Towards Deployment of Robust Cooperative AI Agents: An Algorithmic Framework for Learning Adaptive Policies

Ahana Ghosh
MPI-SWS
gahana@mpi-sws.org

Sebastian Tschiatschek
University of Vienna
sebastian.tschiatschek@univie.ac.at

Hamed Mahdavi
MPI-SWS
hmahdavi@mpi-sws.org

Adish Singla
MPI-SWS
adishs@mpi-sws.org

ABSTRACT

We study the problem of designing an AI agent that can robustly cooperate with agents of unknown type (i.e., previously unobserved behavior) in multi-agent scenarios. Our work is inspired by real-world applications in which an AI agent, e.g., a virtual assistant, has to cooperate with new types of agents/users after its deployment. We model this problem via parametric Markov Decision Processes where the parameters correspond to a user's type and characterize her behavior. In the test phase, the AI agent has to interact with a user of an unknown type. We develop an algorithmic framework for learning adaptive policies: our approach relies on observing the user's actions to make inferences about the user's type and adapting the policy to facilitate efficient cooperation. We show that without being adaptive, an AI agent can end up performing arbitrarily bad in the test phase. Using our framework, we propose two concrete algorithms for computing policies that automatically adapt to the user in the test phase. We demonstrate the effectiveness of our algorithms in a cooperative gathering game environment for two agents.

KEYWORDS

Learning agent-to-agent interactions; Machine learning; Reinforcement learning

ACM Reference Format:
Ahana Ghosh, Sebastian Tschiatschek, Hamed Mahdavi, and Adish Singla. 2020. Towards Deployment of Robust Cooperative AI Agents: An Algorithmic Framework for Learning Adaptive Policies. In *Proc. of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2020), Auckland, New Zealand, May 9–13, 2020*, IFAAMAS, 9 pages.

1 INTRODUCTION

An increasing number of multi-agent systems are used in applications like autonomous driving [14, 31, 43], gaming [39], and energy distribution [26]. In these applications, the agents have to cooperate and show mutual awareness to achieve optimal performance [2, 11, 42]. Commonly such applications are approached by jointly training policies for all deployed agents [28]. Hence from the perspective of any particular agent, the set of other agents it might encounter is fixed or known a priori and each agent can thus account for the other agents' preferences and behaviour.

In practice, however, this is often unrealistic because in many applications we might encounter new types of agents after deployment. In particular, our work is inspired by applications like virtual assistants where an AI agent has to cooperate with other agents (i.e., users) of previously unobserved behavior after deployment. In this application setting, an AI agent that does not account for the user's preferences and behavior typically degrades the utility for the human users [2, 11, 12, 38, 42]. However, accounting for the user's characteristics is challenging because the AI agent needs to (a) infer information about the interacting user and (b) be able to interact efficiently with a large number of different users, each possibly showing different behavior. In particular, during development of an AI agent, it is often only possible to interact with a limited number of users and the AI agent needs to generalize to new types of users after deployment. These challenges resemble those in multi-agent reinforcement learning settings in which an AI agent faces unknown agents at test time [10] (including human-agent and agent-agent scenarios).

In this paper, we study the problem of designing AI agents that can robustly cooperate with agents of unknown type after deployment. More specifically, we consider a setting in which the AI agent only has access to the reward information during its development while no (explicit) reward information is available once the agent is deployed. This setting appears naturally when it is difficult/expensive/disruptive to supply or compute reward information while a multi-agent system is in use, as for instance in our motivating application of a virtual assistant. As shown in this paper, an AI agent can only achieve high utility in this setting if it is adaptive to its user while a non-adaptive AI agent can perform arbitrarily bad. We propose an algorithmic framework for designing robust adaptive policies for our considered setting and derive two such policies—one of the policy comes with strong theoretical robustness guarantees at test time, while the other is inspired by recent deep-learning approaches for RL and is easier to scale to larger problems. Both policies build upon inferring the user's properties by observing their actions and leverage these inferences to act robustly.

Another way of approaching the studied problem is within a POMDP framework in which the other agent's behavior is characterized by a latent unobserved feature of the state [8, 16]. In contrast to these POMDP based approaches, we decouple inference about the agent and action selection which makes the problem more amenable for theoretical analysis and allows us to derive rigorous performance guarantees. Our approach is also related to ideas

Proc. of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2020), B. An, N. Yorke-Smith, A. El Fallah Seghrouchni, G. Sukthankar (eds.), May 9–13, 2020, Auckland, New Zealand.

of multi-task, meta-learning, and generalization in reinforcement learning [4]. However, most of these approaches require access to reward information at test time and rarely offer theoretical guarantees for robustness (a thorough discussion of related work is presented in Section 7).

In the following, we highlight our main contributions:

- We provide a generic framework for designing robust policies for interacting with agents with unknown behavior. Our framework is amenable for theoretical analysis allowing us to prove rigorous robustness guarantees for algorithms building on our framework (Section 3).
- We propose two concrete algorithms according to our framework: AdaptPool which pre-computes a set of best-response policies and executes them adaptively based on inferences of the other agent's type; and AdaptDQN which implements adaptive policies by a neural network in combination with an inference module (Sections 4 and 5).
- We empirically demonstrate the effectiveness of our approach when facing unknown agents in a cooperative gathering game environment (Section 6).

2 THE PROBLEM SETUP

We formalize the problem of designing an AI agent that can robustly cooperate with another agent of unknown type in a reinforcement learning (RL) framework. The two agents are hereafter referred to as agent $\mathbb{U}$ and agent $\mathbb{I}$: here, agent $\mathbb{I}$ represents the AI agent whereas agent $\mathbb{U}$ could be another AI agent or a human user. Our goal is to develop a learning algorithm for agent $\mathbb{I}$ that leads to high utility even in cases when the behavior of agent $\mathbb{U}$ and its committed policy is unknown.

2.1 The setting

We model the preferences and induced behavior of agent $\mathbb{U}$ via a parametric space Θ. From agent $\mathbb{I}$'s perspective, each $\theta \in \Theta$ leads to a parameterized MDP $\mathcal{M}(\theta) := (S, A, T_\theta, R_\theta, \gamma, \mathcal{D}_0)$ consisting of the following:

- a set of states S, with $s \in S$ denoting a generic state.
- a set of actions A, with $a \in A$ denoting a generic action of agent $\mathbb{I}$.
- a transition kernel parameterized by θ as $T_\theta(s' \mid s, a)$, which is a tensor with indices defined by the current state s, the agent $\mathbb{I}$'s action a, and the next state s'. In particular, $T_\theta(s' \mid s, a) = \mathbb{E}_{a^{\mathbb{U}}}[T^{\mathbb{U},\mathbb{I}}(s' \mid s, a, a^{\mathbb{U}})]$, where $a^{\mathbb{U}} \sim \pi_\theta^{\mathbb{U}}(\cdot \mid s)$ is sampled from agent $\mathbb{U}$'s policy in state s. That is, $T_\theta(s' \mid s, a)$ corresponds to the transition dynamics derived from a two-agent MDP with transition dynamics $T^{\mathbb{U},\mathbb{I}}$ and agent $\mathbb{U}$'s policy $\pi_\theta^{\mathbb{U}}$.
- a reward function parameterized by θ as $R_\theta : S \times A \to [0, r_{\max}]$ for $r_{\max} > 0$. This captures the preferences of agent $\mathbb{U}$ that agent $\mathbb{I}$ should account for.
- a discount factor $\gamma \in [0, 1)$ weighing short-term rewards against long-term rewards.
- an initial state distribution $\mathcal{D}_0$ over S.

