of the tuned experiments. Figure 6.3 shows a summary of the results. All methods solved games A, B, and D. The common knowledge methods all also solved games C, E, and F. (In some games their performance curves are almost identical, making them difficult to see.) The interesting differences occur among the player-by-player approaches in games C, E, and F.

Among the player-by-player methods, IRL with AVD arguably performed best, coming close to solving game E on average and achieving the highest average score on game C. PSAD arguably performs second best, performing only marginally worse on game E and achieving the second highest average score among the player-by-player methods on game C. SAD, BSAD, and ASAD are the middle of the pack—SAD and BSAD perform relatively poorly on game C while ASAD fails to solve game F. IRL performs the worst, receiving the lowest average score across all methods in games C, E, and F.

Note that while IRL with AVD, SAD, and ASAD all solved game E for their respective hyperparameter settings in the hyperparameter sensitivity experiments, all failed to solve it in the tuned experiments. Similarly, ASAD failed to solve game F, despite that it solved it with the same configuration in the hyperparameter sensitivity experiments. These differences are a result of the maximization bias.

Chapter 7

Solving Common-Payoff Games

Chapter 4 introduced the player-by-player family of algorithms. Player-by-player algorithms often achieve good performance in practice, but have no guarantee of recovering optimal joint policies. Chapter 5 introduced the common knowledge approach, which uses a central coordinator to determine the joint policy. Algorithms that use a central coordinator have optimality guarantees. Chapter 6 empirically reiterated this difference of guarantees—player-by-player algorithms are sometimes effective in practice, but common knowledge approaches are far more robust at recovering optimal joint policies.

It is tempting to conclude that entities concerned with optimality should prefer common knowledge approaches. But, as has been discussed, common knowledge approaches scale very poorly. This leads to the question: *What algorithm should a entity who is hoping to find an optimal joint policy choose for a game that has thousands or tens of thousands of world states or a game that has hundreds of actions?* Exact common knowledge approaches do not scale to games of this size. And although the deep MARL community has proposed methods to scale approximate common knowledge approaches, these approaches require such extreme compromises that they are unable to find optimal joint policies on games as small as tiny Hanabi [19]. Unsatisfyingly, among existing algorithms, the best answer to this question may be a tabular player-by-player approach.

This chapter argues that it is possible to do better. Toward that end, it proposes cooperative approximate policy iteration (CAPI), a novel instance of

approximate policy iteration operating within the PuB-MDP. Like approaches coming from the deep MARL community, CAPI uses deep learning and makes compromises to gain scalability. But unlike these approaches, CAPI's compromises are modest and it prioritizes correctness over scalability.

7.1 Cooperative Approximate Policy Iteration

The enumeration below describes CAPI's decision making process in language. The same process is described symbolically in Algorithm 3.

1. At each decision point, CAPI takes a public belief state b as argument.

2. CAPI's policy dictates a distribution over prescription vectors $\pi(b)$ as a function of the public belief state. CAPI either tabularly maintains a separate distribution for each public state in the game (i.e. $\pi(b) = \pi[s_{\text{pub}}(b)]$ or produces distribution by passing the public belief state through a policy network (i.e. $\pi(b) = \pi_\theta(b)$).

3. CAPI acquires K prescription vectors from $\pi(b)$. CAPI acquires them either by sampling or by taking the K-most-likely. Sampling is more parallelizable but K-most-likely is more stable.

4. CAPI evaluates each of the K prescription vectors. For each prescription vector $\Gamma^{(k)}$, this involves

   (a) Computing expected reward $r^{(k)}$ for $\Gamma^{(k)}$ given b.
   (b) Computing the next belief state $b^{(k,O_{\text{pub}})}$ for each O_{pub}.
   (c) Estimating the value $v^{(k,O_{\text{pub}})}$ of $b^{(k,O_{\text{pub}})}$ using the value function.
   (d) Computing the probability distribution $p^{(k)}$ over public observations given b and $\Gamma^{(k)}$.

   The assessed value is the expected reward plus the expected estimated value of the next belief state

$$q^{(k)} \leftarrow r^{(k)} + \mathop{\mathbb{E}}_{O_{\text{pub}} \sim p^{(k)}} v^{(k,O_{\text{pub}})}.$$

Algorithm 3 CAPI

procedure ACT(b)
  $[\Gamma^{(k)}] \leftarrow$ prescription_vectors($\pi(b), K$)
  $[r^{(k)}] \leftarrow$ expected_reward($b, [\Gamma^{(k)}]$)
  $[b^{(k,O_{\text{pub}})}] \leftarrow$ next_belief($b, [\Gamma^{(k)}]$)
  $[v^{(k,O_{\text{pub}})}] \leftarrow$ estimate_value($[b^{(k,O_{\text{pub}})}]$)
  $[p^{(k)}] \leftarrow$ public_observation_probabilities($b, [\Gamma^{(k)}]$)
  $[q^{(k)}] \leftarrow [r^{(k)}] +$ public_expectation($[p^{(k)}], [v^{(k,O_{\text{pub}})}]$)
  $k^* \leftarrow$ argmax($[q^{(k)}]$)
  add_to_buffer($b, \Gamma^{(k^*)}, q^{(k^*)}$)
  if explore **then**
    return random($[\Gamma^{(k)}]$)
  return $\Gamma^{(k^*)}$

5. CAPI trains its policy output to more closely resemble the most highly assessed prescription vector $\Gamma^{(k^*)}$ and the value function output to more closely resemble the corresponding value $q^{(k^*)}$.

6. CAPI returns a random prescription vector among those it assessed if it is exploring. Otherwise it returns the most highly assessed prescription vector.

CAPI runs without sampling transitions, meaning that the episode is played out for every transition (i.e. every branch of the public tree) that occurs with positive probability. After each episode, CAPI trains its value function and policy and wipes its memory buffer.

As of yet, two important details have yet to be explained. First: *How can CAPI keep a value function over an exponential number of public belief states?* To do so, CAPI adopts DeepStack's approach (described in 5.2) and combines its lossless public belief compression with a parameterized value network. While this approach does not scale to games having large amounts of private information (e.g. Hanabi), it is sufficient for games of the size considered this chapter.

Second: *How can CAPI maintain a policy over an exponentially large action (prescription vector) space?* To do so, CAPI adopts BAD's approach [19]

and factorizes its policy across information states (as is described in Section 5.2). This parameterization reduces the space required to store the distribution from $|\mathcal{A}_i|^{N \cdot |\mathcal{S}_i|}$ to $|\mathcal{A}_i| \cdot N \cdot |\mathcal{S}_i|$, making it explicitly manageable in the games considered in this chapter. While this parameterization is constraining in that direct optimization by gradient ascent will only guarantee a local optima, decision-time search gives CAPI the opportunity to escape these local optima. CAPI can optionally exploit this parameterization to add structured exploration by adding noise to rows of the $\pi(b)$.

7.2 Experiments

7.2.1 Problem Domains

The experiments employ two games from OpenSpiel [33].

Trade Comm

The first is an emergent communication game called Trade Comm. The game occurs in four steps, as the enumeration below describes.

1. Each player is independently dealt one of `num_items` items from separate piles with uniform probability. Each player observes its own item but not its teammate's.

2. Player 1 makes one of `num_utterances` utterances. Player 2 observes player 1's utterance.

3. Player 2 makes one of `num_utterances` utterances. Player 1 observes player 2's utterance.

4. Both players simultaneously request one of the `num_items` $*$ `num_items` possible trades.

The trade is successful if and only if both player 1 asks to trade its item for player 2's item and player 2 asks to trade its item for player 1's item. Both players receive a reward of one if the trade is successful and zero otherwise. The experiments set `num_items` = `num_utterances` = 15. This means that

there is exactly enough bandwidth for the players to losslessly communicate their items and an optimal joint policy receives an expected reward of one.

This deceivingly easy sounding game nicely illustrates the difficulty of common-payoff games. It is too large to be tackled directly by using a public POMDP transformation and POMDP solution methods (the combination of which have only been applied to games having fewer than 10 actions, whereas Trade Comm has 100s). But at the same time, independent deep reinforcement learning [42], [43] catastrophically fails to learn a good policy.

Abstracted Tiny Bridge

The second game is abstracted tiny bridge, a small common-payoff version of contract bridge retaining some interesting strategic elements. In the game, each player is dealt one of 12 hands as a private observation. The two players then bid to choose the contract. The common payoff is determined by the chosen contract, the hand of the player who chose the contract, and the hand of player who did not chose the contract. The challenge of the game is that the players must use their bids (actions) both to signal their hands and to select the contract, for which there are increasingly limited options as more bids are made. The exact rules are detailed in OpenSpiel [33].

Despite its name, abstracted tiny bridge is much larger than games traditionally considered in engineering or Dec-POMDP literature, having over 50,000 world states. For reference, Mars Rover [2], the largest game considered by many Dec-POMDP papers, has only 256.

7.2.2 Setup

For both games, IRL and SAD with tabular epsilon-greedy Q-learning were the best baselines that the experimenter tried (both outperformed deep variants). The experimenter tuned the baselines for Trade Comm over 25 hyperparameter settings and the baselines for tiny abstracted bridge over 9 hyperparameter settings.

For both games, CAPI used a four layer neural network with 256 hidden units and $K = 10{,}000$ prescription vectors and ϵ-greedy exploration. In Trade

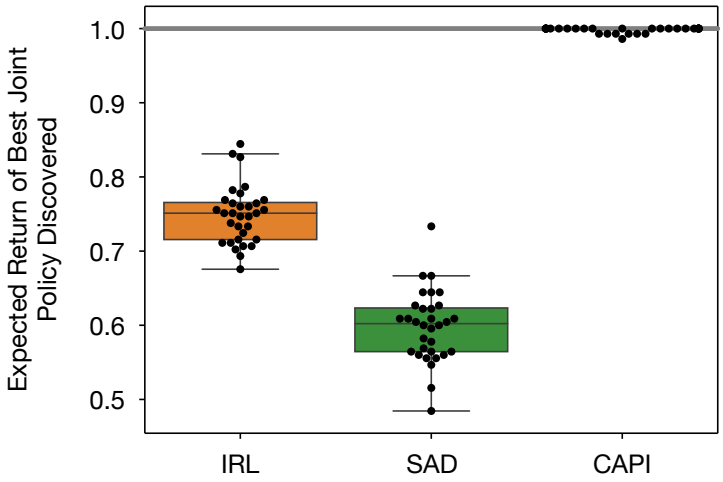

Figure 7.1: Performance comparison on Trade Comm.

Comm, CAPI used a dual headed network (one value head, one policy head); it acquired its prescription vectors by sampling from the policy head. In abstracted tiny bridge, CAPI kept its policy as a function of the public state; deterministically acquired the most-likely prescription vectors from the policy; and during training, added structured exploration by randomly setting a row of $\pi(b)$ to uniform random before taking the K-most-likely.

7.2.3 Results

Trade Comm

Figure 7.1 shows the results for IRL and SAD after 24 hours (between 50 million and 70 million episodes) and results for CAPI after 2000 episodes. To give context to the scores on the graph, the optimal return is one (gray line) and the best joint policies that do not require coordination achieve an expected return of only 1/225.

All three methods perform well relative to the best no-coordination joint policy. That being said, CAPI outperforms both IRL and SAD, solving the game in 25 out of the 32 runs and nearly solving it on the remaining 7. In contrast, neither IRL nor SAD solve Trade Comm on any of their respective 32 runs. This thesis hypothesizes that the especially poor performance of SAD is due to the fact that it is being applied to utterances. Unlike actions in the

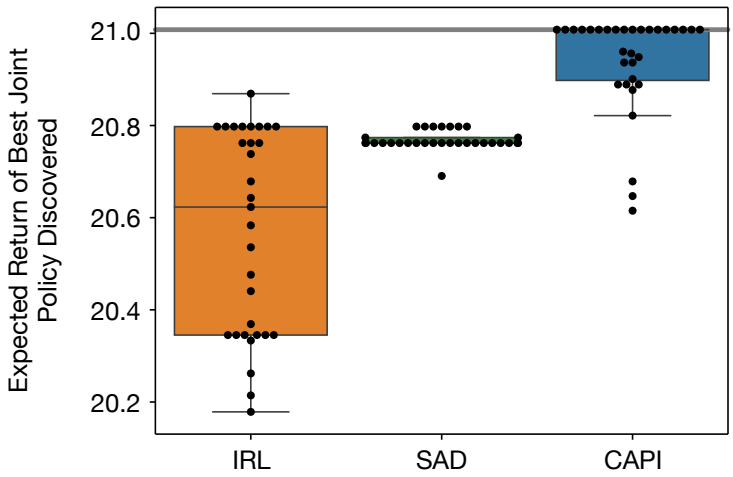

Figure 7.2: Performance comparison on abstracted tiny bridge.

general case, utterances in Trade Comm have no affect on the world state trajectory. Instead, a large part of the benefit to exploring over utterances comes from testing the other player's response. Thus, by announcing its counterfactual greedy utterance to its teammate, the exploring player may be blunting the benefit of its exploration.

