

Zero-Shot Coordination in Ad Hoc Teams with Generalized Policy Improvement and Difference Rewards

Rupal Nigam

*The Grainger College of Engineering
University of Illinois Urbana-Champaign
Champaign, IL 61820
rupaln2@illinois.edu*

Niket Parikh

*The Grainger College of Engineering
University of Illinois Urbana-Champaign
Champaign, IL 61820
niketnp2@illinois.edu*

Hamid Osooli

*The Grainger College of Engineering
University of Illinois Urbana-Champaign
Champaign, IL 61820
hosooli2@illinois.edu*

Mikihisa Yuasa

*The Grainger College of Engineering
University of Illinois Urbana-Champaign
Champaign, IL 61820
myuasa2@illinois.edu*

Jacob Heglund

*The Grainger College of Engineering
University of Illinois Urbana-Champaign
Champaign, IL 61820
jheglun2@illinois.edu*

Huy T. Tran

*The Grainger College of Engineering
University of Illinois Urbana-Champaign
Champaign, IL 61820
huytran1@illinois.edu*

Abstract—Real-world multi-agent systems may require ad hoc teaming, where an agent must coordinate with other previously unseen teammates to solve a task in a zero-shot manner. Prior work often either selects a pretrained policy based on an inferred model of the new teammates or pretrains a single policy that is robust to potential teammates. Instead, we propose to leverage *all* pretrained policies in a zero-shot transfer setting. We formalize this problem as an ad hoc multi-agent Markov decision process and present a solution that uses two key ideas, generalized policy improvement and difference rewards, for efficient and effective knowledge transfer between different teams. We empirically demonstrate that our algorithm, Generalized Policy Improvement for Ad hoc Teaming (GPAT), successfully enables zero-shot transfer to new teams in three simulated environments: cooperative foraging, predator-prey, and Overcooked. We also demonstrate our algorithm in a real-world multi-robot setting.

Index Terms—ad hoc teaming, zero-shot coordination, reinforcement learning

I. INTRODUCTION

Ad hoc teaming (AHT) is an open challenge for multi-agent systems, in which an autonomous agent must successfully coordinate with other unknown agents [1]. Consider a search-and-rescue mission where robots are deployed from different organizations and expected to cooperate with each other on the fly—these robots may have different biases in how they achieve a given objective (e.g., risky vs. risk-averse search) or have different capabilities (e.g., sensing vs. manipulation). Adapting to such differences would enable agents to effectively and autonomously complete tasks where the team is unknown prior to deployment. Here, we focus on *zero-shot coordination* (ZSC) for AHT, where the controlled agent, or the *learner*, is able to pretrain with various teams but then must coordinate with a new team with no online learning [2].

Type-based approaches are a prominent class of methods for AHT, where the learner pretrains with a set of potential teammate types, then infers the best pretrained policy to use with the new team at test time [3]–[5]. However, these methods struggle to handle new teams not seen during pretraining and require online inference of the new team type. An alternative approach aims to pretrain a learner that is robust to new teams through careful generation of diverse pretraining teams [6]–[8]. However, these methods often require large training populations, may suffer from overfitting, and can struggle to generalize to out-of-distribution teammates [9]. Generating many training teammates may also be infeasible for real-world applications and would require great amounts of computational and hardware resources.

We address these challenges through two key ideas. Our first idea is to leverage a library of pretrained learner policies, but instead of choosing one at test time based on the inferred team type, we dynamically leverage the whole library with *no online inference or learning*. We specifically use a *generalized policy improvement* (GPI) policy to select from pretrained policies given the current state, motivated by the fact that GPI can guarantee improvement over library policies in a transfer learning setting [10]. However, this guarantee is only valid when the dynamics are constant and the reward function differs—for AHT, dynamics now change due to new teammate behaviors, while the reward stays the same. Our second idea is then to use *difference rewards* to define the value functions of pretrained policies used by a GPI policy. Difference rewards address the multi-agent credit assignment problem by approximating the contribution of an individual agent to a team reward [11]. We use this idea to reduce the impact of the distribution shift induced by new teammates on a GPI policy.

We integrate these ideas into an end-to-end algorithm for

ZSC in AHT and demonstrate the benefits of our method in three simulated environments and a real-world multi-robot setting. We summarize our contributions as follows:

- 1) we formalize a problem set up for ZSC in AHT (Section IV),
- 2) we propose an algorithm for ZSC in AHT based on GPI with difference rewards (Section V),
- 3) we empirically demonstrate the benefits of our method relative to baselines in three simulated environments and demonstrate its use in a multi-robot system (Section VI).

II. RELATED WORK

Type-based approaches to AHT use a pretrained library of policies, where the learner selects an appropriate response based on inferred teammate behavior. For instance, [3] updates a prior over the pretrained policies through online inference with the new teammate to determine the most likely teammate type, and acts using the corresponding learner policy. Similarly, [4] and [12] infer a teammate’s policy using a similarity metric and select a complementary learner policy from the library to coordinate in Team Space Fortress. [13] leverages Gibbs sampling to update a distribution over possible learner policies in Hanabi and similarly use this distribution to select a pretrained policy. [5] extends these methods by employing a mixture-of-experts approach to mix pretrained policies in the Overcooked environment, rather than selecting a single policy. Their method first identifies behaviors through unsupervised clustering of previously collected data, trains learner policies for each behavior, and then uses online samples to update belief weights over each policy for a mixture-of-experts model. However, these methods require observations with the unseen teammate to appropriately infer the best response from the pretrained library, and thus are not truly zero-shot. Many of them also implement a single pretrained policy, limiting their ability to combine pretrained skills at execution.

Another class of methods focuses on training a single robust policy by generating a diverse training pool of teammates for ZSC. [6] achieves this by breaking symmetries inherent in the task through random relabeling of the teammate’s states and actions. [14] generates a diverse set of training teammates by using different random seeds and checkpoints during the training of agents. [8] expands this concept by searching over the reward space to construct a training pool. [7] enhances diversity in training teammates using an entropy bonus and employing a prioritization strategy to select training teammates. Lastly, [15] ensures diversity in teammate policies by considering intrinsic rewards with random navigation. However, these methods can also be prone to overfitting and may struggle to generalize to teammates outside of the training pool [9]. Additionally, many of these methods require large training pools, resulting in high computational costs. Finally, it may not be possible to define pretraining teammates in certain real-world settings, due to, for example, limited resources and access to robot platforms. Instead, it may be more practical in certain settings for a few representative teammates to be given to the learner.