Our goal is to develop a learning algorithm that achieves high utility even in cases when θ is unknown. In line with the motivating applications discussed above, we consider the following two phases:

- **Training (development) phase.** During development, our learning algorithm can interact with a limited number of different MDPs $\mathcal{M}(\theta)$ for $\theta \in \Theta^{\text{train}} \subseteq \Theta$: here, agent $\mathbb{I}$ can observe rewards as well as agent $\mathbb{U}$'s actions needed for learning purposes.
- **Test (deployment) phase.** After deployment, our learning algorithm interacts with a parameterized MDP as described above for unknown $\theta^{\text{test}} \in \Theta$: here, agent $\mathbb{I}$ only observes agent $\mathbb{U}$'s actions but not rewards.

2.2 Utility of agent $\mathbb{I}$

Let us denote the set of stationary Markov policies as $\Pi = \{\pi \mid \pi : S \times A \to [0, 1]\}$. Then, for a fixed policy $\pi \in \Pi$ of agent $\mathbb{I}$, we define its total expected reward in the MDP $\mathcal{M}(\theta)$ as follows:

$$J_\theta(\pi) = \mathbb{E}\left[\sum_{\tau=1}^{\infty} \gamma^{\tau-1} R_\theta(s_\tau, a_\tau) \mid \mathcal{D}_0, T_\theta, \pi\right], \tag{1}$$

where the expectation is over the stochasticity of policy π and the transition dynamics T_θ. Here s_τ is the state at time τ; for $\tau = 1$, s_τ comes from the distribution $\mathcal{D}_0$.

For known θ. When the underlying parameter θ is known, the task of finding the best response policy of agent $\mathbb{I}$ reduces to the following:

$$\pi_\theta^* = \arg\max_{\pi \in \Pi} J_\theta(\pi). \tag{2}$$

For unknown θ. However, when the underlying parameter $\theta \in \Theta$ is unknown[1], we define the best response (in a minmax sense) policy $\pi \in \Pi$ of agent $\mathbb{I}$ as:

$$\pi_\Theta^* = \arg\min_{\pi \in \Pi} \max_{\theta \in \Theta} \left(J_\theta(\pi_\theta^*) - J_\theta(\pi)\right). \tag{3}$$

Clearly, $J_\theta(\pi_\theta^*) - J_\theta(\pi_\Theta^*) \geq 0 \ \forall \theta \in \Theta$. In general, this gap can be arbitrarily large, as formally stated in the following proposition.

Proposition 2.1. *There exists a problem instance where the performance of agent $\mathbb{I}$ can be arbitrarily worse when agent $\mathbb{U}$'s type θ^{test} is unknown. In other words, the gap $\max_{\theta \in \Theta} \left(J_\theta(\pi_\theta^*) - J_\theta(\pi_\Theta^*)\right)$ is arbitrarily high.*

The proof is available in the longer version of the paper. The proof of the above proposition is given via explicit construction of an example. Proposition 2.1 shows that the performance of agent $\mathbb{I}$ can be arbitrarily bad when it doesn't know θ^{test} and is restricted to execute a fixed stationary Markov policy. In the next section, we present an algorithmic framework for designing robust policies for agent $\mathbb{I}$ for unknown θ^{test}.

3 DESIGNING ROBUST POLICIES

In this section, we introduce our algorithmic framework for designing robust policies for the AI agent $\mathbb{I}$.

[1]Here, we assume that the parametric forms of T_θ and R_θ in the MDP $\mathcal{M}(\theta)$ are known to the agent $\mathbb{I}$, but θ itself is unknown.

Framework 1 Algorithmic framework for robust policies

Training phase

Input: parameter space Θ^{train}
adaptive policy $\psi \leftarrow \textsc{Training}(\Theta^{\text{train}})$

Test phase

Input: adaptive policy ψ
$O_0 \leftarrow ()$
for $t = 1, 2, \ldots$ **do**
  Observe current state s_t
  Estimate $\mathbb{U}$'s type as $\theta_t \leftarrow \textsc{Inference}(O_{t-1})$
  Take action $a_t \sim \psi(\cdot \mid s_t, \theta_t)$
  Observe $\mathbb{U}$'s action $a_t^{\mathbb{U}}$; $O_t \leftarrow O_{t-1} \oplus (s_t, a_t^{\mathbb{U}})$
end for

3.1 Algorithmic framework

Our approach relies on observing the behavior (i.e., actions taken) to make inferences about the agent $\mathbb{U}$'s type θ and adapting agent $\mathbb{I}$'s policy accordingly to facilitate efficient cooperation. This is inspired by how people make decisions in uncertain situations (e.g., ability to safely drive a car even if the other driver on the road is driving aggressively). The key intuition is that at test time, the agent $\mathbb{I}$ can observe agent $\mathbb{U}$'s actions which are taken as $a^{\mathbb{U}} \sim \pi_\theta^{\mathbb{U}}(\cdot \mid s)$ when in state s to infer θ, and in turn use this additional information to make an improved decision on which actions to take. More formally, we define the observation history available at the beginning of timestep t as $O_{t-1} = (s_\tau, a_\tau^{\mathbb{U}})_{\tau=1,\ldots,t-1}$ and use it to infer the type of agent $\mathbb{U}$ and act appropriately.

In particular, we will make use of an Inference procedure (see Section 5). Given O_{t-1}, this procedure returns an estimate of the type of agent $\mathbb{U}$ at time t given by $\theta_t \in \Theta$. Then, we consider stochastic policies of the form $\psi : S \times A \times \Theta \to [0, 1]$. The space of these policies is given by $\Psi = \{\psi \mid \psi : S \times A \times \Theta \to [0, 1]\}$. For a fixed policy $\psi \in \Psi$ of agent $\mathbb{I}$, we define its total expected reward in the MDP $\mathcal{M}(\theta)$ as follows:

$$J_\theta(\psi) = \mathbb{E}\left[\sum_{\tau=1}^{\infty} \gamma^{\tau-1} R_\theta(s_\tau, a_\tau) \mid \mathcal{D}_0, T_\theta, \psi\right]. \tag{4}$$

At any time t, we have $a_t \sim \psi(\cdot \mid s_t, \theta_t)$ and $O_{t-1} = (s_\tau, a_\tau^{\mathbb{U}})_{\tau=1,\ldots,t-1}$ is generated according to $a_\tau^{\mathbb{U}} \sim \pi_\theta^{\mathbb{U}}(\cdot \mid s_\tau)$.

We seek to find the policy for agent $\mathbb{I}$ given by the following optimization problem:

$$\min_{\psi \in \Psi} \max_{\theta \in \Theta} \left(J_\theta(\pi_\theta^*) - J_\theta(\psi)\right) \tag{5}$$

In the next two sections, we will design algorithms to optimize the objective in Equation (5) following the framework outlined in Framework 1. In particular, we will discuss two possible architectures for policy ψ and corresponding Training procedures in Section 4. Then, in Section 5, we describe ways to implement the Inference procedure for inferring agent $\mathbb{U}$'s type using observed actions. Below, we provide theoretical insights into the robustness of the proposed algorithmic framework.

3.2 Performance analysis

We begin by specifying three technical questions that are important to gain theoretical insights into the robustness of the proposed framework:

- Q.1 Independent of the specific procedures used for Training and Inference, the first question to tackle is the following: When agent $\mathbb{U}$'s true type is θ^{test} and agent $\mathbb{I}$ uses a best response policy for $\pi^*_{\hat{\theta}}$ such that $||\theta^{\text{test}} - \hat{\theta}|| \le \epsilon$, what are the performance guarantees on the total utility achieved by agent $\mathbb{I}$? (see Theorem 3.1).
- Q.2 Regarding Training procedure: When agent $\mathbb{U}$'s type is θ^{test} and the inference procedure outputs $\hat{\theta}$ such that $||\theta^{\text{test}} - \hat{\theta}|| \le \epsilon$, what is the performance of policy ψ? (see Section 4).
- Q.3 Regarding Inference procedure: When agent $\mathbb{U}$'s type is θ^{test}, can we infer $\hat{\theta}$ such that either $||\theta^{\text{test}} - \hat{\theta}||$ is small, or agent $\mathbb{U}$'s policies $\pi_{\hat{\theta}}^{\mathbb{U}}$ and $\pi_{\theta^{\text{test}}}^{\mathbb{U}}$ are approximately equivalent? (see Section 5).