Abstracted Tiny Bridge

Figure 7.2 shows the results for IRL and SAD after 10 million episodes and results for CAPI after 100 thousand episodes. To give context to the scores on the graph, the optimal return is the gray line and the best no-coordination joint policy (that is known) achieves an expected return of 20.32.

IRL exhibits the largest variance, occasionally performing worse than the aforementioned no-coordination joint policy (in general there do exist Nash equilibria with lower returns than no-coordination joint policies) but sometimes performing on par with or better than SAD. In contrast, SAD shows consistency, repeatedly finding joint policies achieving similar scores and outperforming IRL on average. CAPI shows the strongest performance, solving the game on 18 of its 32 runs, in contrast with IRL and SAD, which do not solve the game on any of their runs and perform worse on average. However, CAPI's runs do not dominate those of IRL and SAD as they did in Trade Comm. In particular, there are three runs that perform worse than many of

those of IRL and SAD. These outliers likely occur because of the difficulties of training the value network on abstracted tiny bridge, which has a more complex value function landscape than Trade Comm.

Chapter 8

Closing Discussion

This thesis's contributions are:

1. Deriving the common knowledge approach to common-payoff games through the lens of public subgame decomposition.

2. Benchmarking a large number of never-before-compared tabular value-based algorithms on The Tiny Hanabi Suite.

3. Introducing CAPI, a novel approximate policy iteration algorithm for common-payoff games, and showing that it is capable of solving common-payoff games orders of magnitudes larger than existing algorithms.

This closing discussion proposes two interesting research directions for future work that build on these contributions. First: *Are there player-by-player approaches that can robustly solve all six games of The Tiny Hanabi Suite?* This thesis found that, even with extensive tuning, existing player-by-player approaches are unable to do so. Pursuing this direction is very accessible because experiments on The Tiny Hanabi Suite are computationally inexpensive and provide quick feedback. Yet, it also is potentially highly impactful—were such a player-by-player algorithm to exist, it is very plausible that a deep variant of the algorithm would achieve strong performance on large games, such as Hanabi.

Second: *Do there exist algorithms that are able to solve small and medium-sized games, such as CAPI does, but that also achieve good performance performance on large games, such as Hanabi?* The existence of such an algorithm

would be a significant and unifying finding. It would likely require effective mechanisms for learning or approximating public belief states and for working with compressed or implicit representations of prescription vectors. Both of these mechanisms would also be of significance for two-player zero-sum games, where the intractability of public belief states and prescription vectors is the limiting factor for applying decision-time planning approaches to larger problems.

References

[1] M. Afshari and A. Mahajan, "Team Optimal Decentralized State Estimation," *2018 IEEE Conference on Decision and Control (CDC)*, pp. 5044–5050, 2018.

[2] C. Amato and S. Zilberstein, "Achieving Goals in Decentralized POMDPs," in *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems - Volume 1*, ser. AAMAS '09, Budapest, Hungary: International Foundation for Autonomous Agents and Multiagent Systems, 2009, pp. 593–600, ISBN: 978-0-9817381-6-1. [Online]. Available: http://dl.acm.org/citation.cfm?id=1558013.1558095.

[3] J. Arabneydi and A. Mahajan, "Team Optimal Control of Coupled Subsystems with Mean-Field Sharing," in *53rd IEEE Conference on Decision and Control*, Dec. 2014, pp. 1669–1674. DOI: 10.1109/CDC.2014.7039639.

[4] R. Aumann, "Agreeing to Disagree," *The Annals of Statistics*, vol. 4, no. 6, pp. 1236–1239, 1976.

[5] A. P. Badia, B. Piot, S. Kapturowski, P. Sprechmann, A. Vitvitskyi, D. Guo, and C. Blundell, *Agent57: Outperforming the Atari Human Benchmark*, 2020. arXiv: 2003.13350 [cs.LG].

[6] N. Bard, J. N. Foerster, S. Chandar, N. Burch, M. Lanctot, H. F. Song, E. Parisotto, V. Dumoulin, S. Moitra, E. Hughes, I. Dunning, S. Mourad, H. Larochelle, M. G. Bellemare, and M. Bowling, "The Hanabi Challenge: A New Frontier for AI Research," *CoRR*, vol. abs/1902.00506, 2019. arXiv: 1902.00506. [Online]. Available: http://arxiv.org/abs/1902.00506.

[7] D. P. Bertsekas and J. N. Tsitsiklis, *Neuro-Dynamic Programming*, 1st. Athena Scientific, 1996, ISBN: 1886529108.

[8] J. Bhandari and D. Russo, "Global Optimality Guarantees For Policy Gradient Methods," *CoRR*, vol. abs/1906.01786, 2019. arXiv: 1906.01786. [Online]. Available: http://arxiv.org/abs/1906.01786.

[9] N. Brown and T. Sandholm, "Safe and Nested Subgame Solving for Imperfect-Information Games," in *Advances in Neural Information Processing Systems 30*, 2017.

[10] ——, "Superhuman AI for Heads-Up No-Limit Poker: Libratus Beats Top Professionals," *Science*, vol. 359, pp. 418–424, 2018.

[11] ——, "Superhuman AI for Multiplayer Poker," *Science*, vol. 365, no. 6456, pp. 885–890, 2019, ISSN: 0036-8075. DOI: 10.1126/science.aay2400. eprint: https://science.sciencemag.org/content/365/6456/885.full.pdf. [Online]. Available: https://science.sciencemag.org/content/365/6456/885.

[12] N. Brown, T. Sandholm, and B. Amos, "Depth-Limited Solving for Imperfect-Information Games," *ArXiv*, vol. abs/1805.08195, 2018.

[13] N. Burch, M. Johanson, and M. Bowling, *Solving Imperfect Information Games Using Decomposition*, 2014. [Online]. Available: https://www.aaai.org/ocs/index.php/AAAI/AAAI14/paper/view/8407.

[14] H. H. Clark and C. R. Marshall, "Definite Knowledge and Mutual Knowledge," in *Elements of Discourse Understanding*, A. K. Joshi, B. L. Webber, and I. A. Sag, Eds., Cambridge, UK: Cambridge University Press, 1981, pp. 10–63.

[15] W. Dabney, G. Ostrovski, D. Silver, and R. Munos, "Implicit Quantile Networks for Distributional Reinforcement Learning," *CoRR*, vol. abs/1806.06923, 2018. arXiv: 1806.06923. [Online]. Available: http://arxiv.org/abs/1806.06923.

[16] J. S. Dibangoye, C. Amato, O. Buffet, and F. Charpillet, "Optimally Solving Dec-POMDPs As Continuous-state MDPs," *J. Artif. Int. Res.*, vol. 55, no. 1, pp. 443–497, Jan. 2016, ISSN: 1076-9757. [Online]. Available: http://dl.acm.org/citation.cfm?id=3013558.3013572.

[17] J. N. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual Multi-Agent Policy Gradients," in *AAAI*, 2018.

[18] J. Foerster, N. Nardelli, G. Farquhar, T. Afouras, P. H. S. Torr, P. Kohli, and S. Whiteson, "Stabilising Experience Replay for Deep Multi-Agent Reinforcement Learning," in *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ser. ICML'17, Sydney, NSW, Australia: JMLR.org, 2017, pp. 1146–1155.

[19] J. Foerster, F. Song, E. Hughes, N. Burch, I. Dunning, S. Whiteson, M. Botvinick, and M. Bowling, "Bayesian Action Decoder for Deep Multi-Agent Reinforcement Learning," in *Proceedings of the 36th International Conference on Machine Learning*, 2019.

[20] V. François-Lavet, P. Henderson, R. Islam, M. G. Bellemare, and J. Pineau, "An Introduction to Deep Reinforcement Learning," *Foundations and Trends in Machine Learning*, vol. 11, no. 3-4, pp. 219–354, 2018, ISSN: 1935-8245. DOI: 10.1561/2200000071. [Online]. Available: http://dx.doi.org/10.1561/2200000071.

[21] M. Gagrani and A. Nayyar, "Decentralized Minimax Control Problems with Partial History Sharing," in *2017 American Control Conference (ACC)*, May 2017, pp. 3373–3379. DOI: 10.23919/ACC.2017.7963468.

[22] M. Gagrani and A. Nayyar, "Thompson Sampling for some Decentralized Control Problems," *2018 IEEE Conference on Decision and Control (CDC)*, pp. 1053–1058, 2018.

[23] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor," *CoRR*, vol. abs/1801.01290, 2018. arXiv: 1801.01290. [Online]. Available: http://arxiv.org/abs/1801.01290.

[24] J. Y. Halpern and Y. Moses, "Knowledge and Common Knowledge in a Distributed Environment," *CoRR*, vol. cs.DC/0006009, 2000. [Online]. Available: http://arxiv.org/abs/cs.DC/0006009.

[25] H. van Hasselt, A. Guez, and D. Silver, "Deep Reinforcement Learning with Double Q-learning," *CoRR*, vol. abs/1509.06461, 2015. arXiv: 1509.06461. [Online]. Available: http://arxiv.org/abs/1509.06461.

[26] M. J. Hausknecht and P. Stone, "Deep Recurrent Q-Learning for Partially Observable MDPs," in *AAAI Fall Symposia*, 2015.

[27] D. Horgan, J. Quan, D. Budden, G. Barth-Maron, M. Hessel, H. van Hasselt, and D. Silver, "Distributed Prioritized Experience Replay," *CoRR*, vol. abs/1803.00933, 2018. arXiv: 1803.00933. [Online]. Available: http://arxiv.org/abs/1803.00933.

[28] H. Hu and J. N. Foerster, "Simplified Action Decoder for Deep Multi-Agent Reinforcement Learning," in *International Conference on Learning Representations*, 2020.

[29] J. Arabneydi and A. Mahajan, "Reinforcement Learning in Decentralized Stochastic Control Systems with Partial History Sharing," in *2015 American Control Conference (ACC)*, Jul. 2015, pp. 5449–5456. DOI: 10.1109/ACC.2015.7172192.

[30] S. Kapturowski, G. Ostrovski, W. Dabney, J. Quan, and R. Munos, "Recurrent Experience Replay in Distributed Reinforcement Learning," in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=r1lyTjAqYX.

[31] V. Kovarık and V. Lisý, "Value Functions for Depth-Limited Solving in Zero-Sum Imperfect-Information Games," *CoRR*, vol. abs/1906.06412, 2019. arXiv: 1906.06412. [Online]. Available: http://arxiv.org/abs/1906.06412.

[32] V. Kovarık, M. Schmid, N. Burch, M. Bowling, and V. Lisý, "Rethinking Formal Models of Partially Observable Multiagent Decision Making," *CoRR*, vol. abs/1906.11110, 2019. arXiv: 1906.11110. [Online]. Available: http://arxiv.org/abs/1906.11110.

[33] M. Lanctot, E. Lockhart, J.-B. Lespiau, V. Zambaldi, S. Upadhyay, J. Pérolat, S. Srinivasan, F. Timbers, K. Tuyls, S. Omidshafiei, D. Hennes, D. Morrill, P. Muller, T. Ewalds, R. Faulkner, J. Kramár, B. D. Vylder, B. Saeta, J. Bradbury, D. Ding, S. Borgeaud, M. Lai, J. Schrittwieser, T. Anthony, E. Hughes, I. Danihelka, and J. Ryan-Davis, *OpenSpiel: A Framework for Reinforcement Learning in Games*, 2019. arXiv: 1908.09453 [cs.LG].

