Journal of Machine Learning Research 24 (2023) 1-96 Submitted 8/22; Revised 4/23; Published 6/23

Strategic Knowledge Transfer

Max Olan Smith MXSMITH@UMICH.EDU
University of Michigan
Computer Science & Engineering
Ann Arbor, MI 48109-2121, USA

Thomas Anthony TWA@GOOGLE.COM
DeepMind
6 Pancras Square
London N1C 4AG, UK

Michael P. Wellman WELLMAN@UMICH.EDU
University of Michigan
Computer Science & Engineering
Ann Arbor, MI 48109-2121, USA

Editor: Amos Storkey

Abstract

In the course of playing or solving a game, it is common to face a series of changing other-agent strategies. These strategies often share elements: the set of possible policies to play has overlap, and the policies are sampled at the beginning of play by possibly differing distributions. As it faces the series of strategies, therefore, an agent has the opportunity to transfer its learned play against the previously encountered other-agent policies. We tackle two problems: (1) how can learned responses transfer across changing opponent strategies, and (2) how can this transfer be used to reduced the cumulative cost of learning in game solving. The first problem we characterize as the *strategic knowledge transfer problem*. For value-based response policies, we demonstrate that *Q-Mixing* approximately solves this problem by appropriately averaging the component Q-values. Solutions to the first problem can be applied to reduce the computational cost of learning-based game solving algorithms. We offer two algorithms that operate within the Policy-Space Response Oracles (PSRO) framework. *Mixed-Oracles* reduces the per-policy construction cost by transferring responses from previously encountered opponents. *Mixed-Opponents* performs strategic knowledge transfer by combining the previously encountered opponents into a single novel policy. Experimental evaluation of these methods on general-sum grid-world games provide evidence about their advantages and limitations in comparison to standard PSRO.

Keywords: Multiagent Learning, Transfer Learning, Reinforcement Learning, Empirical Game Theoretic Analysis

1. Introduction

Discovering a new policy in a multiagent system, or ***game***, frequently poses a bottleneck to applying learning-based game-solving algorithms. This limitation stems from the policy-learning process, which demands extensive simulation and learning from a multitude of experiences. The issue becomes particularly critical as even the simplest games can possess

large sets of potential policies. Furthermore, discovering each policy is constrained by experiences generated from the previously discovered, and then subsequently employed, policies played by the other agents (hereafter, opponents[1]) in the system. This implies that solving a game may necessitate many iterations of policy learning to uncover all the policies in the game's solution.

In this work, we investigate one avenue to reduce costs in iterative policy learning: through ***strategic knowledge transfer***. Transfer learning exploits commonality between current and previously encountered problems. In our context, *strategic* knowledge transfer exploits the fact that the set of opponent policies persists and expands across iterations. In learning to play against a known population of opponents, we wish to avoid relearning to play against opponents already considered, and minimize experiential variance induced by sampling opponent behavior.

To explain what we mean by ***strategic knowledge***, we first require some terminology for agent behavior. We describe an agent's behavior by its ***policy***, technically a mapping from states to distributions over actions, defined for all states. A ***strategy*** is a distribution over policies that an agent can play. When a strategy's distribution supports multiple policies it is said to be a ***mixed strategy***; if it supports only a single policy it is a ***pure strategy***. Agents draw their policy from this distribution at the beginning of a game instance, and play the sampled policy throughout the duration of the game.

In multiagent settings, the result of playing a policy depends on the environment, combined with strategies of the other agents in the system. Taking the environment as fixed, the ideal behavior for an agent is a function of its strategic setting, that is, the opponent behavior. Let us define a ***best response*** (BR) as a policy maximizing the agent's objective with respect to a particular setting of opponent strategies. More flexibly, we refer to an ***approximate* BR** (ABR) policy, or in shorthand a ***response policy***, as one that is designed for (but may not fully achieve) optimal response. Computation of a response policy is by definition relative to opponent strategies, and thus incorporates what we are referring to as strategic knowledge. At issue in the transfer problem is how to leverage strategic knowledge accrued from learning a response policy in one context, in deriving a response policy for a different but related strategic context.

Strategic knowledge transfer is relevant in many multiagent reasoning frameworks. For concreteness, we motivate our techniques within the context of solving games through Empirical Game Theoretic Analysis (EGTA) (Tuyls et al., 2020; Wellman, 2016). The defining feature of this class of algorithms is simulation-based construction of ***empirical game models*** over restricted strategy sets, intended to approximate a true game of interest. One realization of EGTA is the algorithm Policy-Space Response Oracles (PSRO) (Lanctot et al., 2017). PSRO interleaves empirical game modeling and reinforcement learning (RL) (Sutton and Barto, 2018). In each iteration, PSRO calculates a solution to the current empirical game and employs RL to derive a new response policy corresponding to this solution. The response policy is then added to the empirical game, expanding the player's restricted strategy set. The repeated response calculation in PSRO provides an opportunity to transfer learned play against previously encountered opponents.

1. The methods presented here apply to general-sum games so the other agents are not necessarily adversarial. We nevertheless call them opponents as a convenient shorthand.

We first consider the direct transfer of policies (Section 3) that model probabilities over actions. Observing practical limitations in this approach, we turn our investigation to value-based policies (Section 4). Value-based policies directly model the expected discounted return that an agent should expect for taking an action. We introduce a method, *Q-Mixing* (Smith et al., 2020), that approximately constructs a best-response to a mixed strategy by appropriately averaging the values of each opponent response policy. Critical to the success of Q-Mixing is the ability of the resulting agent to maintain an accurate belief in the current opponent policy. In Section 5, we explore belief maintenance methods and the factors contributing to their success.

We next introduce two methods (Smith et al., 2021) for strategic knowledge transfer to reduce the computational cost of PSRO (Section 6). *Mixed-Oracles* learns separate response policies to each policy in the opponent's mixed strategy; and then, combines the response policies to approximate a response to the full mixed strategy. *Mixed-Opponents* constructs a novel opponent policy that represents an amalgamation of the full mixed strategy, facilitating the transfer of non-optimal behaviors. A response is then calculated against this novel policy, benefiting from the variance reduction resulting from the elimination of sampling over opponent policies. Both of these algorithms employ, but are not limited to, Q-Mixing as a subroutine capable of aggregating strategies into a single policy. When compared to standard PSRO, both methods exhibit a decrease in the cumulative cost necessary to solve a game.

2. Background

2.1 Reinforcement Learning

The reinforcement learning problem can be framed within the context of a Markov decision process (MDP). In an MDP, an agent at time t observes the state of the environment $s^t \in \mathcal{S}$ and takes action $a^t \in \mathcal{A}$ based on this information. The agent's *policy* $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ describes their behavior in every state. Conditional on the chosen action, the environment stochastically transitions into the next state s^{t+1} and generates reward r^{t+1} for the agent. The environment's dynamics are given by the probabilistic transition function $p(s^{t+1}, r^{t+1} \mid s^t, a^t)$. The transition function p can equivalently be expressed as a mapping from (state, action, next state, reward) to a likelihood $p : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \times \mathbb{R} \to [0, 1]$. The sequence of agent and environment interactions ending in a terminal state is called an ***episode***.

The agent's goal is to maximize their cumulative discounted reward called their ***return***. The return an agent following policy π expects to receive in a state is described by their *(state-)value function* V or *action-value function* Q:

$$V(s \mid \pi) = \sum_{a \in \mathcal{A}} \pi(a \mid s) \sum_{s' \in \mathcal{S}} \sum_{r \in \mathbb{R}} p(s', r \mid s, a) \left[r + \gamma V(s' \mid \pi) \right], \tag{1}$$

$$Q(s, a \mid \pi) = \sum_{s' \in \mathcal{S}} \sum_{r \in \mathbb{R}} p(s', r \mid s, a) \left[r + \gamma V(s' \mid \pi) \right]. \tag{2}$$

Q-learning is one such algorithm for learning action values that trains a value predictor to match a temporal-difference (TD) target (Watkins and Dayan, 1992). In the case of finite discrete states and actions, the Q-learning algorithm learns action values through the

following update function:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right], \tag{3}$$

where α is the learning rate, and γ is a discount factor that ensures theoretical convergence and weights near-term rewards higher. Q-learning can be used jointly with function approximation to relax the requirement of finite discrete states. In this version, the action-value function $Q(s, a \mid \theta)$ is represented by a function approximator parameterized by θ. The parameters θ are often left implicit for compactness of notation. Q-learning with a function approximator is optimized using gradient descent with the following objective function:

$$\mathcal{L} = \mathbb{E}_{s,a,r,s'} \left[\frac{1}{2} \| (r + \gamma \max_{a'} Q(s', a' \mid \theta)) - Q(s, a \mid \theta) \|^2 \right]. \tag{4}$$

The term $(r + \gamma \max_{a'} Q(s', a' \mid \theta))$ is referred to as the ***TD target***, or just simply the *target*. This version of Q-learning is often called Deep Q-network (DQN) (Mnih et al., 2015) after the deep neural network architecture that demonstrated its first major success. DQN also marks one of the first major achievements of the field of deep reinforcement learning (DRL or DeepRL), where function approximators from deep learning are studied within the reinforcement learning problem framework. A well-known problem with DQN is that it tends to over-estimate Q-values. Double DQN (van Hasselt et al., 2016) extends DQN to mitigate this by providing a more stable value prediction target. This is realized by having the Q-values in the target be predicted according to a set of slowly changing parameters.

In a *partially observable* environment, agents do not have access to the complete state, but rather their observation $o^t \in \mathcal{O}$ reflects evidence about the state. Due to the uncertainty inherent in multiagent systems, as a result of the opponent, we will consider observations instead of states throughout this work.

2.2 Multiagent Reinforcement Learning

For multiagent settings, we employ subscripts to index and distinguish the agent-specific components of some element. Agent i's policy is thus $\pi_i : \mathcal{O}_i \to \Delta(\mathcal{A}_i)$, and the vector of opponent policies is denoted with the negated index π_{-i}. Boldface is used to designate elements that combine across all agents (e.g., joint-action $\boldsymbol{a}$).

Joint-action value (JAV) (Claus and Boutilier, 1998) functions extend action-value functions to consider an agent's expected return given the actions of all agents:

$$Q_i(o_i, \boldsymbol{a} \mid \boldsymbol{\pi}) = \sum_{o_i' \in \mathcal{O}_i} \sum_{r_i \in \mathbb{R}} p(s_i', r_i \mid o_i, \boldsymbol{a}) \left[r_i + \gamma V_i(o_i' \mid \boldsymbol{\pi}) \right]. \tag{5}$$

JAV learning reduces variance encountered in value learning, because it directly controls for the confounding effects of the other agents in the system. However, this method requires the assumption that opponent actions are visible, which is not always the case. This has led to the study of the *centralized learning and decentralized execution framework*. In this framework, a practitioner trains agents that can exploit additional information during training, so long as, the agent does not rely on the extra information during evaluation (Tan,

1993; Kraemer and Banerjee, 2016). Researchers have investigated several forms of additional training information, such as opponent actions (Claus and Boutilier, 1998), opponent states (Rashid et al., 2018), and coordination signals (Greenwald and Hall, 2003).

A key question then becomes: how to create a policy that can be evaluated without relying on the additional information. This question has been primarily studied when assuming access to opponents' actions as extra training information. A popular approach is to decompose the joint-action value into independent Q-values for each agent (Guestrin et al., 2001; He et al., 2016; Sunehag et al., 2018; Rashid et al., 2018; Mahajan et al., 2019). An alternative is to learn a centralized critic, which can train independent agent policies (Gupta et al., 2017; Lowe et al., 2017; Foerster et al., 2018b). Some have proposed constructing metadata about the agent's current policies as a way to reduce the learning instability present in environments where the opponents' policies are changing (Foerster et al., 2017; Omidshafiei et al., 2017).

Most of the aforementioned techniques train the agents concurrently. In contrast, our context assumes that in each application of learning the opponent plays a distribution over a stationary, or non-learning, set of policies. This removes the need to account for moving targets within a learning operation (Foerster et al., 2018a; Tesauro, 2003), and rather relies on a broader iterative process to address the joint dynamics of multiagent learning.

2.3 Opponent Modeling

Some multiagent RL methods build a predictive model of their opponent and use this to inform their decision making processes. He et al. (2016) introduce DRON, which uses a learned latent action prediction of the opponent as conditioning information to the policy (in a similar nature to the opponent-actions in the joint-action value area). They also present a variant of DRON that uses a Mixture-of-Experts (Jacobs et al., 1991) operation to marginalize over the possible opponent behaviors. More formally, they compute the expected Q-value by marginalizing the opponent's action space:

$$\sum_{a_{-i}} \pi_{-i}(a_{-i} \mid s_i) Q_i(s_i, a_i, a_{-i}). \tag{6}$$

Q-Mixing employs a similar style of operation; however, it marginalizes over the policy-space of the opponent instead of the action-space. Moreover, Q-Mixing depends on independent BR Q-values against each opponent policy, whereas DRON learns a single Q-network. Bard et al. (2013) propose implicitly modeling opponents through the payoffs received from playing against a portfolio of the agent's policies.

Q-Mixing also bears resemblance to Bayesian Policy Reuse (BPR) (Rosman et al., 2016). Both methods leverage a library of precomputed policies; however, BPR performs single-agent task identification to select which policy to play during each episode, whereas, Q-Mixing maintains an online belief and uses this to compute Q-values. The family of PEPPER algorithms have investigated applying BPR to games (Crandall, 2012). This line of work primarily focuses on identification of opponent-policy switching during a single play (episode) (Hernandez-Leal and Kaisers, 2017a,b); whilst, this study assumes that the opponent policy is constant for the full duration of each episode.

The multi-task community has also separately explored approaches that have a similar style of machinery. Progressively growing neural networks is a similar line of work (Rusu

et al., 2016), focused on a stream of new tasks. In our multiagent scenario, the stream of new tasks is learning responses to each opponent policy independently. They then ensemble these networks together for inference; however, this operation shares no commonalities with Q-Mixing. Schwarz et al. (2018) found that this ensembled network growth could be handled with policy distillation.

2.4 Empirical Game Theory

A ***normal-form game*** (NFG) $\Gamma = (\Pi, U, n)$ represents an interaction among n players, where each player i has a strategy set $\Pi_i = \{\pi_i^j\}_{j=0}^{k_i}$ with k_i policies. A player's *strategy* defines their choice of policy to play at the start of a game. An agent is said to be playing a ***pure strategy*** if they deterministically select a policy $\pi_i \in \Pi_i$. Otherwise, they randomize policy selection with a ***mixed strategy*** $\sigma_i \in \Delta(\Pi_i)$. The coefficients defining the sampling process of a mixed strategy is referred to as the strategy's ***mixture***. At the end of the game, each player receives a payoff $U_i : \Pi \to \mathbb{R}$.

A BR to an opponent's strategy σ_{-i} is a policy with maximum return against σ_{-i}, and is not necessarily unique. In framing a BR task, we may attribute to agent i a prior belief σ_{-i}^0 about the opponent strategy. In the course of play the agent may update this belief based on observations of opponent actions. We denote by σ_{-i}^t the agent's belief at time t about the opponent strategy.

An ***empirical game*** is a game model estimated from simulation data. We consider empirical NFGs (ENFGs), which represent a joint payoff function over finite strategy sets. These strategy sets, one for each player, are restricted versions of the full set of policies as defined by the MDP underlying the RL formulation of the system. Entries in the ENFG are point estimates of the corresponding payoffs from the full game, acquired through black-box simulation of the return for each strategy profile. ENFGs are notationally distinguished from the corresponding full game by a tilde-hat mark $\tilde{\Gamma} = (\tilde{\Pi}, \tilde{U}, n)$.

The study of empirical games is formalized under the algorithmic framework ***Empirical Game Theoretic Analysis*** (EGTA) (Wellman, 2006). Walsh et al. (2002) were the first to explicitly estimate an ENFG from agent-based simulation. They analyzed pricing games and auctions with up to twenty players over a small number of hand-crafted heuristic strategies.

An important problem in EGTA is how to select strategies to add to the empirical game. Phelps et al. (2006) were the first to incorporate automated strategy generation in EGTA. To generate a new strategy, they used genetic search given a designer-specified fitness function. Schvartzman and Wellman (2009a) framed the ***strategy exploration problem***, and investigated a variety of heuristic methods. These included variants of computing ABR to Nash equilibrium (NE), along the lines of the double oracle (DO) algorithm (McMahan et al., 2003). Schvartzman and Wellman (2009b) demonstrated the efficacy of reinforcement learning as an ABR oracle for EGTA.

The PSRO framework was designed to provide a flexible template for reasoning about complex games exploiting Deep RL as a powerful BR technique. A key idea of PSRO is to abstract the response target beyond NE, through the concept of a ***Meta-Strategy Solver*** (MSS). MSSs offer a lens that unifies many existing algorithms in multiagent learning and game theory. With an NE solver as its MSS, PSRO corresponds to DO. Different algorithms for computing NE (e.g., based on linear-programming, replicator dynamics (Tay-

lor and Jonker, 1978), regret-minimization (Blum and Mansour, 2007), or regret-matching (Hart and Mas-Colell, 2000) can be considered distinct MSSs, to the extent they may produce results differing in accuracy or equilibrium selection. An MSS that selects a uniform distribution over policies in the current strategy set produces the classic technique of fictitious play (Brown, 1951). Self-play (Silver et al., 2017) results from the MSS that returns a pure strategy containing only the newest policy.

Recent work has investigated and evaluated a variety of new MSSs for strategy evaluation. For example, Wang et al. (2019) propose a weighted combination of NE and uniform profiles. Wright et al. (2019) use NE as primary MSS, but then adjust the BR to improve response to a decay-weighted linear combination of previous solutions. Omidshafiei et al. (2019) employ Markov-Conley Chains to define a solution concept that relates to their policy evaluation measure α-rank, and Muller et al. (2020) investigated its use as an MSS. Marris et al. (2021) proposed MSSs based on various forms of correlated equilibrium. Indeed, any established or new solution concepts is a ready candidate to serve as an MSS for strategy exploration.

2.5 Transfer Learning

Transfer learning is the study of reusing knowledge gained in one context to facilitate learning in a related but different context. Opportunities for transfer may arise in learning tasks, domains, policies, or any other learning target. Within the field of transfer learning, this study addresses two main questions: what type of knowledge is transferred, and how the knowledge is transferred. Both questions are framed within the context of a game, where the knowledge consists of response policies, and the transfer target is a different strategic scenario.

Previous work on how to transfer knowledge has tended to follow one of two main directions (Pan and Yang, 2010; Lampinen and Ganguli, 2019). The *representation transfer* direction considers how to abstract away general characteristics about the task that are likely to apply to later problems. Ammar et al. (2015) present an algorithm where an agent collects a shared general set of knowledge that can be used for each particular task. The second direction directly transfers parameters across tasks; appropriately called *parameter transfer*. Taylor et al. (2005) show how policies can be reused by creating a projection across different tasks' state and action spaces.

In the literature, transferring knowledge about the opponent's strategy is considered intra-agent transfer (Silva and Costa, 2019). The focus of this area is on *adapting to other agents*. One line of work in this area focuses on ad hoc teamwork, where an agent must learn to quickly interact with new teammates (Barrett and Stone, 2015; Bard et al., 2020). The main approach relies on already having a set of policies available, and learning to select which policy will work best with the new team (Barrett and Stone, 2015). Banerjee and Stone (2007) propose learning features that are independent of the game, which can either be qualities general to all games or strategies.

In contrast to prior work, our focus is not on adapting to entirely new opponents, but rather on transferring knowledge about response policies to new configurations or distributions of already encountered opponent policies. In other words, responses to opponent policies are the *source* of information to transfer.

2.6 Multi-Task Learning

Multiagent learning is analogous to multi-task learning. In this analogy, responding to each strategic context is analogous to solving a different task. The opponents' strategies relate to distributions over shared sets of tasks. Similar analogies from strategies to tasks can be made with objectives, goals, contexts, etc. (Kaelbling, 1993; Ruder, 2017).

The multi-task learning community has broadly categorized learnable knowledge into two groups (Snel and Whiteson, 2014). *Task-relevant* knowledge pertains to a specific task (Jong and Stone, 2005; Walsh et al., 2006), while *domain-relevant* knowledge is common across all tasks (Caruana, 1997; Foster and Dayan, 2002; Konidaris and Barto, 2006). Some work has bridged the gap between these settings; for example, knowledge about a task could be a curriculum to apply across tasks (Czarnecki et al., 2018). In task-relevant learning, a leading method is to identify state information that is irrelevant to decision making and abstract it away (Jong and Stone, 2005; Walsh et al., 2006). Our work falls into the same task-relevant category, where we are interested in learning responses to specific opponent policies.

3. Opponent Mixture Transfer Problem

With the background above, we are ready to give a precise definition of a particular problem in strategic knowledge transfer. Namely, suppose we have a set of response policies, each an ABR to a specified opponent policy. Now, we are faced with a randomized opponent, whose behavior is a distribution, or mixture, over these same opponent policies. Intuitively, the response policies should contain information useful for generating an ABR to this mixture.

3.1 Problem Definition

This setup frames what we call the ***opponent mixture transfer problem***. At a high level, this question asks how response policies to all of the opponent's pure strategies, or policies, can be transferred to generate a response to a given opponent mixed strategy.

Problem 1 (Opponent Mixture Transfer Problem)**.** *Given:*

- *the mixed strategy played by the opponent,* $\sigma_{-i} \in \Delta(\Pi_{-i})$*, and*
- *the set of responses to each opponent policy,* $\{\mathrm{BR}(\pi_{-i})\}_{\pi_{-i} \in support(\sigma_{-i})}$*,*

construct a response, $\mathrm{BR}(\sigma_{-i})$*, to the opponent mixture* σ_{-i}*.*

The efficacy of a transfer method is judged relative to the cost of deriving a response policy without strategic transfer—that is, by explicit training against σ_{-i}.

The opponent mixture transfer problem may arise in various contexts. Our particular motivating context is within population-based game-solving algorithms, which compute response policies to play against mixtures drawn from an evolving population of policies. The transfer opportunity arises from the persistence of components of the opponent population across response computations. Thus, BRs calculated for opponent policies present in the population are likely to remain relevant for a significant period as the population evolves. In Section 6, we specifically focus on such a population-based context, namely the PSRO

algorithm, and demonstrate how solutions to the opponent mixture transfer problem can improve the efficiency of game-solving using PSRO.

3.2 Direct Policy Transfer

Ideally, we want a method that solves the opponent mixture transfer problem by a direct operation on the component response policies. Such direct transfer would require no further assumptions about the response policies' implementations or underlying derivations, and would therefore be applicable in the largest possible set of contexts. Effective play would be transferred from the knowledge contained in the response policies, weighted by probabilities that their targets are played by the opponent.

A general solution of this kind is not possible, however, as we demonstrate below through a counterexample. Directly transferring policies requires at minimum that all actions necessary for the mixed-strategy response be among the actions possibly taken in the pure-strategy responses. To understand this limitation, we introduce an example we call the *direct policy transfer game*, depicted in Table 1.

		Player 2	
		π_2^0	π_2^1
Player 1	π_1^0	10, -10	-10, 10
	π_1^1	-10, 10	10, -10
	π_1^2	1, -1	1, -1

Table 1: **Direct policy transfer game.**

Let us take the perspective of Player 1 in this game, and suppose we are tasked with responding to the uniform mixed strategy of Player 2. Following the opponent mixture transfer problem, our inputs are the opponent's mixture $\sigma_2^{\mathrm{U}} \leftarrow (0.5, 0.5)$, and the BRs to each of their supported policies:

$$\mathrm{BR}(\pi_2^0) \to \pi_1^0 \qquad\qquad \mathrm{BR}(\pi_2^1) \to \pi_1^1.$$

From these inputs alone, however we cannot construct the correct $\mathrm{BR}(\sigma_2^{\mathrm{U}})$, which is π_1^2. In this case, the BR to the mixture does not involve the response policies to mixture components. We would need additional information to identify π_1^2 as the optimal response policy.

Despite this counterexample, there may still be games where direct policy transfer can be effective. In some cases relaxations may also be safely made to guarantee correctness. Therefore, for completeness, we provide a possible avenue for directly transferring policies in Appendix A, which works in limited settings.

4. Value Function Transfer

We now turn towards adopting assumptions about the underlying implementation of the response policies. Specifically, we consider *value-based* policies, which are policies derived from an action-value function. As value-based policy derivation is a ubiquitous technique

in reinforcement learning, it lends support to the pursuit of exploiting value representations in strategic transfer.

With respect to the opponent mixture transfer problem, assuming value-based policies means that each response policy $\text{BR}(\pi_{-i})$ has an associated value function $Q_i(\cdot \mid \pi_{-i})$. The value function captures the expected return of *all state-action pairs*, which in principle provides the ability to evaluate any policy, not just the response policies already constructed.

The distinction is clear when we recall the example above (Section 3.2). We could not construct $\text{BR}(\sigma^{\text{U}}_{-i})$, because the solution π^2_1 was not contained in the component response policies. Instead, now with the value-based assumption, we have access to the value function for both response policies. This provides information about the quality of π^2_1 as it pertains to each of the opponent's policies, and can be used to exactly derive that π^2_1 is the best-response to σ^{U}_{-i}.

4.1 Normal-Form Game

To build up to a general solution, we first consider the simplified setting of a normal-form game, which notably has just a single state. The episode plays out by all players participating in one round of simultaneous action selection. All players then receive a single reward as their payoff and the episode concludes. The single-state setting is essentially a problem of bandit learning, where our opponent's strategy will set the reward of each arm for an episode.

Following from the bandit learning problem, intuitively, our expected reward against a mixture of opponents is proportional to the payoff against each opponent weighted by their respective likelihood. This insight motivates our method *Q-Mixing*, where we first train value-based best-responses to each opponent policy. Then we appropriately average the Q-values following the likelihood of playing against each policy within the mixture. To formalize this relationship, we introduce state- and action-values that are conditioned on fixed opponent strategies, called the Strategic Response Value (SRV) (Definition 1) and Strategic Response Q-Value respectively (SRQV) (Definition 2).

Definition 1 (Strategic Response Value)**.** *An agent's* π_i strategic response value *is its expected return given an observation, when playing* π_i *against a specified opponent strategy:*

$$V_{\pi_i}(o^t_i \mid \sigma^t_{-i}) = \mathbb{E}_{\sigma^t_{-i}}\left[\sum_{\boldsymbol{a}} \boldsymbol{\pi}(a_i \mid o^t_i) \sum_{o'_i, r_i} p(o'_i, r_i \mid o^t_i, \boldsymbol{a}) \left[r_i + \gamma \cdot V_{\pi_i}(o'_i \mid \sigma^{t+1}_{-i})\right]\right].$$

Let the optimal SRV *be*

$$V^*_i(o^t_i \mid \sigma^t_{-i}) = \max_{\pi_i} V_{\pi_i}(o^t_i \mid \sigma^t_{-i}).$$

Definition 2 (Strategic Response Q-Value)**.** *An agent's* π_i strategic response Q-value *is its expected return for an action given an observation, when playing* π_i *against a specified opponent strategy:*

$$Q_{\pi_i}(o^t_i, a^t_i \mid \sigma^t_{-i}) = \mathbb{E}_{\sigma^t_{-i}}\left[r^t_i\right] + \gamma \mathbb{E}_{o^{t+1}_i}\left[V_{\pi_i}(o^{t+1}_i \mid \sigma^{t+1}_{-i})\right],$$

where $r_i^t \equiv r_i(o_i^t, a_i^t, a_{-i}^t)$. *Let the* optimal SRQV *be*

$$Q_i^*(o_i^t, a_i^t \mid \sigma_{-i}^t) = \max_{\pi_i} Q_{\pi_i}(o_i^t, a_i^t \mid \sigma_{-i}^t).$$

For generality in the definitions above we notate the opponent's strategy as mixed: σ_{-i}. When the opponent plays a pure strategy we can substitute the notation π_{-i}. For example, the SRQV against an opponent's first policy is $Q(\cdot \mid \pi_{-i}^0)$.

Q-Mixing captures the relationship between the SRQVs against a mixed-strategy opponent and the component SRQVs against each policy in the opponent's mixed strategy. In the single-state setting, weighting the SRQV against each opponent policy by the opponent's distribution supports a BR to that mixture. We define this relationship formally in Theorem 3, and refer to the single-state formulation as *Q-Mixing: Prior*.

Theorem 3 (Single-State Q-Mixing)**.** *Let* $Q_i^*(\cdot \mid \pi_{-i})$, $\pi_{-i} \in \Pi_{-i}$, *denote the optimal strategic response Q-value against opponent policy* π_{-i}. *Then for any opponent mixture* $\sigma_{-i} \in \Delta(\Pi_{-i})$, *the optimal strategic response Q-value is given by*

$$Q_i^*(a_i \mid \sigma_{-i}) = \sum_{\pi_{-i} \in \Pi_{-i}} \sigma_{-i}(\pi_{-i}) \cdot Q_i^*(a_i \mid \pi_{-i}).$$

Proof The definition of Q-value is as follows (Sutton and Barto, 2018):

$$Q_i^*(a_i) = \sum_{r_i} p(r_i \mid a_i) \cdot r_i.$$

In a multiagent system, the dynamics model p suppresses the complexity introduced by the other agents. We can unpack the dynamics model to account for the other agents as follows:

$$p(r_i \mid a_i) = \sum_{\pi_{-i}} \sum_{a_{-i}} \pi_{-i}(a_{-i}) \cdot p(r_i \mid \boldsymbol{a}).$$

We can then unpack the strategic response value as follows:

$$Q_i^*(a_i \mid \pi_{-i}) = \sum_{a_{-i}} \pi_{-i}(a_{-i}) \sum_{r_i} p(r_i \mid \boldsymbol{a}) \cdot r_i.$$

Now we can rearrange the expanded Q-value to explicitly account for the opponent's strategy. The independence assumption enables the following re-writing by letting us treat the opponent's mixed strategy as a constant condition:

$$\begin{aligned} Q_i^*(a_i \mid \sigma_{-i}) &= \sum_{r_i} \sum_{\pi_{-i}} \sigma_{-i}(\pi_{-i}) \sum_{a_{-i}} \pi_{-i}(a_{-i}) \cdot p(r_i \mid \boldsymbol{a}) \cdot r_i \\ &= \sum_{\pi_{-i}} \sigma_{-i}(\pi_{-i}) \sum_{a_{-i}} \pi_{-i}(a_{-i}) \sum_{r_i} p(r_i \mid \boldsymbol{a}) \cdot r_i \\ &= \sum_{\pi_{-i}} \sigma_{-i}(\pi_{-i}) \cdot Q_i^*(a_i \mid \pi_{-i}). \end{aligned}$$

■

		Player 2		
		π_2^{R}	π_2^{P}	π_2^{S}
Player 1	π_1^{R}	$0,\ 0$	$-1,\ 1$	$1,\ -1$
	π_1^{P}	$1,\ -1$	$0,\ 0$	$-1,\ 1$
	π_1^{S}	$-1,\ 1$	$1,\ -1$	$0,\ 0$

Table 2: **Rock-Paper-Scissors.**

4.1.1 Didactic Example

Consider the Rock-Paper-Scissors (RPS) game illustrated in Table 2. Each episode consists of only a single state, then the agents simultaneously submit actions, and receive their rewards. From the perspective of Player 1, our opponent, Player 2, has the choice of three policies: rock π_2^{R}, paper π_2^{P}, and scissors π_2^{S}. Notice, that in this game policies are analogous to primitive actions, and that is not generally the case. For each of the opponent's policies we can determine the optimal SRQV by inspection:

$$Q_1^*(\cdot \mid \pi_2^{\mathrm{R}}) = \begin{bmatrix} 0 \\ 1 \\ -1 \end{bmatrix}, \qquad Q_1^*(\cdot \mid \pi_2^{\mathrm{P}}) = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}, \qquad Q_1^*(\cdot \mid \pi_2^{\mathrm{S}}) = \begin{bmatrix} 1 \\ -1 \\ 0 \end{bmatrix}.$$

The SRQVs are found by first fixing the opponent's policy. In effect, the game is reduced to a 3×1 matrix game with known payoffs for each policy. This reduction removes any stochasticity introduced into payoff estimation that would result in sampling from a distribution of opponent policies. From the smaller matrix game, we need only consider our agent's payoffs (for Player 1 these are the first value in each cell of the game matrix).

Playing deterministically in RPS makes you an easily exploitable player. A stronger opponent may randomly choose between rock and paper yielding the mixed strategy $\sigma_{-i} = (0.5, 0.5, 0.0)$. Using single-state Q-Mixing we can compute the SRQV to said mixed strategy, assuming we know the mixture:

$$\begin{aligned} Q_1^*(\cdot \mid \sigma_{-i}) &= \sum_{\pi_{-i}} \sigma_{-i}(\pi_{-i}) Q_1^*(\cdot \mid \pi_{-i}) \\ &= 0.5 \begin{bmatrix} 0 \\ 1 \\ -1 \end{bmatrix} + 0.5 \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} + 0.0 \begin{bmatrix} 1 \\ -1 \\ 0 \end{bmatrix} \\ &= \begin{bmatrix} -0.5 \\ 0.5 \\ 0 \end{bmatrix}. \end{aligned}$$

Therefore, we know that our optimal policy is π_1^{P}. When we play paper, we have the opportunity to win (unlike playing rock), and have no opportunity to lose (like scissors). The worst we can do is tie, making it our best-response.

4.2 Leveraging Information from the Past

Next, we consider enriching the previous setting to incorporate repeated interaction between the agents across an evolving observation distribution. The joint effect of the agents' actions influences this distribution and affords the opportunity to gather information about their opponent during an episode. Methods in this setting need to (1) leverage information from the past to update its' belief about their opponent, and (2) grapple with the uncertainty about the future. Accordingly, extending Q-Mixing into this setting requires quantification of the agent's current belief about their opponent and their future uncertainty.

We will begin by focusing on the first condition: the method must leverage information from the past to update its' belief about their opponent. When compared to the single-state setting, each agent now has access to a history of their observations $\mathcal{O}_i^{0:t}$. Additionally, we will not presently take into consideration that the agent may gain future evidence about the identity of their opponent's policy (see Section 4.3). In essence, the current setting reflects that of the penultimate state of a game. Where each agent has all of the previous play to consider; however, they know that they are deciding their final action for this episode.

During an episode the actual observations experienced generally depend on the identity of the opponent's policy, which is drawn from their mixed strategy. Let

$$\psi_i : \mathcal{O}_i^{0:t} \to \Delta(\Pi_{-i}) \tag{7}$$

represent the agent's current belief about the opponent's policy using the observations during play as evidence. From this prediction, we propose an approximate version of Q-Mixing that accounts for past information. The approximation works by first predicting the relative likelihood of each opponent policy given the current observation. Then it weights the Q-value-based BRs against each opponent by their relative likelihood.

Figure 1 provides a conceptual illustration of the benefits and limitations of this new prediction-based Q-Mixing. At any given timestep t during the episode, the information available to an agent about the opponents may be insufficient to perfectly identify their policy. Nevertheless, the agent maintains a belief σ_{-i}^t of the identity of their opponent's policy. The yellow area above a timestep represents the uncertainty reduction from an updated prediction of the opponent σ_{-i}^t compared to the baseline prediction of the prior σ_{-i}^0. Crucially, this definition of Q-Mixing does not consider updating the opponent distribution from new information in the future (blue area in Figure 1).

Let the previously defined ψ be the ***opponent policy classifier*** (OPC), which predicts the identity of the opponent policy. We then augment Q-Mixing to weight the importance of each BR as follows:[2]

$$Q_{\pi_i}(o_i, a_i \mid \sigma_{-i}) = \sum_{\pi_{-i}} \psi_i(\pi_{-i} \mid o_i, \sigma_{-i}) Q_{\pi_i}(o_i, a_i \mid \pi_{-i}). \tag{8}$$

We refer to this quantity as ***Q-Mixing***, or ***Q-Mixing: X***, where X describes ψ. The version of Q-Mixing introduced in the single-state setting will be therefore referred to as *Q-Mixing: Prior*, where the prior belief σ_{-i}^0 is the known mixed strategy of the opponent. By

2. The notation on the observation that explicitly demarcates the inclusion of the full history is suppressed. This is for both ease of reading and because the requisite amount of history required will vary per game.