We illustrate these two common approaches for AHT in Figure 1, alongside our proposed approach. Other AHT approaches are complementary to this paper. [16] and [17] address AHT with the additional challenge of partial observability, where teammate actions and environment rewards are unobservable, so the learner updates its belief prior over the library using online observations. [18] investigates the benefit of communication between teammates, while in this work we assume no communication between agents. [19] investigates the adaptation ability of AHT approaches in a few-shot, rather than zero-shot, coordination setting. [20] leverages graph structures to handle ad hoc teams where agents can enter and leave the team. [21] considers extending AHT to settings where multiple learners are present. We assume fixed teams with a single learner. [22] consider AHT with humans, but focus on the benefits of incorporating explainable AI. Finally, self-play (SP) methods [23] have been successful in zero-sum settings [24], which are a type of AHT problem, but are ill-suited for cooperative AHT because they cannot coordinate effectively with non-SP agents [6], [25].

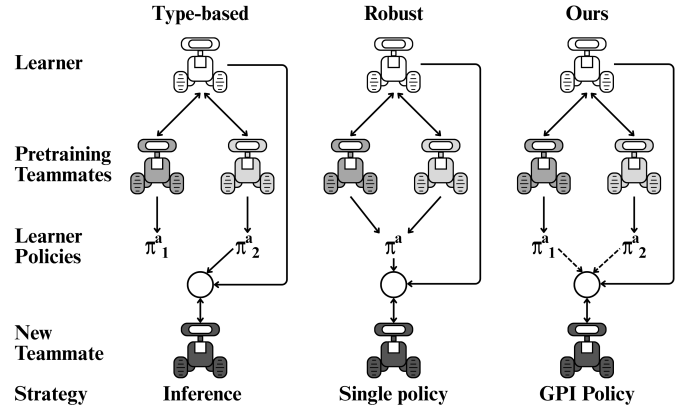


Fig. 1: Common methods for AHT either select a pretrained policy based on the inferred behavior of the teammate (Type-based) or train a single policy that is robust to potential teammates by generating a diverse training pool (Robust). We instead dynamically select which policy to use at every timestep without inference for ZSC.

III. PRELIMINARIES

A. Multi-agent Reinforcement Learning

We model our problem as a multi-agent Markov decision process (MMDP) defined by a tuple $\langle \mathcal{S}, \mathcal{N}, \{\mathcal{A}^i\}_{i \in \mathcal{N}}, p, r, \gamma \rangle$ [26]. Here, $\mathcal{S}$ is the state space, $\mathcal{N}$ is the set of agents, $\mathcal{A}^i$ is the action space of agent i , and $\gamma \in [0, 1]$ is the discount factor. Let $\mathcal{A} := \times_{i \in \mathcal{N}} \mathcal{A}^i$ be the joint action space. Then $p : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ is the state transition function and $r : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$ is the team reward function. At time step t , each agent $i \in \mathcal{N}$ executes an action $a_t^i \in \mathcal{A}^i$ given the current state $s_t \in \mathcal{S}$, after which the system transitions to state $s_{t+1} \in \mathcal{S}$ and the team receives reward $r(s_t, \mathbf{a}_t, s_{t+1})$, where $\mathbf{a}_t \in \mathcal{A}$ is the joint action. Let $\pi^i : \mathcal{S} \times \mathcal{A}^i \rightarrow [0, 1]$ be

an individual policy for agent i and $\pi(\mathbf{a}|s) := \prod_{i \in \mathcal{N}} \pi^i(a^i|s)$ be the resulting joint policy of all agents. The performance of a joint policy π can be described by its action-value function,

$$Q^\pi(s, \mathbf{a}) := \mathbb{E}_{p, \pi} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, \mathbf{a}_t, s_{t+1}) \mid s_0 = s, \mathbf{a}_0 = \mathbf{a} \right]. \quad (1)$$

B. Generalized Policy Improvement

Consider a set of source tasks $\mathcal{R} = \{r_i\}_{i=1}^n$, where each task $r \in \mathcal{R}$ is defined as a linear reward,

$$r(s, \mathbf{a}, s') = \phi(s, \mathbf{a}, s')^\top \mathbf{w}, \quad (2)$$

where $\phi : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}^d$ is a function mapping to d features and $\mathbf{w} \in \mathbb{R}^d$ is a weight vector specifying preferences over features. Following [10], define the successor features (SFs) of a policy π as,

$$\psi^\pi(s, \mathbf{a}) := \mathbb{E}_{p, \pi} \left[\sum_{t=0}^{\infty} \gamma^t \phi(s_t, \mathbf{a}_t, s_{t+1}) \mid s_0 = s, \mathbf{a}_0 = \mathbf{a} \right]. \quad (3)$$

Then, the action-value function of π on task r , $Q_r^\pi(s, \mathbf{a})$, can be represented as,

$$Q_r^\pi(s, \mathbf{a}) = \psi^\pi(s, \mathbf{a})^\top \mathbf{w}. \quad (4)$$

Now assume that we pretrain an agent on the set of source tasks $\mathcal{R}$ to generate a set of optimal policies $\Pi = \{\pi_i^*\}_{i=1}^n$, where π_i^* is an optimal policy for task $r_i \in \mathcal{R}$. Given a new target task $r_{n+1} = \phi(s, \mathbf{a}, s')^\top \mathbf{w}_{n+1}$, we can use a GPI policy, π' , defined as,

$$\pi'(s) \in \operatorname{argmax}_{\mathbf{a} \in \mathcal{A}} \max_{\pi \in \Pi} Q_{r_{n+1}}^\pi(s, \mathbf{a}), \quad (5)$$

to perform no worse than any policy in Π on this task. If we compute the set of SFs associated with each policy in Π , $\Psi = \{\psi^{\pi_i^*}\}_{i=1}^n$, we can compute the set $\{Q_{r_{n+1}}^{\pi_i^*}\}_{i=1}^n$ using Equation (4) and implement π' with no additional learning on the new target task. If additional learning is allowed, we can use π' to quickly optimize a policy for r_{n+1} , for example using SFQL (Algorithm 3 in [27]). We use GPI as a framework for ZSC in AHT.