[34] A. Lazaridou, A. Peysakhovich, and M. Baroni, "Multi-Agent Cooperation and the Emergence of (Natural) Language," *ArXiv*, vol. abs/1612.07182, 2017.

[35] A. Lerer, H. Hu, J. Foerster, and N. Brown, *Improving Policies via Search in Cooperative Partially Observable Games*, 2019. arXiv: 1912.02318 [cs.AI].

[36] L. Lessard and A. Nayyar, "Structural Results and Explicit Solution for Two-Player LQG Systems on a Finite Time Horizon," in *52nd IEEE Conference on Decision and Control*, Dec. 2013, pp. 6542–6549. DOI: 10.1109/CDC.2013.6760924.

[37] D. K. Lewis, *Convention: A Philosophical Study*. Wiley-Blackwell, 1969.

[38] R. Lowe, Y. WU, A. Tamar, J. Harb, O. Pieter Abbeel, and I. Mordatch, "Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., Curran Associates, Inc., 2017, pp. 6379–6390. [Online]. Available: http://papers.nips.cc/paper/7217-multi-agent-actor-critic-for-mixed-cooperative-competitive-environments.pdf.

[39] L. C. MacDermed and C. L. Isbell, "Point Based Value Iteration with Optimal Belief Compression for Dec-POMDPs," in *Advances in Neural Information Processing Systems 26*, 2013.

[40] A. Mahajan and M. Mannan, "Decentralized Stochastic Control," *Annals of Operations Research*, vol. 241, no. 1-2, pp. 109–126, Jun. 2014, ISSN: 1572-9338. DOI: 10.1007/s10479-014-1652-0. [Online]. Available: http://dx.doi.org/10.1007/s10479-014-1652-0.

[41] T. P. Minka, "Expectation Propagation for Approximate Bayesian Inference," in *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence*, ser. UAI'01, Seattle, Washington: Morgan Kaufmann Publishers Inc., 2001, pp. 362–369, ISBN: 1558608001.

[42] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous Methods for Deep Reinforcement Learning," *CoRR*, vol. abs/1602.01783, 2016. arXiv: 1602.01783. [Online]. Available: http://arxiv.org/abs/1602.01783.

[43] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller, "Playing Atari with Deep Reinforcement Learning," *CoRR*, vol. abs/1312.5602, 2013. arXiv: 1312.5602. [Online]. Available: http://arxiv.org/abs/1312.5602.

[44] M. Moravcik, M. Schmid, K. Ha, M. Hladík, and S. J. Gaukrodger, "Refining Subgames in Large Imperfect Information Games," in *AAAI*, 2016.

[45] M. Moravčık, M. Schmid, N. Burch, V. Lisý, D. Morrill, N. Bard, T. Davis, K. Waugh, M. Johanson, and M. Bowling, "DeepStack: Expert-Level Artificial Intelligence in Heads-Up No-Limit Poker," *Science*, vol. 356, no. 6337, pp. 508–513, 2017, ISSN: 0036-8075. DOI: 10.1126/science.aam6960. eprint: https://science.sciencemag.org/content/356/6337/508.full.pdf. [Online]. Available: https://science.sciencemag.org/content/356/6337/508.

[46] R. Nair, M. Tambe, M. Yokoo, D. Pynadath, and S. Marsella, "Taming Decentralized POMDPs: Towards Efficient Policy Computation for Multiagent Settings," in *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, ser. IJCAI'03, Acapulco, Mexico: Morgan Kaufmann Publishers Inc., 2003, pp. 705–711.

[47] A. Nayyar, A. Mahajan, and D. Teneketzis, "Decentralized Stochastic Control with Partial History Sharing: A Common Information Approach," *IEEE Transactions on Automatic Control*, 2013.

[48] F. A. Oliehoek, "Sufficient Plan-time Statistics for Decentralized POMDPs," in *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, ser. IJCAI '13, 2013.

[49] F. A. Oliehoek and C. Amato, *A Concise Introduction to Decentralized POMDPs*. 2016.

[50] Y. Ouyang, S. M. Asghari, and A. Nayyar, "Optimal Local and Remote Controllers with Unreliable Communication: the Infinite Horizon Case," in *2018 Annual American Control Conference (ACC)*, Jun. 2018, pp. 6634–6639. DOI: 10.23919/ACC.2018.8431772.

[51] Y. Ouyang, H. Tavafoghi, and D. Teneketzis, "Dynamic Oligopoly Games with Private Markovian Dynamics," in *2015 54th IEEE Conference on Decision and Control (CDC)*, Dec. 2015, pp. 5851–5858. DOI: 10.1109/CDC.2015.7403139.

[52] E. Pacuit and O. Roy, "Epistemic Foundations of Game Theory," in *The Stanford Encyclopedia of Philosophy*, 2015.

[53] A. Perea, *Epistemic Game Theory*. Cambridge University Press, 2012.

[54] J. Pineau, G. Gordon, and S. Thrun, "Anytime Point-based Approximations for Large POMDPs," *J. Artif. Int. Res.*, vol. 27, no. 1, pp. 335–380, Nov. 2006, ISSN: 1076-9757. [Online]. Available: http://dl.acm.org/citation.cfm?id=1622572.1622582.

[55] R. Powers and Y. Shoham, "New Criteria and a New Algorithm for Learning in Multi-Agent Systems," in *Proceedings of the 17th International Conference on Neural Information Processing Systems*, ser. NIPS'04, Vancouver, British Columbia, Canada: MIT Press, 2004, pp. 1089–1096.

[56] Z. Rabinovich, C. V. Goldman, and J. S. Rosenschein, "The Complexity of Multiagent Systems: the Price of Silence," in *AAMAS*, 2003.

[57] T. Rashid, M. Samvelyan, C. S. de Witt, G. Farquhar, J. N. Foerster, and S. Whiteson, "QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning," in *ICML*, 2018.

[58] S. Ross, J. Pineau, S. Paquet, and B. Chaib-draa, "Online Planning Algorithms for POMDPs," *The Journal of Artificial Intelligence Research*, vol. 32 2, pp. 663–704, 2008.

[59] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized Experience Replay," *CoRR*, vol. abs/1511.05952, 2016.

[60] J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel, T. Lillicrap, and D. Silver, *Mastering Atari, Go, Chess and Shogi by Planning with a Learned Model*, 2019. arXiv: 1911.08265 [cs.LG].

[61] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," *CoRR*, vol. abs/1707.06347, 2017. arXiv: 1707.06347. [Online]. Available: http://arxiv.org/abs/1707.06347.

[62] G. Shani, J. Pineau, and R. Kaplow, "A Survey of Point-Based POMDP Solvers," *Autonomous Agents and Multi-Agent Systems*, vol. 27, no. 1, pp. 1–51, Jul. 2013, ISSN: 1387-2532.

[63] Y. Shoham and K. Leyton-Brown, *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge, UK: Cambridge University Press, 2009, ISBN: 978-0-521-89943-7.

[64] D. Silver and J. Veness, "Monte-Carlo Planning in Large POMDPs," in *Advances in Neural Information Processing Systems 23*, J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, Eds., Curran Associates, Inc., 2010, pp. 2164–2172. [Online]. Available: http://papers.nips.cc/paper/4031-monte-carlo-planning-in-large-pomdps.pdf.

[65] T. Smith and R. G. Simmons, "Point-Based POMDP Algorithms: Improved Analysis and Implementation," *CoRR*, vol. abs/1207.1412, 2012. arXiv: 1207.1412. [Online]. Available: http://arxiv.org/abs/1207.1412.

[66] S. Sokota, R. D'Orazio, K. Javed, H. Haider, and R. Greiner, "Simultaneous Prediction Intervals for Patient-Specific Survival Curves," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, International Joint Conferences on Artificial Intelligence Organization, Jul. 2019, pp. 5975–5981. DOI: 10.24963/ijcai.2019/828. [Online]. Available: https://doi.org/10.24963/ijcai.2019/828.

[67] A. Somani, N. Ye, D. Hsu, and W. S. Lee, "DESPOT: Online POMDP Planning with Regularization," in *Advances in Neural Information Processing Systems 26*.

[68] M. T. J. Spaan and N. Vlassis, "Perseus: Randomized Point-Based Value Iteration for POMDPs," *J. Artif. Int. Res.*, vol. 24, no. 1, pp. 195–220, Aug. 2005, ISSN: 1076-9757. [Online]. Available: http://dl.acm.org/citation.cfm?id=1622519.1622525.

[69] P. Stone, G. A. Kaminka, S. Kraus, and J. S. Rosenschein, "Ad Hoc Autonomous Agent Teams: Collaboration without Pre-Coordination," in *Proceedings of the Twenty-Fourth Conference on Artificial Intelligence*, Jul. 2010.

[70] P. Sunehag, G. Lever, A. Gruslys, W. M. Czarnecki, V. Zambaldi, M. Jaderberg, M. Lanctot, N. Sonnerat, J. Z. Leibo, K. Tuyls, and T. Graepel, "Value-Decomposition Networks For Cooperative Multi-Agent Learning Based On Team Reward," in *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, ser. AAMAS '18, Stockholm, Sweden: International Foundation for Autonomous Agents and Multiagent Systems, 2018, pp. 2085–2087.

[71] M. Šustr, V. Kovařık, and V. Lisý, "Monte Carlo Continual Resolving for Online Strategy Computation in Imperfect Information Games," in *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, 2019.

[72] R. S. Sutton and A. G. Barto, "Reinforcement Learning: An Introduction," in. USA: A Bradford Book, 2018, ch. 8.9, ISBN: 0262039249, 9780262039246.

[73] M. Tan, "Multi-Agent Reinforcement Learning: Independent vs. Cooperative Agents," in *Readings in Agents*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1997, pp. 487–494, ISBN: 1558604952.

[74] H. Tavafoghi, Y. Ouyang, and D. Teneketzis, "On Stochastic Dynamic Games with Delayed Sharing Information Structure," in *2016 IEEE 55th Conference on Decision and Control (CDC)*, Dec. 2016, pp. 7002–7009. DOI: 10.1109/CDC.2016.7799348.

[75] H. Tavafoghi, Y. Ouyang, and D. Teneketzis, "A Sufficient Information Approach to Decentralized Decision Making," *2018 IEEE Conference on Decision and Control (CDC)*, pp. 5069–5076, 2018.

[76] P. Vanderschraaf and G. Sillari, "Common Knowledge," in *The Stanford Encyclopedia of Philosophy*, 2013.

[77] M. M. Vasconcelos and N. C. Martins, "The Structure of Optimal Communication Policies for Remote Estimation Over the Collision Channel with Private and Common Observations," in *2016 IEEE 55th Conference on Decision and Control (CDC)*, Dec. 2016, pp. 320–326. DOI: 10.1109/CDC.2016.7798289.

[78] R. J. Williams, "Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning," *Mach. Learn.*, vol. 8, no. 3–4, pp. 229–256, May 1992, ISSN: 0885-6125. DOI: 10.1007/BF00992696. [Online]. Available: https://doi.org/10.1007/BF00992696.

[79] C. A. S. de Witt, J. N. Foerster, G. Farquhar, P. H. S. Torr, W. Boehmer, and S. Whiteson, "Multi-Agent Common Knowledge Reinforcement Learning," *CoRR*, vol. abs/1810.11702, 2018. arXiv: 1810.11702. [Online]. Available: http://arxiv.org/abs/1810.11702.

[80] M. Zaheer, S. Sokota, E. J. Talvitie, and M. White, "Selective Dyna-style Planning Under Limited Model Capacity," in *ICML*, 2020.

[81] K. Zhang, E. Miehling, and T. Başar, *Online Planning for Decentralized Stochastic Control with Partial History Sharing*, 2019. arXiv: 1908.02357 [cs.LG].

Appendix A

Public Subgame Decomposition

The term *subgame decomposition* [13] refers to the idea of decomposing a game into subgames that can be analyzed independently. In two-player zero-sum imperfect information games, subgame decomposition has led to significant advancements in game playing capabilities [10], [45]. As this appendix details, subgame decomposition is also possible in common-payoff games and leads directly to approximate value iteration results for the PuB-MDP.