3.2.1 Smoothness properties. For addressing Q.1, we introduce a number of properties characterizing our problem setting. These properties are essentially smoothness conditions on MDPs that enable us to make statements about the following intermediate issue: For two types θ, θ', how "similar" are the corresponding MDPs $\mathcal{M}(\theta)$, $\mathcal{M}(\theta')$ from agent $\mathbb{I}$'s point of view?

The first property characterizes the smoothness of rewards for agent $\mathbb{I}$ w.r.t. parameter θ. Formally, the parametric MDP $\mathcal{M}(\theta)$ is α-smooth with respect to the rewards if for any θ and θ' we have

$$\max_{s \in S, a \in A} |R_\theta(s, a) - R_{\theta'}(s, a)| \le \alpha \cdot r_{\max} \cdot ||\theta - \theta'||_2 \tag{6}$$

The second property characterizes the smoothness of policies for agent $\mathbb{U}$ w.r.t. parameter θ; this in turn implies that the MDP's transition dynamics as perceived by agent $\mathbb{I}$ are smooth. Formally, the parametric MDP $\mathcal{M}(\theta)$ is β-smooth in the behavior of agent $\mathbb{U}$ if for any θ and θ' we have

$$\max_{s \in S} \mathrm{KL}\big(\pi_\theta^{\mathbb{U}}(. \mid s); \pi_{\theta'}^{\mathbb{U}}(. \mid s)\big) \le \beta \cdot ||\theta - \theta'||_2. \tag{7}$$

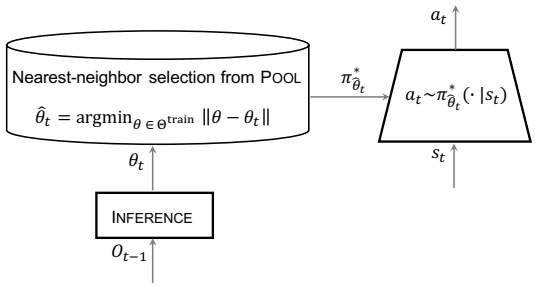
For instance, one setting where this property holds naturally is when $\pi_\theta^{\mathbb{U}}$ is a soft Bellman policy computed w.r.t. a reward function for agent $\mathbb{U}$ which is smooth in θ [17, 46].

The third property is a notion of influence as introduced by Dimitrakakis et al. [5]: This notion captures how much one agent can affect the probability distribution of the next state with her actions as perceived by the second agent. Formally, we capture the influence of agent $\mathbb{U}$ on agent $\mathbb{I}$ as follows:

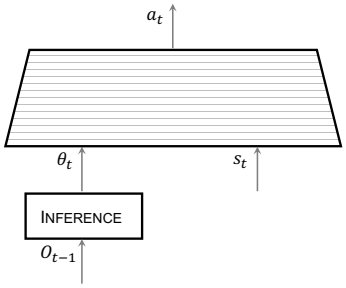
$$\mathcal{I}_{\mathbb{U}} := \max_{s \in S} \Big(\max_{a, b, b'} \left\| T^{\mathbb{U},\mathbb{I}}(. \mid s, a, b) - T^{\mathbb{U},\mathbb{I}}(. \mid s, a, b') \right\|_1 \Big), \tag{8}$$

where a represents the action of agent $\mathbb{I}$, b, b' represents two distinct actions of agent $\mathbb{U}$, and $T^{\mathbb{U},\mathbb{I}}$ is the transition dynamics of the two-agent MDP (see Section 2.1). Note that $\mathcal{I}_{\mathbb{U}} \in [0, 1]$ and this allows us to do fine-grained performance analysis: for instance, when $\mathcal{I}_{\mathbb{U}} = 0$, then agent $\mathbb{U}$ doesn't affect the transition dynamics as perceived by agent $\mathbb{I}$ and we can expect to have better performance for agent $\mathbb{I}$.

3.2.2 Guarantees. Putting this together, we can provide the following guarantees as an answer for Q.1:



(a) Test phase in Framework 1 with policy ψ trained using ADAPTPOOL procedure.



(b) Test phase in Framework 1 with policy ψ trained using ADAPTDQN procedure.

Figure 1: Two different instantiations of Framework 1 with the adaptive policy ψ trained using procedures ADAPTPOOL and ADAPTDQN. (a) ADAPTPOOL trains a set of best response policies $\{\pi^*_\theta \mid \theta \in \Theta_{\text{train}}\}$. In the test phase at time step t with θ_t as the output of INFERENCE, the action a_t is sampled from a distribution $\pi^*_{\hat{\theta}_t}(\cdot \mid s_t)$ where $\hat{\theta}_t$ is the nearest match for θ_t in the set Θ_{train}. (b) ADAPTDQN trains one deep Q-Network (DQN) with an augmented state space given by (s, θ). At time t, with θ_t as the output of INFERENCE, the DQN network is given as input a tuple (s_t, θ_t) and the network outputs an action a_t.

THEOREM 3.1. *Let $\theta^{\text{test}} \in \Theta$ be the type of agent $\mathbb{U}$ at test time and agent $\mathbb{I}$ uses a policy $\pi^*_{\hat{\theta}}$ such that $||\theta^{\text{test}} - \hat{\theta}||_2 \leq \epsilon$. The parameters $(\alpha, \beta, \mathcal{I}_{\mathbb{U}})$ characterize the smoothness as defined above. Then, the total reward achieved by agent $\mathbb{I}$ satisfies the following guarantee:*

$$J_{\theta^{\text{test}}}(\pi^*_{\hat{\theta}}) \geq J_{\theta^{\text{test}}}(\pi^*_{\theta^{\text{test}}}) - \frac{\epsilon \cdot \alpha \cdot r_{\max}}{1-\gamma} - \frac{\mathcal{I}_{\mathbb{U}} \cdot \sqrt{2 \cdot \beta \cdot \epsilon} \cdot r_{\max}}{(1-\gamma)^2}$$

The proof of the theorem is provided in the longer version of the paper. The proof builds up on the theory of approximate equivalence of MDPs by Even-Dar and Mansour [6]. In the next two sections, we provide specific instantiations of TRAINING and INFERENCE procedures.

4 TRAINING PROCEDURES

In this section, we present two procedures to train adaptive policies ψ (see TRAINING in Framework 1).

4.1 Training procedure ADAPTPOOL

The basic idea of ADAPTPOOL is to maintain a pool POOL of best response policies for $\mathbb{I}$ and, in the test phase, switch between these policies based on inference of the type θ^{test}.

4.1.1 Policy architecture of ADAPTPOOL. The adaptive pool based policy ψ consists of a pool (POOL) of best response policies corresponding to different possible agent $\mathbb{U}$'s types θ, and a nearest-neighbor policy selection mechanism. In particular, when invoking ADAPTPOOL for state s_t and inferred agent $\mathbb{U}$'s type θ_t, the policy first identifies the most similar agent $\mathbb{U}$ in POOL, i.e., $\hat{\theta}_t = \arg\min_{\theta \in \Theta^{\text{train}}} \|\theta - \theta_t\|$, and then executes an action $a_t \sim \pi^*_{\hat{\theta}_t}(\cdot \mid s_t)$ using the best response policy $\pi^*_{\hat{\theta}_t}$.