C. Difference Rewards

In cooperative MARL, multiple agents interact within a shared environment to achieve a common goal, represented as a shared, team reward. A core challenge in this setting is the multi-agent credit assignment problem: determining which agent(s) were responsible for the reward resulting from their collective actions. Difference rewards offer a solution to this by approximating an individual agent's contribution to the team reward [11], [28]. Instead of a team reward signal, each agent computes their individual difference reward and optimizes a policy to maximize its expected return. More formally, the difference reward for agent i , Δr^i , is defined as,

$$\Delta r^i(s, \mathbf{a}, s') := r(s, \mathbf{a}, s') - \mathbb{E}_{b^i \sim \pi^i} [r(s, \langle \mathbf{a}^{-i}, b^i \rangle, s')], \quad (6)$$

where $\mathbf{a}^{-i}$ is the joint action of all agents other than i . We extend the concept of difference rewards to the AHT setting so the learner can overcome the credit assignment problem.

IV. PROBLEM FORMULATION

We modify the general formulation of MMDPs for AHT as follows. Let $\mathbf{a} \in \mathcal{N}$ be the *learner* (i.e., the agent whose policy we aim to optimize) and $\mathcal{N}_u = \mathcal{N} \setminus \{\mathbf{a}\}$ be the complementary set of all teammates (i.e., uncontrolled agents). We assume that each teammate follows a fixed policy, which is unknown to the learner. Teammate policies may be suboptimal with respect to the team reward r and the ad hoc team considered due to, e.g., the teammates being trained for a different task or with different teammates, or being humans and having inherent biases towards different goals. We formally define this model as an ad hoc MMDP.

Definition 1 (Ad Hoc MMDP). An ad hoc MMDP is defined by a tuple $M := \langle \mathcal{S}, \mathcal{N}, \mathbf{a}, \{\mathcal{A}^i\}_{i \in \mathcal{N}}, p, r, \{\pi^i\}_{i \in \mathcal{N}_u}, \gamma \rangle$, where $\mathbf{a} \in \mathcal{N}$ is the learner, $\mathcal{N}_u = \mathcal{N} \setminus \{\mathbf{a}\}$ is the complementary set of teammates, and $\pi^i : \mathcal{S} \times \mathcal{A}^i \rightarrow [0, 1]$ is the fixed policy of teammate i .

We assume that the reward function, r , is non-negative. Note that any bounded reward can be transformed into a non-negative reward because scalar addition renders the reward to be policy invariant. We refer to an ad hoc MMDP M as an ad hoc team. The performance of a learner policy $\pi^{\mathbf{a}}$ in ad hoc team M can be described by its action-value function,

$$Q^{\pi^{\mathbf{a}}, \pi^{-\mathbf{a}}}(s, \mathbf{a}^{\mathbf{a}}) := \mathbb{E}_{p, \pi^{\mathbf{a}}, \pi^{-\mathbf{a}}} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, \mathbf{a}_t, s_{t+1}) \mid s_0 = s, \mathbf{a}_0^{\mathbf{a}} = \mathbf{a}^{\mathbf{a}} \right], \quad (7)$$

where $\pi^{-\mathbf{a}}$ is the joint policy of all teammates induced by $\{\pi^i\}_{i \in \mathcal{N}_u}$. Our objective is to compute an optimal learner policy, $\pi^{\mathbf{a}}$, which satisfies,

$$Q^{\pi^{\mathbf{a}}, \pi^{-\mathbf{a}}}(s, \mathbf{a}^{\mathbf{a}}) := \max_{\pi^{\mathbf{a}}} Q^{\pi^{\mathbf{a}}, \pi^{-\mathbf{a}}}(s, \mathbf{a}^{\mathbf{a}}), \quad (8)$$

for all $s \in \mathcal{S}$ and $\mathbf{a}^{\mathbf{a}} \in \mathcal{A}^{\mathbf{a}}$. Optimizing a learner policy for an ad hoc team M is equivalent to solving a single-agent Markov decision process with a transition function $\tilde{p}$ that captures the impact of teammate policies $\pi^{-\mathbf{a}}$.

Assume we are given a partially specified ad hoc team $M_{\mathcal{N}_u} = \langle \mathcal{S}, \cdot, \mathbf{a}, \mathcal{A}^{\mathbf{a}}, p, r, \cdot, \gamma \rangle$ and m possible ad hoc teammates $\{\langle \mathcal{A}^i, \pi^i \rangle\}_{i=1}^m$. Then let $\mathcal{M}$ be the set of possible ad hoc teams induced by those teammates. Given such a set, we formalize our ZSC in AHT problem as follows.

Problem 1 (Zero-shot Coordination for AHT). Let $\mathcal{M}_0 = \{M_i\}_{i=1}^n \subseteq \mathcal{M}$ be a given set of source ad hoc teams with which the learner can pretrain. Our objective is to synthesize an optimal learner policy $\pi_{n+1}^{\mathbf{a}}$ for a new ad hoc team $M_{n+1} \in \mathcal{M} \setminus \mathcal{M}_0$ by leveraging information from pretraining on $\mathcal{M}_0$ but with no online learning with the new team M_{n+1} .

V. OUR METHOD

A. Key Ideas: Generalized Policy Improvement and Difference Rewards

We address the AHT problem defined in Problem 1 through two key ideas. First, we use a GPI policy of the form in Equation (5) to *dynamically* leverage a library of pretrained learner policies to coordinate with a new ad hoc team with no online learning. By dynamic, we mean that GPI allows the learner to use different pretrained policies throughout the execution of a single episode, rather than being restricted to only using the best-matching pretrained policy (as in type-based methods) or a single robust policy (as in robust pretraining AHT methods). This ability to dynamically leverage policies is important, for example, in scenarios where a learner must use multiple pretrained skills to complete a task. Furthermore, this approach does not require online inference.

We are also motivated by the fact that a GPI policy guarantees improvement over each library policy in single-agent zero-shot transfer settings where dynamics are fixed and rewards are changed [27]. However, in our setting, new ad hoc teammates induce new dynamics due to their (potentially) different policies, while our team rewards are fixed. These new dynamics make the set of pretrained SFs no longer valid, which prevents instant evaluation of the value functions for the new task that GPI requires. To ensure policy improvement, one would need to perform policy evaluation for *each pretrained policy* with respect to the new ad hoc team, which requires many online samples.