A.1 Derivation

Definition A.1.1. *Let b_s be a distribution over $I_{pub}(s)$ for some $s \in \mathcal{S}_{pub}$. The* ***public subgame*** *$G(b_s)$ is defined as follows. With probability $b_s(h)$, the game begins by transitioning to world state $w^1 = w(h)$ with each player i receiving observation $O_i^1 = s_i(h)$.[1] The game proceeds as the original game would from h. When b_s is induced by a policy profile π it can be written $G(s, \pi) \coloneqq G(b_s)$.[2]*

A performance metric is required to analyze public subgame decomposition.

[1] Entrance probabilities for subgames can also be defined in terms of the joint distribution $P^\pi(s_{\text{priv}(1)}, \ldots, s_{\text{priv}(N)}, w \mid s)$, where $s_{\text{priv}(i)}$ denotes the sequence of private observations for player i.

[2] This definition differs slightly from that of two-player, zero-sum literature [13], [31], [32], [71], which defines the subgame as a set of histories, rather than a game. This is reflective of the difference between the public knowledge approaches to adversarial games and common-payoff games. The former is unable to construct the public subgame (as this thesis defines it) because the opponent's policy (and therefore $P^\pi(h \mid s)$) is unknown. As a result, a gadget game is required [9], [44], [71]. In contrast, the public knowledge approach to common-payoff games makes use of the public subgame directly since the coordinator centrally manages all players' policies.

Definition A.1.2. *A joint policy π is **ϵ-optimal** or has **suboptimality bounded by ϵ** if*

$$\max_{\pi'} u(\pi') - u(\pi) < \epsilon.$$

This appendix proves that the suboptimality of public subgame decomposition decomposes linearly by its subgame solution error, value function error, and posterior error. Building on this result, it shows that an approximate value function can be acquired via approximate value iteration and prove that the induced policy is approximately optimal.

Observe first that a policy profile constructed by aggregating good public subgame solutions has low suboptimality.

Lemma A.1.1. *Let π be a policy profile such that decisions at each $s \in \mathcal{S}_{pub}$ are dictated by an ϵ-optimal solution to the public subgame $G(s, \pi)$. (Note that π is well-defined by induction.) Then π is $d \times \epsilon$-optimal policy profile, where d is the depth of the game.*

Taken alone, this result is not very helpful. Solving the initial public subgame with ϵ-precision is the same as solving the entire game with ϵ-precision. The importance of this result is that it shows that full joint policies are not needed to compute full policy profiles for public subgames. As long as the prescription vector for the first step of the public subgame has an ϵ-optimal extension, Lemma A.1.1 can be applied. This is important because it allows for depth-limited solving. After finding a policy profile up to some depth-limit, a value function can be queried to check whether a low-suboptimality extension of this policy profile exists. However, a value function over observations (or histories) does not suffice: its input must have sufficient information to construct a subgame [31], [48].

Definition A.1.3. *The **optimal value function** v maps a public belief state to the expected return of playing optimally in the induced public subgame*

$$v\left(b_s\right) \coloneqq \max_{\pi} u^{G(b_s)}(\pi) = \max_{\pi} \sum_{h \in I_{pub}(s)} b_s(h) u(\pi \mid h).$$

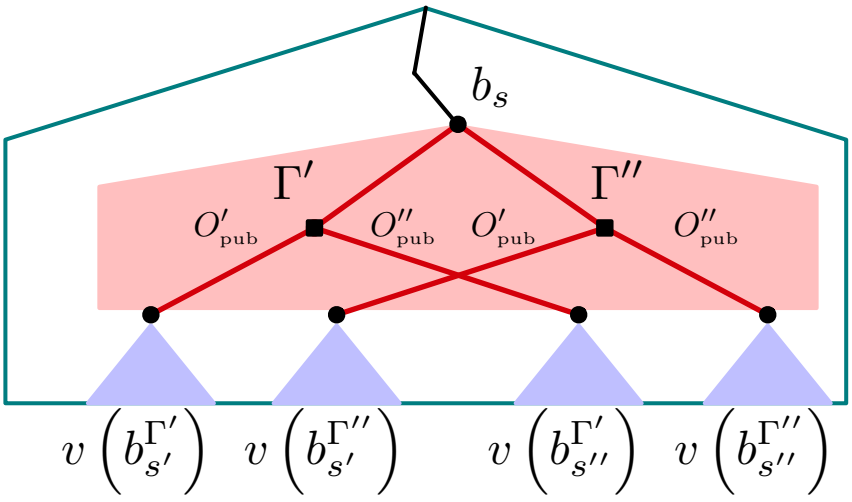

Figure A.1: A visualization of a depth-limited public subgame.

The optimal value function allows us to define a depth-limited public subgame that retains the important properties of the public subgame.

Definition A.1.4. *The **depth-limited public subgame** $G_T(b_s)$ is defined as follows. Each state in the game is a public belief state; the initial state is $b^{\Gamma^0}_{s^0} := b_s$. Each player i acts by choosing a prescription Γ_i. The game transitions from state $b^{\Gamma^{:t}}_{s^t}$ to state $b^{\Gamma^{:t+1}}_{s^{t+1}}$ with observation $O_{pub}(s^{t+1})$ under joint action Γ^{t+1} with probability*

$$P^{\Gamma^{t+1}}(b^{\Gamma^{:t+1}}_{s^{t+1}} \mid b^{\Gamma^{:t}}_{s^t}) := \sum_{h,h'} b^{\Gamma^{:t}}_{s^t}(h)\mathcal{T}(h, \Gamma^{t+1}(h), h')$$

where

$$b^{\Gamma^{:t+1}}_{s^{t+1}}(h') \propto b^{\Gamma^{:t}}_{s^t}(h)\mathcal{T}(h, \Gamma^{t+1}(h), h').$$

The game terminates after T steps with payoff dictated by optimal value of contemporaneous public belief state. Symbolically, the utility of a policy profile π in the depth-limited public subgame is defined by

$$u^{G_T(b_s)}(\pi) := \sum_{s' :\, s \sqsubset s' \in \mathcal{S}^{t+T}_{pub}} P^{\pi}_{b_s}(s') v\left(P^{\pi}_{b_s}|_{I_{pub}(s')}(\cdot \mid s')\right)$$

where

$$P^{\pi}_{b_s}(h) := \sum_{h' \in I_{pub}(s)} b_s(h') P^{\pi}(h \mid h') \;\; \textit{and} \;\; s \in \mathcal{S}^t_{pub}.$$

Depth-limited public subgames have two important properties:

1. They are *fully observable* common-payoff games—they can directly be treated as MDPs and solved centrally, as is described in Figure A.1. The game begins in b_s. The coordinator is given a choice between two possible prescriptions Γ' and Γ''. Both choices generate observations O'_{pub} and O''_{pub} with positive probabilities (inducing public states s' and s'' respectively). Accordingly, there are four possible public belief states at the next time step. The value of public belief state is determined by the value of the optimal policy of the public subgame that would be induced by that public belief state.

2. Policy profiles that are ϵ-optimal have extensions to the public subgame that ϵ-optimal.

These two properties allow the construction of a policy profile by aggregating solutions depth-limited public subgames.

In practice, it is unrealistic to assume access to the optimal function v; it may also be unrealistic to assume the exact public belief state. To make these assumptions more realistic, consider an approximate depth-limited public subgame.

Definition A.1.5. *Define $G_T(\hat{b}_s, \hat{P}, \hat{v})$ as an **approximate depth-limited public subgame**. This approximate depth-limited public subgame follows the same structure as a depth-limited public subgame except that b_s is replaced by $\hat{b}_s$, P is replaced by $\hat{P}$, and v is replaced by $\hat{v}$.*

When $\hat{b}_s$, $\hat{P}$, and $\hat{v}$ are similar to b_s, P and v, the value of a policy profile assessed by $G_T(\hat{b}_s, \hat{P}, \hat{v})$ is similar to the value that would be assessed in $G_T(\hat{b}_s)$.

Lemma A.1.2. *Let $\hat{b}_s$ and $\hat{P}$ be such that for any π the total variation error over the induced public belief states is bounded as*

$$\frac{1}{2} \sum_{s' \in \mathcal{S}_{pub}^{t+T}} |\hat{P}^{\pi}_{\hat{b}_s}(s') - P^{\pi}_{b_s}(s')| < \epsilon_p^s$$

and such that given any $s' \in \mathcal{S}_{pub}^{t+T}$, the total variation error over the induced public belief states is bounded as

$$\frac{1}{2} \sum_{h \in I_{pub}(s')} |\hat{P}^{\pi}_{\hat{b}_s}(h \mid s') - P^{\pi}_{b_s}(h \mid s')| < \epsilon^h_p.$$

Let $\hat{v}$ be such that, given any b, $|\hat{v}(b) - v(b)| < \epsilon_v$. Then $u^{G_T(b_s)}$ and $u^{G_T(\hat{b}_s, \hat{P}, \hat{v})}$ can disagree on the value of a policy profile by no more than

$$\epsilon_v + 2 \max_z |u(z)|(\epsilon^s_p + \epsilon^h_p).$$

Combining Lemma A.1.2 with Lemma A.1.1 and some additional work leads to a result analogous to Theorem 1 of DeepStack—aggregating good solutions to approximate depth-limited public subgames with small errors yields a good solution to the complete game.

Theorem A.1.3. *Consider a finite common-payoff FOG of depth bounded by d. Let π be a policy profile such that decisions at each $s \in \mathcal{S}_{pub}$ are dictated by an ϵ_r-solution to the auxiliary game $G_T(\hat{b}_s, \hat{P}, \hat{v})$ where ϵ_v, ϵ^s_p, and ϵ^h_p bound errors as in Lemma A.1.2 and b_s is the public belief state induced by the trunk policy. Then π is ϵ-optimal, where ϵ is bounded by*

$$d \times [\epsilon_r + 2\epsilon_v + 4 \max_z |u(z)|(\epsilon^s_p + \epsilon^h_p)].$$

Hence, if an oracle offers a value function ϵ_v-close to the optimal value function, an approximately optimal policy profile can be constructed. Alternatively, it can be acquired via approximate value iteration.

Theorem A.1.4. *Let $\hat{v}_0$ be a map from public belief states to reals such that, for public belief states over terminal histories, $\hat{v}_0$ agrees with v. Let $\hat{v}_i$ share its domain and codomain with $\hat{v}_0$ and map each public belief state to the value that would be assessed by approximate depth-limited planning with errors ϵ^s_p, ϵ^h_p, and ϵ_r defined as as in Lemma A.1.2, using the value function $\hat{v}_{i-1}$. Then for $i > d$, where d is the depth of the game, the distance between $\hat{v}_i$ and v in the infinity norm is bounded by*

$$d \times [\epsilon_r + 2 \max_z |u(z)|(\epsilon^s_p + \epsilon^h_p)].$$

Plugging Theorem A.1.4 into Theorem A.1.3 shows the structure of the suboptimality of the policy profile induced by a value function acquired from approximate value iteration.

Theorem A.1.5. *Let π be the policy profile induced by $\hat{v}$, ϵ_r, ϵ_p, where $\hat{v}$ is a value function acquired from ϵ_r, ϵ_p, and the procedure described in Lemma A.1.4 after a sufficient number of iterations. Then π is ϵ-optimal, where ϵ is bounded by*

$$(2d^2 + d) \times \epsilon_r + (4d^2 + 4d) \times \max_z |u(z)|(\epsilon_p^s + \epsilon_p^h).$$

Observe that the case in which each auxiliary game has a depth limit of one corresponds exactly to approximate value iteration in the PuB-MDP.