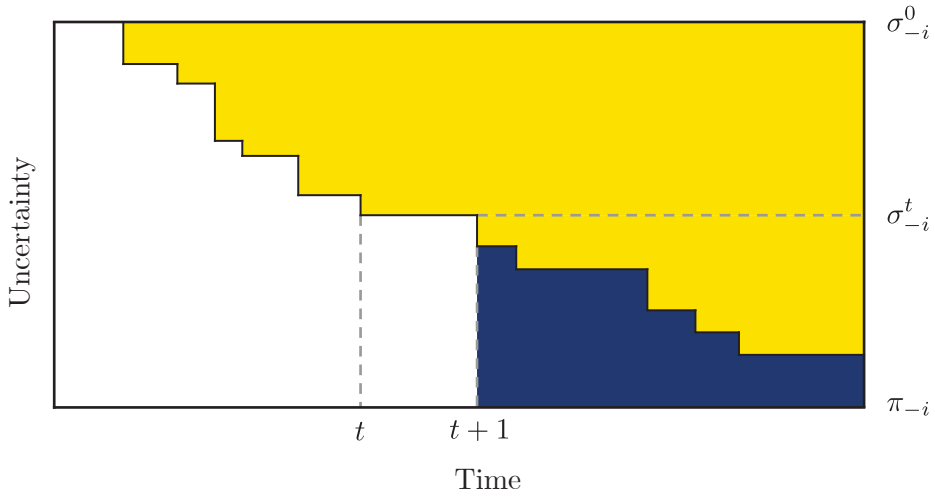


Figure 1: **Conceptual opponent uncertainty over time.** The yellow area represents a uncertainty reduction, for some measure, as a result of updating belief about the distribution of the opponent. The blue area represents approximation error incurred by Q-Mixing.

continually updating the opponent distribution during play, the adjusted Q-Mixing result better responds to the actual opponent.

An ancillary benefit of the opponent classifier is that poorly estimated Q-values tend to have their impact minimized. For example, if an observation occurs only against the second opponent policy, then the Q-value against the first opponent policy would not necessarily be trained well, and thus could distort the policy from Q-Mixing. These poorly trained cases correspond to unlikely opponents and get reduced weighting in the version of Q-Mixing augmented by the classifier.

4.2.1 Running-With-Scissors

We first evaluate Q-Mixing on the *Running With Scissors* (RWS) grid-world game (Vezhnevets et al., 2020; Leibo et al., 2021). This game is a temporal extension of the classic rock-paper-scissors (RPS) game. With this environment we pose the following questions:

1. Can Q-Mixing transfer Q-values from pure-strategy responses to generate *a* mixed-strategy response?

2. Is Q-Mixing capable of transferring Q-values across *all* mixed strategies?

3. Does incorporating an OPC that updates the opponent distribution in Q-Mixing enhance its performance?

In RWS, the agents begin by collecting rock, paper, and scissor items scattered throughout the gird-world. These are added to the player's inventory v_i, which is initialized to have one of each item. The game ends when a player challenges the other to play RPS. Each

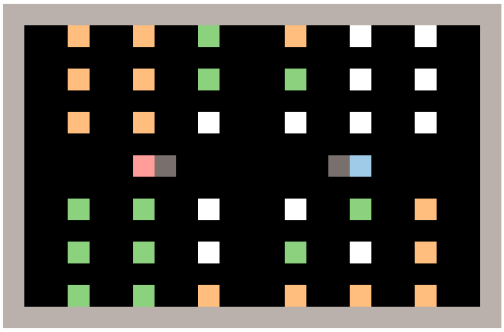


Figure 2: **Running With Scissors example initial state visualization.** The players (red, blue) randomly spawn in one of two positions with a random orientation (dark gray). The environment contains rock (orange), paper (white), and scissor (green) items scattered throughout the room. The item spawn positions remain fixed; however, certain spawn positions stochastically chose the item, while the rest have deterministically chosen items.

player plays a distribution over the actions following the distribution of items in their inventory. The reward can then be calculated as:

$$r_i = \frac{v_i}{\|v_i\|} M \left(\frac{v_{-i}}{\|v_{-i}\|} \right)^{\mathrm{T}} = -r_{-i}, \qquad M = \begin{bmatrix} 0 & -1 & 1 \\ 1 & 0 & -1 \\ -1 & 1 & 0 \end{bmatrix}.$$

The game is a two-player zero-sum game with a small state and action space, enabling inexpensive simulation. Moreover, the game is partially observable, because the agents are only able to view a small 5×5 sub-grid around their position instead of the full 13×21 grid. A rendering of the RWS game can be seen in Figure 2, and additional information about the game is provided in Section F.

In our experiments, we assume the perspective of Player 1 and learn LSTM-based (Hochrieter and Schmidhuber, 1997) response policies using Double DQN (van Hasselt et al., 2016). The state space is a one-hot encoding of the 10 possible occupants of each cell. Resulting in a ravelled vector with length 253, including observation of the player's inventory. The agent has the option of selecting one of 9 actions: move in the four cardinal directions, rotate left or right, challenge (fire/beam) their opponent, or to take no action. The LSTM has a memory size of 128, and the output is projected through a series of fully-connected layers with sizes [128, 64, 64, 9].

We learn three different policies for Player 2 that are then fixed for evaluation. Each of these policies is specialized to prefer collecting one of the three items. To train such a policy, an auxiliary reward of 1 is added each time the agent collects their preferred item.

4.2.2 Transfer onto A Mixed Strategy

With our game and opponents established, we can address our first research question: can Q-Mixing effectively transfer Q-values from pure-strategy responses to generate a mixed-strategy response? To assess this, we examine the method's effectiveness in generating a response policy to a uniform mixed strategy composed of three opponent policies. As a

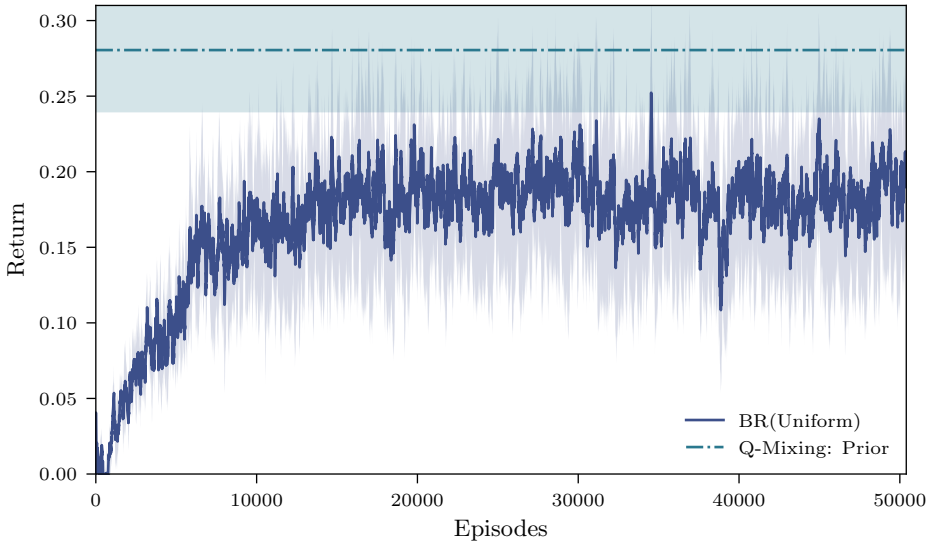


Figure 3: **BR(Uniform)'s learning curve.** BR(Uniform)'s performance is reported in terms of a sliding window of return over the last 100 episodes. Q-Mixing transfers BRs the the opponent's policies that were trained using equal fractions of the training budget available to BR(Uniform). Q-Mixing is then evaluated by simulating its return against each opponent policy for 100 episodes. Performance for both methods is shown with a 95% bootstrap confidence interval.

baseline, we use a best-response policy trained directly against the mixed strategy, denoted as BR(Uniform). Constructing a Q-Mixing policy first requires training best-responses directly against the individual opponent policies. Simultaneously, best-responses are created directly against the individual opponent policies. These pure strategy best-responses employ the same neural network architecture but divide the simulation budget allotted to BR(Uniform) evenly. By allocating the simulation budget in this manner, we account for outcomes potentially influenced by access to larger amounts of simulation data. The responses thus obtained are then utilized to construct the Q-Mixing: Prior policy.

We plot BR(Uniform)'s training curve in Figure 3. On this plot, we also include the simulated performance of Q-Mixing: Prior. It is important to note here that Q-Mixing: Prior requires having previously trained best-responses to each of the opponents individual policies. Since we are only investigating here the quality of this transfer operation, we do not account for this prior simulation time. This disclaimer in mind, we find that Q-Mixing: Prior is able to successfully transfer Q-values to generate a successful best-response policy. In fact, the Q-Mixing: Prior policy outperforms BR(Uniform) without requiring any addition training against the objective. This performance gap can be potentially explained by (a) benefits from specialization, and (b) limitations of RL as an approximate best-response oracle. Q-Mixing allows best-responses to be trained directly against each opponent policy, disentangling the belief in the opponent's policy from the value of each action. As a result, the Q-values need not be risk-averse and choose sub-optimal responses, which is crucial for BR(Uniform) as it must concern itself with the other opponent policies. As for

the limitations of RL as an approximate best-response, we accept this limitation because any approximate method will have its drawbacks. Moreover, both methods were treated with equal hyperparameter tuning and training budget (measured in training timesteps). A practitioner should prefer Q-Mixing, because this result suggests it is much easier to produce quick and strong results (at least for this game). In summary, Q-Mixing shows efficacy in transferring strategic knowledge from opponent policies onto an opponent mixed strategy, confirming our first research question.

4.2.3 Opponent Strategy Space Coverage

The previous result suggests that Q-Mixing may be particularly advantageous when we need to repeatedly generate responses to differing opponent strategies. We investigate this possibility in our second research question. Can Q-Mixing transfer Q-values across *all* of the opponent's mixed strategies?

In order to investigate this question we evaluate the same two methods against a representative coverage of the entire strategy space of the opponent. The strategy sets we consider are all mixtures truncated to the tenths place (e.g., [0.3, 0.4, 0.3]). Q-Mixing methods depend on the changing opponent mixed strategy, unlike BR(Uniform), which is unchanged across opponent strategies. Therefore, when evaluating BR(Uniform) we simulate its performance against each respective opponent policy for 300 episodes[3]. Then the expected performance against each mixed strategy can be calculated by appropriately averaging the mean returns against the respective policies. As for Q-Mixing, we must simulate the performance against each opponent mixture independently. Q-Mixing must be updated to condition on each opponent mixture, and then be simulated against each opponent policy for 300 episodes.

We evaluate both methods based on their return and their *normalized return*. The normalized return, normalizes an estimated return against an opponent policy by the return received by its respective BR. The normalized returns are then averaged according to the opponents mixture. Looking at both performance metrics allows us to more fairly compare the distribution of returns across different opponent policies. For example, consider an opponent with two policies, the later being vulnerable to dramatic exploitation. If we estimated our returns against these policies as 1 and 1000 respectively, then any evaluations against a mixture of these two policies would not fairly account for the performance to the first opponent policy.

Figure 4 shows Q-Mixing's performance across the opponent mixed-strategy space. As we can see in both plots, Q-Mixing strictly dominates the performance of BR(Uniform). A Q-Mixing method that perfectly transfers the component BR knowledge would have a curve that is a horizontal line at 1. This upper bound is unrealistic in practice, because it effectively requires perfectly identifying the opponent policy prior to play. Nevertheless, the difference between Q-Mixing: Prior and 1 represents the potential room for improvement. An astute reader then may wonder how is it possible for Q-Mixing: Prior to achieve a performance greater than 1? This results is from the serendipitous circumstance where the Q-values from the other responses offer advantageous information when weighted together.

3. The number of episodes was chosen because the mean return empirically converged by 300 episodes.

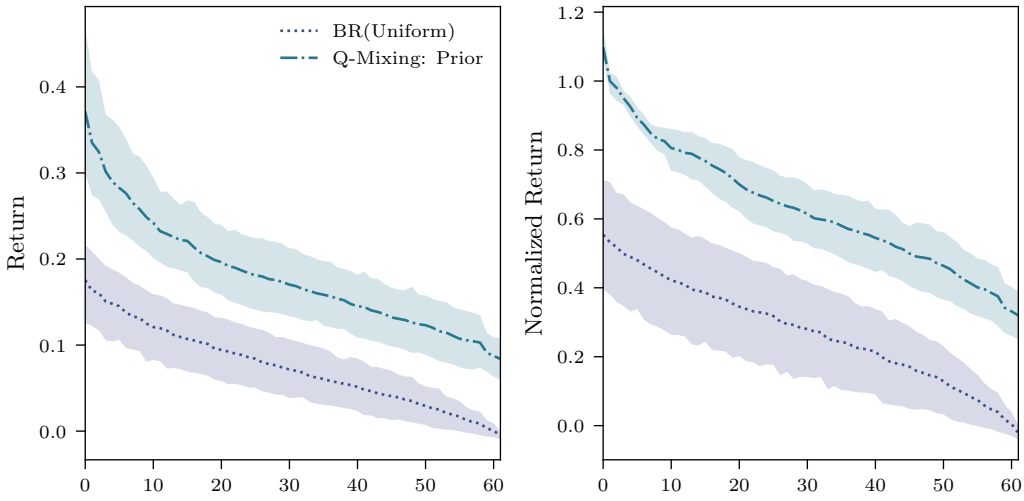


Figure 4: **Coverage of Q-Mixing: Prior on RWS.** The opponent strategies are sorted per-BR-method by the BR's return. Shaded region represents a 95% bootstrap confidence interval over five random seeds. The two methods are trained using the same simulation budget. The left plot, evaluates each method by their return. On the right plot, we instead normalize the return by the performance of the BR to each opponent policy.

4.2.4 Q-value Regularization

A possible explanation for the performance improvements observed in Q-Mixing is that it benefits from value ensembling. Value ensembling has been shown to reduce noise in BR learning, leading to a more stable training target (Anschel et al., 2017). In the context of Q-Mixing, this would suggest that the advantage is derived from the aggregation of value predictors, rather than the specialization of per-opponent policy responses or their weighting. To determine if regularization is the cause of our previous findings, we examine the benefits of regularization in this domain using a set of independently trained Q-functions. If averaging these Q-functions corresponds to performance improvements, then regularization could be a plausible explanation for the performance improvements seen in Q-Mixing.

In Figure 5, we show the performance of uniformly averaging the Q-values from an increasing number of the independently trained DQNs. There appears to be no consistent trend in performance improvement or degradation as additional DQNs are introduced. Going from one to two DQNs results in an improvement, but adding a third DQN eliminates the previous gains and further reduces performance. In the same figure, we show the performance of each DQN in isolation. The individual performance allows us to better understand the regularization results: changes in regularization performance coincided with the relative performance of each newly added Q-function.

Next, we explore whether any trend emerges when regularizing the outputs of two DQNs together. This experiment is inspired by the hypothesis that the order in which DQNs are added during evaluation may have confounded our previous results when regularizing

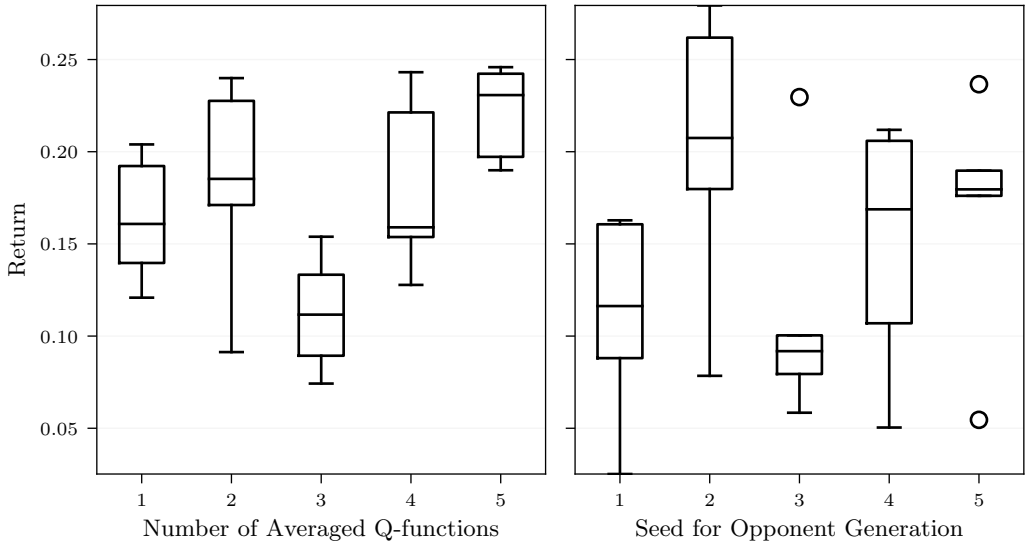


Figure 5: **Effects of averaging Q-values from DQN.** (Left) performance of averaging the Q-values from an increasing number of DQNs against the uniform mixed-strategy opponent. (Right) the respective performance of the individual DQNs.

a group of DQNs. In Figure 6, we plot the improvement in return when an additional DQN is averaged with a baseline DQN. 12 of the 25 combinations saw an improvement in performance from the addition of another Q-value estimate. In conjunction with the previous results, we find no compelling evidence to suggest that Q-value regularization is the primary factor contributing to the benefits of Q-Mixing.

4.2.5 Opponent Classification

Our third research question is: can the use of an OPC that updates the opponent distribution in Q-Mixing improve its performance? During play against an opponent sampled from the mixed strategy, the player is able to gather evidence about which opponent they are playing. We hypothesize that leveraging this evidence to weight the importance of the respective BR's Q-values higher will improve Q-Mixing's performance.

To verify this hypothesis, we train an OPC using the replay buffers associated with each BR policy. These are the same buffers that were used to train the BRs, and cost no additional compute to collect. This data is used to train an OPC that outputs a distribution over opponent pure strategies for each observation. The OPC is implemented with a deep neural network with the same architecture as the policies, save the last layer that has size that equals the number of opponent policies (in this case, 3). The classifier is trained to predict from an observation, sampled across the replay buffers, the respective pure strategy index it occurred against with a cross-entropy loss.

We evaluate Q-Mixing: OPC by testing the performance on a representative coverage of the mixed-strategy opponents illustrated in Figure 7. We found that the Q-Mixing: OPC policy performed stronger against the full opponent strategy coverage than Q-Mixing:

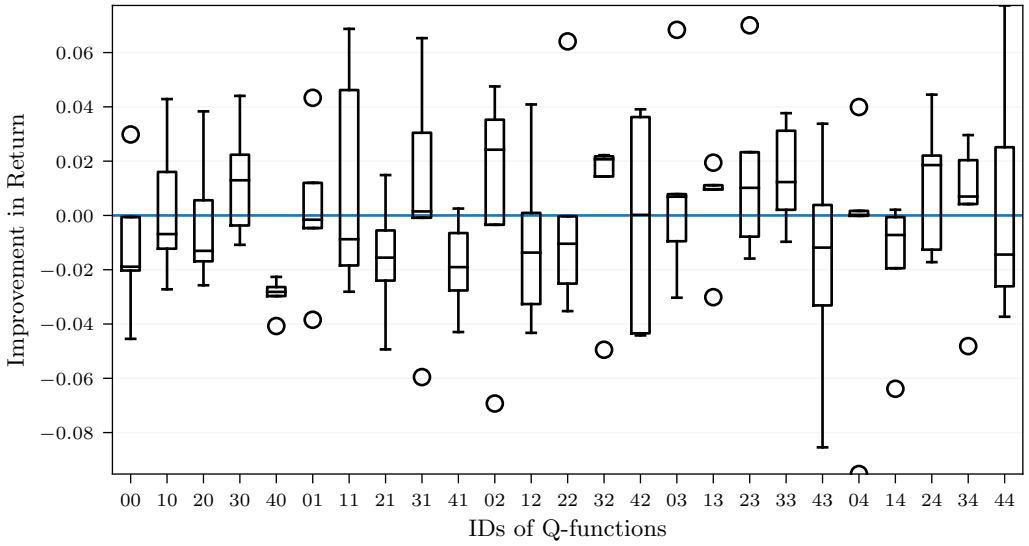


Figure 6: **Performance improvement of averaging an additional DQN.** The x-axis denotes the IDs of the baseline DQN (first index), and the additional DQN (second index) added to regularize its' Q-values.

Prior. This result supports our hypothesis that an OPC can identify the opponent's pure strategy and enable Q-Mixing to chose the correct BR policy. However, Q-Mixing: OPC method has a larger variance in return. This variance is not found in the normalized return, suggesting that the larger variance is a result of the range of exploitability of the respective opponent policies. Previously this trend in variance was thought to be a result of opponent missclassification.

4.3 Accounting for Future Uncertainty

Q-Mixing as we have seen it so far can handle the timeless setting (single-state), and consider information from the past and present. So far, the belief in the opponent's policy is assumed constant into the future. However, the future offers additional opportunities to gather evidence that may influence the belief in the opponent's policy. And in these future states, the belief at that point should be updated to reflect the cumulative evidence gathered about the opponent's policy.

4.3.1 Opponent-Policy Identification Game

To illustrate this important detail we introduce the *opponent-policy identification game*. This game, illustrated in Figure 8, is a form of coordination game where an agent has the option to pay a cost to observe the opponent's policy prior to coordination. Success in this game requires that the agent can appropriately trade-off the cost of information gathering with the benefit of a more informed future belief in the opponent's policy.

At the root, the opponent draws a policy from their mixed strategy $\sigma_{-i} \in \Delta(\{\pi^{\mathrm{L}}_{-i}, \pi^{\mathrm{R}}_{-i}\})$. Player i begins in initial state s^0_i, where they have no information about the opponent's

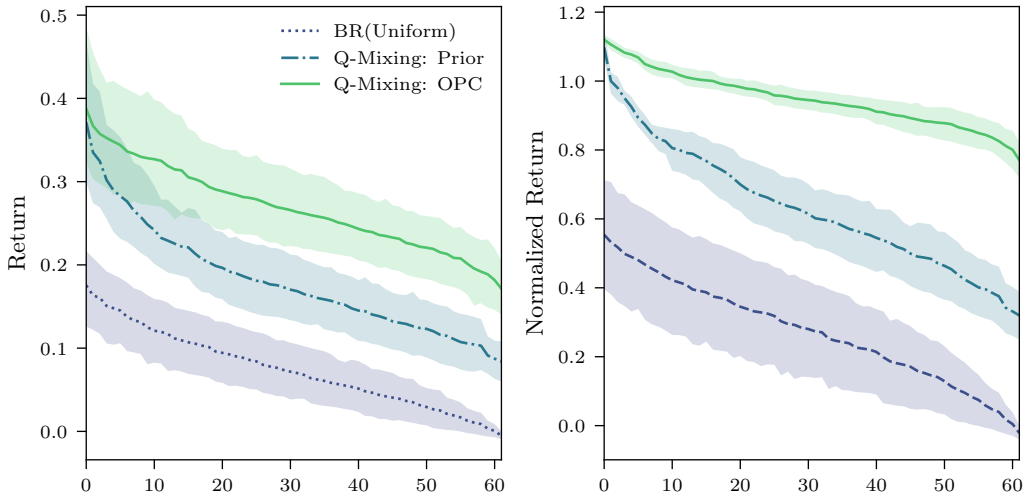


Figure 7: **Coverage of Q-Mixing: OPC on RWS.** The opponent strategies are sorted per-BR-method by the BR's return. Shaded region represents a 95% bootstrap confidence interval over five random seeds. The two methods are trained using the same simulation budget. The left plot, evaluates each method by their return. On the right plot, we instead normalize the return by the performance of the BR to each opponent policy.

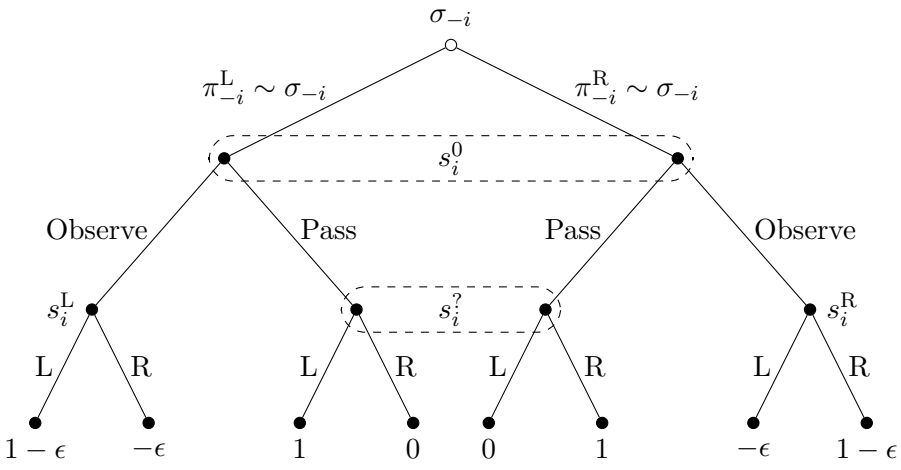


Figure 8: **Opponent-policy identification game.** Leaf node values reflect the full return for player i.

policy. From this state, player i has the option to observe the opponent's identity—that is, whether it chose the L or R policy—for a cost of ϵ. If they exercise this ability, they transition to a state of knowing the opponent's identity $\{s_i^{\mathrm{L}}, s_i^{\mathrm{R}}\}$ (where the superscript corresponds to the opponent's policy). If they pass on this opportunity, then they remain in a state of ignorance $s_i^?$.

After player i has observed or passed, the player chooses L or R. If i's choice matches the opponent's policy label, the reward to both players is 1; otherwise they receive no reward. The total return, or payoff, for player i in this game is then the cost of observation, if exercised, plus the reward conditional on successful coordination.

The BR to an opponent in this game depends on both the opponent's strategy σ_{-i} and the cost of observation ϵ. For instance, consider the uniform mixed strategy opponent with a small observation cost:

$$\sigma^{\mathrm{U}}_{-i} \leftarrow \mathrm{Uniform}(\{\pi^{\mathrm{L}}_{-i}, \pi^{\mathrm{R}}_{-i}\}) \qquad \epsilon \leftarrow 0.03.$$

The best-response in this setting, $\mathrm{BR}(\sigma^{\mathrm{U}}_{-i})$, is to observe the opponent's policy and play the appropriate response. Consequently, $\mathrm{BR}(\sigma^{\mathrm{U}}_{-i})$ receives a return of 0.97. Had this response policy chosen to not observe the opponent's identity, then they can do no better than chance during coordination and would receive a return of 0.5.

Now, let us consider our transfer learning problem within the aforementioned setting of the opponent-policy identification game. The problem asks us to construct $\mathrm{BR}(\sigma^{\mathrm{U}}_{-i})$ from $\{\mathrm{BR}(\pi^{\mathrm{L}}_{-i}), \mathrm{BR}(\pi^{\mathrm{R}}_{-i})\}$ given σ_{-i}. The pure strategy best-responses in effect already know the identity of the opponent. This means that they never decide to observe the identity, as doing so incurs an unnecessary cost without providing any additional information. Hence, we can summarize the pure strategy responses as follows (the BRs are rewritten as conditioned policies for ease of notation):

$$\pi_i(s^0_i \mid \pi^{\mathrm{L}}_{-i}) = \mathrm{Pass} \qquad \pi_i(s^0_i \mid \pi^{\mathrm{R}}_{-i}) = \mathrm{Pass}$$
$$\pi_i(s^?_i \mid \pi^{\mathrm{L}}_{-i}) = \mathrm{L} \qquad \pi_i(s^?_i \mid \pi^{\mathrm{R}}_{-i}) = \mathrm{R}.$$

From these component policies we cannot construct $\mathrm{BR}(\sigma^{\mathrm{U}}_{-i})$, because it contains a new strategic behavior not present in the provided responses: the need to gather additional information about the opponent. As we saw in the problem setup, $\mathrm{BR}(\sigma^{\mathrm{U}}_{-i})$ receives a higher return if it chooses to take the observe action and appropriately respond. The information-gathering behavior fundamental to the correctness of a mixed-strategy response is, in this example, not present in responses to the opponent pure strategies.

In the general opponent-policy identification game, we can successfully transfer responses when acquiring additional information on the identity of the opponent is not worthwhile. Information-gathering is not worthwhile when

$$1 - \epsilon < \max_{\pi_{-i}} \sigma_{-i}(\pi_{-i}),$$

where $\sigma_{-i}(\pi_{-i})$ is the probability of the opponent playing π_{-i}. If this inequality holds, then the cost for observation outweighs its benefit. Conversely, if the inequality is reversed, then $\mathrm{BR}(\sigma^{\mathrm{U}}_{-i})$ stands to benefit from taking actions to acquire knowledge about the opponent's identity. This information-gathering behavior will not be evident in the component response policies, because it is not optimal for them to incur the information-gathering cost. As a result, $\mathrm{BR}(\sigma^{\mathrm{U}}_{-i})$ cannot be constructed without injecting additional strategic knowledge related to opponent policy identification.

4.3.2 Q-Mixing with Value Iteration

To account for future uncertainty, Q-Mixing must be able to update its successor observation values given future evidence of the opponent's policy. This can be done by expanding the Q-value into its components: expected reward under the current belief in the opponent's policy, and our expected next observation value. By updating the second term to recursively reference a new opponent belief we can account for changing beliefs in the future. The extended formulation, Q-Mixing: Value Iteration (QMVI), is given by:

$$Q_i^*(o_i^t, a_i^t \mid \sigma_{-i}) = \sum_{\pi_{-i} \in \Pi_{-i}} \psi_i(\pi_{-i} \mid o_i^t, \sigma_{-i}) \cdot \left[r_i(o_i^t, a_i^t \mid \pi_{-i}) + \gamma \mathbb{E}_{o_i^{t+1}} \left[V^*(o_i^{t+1} \mid \sigma_{-i}) \right] \right]. \quad (9)$$

If we assume that we have access to both a dynamics model and the observation distribution dependent on the opponent, then we can directly solve for this quantity through Value Iteration (Algorithm 1). These requirements are quite strong, essentially requiring perfect knowledge of the system with regards to all opponent policies. The additional step of Value Iteration also carries a computational burden, as it requires iterating over the full state and action spaces. Though these costs may render QMVI infeasible in practice, we provide Algorithm 1 below as a way to ensure correctness in Q-values.

Algorithm 1: Value Iteration: Q-Mixing

Input: $\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \epsilon, \gamma$

$V_0(s \mid \sigma_{-i}) \leftarrow \sum_{\pi_{-i}} \sigma_{-i}(\pi_{-i}) Q(s, a \mid \pi_{-i})$

do

$Q_t(s, a \mid \sigma_{-i}) \leftarrow \sum_{\pi_{-i}} \psi(\pi_{-i} \mid s, \sigma_{-i}) \sum_{s',\ r} \mathcal{T}(s', r \mid s, a, \pi_{-i})[r + \gamma V_{t-1}(s' \mid \sigma_{-i})]$

$V_t(s \mid \sigma_{-i}) \leftarrow \max_a Q_t(s, a \mid \sigma_{-i})$

$\pi_t(s \mid \sigma_{-i}) \leftarrow \arg\max_a Q_t(s, a \mid \sigma_{-i})$

while $\exists_{s \in \mathcal{S}} \ |V_t(s) - V_{t-1}(s)| > \epsilon$

Output: $V_t,\ Q_t,\ \pi_t$

QMVI is reducible into the traditional value iteration algorithm. Consider we construct a new aggregate MDP by combining both the original MDP and the opponent's dynamics (represented by the combination of their strategy and policies). The aggregated MDP is both stationary and may be stochastic, because the opponent is not learning and may randomize their play. Convergence properties of value iteration are inherited by QMVI by considering the application of value iteration on said aggregate MDP.

5. Opponent-Policy Belief

As we have seen from both policy- and value-based strategic knowledge transfer, a crucial component to their success is the ability to identify their opponent's policy. It comes at no surprise that response knowledge against a particular opponent policy may prove fruitless if you cannot reasonably expect that you are interacting with that opponent policy. With this in mind, we turn our attention towards the problem of modeling the agent's belief of their opponent's current policy.

Correctly identifying your opponent's policy facilitates playing the appropriate best-response. The identification problem is typically not straightforward due to the limited information that can be observed about your opponent's strategy. Reasonably, it will not be the case that the opponent will tell their competitor their policy. Instead, one must collect information about their opponent and use this to inform a belief of the opponent's policy.

Information about the opponent's strategy may be collected prior to gameplay and during interaction with a particular policy. The former information informs one's *prior* on the opponent's policy. Whereas, the later pertains to *evidence* gained to inform the *likelihood* of the current opponent policy. The likelihood may be calculated with only the evidence at the current timestep, or may use the full *history* of evidence gained from the start of the episode until the present. A prior and history-based likelihood model together constitute a fully informed belief model. This raises a key question: what components to the belief model are critical to its success?

Before we answer this question, we must first understand what makes a good likelihood and prior. In the preceding section we demonstrated the efficacy of two methods for maintaining a belief of the opponent's current policy: (1) the prior defined by the opponent's mixed strategy, and (2) a neural network likelihood model (trained through the auxiliary task of classification). Both of these methods failed to fully take advantage of all of the information available; in fact, each method lacks what the other provides. The prior method fails to account for evidence during play with a particular opponent policy, and the classifier fails to be amenable to changes in the prior belief of the opponent policy. Moreover, both previous options do not consider any historical evidence in their belief calculation.

5.1 Opponent Likelihood Model

We begin by investigating the design of an opponent likelihood model. In the previous section, we trained a neural network classifier to predict which opponent we are playing against based on the game history. Assuming a well calibrated model, this will accurately predict the posterior probability of each opponent, but only if the distribution over opponent policies in the data used to train the model was the same as the opponent's strategy used in Q-Mixing. If the distribution of the data was to differ from the data experienced by the Q-Mixing policy, we would need to correct for this change.

Let $\bar{\sigma}$ be the opponent mixture used to train the OPC and $h \in \mathcal{H}$ be any realizable history, then we can define and unpack OPC ψ by Bayes Theorem as follows (up to model approximation errors):

$$\begin{aligned} \psi(\pi \mid h) &= \frac{p(h \mid \pi) \cdot \bar{\sigma}(\pi)}{p(h \mid \bar{\sigma})} \\ &\propto p(h \mid \pi) \cdot \bar{\sigma}(\pi). \end{aligned} \tag{10}$$

This results in two terms directly corresponding to the likelihood and prior of our opponent's identity. Previously, when investigating the opponent identification, a neural network was trained to implicitly model this distribution through classification. A prior on the opponent's policy is implicitly trained into the OPC through the class balance $\bar{\sigma}$ in the dataset. In other words, if the data used to trained the OPC contained mostly experience against

opponent two, then this may bias the OPC to favor predicting that opponent. In the previous experiments the data was balanced across classes; following this, the OPC adopted a uniform prior.

When using Q-Mixing in a training procedure (such as in Section 6) to respond to a mixed-strategy opponent, our agent has access to the opponent's mixture. The OPC fails to take advantage of this new prior knowledge; the distribution used to train the likelihood model now fails to match the true opponent distribution. However, because the class distribution in training $\bar{\sigma}$ is known, rather than using the OPC directly, we can treat it as an Opponent Likelihood Model (OLM), up to a multiplicative constant.

Instead of assuming the same prior $\bar{\sigma}$ across the entire opponent strategy-space we would like to update ψ, our likelihood model, when given new information about the prior distribution of the opponent policies. Consider a different mixed strategy σ, we would like to update ψ so that it instead approximates the distribution:

$$\begin{aligned} p(\pi \mid h, \sigma) &= \frac{p(h \mid \pi) \cdot \sigma(\pi)}{p(h \mid \sigma)} \\ &\propto p(h \mid \pi) \cdot \sigma(\pi). \end{aligned} \tag{11}$$

By rearranging Equation 10 we get the likelihood in terms of our trained model:

$$p(h \mid \pi) \propto \frac{\psi(\pi \mid h)}{\bar{\sigma}(\pi)}. \tag{12}$$

Then we may substitute Equation 12 into Equation 11, facilitating a correction for the prior:

$$\begin{aligned} p(\pi \mid h, \sigma) &\propto \frac{\psi(\pi \mid h)}{\bar{\sigma}(\pi)} \sigma(\pi) \\ &= \psi(\pi \mid h) \frac{\sigma(\pi)}{\bar{\sigma}(\pi)} \end{aligned} \tag{13}$$

This new formulation addresses the concerns we raised earlier. First, we account for prior knowledge of the opponent by directly using our prior σ. Second, we can learn a likelihood model of the evidence h through our previous method for constructing an OPC. Finally, we correct for the prior used during the training of the evidence likelihood.