We address this issue through our second idea—instead of having GPI operate over value functions of pretrained policies evaluated with respect to the team reward, we have it operate over value functions with respect to the learner’s difference rewards. Note that these value functions still assume teammate dynamics associated with the ad hoc teams used during pretraining; that is, we do not correct them to account for the new ad hoc team and therefore policy improvement is still not guaranteed. However, we hypothesize that evaluating with respect to the learner’s difference rewards emphasizes contributions of the learner towards the team reward (while correspondingly de-emphasizing contributions of teammates), which will then reduce the impact of the distribution shift induced by the new ad hoc team on the actions selected by the GPI policy. This idea is illustrated in Figure 2.

B. An Algorithm for ZSC in AHT

Based on the ideas presented in Section V-A, we propose an algorithm, GPI for Ad Hoc Teaming (GPAT), to address ZSC in AHT. Our algorithm is composed of three primary steps, visualized in Figure 3 and discussed below. Pseudocode is provided in Algorithm 1. We consider two settings, one where the team reward r can be modeled as a linear reward and the more general case of any team reward r . For the linear reward setting, we assume the features ϕ are fixed and heuristically defined such that r can be modeled using Equation (2) through a weight vector $\mathbf{w}$. Our algorithm can

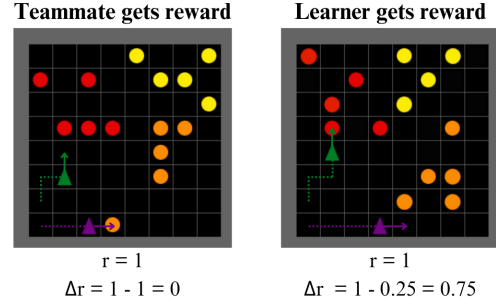


Fig. 2: Difference rewards allow the learner to estimate its own contribution to the team reward. We illustrate this in a foraging example where collecting any object gives a team reward of 1. (left) the teammate action results in the reward and (right) the learner action results in the reward. While the team reward is the same in both scenarios, Δr reflects the learner’s contribution. We use this idea to reduce the effect of the distribution shift induced by new teammates.

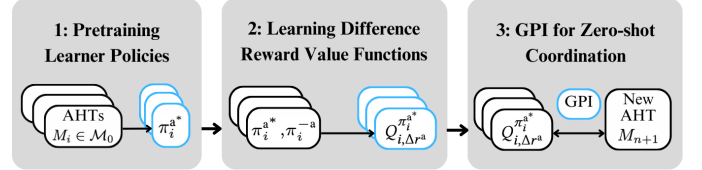


Fig. 3: Our method, GPAT, addresses ZSC in AHT through 3 steps: (1) pretraining learner policies, (2) learning difference reward value functions to address the distribution shift induced by new teammates, and (3) dynamically leveraging this library at every time step by applying a GPI policy.

be extended to incorporate feature and weight learning, as in [29], [30].

Step 1: Pretraining Learner Policies.: Given a set of source ad hoc teams $\mathcal{M}_0$, we first optimize a learner policy for each source ad hoc team. The output of this step is a library of learner policies $\Pi^a = \{\pi_i^a\}_{i=1}^n$, where π_i^a is the optimal learner policy for ad hoc team $M_i \in \mathcal{M}_0$. Any single-agent RL algorithm can be used for this step. In this work, we used Q-Learning with SFs (SFQL, Algorithm 3 in [27]) for linear reward settings. This process produces a set of optimal learner SFs $\Psi^a = \{\psi^{\pi_i^a, \pi_i^{-a}}\}_{i=1}^n$, where the learner SFs for learner policy π^a in ad hoc team M_i are defined as,

$$\psi^{\pi^a, \pi_i^{-a}}(s, a^a) = \mathbb{E}_{p, \pi^a, \pi_i^{-a}} \left[\sum_{t=0}^{\infty} \gamma^t \phi(s_t, \mathbf{a}_t, s_{t+1}) \mid s_0 = s, a_0^a = a^a \right], \quad (9)$$

where π_i^{-a} is the joint policy of all teammates in ad hoc team M_i . Following Equation (9), we use these learner SFs to define corresponding action-value functions for policies in Π^a as,

$$Q^{\pi^a, \pi_i^{-a}}(s, a^a) = \psi^{\pi^a, \pi_i^{-a}}(s, a^a)^\top \mathbf{w}. \quad (10)$$

We refer to $\psi^{\pi^a, \pi_i^{-a}}$ as $\psi_i^{\pi^a}$ and $Q^{\pi^a, \pi_i^{-a}}$ as $Q_i^{\pi^a}$ hereafter to simplify notation, where $\psi_i^{\pi^a}$ and $Q_i^{\pi^a}$ are the SFs and action-

Algorithm 1 Generalized Policy Improvement for Ad Hoc Teaming

Step 1: Pretraining learner policies

Require: $\Pi^{-a} \equiv \{\pi_1^{-a}, \dots, \pi_n^{-a}\}$

- 1: **for** team $\in 1, \dots, i$ **do**
- 2: **if** linear reward **then**
- 3: Optimize π_i^a with π_i^{-a} using SFQL or SFDQN
- 4: **else if** general reward **then**
- 5: Optimize π_i^a with π_i^{-a} using any RL algorithm

Step 2: Learning difference reward value functions

Require: $\Pi^a \equiv \{\pi_1^a, \dots, \pi_n^a\}, \Pi^{-a}, r(s, \mathbf{a}), \phi(s, \mathbf{a})$

- 1: **for** team $\in 1, \dots, i$ **do**
- 2: **for** timestep $\in 1, \dots, T_{DR}$ **do**
- 3: $s \leftarrow \text{env.reset}()$
- 4: **while** not done **do**
- 5: $\mathbf{a} \leftarrow \pi_i^a, \pi_i^{-a}$
- 6: $s', r \leftarrow \text{env.step}(\mathbf{a})$
- 7: $\Delta r \leftarrow \text{compute_dr}(r, a^{-a}) \triangleright \text{Equation (6)}$
- 8: **if** linear reward **then**
- 9: $\mathcal{D} \leftarrow \Delta r, \phi(s, \mathbf{a})$
- 10: **else if** general reward **then**
- 11: $\delta \leftarrow \Delta r + \gamma Q_{i, \Delta r}(s', \pi^a(s')) - Q_{i, \Delta r}(s, a^a)$
- 12: $\theta \leftarrow \theta + \alpha \delta \nabla_{\theta} Q_{i, \Delta r}(s, a^a)$
- 13: **if** linear reward **then**
- 14: $w_{i, \Delta r} \leftarrow \text{least_squares}(\mathcal{D}) \triangleright \text{Equation (2)}$