A.2 Proofs

Lemma A.1.1

Proof. First observe that this result holds trivially when $d = 1$. Now assume the result for $d \leq k$ and consider the case that $d = k + 1$. Then note that, for each public state s consistent with one time step having passed, $G(s, \pi)$ is a common-payoff FOG of depth less than or equal to k. Then, by inductive hypothesis, the aggregation of the subgame strategies over the last k time steps has suboptimality bounded by $k \times \epsilon$ with respect to the public subgame $G(s, \pi)$ for any $s \in \mathcal{S}^1_{\text{pub}}$. Denote by $\tilde{\pi}$ the ϵ-optimal solution to the initial public subgame that dictates π for the first time step. Then observe

$$\begin{aligned}
&\max_{\pi'} u(\pi') - u(\pi) && \text{(A.1)}\\
&= \max_{\pi'} u(\pi') - u(\tilde{\pi}) + u(\tilde{\pi}) - u(\pi) && \text{(A.2)}\\
&\leq \epsilon + u(\tilde{\pi}) - u(\pi) && \text{(A.3)}\\
&= \epsilon + \sum_{s \in \mathcal{S}^1_{\text{pub}}} P^{\pi}(s)[u(\tilde{\pi} \mid s) - u(\pi \mid s)] && \text{(A.4)}\\
&\leq \epsilon + \sum_{s \in \mathcal{S}^1_{\text{pub}}} P^{\pi}(s)[\max_{\pi'} u(\pi' \mid s) - u(\pi \mid s)] && \text{(A.5)}\\
&\leq \epsilon + k \times \epsilon && \text{(A.6)}\\
&= (k + 1) \times \epsilon. && \text{(A.7)}
\end{aligned}$$

Line (A.3) follows from the fact that the initial public subgame is the game itself. Line (A.4) factors by invoking the fact that $\tilde{\pi}$ and π agree over the first time step. Line (A.5) upper bounds the utility of $\tilde{\pi}$ in the public subgame $G(s, \pi)$ by the utility of the optimal policy profile for each $s \in \mathcal{S}^1_{\text{pub}}$. Line (A.6) invokes the inductive hypothesis using the fact that the public subgame is itself a common-payoff FOG. □

Lemma A.2.1. *Take b_s and b'_s such that $\delta(b_s, b'_s) < \epsilon_p$ where δ denotes total variation distance. Then $G(b_s)$ and $G(b'_s)$ can disagree on the value of a policy profile by no more than $2 \max_z |u(z)| \epsilon_p$.*

Proof. Fix π. Then

$$|u^{G(b_s)}(\pi) - u^{G(b'_s)}(\pi)| = |\sum_{h \in I_{\text{pub}}(s)} u(\pi \mid h)[b_s(h) - b'_s(h)]| \tag{A.8}$$

$$\leq 2 \max_z |u(z)| \epsilon_p. \tag{A.9}$$

This follows from the definition of a public subgame and generalized Cauchy-Schwarz. □

Lemma A.2.2. *Let f_1 and f_2 be functions sharing a domain and satisfying the inequality $|f_1(x) - f_2(x)| < \epsilon$. Then it follows that $|[\sup_x f_1(x)] - [\sup_x f_2(x)]| < \epsilon$.*

Proof. This is immediate from the chain of inequalities

$$\begin{aligned}
\sup_x f_2(x) - \epsilon &\leq f_1(\arg\sup_x f_2(x)) \\
&\leq \sup_x f_1(x) \\
&\leq f_2(\arg\sup_x f_1(x)) + \epsilon \\
&\leq \sup_x f_2(x) + \epsilon.
\end{aligned}$$

□

Lemma A.2.3. *The Lipschitz constant of the optimal value function v is bounded by $2 \max_z |u(z)|$ with respect to total variation distance.*

Proof. This is immediate from Lemma A.2.1 and Lemma A.2.2. □

Lemma A.1.2

Proof. For brevity, define $\overline{u} \coloneqq \max_z |u(z)|$ and leave function restrictions and the index set of summations implicit. Then

$$|u^{G_T(b_s P,v)}(\pi) - u^{G_T(\hat{b}_s,\hat{P},\hat{v})}(\pi)| \tag{A.10}$$

$$= |\sum_{s'} P^{\pi}_{b_s}(s') v(P^{\pi}_{b_s}(\cdot \mid s')) - \hat{P}^{\pi}_{\hat{b}_s}(s') \hat{v}(\hat{P}^{\pi}_{\hat{b}_s}(\cdot \mid s'))| \tag{A.11}$$

$$\leq \epsilon_v + |\sum_{s'} P^{\pi}_{b_s}(s') v(P^{\pi}_{b_s}(\cdot \mid s')) - \hat{P}^{\pi}_{\hat{b}_s}(s') v(\hat{P}^{\pi}_{\hat{b}_s}(\cdot \mid s'))| \tag{A.12}$$

$$\leq \epsilon_v + 2\overline{u}\epsilon^h_p + |\sum_{s'} v(P^{\pi}_{b_s}(\cdot \mid s'))[P^{\pi}_{b_s}(s') - \hat{P}^{\pi}_{\hat{b}_s}(s')]| \tag{A.13}$$

$$\leq \epsilon_v + 2\overline{u}(\epsilon^s_p + \epsilon^h_p). \tag{A.14}$$

Line (A.11) holds from definition, (A.12) by assumption on $\hat{v}$, (A.13) from Lemma A.2.3 and assumption on $\hat{P}_{\hat{b}_s}$, and (A.14) from assumption on $\hat{P}_{\hat{b}_s}$. □

Lemma A.2.4. *Let π be ϵ_r-optimal in $G_T(\hat{b}_s, \hat{P}, \hat{v})$, where ϵ_v, ϵ^s_p, and ϵ^h_p upper bound errors as in the Lemma A.1.2. Then π is ϵ-optimal in $G_T(b_s)$ where ϵ is no more than $\epsilon_r + 2\epsilon_v + 4\max_z |u(z)|(\epsilon^s_p + \epsilon^h_p)$.*

Proof.

$$|\max_{\pi'} u^{G_T(b_s,P,v)}(\pi') - u^{G_T(b_s,P,v)}(\pi)| \tag{A.15}$$

$$= |\max_{\pi'} u^{G_T(b_s,P,v)}(\pi') - \max_{\pi'} u^{G_T(\hat{b}_s,\hat{P},\hat{v})}(\pi') \tag{A.16}$$

$$+ \max_{\pi'} u^{G_T(\hat{b}_s,\hat{P},\hat{v})}(\pi') - u^{G_T(\hat{b}_s,\hat{P},\hat{v})}(\pi)| \tag{A.17}$$

$$+ |u^{G_T(\hat{b}_s,\hat{P},\hat{v})}(\pi) - u^{G_T(b_s,P,v)}(\pi)| \tag{A.18}$$

$$\leq \epsilon_r + 2\epsilon_v + 4\max_z |u(z)|(\epsilon^s_p + \epsilon^h_p). \tag{A.19}$$

The first equality holds since the line just adds and subtracts $\max_{\pi'} u^{G_T(b_s,\hat{P},\hat{v})}(\pi')$ and $u^{G_T(\hat{b}_s,\hat{P},\hat{v})}(\pi)$. Line (A.16) is bounded by $\epsilon_v + 2\max_z |u(z)|(\epsilon^s_p + \epsilon^h_p)$ by Lemma A.2.2 and Lemma A.1.2. Line (A.17) is bounded by ϵ_r by assumption. Line (A.18) is bounded by $\epsilon_v + 2\max_z |u(z)|(\epsilon^s_p + \epsilon^h_p)$ by Lemma A.2.2 and Lemma A.1.2. □

Theorem A.1.3.

Proof. From Lemma A.1.1, the suboptimality is bounded by $d \times \epsilon$, where ϵ is an upper bound on the suboptimality of optimal public subgame extensions. From Lemma A.2.4, an ϵ_r-optimal policy profile in $G_T(\hat{b}_s, \hat{P}, \hat{v})$ is at worst an $\epsilon_r + 2\epsilon_v + 4 \max_z |u(z)|(\epsilon_p^s + \epsilon_p^h)$-optimal policy profile in $G_T(b_s^\pi, P, v)$. By construction of $G_T(b_s^\pi, P, v)$, an ϵ-optimal policy profile has an extension to $G(s, \pi)$ bounded by the same suboptimality. The result follows. □

Lemma A.2.5. *Let π be ϵ_r-optimal in $G_T(\hat{b}_s, \hat{P}, \hat{v})$, where ϵ_v, ϵ_p^s, and ϵ_p^h upper bound errors as in the Lemma A.1.2. Then*

$$|\max_{\pi'} u^{G_T(b_s)}(\pi') - u^{G_T(\hat{b}_s, \hat{P}, \hat{v})}(\pi)| < \epsilon_r + 2\epsilon_v + 4 \max_z |u(z)|(\epsilon_p^s + \epsilon_p^h).$$

Proof.

$$\begin{aligned}
&|\max_{\pi'} u^{G_T(b_s, P, v)}(\pi') - u^{G_T(\hat{b}_s, \hat{P}, \hat{v})}(\pi)| && \text{(A.20)}\\
&= |\max_{\pi'} u^{G_T(b_s, P, v)}(\pi') - \max_{\pi'} u^{G_T(\hat{b}_s, \hat{P}, \hat{v})}(\pi') && \text{(A.21)}\\
&\quad + \max_{\pi'} u^{G_T(\hat{b}_s, \hat{P}, \hat{v})}(\pi') - u^{G_T(\hat{b}_s, \hat{P}, \hat{v})}(\pi)| && \text{(A.22)}\\
&\leq \epsilon_r + \epsilon_v + 2 \max_z |u(z)|(\epsilon_p^s + \epsilon_p^h). && \text{(A.23)}
\end{aligned}$$

The first equality holds since the line are just adds and subtracts $\max_{\pi'} u^{G_T(\hat{b}_s, \hat{P}, \hat{v})}(\pi)$. Line (A.21) is bounded by $\epsilon_v + 2 \max_z |u(z)|(\epsilon_p^s + \epsilon_p^h)$ by Lemma A.2.2 and Lemma A.1.2. Line (A.22) is bounded by ϵ_r by assumption. □

Theorem A.1.4

Proof. Define the height of a history to be the maximum number of time steps it could take to terminate from that history. Proceed by induction. By assumption, at public belief states over histories at height zero, $\hat{v}_0$ is exact. Now assume that at height k, $\hat{v}_k$ has error bounded by

$$k \times [\epsilon_r + 2 \max_z |u(z)|(\epsilon_p^s + \epsilon_p^h)].$$

Then observe from Lemma A.2.5 that an assessed value of b_s differs from its optimal value by no more than $\epsilon_r + 2 \max_z |u(z)|(\epsilon_p^s + \epsilon_p^h) + \epsilon_v$. Then by

assumption, at height $k+1$, $\hat{v}_{k+1}$ has error bounded by

$$\begin{aligned}&\epsilon_r + 2\max_z |u(z)|(\epsilon_p^s + \epsilon_p^h) + k \times [\epsilon_r + 2\max_z |u(z)|(\epsilon_p^s + \epsilon_p^h)] \\ &= (k+1) \times [\epsilon_r + 2\max_z |u(z)|(\epsilon_p^s + \epsilon_p^h)].\end{aligned}$$

Then by induction, the result holds. □

Theorem A.1.5

Proof. Again abbreviating $\max_z |u(z)|$ by $\overline{u}$, observe

$$\max_{\pi'} u(\pi') - u(\pi) \tag{A.24}$$
$$\leq d\left[\epsilon_r + 4\overline{u}(\epsilon_p^s + \epsilon_p^h) + 2\epsilon_v\right] \tag{A.25}$$
$$\leq d\left[\epsilon_r + 4\overline{u}(\epsilon_p^s + \epsilon_p^h) + 2d[\epsilon_r + 2\overline{u}(\epsilon_p^s + \epsilon_p^h)]\right] \tag{A.26}$$
$$= (2d^2 + d)\epsilon_r + (4d^2 + 4d)\overline{u}(\epsilon_p^s + \epsilon_p^h). \tag{A.27}$$

Line (A.25) follows from Theorem A.1.3. Line (A.26) follows from Theorem A.1.4. The final line is just rearrangement. □

Appendix B

The Tiny Hanabi Suite

Although the tiny Hanabi games are most naturally thought of as temporally-extended, the payoff functions can be succinctly described as payoff tables, as is shown in Figure B.1, Figure B.2, Figure B.3, Figure B.4, Figure B.5, and Figure B.6.

- The uppercase numerals in the leftmost column represent the card dealt to player one.
- The lowercase numerals in the uppermost column represent the card dealt to player two.
- The uppercase letters represent player one's action.
- The lowercase letters represent player two's action.
- The integer corresponding to a particular quadruplet is the payoff to that trajectory.

In games A, B, C, and D, `num_cards`= 2 and `num_actions`= 2. In game E, `num_cards`= 2 and `num_actions`= 3. In game F, `num_cards`= 3 and `num_actions`= 2.

B.1 Game A

Card		i		ii	
	Action	a	b	a	b
1	A	0	1	0	1
	B	0	0	3	2
2	A	3	3	2	0
	B	3	2	3	3

Figure B.1: Game A of The Tiny Hanabi Suite.