By substituting the method used to compute likelihoods in Q-Mixing, we conduct an ablation study to examine the impact of both the new prior knowledge and the evidence obtained during gameplay. The respective impacts are assessed by establishing four version of Q-Mixing, as listed in Table 3, each varying in the information sources that inform their likelihood calculations. The version of Q-Mixing developed in this section, incorporating both sources of information, is hereafter referred to as *Q-Mixing: OLM*:

$$Q_{\pi_i}(h_i, a_i \mid \sigma_{-i}) \propto \sum_{\pi_{-i}} \psi_i(\pi_{-i} \mid h_i) \frac{\sigma_{-i}(\pi_{-i})}{\bar{\sigma}_{-i}(\pi_{-i})} Q_{\pi_i}(h_i, a_i \mid \pi_{-i}). \tag{14}$$

This version of Q-Mixing is constructed derived from substituting the classifier-based likelihood in Equation 8 with the corrected likelihood in Equation 13. The likelihood model ψ_i

Formula	Prior	Likelihood
$Q_{\pi_i}(h_i, a_i \mid \sigma_{-i}) \propto \sum_{\pi_{-i}} \bar{\sigma}_{-i}(\pi_{-i}) Q_{\pi_i}(h_i, a_i \mid \pi_{-i})$		
$Q_{\pi_i}(h_i, a_i \mid \sigma_{-i}) \propto \sum_{\pi_{-i}} \sigma_{-i}(\pi_{-i}) Q_{\pi_i}(h_i, a_i \mid \pi_{-i})$	✓	
$Q_{\pi_i}(h_i, a_i \mid \sigma_{-i}) \propto \sum_{\pi_{-i}} \psi_i(\pi_{-i} \mid h_i) Q_{\pi_i}(h_i, a_i \mid \pi_{-i})$		✓
$Q_{\pi_i}(h_i, a_i \mid \sigma_{-i}) \propto \sum_{\pi_{-i}} \psi_i(\pi_{-i} \mid h_i) \frac{\sigma_{-i}(\pi_{-i})}{\bar{\sigma}_{-i}(\pi_{-i})} Q_{\pi_i}(h_i, a_i \mid \pi_{-i})$	✓	✓

Table 3: **Comparison of Q-Mixing with differing opponent likelihood models.** The prior column denotes that the likelihood model can correct for updated prior knowledge. The likelihood column denotes whether the model updates the posterior given evidence during play.

here is defined assuming perfect recall, or access to the full history h_i. It can be calculated as the product of the probability of each observation occurring against said opponent:

$$\psi_i(\pi_i \mid h_i) = \prod_{o_i \in h_i} \psi_i(\pi_i \mid o_i). \tag{15}$$

This can be efficiently calculated with a stateful policy that maintains the previous timesteps' belief and updates its posterior with the current observation's likelihood. We also consider a simpler likelihood that only depends on the current observation. Comparing these versions illuminates the predictive power of a single observation.

5.2 Baseline Opponent Classifiers

Before we can begin to understand the details of what constitutes a good method for maintaining an opponent identifier, we must first understand where we start to contextualize progress. Do we even need to perform opponent identification? If the tools we already have perform well then pursing this investigation may be a nonstarter.

In the simplest case, let's suppose that we do not need to explicitly consider the identity of the opponent. This amounts to selecting a best-response policy irrespective of information provided about the opponent's strategy. In Figure 9, we compare Q-Mixing: Prior directly against the best-response policies. We look at each method by evaluating their coverage across the opponent's strategy space in the RWS game. This is the same experimental methodology that we introduced in Section 4.

The best-responses to the opponent's policies are denoted BR(X), where X is the index of the opponent's policy in their strategy set. These baselines let us investigate the option of choosing the simplest opponent classifier: a classifier that outputs a constant value. These also allow us to see if any one particular best-response policy dominates the others. Such a dominant policy could serve as a sufficiently good policy against the mixed strategy.

We also consider the best-response to the opponent's uniform mixed strategy, labelled BR(Uniform). BR(Uniform) implicitly should perform both opponent belief maintenance and best-response to perform well. Investigation of this baseline should offer some insights into how easily identifiable the opponent's policy is from the observation. If this is an easy task, then the implicit two-step response strategy of identification and then response is easy

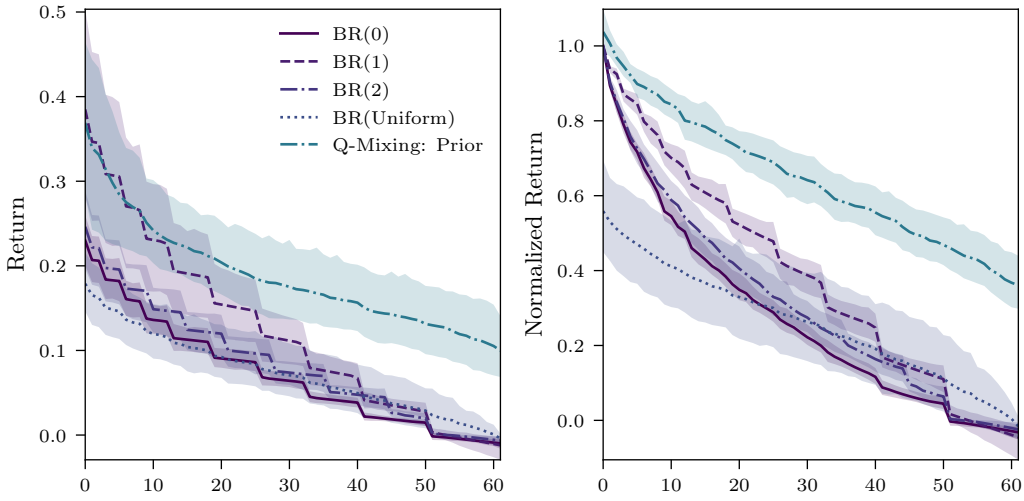


Figure 9: **Coverage of best-response policies.** The opponent strategies are sorted per-BR-method by the BR's return. Shaded region represents a 95% bootstrap confidence interval over five random seeds. The two methods are trained using the same simulation budget. The left plot, evaluates each method by their return. On the right plot, we instead normalize the return by the performance of the BR to each opponent policy.

to perform at all game states. Therefore, a good BR(Uniform) should highly correlate its Q-values with the observation features corresponding to the opponent's identity.

In Figure 9, we can see that the best-responses to the opponent pure strategies generalize poorly. This can be seen in the right plot, where the the curves start out at 1.0, which denotes that the response policy was playing against the corresponding opponent pure strategy. Then when the opponent plays any mixed strategy, the performance quickly falls off. Recall, that the normalized return reports the proportion of the return received when compared to the return received by playing the true response policies. A decline in this chart suggests that the pure strategy response policies are unable to exploit the different opponent policies. This indicates that the opponent's strategy space is sufficiently diverse that we cannot rely on just a response to an opponent pure strategy.

In the same figure, we can also see that BR(Uniform) at its best performs at just over half its potential. A reasonable retort is to remark that this result indicates a failure in the approximate best-response oracle. It is true that using Deep RL as an approximate oracle method can result in variable performance to the whim of policy implementation and training hyperparameters. It is possible that under the perfect settings, Deep RL may produce a response policy to the mixed strategy that also exhibits generalization capabilities. Despite this true criticism, in practice, finding that setting may consume more resources than simply directly tackling the problem from a less idealized setting. In this work, we spent roughly equivalent resources attempting to optimize the learning settings for the responses to the pure and mixed strategies. Therefore, we argue that the results herewithin represent a roughly fair practical comparison of methods.

Q-Mixing: Prior outperforms both the pure- and mixed-strategy best-response policies. This should come at no surprise, because Q-Mixing: Prior can use the *additional* information of the opponent's strategy at evaluation time. The other methods offer no flexibility given the known change in opponent strategy. Therefore, there is a clear benefit to explicitly including a mechanism for opponent identification within a policy. Note, that in Figure 9 when interpreting outperformance, it is not the case that Q-Mixing: Prior is better than the specialized BRs against their respective opponents. The performance of each curve is sorted for each method separately, the strategies represented by each point on the x-axis often differ between methods.

5.3 Frequency-Based Classification

The result from the previous section indicate that strictly using best-responses may be insufficient. However, each of these respective best-responses performs well against their respective opponent. Q-Mixing's ability to fully utilize the component best-responses depends on its efficacy at identifying the opponent's policy. In this section we increase the complexity of our Q-Mixing policy by a single step. Instead of considering only a prior (fixed or adjusted to the correct opponent strategy), we investigate inclusion of evidence during play to inform the identity of the opponent. As the trained classifier we saw in Section 4 used a neural network, attempting to understand its success and failure modes presents itself as an entirely disparate research direction.

As a stopgap, we will now consider an *observation frequency* based opponent-policy likelihood. This likelihood model is based of a simple statistic, and will allow us to dive into what contributes to the success of Q-Mixing. It will weight the SRQVs by the relative frequency of the observation occurring against each opponent. We use the replay buffers $\mathbb{B}_i$ used to train the respective responses i as datasets to calculate the observation frequencies. More formally, we compute the weight of assigned to each response j for player i as:

$$w_j(o_i) = \frac{\sum_{\bar{o_i} \in \mathbb{B}(\mathrm{BR}_i(\pi^j_{-i}))} [\bar{o_i} = o_i]_{\mathbb{1}}}{\sum_{\bar{o_i} \in \bigcup_k \mathbb{B}(\mathrm{BR}_i(\pi^k_{-i}))} [\bar{o_i} = o_i]_{\mathbb{1}}}, \tag{16}$$

and *Q-Mixing: Observation Frequency* (Freq) follows from this formulation:

$$Q_i^{\mathrm{Freq}}(o_i, a_i \mid \sigma_{-i}) = \begin{cases} \sum_{\pi^j_{-i}} w_j(o_i) \cdot Q_i(o_i, a_i \mid \pi^j_{-i}) & o \in \bigcup_k \mathbb{B}(\mathrm{BR}_i(\pi^k_{-i})) \\ \sum_{\pi^j_{-i}} \frac{1}{|\Pi_{-i}|} \cdot Q_i(o_i, a_i \mid \pi^j_{-i}) & \text{Otherwise} \end{cases}. \tag{17}$$

We compare Q-Mixing: Freq with its prior-only versions of Q-Mixing and the learned classifier in Figure 10. Q-Mixing: Uniform Prior is another baseline introduced here, representing the performance of Q-Mixing when it is not given the opponent's strategy, but rather maintains a constant strategy (in this case, the uniform strategy) across all opponent strategies. Interestingly, Q-Mixing: Uniform Prior outperforms its opponent across all strategies. This outcome can be attributed to the tendency of value functions to overestimate values. As a result, the value functions associated with less likely opponents are typically suppressed. This phenomenon acts as an implicit form of opponent modeling, which arises due to the specific experimental setup chosen in this case. However, this observation may not hold true in general.

	0	1	2
0	18,854	341	237
1		20,464	424
2			21,040

Table 4: **Unique observations common across the responses' replay buffers.** The rows and columns represent the replay buffers of the approximate best-response policies' replay buffers. Each cell represents the count of the number of unique observations that occur in both replay buffers.

The result in Figure 10 indicates that the performance of Q-Mixing: Freq falls between Q-Mixing: Prior and Q-Mixing: Uniform Prior. This finding is initially concerning, as it suggests that the evidence does not aid in opponent identification. In fact, Q-Mixing: Freq performs only marginally better than Q-Mixing: Uniform Prior, which receives no information about the opponent's strategy.

Why doesn't the observational evidence lead to improvements in Q-Mixing? To answer this question, we need to examine the replay buffers underlying Q-Mixing: Freq's implementation. Each replay buffer contains 100,000 experiences and corresponding observations. From these experiences, the replay buffers have 18,854, 20,464, and 21,040 unique states, respectively. This implies that agents do revisit observations quite frequently.

However, the more critical question is how many of these unique observations co-occur between pairs of replay buffers? Co-occurrence indicates that the agent should be uncertain about their opponent's identity, and the relative frequency could provide valuable information. In Table 4, we present the co-occurrence frequencies. This table reveals that very few observations (less than 500) ever co-occur between any two replay buffers. This means that if an observation appears in a replay buffer, it is highly informative of the opponent's identity.

So far we have established that our frequency baseline fails to effectively use evidence during play to improve Q-Mixing. Next, we saw that if an observation is in one of the response policies' replay buffers then it is highly informative of the opponent's identity. This suggests an intuitive contradiction with the former result. The resolution to this conflict lies in the second case present in the observation-frequency formula (Equation 17). What happens when an observation *is not* present in any replay buffer?

We begin by simulating Q-Mixing: Freq against each respective opponent for 300 episodes. From this simulation results we looked at the proportion of observations encountered that did not occur in any replay buffers. We refer to this situation as a cache-miss. The percentage of cache-misses against each opponent is $87 \pm 1\%$, $69 \pm 3\%$, and $85 \pm 2\%$. Here in lies the problem with this baseline. Despite the replay buffers being largely informative, the information is never able to bear fruit, because the agent is mostly experiencing novel states. Thus, it behaves only slightly better than Q-Mixing: Uniform Prior, because upon a cache-miss it implements the Q-Mixing: Uniform Prior policy (the second case in Equation 17).

The previous result demonstrates that the observations used to construct our policy often differ from those experienced during evaluation. The key takeaway from this experiment

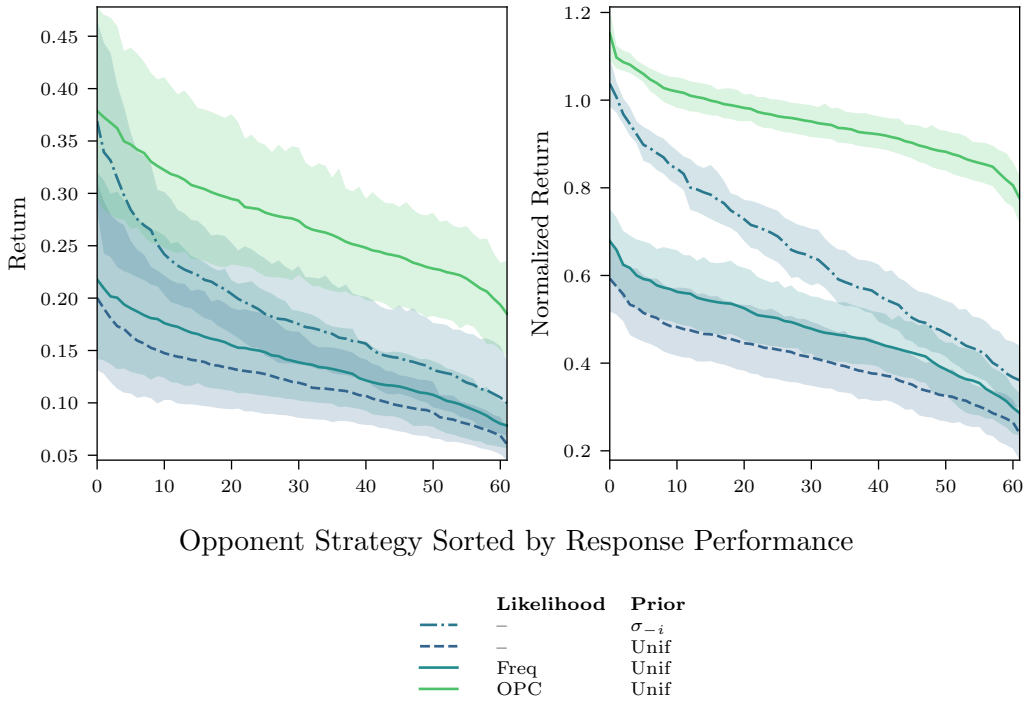


Figure 10: **Coverage of baseline variants of Q-Mixing.** The opponent strategies are sorted per-BR-method by the BR's return. Shaded region represents a 95% bootstrap confidence interval over five random seeds. The two methods are trained using the same simulation budget. The left plot, evaluates each method by their return. On the right plot, we instead normalize the return by the performance of the BR to each opponent policy. This is the only figure that contains results for Q-Mixing: Freq that does not maintain a belief.

is that this version of Q-Mixing: Freq does not take the past into account. Currently, the classifier receives the present observation and predicts the opponent's identity. This prediction is based on the probability of each opponent for the current observation, which can be used to create and maintain a belief about the opponent's identity throughout an episode.

Upon revisiting our observation-frequency baseline, we can readily identify the significant benefits of belief maintenance. In the initial version of Q-Mixing: Freq, the predicted opponent likelihood becomes the uniform distribution in the event of a cache miss. However, by maintaining a belief over time, the likelihood from the previous prediction is preserved in the case of a cache miss. This approach facilitates the accumulation of evidence across time, even when faced with new observations. From this point forward, we will focus on a version of Q-Mixing: Freq that incorporates belief maintenance over time.

Figure 11 displays the coverage curves for various Q-Mixing: Freq variants. These plots show that maintaining a belief significantly enhances the performance of Q-Mixing: Freq. These improvements can be attributed to the agent's ability to accumulate all evidence about their opponent. Additionally, when the agent encounters a new observation, they no longer have complete uncertainty about their opponent. Instead, their previous belief about their opponent remains. As most observations experienced were novel, this means that the agent can now historic evidence of their opponent.

Continuing from the Q-Mixing: Freq results, we can see that the prior plays a much smaller role in the performance of opponent identification. An informative prior provides a small increase in performance across the opponent's strategy space. The prior serves a minor role, because throughout an episode enough evidence about the opponent is collected to overwhelm the contribution from the prior. Still, a prior offers benefits early in an episode, by allowing an agent to deviate to a more exploitative response policy earlier. In practice, we will have access to the true opponent strategy; therefore, this result unsurprisingly suggests that it should be used as the prior.

5.4 Learned Classifier

Finally, we return to our learned opponent classifier. We provide both coverage curves for Q-Mixing: OPC in Figure 12 and opponent classification accuracy of all methods (at the end of the episode) in Table 5. The learned classifier further improves upon the frequency baseline, as shown in Figure 13. The previous investigation into the performance of the frequency baseline offers insights into the benefits gained from the learned version. With regards to extracting evidence of the opponent's identity: the OPC learns to extract features salient for opponent classification from its observations; whereas, the frequency baseline can only gather evidence from known observations contained in a replay buffer.

In summary, we compile all of the coverage curves analyzed in this section in Figure 13. The key take-away is that both the prior and evidence gained matter for successful belief calculation. The prior, is often of less importance, because it is typical to gain sufficient evidence during play to overwhelm the impact of the prior. Nevertheless, in games where the first few moves are crucial, the prior may prove fundamental to the method's success. A trained opponent-policy classifier may also learn to extract features that are predictive of the opponent's policy even in novel settings.

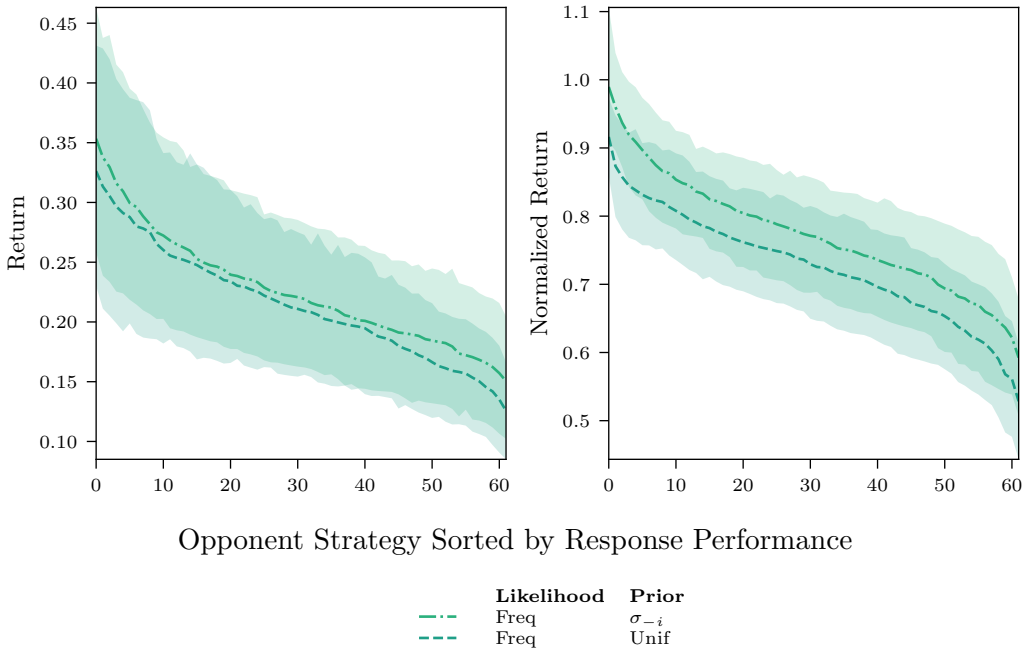


Figure 11: **Coverage of variants of Q-Mixing with a frequency-based likelihood.** The opponent strategies are sorted per-BR-method by the BR's return. Shaded region represents a 95% bootstrap confidence interval over five random seeds. The two methods are trained using the same simulation budget. The left plot, evaluates each method by their return. On the right plot, we instead normalize the return by the performance of the BR to each opponent policy.

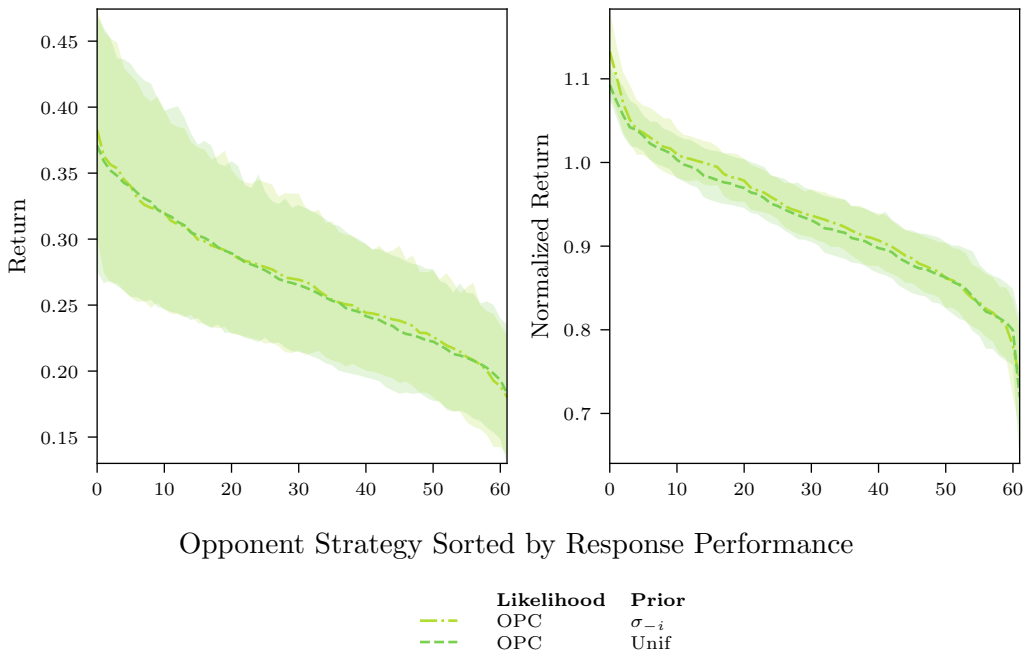


Figure 12: **Coverage of variants of Q-Mixing with a learned likelihood.** The opponent strategies are sorted per-BR-method by the BR's return. Shaded region represents a 95% bootstrap confidence interval over five random seeds. The two methods are trained using the same simulation budget. The left plot, evaluates each method by their return. On the right plot, we instead normalize the return by the performance of the BR to each opponent policy.

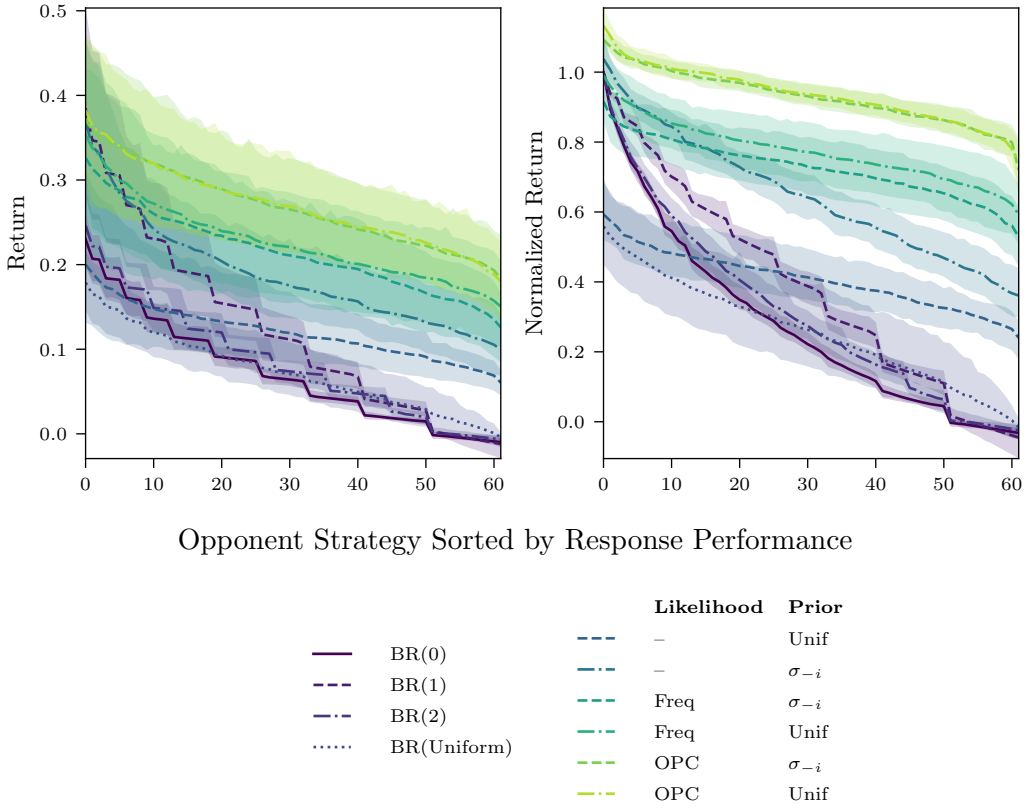


Figure 13: **Coverage of all methods on RWS.** The opponent strategies are sorted per-BR-method by the BR's return. Shaded region represents a 95% bootstrap confidence interval over five random seeds. The two methods are trained using the same simulation budget. The left plot, evaluates each method by their return. On the right plot, we instead normalize the return by the performance of the BR to each opponent policy. A detailed description of generating coverage curves is supplied in Appendix D. Appendix D also includes the specific opponent strategies in terms of ranking. Appendix E includes depictions of the agent's belief in the opponent's policy throughout the course of an episode for the relevant methods.

Likelihood	Prior	Opponent 0	Opponent 1	Opponent 2
Freq	Unif	0.67 ± 0.16	0.61 ± 0.10	0.63 ± 0.06
Freq	σ_{-i}	0.73 ± 0.09	0.69 ± 0.12	0.78 ± 0.05
OPC	Unif	0.89 ± 0.03	0.89 ± 0.02	0.90 ± 0.03
OPC	σ_{-i}	0.92 ± 0.04	0.93 ± 0.03	0.91 ± 0.03

Table 5: **Opponent classification accuracy at end of episode.** Accuracy is calculated over 100 episodes and intervals are calculated from an empirical bootstrap across 5 seeds. Accuracy is only calculated for the final timestep in an episode. Tie-breaking favors the smaller opponent index, resulting in high accuracy for Q-Mixing: Freq.

5.5 Future Research Directions

In this study we consider only a simple learned classifer for the OPC. Instead, more sophisticated methods for reasoning about the opponent's policy offers ample room for future improvements for Q-Mixing. A set of assumptions that can be made includes that *all* players have fixed strategy sets. Under these assumptions, agents could maintain more sophisticated beliefs about their opponents (Zheng et al., 2018), and extend this to recursive-reasoning procedures (Yang et al., 2019). This line of work primarily focuses on other-player policy identification and presents a promising future direction for enhancing the quality of the OPC.

Another potential extension of the OPC is to explore alternative objectives. Rather than solely focusing on predicting the opponent, in safety-critical situations, an agent may want to consider an objective that accounts for inaccurate opponent predictions. The Restricted Nash Response (Johanson et al., 2007) embodies this measure by striking a balance between maximizing performance if the prediction is correct and maintaining reasonable performance if the prediction is inaccurate.

While both of these research directions revolve around opponent-policy prediction, they address different problem statements. Most notably, these works do not consider varying the distribution of opponent policies as we have investigated in this work. As a result, adapting these methods to this distinct problem domain presents a fruitful opportunity for future research.

6. Strategic Transfer in PSRO

Up to this point, we have focused on the issue of transferring responses across opponent strategies. Now, we turn our attention to investigating how these advances may reduce the cumulative cost of learning in game-solving algorithms. PSRO is one such algorithm for learning a solution to a in multiagent systems by interleaving empirical game analysis with Deep RL. At each iteration, Deep RL is invoked to train a best-response to a mixture of opponent policies. The learning problems faced across each iteration share a common structure. This common structure provides us with the opportunity to transfer knowledge acquired in previous iterations to assist in training subsequent policies. When consecutive best-response problems are compared only two changes may occur: inclusion of an additional

policy in each opponent's strategy set, and a change in the distribution with which each opponent samples their policies. In this section, we introduce two variants of PSRO that exploit this common strategic structure to reduce the amount of simulation required during Deep RL training.

A particular challenge for the RL step in PSRO is that the learner must derive a response under uncertainty about opponent policies. The profile derived from the empirical game is generally a mixed-strategy profile, as in strategically complex environments randomization is often a necessary ingredient for equilibrium. The opponent draws from this mixture are unobserved, adding uncertainty to the multiagent environment. We address this challenge through variants of PSRO in which all RL is applied to environments where opponents play pure strategies. The proposed methods employ, but are not limited to, the machinery of *Q-Mixing* to facilitate operations on pure strategies instead of mixed strategies. We propose and evaluate two such methods, which work in qualitatively different ways: ***Mixed-Oracles*** learns separate BRs to each pure strategy in a mixture and combines the results from learning to approximate a BR to the mixture. ***Mixed-Opponents*** constructs a single pure opponent policy that represents an aggregate of the mixed strategy and learns a BR to this policy.

Our methods promise advantages beyond those of learning in a less stochastic environment. Mixed Oracles transfers learning across epochs, exploiting the Q-functions learned against a particular opponent policy in constructing policies for any other epoch where that opponent policy is encountered. Mixed Opponents applies directly over the joint opponent space, and so has the potential to scale beyond two-player games.

6.1 Policy-Space Response Oracles

PSRO solves a game by constructing an approximate model of the game defined over a restricted strategy set. The game model is solved analytically and its solution is used as an approximate solution to the full game. To build the game model, each player begins with a strategy-set Π_i^0 that contains either a set of existing policies, or contains only a uniform random policy. An ENFG $\tilde{U}^{\Pi^0}$ can be constructed through simulating each strategic profile and entering total payoffs in the respective cell.

PSRO now iteratively improves the quality of the game model, through the expansion of the players' strategy sets, until it captures an approximate solution to the full game. At each epoch e of the PSRO algorithm a new policy is constructed for each player by best-responding to an opponent profile $\sigma_{-i}^{*,e-1}$ from the currently constructed ENFG: $\pi_i^e \in \text{BR}(\sigma_{-i}^{*,e-1})$. These policies are then added to each player's strategy set, $\Pi_i^e \leftarrow \Pi_i^{e-1} \cup \{\pi_i^e\}$, and the new profiles are simulated to expand the ENFG. Algorithm 2 presents the full PSRO algorithm as defined by Lanctot et al. (2017).

One of the key design choices in iterative empirical game-solving is choosing which policies to add to the ENFG. This was first studied by Schvartzman and Wellman (2009a), who termed it the ***strategy exploration problem***. Figure 14 depicts where strategy exploration fits into the PSRO algorithm. We want to add policies that both bring the solution to the ENFG closer to the solution of the full game and that can be calculated efficiently. In PSRO, the strategy exploration problem is decomposed into two steps: empirical-game solving and BR via RL. In the empirical-game solving step, PSRO derives a profile $\sigma^{*,e}$

Algorithm 2: Policy-Space Response Oracles (Lanctot et al., 2017)

Input: Initial policy sets for all players Π^0
Simulate utilities $\tilde{U}^{\Pi^0}$ for each joint $\pi^0 \in \Pi^0$
Initialize solution $\sigma_i^{*,0} = \text{Uniform}(\Pi_i^0)$
while *epoch e in* $\{1, 2, \dots\}$ **do**
  for *player* $i \in [[n]]$ **do**
    for *many episodes* **do**
      $\pi_{-i} \sim \sigma_{-i}^{*,e-1}$
      Train π_i^e over $\tau \sim (\pi_i^e, \pi_{-i})$
    $\Pi_i^e = \Pi_i^{e-1} \cup \{\pi_i^e\}$
  Simulate missing entries in $\tilde{U}^{\Pi^e}$ from Π^e
  Compute a solution $\sigma^{*,e}$ from $\tilde{\Gamma}^e$
Output: Current solution $\sigma_i^{*,e}$ for player i

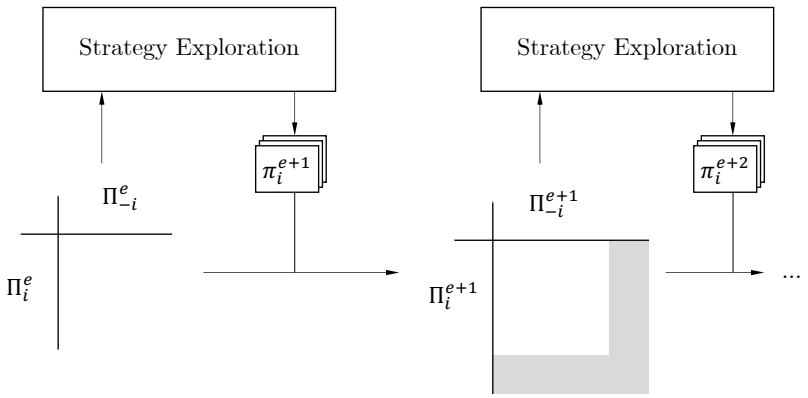


Figure 14: **Overview of the Policy-Space Response Oracles (PSRO) algorithm.** PSRO constructs an empirical game that models a complex underlying game by iteratively performing strategy exploration and profile simulation.

from the current empirical game. The method for choosing this profile is the MSS, formally a function from empirical games to solution profiles MSS : $\tilde{\Gamma}^e \to \sigma^{*,e}$. For example, the MSS might compute a Nash equilibrium of $\tilde{\Gamma}^e$.[4] In the BR step, PSRO generates a new policy via RL (the *response oracle*), training against the target opponent profile computed by the MSS. The choice of MSS and response oracle algorithm constitutes a strategy exploration approach and determines the convergence speed of PSRO. Figure 15 zooms in on the strategy exploration portion of PSRO, completing the illustration of Figure 14.

We observe and address two opportunities for improvement within the standard version of PSRO (Algorithm 2). First, note that the only thing that changes from one epoch to the

4. As noted above, a variety of other MSSs have been proposed, and assessing their relative merits is a topic of active research (Balduzzi et al., 2019; Wang et al., 2022). In experiments reported here, we adopt NE as the baseline MSS; however, our methods readily apply to any MSS.

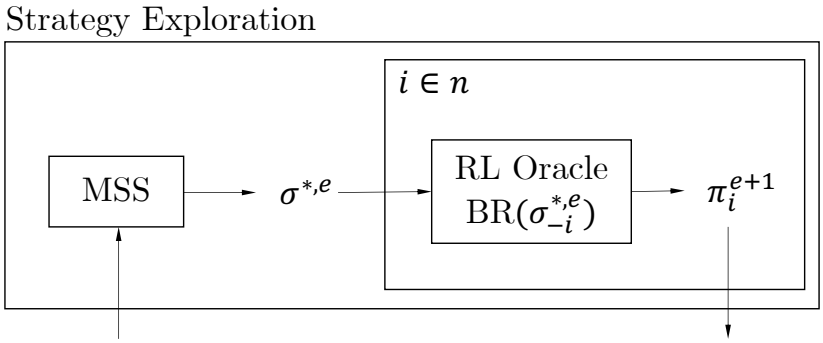


Figure 15: **Strategy exploration in PSRO.** An MSS computes a solution $\sigma^{*,e}$ to the empirical game, which defines the response target. The approximate best-response oracle applies deep RL with opponents fixed at $\sigma^{*,e}_{-i}$ to generate a new policy π_i^{e+1} for each player.

next is the opponent strategy profile σ_{-i}; the transition dynamics remain the same, and the mixed profile σ_{-i} itself likely contains policies encountered in previous epochs. This suggests a significant opportunity to transfer learning across epochs; however, the BR calculation step works by training anew at each epoch. Furthermore, because it is responding to some of the same strategies, the new policy learnt may be similar to the strategies added in previous epochs, and may not be the most useful addition to the ENFG. Secondly, the opponent profiles are mixtures (i.e., distribution over opponent policies). This makes the environment dynamics more stochastic from the perspective of the RL agent, making learning more difficult. In Section 6.2 we propose an algorithm that transfers knowledge between iterations, and only trains against the single new opponent policy. In Section 6.3 we present a second algorithm that avoids responding to similar opponent strategies on subsequent iterations, while also addressing the opponent variance issue and providing scalability to multiple other agents.

6.2 Mixed-Oracles

The first problem we address is that during BR calculation there is an opportunity for transferring previously learned information. In each epoch, each player learns a BR to a mixed profile of opponent policies. This mixture typically involves the newly added policies (one per player) for this epoch, but may also include many policies from previous epochs. Training in previous epochs already captured experience against those policies, so including them in further training may be redundant.

The *Mixed-Oracles* algorithm is a variant of PSRO for two-player games, with a modified BR oracle designed to transfer learning across epochs. This method works by learning and maintaining a collection of BRs to each opponent policy $\Lambda_i^e = \left\{\lambda_i^1, \ldots, \lambda_i^e\right\}$, where λ_i^e is the BR to π_{-i}^{e-1}. During each epoch of Mixed-Oracles, a BR is learned for the single new opponent policy, rather than for the mixed opponent-profile generated by the MSS. A BR to the MSS-generated target mixture is then *constructed* from the collection of BR results for constituent policies in the mixture. Constructing the new policy is done through a general *TransferOracle* function that maps a set of policies and a distribution over the policies into

a single policy

$$\text{TransferOracle} : \Pi \times \Delta(\Pi) \to \pi. \tag{18}$$

The resulting policy should approximately aggregate the behavior of the component policies.