Step 3: GPI for zero-shot coordination

Require: $\mathcal{Q}_{\Delta r^a} \equiv \{Q_{1, \Delta r^a}, \dots, Q_{n, \Delta r^a}\}, \pi_{n+1}^{-a}$

- 1: $s \leftarrow \text{env.reset}()$
 - 2: **while** not done **do**
 - 3: $a^a \leftarrow \arg\max_b \max_i Q_{i, \Delta r^a}^{\pi_i^*}(s, b) \triangleright \text{Equation (11)}$
 - 4: $\mathbf{a} \leftarrow (a^a, \pi^{-a}(s))$
 - 5: $s, r \leftarrow \text{env.step}(\mathbf{a})$
-

value function for learner policy π^a in ad hoc team $M_i \in \mathcal{M}$. We used PPO [31] for general reward settings.

Step 2: Learning Difference Reward Value Functions. Given the library of pretrained learner policies Π^a , we now perform policy evaluation with respect to the learner’s difference reward Δr^a , rather than the team reward r . Because the learner policies are deterministic, we assume a uniform learner policy when computing the difference rewards as in [32]. The output of this step is a set $\mathcal{Q}_{\Delta r^a}^a = \{Q_{i, \Delta r^a}^{\pi_i^*}(s, a^a)\}_{i=1}^n$, where $Q_{i, \Delta r^a}^{\pi_i^*}$ is the value function of policy π_i^a with respect to the learner’s difference reward in $M_i \in \mathcal{M}$.

Linear Reward Setting. We model the learner’s difference reward as $\Delta r^a(s, \mathbf{a}, s') = \phi(s, \mathbf{a}, s')^\top \mathbf{w}_{\Delta r^a}$. Given this model, Ψ^a , and rollouts from the optimal learners in each ad hoc team $M_i \in \mathcal{M}_0$, we can now approximate $\mathcal{Q}_{\Delta r^a}^a$ by simply estimating $\mathbf{w}_{\Delta r^a}$ with linear regression and using Equation (10). We used Step 2 from Algorithm 1 for this work, and show that a sufficiently accurate $\mathbf{w}_{\Delta r^a}$ can be learned in **as few as 10 episodes** in Section VI. Note that this policy evaluation step can be performed during Step 1 using the same sampled

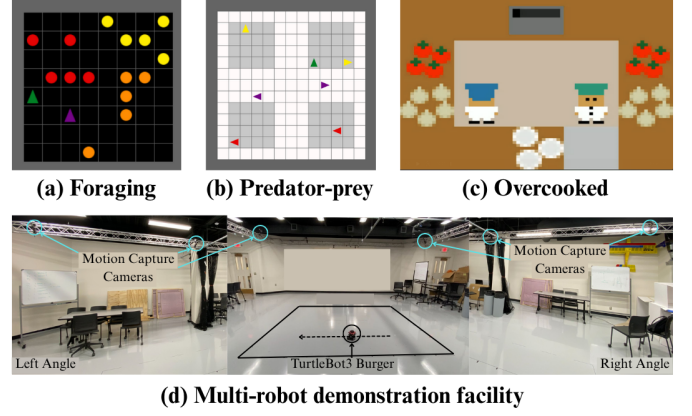


Fig. 4: Multi-agent environments used in our experiments, where the learner agent is green. (a) Circles represent different types of objects agents aim to collect. (b) Agents aim to capture easy (yellow) and hard (red) prey. (c) Agents must coordinate to cook and serve a tomato onion soup. (d) Our multi-robot demonstration uses TurtleBot3 Burgers.

experiences when using linear rewards—we simply separate this step for presentation purposes.

General Reward Setting. Any policy evaluation method can be used to estimate $\mathcal{Q}_{\Delta r^a}^a$ in the general reward setting. We use a TD-learning approach outlined in Step 2 from Algorithm 1 for this work, which is a simplified version of fitted Q-iteration (FQI) algorithms [33], [34]. This process can be computationally expensive, but allows one to model a broader class of rewards.

Step 3: GPI for Zero-shot Coordination. We finally use a GPI policy for ZSC in a target (new) ad hoc team M_{n+1} . Given $\mathcal{Q}_{\Delta r^a}^a$, we define the GPI policy for the learner as,

$$\pi^a(s) \in \arg\max_{a^a \in \mathcal{A}^a} \max_{i \in \{1, \dots, n\}} Q_{i, \Delta r^a}^{\pi_i^*}(s, a^a). \quad (11)$$

VI. EXPERIMENTS

A. Experimental Setup

Environments. We empirically demonstrate GPAT’s performance in a multi-agent foraging environment inspired by [16], [29], a multi-agent predator-prey environment as in [16], [35], and Overcooked [25], [36]. The environments are illustrated in Figure 4. We assume linear rewards for all environments, though we also consider a general reward setting for foraging within our ablation study.

The foraging environment has 2 agents in a team (one learner and one teammate) that aim to collect 3 types of objects (red, orange, and yellow) in an 8×8 grid. We define environment features ϕ as 1×3 vectors, where each element of ϕ is the number of objects of a given type that are collected in a state transition. The team reward is then defined using $\mathbf{w} = [1, 1, 1]$. Following [29], our state representations are agent-centric and toroidal, such that the agent is always in the upper left corner and the grid is wrapped around the edges of the environment. The representation has 5 channels: one for

TABLE I: Foraging experiment descriptions. The reward weights and objects collected vectors correspond to [red, orange, yellow]. Skills from pretrained learners that are useful for the new AHT, and used by the oracle, are bolded. We design experiments to test 3 scenarios with differing percentages of useful pretrained skills.