B.2 Game B

Card		i		ii	
	Action	a	b	a	b
1	A	1	0	0	1
	B	1	0	0	1
2	A	0	1	1	0
	B	0	0	1	0

Figure B.2: Game B of The Tiny Hanabi Suite.

B.3 Game C

Card		i		ii	
	Action	a	b	a	b
I	A	3	0	2	0
	B	0	3	3	3
II	A	2	2	0	1
	B	3	0	0	2

Figure B.3: Game C of The Tiny Hanabi Suite.

B.4 Game D

Card		i		ii	
	Action	a	b	a	b
I	A	3	0	3	0
	B	1	3	3	0
II	A	3	2	0	1
	B	0	2	0	0

Figure B.4: Game D of The Tiny Hanabi Suite.

B.5 Game E

Card		i			ii		
	Action	a	b	c	a	b	c
I	A	10	0	0	0	0	10
	B	4	8	4	4	8	4
	C	10	0	0	0	0	10
II	A	0	0	10	10	0	0
	B	4	8	4	4	8	4
	C	0	0	0	10	0	0

Figure B.5: Game E of The Tiny Hanabi Suite.

B.6 Game F

Card		i		ii		iii	
	Action	a	b	a	b	a	b
I	A	0	3	0	0	3	1
	B	3	2	0	1	2	1
II	A	0	2	1	2	0	1
	B	0	1	1	2	0	3
III	A	1	3	0	3	3	1
	B	1	2	2	2	3	0

Figure B.6: Game F of The Tiny Hanabi Suite.

Appendix C

Tiny Hanabi Graphs

C.1 IRL

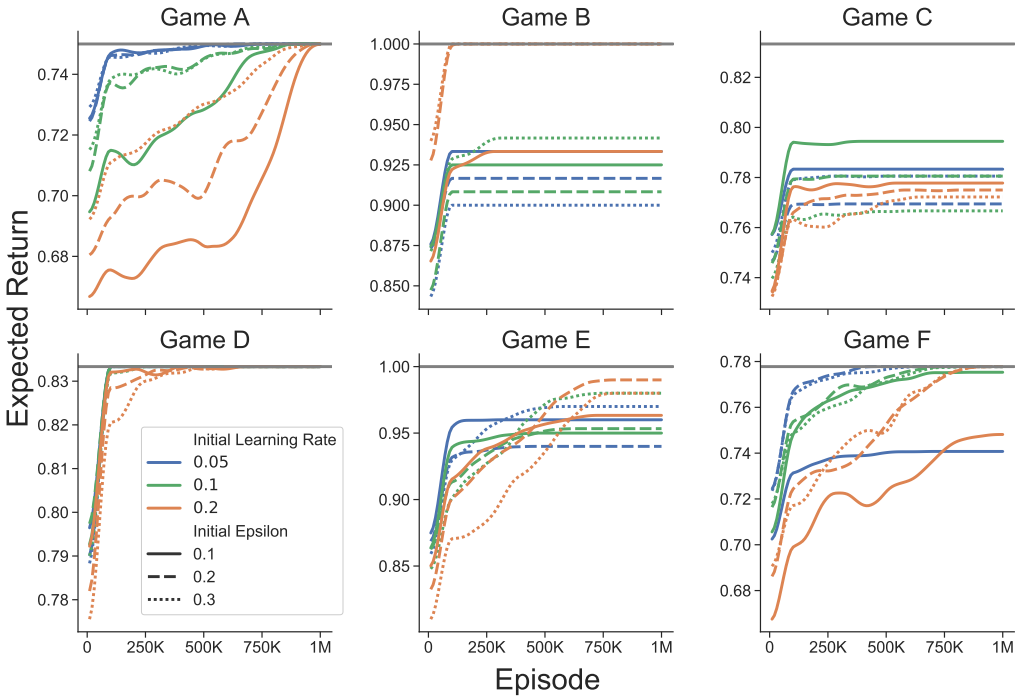

Figure C.1: IRL in tiny Hanabi.

IRL (Figure C.1) solved game A and D with all of the hyperparameter settings. To solve game B it required a large initial learning rate and substantial exploration. To solve game F it required substantial exploration. It failed to solve games C and E.

C.2 IRL with AVD

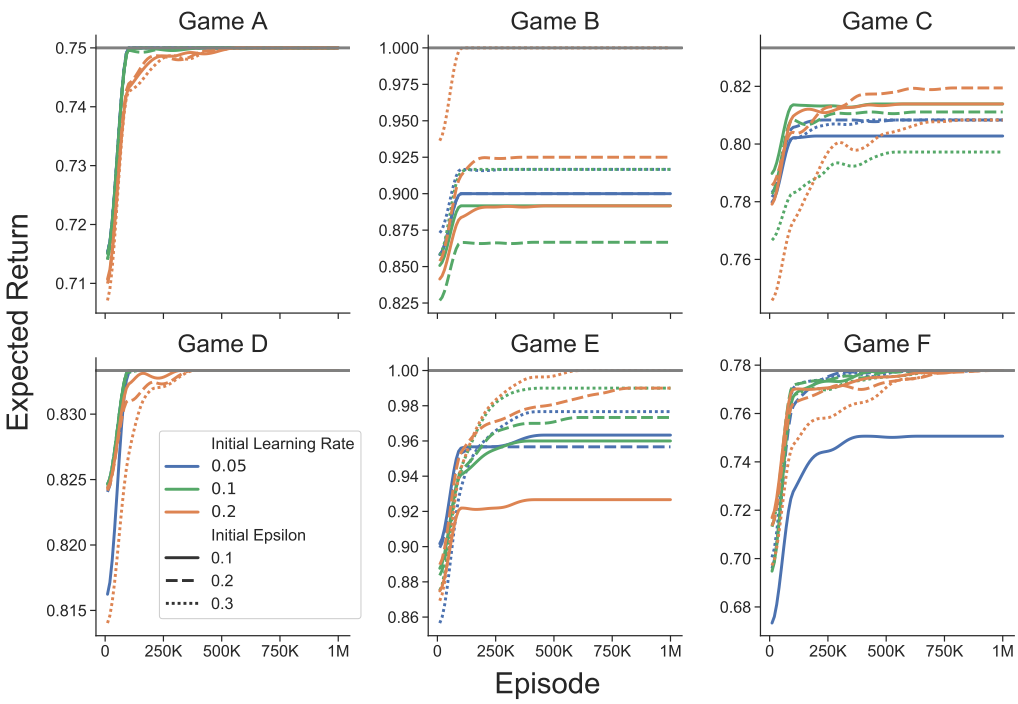

Figure C.2: IRL with AVD in tiny Hanabi.

IRL with AVD (Figure C.2) solved game A and D with all of the hyperparameter settings. It solved game F on all of the hyperparameter settings except the one with the lowest learning rate and the lowest exploration rate. It solved game B and E only with the hyperparameter setting having the highest learning rate and the highest exploration rate. It failed to solve game C. Compared to IRL, IRL with AVD performed better.

C.3 SAD

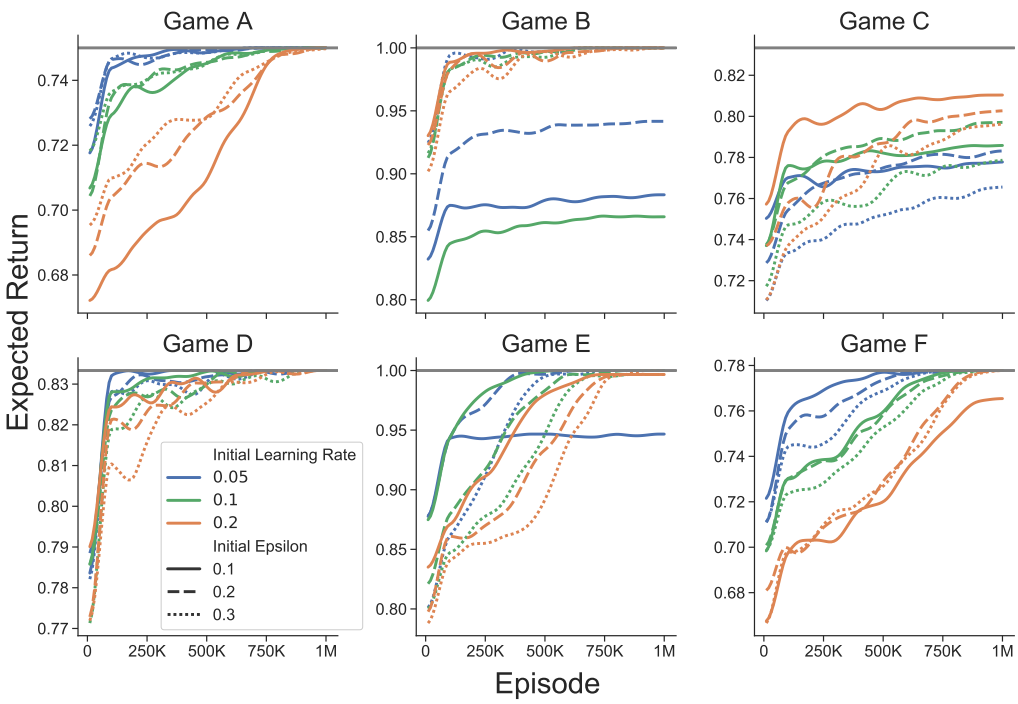

Figure C.3: SAD in tiny Hanabi.

SAD (Figure C.3) solved games A and D with all of the hyperparameter settings. It solved game B for sufficiently high learning rates and exploration rates. It solved game F for every hyperparameter setting except the one having the highest learning rate and lowest exploration rate. It solved game E for sufficiently high exploration rates. It failed to solve game C. Compared to IRL, SAD performed better.

C.4 SAD with AVD

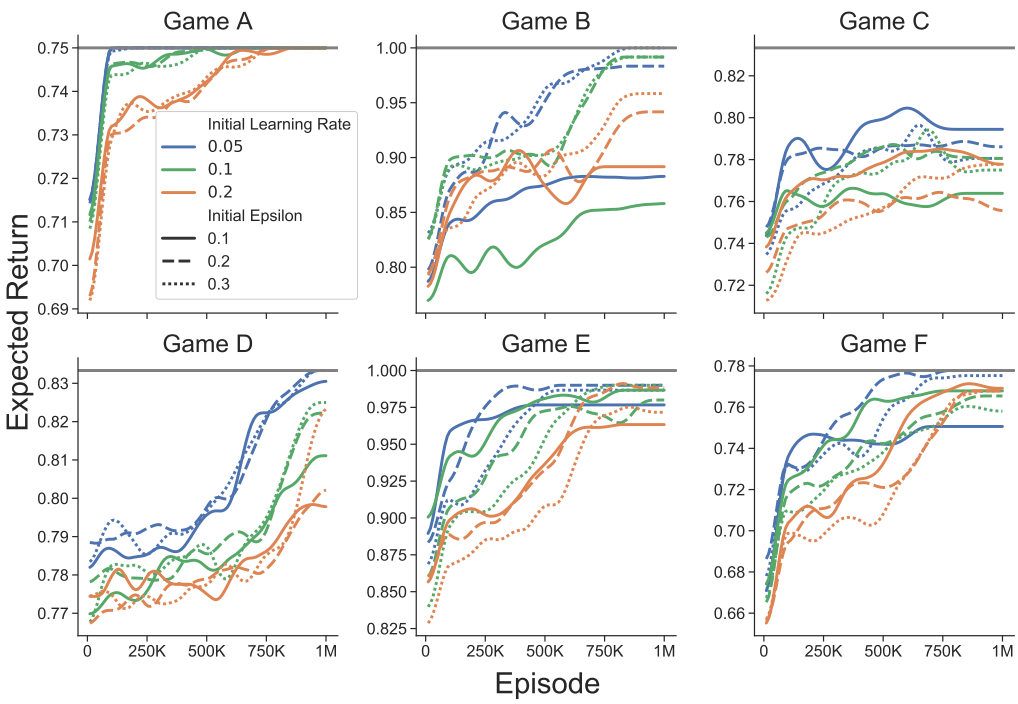

Figure C.4: SAD with AVD in tiny Hanabi.

SAD with AVD (Figure C.4) solved games A with all of the hyperparameter settings. It solved games B and D only with a low learning rate and a large (and also medium in the case of game D) exploration rate. It solved game F only with a low learning rate and a medium exploration rate. It failed to solve games E and C. Compared to SAD, SAD with AVD performed worse.