4.1.2 Training process. During training we compute a pool of best response policies POOL for a set of possible agent $\mathbb{U}$'s types Θ^{train}, see Algorithm 1.

4.1.3 Guarantees. It turns out that if the set of possible agent $\mathbb{U}$'s types Θ^{train} is chosen appropriately, Framework 1 instantiated with ADAPTPOOL enjoys strong performance guarantees. In particular, choosing Θ^{train} as a sufficiently fine ϵ'-cover of the parameter space Θ, ensures that for any $\theta^{\text{test}} \in \Theta$, that we might encounter at test time, we have considered a sufficiently *similar* agent $\mathbb{U}$ during training and hence can execute a best response policy which achieves good performance.

COROLLARY 4.1. *Let Θ^{train} be an ϵ'-cover for Θ, i.e., for all $\theta \in \Theta$, $\exists \theta' \in \Theta^{\text{train}}$ s.t. $||\theta - \theta'||_2 \leq \epsilon'$. Let $\theta^{\text{test}} \in \Theta$ be the type of agent $\mathbb{U}$ and the INFERENCE procedure outputs θ_t such that $||\theta_t - \theta^{\text{test}}||_2 \leq \epsilon''$. Let $\epsilon := \epsilon' + \epsilon''$. Then, at time t, the policy $\pi^*_{\hat{\theta}_t}$ used by agent $\mathbb{I}$ has the following guarantees:*

$$J_{\theta^{\text{test}}}(\pi^*_{\hat{\theta}_t}) \geq J_{\theta^{\text{test}}}(\pi^*_{\theta^{\text{test}}}) - \frac{\epsilon \cdot \alpha \cdot r_{\max}}{1-\gamma} - \frac{\mathcal{I}_{\mathbb{U}} \cdot \sqrt{2 \cdot \beta \cdot \epsilon} \cdot r_{\max}}{(1-\gamma)^2}$$

Corollary 4.1 follows from the result of Theorem 3.1 given that the pool of policies trained by ADAPTPOOL is sufficiently rich. Note that the accuracy ϵ'' of INFERENCE would typically improve over time and hence the performance of the algorithm is expected to improve over time in practice, see Section 6.3. Building on the idea of ADAPTPOOL, next we provide a more practical implementation of the training procedure which does not require to maintain an explicit pool of best response policies and therefore is easier to scale to larger problems.

4.2 Training procedure ADAPTDQN

ADAPTDQN builds on the ideas of ADAPTPOOL: Here, instead of explicitly maintaining a pool of best response policies for agent $\mathbb{I}$, we have a policy network trained on an augmented state space $S \times \Theta$. This policy network resembles a Deep Q-Network (DQN) architecture [20], but operates on an augmented state space and takes as input a tuple (s, θ). A similar architecture was used by Hessel et al. [13], where one policy network was trained to play 57 Atari games, and the state space was augmented with the index of the game. In the test phase, agent $\mathbb{I}$ selects actions given by this policy network.

Algorithm 1 AdaptPool: Training

Input: Parameter space Θ^{train}
Pool $\leftarrow \{\}$
for each $\theta^{\text{iter}} \in \Theta^{\text{train}}$ **do**
  $\pi^*_{\theta^{\text{iter}}} \leftarrow$ best response policy for MDP $\mathcal{M}(\theta^{\text{iter}})$
  Pool $\leftarrow$ Pool $\cup \{(\theta^{\text{iter}}, \pi^*_{\theta^{\text{iter}}})\}$
end for
return Pool

4.2.1 Policy architecture of AdaptDQN. The adaptive policy ψ consists of a neural network trained on an augmented state space $S \times \Theta$. In particular, when invoking AdaptDQN for state s_t and inferred agent $\mathbb{U}$'s type θ_t, we use the augmented state space (s_t, θ_t) as input to the neural network. The output layer of the network computes the Q-values of all possible actions corresponding to the augmented input state. Agent $\mathbb{I}$ selects the action with the maximum Q-value.

4.2.2 Training process. Here, we provide a description of how we train the policy network using the augmented state space, see Algorithm 2. During one iteration of training the policy network, we first sample a parameter $\theta^{\text{iter}} \sim \Theta^{\text{train}}$. We then obtain the optimal best response policy $\pi^*_{\theta^{\text{iter}}}$ of agent $\mathbb{I}$ for the MDP $\mathcal{M}(\theta^{\text{iter}})$. We compute the *vector* of all Q-values corresponding to this policy, i.e., $Q(s, a)\ \forall s \in S, a \in A$ (represented by $Q^{\pi^*_{\theta^{\text{iter}}}}$ in Algorithm 2), using the standard Bellman equations [36]. In our setting, we use these pre-computed Q-values to serve as the target values for the associated parameter θ^{iter} for training the policy network. The loss function used for training is the standard squared error loss between the target Q-values computed using the procedure described above and those given by the network under training. The gradient of this loss function is used for back-propagation through the network. Multiple such iterations are carried out during training, until a convergence criteria is met. For more details on Deep Q-Networks, we refer the reader to [20].

5 INFERENCE PROCEDURE

In the test phase, the inference of agent $\mathbb{U}$'s type θ^{test} from an observation history O_{t-1} is a key component of our framework, and crucial for facilitating efficient collaboration. Concretely, Theorem 3.1 implies that a best response policy $\pi^*_{\hat{\theta}}$ also achieves good performance for agent $\mathbb{U}$ with true parameter θ^{test} if $||\hat{\theta} - \theta^{\text{test}}||$ is small and MDP $\mathcal{M}(\theta)$ is smooth w.r.t. parameter θ as described in Section 3.2.

There are several different approaches that one can consider for inference, depending on the application setting. For instance, we can use probabilistic approaches as proposed in the work of Everett and Roberts [7] where a pool of agent $\mathbb{U}$'s policies $\pi^{\mathbb{U}}_\theta\ \forall\ \theta \in \Theta$ is maintained and inference is done at run time via simple probabilistic methods. Based on the work by Grover et al. [10], we can also maintain a more compact representation of agent $\mathbb{U}$'s policies and then apply probabilistic methods on this representation. Yet another approach is to consider meta-learning based approaches to build models of the encountered agents [24].

Algorithm 2 AdaptDQN: Training

Input: Parameter space Θ^{train}
$\psi \leftarrow$ Init. policy network on augmented state space
while *convergence criteria is not met* **do**
  sample $\theta^{\text{iter}} \sim$ Uniform(Θ^{train})
  $\pi^*_{\theta^{\text{iter}}} \leftarrow$ best response policy for MDP $\mathcal{M}(\theta^{\text{iter}})$
  $Q^{\pi^*_{\theta^{\text{iter}}}} \leftarrow$ Q-values for policy $\pi^*_{\theta^{\text{iter}}}$ in MDP $\mathcal{M}(\theta^{\text{iter}})$
  Train ψ for one episode:
    (i) by augmenting the state space with θ^{iter}
    (ii) by using target Q-values $Q^{\pi^*_{\theta^{\text{iter}}}}$
end while
return ψ

We can also do inference based on ideas of inverse reinforcement learning (IRL) where the observation history O_{t-1} serves the purpose of demonstrations [1, 46]. This is particularly suitable when the parameter θ exactly corresponds to the rewards used by agent $\mathbb{U}$ when computing its policy $\pi^{\mathbb{U}}_\theta$. In fact, this is the approach that we follow for our inference module in the experiments, and in particular, we employ the popular IRL algorithm, namely Maximum Causal Entropy (MCE) IRL algorithm [46]. We refer the reader to Section 6.2 for more details.