		Source AHT 1		Source AHT 2		New AHT	
		Teammate	Learner	Teammate	Learner	Teammate	Oracle
<i>Experiment 1</i> (50% useful prior skills)	Reward weights	[1.0, -0.5, -0.5]	[1.0, 1.0, 1.0]	[-0.5, 1.0, -0.5]	[1.0, 1.0, 1.0]	[-0.5, -0.5, 1.0]	[1.0, 1.0, 1.0]
	Objects collected	[4.9, 0.0, 0.0]	[0.1, 5.0, 4.9]	[0.0, 4.6, 0.0]	[5.0, 0.2, 4.9]	[0.1, 0.1, 4.8]	[4.9, 4.9, 0.1]
	Preferred objects	red	orange , yellow	orange	red , yellow	yellow	red , orange
<i>Experiment 2</i> (100% useful prior skills)	Reward weights	[0.0, 1.0, 1.0]	[1.0, 1.0, 1.0]	[1.0, 0.0, 1.0]	[1.0, 1.0, 1.0]	[-0.5, -0.5, 1.0]	[1.0, 1.0, 1.0]
	Objects collected	[0.0, 4.9, 4.5]	[5.0, 0.1, 0.5]	[5.0, 0.0, 4.8]	[0.0, 5.0, 0.2]	[0.1, 0.1, 4.8]	[4.9, 4.9, 0.1]
	Preferred objects	orange, yellow	red	red, yellow	orange	yellow	red , orange
<i>Experiment 3</i> (0% useful prior skills)	Reward weights	[0.0, 1.0, 1.0]	[1.0, 1.0, 1.0]	[1.0, 0.0, 1.0]	[1.0, 1.0, 1.0]	[1.0, 1.0, 0.0]	[1.0, 1.0, 1.0]
	Objects collected	[0.0, 4.9, 4.5]	[5.0, 0.1, 0.5]	[5.0, 0.0, 4.8]	[0.0, 5.0, 0.2]	[4.8, 2.8, 0.2]	[0.1, 2.2, 4.7]
	Preferred objects	orange, yellow	red	red, yellow	orange	red, orange	yellow

each object type, one for the teammate, and one for walls. We cluster each object type in separate quadrants of the grid. Within each cluster, objects spawn in random locations at the start of each episode, and agents spawn in the lower left quadrant.

The predator-prey environment has 3 agents in a team (predators) that aim to capture 4 prey in a 13×13 grid. Prey randomly move within their shaded regions. We consider easy prey (yellow), which can be captured by a single predator, and hard prey (red), which must be captured by two predators. We define environment features and state representations similar to those in the foraging environment, with the team reward defined using $\mathbf{w} = [1, 1, 1, 1]$.

The Overcooked environment has 2 agents in a team that aim to cook and deliver soups in a 2×3 room—we use the Cramped Room layout, modified to include an additional soup ingredient, as in [8], to increase coordination complexity (frequent interactions due to the small room size) and task complexity, as recommended in [9]. The agents must place an onion in the pot, place a tomato in the pot, start up the stove, wait for the soup to finish cooking, plate the soup, and finally deliver the soup. Following prior work [8], we define environment features to consider 5 rewarding events (potting onion, potting tomato, picking up a dish, picking up the soup, delivering the soup) and define the team reward as $\mathbf{w} = [3, 3, 3, 5, 20]$. We use the default fully-observable state representation provided in the published environment.

Teammate Policies.: For the foraging environment, we consider 2 source ad hoc teams in each experiment, where each experiment considers different combinations of source and new target ad hoc teams. We design teammate policies to have different preferences for which object types (colors) they collect—the ideal learner collects the objects ignored by its teammate. We optimize teammate policies using SFQL [29]. Table I defines our source and new target ad hoc teams, including statistics for the average number of object types each teammate collects in an episode. We also include statistics for the resulting pretrained learners and an oracle learner that is able to train with the new target ad hoc team.

For the predator-prey environment, we consider 2 source ad hoc teams, where the teammates use heuristic greedy policies designed so that the agents have preferences for which prey they target—similar to the foraging environment, the ideal learner targets prey ignored by its teammates.

The teammate in Overcooked follows heuristic greedy policies designed such that the agent has preferences for performing specific tasks for the recipe. Thus, the ideal learner performs the complementary tasks so the soup recipe can be completed and delivered.

Learner Policies.: For the foraging and predator-prey environments, the learner policies were trained using SFQL [29] with simple multilayer perceptron networks with two hidden layers of sizes 64 and 128. We used ϵ -greedy exploration during training with $\epsilon = 0.1$. Updates were done with batch sizes of 10 with a discount factor of $\gamma = 0.95$ and a learning rate of $\alpha = 3 \times 10^{-5}$. All policies were trained for 2,500,000 timesteps. For the more complex Overcooked environment, we implemented an SF-DQN algorithm using the Stable Baselines3 DQN algorithm [37]. We used the default hyperparameters, except for $\gamma = 0.95$, and 2,500,000 timesteps.

Baselines.: We implement GPAT with linear rewards in our experiments (unless otherwise noted) and learn w_{Δ_r} with 10 episodes, based on preliminary experiments ranging from 10-100 evaluation episodes. We compare GPAT to the following baselines:

- *Oracle*: We train a learner agent from scratch (using SFQL or SFDQN) with the new target ad hoc team to represent an oracle learner.
- *Robust*: We pretrain a learner agent with all source ad hoc teams (where the team is randomly sampled during training) to represent robust ZSC methods like [6]–[8]. For a fair comparison, we train robust learners (using SFQL or SFDQN) for the total timesteps used to train the entire learner policy libraries used by other methods. We assume the source ad hoc teams are given to the learner, so we do not consider the source ad hoc team generation process.

TABLE II: Experiment results, showing IQM returns for GPAT compared to baselines [3], [6], [8], measured using 1000 episodes with 10 replicates and 95% confidence intervals. The percentage optimality with respect to the oracle is also reported in parentheses.