C.5 BSAD

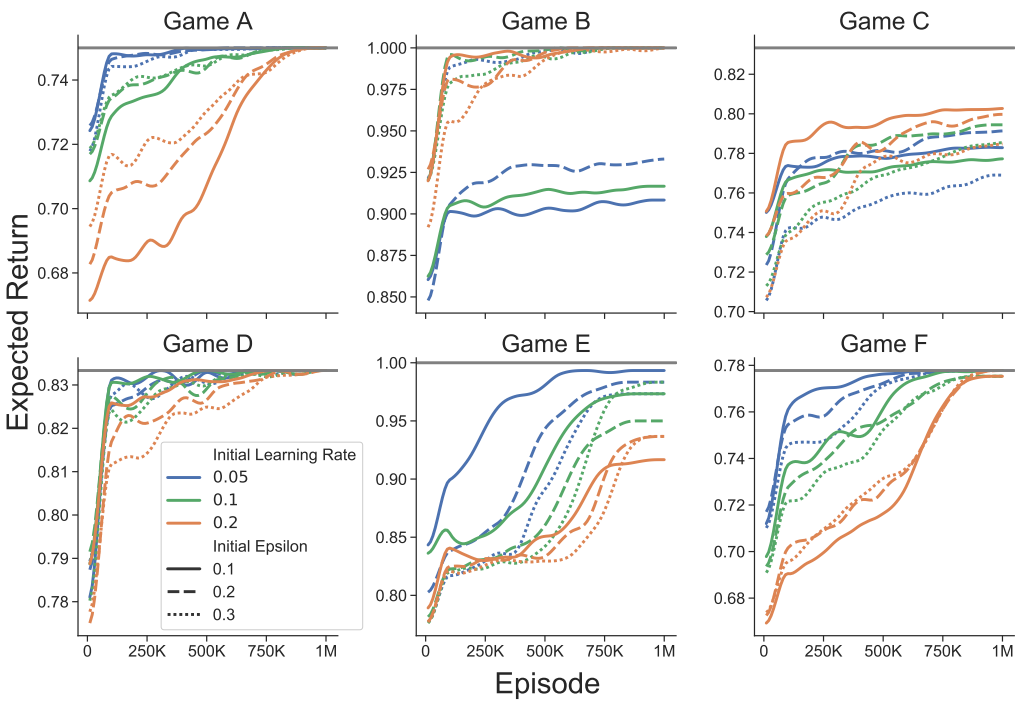

Figure C.5: BSAD in tiny Hanabi.

BSAD (Figure C.5) solved games A and D with all of the hyperparameter settings. It solved games B consistently with a large learning rate and game F consistently with medium and small learning rates. It failed to solve games E and C. Compared to SAD, BSAD performed worse.

C.6 BSAD with AVD

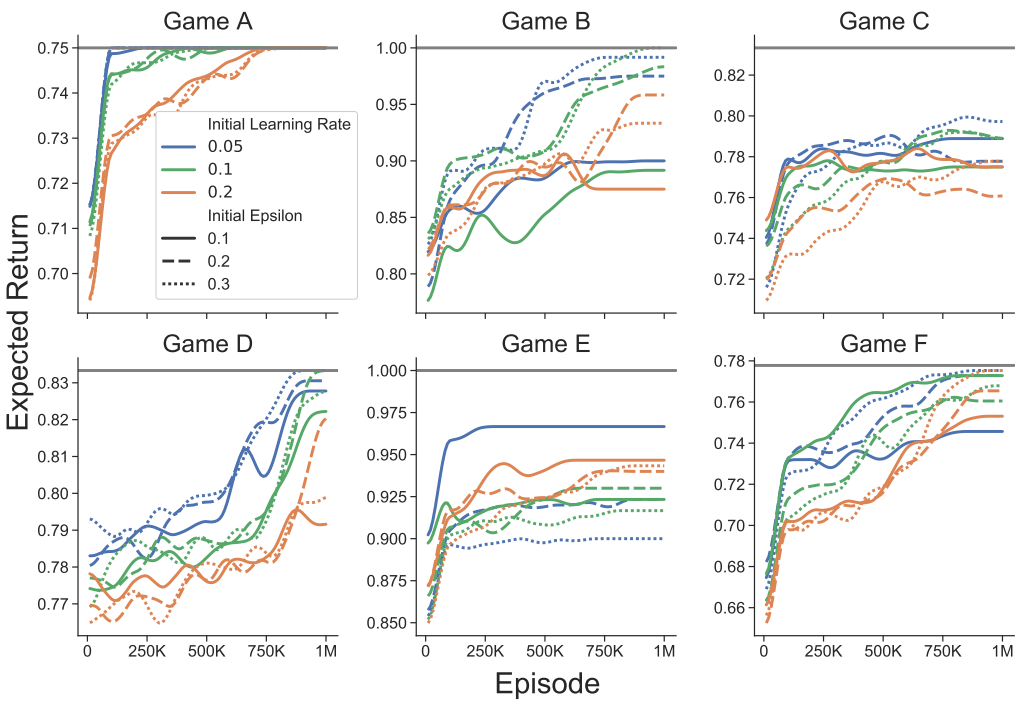

Figure C.6: BSAD with AVD in tiny Hanabi.

BSAD with AVD (Figure C.6) solved game A with all of the hyperparameter settings. It solved game B with a medium learning rate and high exploration. It solved game D with a low learning rate and low exploration and a medium learning rate and medium exploration. It failed to solve games C, E, and F. Compared to BSAD, BSAD with AVD performed worse.

C.7 ASAD

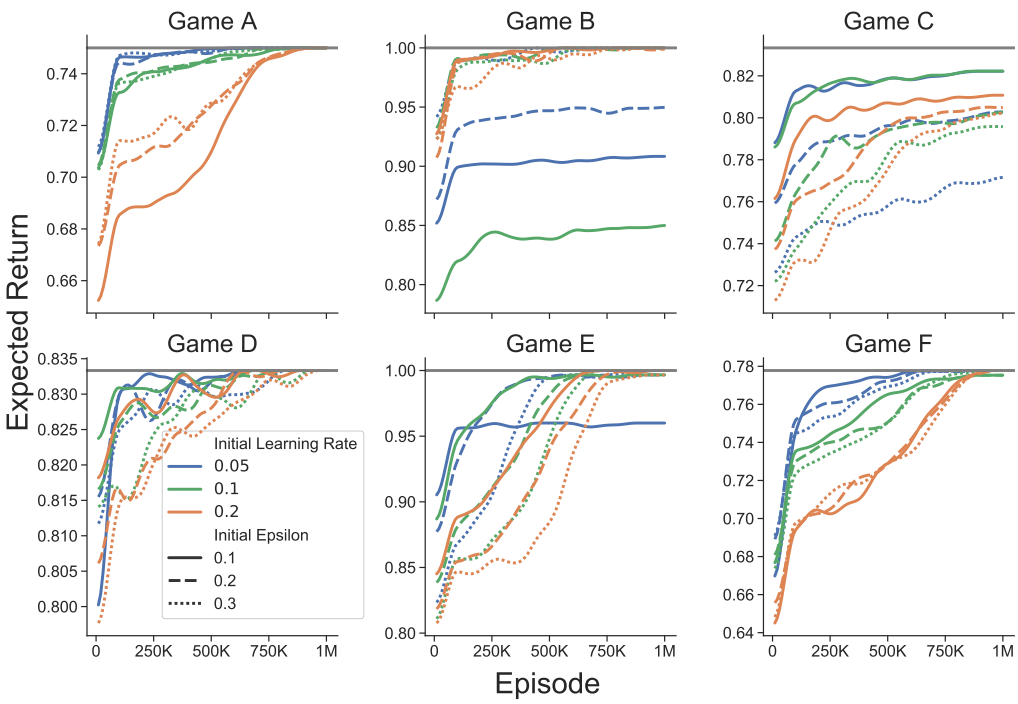

Figure C.7: ASAD in tiny Hanabi.

ASAD (Figure C.7) solved games A and D with all of the hyperparameter settings. It solved games B, E, and F on most of the hyperparameter settings. It failed to solve game C. Compared to SAD, ASAD performed comparably.

C.8 ASAD with AVD

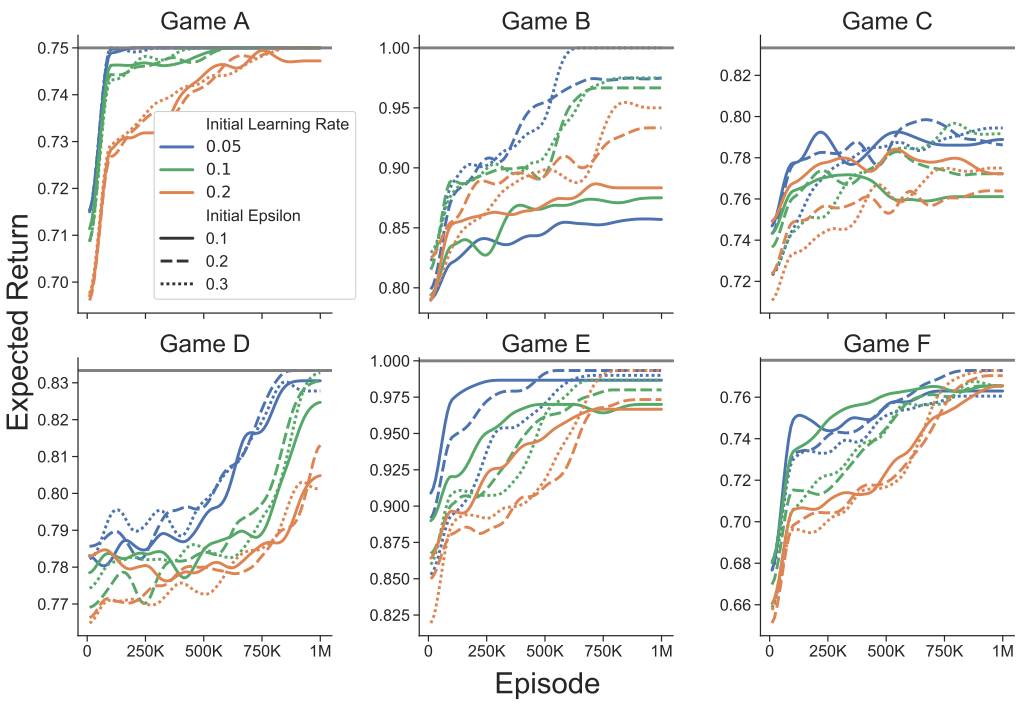

Figure C.8: ASAD with AVD in tiny Hanabi.

ASAD with AVD (Figure C.7) solved game A with all of the hyperparameter settings except the one having the highest learning rate and least exploration. It solved game B only with the hyperparameter setting having the lowest learning rate and highest exploration rate. It solved games A and D on all of the hyperparameter settings. It solved games B, E, and F on most of the hyperparameter settings. It failed to solve game C. Compared to ASAD, ASAD with AVD performed worse.

C.9 PSAD

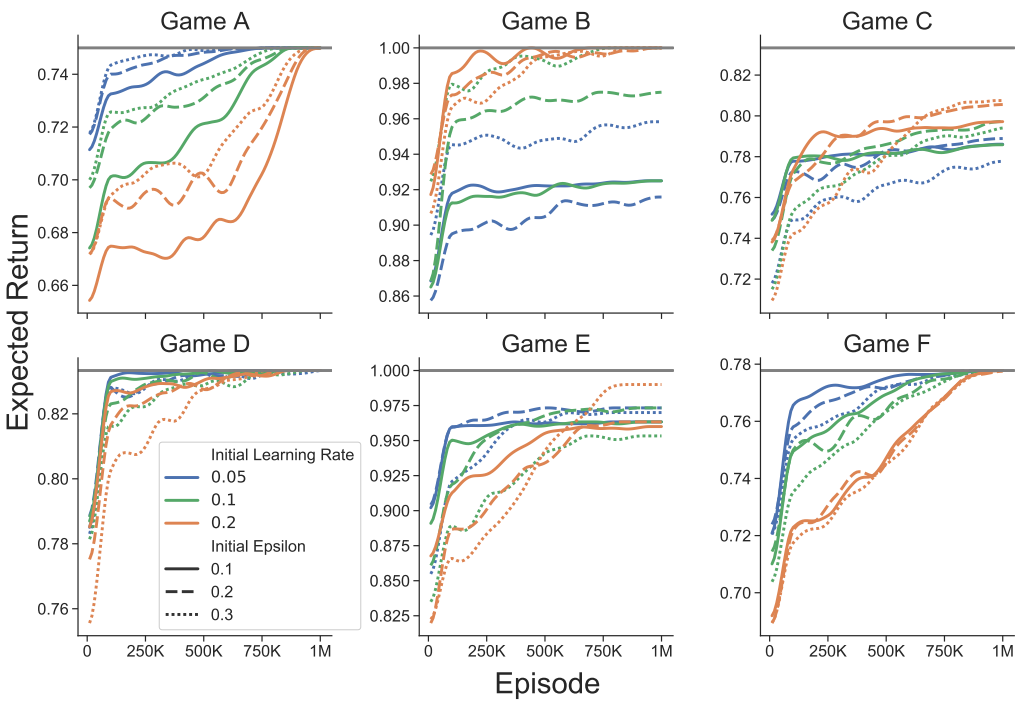

Figure C.9: PSAD in tiny Hanabi.