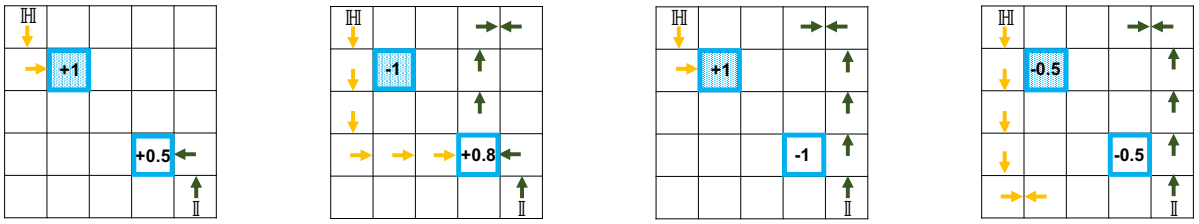
6 EXPERIMENTS

Below, we provide details of the environment, the experimental setup, and then discuss results.

6.1 Environment details

We evaluate the performance on a gathering game environment, a variant of the environments considered by Leibo et al. [18] and Raileanu et al. [27], see Figure 2. There are two agents and the objective is to maximize the total reward by collecting fruits while avoiding collisions. Agent $\mathbb{U}$ is assisted by agent $\mathbb{I}$ in achieving this objective. The environment has a 5x5 grid space resulting in 25 grid cells and the state space is determined by the joint location of agent $\mathbb{U}$ and agent $\mathbb{I}$ (i.e., $|S| = 25 \times 25$). Actions are given by A ={*'step up', 'step left', 'step down', 'step right', 'stay'*}. Each action is executed successfully with 0.8 probability; with *random move probability* of 0.2, the agent is randomly placed in one of the four neighboring cells located in vertical or horizontal positions. Two types of fruit objects are placed in two fixed grid cells (shown by 'shaded blue' and 'blue' cells). The rewards associated with these two fruit types are given by the parameter vector $\theta \in \Theta$ where $\Theta := [-1, +1]^2$. In our environment, the location of these two fruit types is fixed and fruits do not disappear (i.e., there is an unlimited supply of each fruit type in their respective locations). The discount factor γ is set to 0.99, and the initial state distribution D_0 corresponds to the agents starting in two corners.

For any fixed θ, agent $\mathbb{U}$'s policy $\pi^{\mathbb{U}}_\theta$ is computed first by ignoring the presence of agent $\mathbb{I}$—this is in line with our motivating applications where agent $\mathbb{U}$ could be a human agent with a policy agnostic to agent $\mathbb{I}$. In order to compute agent $\mathbb{U}$'s policy $\pi^{\mathbb{U}}_\theta$, we consider agent $\mathbb{U}$ operating in a single-agent MDP denoted as $\mathcal{M}^{\mathbb{U}}(\theta) = (S^{\mathbb{U}}, A, R^{\mathbb{U}}_\theta, T^{\mathbb{U}}, \gamma, D^{\mathbb{U}}_0)$ where (i) $s \in S^{\mathbb{U}}$ corresponds to the location of agent $\mathbb{U}$ in the grid-space, (ii) the action space is the



(a) $\theta = [+1, +0.5]$ **(b)** $\theta = [-1, +0.8]$ **(c)** $\theta = [+1, -1]$ **(d)** $\theta = [-0.5, -0.5]$

Figure 2: We evaluate the performance on a gathering game environment, see Section 6.1 for details. The above four illustrations correspond to four different θ parameters, highlighting agent $\mathbb{U}$'s policy $\pi_\theta^{\mathbb{U}}$ and the best response policy π_θ^* for agent $\mathbb{I}$.

same as described above, (iii) the reward function $R_\theta^{\mathbb{U}}$ corresponds to rewards associated with two fruit types given by θ, (iv) $T^{\mathbb{U}}$ corresponds to transition dynamics of agent $\mathbb{U}$ alone in the environment, (v) discount factor $\gamma = 0.99$, and (vi) $D_0^{\mathbb{U}}$ corresponds to agent $\mathbb{U}$ starting in the upper-left corner. Given $\mathcal{M}^{\mathbb{U}}(\theta)$, we compute $\pi_\theta^{\mathbb{U}}$ as a soft Bellman policy which is suitable to capture near-optimal and stochastic agent behaviour in applications [46].

From agent $\mathbb{I}$'s point of view, each θ gives rise to a parametric MDP $\mathcal{M}(\theta)$ in which agent $\mathbb{I}$ is operating in the game along with the corresponding agent $\mathbb{U}$. Transition dynamics T_θ in $\mathcal{M}(\theta)$ are obtained by marginalizing out the effect of agent $\mathbb{U}$'s policy $\pi_\theta^{\mathbb{U}}$. Reward function R_θ in $\mathcal{M}(\theta)$ corresponds to the reward associated with fruits which depends on θ; in addition to collecting fruits, agent $\mathbb{I}$ should avoid collision or close proximity to agent $\mathbb{U}$—this is inspired by the multi-agent path finding (MAPF) problem setting, see [32, 33]. This is modelled by a collision cost of -5 when agent $\mathbb{I}$ is in the same cell as agent $\mathbb{U}$, and a proximity cost of -2 when agent $\mathbb{I}$ is in one of the four neighboring cells located in vertical or horizontal positions.

For our experiments, we consider an episodic setting where two agents play the game repeatedly for multiple episodes enumerated as $e = 1, 2, \ldots$. Each episode of the game lasts for 500 steps. Now, to translate the episode count to time steps t as used in Framework 1 (line 3), we have $t = 500 \times e$ at the end of the e^{th} episode.

6.2 Experimental setup

6.2.1 Baselines and implementation details. We use three baselines to compare the performance of our algorithms: (i) Rand corresponds to picking a random $\theta \in \Theta$ and using best response policy π_θ^*, (ii) FixedMM corresponds to the fixed best response (in a minmax sense) policy in Eq. 3, and (iii) FixedBest is a variant of FixedMM and corresponds to the fixed best response (in a average sense) policy.

We implemented two variants of AdaptPool which store policies corresponding to $\epsilon' = 1$ and $\epsilon' = 0.25$ covers of Θ (see Corollary 4.1), denoted as AdaptPool$_1$ and AdaptPool$_{0.25}$ in Figure 3. Next, we give specifications of the trained policy network used in AdaptDQN. We used Θ^{train} to be a 0.25 level discretization of Θ. The trained network ψ has 3 hidden layers with leaky RELU-units (with $\alpha = 0.1$) having 64, 32, and 16 hidden units respectively, and a linear output layer with 5 units (corresponding to the size of action set $|A|$) (see [20] for more details on training Deep Q-Network). The input to the neural network is a concatenation of the location of the 2 agents, and the parameter vector θ_t, where $|\theta_t| = 2$ (this corresponds to the augmented state space described in Section 4.2). The location of each agent is represented as a one-hot encoding of a vector of length 25 corresponding to the number of grid cells Hence the length of the input vector to the neural network is $25 \times 2 + 2$ (= 52). During training, the agent $\mathbb{I}$ implemented epsilon-greedy exploratory policies (with exploration rate decaying linearly over training iterations from 1.0 to 0.01). Training lasted for about 50 million iterations.