	Foraging			Predator-Prey	Overcooked
	Experiment 1	Experiment 2	Experiment 3		
Oracle	8.087 $\pm$ 0.011	8.082 $\pm$ 0.011	8.091 $\pm$ 0.012	2.547 $\pm$ 0.008	607.76 $\pm$ 0.735
GPAT (ours)	7.755 $\pm$ 0.014(95.9%)	7.635 $\pm$ 0.019(94.5%)	5.641 $\pm$ 0.024(69.7%)	2.202 $\pm$ 0.011(86.5%)	218.07 $\pm$ 5.401(35.9%)
Robust	5.998 $\pm$ 0.018(74.2%)	6.590 $\pm$ 0.016(81.5%)	6.573 $\pm$ 0.016(81.2%)	2.077 $\pm$ 0.011(81.5%)	33.05 $\pm$ 2.078(5.4%)
PLASTIC	6.438 $\pm$ 0.010(79.6%)	6.235 $\pm$ 0.012(77.1%)	6.238 $\pm$ 0.013(77.1%)	2.096 $\pm$ 0.008(82.3%)	9.69 $\pm$ 0.328(1.6%)

TABLE III: Ablation results in the foraging environment, showing IQMs returns for GPAT with a linear reward (ours), general reward (GR), and without difference rewards (w/o DR), measured using 1000 episodes with 10 replicates and 95% confidence intervals. We also report the percentage optimality and the percentage each pretrained policy is used.

	Experiment 1			Experiment 2			Experiment 3		
	Return	% π_1	% π_2	Return	% π_1	% π_2	Return	% π_1	% π_2
GPAT (ours)	7.755 $\pm$ 0.014(95.9%)	56.5%	43.5%	7.635 $\pm$ 0.019(94.5%)	50.1%	49.9%	5.641 $\pm$ 0.024(69.7%)	10.0%	90.0%
GPAT with GR	7.266 $\pm$ 0.022(89.8%)	56.2%	43.8%	7.492 $\pm$ 0.021(92.7%)	45.2%	54.8%	6.282 $\pm$ 0.024(77.6%)	5.6%	94.4%
GPAT w/o DR	7.066 $\pm$ 0.019(87.4%)	34.7%	65.3%	6.625 $\pm$ 0.019(82.0%)	72.4%	27.6%	6.206 $\pm$ 0.015(76.7%)	22.6%	77.4%

- *PLASTIC*: We use the best pretrained learner policy from the library Π^a to represent a best-case scenario for type-based methods like [3], [5], [13]. We skip the online inference phase and simply use the best pretrained policy to mimic a setting where inference is already performed and correct.

B. Results

Our results show inter-quartile mean (IQM) evaluation returns for the learner with the new target partner with 95% confidence intervals (CIs), following the recommendations from [38]. Higher IQM scores are better. The CIs are estimated using percentile bootstrap with stratified sampling with 1,000 bootstrap resamples. We saw similar trends when analyzing mean and median results (not shown), and thus only show IQM results because it is more robust to outliers than the mean and has less bias than the median [38].

Foraging.: We evaluate GPAT in the three experiments shown in Table I. Experiment 1 requires the learner to leverage one of the two skills learned from each pretrained policy in order to optimally coordinate with the new teammate. Experiment 2 requires the learner to leverage all skills learned during pretraining. Experiment 3 contains no relevant skills in the learner’s pretrained policies. Table II shows results for these experiments. We see that GPAT (with a linear reward) outperforms all baselines (other than the oracle) for Experiments 1 and 2, likely because it is able to effectively use both pretrained skills. In comparison, the Robust baseline is likely struggling to generalize its pretrained skills to the new out-of-distribution teammate. The PLASTIC baseline is limited to using the best-matching pretrained policy and thus can only implement one of the needed skills. However, GPAT performs worse than both baselines in Experiment 3, likely due to the learner’s pretrained library not containing the skill

needed to coordinate with the new teammate (i.e., collecting yellow objects). In principle, GPAT could perform similarly to PLASTIC, since it’s library contains the best-matching policy used by PLASTIC—however, we see that GPAT is outperformed by PLASTIC, indicating that it occasionally picks the worse policy, likely due to inaccuracies incurred by the distribution shift induced by the new teammate. Note that our ablation results (shown in Table III) show that GPAT with general rewards performs similarly to our baselines in Experiment 3. We discuss this point more in the ablation discussion. Overall, these experiments suggest that GPAT can effectively achieve ZSC when its library has at least some relevant skills, but can struggle when there are no relevant skills in the library.

Predator-Prey.: In the predator-prey experiment, the first pretrained learner policy learns to target the top-right and bottom-right prey, while the second pretrained learner policy learns to target the top-left and bottom-left prey. To coordinate optimally with the new AHT, the learner must target the top-left and bottom-right prey, thus combining both pretrained skills. As shown in Table II, we see that GPAT outperforms our baselines due to its ability to correctly use both pretrained policies as needed. Thus, GPAT can coordinate well with multiple teammates and handle dynamic environments.

Overcooked.: In Overcooked, the first pretrained learner policy trains with an onion-preferring teammate to learn to pot tomatoes, cook the soup, plate the soup, and deliver the soup. The second pretrained learner policy trains with a tomato-preferring teammate to learn to pot onions, cook the soup, plate the soup, and deliver the soup. To coordinate optimally with the new dish-preferring teammate, the learner must pot both onions and tomatoes before cooking the soup. Table II shows that GPAT significantly outperforms our baselines. Compared to other environments, there is also a greater optimality gap

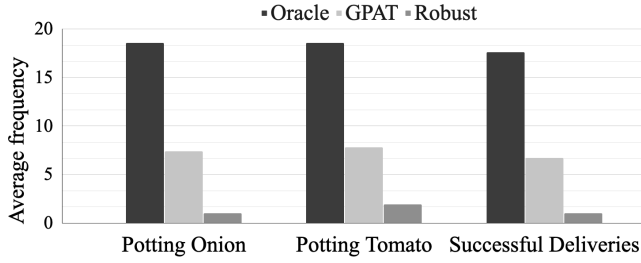


Fig. 5: The average number of times the learner performs various subtasks over an episode, measured using 1000 episodes with 10 replicates. The Robust learner struggles to pot onions and tomatoes, despite learning how to optimally perform both during pretraining.

between all methods and the oracle policy, demonstrating the difficulty of the task. We also see a larger improvement in performance between GPAT and our baselines, suggesting that the method can scale, and may be more effective than alternatives for more complex tasks.

We further investigate why the Robust learner struggles to perform well, despite converging to optimal returns during pretraining with library teammates. Figure 5 visualizes the number of times the learner pots onions or pots tomatoes, along with how many deliveries the team successfully made. We observe that the Robust learner does not perform the required subtasks often enough, indicating that it may have overfit to the training teammates and does not know how to optimally use those skills with the new teammate.