By reusing learned behaviors from previous epochs, Mixed-Oracles allows us to focus training exclusively on new opponent policies. The key design choice is how to combine knowledge from the BRs to individual policies into a BR to any distribution of said policies. We provide a general description of Mixed-Oracles, where the TransferOracle method is abstract, as Algorithm 3.

Section 4 introduces *Q-Mixing* as an approach for constructing policies against any mixture of opponent strategies. This method utilizes Q-values learned against each individual opponent strategy, making it well-suited for supporting the desired transfer. As we observed in Section 3, direct policy transfer is not always feasible. Consequently, Q-Mixing's use of values makes it particularly appropriate for aiding Mixed-Oracles. Specifically, Q-Mixing calculates the average Q-values learned against each opponent policy π_{-i}, weighted by their likelihood in the opponent mixture σ_{-i}:

$$Q_i(o_i, a_i \mid \sigma_{-i}) = \sum_{\pi_{-i}} \psi_i(\pi_{-i} \mid o_i, \sigma_{-i}) Q_i(o_i, a_i \mid \pi_{-i}), \tag{19}$$

where ψ determines the relative likelihood of playing an opponent $\psi_i : \mathcal{O}_i \to \Delta(\Pi_{-i})$. In this study, we use Q-Mixing as our TransferOracle, where ψ is the prior over the opponent distribution as given by an MSS.

Algorithm 3: Mixed-Oracles

Input: Initial policy sets for all players Π^0
Simulate utilities $\tilde{U}^{\Pi^0}$ for each joint $\pi \in \Pi^0$
Initialize solutions $\sigma_i^{*,0} = \text{Uniform}(\Pi_i^0)$
Initialize pure-strategy BRs $\Lambda_i^0 = \emptyset$
while *epoch e in* $\{1, 2, \ldots\}$ **do**
  # Best respond to each new opponent.
  for *player* $i \in [[n]]$ **do**
    for *many episodes* **do**
      Train λ_i^e over $\tau \sim (\lambda_i^e, \pi_{-i}^{e-1})$
    $\Lambda_i^e = \Lambda_i^{e-1} \cup \{\lambda_i^e\}$
  # Generate new policies.
  for *player* $i \in [[n]]$ **do**
    $\pi_i^e \leftarrow \text{TransferOracle}(\Lambda_i^e, \sigma_{-i}^{*,e-1})$
    $\Pi_i^e = \Pi_i^{e-1} \cup \{\pi_i^e\}$
  Simulate missing entries in $\tilde{U}^{\Pi^e}$ from Π^e
  Compute a solution $\sigma^{*,e}$ from $\tilde{\Gamma}^e$
Output: Current solution $\sigma_i^{*,e}$ for player i.

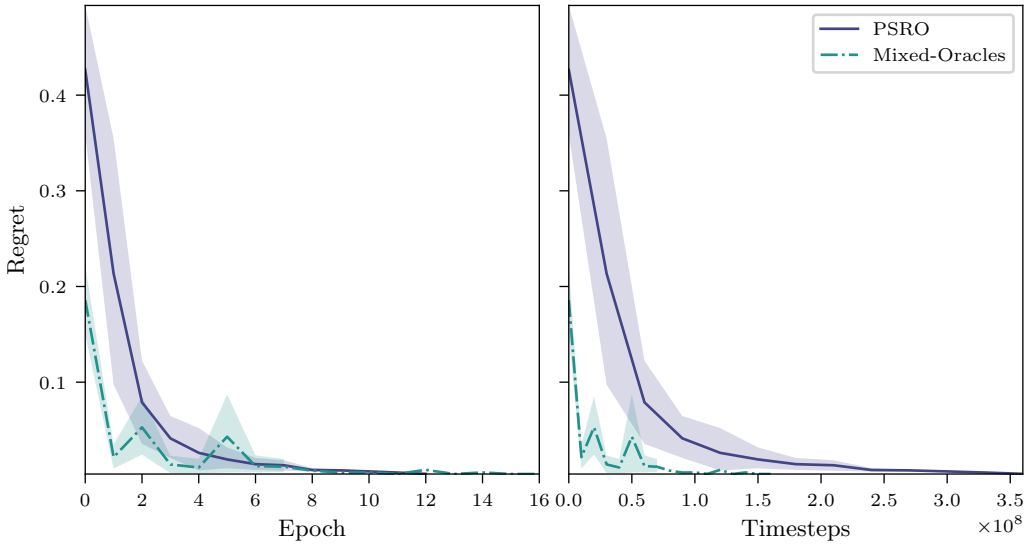


Figure 16: **Mixed-Oracles on the RWS game.**

6.2.1 Empirical Convergence of Mixed-Oracles

Does Mixed-Oracles yield a solution of similar quality to PSRO while using fewer simulation timesteps? We evaluate this question by comparing both methods cumulative simulation timesteps usage during game solving. We compare the methods on two distinct games: RWS and Gathering (Leibo et al., 2021)[5].

RWS, a competitive two-player game, is employed to evaluate the transfer learning technology. In RWS, a trivial Nash equilibrium can be discovered without learning: all players collect no items. When all players opt for this strategy, they play the Nash equilibrium with equal weight assigned to rock, paper, and scissors. To uncover a non-trivial equilibrium, we initialize the strategy set for all players to include three policies, each specializing in collecting a particular item. This modification is applied to all algorithms in subsequent RWS experiments throughout this work.

Gathering, on the other hand, presents several complementary characteristics: it is a commons game (neither strictly competitive nor cooperative), can accommodate any number of players, and is also a grid-world game. In Gathering, agents compete to harvest apples from an orchard without over-harvesting collectively. Over-harvesting inhibits future apple regrowth, leading to a collectively worse outcome. The Gathering game is implemented as a grid-world game where agents can only see a small area in front of them. For tractability, this work considers categorical observations of occupants of each cell in the gridworld, instead of RGB observations. Agents can move in cardinal directions, rotate in either direction, take no action, or "tag" a player in front of them. Tagging another player temporarily removes them from the game. Agents receive a reward for picking an apple, and apples regrow at a rate proportional to the number of nearby unharvested apples.

5. For extended details on both games, see Appendix F.

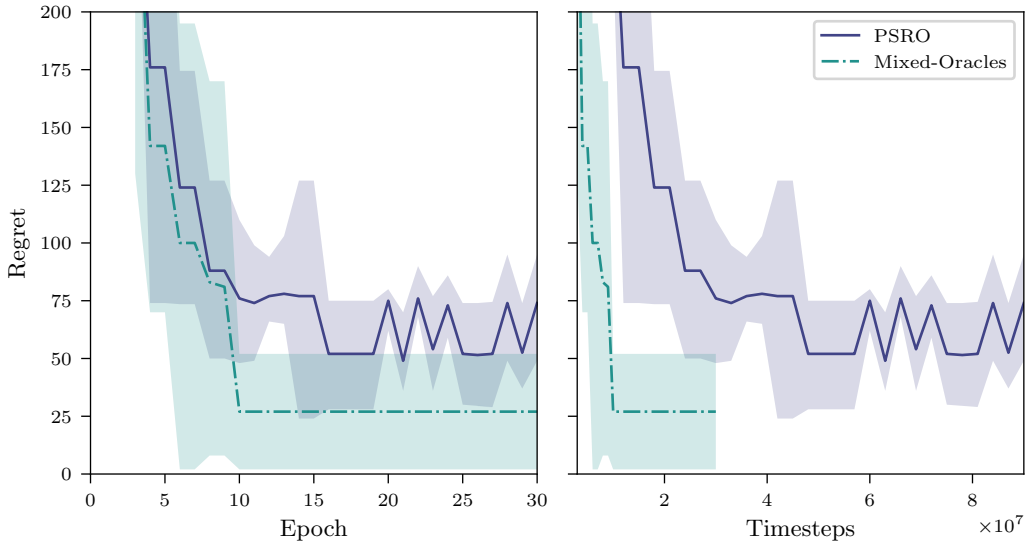


Figure 17: **Mixed-Oracles on the Gathering game.**

Figure 16 compares convergence speed, measured by their regret over time, of Mixed-Oracles and PSRO on the RWS game. Both Mixed-Oracles and PSRO converge to an equilibrium within the budgeted 3.5×10^8 timesteps. However, at 1.5×10^8 timesteps Mixed-Oracles converged to an equilibrium; whereas, PSRO has not. Mixed-Oracles converges in a similar number of epochs as PSRO seen on the left plot with both algorithms converging around six. However, Mixed-Oracles achieves this solution while requiring less usage of the environment simulator (measured in timesteps on the right plot). It is worth recalling here that both PSRO and Mixed-Oracles in this experiment is initialized with a strategy-set containing the three specialized policies (rock, paper, and scissors). This means that both algorithms contain many equilibria before performing any strategy exploration. Therefore, we turn towards investigating other games where the strategy exploration step of PSRO has a greater influence on the algorithm's performance.

Figure 17 similarly compares both algorithms on the Gathering game with two-players and the small map. At epoch 10, Mixed-Oracles has converged to a solution with roughly 25 regret. At the same time, PSRO approximately triple the regret. By the end of PSRO's runtime it improves its performance to roughly 50 regret, double that of Mixed-Oracles. The reduction can be regret may be contributed to by two factors (a) reduction cost of each epoch (measured in simulated timesteps) allows more epochs to be run, and/or (b) noise introduced through Q-Mixing's approximation helps in exploring the game's strategy space. In this game, both algorithms appear to converge roughly around 10 epochs. This suggests that the later factor, noise-induced exploration, may have played a larger role in the overall algorithm's performance improvements.

6.3 Mixed-Opponents

We next examine PSRO's strategy exploration method that is defined by BR to the opponent profile generated by an MSS. This design choice was motivated by solving for Nash Equilibrium, where failure to add a beneficial deviation to the restricted strategy set indicated convergence. However, in an extensive form game it can take infeasibly many iterations to achieve convergence, meaning that in practice the theoretical convergence in the limit may not be helpful.

In other words, BR within a single iteration serves the short-term goal of checking for convergence presently, but may not serve the long-term goal of building a rich empirical game. This distinction is akin to the exploration-exploitation dilemma: we may solve a game faster by choosing not to respond to the current Nash equilibrium, if it leads to strategies that are useful in improving our empirical game. Exploration-focused objectives promise to reduce the number of iterations of PSRO, instead of reduce the number of timesteps of a single RL application.

In this section, we introduce Mixed-Opponents, a variant of PSRO that incorporates an exploration-focused objective. The key insight behind this objective is that each opponent policy is greedy, which leads to the suppression of potentially useful information already learned about non-greedy actions. For instance, many opponent policies may agree on a second-best action that is never played. We propose Mixed-Opponents, which employs the transfer learning methods developed in this work to help us explore this direction. Similar to Mixed-Oracles, Mixed-Opponents maintains the advantage of responding to specific opponent policies, resulting in less stochastic learning signals and cheaper response learning. To begin, we will present Mixed-Opponents through a constructed example.

6.3.1 Constructed Example

In this example, we consider a hypothetical run of PSRO to solve Rock-Paper-Scissors (RPS) with Nash equilibrium as its solution concept. We will focus on Player 1's learning, so will assume that Player 2 will learn exact Q-values against Player 1's meta-strategy. However we will consider Player 1 to only produce approximate best responses. This is to model approximate reinforcement learning such as observed in experiments on the abstracted RPS game, running-with-scissors For simplicity, we will consider each player adding a strategy to the empirical game in turn, rather than simultaneously.

The below table shows a possible outcome of two iterations of PSRO.

R P S	R P S
$\pi_1^1 = (0.0, 0.3, 0.7)$	$Q_2^1 = (0.4, -0.7, 0.3)$
$\pi_1^2 = (0.4, 0.6, 0.0)$	$Q_2^2 = (-0.6, 0.4, 0.2)$

The run begins by Player 1 playing an arbitrary initial strategy π_1^1, which consists of mostly playing S. This play induced a high value for Player 2's R action, denoted in Q_2^1. In turn, Player 1 approximately best-responds to R, by playing mostly P, and never playing S, in π_1^2. Player 1's meta-strategy now plays pure π_1^2, against which both P and S score well, as shown by the values of Q_2^2.

These policies construct an empirical game, which closely resembles the matching pennies game, with the following payoffs:

		Player 2	
		π_2^1	π_2^2
Player 1	π_1^1	$-0.4, 0.4$	$0.7, -0.7$
	π_1^2	$0.6, -0.6$	$-0.4, 0.4$

This game is then solved by the MSS resulting in the following solution:

$$\sigma_1^2 = (0.47\,\pi_1^1,\ 0.52\,\pi_1^2)$$
$$\sigma_2^2 = (0.52\,\pi_2^1,\ 0.48\,\pi_2^2).$$

Because Player 2's two policies play only R and P respectively, we can rewrite Player 2's meta-strategy σ_2^2 in terms of the primitive actions in the full game $(0.52\,\mathrm{R},\ 0.48\,\mathrm{P},\ 0.0\,\mathrm{S})$. If, as in PSRO, Player 1 were then to add their ABR to this meta-strategy, which is P, then their strategy set does not contain the Nash equilibrium of the game. The problem is that there are already strategies in the empirical game that can defeat both of Player 2's strategies, so the new strategy is not very useful. This can be detected by inspecting Player 2's Q-functions: Q_2^1 has a very low valuation of P, and Q_2^2 has a very low valuation of R. This phenomena is illustrated in Figure 18a. Player 1 could instead respond to S, which is moderately highly valued by both opponent Q functions, as it is more relevant. To detect this we can mix the opponent strategies through their action-value estimates, rather than their action distributions. This results in the following Q-values: $Q_2^{\text{Mix}} = (0.46, 0.414, 0.626)$. The ABR to this is R, which more usefully extends Player 1's strategy set to include the Nash equilibrium of this game. The two different approaches are illustrated in Figure 18b.

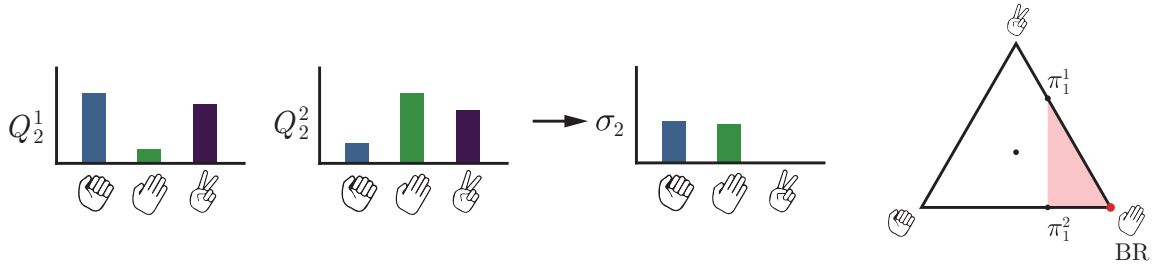
6.3.2 Mixed-Opponents

This example motivates our second algorithm: Mixed-Opponents. This method also employs a combination method, but instead of combining results from training against previously encountered opponents, it combines the strategies of the opponent mixture themselves to construct a single new opponent policy as a target for training. We refer to the method for generating a new opponent policy from a mixture of opponents the OpponentOracle, and it has the same functional form as the TransferOracle. The generalized Mixed-Opponent algorithm is shown in Algorithm 4. We employ Q-Mixing (Equation 19) as our OpponentOracle. In contrast to Mixed-Oracles, which uses Q-Mixing to transfer Q-values across epochs, here we apply it to average Q-values to define a variant training objective.

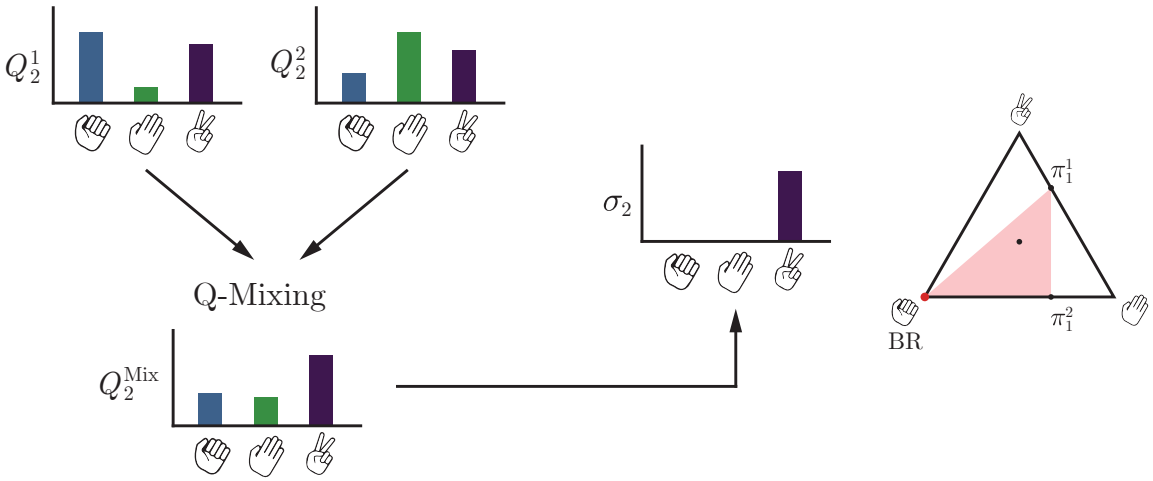
6.3.3 Empirical Convergence of Mixed-Opponents

The first research question we address is: does Mixed-Opponents lead to a solution of similar quality compared to PSRO while utilizing fewer simulation timesteps? To answer this question, we follow the same experimental procedure used for the Mixed-Oracles experiment in the previous section.

Figure 20 compares the convergence speed of both algorithms on the Gathering-Open game. After 25 epochs, Mixed-Opponents has discovered a solution with approximately 50% less regret. However when both algorithms are compared via timesteps, at 5×10^7



(a) PSRO with Nash equilibrium as a solution concept. The two opponents' Q-functions inform greedy policies that each play rock and paper, respectively. When the meta-strategy for Player 2 is $\sigma_2 = (0.52\,\pi_2^1,\ 0.48\,\pi_2^2)$, Player 1 will add paper as a BR.



(b) Mixed-Opponents first mixes the opponent policies by their Q-values (in this example using the Q-Mixing algorithm). The BR to the mixed opponent is rock, and its inclusion expands the strategy space to include the Nash equilibrium of RPS (middle dot).

Figure 18: **Empirical game expansion resulting from different strategy exploration methods.**

Algorithm 4: Mixed-Opponents

Input: Initial policies for all players Π^0
Simulate $\tilde{U}^{\Pi^0}$ for each joint $\pi \in \Pi^0$
Initialize solutions $\sigma_i^{*,0} = \text{Uniform}(\Pi_i^0)$
while *epoch e in* $\{1, 2, \ldots\}$ **do**
  for *player* $i \in [[n]]$ **do**
    $\pi_{-i} \leftarrow \text{OpponentOracle}(\Pi_{-i}^{e-1}, \sigma_{-i}^{*,e-1})$
    for *many episodes* **do**
      Train π_i^e over $\tau \sim (\pi_i^e, \pi_{-i})$
    $\Pi_i^e = \Pi_i^{e-1} \cup \{\pi_i^e\}$
  Simulate missing entries in $\tilde{U}^{\Pi^e}$
  Compute a solution $\sigma^{*,e}$ from $\tilde{\Gamma}^e$
Output: Solution $\sigma_i^{*,e}$ for player i.

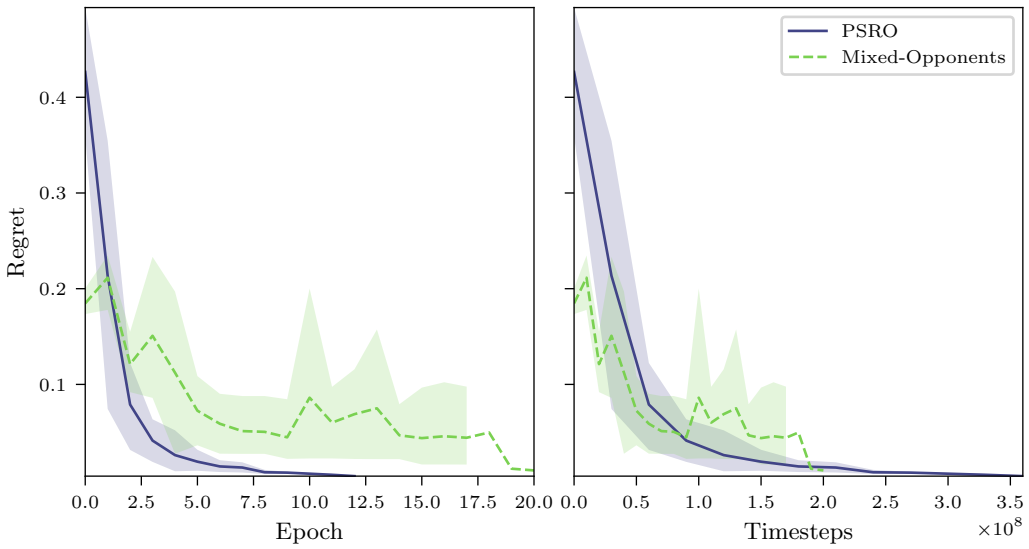


Figure 19: **Mixed-Opponents on the RWS game.**

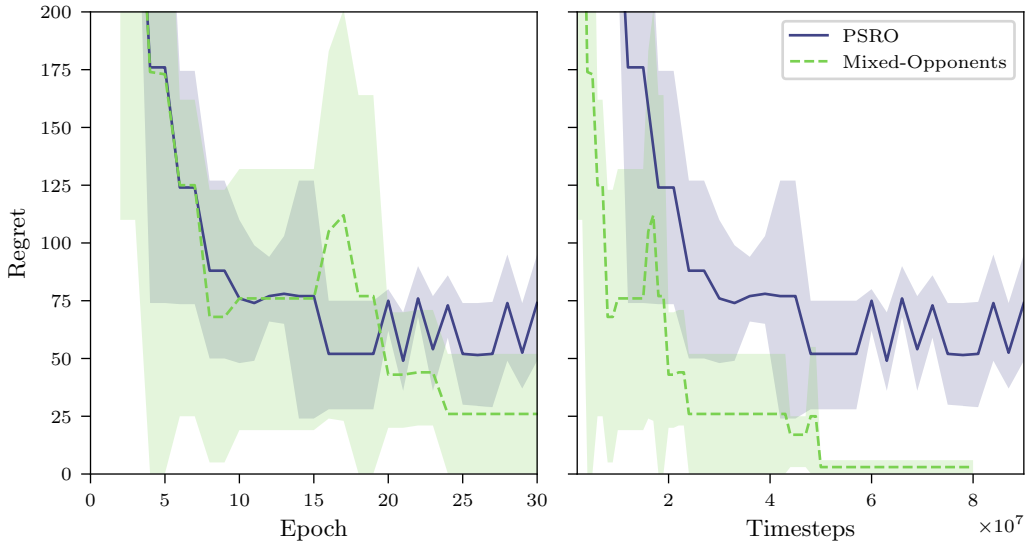


Figure 20: **Mixed-Opponents on the Gathering game.**

timesteps, Mixed-Opponents has found a nearly no regret solution; whereas, PSRO has approximately 50 regret.

Mixed-Opponent's performance on the RWS game is shown in Figure 19. In this result, we can see that PSRO behaves as expected: decreasing regret through epochs and converging to a low-regret solution. On the other hand, Mixed-Opponents, behaves erratically and does not show convergence. What appears as a complete failure of the method, provides the practitioner some valuable insights into the utility gained through differing strategy exploration methods. Recall from the discussion on Mixed-Oracles applied to RWS that Mixed-Oracles was initialized to contain several equilibria. The same consideration also applies to Mixed-Opponents; meaning that Mixed-Opponents need not explore the strategy space of the game to discover equilibria. This means that we are evaluating Mixed-Opponents in a setting which contradicts its motivation. Naturally, any gains Mixed-Opponents offers by discovering new strategies would not be advantageous. This also highlights a downside of Mixed-Opponents: it may fail to *exploit* the discovered strategy space. Instead of discovering new strategies, this setting requires exploiting the discovered strategy space to compute an accurate equilibrium response policy. We discuss this in more detail in Section 6.5.

6.3.4 Many-Player Games

An advantage of Mixed-Opponents over Mixed-Oracles is that it can be applied to games with more than two players. It is natural to then ask if the trends we observed in the previous experiment, on two-player games, extend to many-player games? We repeat the previous Mixed-Opponent analysis on the Gathering game; however, we will now look a three-player version of the game on the "open" map (maps define spawn points and orchard configurations). We limit each profile in the empirical game to three simulations, to handle

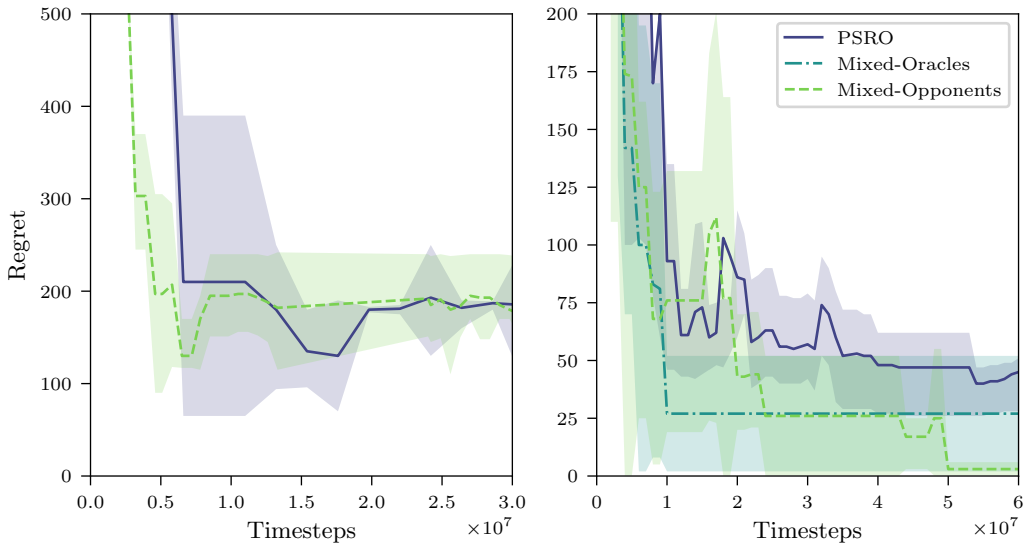


Figure 21: **Extensions beyond two players and using shared hyperparameters.** (Left) Mixed-Opponents evaluated on the Gathering-Open game with 3 players. (Right) Comparison of algorithms when set to the same Deep RL hyperparameters.

the combinatorial explosion of profiles. Figure 21 shows our results where Mixed-Opponents finds a similar quality solution to PSRO in half of the time.

6.4 Hyperparameter Selection Ablation

In the previous experiments PSRO used a separate set of hyperparameters from the proposed algorithms. These two sets of hyperparameters were specialized for low- and high-variance outcomes of state induced by facing pure- and mixed-strategy opponents respectively. This was motivated by the assumption that lower variance would require less training. This raises the question: does the differing hyperparameters explain the performance gap between the algorithms?

In this section, we question that assumption and ask: do Mixed-Oracles and Mixed-Opponents perform at least as well as PSRO when given the same Deep RL hyperparameters? To answer this question we run all three algorithms with the same set of hyperparameters, forcing all to adopt the same simulation budget.

We report results for the Gathering-Small game in Figure 21. The trends observed previously reoccur: Mixed-Oracles and Mixed-Opponents find solutions at least as good as PSRO after 6×10^7 timesteps. Moreover, by 2.5×10^7 timesteps both Mixed-Oracles and Mixed-Opponents have converged to a regret of approximately 25, while PSRO has a regret of roughly 50. These results suggest that our hyperparameter selection methodology does not explain the results from the preceding experiments.

6.5 Strategy Exploration-Exploitation Dilemma

DO uses response to Nash as a way to guarantee theoretical convergence in the limit. Specifically, if a new best-response strategy cannot be constructed to add to the empirical game for any player then a Nash solution is found. PSRO with Nash as a solution adopts this guarantee by inheriting the same algorithmic structure as DO. However, unlike DO, PSRO is applied to games where we cannot reasonably run the algorithm long enough to realize convergence. Instead of guaranteeing that each new strategy serves the additional role of a convergence check, we can also choose to select strategies that reduce the time till convergence. This trade-off is analogous to the exploration-exploitation dilemma in single-agent RL, where now it may be advantageous to first *explore* and add diverse strategies, then *exploit* and attempt to solve the game.

Rectified Nash PSRO (Balduzzi et al., 2019) is an example of a exploration method that encourages agents to "amplify their strengths and ignore their weaknesses." This leads to the inclusion of generally weak agents that have diverse niches. The rectified Nash objective focuses on increasing the effective diversity of the population, and does not directly optimize towards solving the game. While response to the Nash PSRO is a an exploitative objective, where it assumes that all strategic cycles in the game are included in the empirical game.

We posit that Mixed-Oracles is similarly an exploitative objective, and that Mixed-Opponents is an exploration objective. Mixed-Oracles includes responses to the MSS's solution but introduces additional approximation errors by purifying pure-strategy best-response policies. Mixed-Opponents, on the other hand, attempts to add a more diverse policy to the population; following the intuition that non-optimal actions may contain interesting strategic dimensions in aggregate. An open question is characterizing how Mixed-Opponents impacts the quality of the empirical game. No algorithm so far acts as a panacea alone; however, mixing the insights gained from all of them together may offer a path towards the remedy.

7. Conclusion

We investigated how transferring knowledge about previously encountered opponents can generalize response knowledge across opponent strategies and improve the efficiency of game-solving algorithms. The story began by introducing a class of problems and characterizing them as strategic knowledge transfer problems. These problems address the use of strategic knowledge accrued while learning response policies in one context toward deriving a response policy for a new strategic context.

We investigate one such problem: the opponent mixture transfer problem. In this problem, we assume responses to each opponent policy and access to the strategic mixture that the opponent is playing. First, we show how a general solution to this problem cannot be constructed without making additional assumptions about the policy's implementation. Then, we introduce *Q-Mixing*, an algorithm that solves the problem under the assumption that all response policies are value-based. Q-Mixing transfers response knowledge across any distribution of known opponents by appropriately weighting the responses' Q-values. We introduced exact methods for Q-Mixing, as well as approximate versions which we empirically demonstrate offer more practical solutions.

Key to the success of Q-Mixing, and solutions to the opponent mixture transfer problem in general, is the maintenance of a belief in the opponent's policy. Belief in the opponent's identity informs the select a suitable response behavior. We performed an in-depth analysis to tease apart what factors contribute to successful belief maintenance. In the games we tested, we saw that a opponent-policy classifier, trained using the replay buffers from the pure strategy response policies, served as an effective opponent-policy likelihood model. Moreover, the belief may be greatly improved by maintaining a posterior likelihood that is repeatedly updated at each observation.

Finally, we turned to the use of transfer learning to reduce the computational cost of iterative game-solving algorithms. We introduced two algorithms that share a common theme of modifying the best-response objective from responding to mixed-strategy opponents to responding to pure-strategy opponents. Responding to a pure strategy rather than a mixture eliminates the opponent sampling process, and thus reduces the variance in experiences during training.

The first algorithm, Mixed-Oracles, trains response policies to each policy in the population. A response-policy to a mixed strategy is then constructed by combining the individual pure-strategy response policies, using Q-Mixing. The second algorithm, Mixed-Opponents, transforms the mixed-strategy response target for PSRO into a pure strategy representing the mixture. It does so by combining the Q-values[6] of the opponent policies supported in the original target mixture, generating a novel policy that captures elements of the previous opponents. Both algorithms reuse strategic knowledge from previous PSRO iterations: the Q-values derived in training BRs. This reuse saves cumulative training time in PSRO, as does the variance reduction associated with responding to pure strategies noted above.

Mixed-Opponents also highlights the potential for novel *strategy discovery* as part of a *strategy exploration* approach in PSRO. As in single-agent RL, introduction of new strategy candidates for game solving must balance consideration of diverse possibilities (exploration) with fine-tuning of known effective solutions (exploitation).

These algorithms demonstrate that contributions to *strategic knowledge transfer* and *strategy discovery* can reduce the computational cost present in iterative game-solving algorithms. Continued interest and advancements in these areas will bring us closer to solving complex games that represent real problems we face in the world today.

Acknowledgments

We thank Mohamed El Banani and Richard Ely Locke Higgins for thoughtful discussions throughout this project.

Work at the University of Michigan on this project was supported in part by funding from the US Army Research Office (MURI grant W911NF-18-1-0208), and a grant from the Effective Altruism Foundation.

6. Mixed-Opponents, like Mixed-Oracles uses Q-Mixing for the combination. Other approaches are possible, and should be investigated in future research.

References

Haitham Bou Ammar, Eric Eaton, José Marcio Luna, and Paul Ruvolo. Autonomous cross-domain knowledge transfer in lifelong policy gradient reinforcement learning. In *24th International Conference on Artificial Intelligence*, IJCAI, pages 3345–3349, 2015.

Oron Anschel, Nir Baram, and Nahum Shimkin. Averaged-DQN: variance reduction and stabilization for deep reinforcement learning. In *34th International Conference on Machine Learning*, ICML, pages 176–185, 2017.

David Balduzzi, Marta Garnelo, Yoram Bachrach, Wojciech M. Czarnecki, Julien Pérolat, Max Jaderberg, and Thore Graepel. Open-ended learning in symmetric zero-sum games. In *36th International Conference on Machine Learning*, ICML, 2019.

Bikramjit Banerjee and Peter Stone. General game learning using knowledge transfer. In *20th International Joint Conference on Artificial Intelligence*, IJCAI, pages 672–677, 2007.

Nolan Bard, Michael Johanson, Neil Burch, and Michael Bowling. Online implicit agent modelling. In *12th International Conference on Autonomous Agents and Multiagent Systems*, AAMAS, pages 255–262, 2013.

Nolan Bard, Jakob N. Foerster, Sarath Chandar, Neil Burch, Marc Lanctot, H. Francis Song, Emilio Parisotto, Vincent Dumoulin, Subhodeep Moitra, Edward Hughes, Iain Dunning, Shibl Mourad, Hugo Larochelle, Marc G. Bellemare, and Michael Bowling. The Hanabi challenge: A new frontier for AI research. *Artificial Intelligence*, 280, 2020.

Samuel Barrett and Peter Stone. Cooperating with unknown teammates in complex domains: A robot soccer case study of ad hoc teamwork. In *29th AAAI Conference on Artificial Intelligence*, AAAI, pages 2010–2016, 2015.

Avrim Blum and Yishay Mansour. Learning, regret minimization, and equilibria. In Noam Nisan, Tim Roughgarden, Éva Tardos, and Vijay V. Vazirani, editors, *Algorithmic Game Theory*, pages 79–101. Cambridge University Press, 2007.

George W. Brown. Iterative solution of games by fictitious play. In T. C. Koopmans, editor, *Activity Analysis of Production and Allocation*, New York, 1951. Wiley.

Rich Caruana. Multitask learning. *Machine Learing*, 28(1):41–75, 1997.

Caroline Claus and Craig Boutilier. The dynamics of reinforcement learning in cooperative multiagent systems. In *15th National Conference on Artificial Intelligence*, AAAI, pages 746–752, 1998.

Jacob W. Crandall. Just add pepper: Extending learning algorithms for repeated matrix games to repeated Markov games. In *11th International Conference on Autonomous Agents and Multiagent Systems*, AAMAS, pages 399–406, 2012.

Wojciech Czarnecki, Siddhant Jayakumar, Max Jaderberg, Leonard Hasenclever, Yee Whye Teh, Nicolas Heess, Simon Osindero, and Razvan Pascanu. Mix & match agent curricula for reinforcement learning. In *35th International Conference on Machine Learning*, ICML, pages 1087–1095, 2018.

Jakob Foerster, Nantas Nardelli, Gregory Farquhar, Triantafyllos Afouras, Philip H. S. Torr, Pushmeet Kohli, and Shimon Whiteson. Stabilising experience replay for deep multi-agent reinforcement learning. In *34th International Conference on Machine Learning*, ICML, pages 1146–1155, 2017.

Jakob Foerster, Richard Y. Chen, Maruan Al-Shedivat, Shimon Whiteson, Pieter Abbeel, and Igor Mordatch. Learning with opponent-learning awareness. In *17th International Conference on Autonomous Agents and Multiagent Systems*, AAMAS, pages 122–130, 2018a.

Jakob N. Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *32nd AAAI Conference on Artificial Intelligence*, AAAI, pages 2974–2982, 2018b.

David Foster and Peter Dayan. Structure in the space of value functions. *Machine Learning*, 49(2):325–346, 2002.