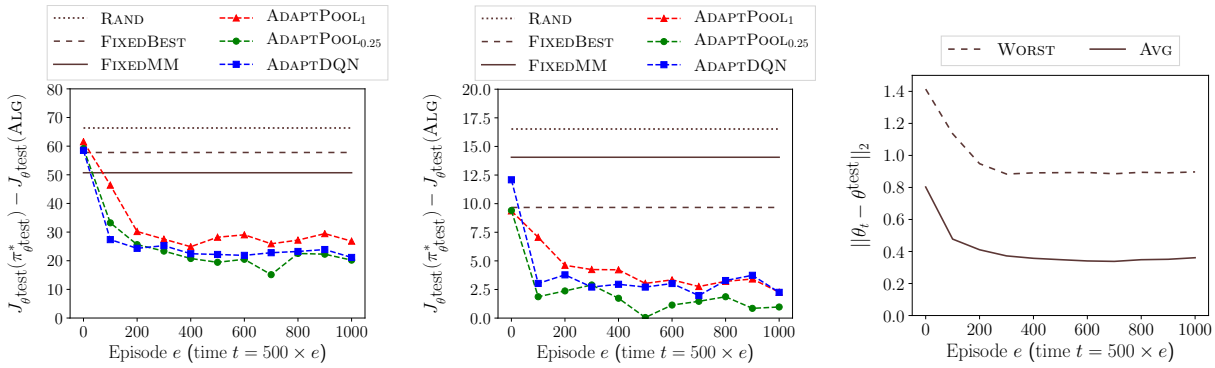
Our inference module is based on the MCE-IRL approach [46] to infer θ^{test} by observing actions taken by agent $\mathbb{U}$'s policy. Note that, we are using MCE-IRL to infer the reward function parameters θ^{test} used by agent $\mathbb{U}$ for computing its policy in the MDP $\mathcal{M}^{\mathbb{U}}(\theta^{\text{test}})$ (see Section 6.1). At the beginning, the inference module is initialized with $\theta_0 = [0, 0]$, and its output at time t given by θ_t is based on history O_{t-1}. In particular, we implemented a sequential variant of MCE-IRL algorithm which updates the estimate θ_t only at the end of every episode e using stochastic gradient descent with learning rate $\eta = 0.001$. We refer the reader to [46] for details on the original MCE-IRL algorithm and to [17] for the sequential variant.

6.3 Results

We evaluate the performance of our algorithms on 441 different θ^{test} obtained by a 0.1 level discretization of the 2-D parametric space $\Theta := [-1, +1]^2$. For a given θ^{test}, the results were averaged over 10 runs.

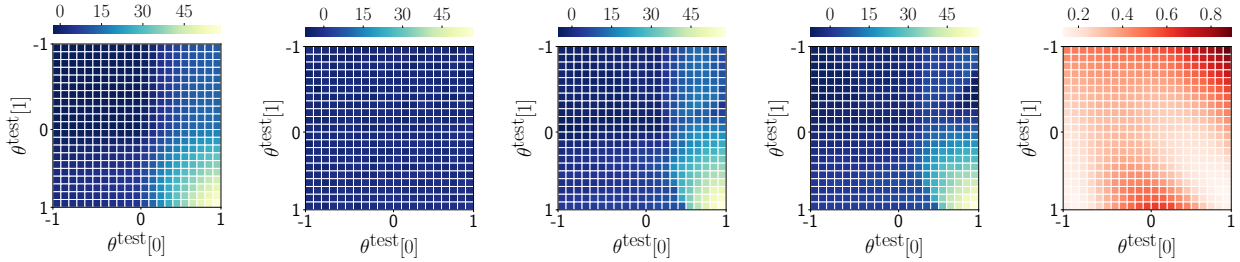
6.3.1 Worst-case and average-case performance of algorithms. Results are shown in Figure 3. As can be seen in Figure 3a, the worst-case performance of both AdaptDQN and AdaptPool is significantly better than that of the three baselines (FixedBest, Rand and FixedMM), indicating robustness of our algorithmic framework. In our experiments, the FixedMM and FixedBest baselines correspond to best response policies π_θ^* for $\theta = [0.1, -1]$ and $\theta = [0, -0.1]$ respectively. Under both these policies, agent $\mathbb{I}$'s behavior is qualitatively similar to the one shown in Figure 2c. As can be seen, under these policies, agent $\mathbb{I}$ avoids both fruits and avoids any collision; however, this does not allow agent $\mathbb{I}$ to assist agent $\mathbb{U}$ in collecting fruits even in scenarios where fruits have positive rewards.

6.3.2 Convergence of the inference module. In Figure 3c, we show the convergence behavior of the inference module. Here,



(a) Total reward: Worst-case (b) Total reward: Average-case (c) Inference module

Figure 3: (a) Worst-case performance of both AdaptDQN and AdaptPool is significantly better than that of the baselines, indicating robustness of our algorithmic framework. (a, b) Two variants of AdaptPool are shown corresponding to 1-cover and 0.25-cover. As expected, the algorithm AdaptPool$_{0.25}$ with larger pool size has better performance compared to the algorithm AdaptPool$_1$. (c) Plot shows the convergence behavior of the inference module as more observational data is gathered: Avg shows the average performance (averaged $\|\theta_t - \theta^{\text{test}}\|$ w.r.t. different θ^{test}) and Worst shows the worst case performance (maximum $\|\theta_t - \theta^{\text{test}}\|$ w.r.t. different θ^{test}).



(a) $\pi^*_{\theta^{\text{test}}}$ (b) FixedBest (c) AdaptPool$_{0.25}$ (d) AdaptDQN (e) Inference Module

Figure 4: (a, b, c, d) Heat map of the total rewards obtained by different algorithms when measured in the episode $e = 1000$. (e) Heat map of the norm $\|\theta_t - \theta^{\text{test}}\|$, i.e., the gap between the estimated and true parameter θ^{test} at the end of episode $e = 1000$. The performance of the inference procedure is poor in cases when different parameter values of θ^{test} results in agent $\mathbb{U}$ having equivalent policies. However, in these cases as well, the performance of our algorithms (AdaptPool$_{0.25}$ and AdaptDQN) is significantly better than the baselines (FixedBest).

Worst shows the worst case performance: As can be seen in the Worst line, there are cases where the performance of the inference procedure is bad, i.e., $\|\theta_t - \theta^{\text{test}}\|$ is large. This usually happens when different parameter values of θ results in agent $\mathbb{U}$ having equivalent policies. In these cases, estimating the exact θ^{test} without any additional information is difficult. In our experiments, we noted that even if $\|\theta_t - \theta^{\text{test}}\|$ is large, it is often the case that agent $\mathbb{U}$'s policies $\pi^{\mathbb{U}}_{\theta_t}$ and $\pi^{\mathbb{U}}_{\theta^{\text{test}}}$ are approximately equivalent which is important for getting a good approximation of the transition dynamics $T_{\theta^{\text{test}}}$. Despite the poor performance of the inference module in such cases, the performance of our algorithms is significantly better than that of the baselines (as is evident in Figure 3a).

6.3.3 Additional results for each individual θ^{test}. Next, we provide additional experimental results corresponding to the algorithms' performance for each individual θ^{test} to gain further insights. These results are presented in Figure 4 in the form of heat maps for each individual θ^{test}: Heat maps either represent performance of policies (in terms of the total reward $J_{\theta^{\text{test}}}(\text{Alg})$) or the performance of inference procedure (in terms of the norm $\|\theta_t - \theta^{\text{test}}\|$). These results are plotted in the episode $e = 1000$ (cf., Figure 3 where the performance was plotted over time with increasing e).