PSAD (Figure C.9) solved games A, D and F with all of the hyperparameter settings. It solved game B consistently with a large learning rate. And it failed to solve games C and E. Compared to SAD, PSAD performed worse.

C.10 PSAD with AVD

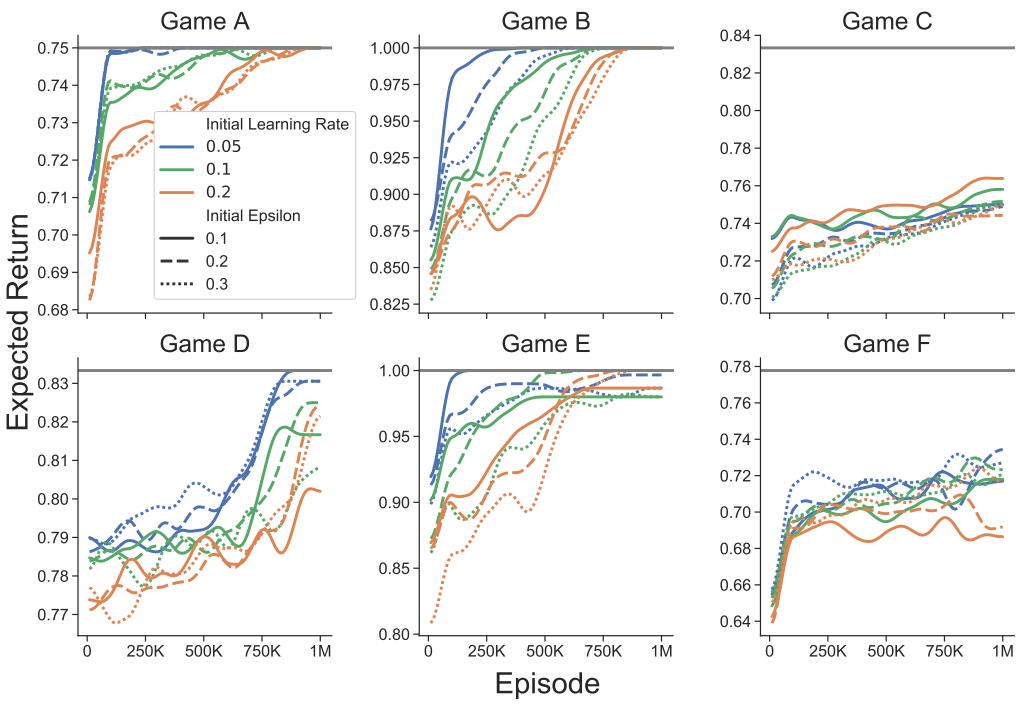

Figure C.10: PSAD with AVD in tiny Hanabi.

PSAD with AVD solved games A and B with all of the hyperparameter settings. It is the only player-by-player method that did so for game B. It only solved game D on the hyperparameter setting with the lowest learning rate and lowest exploration rate. It solved game E on a few hyperparameter settings. It failed to solve games C and F. Compared to PSAD, PSAD with AVD performed worse.

C.11 PuB-MDP

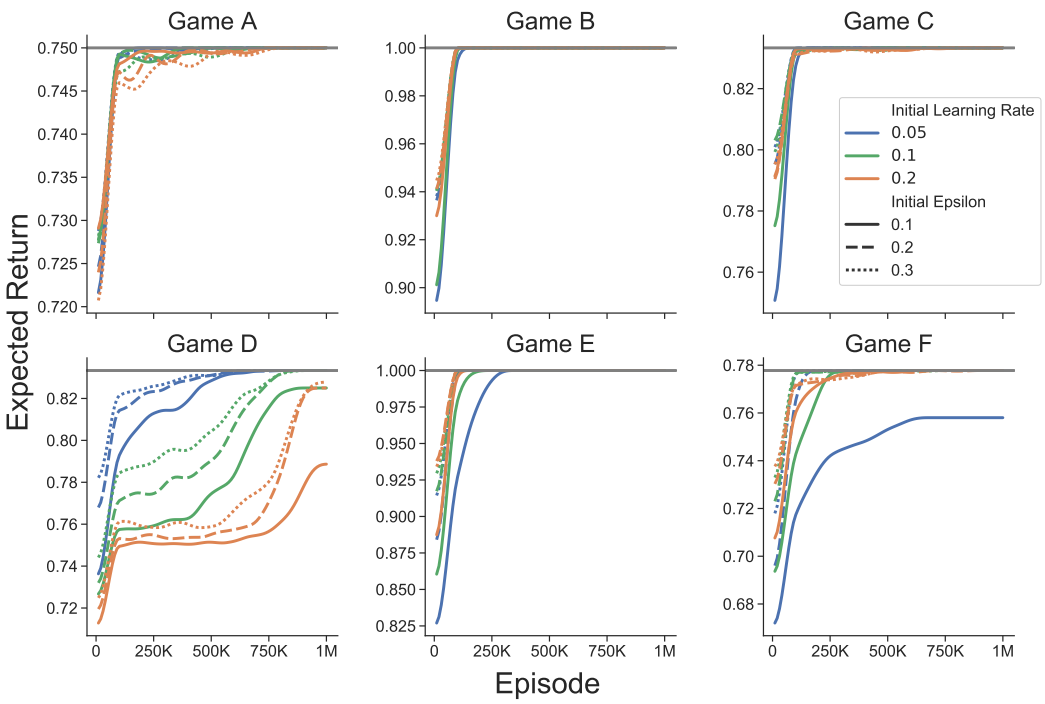

Figure C.11: PuB-MDP for tiny Hanabi.

The PuB-MDP approach solved games A, B, C, and E with all of the hyperparameter settings. On game D, a smaller learning rate appears more favorable to convergence within the allotted horizon. On game F, all hyperparameter settings except the one having the smallest learning rate and the smallest exploration rate converged to the optimal policy.

C.12 TB-MDP

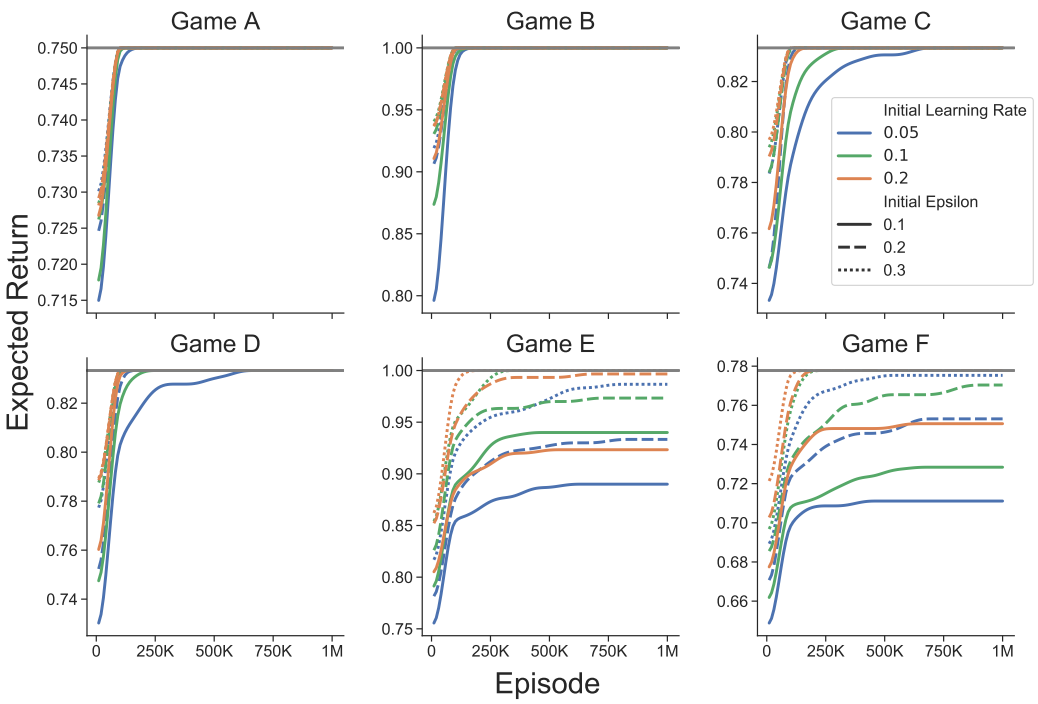

Figure C.12: TB-MDP for tiny Hanabi.

The TB-MDP approach (Figure C.12) solved games A, B, C, and D on all hyperparameter settings. On games D and F, a larger learning rate and exploration rate appear more favorable to convergence.

C.13 VB-MDP

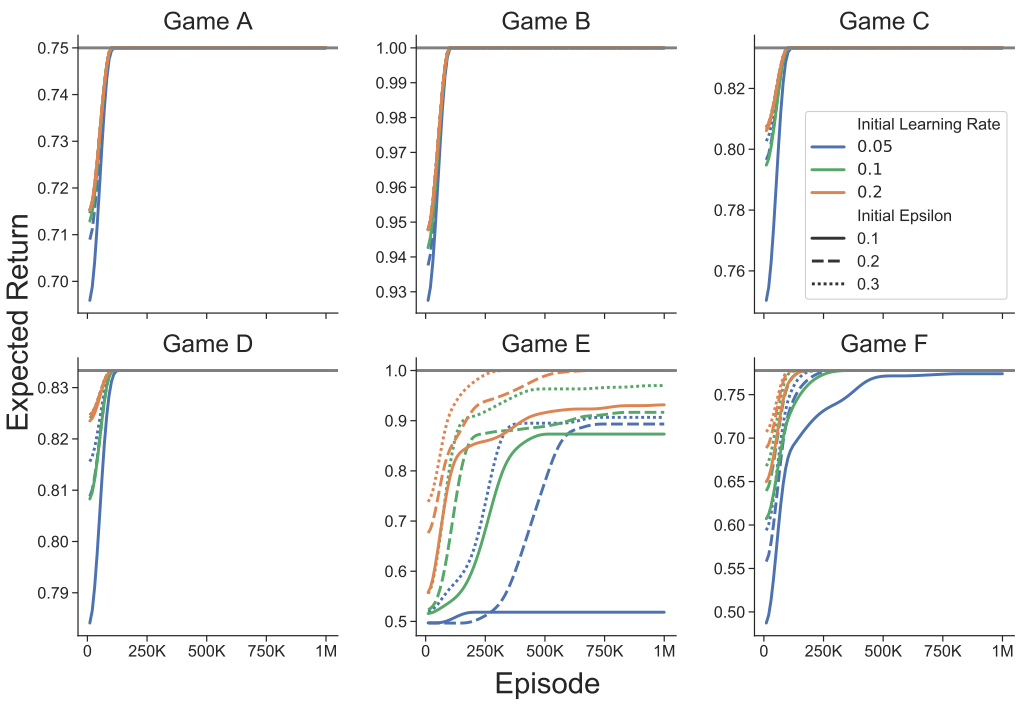

Figure C.13: VB-MDP for tiny Hanabi.

The VB-MDP (Figure C.13) solved games A, B, C, and D on all hyperparameter settings. On games D and F, a larger learning rate and exploration rate appear more favorable to convergence.