Amy Greenwald and Keith Hall. Correlated-Q learning. In *20th International Conference on Machine Learning*, ICML, pages 242–249, 2003.

Carlos Guestrin, Daphne Koller, and Ronald Parr. Multiagent planning with factored MDPs. In *14th International Conference on Neural Information Processing Systems*, NeurIPS, pages 1523–1530, 2001.

Jayesh K. Gupta, Maxim Egorov, and Mykel Kochenderfer. Cooperative multi-agent control using deep reinforcement learning. In Gita Sukthankar and Juan A. Rodriguez-Aguilar, editors, *Autonomous Agents and Multiagent Systems: AAMAS 2017 Workshops, Best Papers*, volume 10642 of *Lecture Notes in Computer Science*, pages 66–83. Springer, 2017.

Sergiu Hart and Andreu Mas-Colell. A simple adaptive procedure leading to correlated equilibrium. *Econometrica*, 68:1127–1150, 2000.

He He, Jordan Boyd-Graber, Kevin Kwok, and Hal Daumé III. Opponent modeling in deep reinforcement learning. In *33rd International Conference on Machine Learning*, ICML, pages 1804–1813, 2016.

Pablo Hernandez-Leal and Michael Kaisers. Learning against sequential opponents in repated stochastic games. In *3rd Multidisciplinary Conference on Reinforcement Learning and Decision Making*, RLDM, 2017a.

Pablo Hernandez-Leal and Michael Kaisers. Towards a fast detection of opponents in repeated stochastic games. In *16th International Conference on Autonomous Agents and Multiagent Systems*, AAMAS, pages 239–257, 2017b.

Sepp Hochrieter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87, 1991.

Marco A. Janssen, Robert Holahan, Allen Lee, and Elinor Ostrom. Lab experiments for the study of social-ecological systems. *Science*, 328(5978):613–617, 2010.

Michael Johanson, Martin Zinkevich, and Michael Bowling. Computing robust counter-strategies. In *30th International Conference on Neural Information Processing Systems*, NeurIPS, 2007.

Nicholas K. Jong and Peter Stone. State abstraction discovery from irrelevant state variables. In *19th International Joint Conference on Artificial Intelligence*, IJCAI, pages 752–757, 2005.

Leslie Pack Kaelbling. Learning to achieve goals. In *13th International Joint Conference on Artificial Intelligence*, IJCAI, pages 1094–1099, 1993.

Daphne Koller and Nimrod Megiddo. The complexity of two-person zero-sum games in extensive form. *Games and Economic Behavior*, 4:528–552, 1992.

George Konidaris and Andrew Barto. Autonomous shaping: Knowledge transfer in reinforcement learning. In *23rd International Conference on Machine Learning*, ICML, pages 489–496, 2006.

Landon Kraemer and Bikramjit Banerjee. Multi-agent reinforcement learning as a rehearsal for decentralized planning. *Neurocomputing*, 190:82–94, 2016.

Andrew K. Lampinen and Surya Ganguli. An analytic theory of generalization dynamics and transfer learning in deep linear networks. In *7th International Conference on Learning Representations*, ICLR, 2019.

Marc Lanctot, Vinicius Zambaldi, Audrūnas Gruslys, Angeliki Lazaridou, Karl Tuyls, Julien Pérolat, David Silver, and Thore Graepel. A unified game-theoretic approach to multi-agent reinforcement learning. In *31st International Conference on Neural Information Processing Systems*, NeurIPS, pages 4193–4206, 2017.

Joel Z. Leibo, Edgar Duéñez-Guzmán, Alexander Sasha Vezhnevets, John P. Agapiou, Peter Sunehag, Raphael Koster, Jayd Matyas, Charles Beattie, Igor Mordatch, and Thore Graepel. Scalable evaluation of multi-agent reinforcement learning with melting pot. In *38th International Conference on Machine Learning*, ICML, pages 6187–6199, 2021.

Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In *31st Conference on Neural Information Processing Systems*, NeurIPS, pages 6382–6393, 2017.

Anuj Mahajan, Tabish Rashid, Mikayel Samvelyan, and Shimon Whiteson. Maven: Multi-agent variational exploration. In *33rd Conference on Neural Information Processing Systems*, NeurIPS, 2019.

Luke Marris, Paul Muller, Marc Lanctot, Karl Tuyls, and Thore Graepel. Multi-agent training beyond zero-sum with correlated equilibrium meta-solvers. In *38th International Conference on Machine Learning*, ICML, pages 7480–7491, 2021.

H. Brendan McMahan, Geoffrey J. Gordon, and Avrim Blum. Planning in the presence of cost functions controlled by an adversary. In *20th International Conference on Machine Learning*, ICML, pages 536–543, 2003.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

Paul Muller, Shayegan Omidshafiei, Mark Rowland, Karl Tuyls, Julien Pérolat, Siqi Liu, Daniel Hennes, Luke Marris, Marc Lanctot, Edward Hughes, Zhe Wang, Guy Lever, Nicolas Heess, Thore Graepel, and Remi Munos. A generalized training approach for multiagent learning. In *8th International Conference on Learning Representations*, ICLR, 2020.

Shayegan Omidshafiei, Jason Pazis, Christopher Amato, Jonathan P. How, and John Vian. Deep decentralized multi-task multi-agent reinforcement learning under partial observability. In *34th International Conference on Machine Learning*, ICML, 2017.

Shayegan Omidshafiei, Christos Papadimitriou, Georgios Piliouras, Karl Tuyls, Mark Rowland, Jean-Baptiste Lespiau, Wojciech M. Czarnecki, Marc Lanctot, Julien Pérolat, and Remi Munos. α-rank: Multi-agent evaluation by evolution. *Scientific Reports*, 9, 2019.

Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, Oct 2010.

S. Phelps, M. Marcinkiewicz, and S. Parsons. A novel method for automatic strategy acquisition in N-player non-zero-sum games. In *5th International Joint Conference on Autonomous Agents and Multiagent Systems*, AAMAS, pages 705–712, 2006.

Martin L. Puterman. *Markov Decision Processes*. John Wiley & Sons, Inc., 2005.

Julien Pérolat, Joel Z. Leibo, Vinicius Zambaldi, Charles Beattle, Karl Tuyls, and Thore Graepel. A multi-agent reinforcement learning model of common-pool resource appropriation. In *31st Conference on Neural Information Processing Systems*, NeurIPS, 2017.

Tabish Rashid, Mikayel Samvelyan, Christian Schroeder de Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *35th International Conference on Machine Learning*, ICML, pages 4295–4304, 2018.

Benjamin Rosman, Majd Hawasly, and Subramanian Ramamoorthy. Bayesian policy reuse. *Machine Learning*, 104:99–127, 2016.

Sebastian Ruder. An overview of multi-task learning in deep neural networks. *CoRR*, abs/1706.05098, 2017.

Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *CoRR*, abs/1606.04671, 2016.

L. Julian Schvartzman and Michael P. Wellman. Exploring large strategy spaces in empirical game modeling. In *AAMAS-09 Workshop on Agent-Mediated Electronic Commerce*, 2009a.

L. Julian Schvartzman and Michael P. Wellman. Stronger CDA strategies through empirical game-theoretic analysis and reinforcement learning. In *8th International Conference on Autonomous Agents and Multiagent Systems*, AAMAS, pages 249–256, 2009b.

Jonathan Schwarz, Jelena Luketina, Wojciech M. Czarnecki, Angieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Progress & compress: A scalable framework for contiual learning. In *35th International Conference on Machine Learning*, ICML, 2018.

Felipe Silva and Anna Costa. A survey on transfer learning for multiagent reinforcement learning systems. *Journal of Artificial Intelligence Research*, 64:645–703, 2019.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of Go without human knowledge. *Nature*, 500: 354–359, 2017.

Max Olan Smith, Thomas Anthony, Yongzhao Wang, and Michael P. Wellman. Learning to play against any mixture of opponents. *CoRR*, abs/2009.14180, 2020.

Max Olan Smith, Thomas Anthony, and Michael P. Wellman. Iterative empirical game solving via single policy best response. In *9th International Conference on Learning Representations*, ICLR, 2021.

Matthijs Snel and Shimon Whiteson. Learning potential functions and their representations for multi-task reinforcement learning. In *13th International Conference on Autonomous Agents and Multiagent Systems*, AAMAS, pages 637–681, 2014.

Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z. Leibo, Karl Tuyls, and Thore Graepel. Value-decomposition networks for cooperative multi-agent learning based on team reward. In *17th International Conference on Autonomous Agents and Multiagent Systems*, AAMAS, 2018.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, second edition, 2018.

Ming Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *10th International Conference on Machine Learning*, ICML, pages 330–337, 1993.

Matthew E. Taylor, Peter Stone, and Yaxin Liu. Value functions for RL-based behavior transfer: A comparative study. In *20th National Conference on Artificial Intelligence*, AAAI, pages 880–885, 2005.

Peter D. Taylor and Leo B. Jonker. Evolutionarily stable strategies and game dynamics. *Mathematical Biosciences*, 40:146–156, 1978.

Gerald Tesauro. Extending Q-learning to general adaptive multi-agent systems. In *16th International Conference on Neural Information Processing Systems*, NeurIPS, pages 871–878, 2003.

Karl Tuyls, Julien Perolat, Marc Lanctot, Edward Hughes, Richard Everett, Joel Z. Leibo, Csaba Szepesvári, and Thore Graepel. Bounds and dynamics for empirical game-theoretic analysis. *Autonomous Agents and Multi-Agent Systems*, 34(7), 2020.

Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double Q-learning. In *30th AAAI Conference on Artificial Intelligence*, AAAI, pages 2094–2100, 2016.

Alexander Sasha Vezhnevets, Yuhuai Wu, Remi Leblond, and Joel Z. Leibo. Options as responses: Grounding behavioural hierarchies in multi-agent reinforcement learning. In *37th International Conference on Machine Learning*, ICML, pages 9733–9742, 2020.

Bernhard von Stengel. Efficient computation of behavior strategies. *Games and Economic Behavior*, 14(50):220–246, 1996.

Thomas J. Walsh, Lihong Li, and Michael L. Littman. Transferring state abstractions between MDPs. In *ICML-06 Workshop on Structural Knowledge Transfer for Machine Learning*, 2006.

William Walsh, Rajarshi Das, Gerald Tesauro, and Jeffrey Kephart. Analyzing complex strategic interactions in multi-agent systems. In *AAAI-02 Workshop on Game Theoretic and Decision Theoretic Agents*, 2002.

Yongzhao Wang, Qiurui Ma, and Michael P. Wellman. Evaluating strategy exploration in empirical game-theoretic analysis. In *23th International Conference on Autonomous Agents and Multi-Agent Systems*, AAMAS, pages 1346–1354, 2022.

Yufei Wang, Zheyuan Ryan Shi, Lantao Yu, Yi Wu, Rohit Singh, Lucas Joppa, and Fei Fang. Deep reinforcement learning for green security games with real-time information. In *33rd AAAI Conference on Artificial Intelligence*, AAAI, pages 1401–1408, 2019.

Christopher J.C.H. Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8:279–292, 1992.

Michael P. Wellman. Methods for empirical game-theoretic analysis (extended abstract). In *21st National Conference on Artificial Intelligence*, AAAI, page 1552–1555, 2006.

Michael P. Wellman. Putting the agent in agent-based modeling. *Autonomous Agents and Multi-Agent Systems*, 30:1175–1189, 2016.

Mason Wright, Yongzhao Wang, and Michael P. Wellman. Iterated deep reinforcement learning in games: History-aware training for improved stability. In *20th ACM Conference on Economics and Computation*, EC, pages 617–636, 2019.

Tianpei Yang, Jianye Hao, Zhaopeng Meng, Chongjie Zhang, Yan Zheng, and Ze Zheng. Towards efficient detection and optimal response against sophisticated opponents. In *28th International Joint Conference on Artificial Intelligence*, IJCAI, pages 623–629, 2019.

Yan Zheng, Zhaopeng Meng, Jianye Hao, Zongzhang Zhang, Tianpei Yang, and Changjie Fan. A deep Bayesian policy reuse approach against non-stationary agents. In *31st International Conference on Neural Information Processing Systems*, NeurIPS, 2018.

Appendix A. Direct Policy Transfer

In this Appendix we present an avenue for solving the opponent mixture transfer problem without making any assumptions about the underlying implementation of the response policies. Unfortunately, going down this path requires making strong assumptions that renders it prohibitive outside of contrived games. It is provided here for completeness and to inspire future directions into more fruitful solutions.

A.1 Direct Policy Transfer

The resulting transferred policy should define the same state distribution over the game as $\text{ABR}(\sigma_{-i})$. This property is called ***realization equivalence*** and was introduced to related stochastic policies (i.e., ***behavioral strategies***) and mixed strategies in sequence-form games (von Stengel, 1996; Koller and Megiddo, 1992). Establishing realization equivalence between our transferred policy and a target $\text{ABR}(\sigma_{-i})$ enables us to verify successful transfer algorithms.

Fundamental to constructing $\text{ABR}(\sigma_{-i})$ is that all possible sequences of actions exhibited by this policy must be realizable. When this is the case, the set of response policies (to each opponent policy) can be ***purified***[7] into a single policy representing a response to a mixed strategy opponent. We begin by showing that in this idealized case that response policies can be transferred to solve the opponent mixture transfer problem (Section 3.1). This solution is not free; however, requiring a $\mathcal{O}(\mathcal{SA}\Pi)$ calculation that limits its applicability to large-scale games.

Unfortunately, even in small-scale games, the requirement that all action sequences are realizable is unrealistic in games of practical interest. We provide a didactic example (Section A.1) that illustrates how the transfer operation can fail without this assumption. Beyond the example, in the worst case this assumption may require that all actions have positive support in all states. If we consider that our policies of interest are response policies that are trying to maximize return, then these policies will likely assign most (often, all) of the probability mass to a single best action. Response policies may be modified to meet this assumption; however, modifying a policy may detract from its performance. In an extreme case, the policy modification may result in an policy taking dangerous actions that result in large negative returns.

A.2 Policy Purification

We begin by showing that a set of policies can be purified through their ***visitation frequency***, or its' discounted joint state-action probabilities, to establish realization equivalence. This result is then related to the task of constructing a response to a know distribution of opponents, and limitations to the approach are discussed.

In order to formally define visitation frequencies we must first establish some primitives. Let $d^0 : \mathcal{S} \to [0, 1]$ or $d^0 \in \Delta(\mathcal{S})$ by the initial state distribution. From this we can derive $\Pr^\pi$ as the probability that the random variables, denoted by capital script, take on the assigned values when π is acting in the environment. This allows us to describe many useful

7. The moniker of purification refers to representing a set of policies as a *pure*-strategy (policy).

probabilities:

$$\Pr^{\pi}(S_0 = s) = d^0(s)$$
$$\Pr^{\pi}(A_t = a \mid S_t = s) = \pi(a \mid s)$$
$$\Pr^{\pi}(S_t = s_t, A_t = a_t \mid S_0 = s_0) = \pi(a_t \mid s_t)\Pr^{\pi}(S_t = s_t \mid S_0 = s_0)$$

$$\Pr^{\pi}(S_t = s_t \mid S_0 = s) = \sum_{s_{t-1}} \sum_{a_{t-1}} p(s_t \mid s_{t-1}, a_{t-1})\pi(a_{t-1} \mid s_{t-1})\Pr^{\pi}(S_{t-1} = s_{t-1} \mid S_0 = s_0)$$

It is worth noting explicitly here that this notation takes the ego-centric viewpoint of one agent; where, they view all other agents as part of the environment. From these tools we formally define visitation frequencies in Definition 4.

Definition 4 (Visitation Frequency (Puterman, 2005, §6.9.2))**.** *A visitation frequency is the discounted probability of a policy π occupying either a state $\rho^{\pi} : \mathcal{S} \to [0, 1]$ or joint state-action $\rho_{\pi} : \mathcal{S} \times \mathcal{A} \to [0, 1]$. For the initial state distribution d^0 and the probability of reaching state and/or action s, a after t timesteps and starting in state s^0 as $\Pr^{\pi}\left(S^t = s^t, A^t = a^t \mid S^0 = s^0\right)$ with discount factor γ the visitation frequency is as follows:*

$$\rho^{\pi}(s, a) \doteq \sum_{s^0 \in \mathcal{S}} d^0(s^0) \sum_{t=0}^{\infty} \gamma^n \cdot \Pr^{\pi}\left(S^t = s, A^t = a \mid S^0 = s^0\right), \tag{20}$$
$$\rho^{\pi}(s) \doteq \sum_{a \in \mathcal{A}(s)} \rho^{\pi}(s, a).$$

This quantity is also referred to as the occupancy frequency *or* occupancy *of a policy.*

Through visitation frequencies we can establish a method for purifying a mixed strategy. This is accomplished by weighting the policies by the likelihood of their respective play, within the mixed strategy, in a particular state. This relationship is established below in Theorem 5.

Theorem 5. *The purified policy π^{σ} of a mixed strategy σ is a the convex combination of each policy's action-distribution and the likelihood of that policy for each state:*

$$\forall s \in \mathcal{S}, \quad \forall a \in \mathcal{A}(s), \quad \pi^{\sigma}(a \mid s) = \sum_{\pi \in support(\sigma)} p(\pi \mid s, \sigma) \cdot \pi(a \mid s). \tag{21}$$

Assuming all state-action pairs are reachable by the pure strategies $\pi \in support(\sigma)$.

Proof We begin by establishing the high-level relationship between the purified-policy and its visitation frequency $x^{\sigma}(s, a)$ via simple probability rules:

$$x^{\sigma}(s, a) = p(s \mid \sigma) \cdot \pi^{\sigma}(a \mid s), \tag{22}$$

$$\pi^{\sigma}(a \mid s) = \frac{x^{\sigma}(s, a)}{p(s \mid \sigma)}. \tag{23}$$

The remainder of this proof focuses on removing $p(s \mid \sigma)$ from Equation 23, by unpacking $x^\sigma(s,a)$ to contribute a corresponding $p(s \mid \sigma)$ for reduction. To this end, we first establish the policy-likelihood for later use:

$$\begin{aligned} p(\pi \mid s, \sigma) &= \frac{p(s \mid \pi) \cdot p(\pi \mid \sigma)}{p(s \mid \sigma)} \\ &= \frac{p(s \mid \pi) \cdot \sigma(\pi)}{p(s \mid \sigma)} \\ p(\pi \mid s, \sigma) \cdot p(s \mid \sigma) &= p(s \mid \pi) \cdot \sigma(\pi) \end{aligned} \tag{24}$$

Now, we focus on expanding $x^\sigma(s,a)$ to include $p(s \mid \sigma)$ using Equation 24:

$$\begin{aligned} x^\sigma(s,a) &= \sum_\pi p(\pi, s) \cdot \pi(a \mid s) \\ &= \sum_\pi p(s \mid \pi) \cdot p(\pi) \cdot \pi(a \mid s) \\ &= \sum_\pi p(s \mid \pi) \cdot \sigma(\pi) \cdot \pi(a \mid s) \\ &= \sum_\pi p(\pi \mid s, \sigma) \cdot p(s \mid \sigma) \cdot \pi(a \mid s) \\ &= p(s \mid \sigma) \cdot \sum_\pi p(\pi \mid s, \sigma) \cdot \pi(a \mid s) \end{aligned} \tag{25}$$

Finally, we substitute Equation 25 into Equation 23:

$$\begin{aligned} \pi^\sigma(a \mid s) &= \frac{x^\sigma(s,a)}{p(s \mid \sigma)} \\ &= \frac{p(s \mid \sigma) \cdot \sum_\pi p(\pi \mid s, \sigma) \cdot \pi(a \mid s)}{p(s \mid \sigma)} \\ &= \sum_\pi p(\pi \mid s, \sigma) \cdot \pi(a \mid s) \end{aligned}$$

■

The construction of a mixed strategy best-response policy is a direct application of this operation. The response policies to each respective opponent policies are played in the same distribution that the opponent plays.

There are two limitations with this approach (1) the need for a policy-state likelihood, and (2) that state-action pairs must be reachable by the component policies. Computing a policy-state likelihood requires a $\mathcal{O}(\mathcal{SA}\Pi)$ computation. This cost rivals directly computing a response to the mixed strategy directly, and thus renders the overall approach unavailing. The other limitation of policy-based transfer approaches is that we have assumed that all joint state-actions are reachable. Unfortunately, this assumption is difficult to guarantee outside of simple games. We illustrate this problem with an example in Section A.1. Due to these outstanding issues, we close the door on policy-based transfer methods and leave them for future work; instead turning towards value-based transfer methods.

Appendix B. Reinforcement Learning as a Response Oracle

Throughout this work we use Double-DQN as an approximate best-response oracle and its success is critically dependent on its hyperparameters. For us to understand the hyperparameters, we must first clarify some implementation details of Double-DQN. The double in Double-DQN refers to two instances of the policies parameters; however, we will consider three policies each with different responsbilities. First, the *acting* policy is generating experiences through interaction with the environment. These experiences are then sent to the *learning* policy that uses them to improve the policy by updating its parameters. As a part of learning new parameters a *target* policy is used as a consistent optimization target. Periodically, the learning policy will be used to update the acting policy or the target policy.

Now from these different policies we can derive our hyperparameters:

- **Batch size:** the number of experiences to use when calculating a single gradient-step.
- **Discount factor (γ):** the weighting of current versus future rewards.
- **Replay capacity:** the number of experiences that an agent will store in their replay buffer. Once capacity is reached items are removed in a first-in first-out (FIFO) order.
- **Minimum replay buffer size:** the minimum amount of experiences that must be in the replay buffer before the agent begins learning. This can be expressed either as an integer number of experiences, or a fraction of the replay buffer's total size.
- **Learning frequency:** how often the acting policy will be updated with the learning policy's parameters, measured by the number of new experiences the acting policy must acquire.
- **Gradient steps per learning policy update:** number of gradient steps (with their own sampled mini-batches) the learning policy takes between each synchronization of its parameters with the acting policy.
- **Target update frequency:** how often the target network's parameters are updated with the learning policy's parameters, measured by the number of new experiences that must be acquired.
- **Exploration timesteps:** the total number of timesteps that the agent is in the exploration phase.
- **Training timesteps:** the total number of timesteps that the agent is learning.
- **Learning rate:** scaling factor weighting the impact of a single gradient step.
- **Gradient norm clip:** clip gradient updates using their global norm.
- **Activation function:** the activation functions used after each hidden layer in the neural network.
- **Hidden layer sizes:** the number of artificial neurons in each hidden layer of the neural network.

Appendix C. PSRO Evaluation Metrics

An instance of PSRO is evaluated by its regret as a function of the cumulative amount of environment transitions that are simulated. For most environments, simulating a transition is often the most costly computational process. This can be the result of complex logic underlying the transition, or the need for real-world interactions. Simulation occurs both when generating a best-response oracle via RL and when simulating the payoff entries to the empirical game. The later term is straightforward to measure for games without infinite horizons; however, the former term is not as easy to measure.

When we are comparing PSRO algorithms via this metric it is crucial to consider how the RL algorithm determines how much simulated data to generate. This is typically specified via a *termination condition*, which defines when the RL algorithm is complete. There are three main termination conditions:

- **Fixed budget:** after a fixed number of timesteps the RL algorithm terminates.
- **Convergence:** the algorithm is complete when a convergence criteria is achieved. The convergence criteria is typically defined as the agent having not improved their return for some window of time. It may also be a function of any other measurable quantity of the agent such as their TD error or gradient magnitude.
- **None:** the RL algorithm does not have any specified termination condition and will continue until the experimenter arbitrarily intervenes.

In PSRO, we are interested in repeated applications of RL, so we omit consideration of having no termination condition. Furthermore, the convergence criteria can be difficult to define due to the unpredictable nature of the agent's learning pattern. An example of this difficulty is that an agent's performance may temporarily decrease during exploration. This could be seen as a good termination condition as it looks like the agent is no longer improving. However, after enough time with decreasing performance, the agent may discover a strong solution and experience a spike in performance. Or not. Therefore, we focus on fixed-budget as our termination condition. While this condition is not perfect it is more robust to the unpredictable behavior of RL agents.

Appendix D. Coverage Curve Details

This section of the appendix contains a breakdown of the coverage curves portrayed in Sections 5.1. Coverage curves depicted how well a method generalized across an opponent's strategy space. They are generated by estimating the payoff against a representative set of opponent strategies. In this case, we consider all strategies truncated to the tenths place. The performances are then sorted in decreasing order. Policies can then be compared by which has a higher coverage curve. Note, this evaluation method treats all opponent strategies with equal importance.

Coverage curves measured with respect to payoffs or returns are not a perfect evaluation method. If an opponent policy is particularly vulnerable to exploitation, then strong response methods to this opponent may appear to perform very well. This is because the payoff against the exploitable opponent, even if it has low support, can dominate the payoff received by any other policy in the opponent's current strategy. To account for this potential confounder, we additionally plot the coverage curves with *normalized return*, which is calculated as follows:

$$\|u(\pi_i, \sigma_{-i})\| = \frac{\sum_{\pi_{-i} \in \sigma_{-i}} \sigma_{-i}(\pi_{-i}) \cdot u(\pi_i, \pi_{-i})}{\sum_{\pi_{-i} \in \sigma_{-i}} \sigma_{-i}(\pi_{-i}) \cdot u(\text{BR}(\pi_i), \pi_{-i})} \tag{26}$$

Since this work focuses on black-box games, we cannot analytically calculate payoffs, but instead, must estimate payoffs through simulation. In order to compute coverage curves we follow two methodologies based off whether the response policy can leverage knowledge of the opponent's strategy. For methods where the response policy cannot use the opponent's strategy (e.g., BR(0) cannot benefit from being told they are playing opponent policy 1), the performance against each opponent policy does not change across strategies. Therefore, we first simulate the performance against each opponent policy, then we can compute the performance against each opponent strategy by appropriately averaging the respective performance against each opponent policy by the likelihood of playing said opponent. On the other hand, methods such as Q-Mixing can condition on the opponent's strategy generating a unique response policy per for each opponent strategy. To evaluate these methods, we must simulate the response policy's performance against each opponent strategy independently. Keeping with a similar sampling procedure as the first methodology, we first simulate the per opponent policy performance and then average these performances by the opponent's strategy. However, this must be uniquely computed for each opponent strategy.

The remainder of this section contains figures showing the opponent's strategy throughout the coverage curves from Section 5.1. We provide the strategy breakdown across all 5 seeds and with respect to both return and normalized return. The performance against a single opponent policy is estimated using the mean return from 300 simulated episodes of RWS.

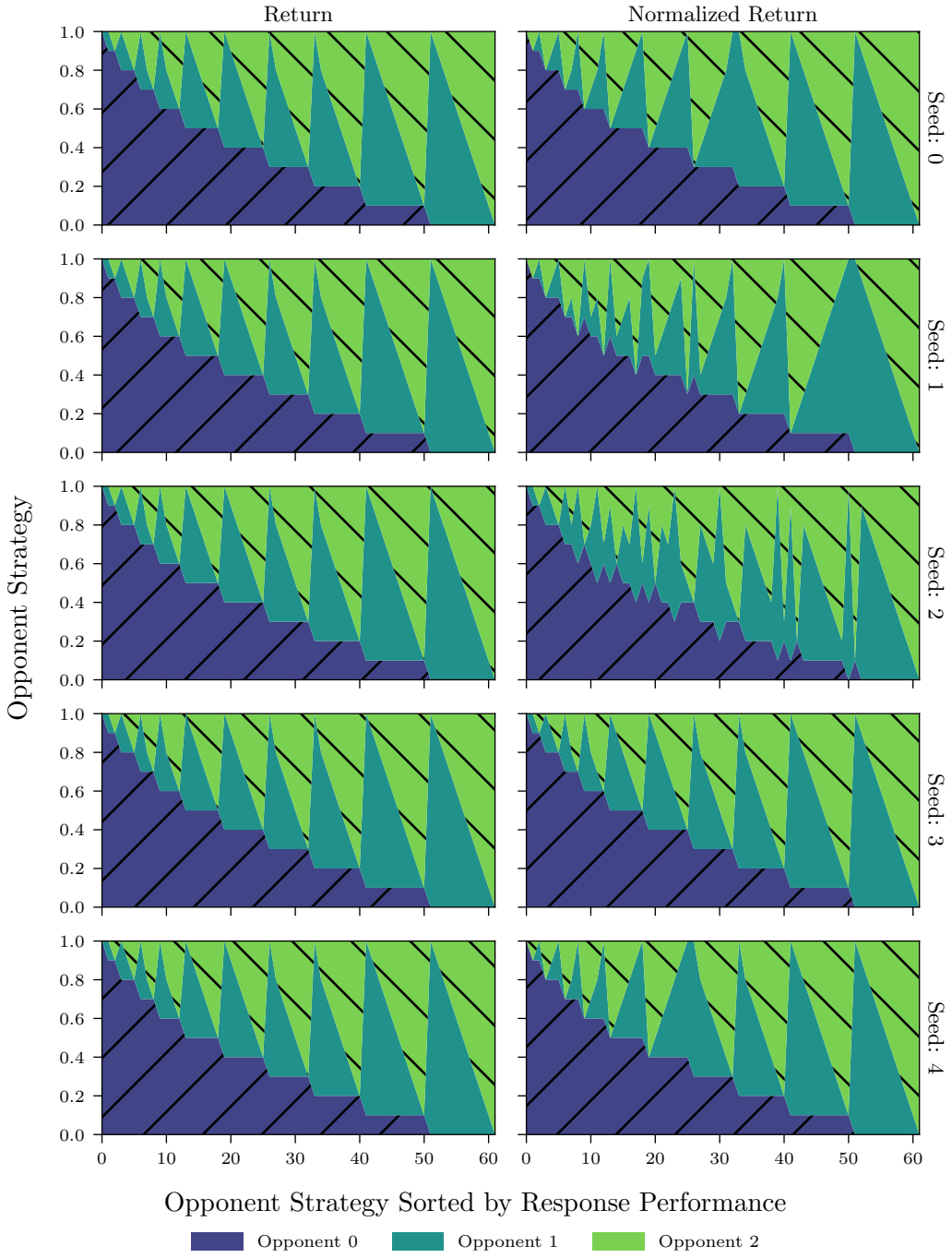


Figure 22: **BR(0)**.

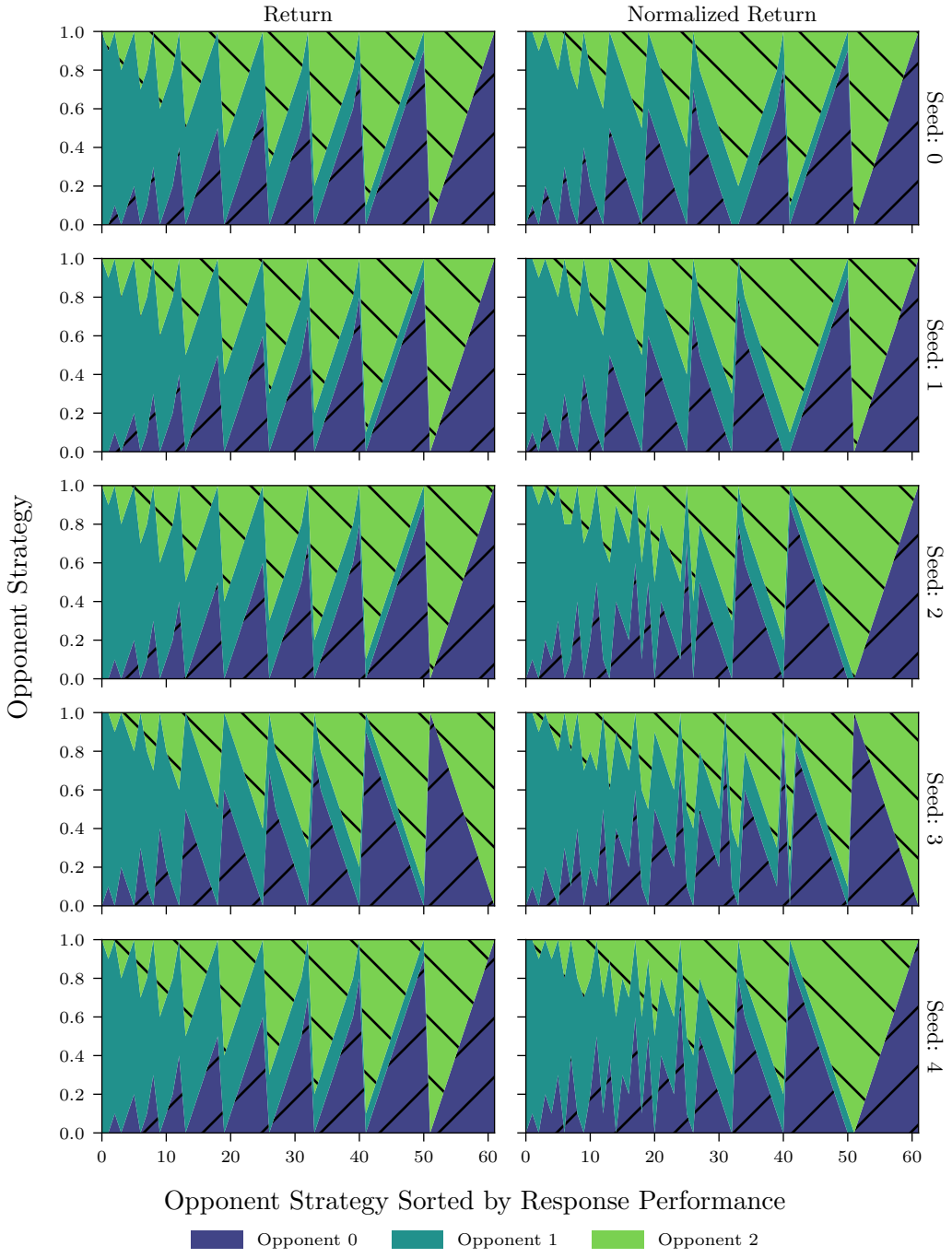


Figure 23: **BR(1)**.

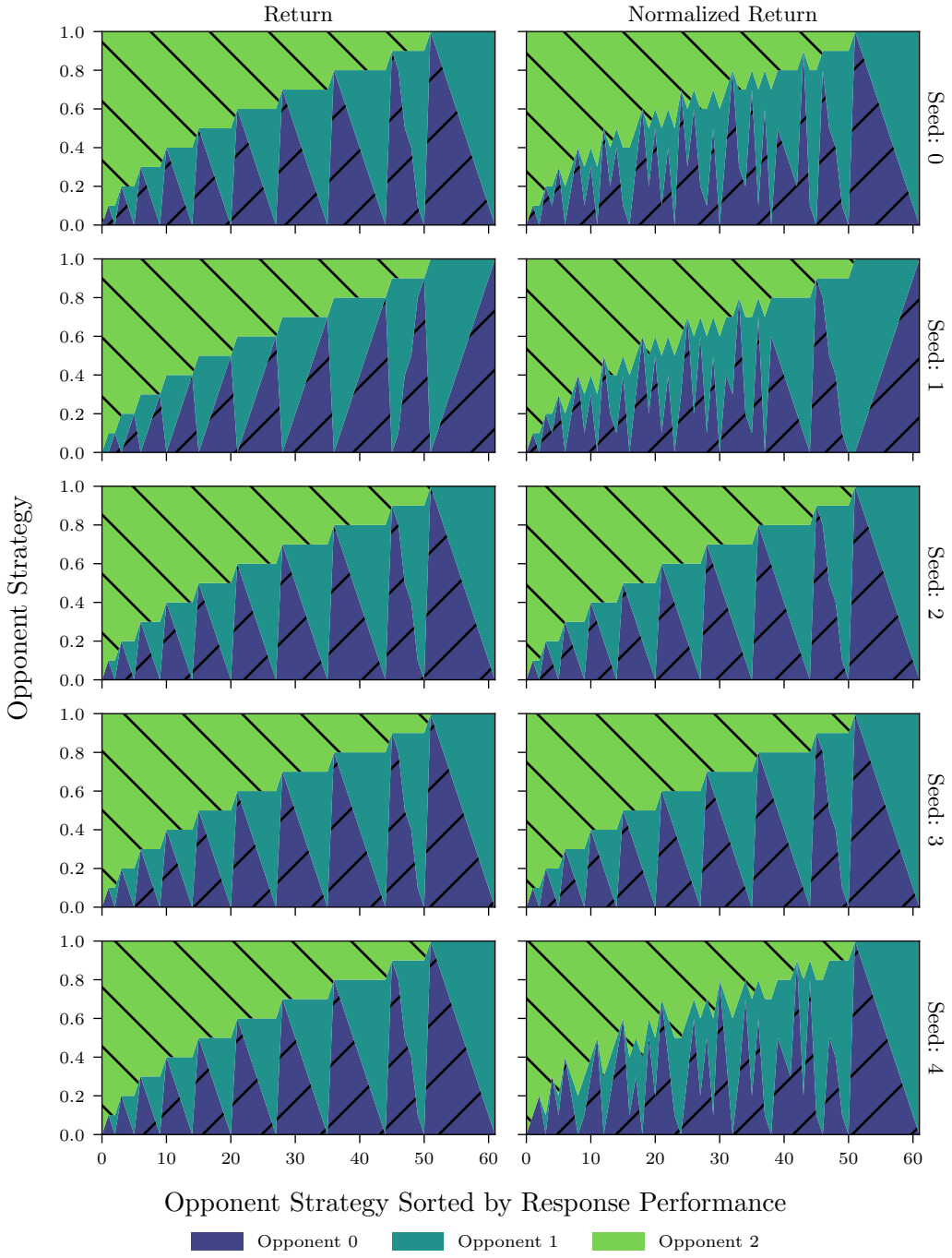


Figure 24: **BR(2)**.