7 RELATED WORK

Modeling and inferring about other agents. The inference problem has been considered in the literature in various forms. For instance, Grover et al. [10] consider the problem of learning policy representations that can be used for interacting with unseen agents when using representation-conditional policies. They also consider the case of inferring another agent's representation (parameters) during test time. Macindoe et al. [19] consider planners for collaborative domains that can take actions to learn about the intent of another agent or hedge against its uncertainty. Nikolaidis et al. [22] cluster

human users into types and aim to infer the type of new users online, with the goal of executing the policy for that type. They test their approach in robot-human interaction but do not provide any theoretical analysis for their approach. Beyond reinforcement learning, the problem of modeling and inferring about other agents has been studied in other applications such as personalization of web search ranking results by inferring user's preferences based on their online activity [35, 40, 41].

Multi-task and meta-learning. Our problem setting can be interpreted as a multi-task RL problem in which each possible type of agent $\mathbb{U}$ corresponds to a different task, or as a meta-learning RL problem in which the goal is to learn a policy that can quickly adapt to new partners. Hessel et al. [13] study the problem of multi-task learning in the RL setting in which a single agent has to solve multiple tasks, e.g., solve all Atari games. However, they do not consider a separate test set to measure generalization of trained agents but rather train and evaluate on the same tasks. Sæmundsson et al. [30] consider the problem of meta learning for RL in the context of changing dynamics of the environment and approach it using a Gaussian processes and a hierarchical latent variable model approach.

Robust RL.. The idea of robust RL is to learn policies that are robust to certain types of errors or mismatches. In the context of our paper, mismatch occurs in the sense of encountering agents of unknown types that have not been encountered at training time and the learned policies should be robust in this situation. Pinto et al. [23] consider training of policies in the context of a *destabilizing adversary* with the goal of coping with model mismatch and data scarcity. Roy et al. [29] study the problem of RL under model mismatch such that the learning agent cannot interact with the actual test environment but only a reasonably close approximation. The authors develop robust model-free learning algorithms for this setting. Similarly, approaches for *domain randomization* aim at learning robust policies by creating a variety of simulated environments and training agents that work well on all of them [15, 37]. In this way, these approaches can be used for closing the reality gap, i.e., the gap between simulation and real-world, which is for instance often encountered in robotics due to a mismatch between simulators and the physical reality.

POMDP based approaches. The setting which we consider in this paper can also be treated within a POMDP framework in which the parameters θ represent a part of the latent state space. In such a framework, a policy can be learned which maps beliefs about the state of the systems to actions and inferences about θ correspond to computing posterior probabilities. In this spirit, Fern et al. [8] developed a decision-theoretic framework whose objective is to observe a goal-directed agent and to select assistive actions. Similarly, Javdani et al. [16] consider the problem of providing assistance to minimize the expected cost-to-go for an agent with unknown goal, and proposed methods for approximately solving the involved POMDPs which are typically intractable to solve optimally.

More complex interactions, teaching, and steering. In our paper, the type of interaction between two agents is limited as agent $\mathbb{I}$ does not affect agent $\mathbb{U}$'s behaviour, allowing us to gain a deeper theoretical understanding of this setting. There is also a related literature on "steering" the behavior of the other agent. For example, (i) the *environment design* framework of Zhang et al. [44], where one agent tries to steer the behavior of another agent by modifying its reward function, (ii) the *cooperative inverse reinforcement learning* of Hadfield-Menell et al. [11], where the human uses demonstrations to reveal a proper reward function to the AI agent, and (iii) the *advice-based interaction* model [3], where the goal is to communicate advice to a sub-optimal agent on how to act.

Dealing with non-stationary agents. The work of [7] is closely related to ours: they design a *Switching Agent Model* (SAM) that combines deep reinforcement learning with opponent modelling to robustly switch between multiple policies. Zheng et al. [45] also consider a similar setting of detecting non-stationarity and reusing policies on the fly, and introduce *distilled policy network* that serves as the policy library. Our algorithmic framework is similar in spirit to these two papers, however, in our setting, the focus is on acting optimally against an unknown agent whose behavior is stationary and we provide theoretical guarantees on the performance of our algorithms. Singla et al. [34] have considered the problem of learning with experts advice where experts are not stationary and are learning agents themselves. However, their work is focused on designing a meta-algorithm on how to coordinate with these experts and is technically very different from ours. A few other recent papers have also considered repeated human-AI interaction where the human agent is non-stationary and is evolving its behavior in response to AI agent (see [21, 25]). Prior work also considers a learner that is aware of the presence of other actors [9, 27].

8 CONCLUSIONS

Inspired by real-world applications like virtual personal assistants, we studied the problem of designing AI agents that can robustly cooperate with unknown agents in multi-agent scenarios. We focused on the important practical aspect that there is often a clear distinction between the training and test phase: the explicit reward information is only available during training but adaptation is also needed during testing. We provided a framework for designing adaptive policies and gave theoretical insights into its robustness. In experiments, we demonstrated that these policies can achieve good performance when interacting with previously unseen agents.

While we focused on collaborative tasks, we would like to point out that our algorithms are also applicable to competitive tasks, and most of our theoretical guarantees still hold. Another interesting setting is when the AI agent $\mathbb{I}$ is dealing with a human user $\mathbb{U}$ with non-stationary behavior. Our theoretical guarantees do not directly apply in this case, however, we expect a certain degree of robustness because our algorithms perform online inference of the type of $\mathbb{U}$, thereby allowing us to track changing behavior.

ACKNOWLEDGMENTS

This work was supported by Microsoft Research through its PhD Scholarship Programme. Sebastian Tschiatschek did a part of this work while being with Microsoft Research.

REFERENCES

[1] Pieter Abbeel and Andrew Y Ng. 2004. Apprenticeship learning via inverse reinforcement learning. In *International Conference on Machine Learning (ICML)*.

[2] Saleema Amershi, Dan Weld, Mihaela Vorvoreanu, Adam Fourney, Besmira Nushi, Penny Collisson, Jina Suh, Shamsi Iqbal, Paul N. Bennett, Kori Inkpen, Jaime Teevan, Ruth Kikin-Gil, and Eric Horvitz. 2019. Guidelines for human-AI interaction. In *CHI*. ACM, Article 3, 13 pages.

[3] Ofra Amir, Ece Kamar, Andrey Kolobov, and Barbara Grosz. 2016. Interactive teaching strategies for agent training. In *International Joint Conferences on Artificial Intelligence (IJCAI)*.

[4] Felipe Leno da Silva and Anna Helena Reali Costa. 2019. A Survey on Transfer Learning for Multiagent Reinforcement Learning Systems. *Journal of Artificial Intelligence Research* 64 (2019), 645–703.

[5] Christos Dimitrakakis, David C Parkes, Goran Radanovic, and Paul Tylkin. 2017. Multi-view Decision Processes: The helper-AI problem. In *Advances in Neural Information Processing Systems (NeurIPS)*.

[6] Eyal Even-Dar and Yishay Mansour. 2003. Approximate Equivalence of Markov Decision Processes. In *Learning Theory and Kernel Machines*, Bernhard Schölkopf and Manfred K. Warmuth (Eds.). Springer, Berlin, Heidelberg, 581–594.

[7] Richard Everett and Stephen J. Roberts. 2018. Learning Against Non-Stationary Agents with Opponent Modelling and Deep Reinforcement Learning. In *AAAI Spring Symposia 2018*.

[8] Alan Fern, Sriraam Natarajan, Kshitij Judah, and Prasad Tadepalli. 2014. A Decision-Theoretic Model of Assistance. *Journal of Artificial Intelligence Research* 50 (2014), 71–104.

[9] Jakob N. Foerster, Richard Y. Chen, Maruan Al-Shedivat, Shimon Whiteson, Pieter Abbeel, and Igor Mordatch. 2018. Learning with Opponent-Learning Awareness. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. 122–130.

[10] Aditya Grover, Maruan Al-Shedivat, Jayesh K. Gupta, Yuri Burda, and Harrison Edwards. 2018. Learning Policy Representations in Multiagent Systems. In *International Conference on Machine Learning (ICML)*. 1797–1806.