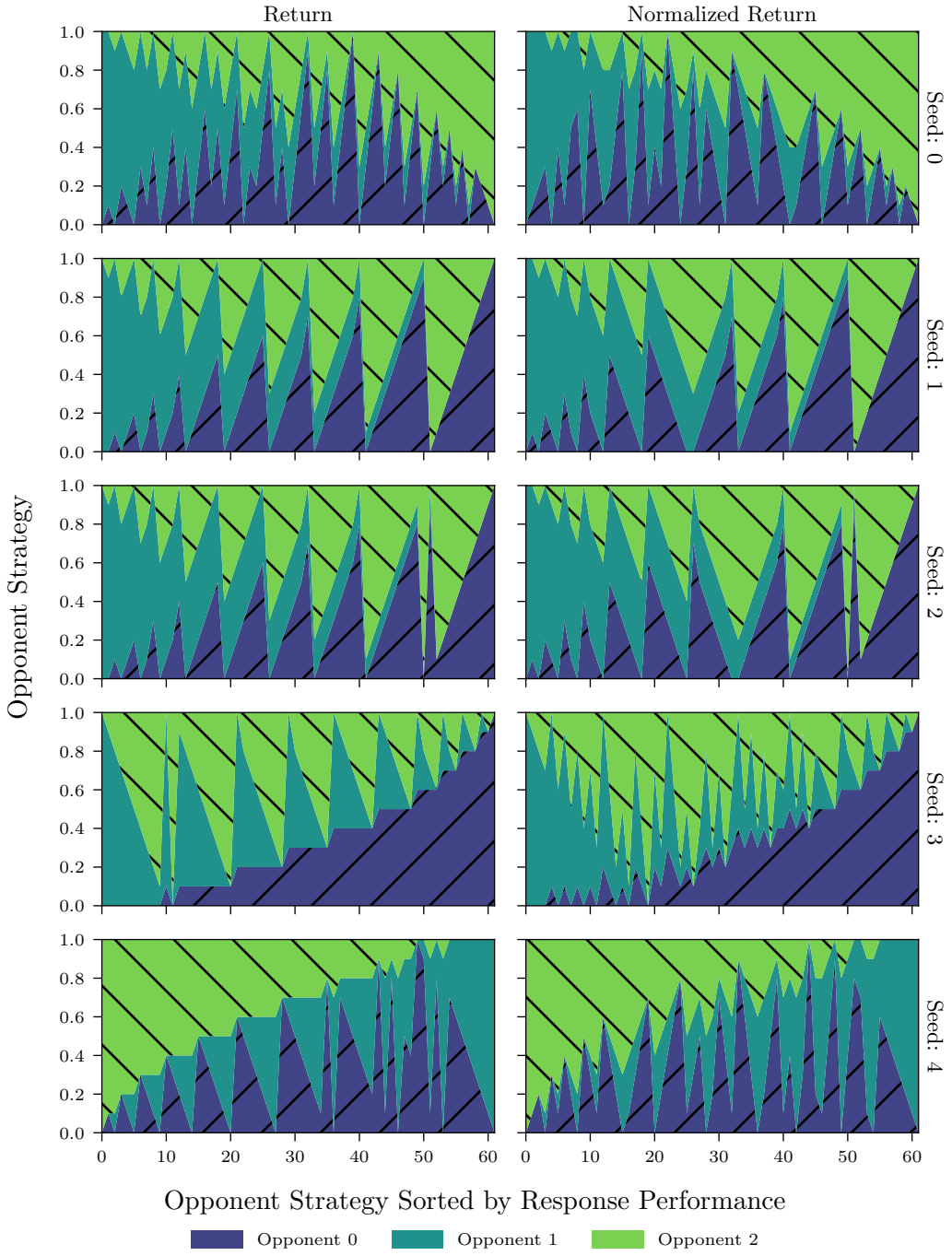


Figure 25: **BR(Uniform)**.

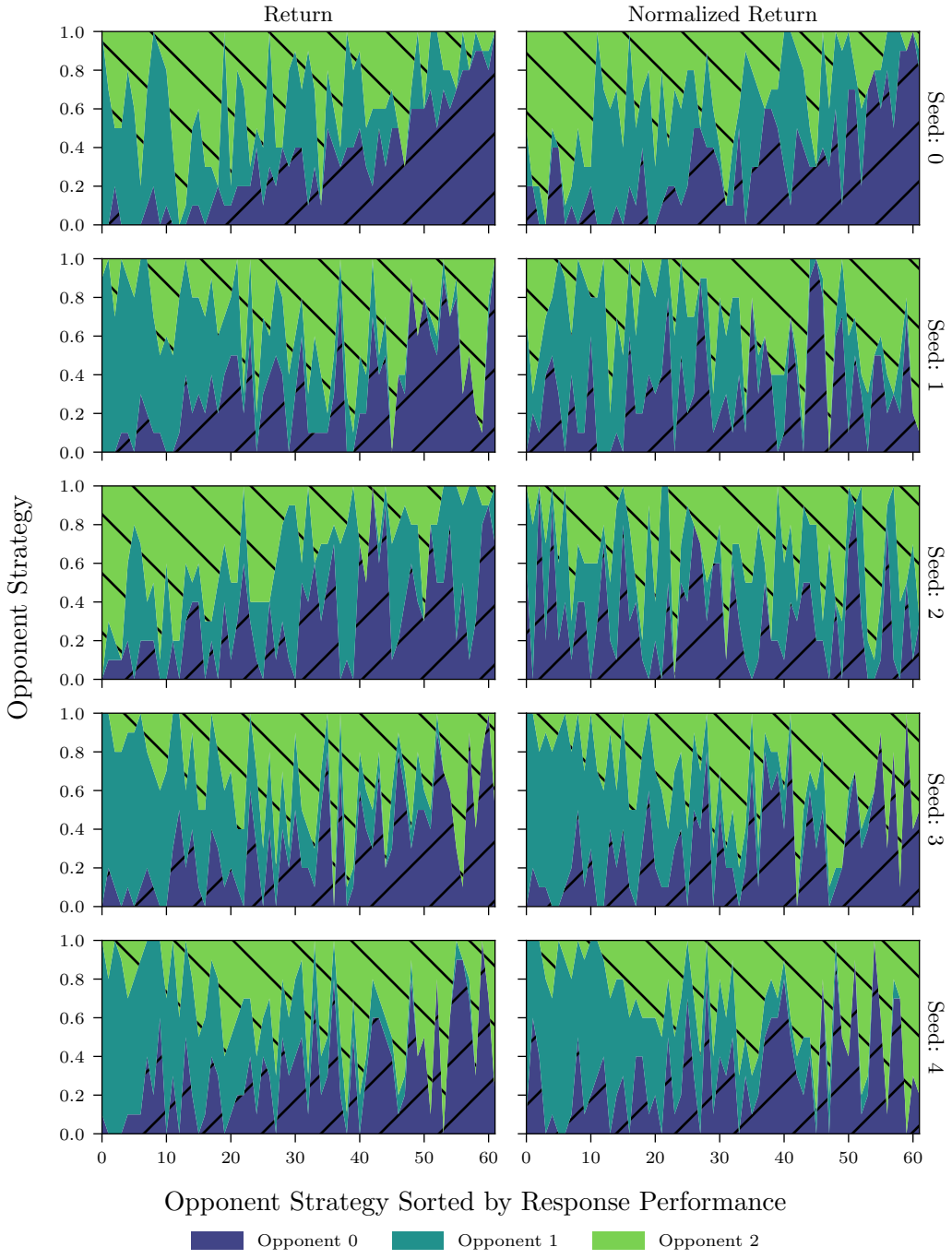


Figure 26: **Q-Mixing: Uniform Prior**.

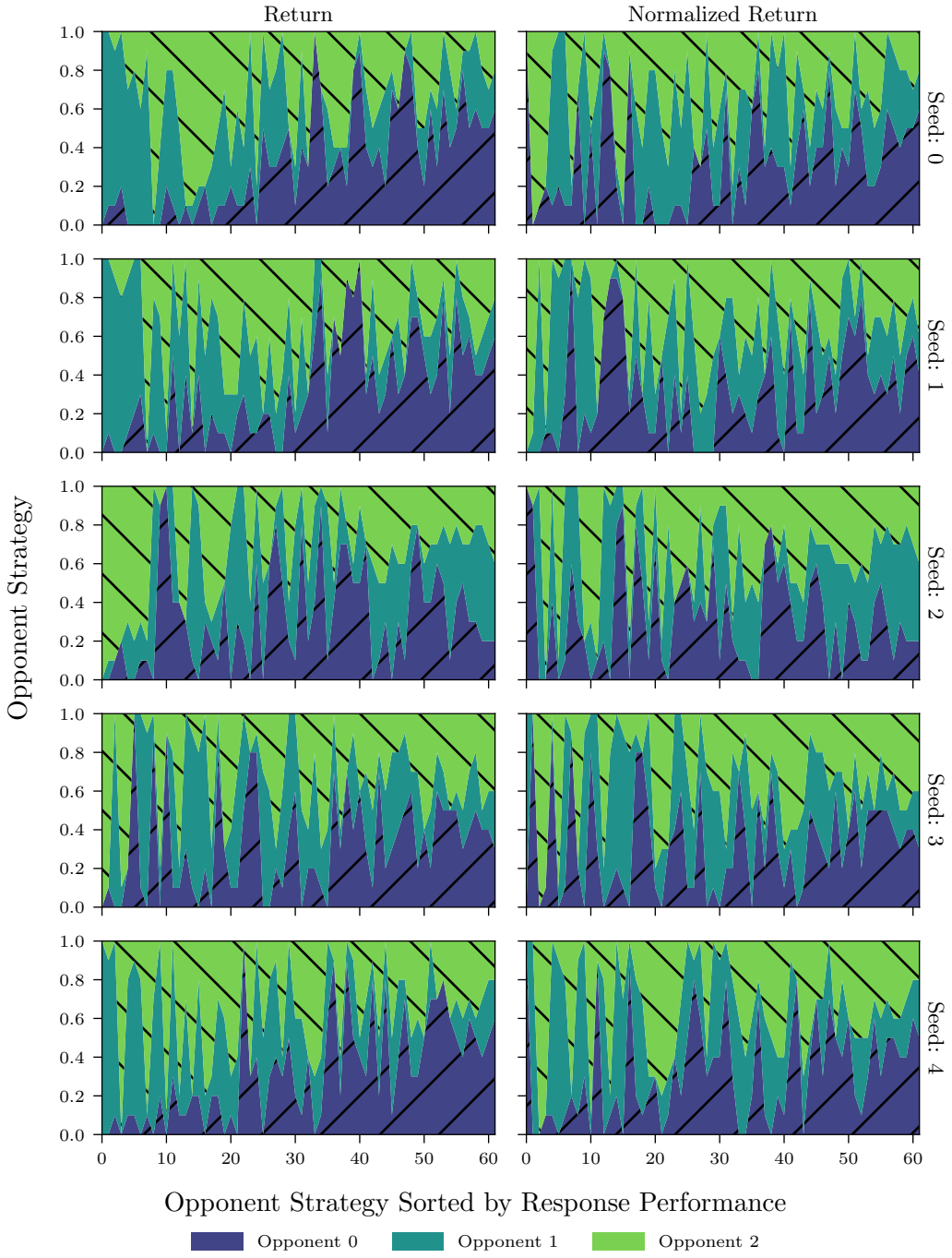


Figure 27: **Q-Mixing: Prior**.

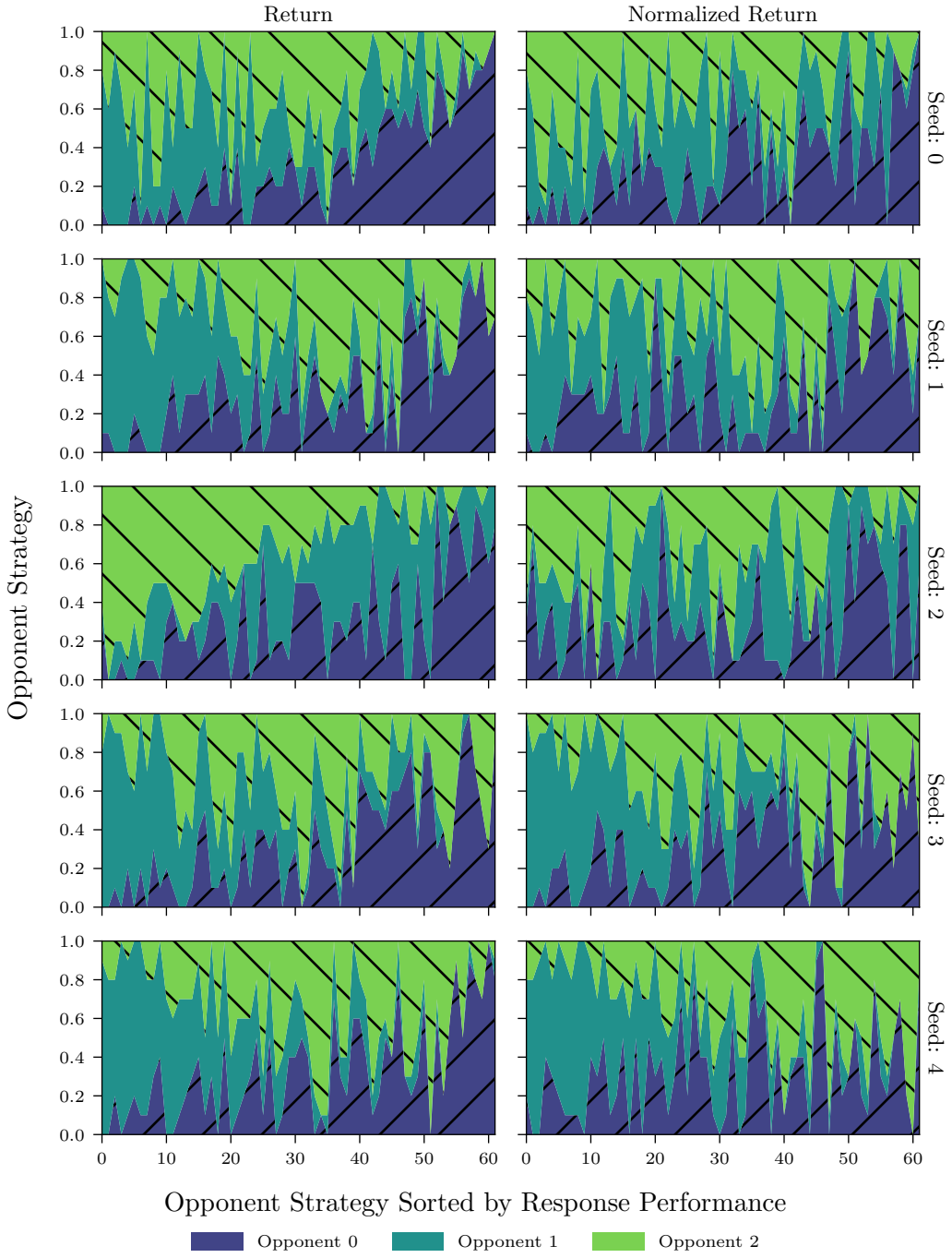


Figure 28: **Q-Mixing: Freq**.

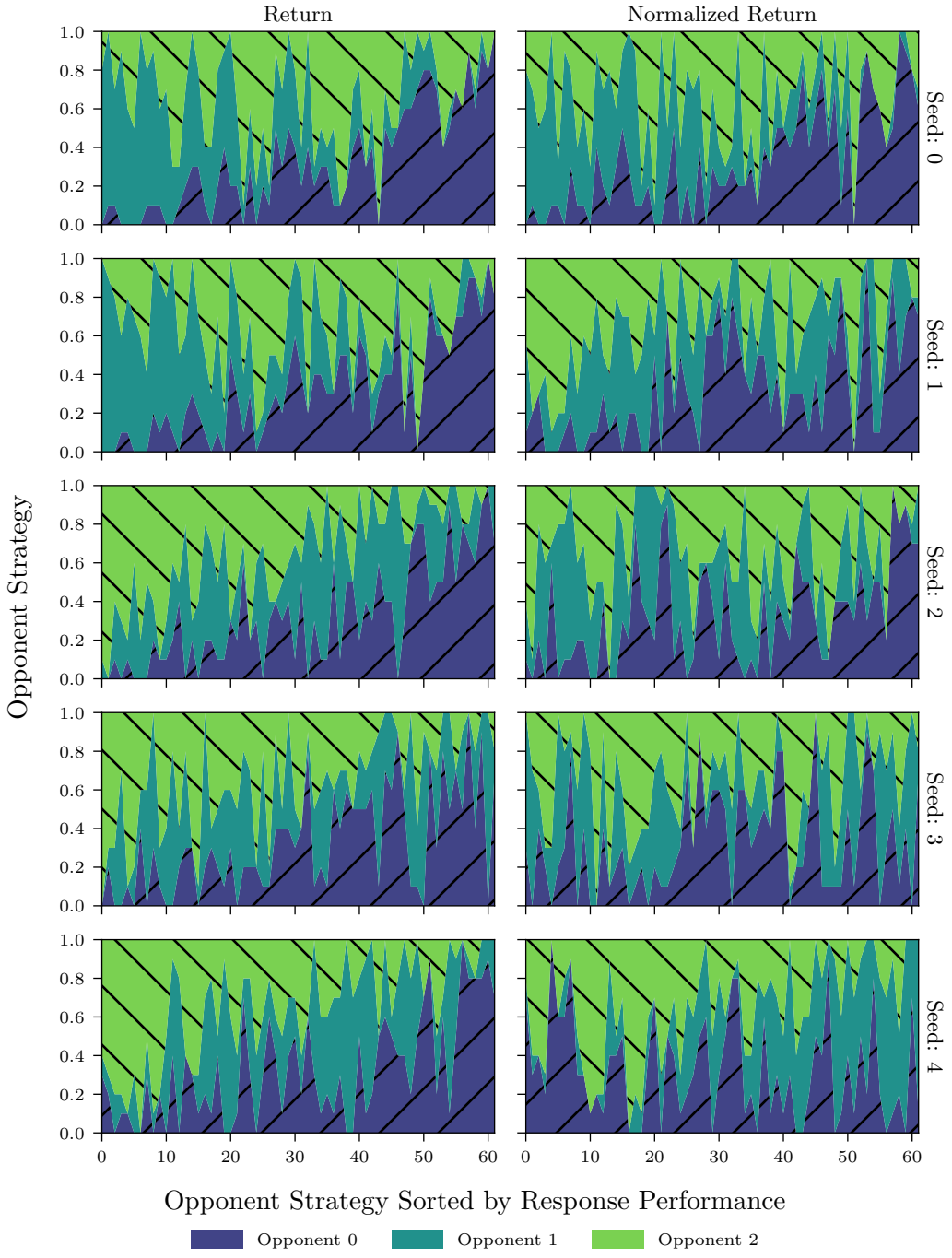


Figure 29: **Q-Mixing: Freq, Belief, Uniform Prior**.

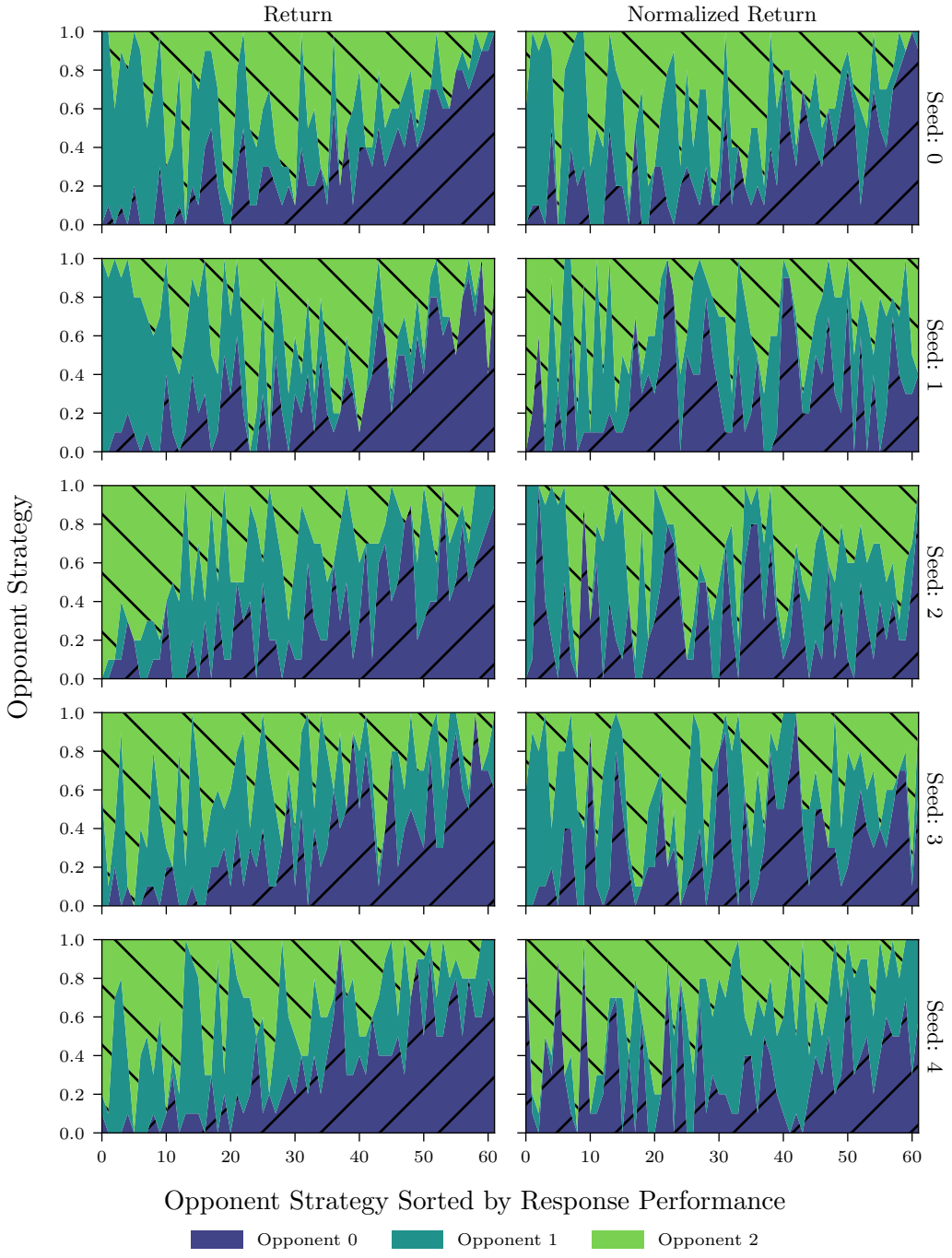


Figure 30: **Q-Mixing: Freq, Belief, Prior**.

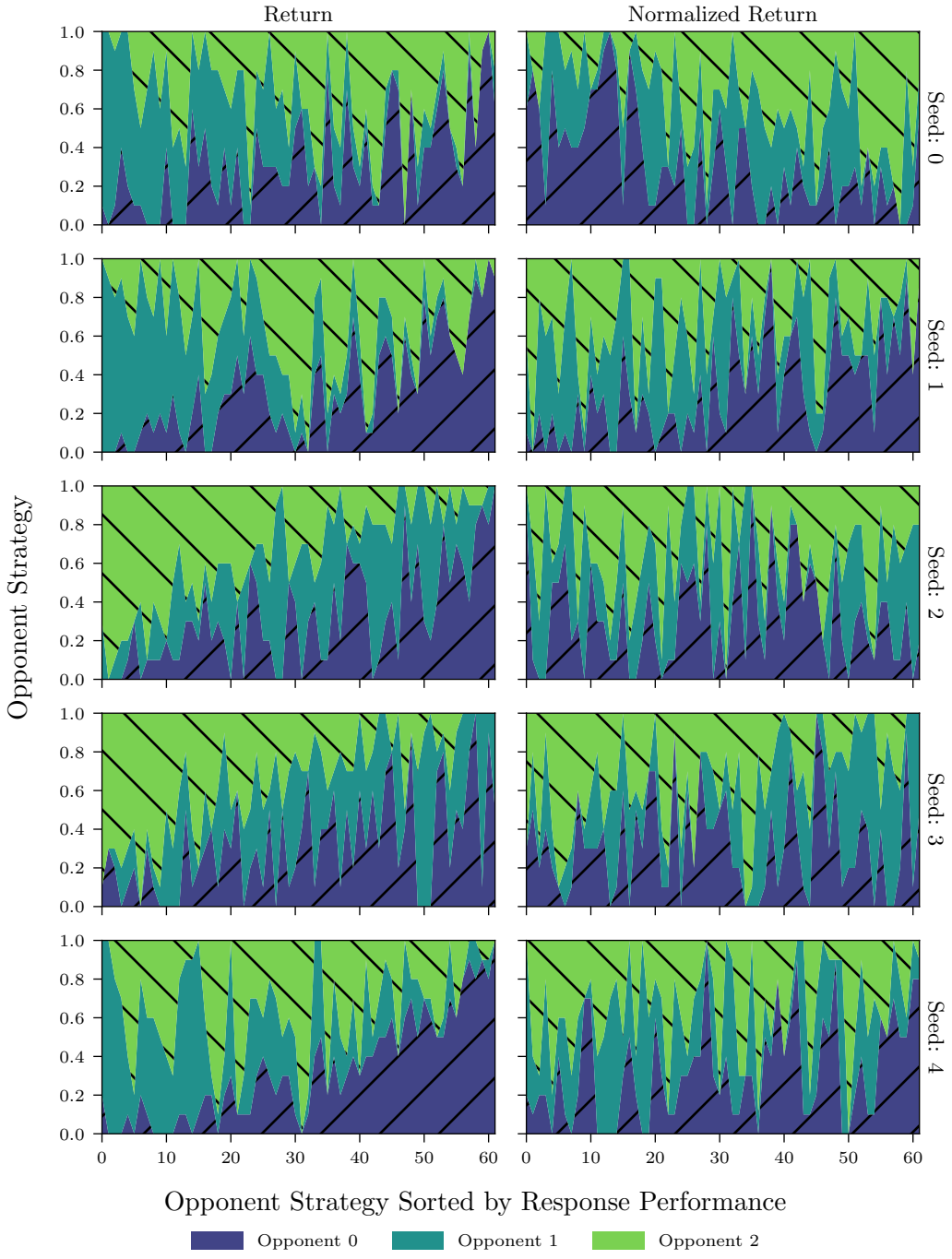


Figure 31: **Q-Mixing: OPC**.

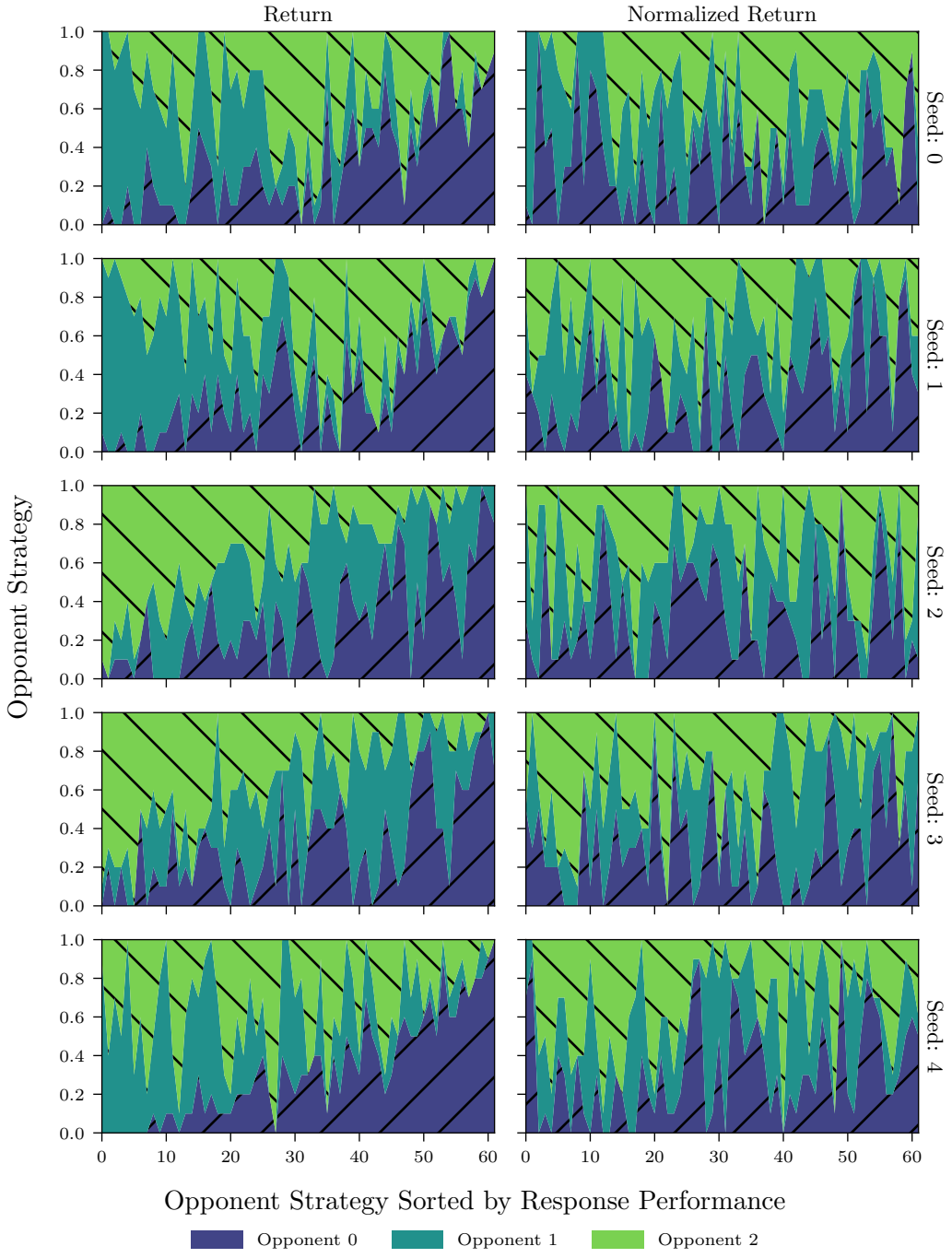


Figure 32: **Q-Mixing: OPC, Belief, Uniform Prior**.

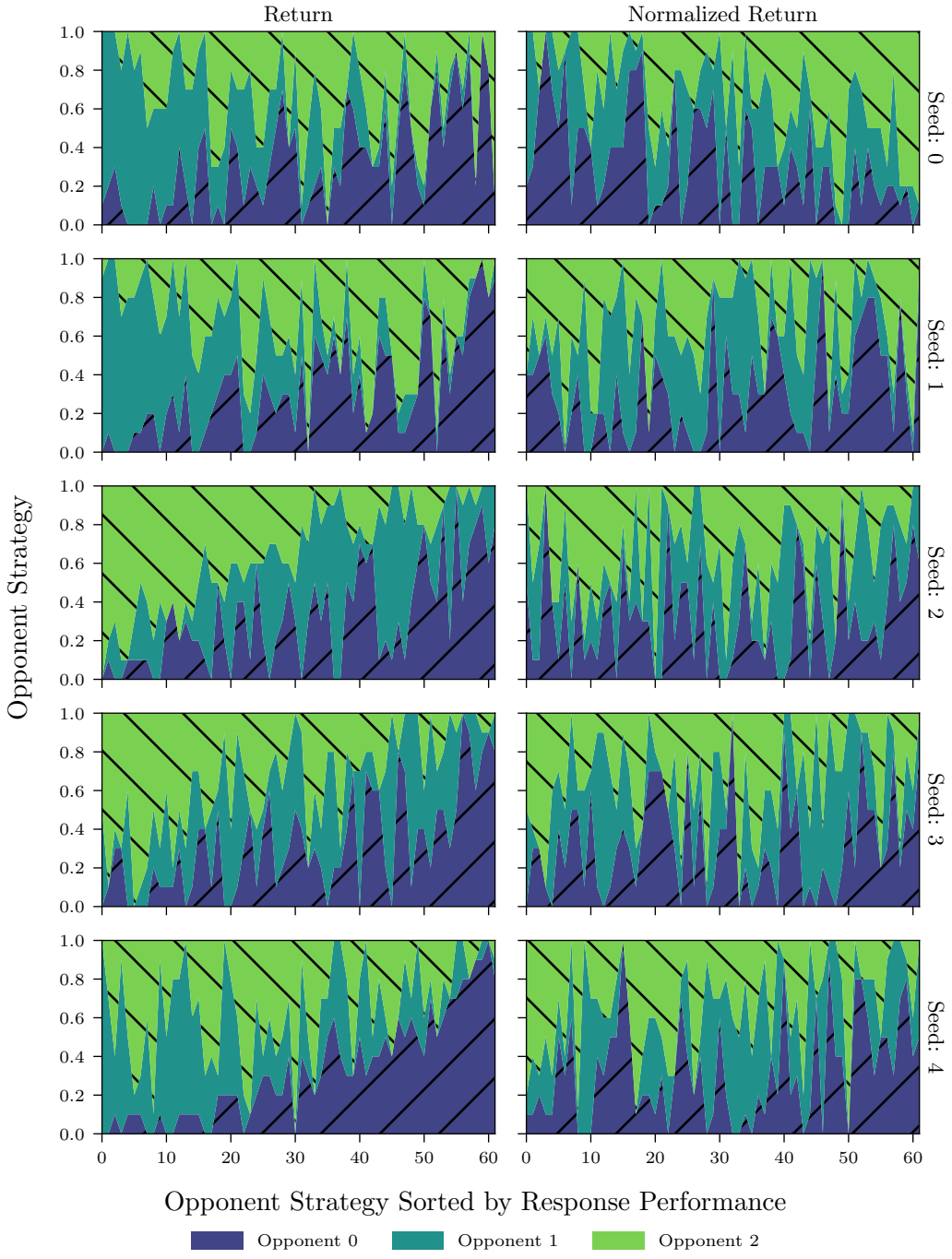


Figure 33: **Q-Mixing: OPC, Belief, Prior**.

Appendix E. Opponent Belief

In this section, we provide examples of an agent's belief in their opponent over the course of an episode. For each relevant response policy, we include three figures. Each figure corresponds to the response policy playing against a fixed opponent policy. For example, Figure 43 shows Q-Mixing: OPC playing against Opponent 0. Therefore in this figure we would expect a successful method to show beliefs weighting Opponent 0 highly.

Within each figure there are fifteen plots. The rows of plots correspond to the response policies generated from each experimental seed. The columns of plots correspond to different episodes of RWS, where the initial game state differs. An episode is terminated after 1000 timesteps, or when an agent chooses to play RPS against their opponent. Early termination is shown by by having zero support prescribed to all opponents after the end of the episode.

The agent's belief is independent across episodes. In other words, in Episode 2, the agent does not inform its belief by evidence gained from Episode 0 or Episode 1.

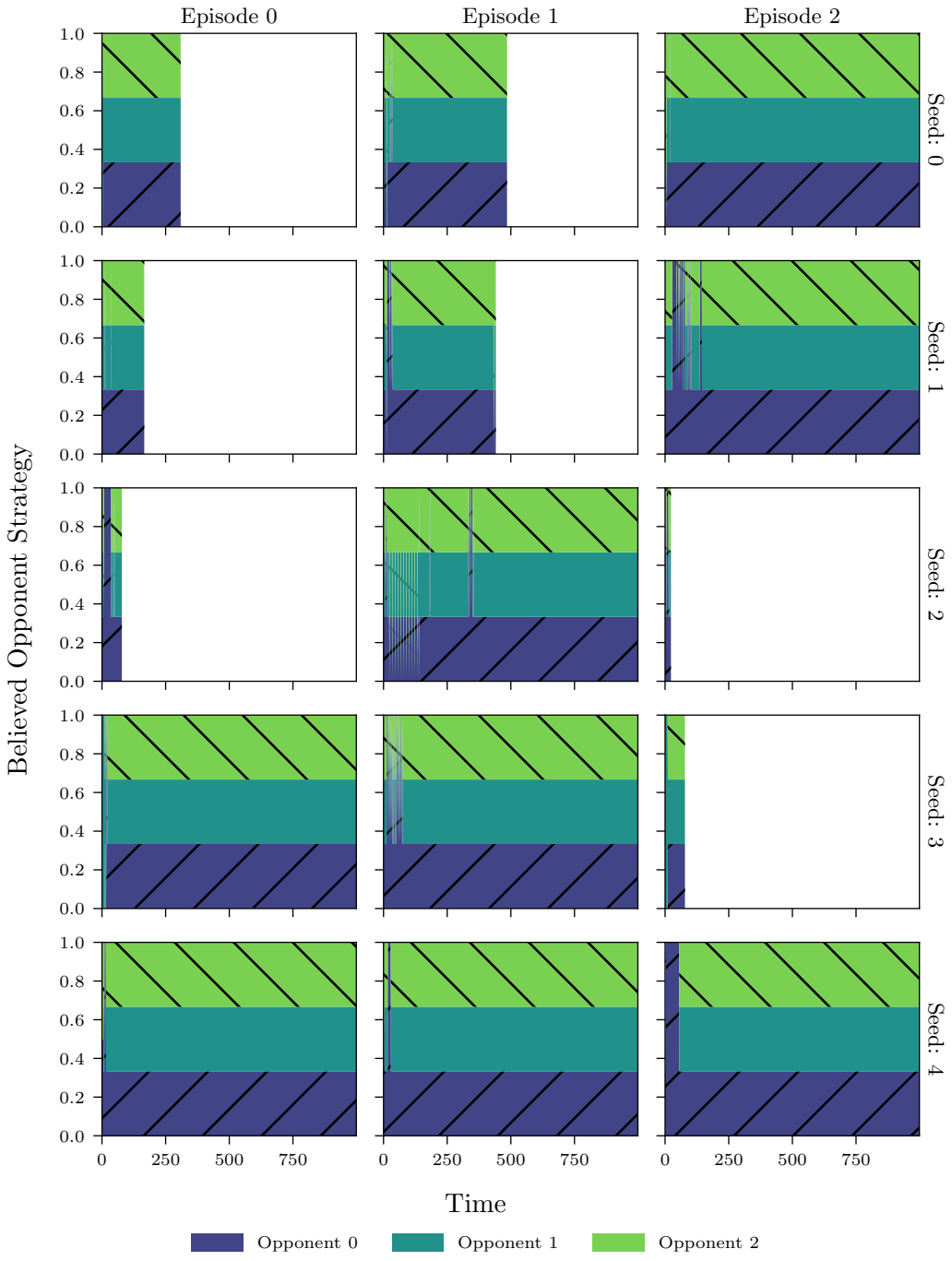


Figure 34: **Q-Mixing: Freq against Opponent 0**.

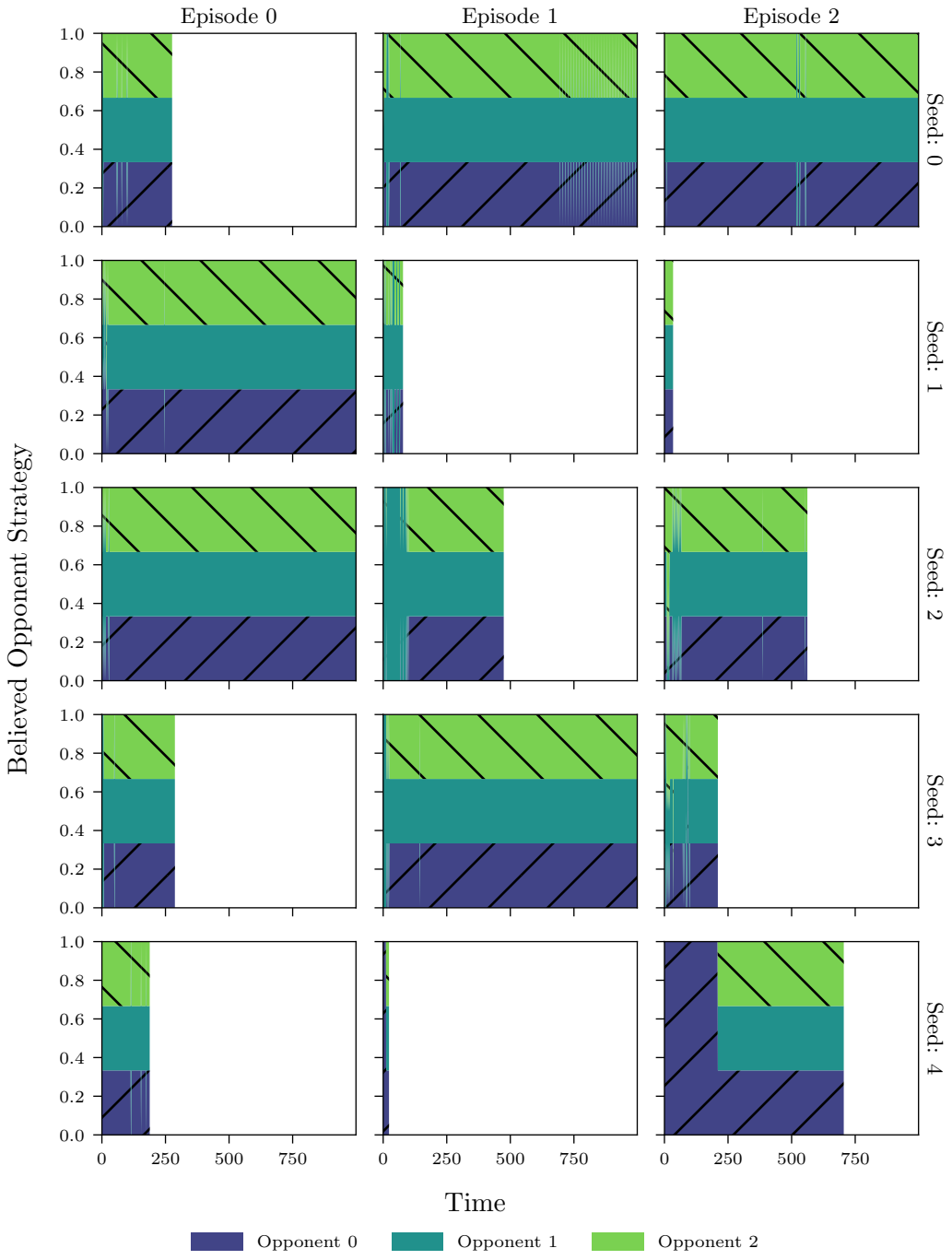


Figure 35: **Q-Mixing: Freq against Opponent 1**.

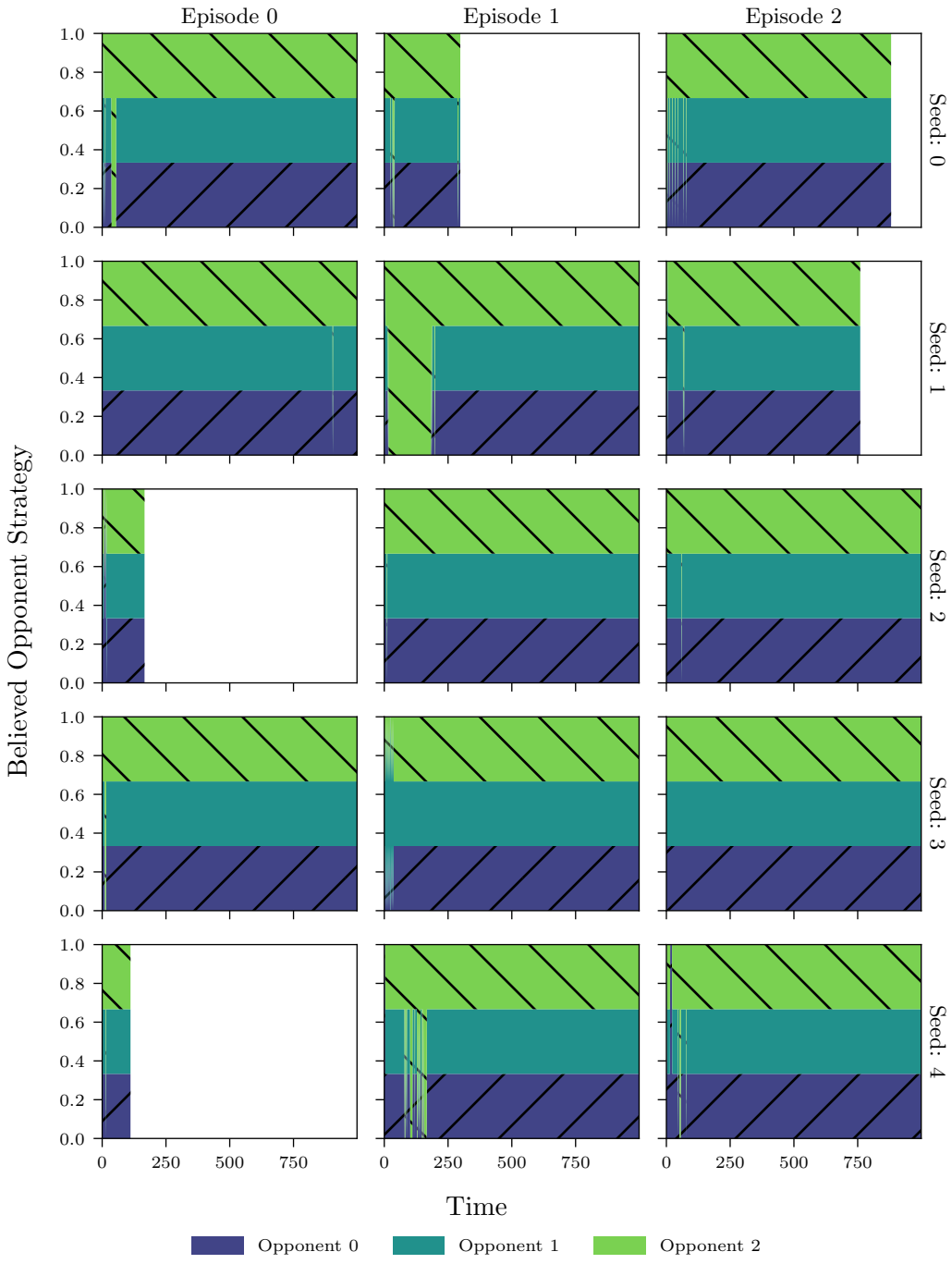


Figure 36: **Q-Mixing: Freq against Opponent 2**.

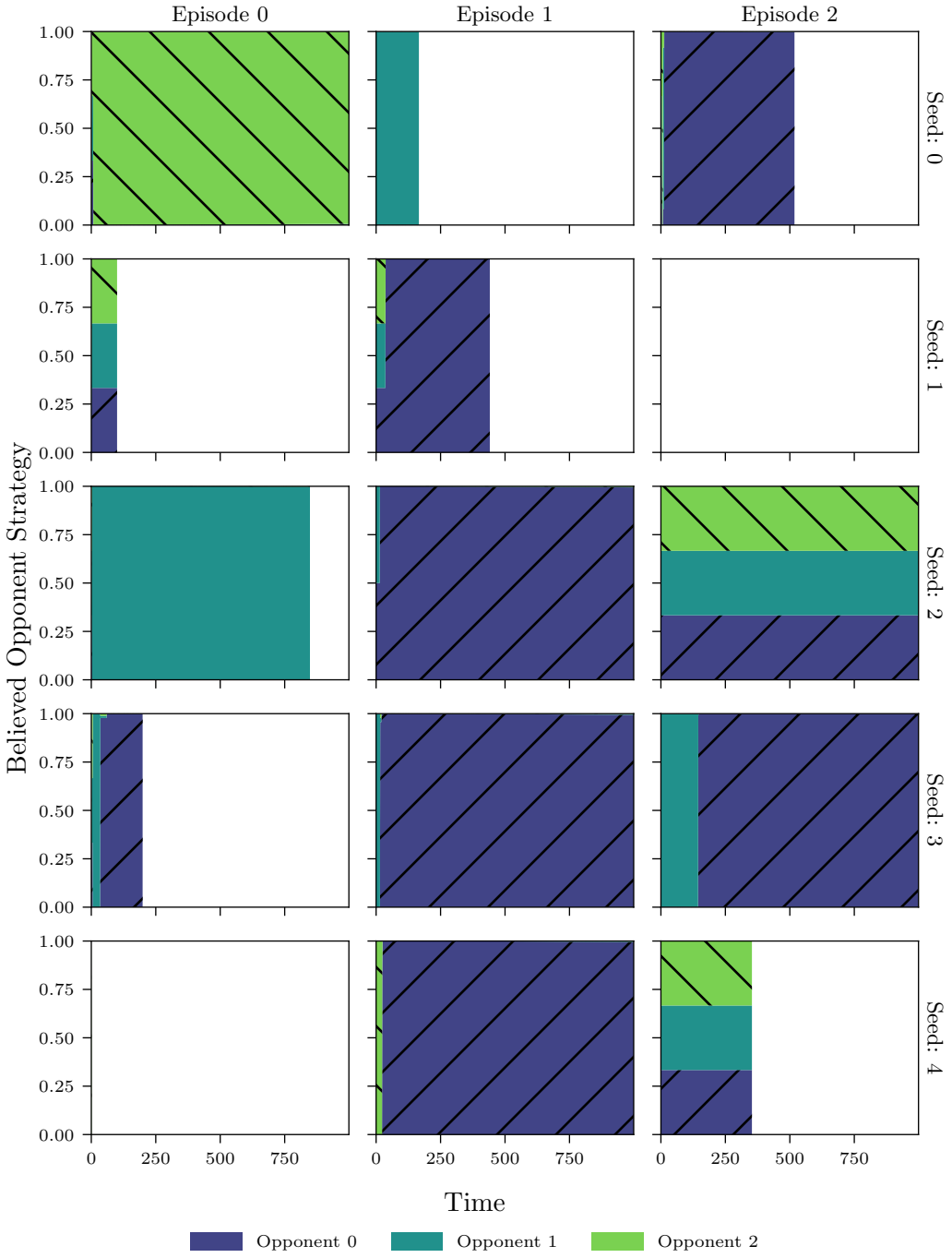


Figure 37: **Q-Mixing: Freq, Belief, Uniform Prior against Opponent 0**.

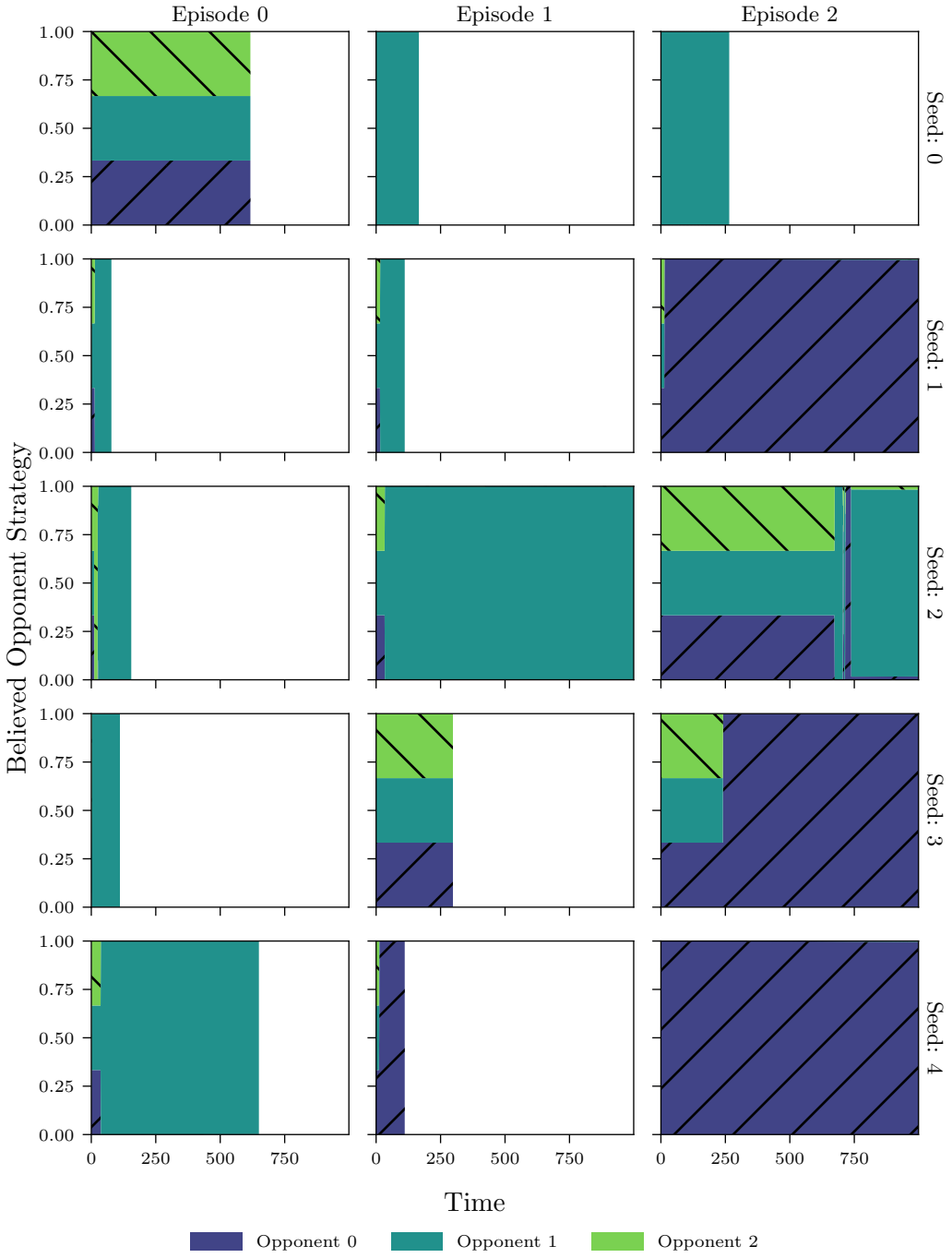


Figure 38: **Q-Mixing: Freq, Belief, Uniform Prior against Opponent 1.**

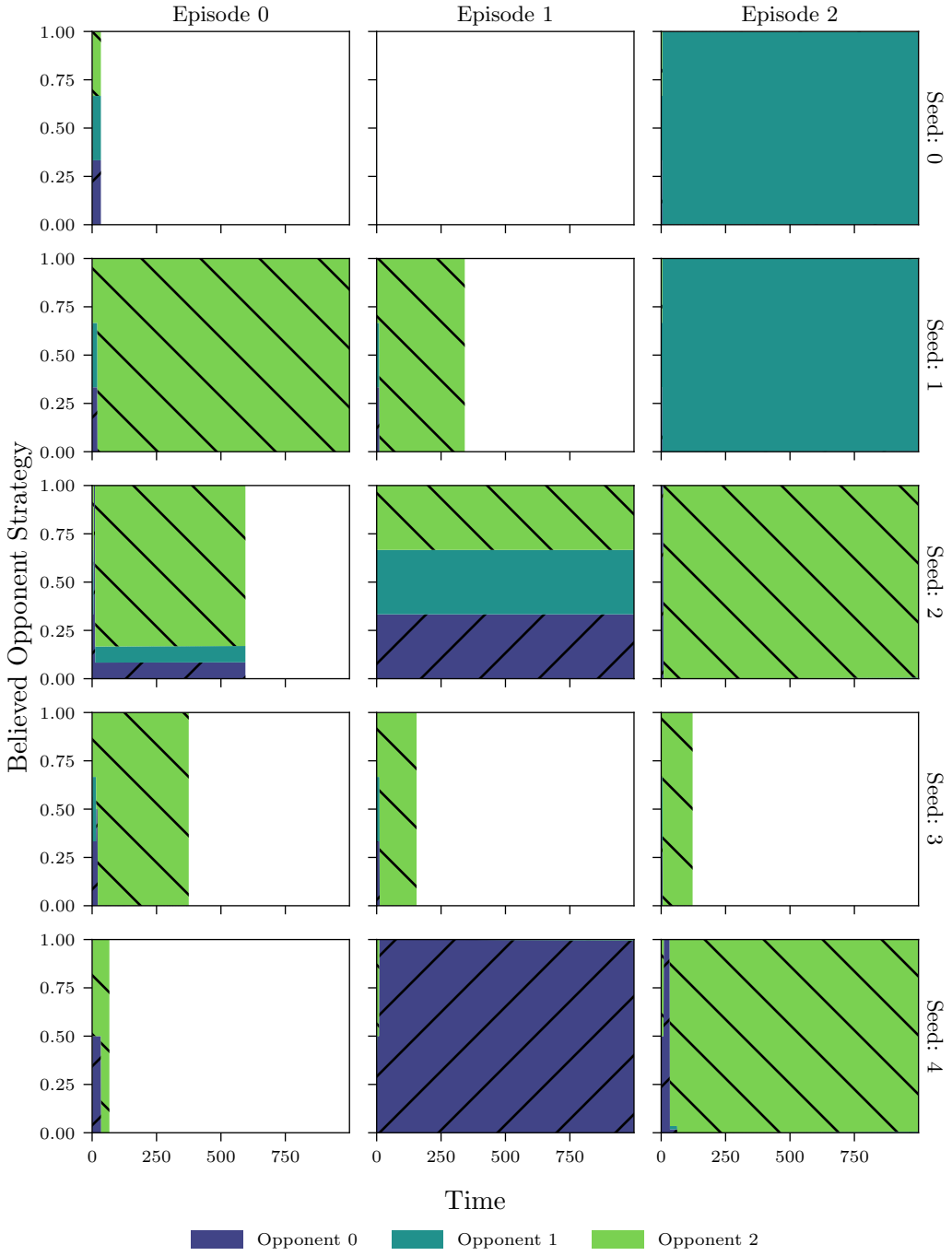


Figure 39: **Q-Mixing: Freq, Belief, Uniform Prior against Opponent 2**.

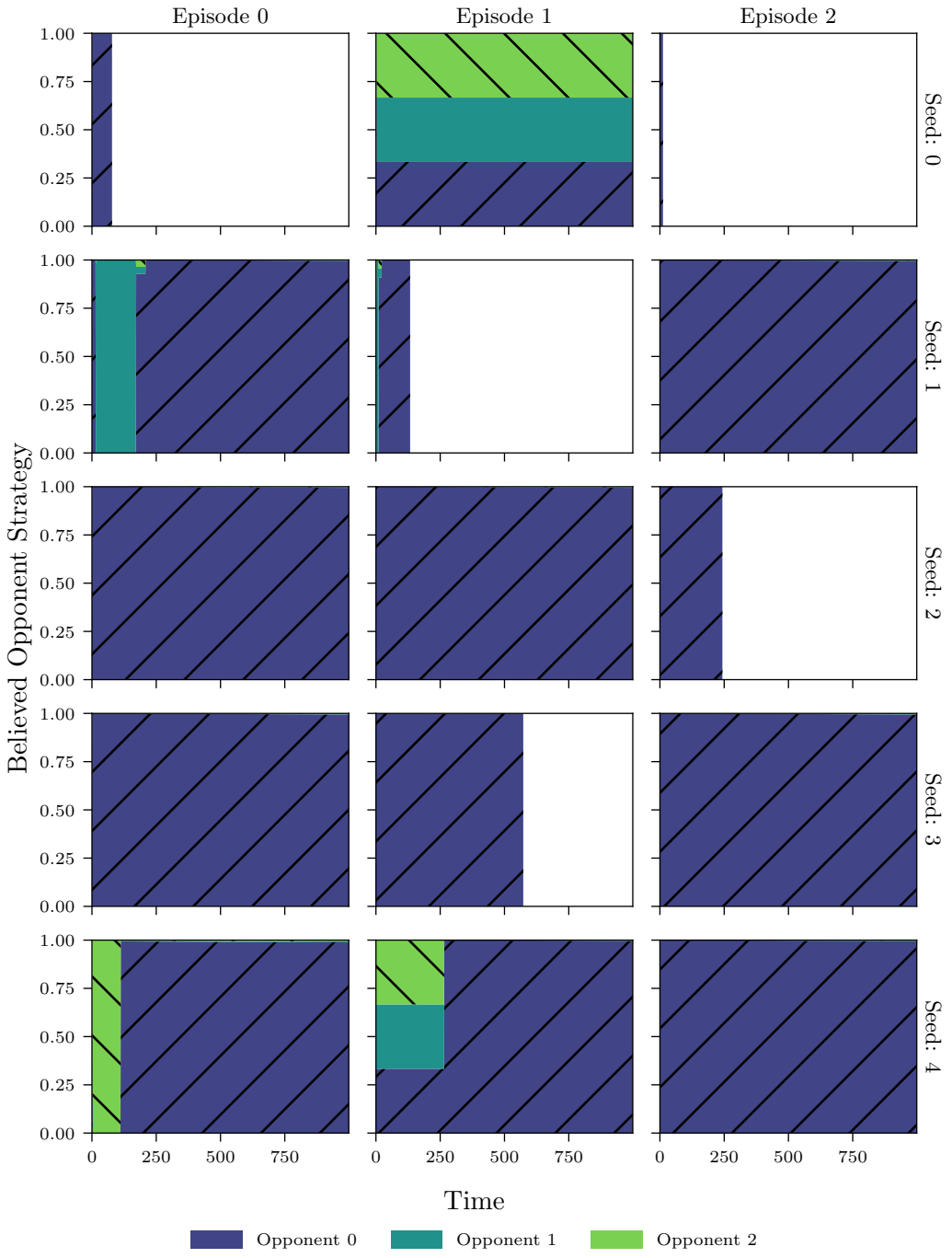


Figure 40: **Q-Mixing: Freq, Belief, Prior against Opponent 0**.

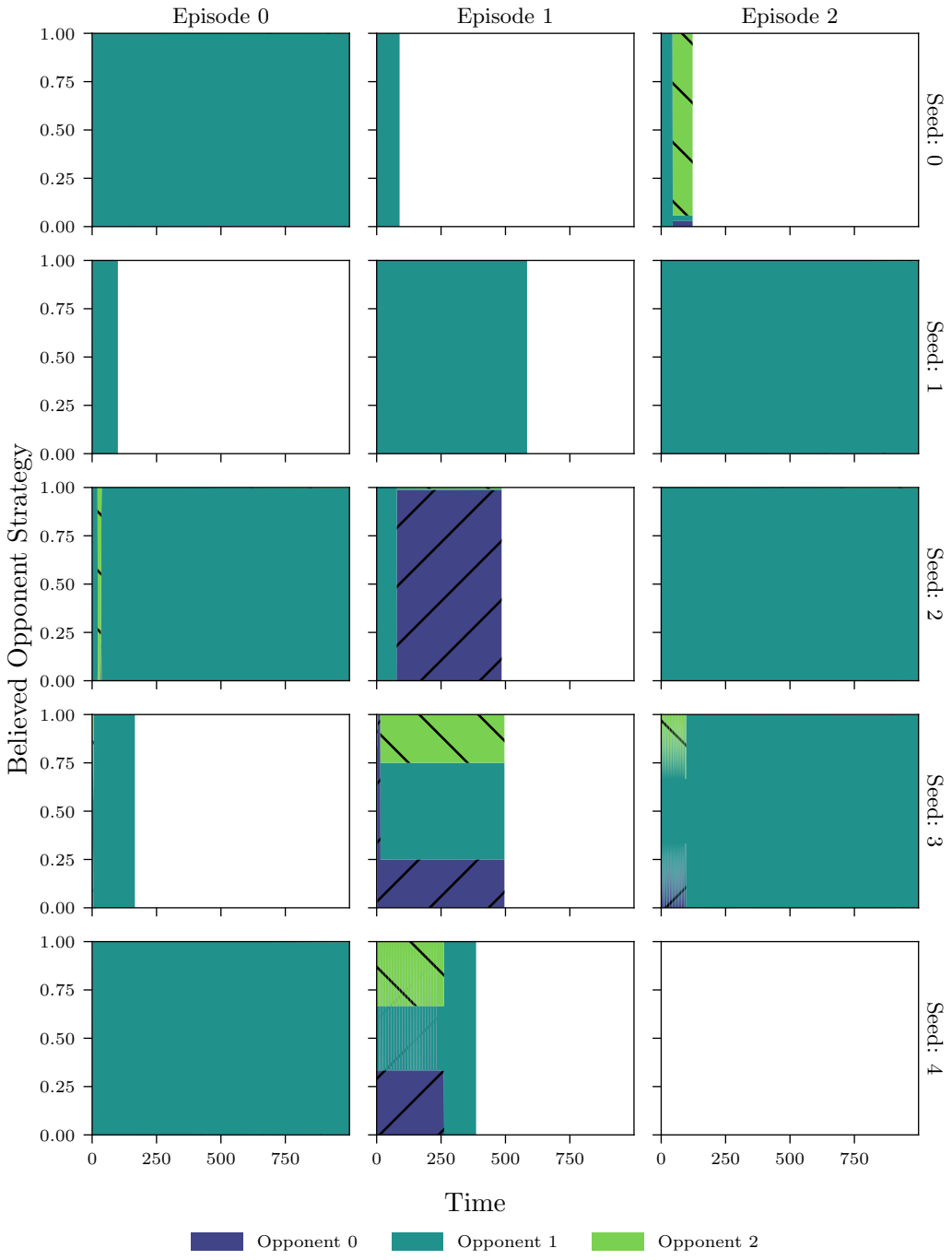


Figure 41: **Q-Mixing: Freq, Belief, Prior against Opponent 1**.

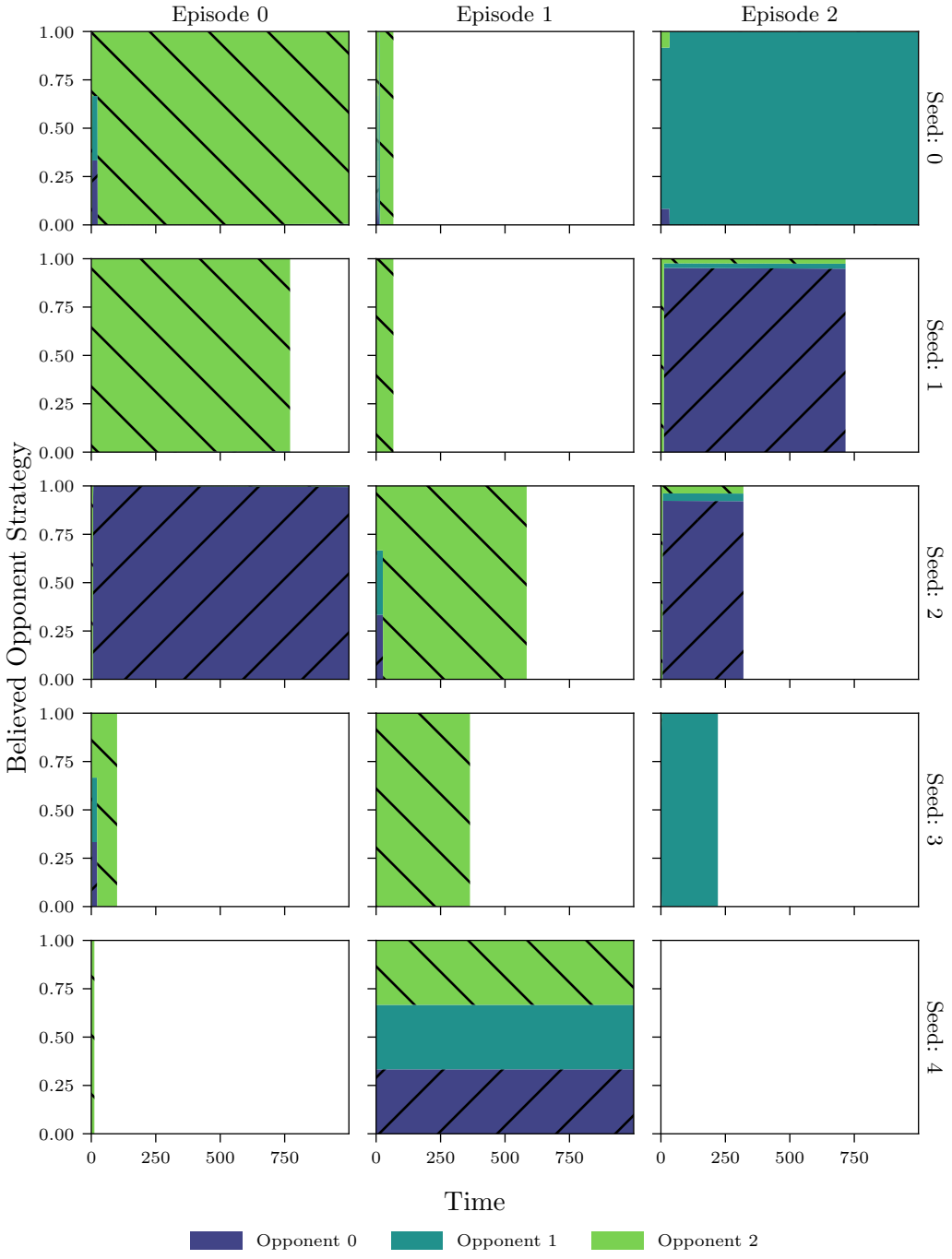


Figure 42: **Q-Mixing: Freq, Belief, Prior against Opponent 2**.

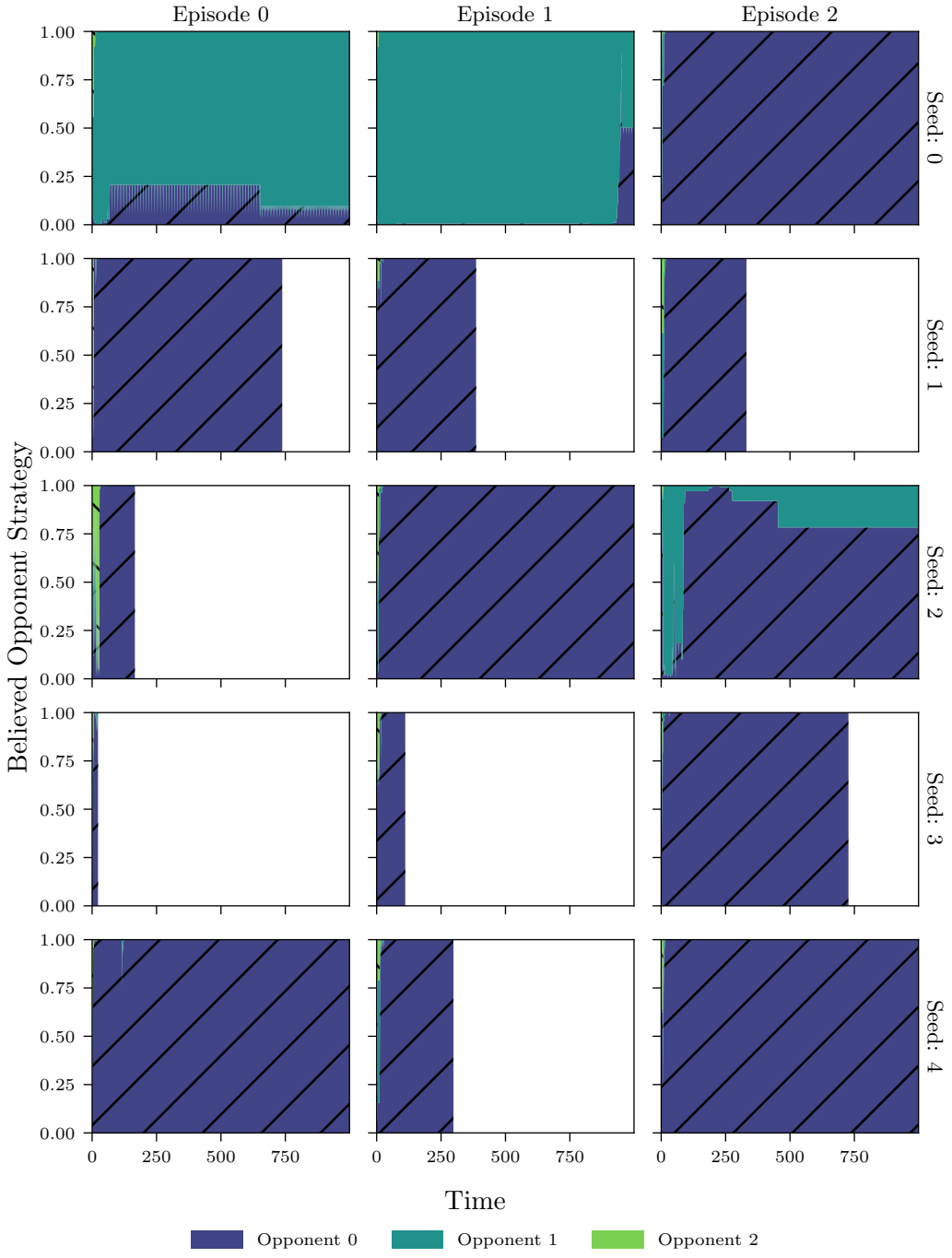


Figure 43: **Q-Mixing: OPC against Opponent 0**.