[11] Dylan Hadfield-Menell, Stuart J. Russell, Pieter Abbeel, and Anca D. Dragan. 2016. Cooperative Inverse Reinforcement Learning. In *Advances in Neural Information Processing Systems (NeurIPS)*.

[12] Luis Haug, Sebastian Tschiatschek, and Adish Singla. 2018. Teaching inverse reinforcement learners via features and demonstrations. In *Advances in Neural Information Processing Systems (NeurIPS)*. 8464–8473.

[13] Matteo Hessel, Hubert Soyer, Lasse Espeholt, Wojciech Czarnecki, Simon Schmitt, and Hado van Hasselt. 2019. Multi-Task Deep Reinforcement Learning with PopArt. In *AAAI*. 3796–3803.

[14] Yeping Hu, Alireza Nakhaei, Masayoshi Tomizuka, and Kikuo Fujimura. 2019. Interaction-aware Decision Making with Adaptive Strategies under Merging Scenarios. *arXiv preprint arXiv:1904.06025* (2019).

[15] Stephen James, Paul Wohlhart, Mrinal Kalakrishnan, Dmitry Kalashnikov, Alex Irpan, Julian Ibarz, Sergey Levine, Raia Hadsell, and Konstantinos Bousmalis. 2019. Sim-to-real via sim-to-sim: Data-efficient robotic grasping via randomized-to-canonical adaptation networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 12627–12637.

[16] Shervin Javdani, Henny Admoni, Stefania Pellegrinelli, Siddhartha S. Srinivasa, and J. Andrew Bagnell. 2018. Shared autonomy via hindsight optimization for teleoperation and teaming. *The International Journal of Robotics Research* 37, 7 (2018), 717–742.

[17] Parameswaran Kamalaruban, Rati Devidze, Volkan Cevher, and Adish Singla. 2019. Interactive Teaching Algorithms for Inverse Reinforcement Learning. In *International Joint Conference on Artificial Intelligence (IJCAI)*.

[18] Joel Z. Leibo, Vinícius Flores Zambaldi, Marc Lanctot, Janusz Marecki, and Thore Graepel. 2017. Multi-agent Reinforcement Learning in Sequential Social Dilemmas. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. 464–473.

[19] Owen Macindoe, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. 2012. Pomcop: Belief space planning for sidekicks in cooperative games. In *AIIDE*.

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

[21] Stefanos Nikolaidis, Swaprava Nath, Ariel D Procaccia, and Siddhartha Srinivasa. 2017. Game-theoretic modeling of human adaptation in human-robot collaboration. In *International Conference on Human-Robot Interaction (HRI)*. 323–331.

[22] Stefanos Nikolaidis, Ramya Ramakrishnan, Keren Gu, and Julie A. Shah. 2015. Efficient Model Learning from Joint-Action Demonstrations for Human-Robot Collaborative Tasks. In *International Conference on Human-Robot Interaction (HRI)*. 189–196.

[23] Lerrel Pinto, James Davidson, Rahul Sukthankar, and Abhinav Gupta. 2017. Robust Adversarial Reinforcement Learning. In *International Conference on Machine Learning (ICML)*. 2817–2826.

[24] Neil C. Rabinowitz, Frank Perbet, H. Francis Song, Chiyuan Zhang, S. M. Ali Eslami, and Matthew Botvinick. 2018. Machine Theory of Mind. In *International Conference on Machine Learning (ICML)*. 4215–4224.

[25] Goran Radanovic, Rati Devidze, David Parkes, and Adish Singla. 2019. Learning to Collaborate in Markov Decision Processes. In *International Conference on Machine Learning (ICML)*. 5261–5270.

[26] Md Shihanur Rahman and AMT Oo. 2017. Distributed multi-agent based coordinated power management and control strategy for microgrids with distributed energy resources. *Energy conversion and management* 139 (2017), 20–32.

[27] Roberta Raileanu, Emily Denton, Arthur Szlam, and Rob Fergus. 2018. Modeling Others using Oneself in Multi-Agent Reinforcement Learning. In *International Conference on Machine Learning (ICML)*. 4254–4263.

[28] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. 2018. QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. In *International Conference on Machine Learning*. 4292–4301.

[29] Aurko Roy, Huan Xu, and Sebastian Pokutta. 2017. Reinforcement Learning under Model Mismatch. In *Advances in Neural Information Processing Systems (NeurIPS)*. 3043–3052.

[30] Steindór Sæmundsson, Katja Hofmann, and Marc Peter Deisenroth. 2018. Meta Reinforcement Learning with Latent Variable Gaussian Processes. In *UAI*.

[31] Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. 2016. Safe, multi-agent, reinforcement learning for autonomous driving. *arXiv preprint arXiv:1610.03295* (2016).

[32] Guni Sharon, Roni Stern, Ariel Felner, and Nathan R Sturtevant. 2015. Conflict-based search for optimal multi-agent pathfinding. *Artificial Intelligence* 219 (2015), 40–66.

[33] David Silver. 2005. Cooperative Pathfinding. *AIIDE* 1 (2005), 117–122.

[34] Adish Singla, Seyed Hamed Hassani, and Andreas Krause. 2018. Learning to Interact With Learning Agents. In *AAAI*. 4083–4090.

[35] Adish Singla, Ryen W White, Ahmed Hassan, and Eric Horvitz. 2014. Enhancing personalization via search activity attribution. In *SIGIR*. 1063–1066.

[36] Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction*. MIT press.

[37] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. 2017. Domain randomization for transferring deep neural networks from simulation to the real world. In *International Conference on Intelligent Robots and Systems (IROS)*. 23–30.

[38] Sebastian Tschiatschek, Ahana Ghosh, Luis Haug, Rati Devidze, and Adish Singla. 2019. Learner-aware Teaching: Inverse Reinforcement Learning with Preferences and Constraints. In *Advances in Neural Information Processing Systems (NeurIPS)*.

[39] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. 2019. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature* (2019), 1–5.

[40] Ryen W White, Wei Chu, Ahmed Hassan, Xiaodong He, Yang Song, and Hongning Wang. 2013. Enhancing personalized search by mining and modeling task behavior. In *International Conference on World Wide Web (WWW)*. 1411–1420.

[41] Ryen W White, Ahmed Hassan, Adish Singla, and Eric Horvitz. 2014. From devices to people: Attribution of search activity in multi-user settings. In *International Conference on World Wide Web (WWW)*. 431–442.

[42] H James Wilson and Paul R Daugherty. 2018. Collaborative intelligence: Humans and AI are joining forces. *Harvard Business Review* 96, 4 (2018), 114–123.

[43] Jiachen Yang, Alireza Nakhaei, David Isele, Hongyuan Zha, and Kikuo Fujimura. 2018. CM3: Cooperative Multi-goal Multi-stage Multi-agent Reinforcement Learning. *arXiv preprint arXiv:1809.05188* (2018).

[44] Haoqi Zhang, David C Parkes, and Yiling Chen. 2009. Policy teaching through reward function learning. In *Conference on Electronic Commerce (EC)*. 295–304.

[45] Yan Zheng, Zhaopeng Meng, Jianye Hao, Zongzhang Zhang, Tianpei Yang, and Changjie Fan. 2018. A Deep Bayesian Policy Reuse Approach Against Non-Stationary Agents. In *Advances in Neural Information Processing Systems (NeurIPS)*. 962–972.

[46] Brian D. Ziebart. 2010. *Modeling Purposeful Adaptive Behavior with the Principle of Maximum Causal Entropy*. Ph.D. Dissertation. Advisor(s) Bagnell, J. Andrew.