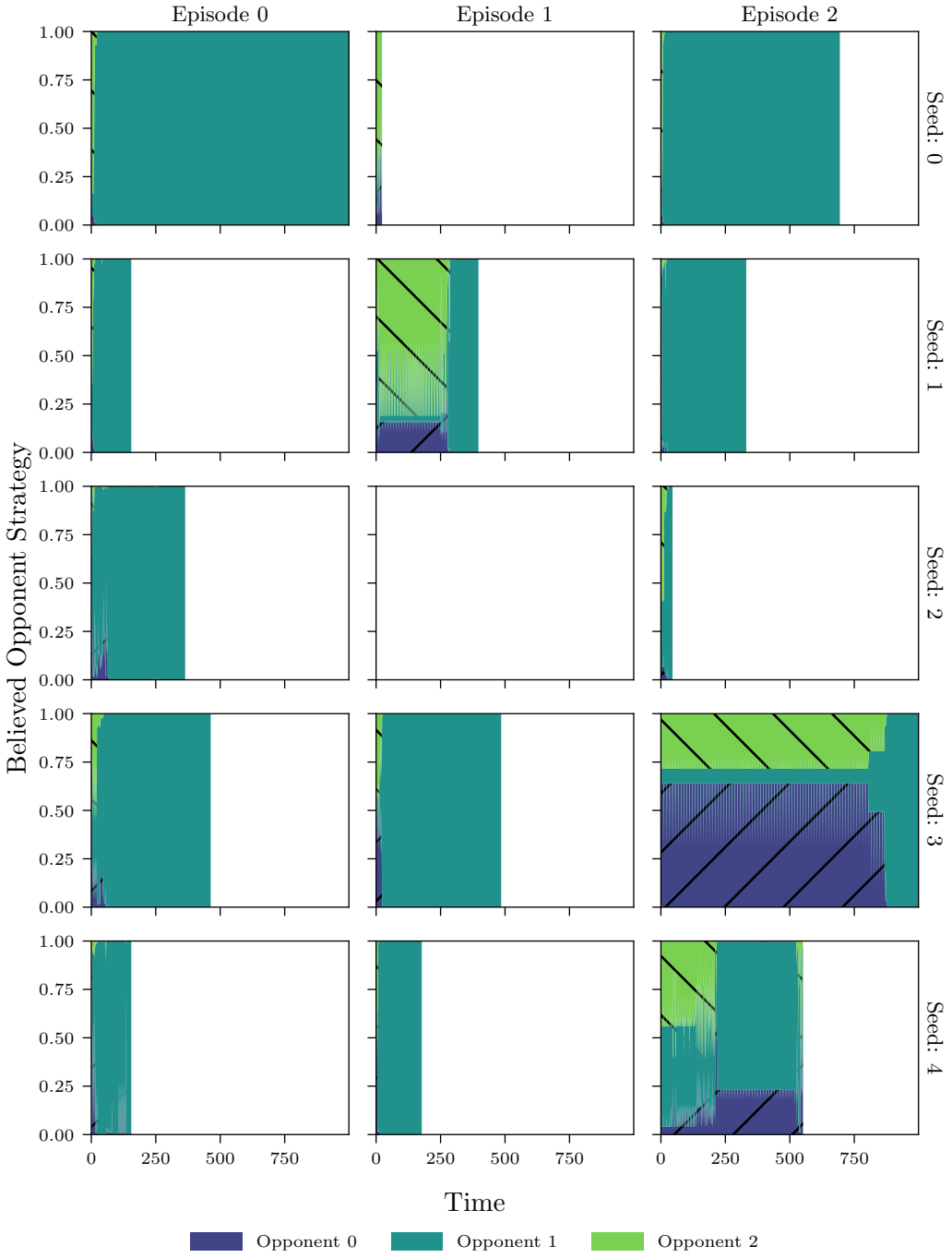


Figure 44: **Q-Mixing: OPC against Opponent 1**.

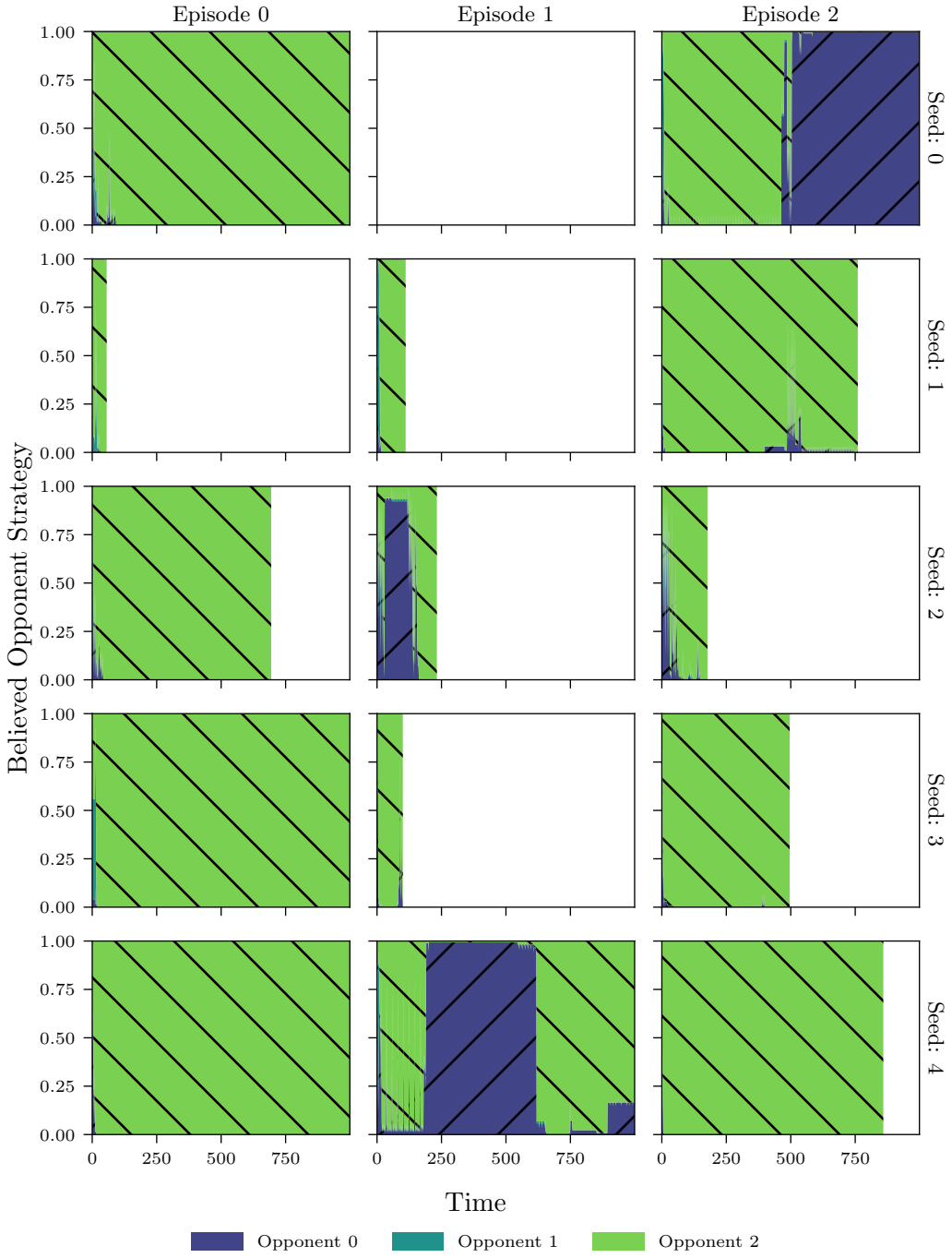


Figure 45: **Q-Mixing: OPC against Opponent 2**.

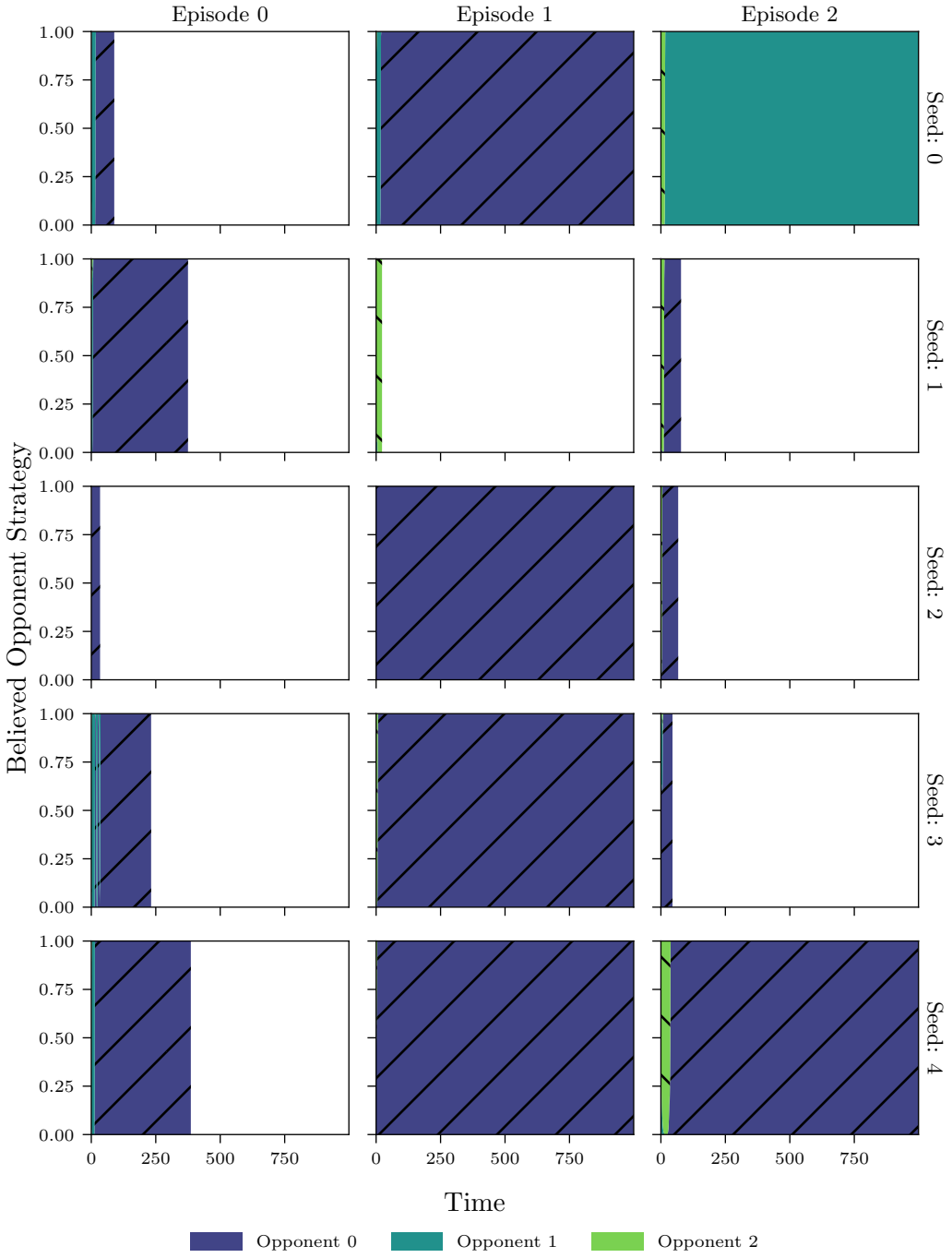


Figure 46: **Q-Mixing: OPC, Belief, Uniform Prior against Opponent 0**.

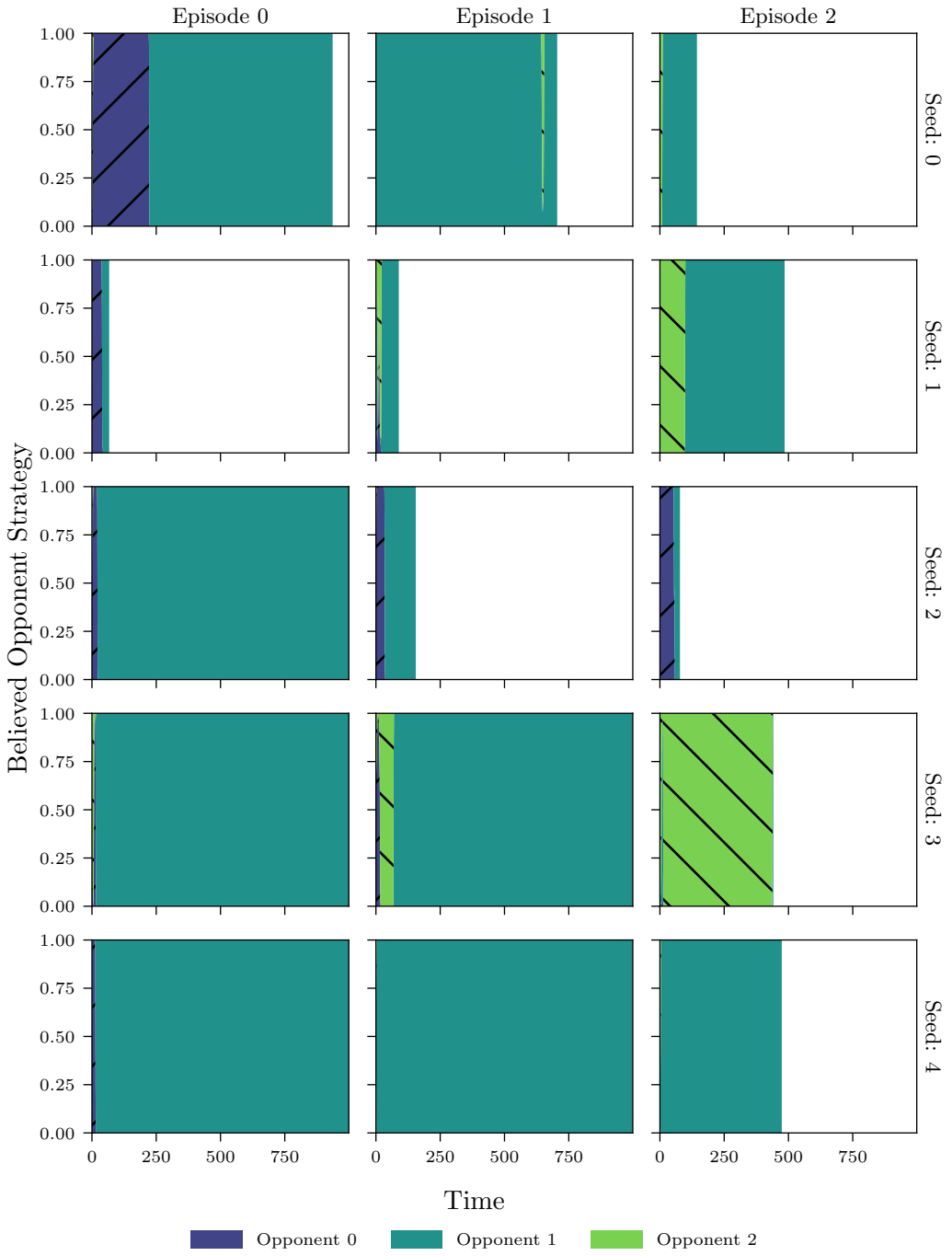


Figure 47: **Q-Mixing: OPC, Belief, Uniform Prior against Opponent 1**.

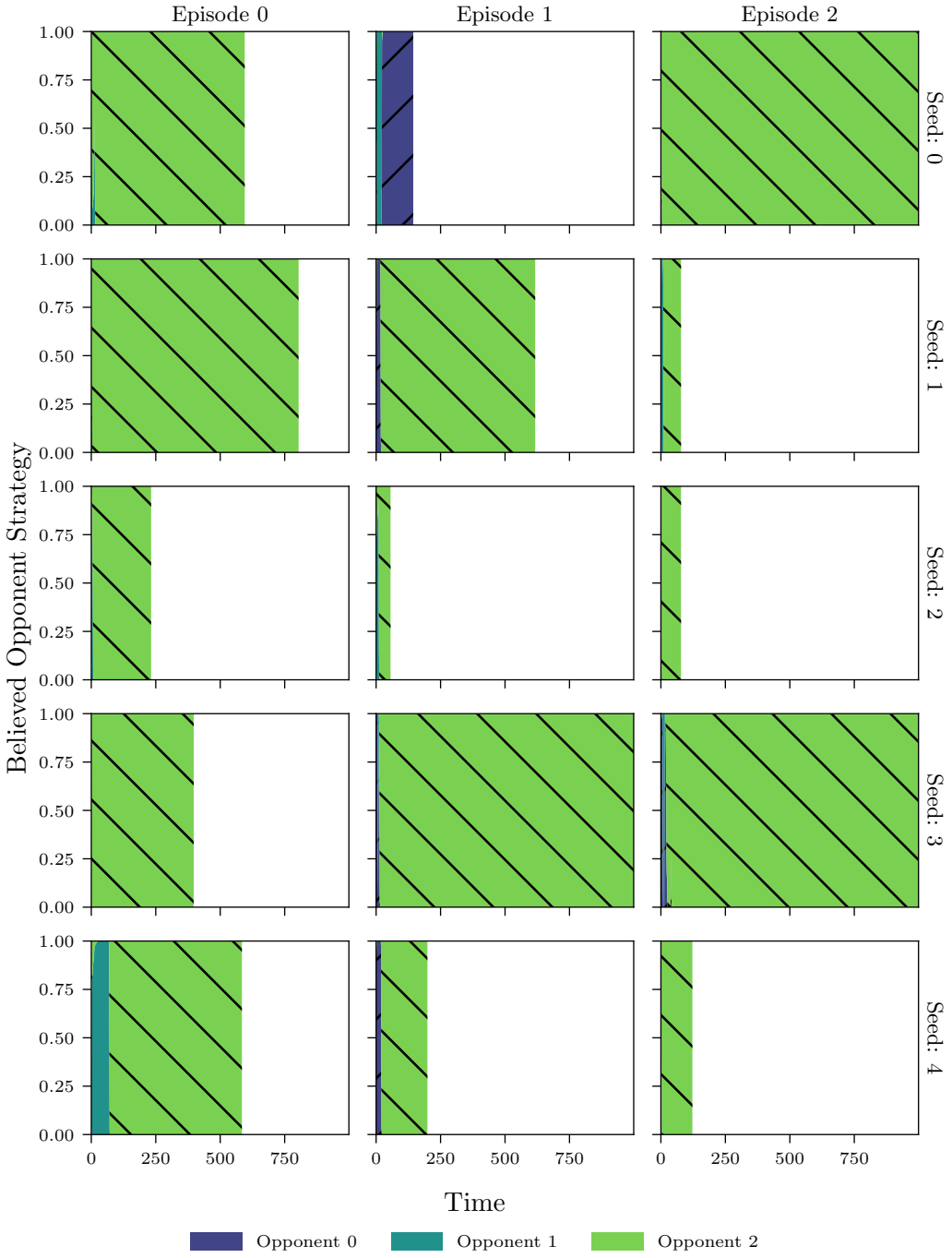


Figure 48: **Q-Mixing: OPC, Belief, Uniform Prior against Opponent 2**.

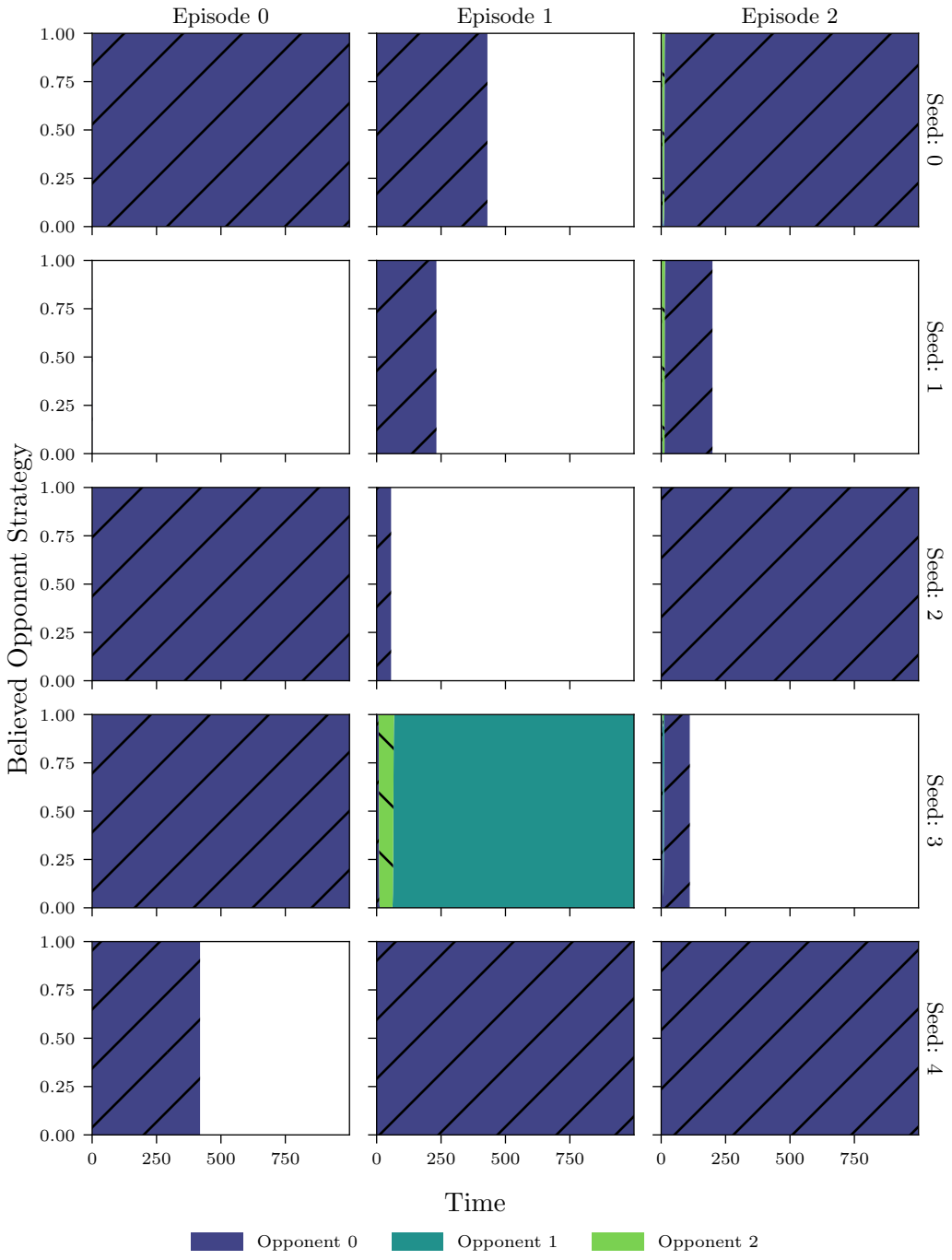


Figure 49: **Q-Mixing: OPC, Belief, Prior against Opponent 0**.

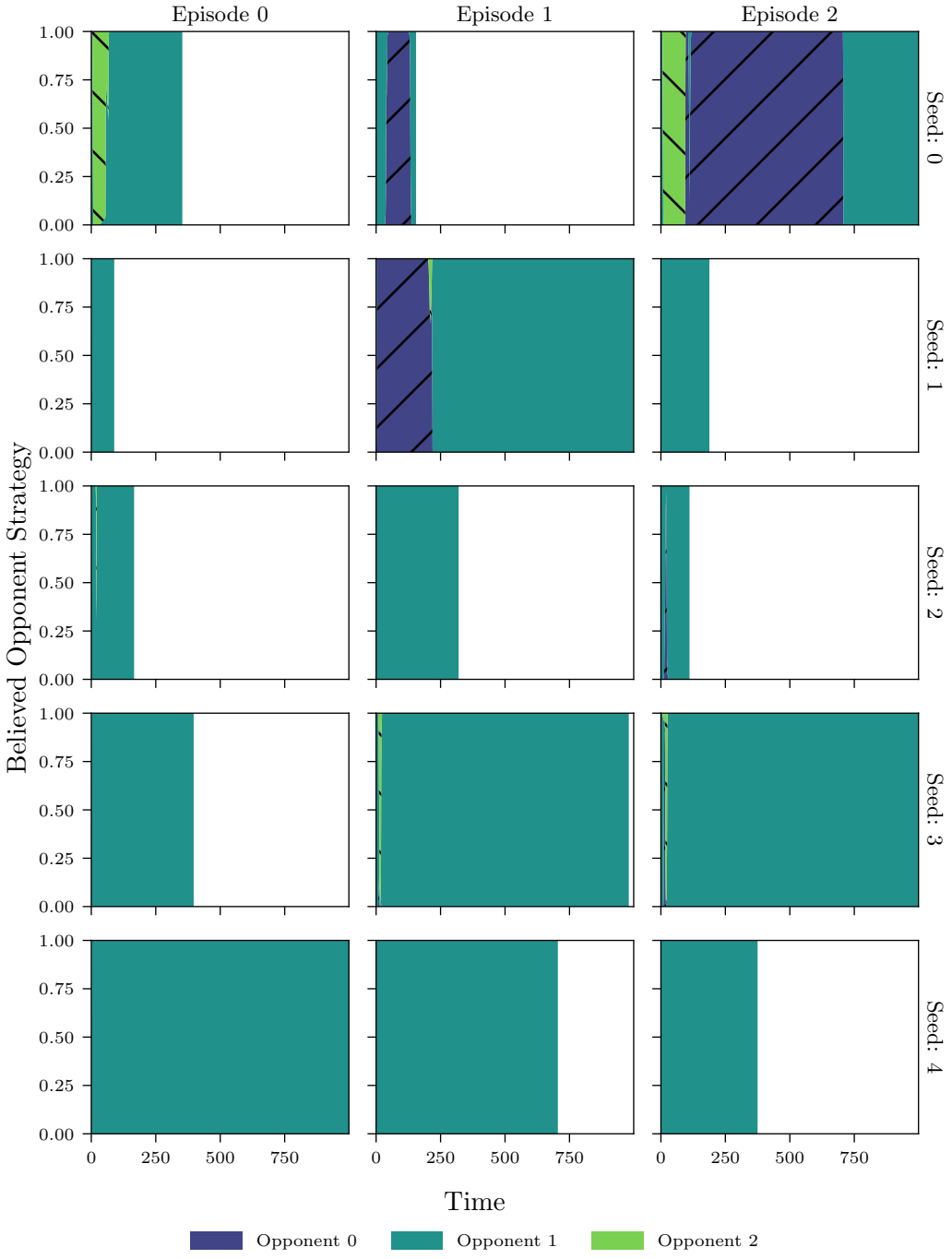


Figure 50: **Q-Mixing: OPC, Belief, Prior against Opponent 1**.

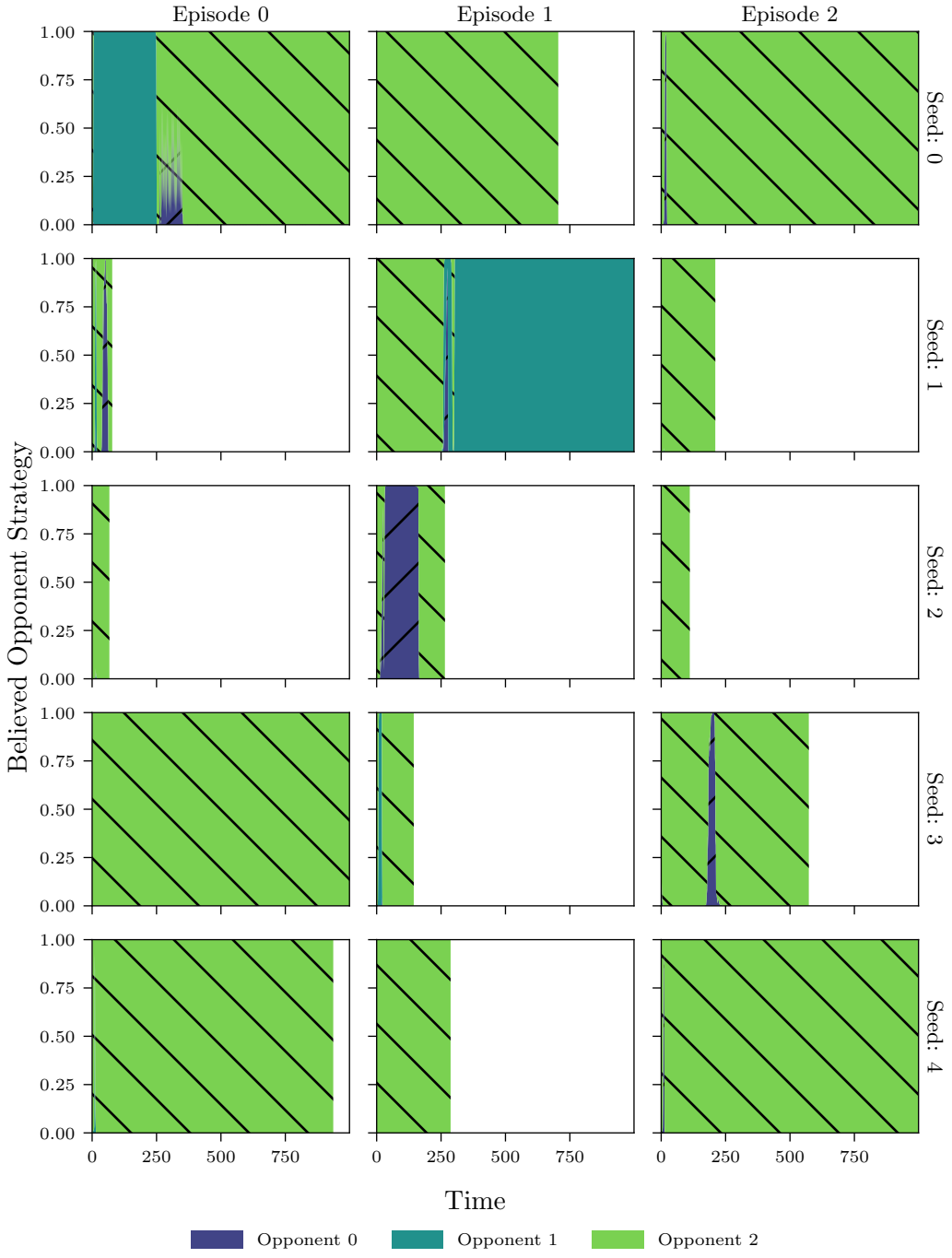


Figure 51: **Q-Mixing: OPC, Belief, Prior against Opponent 2**.

Appendix F. Games

F.1 Running With Scissors (RWS)

RWS is a temporarily extended version of RPS. The agents begin by collecting rock, paper, and scissor items scattered throughout the gird-world. These are added to the player's inventory v_i, which is initialized to have one of each item. The game ends when a player challenges the other to play RPS. Each player plays a distribution over the actions following the distribution of items in their inventory. The reward can then be calculated as:

$$r_i = \frac{v_i}{\|v_i\|} M \left(\frac{v_{-i}}{\|v_{-i}\|} \right)^{\mathrm{T}} = -r_{-i}, \qquad M = \begin{bmatrix} 0 & -1 & 1 \\ 1 & 0 & -1 \\ -1 & 1 & 0 \end{bmatrix}.$$

The game is a two-player zero-sum game with a small state and action space, enabling inexpensive simulation. The general map layout it depicted in Figure 52. In it we can see that the players randomly spawn in a fixed set of places. Moreover, items within the grid world can spawn both deterministically and stochastically. This means that if an agent does not know what spawned in a particular spot they cannot accurately infer the opponent's inventory. This is critical, because the game is partially observable. Agents are only able to view a small 5×5 sub-grid in around their position instead of the full 13×21 grid.

A particular instance of gameplay is provided in Figure 53. In this game, we can see that the starting observations of each player can view the spawn of two randomly spawned items (shown in the two right sub-grids). This gives each player private information of the state of the game.

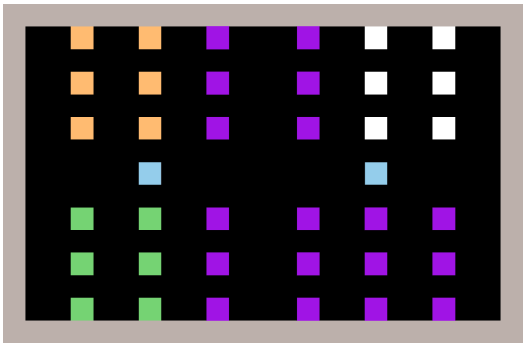


Figure 52: **RWS map layout.** The blue squares represent the possible spawn points of the players. Items either spawn deterministically as rock (orange), paper (white), scissors (green); or, one of the three possible items is randomly spawned into each position (purple). Black cells are empty, and light-gray represents walls.

F.2 Gathering (aka Harvest)

The *Gathering* environment is a common-pool resource game, where the goal of each agent is to collect apples. The apples regrow proportional to the number of nearby unharvested apples. Naturally this presents a dilemma for the players: each want to pick as many apples

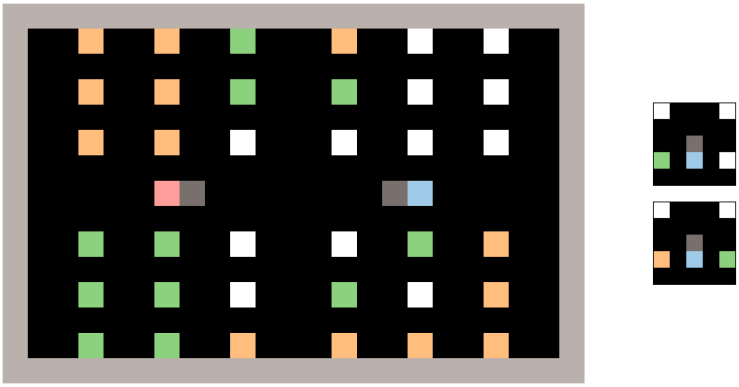


Figure 53: **RWS example observations.** (Left) The full state of the RWS game. (Right) The two player's observations. Blue is used to represent self in both observations, whereas red is used in the full state to distinguish between the two agents.

as possible; however, if they over-harvest the throughput of apples diminishes — potentially falling to zero.

The original designers of this game were interested in modelling the governance of social-ecological systems using human subjects (Janssen et al., 2010). An important feature of the human experiments is they often contained the option for a participant to pay a fee to fine another participant. Pérolat et al. (2017) adapted this game to agent-based modelling by endowing each agent with a "time-out beam" (hereafter, LASER), which serves this purpose. The LASER extends 20 tiles in front of the current agent, and has a width of 5 tiles. If an agent tags another agent with this LASER, the taggee is removed from the game for 25 timesteps.

On their quest to collect apples each agent will simultaneously select one of eight possible actions:

$$\{\text{UP, RIGHT, DOWN, LEFT, ROTATE-RIGHT, ROTATE-LEFT, LASER, NOOP}\}.$$

The first four actions represent moving in the respective cardinal directions from the perspective of the agent. The second set of two actions – ROTATE-RIGHT and ROTATE-LEFT, adjust the perspective of the agent by having them turn 90 degrees in the corresponding direction. This is an important capability of the agent because it allows the agent to aim its LASER, which only fires the direction the agent is facing.

The agent's observe a rectangular window of 10 squares forward (including their position), and 10 to the left and right (including their position). The result is a [10, 20] window, where each cell contains: food, agent, opponent, wall, or nothing. This gives our agent an observation shape of [10, 20, 4] which is ravelled into a single vector of length 80. This is processed by a two-hidden layer fully-connected neural network with 50 units at each layer and ReLU activations.

Our implementation is based on the the open source implementation: `https://github.com/HumanCompatibleAI/multi-agent`. We modified this version of the environment to remove an erroneous edge-case where both agents could time-out each other in the same

timestep. This led to degenerate solutions where the agents would effectively stun lock each other, so neither player could obtain any reward.

We investigate two versions of this game that differ in the configuration of the map (apple locations, spawn points, etc.): (1) *Gathering-Small* has all apples in a dense grove central in the map, and (2) *Gathering-Open* is a larger map with many spread-out apple groves. The former environment will be used to force two agents to interact, while the latter will allow the study of interactions between more than two players.

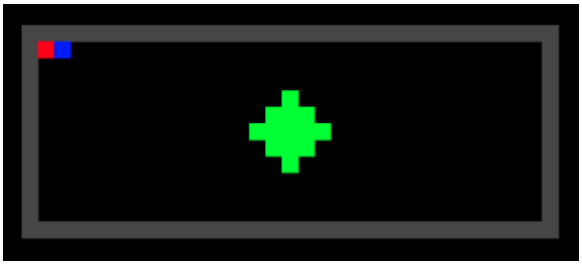
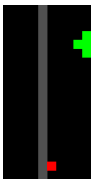


Figure 54: **The Gathering-Small environment.** Players randomly spawn in either the red or blue (one player per spawn).

(a) Player 0's perspective at the beginning of the episode.



(b) Player 0's perspective after turning right.

Figure 55: **Example observation from the Gathering-Default environment.** Note that the agents cannot distinguish between their opponents.

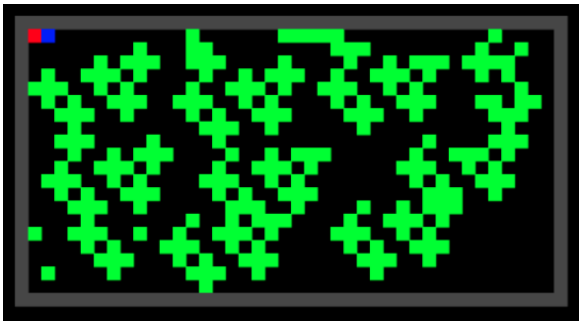


Figure 56: **The Gathering-Open environment.** Players randomly spawn in one of many locations throughout the map.