Ablation Results. Table III shows results from an ablation study for GPAT. We used PPO from Stable Baselines3 [37] to train learner policies with general rewards, and performed policy evaluation using TD-updates (see Algorithm 1) with 2500 episodes, based on preliminary experiments ranging from 100-5000 episodes. We see that GPAT with linear rewards outperforms general rewards in Experiments 1 and 2, likely due to easier reward learning. This, along with the much greater sample efficiency of learning $w_{\Delta r}$ compared to learning $Q_{\Delta r}$, is why we use GPAT with linear rewards in the main results. However, GPAT with general rewards outperforms linear rewards in Experiment 3, where it performs similarly to the Robust and PLASTIC baselines. Recall that in Experiment 3, the pretraining library does not contain any relevant skills for the new teammate. Thus, we hypothesize that the noisier policy learned by general rewards is actually advantageous in this scenario.

We also see that removing difference rewards decreases performance. Analyzing the pretrained policy use in Table III, we see that GPAT without DR uses one pretrained policy much more than the other, which further supports the idea that GPI without DR is unable to switch policies as needed. In the ideal scenario for Experiments 1 and 2, we expect the learner to select each pretrained policy a roughly equal number of times. Figure 6 shows an example state where GPAT without DR is unable to switch between policies. In

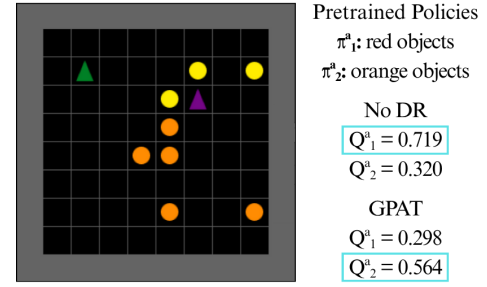


Fig. 6: Q-values for a state from Experiment 2 in the foraging environment, normalized by the maximum for each method. Without difference rewards, the learner is unable to correctly switch from the first pretrained policy to the second once all red objects have been collected.

this state, all red objects have been collected. Since the new teammate prefers collecting yellow objects, the learner should switch to collecting orange (using Q_2^a). However, GPAT without DR is unable to appropriately switch policies because it overestimates the value of π_1^a , since its value functions are with respect to the team reward and thus assume the teammate will collect the remaining objects.

We further investigate the importance of difference rewards through Figure 7, which visualizes normalized value maps of the oracle learner, GPAT without DR, and GPAT, given a fixed initial state for Experiment 2. We observe that our method without difference rewards overestimates the value of collecting the yellow objects, which is the new teammate's preference. Once again, this overestimation is because the value function is with respect to the team reward. We also compute the average percent error of the values with respect to the oracle's, and see that the learner without DR has a higher error than GPAT. Thus, DR results in more aligned values despite the new dynamics induced by the new teammate.

C. Real World Multi-Robot Demonstration

We also demonstrate our method in a real-world multi-robot setting using Robotis Turtlebot3 Burgers in the foraging environment. We employ two TurtleBot3 Burger robots, each running a separate instance of the Robot Operating System (ROS) and maintaining its own identification parameters. A Qualysis motion capture system, with four passive markers attached to the robots, provides precise localization within a 12 ft. $\times$ 12 ft. grid-world. At every time step, we use GPAT to generate control commands, which are then transmitted to the robots through ROS. The trajectories the robots take are visualized in Figure 8. We observe the learner robot (green) collects the red and orange objects while the teammate robot (purple) collects the yellow objects, as expected. A video illustrating these real-world experiments is available online.¹

¹Robot video: <https://tinyurl.com/5546zndk>

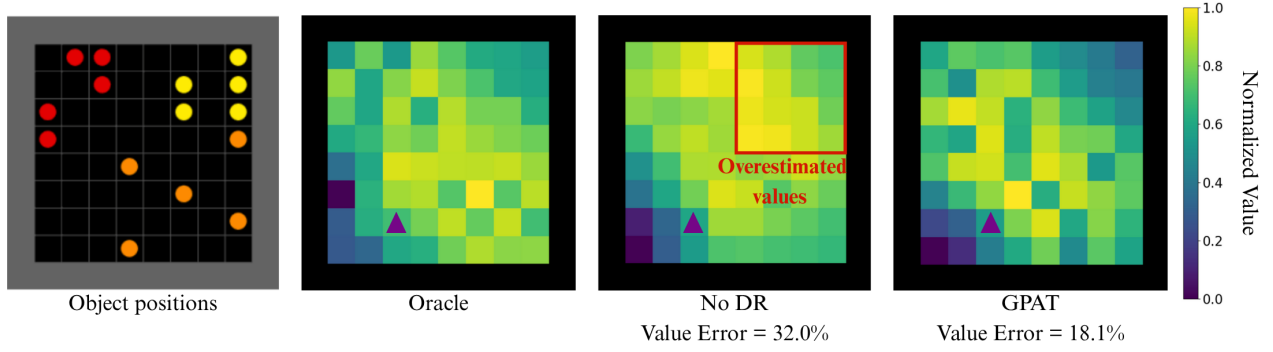


Fig. 7: Normalized value maps for an initial state in the foraging environment for Experiment 2, where we fix the teammate in the bottom-left corner and show the value of the learner being in each possible grid position. We also provide the average percent error of the values with respect to the oracle’s. We see that without DR the learner overestimates the value of being near yellow objects, which is actually the new teammate’s preferred object. Overall, without DR we see a higher %-error than GPAT.

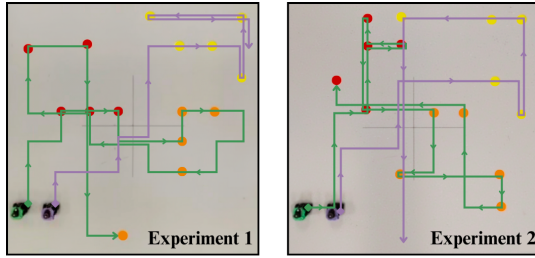


Fig. 8: Multi-robot demonstration of the foraging environment. We overlay the trajectories taken by the robots for two experiments from Table I (green=learner, purple=teammate).

VII. CONCLUSION

In this work, we propose GPAT, a novel approach that leverages GPI and difference rewards for ZSC in AHT. In contrast to previous methods that use a single policy from a pretrained library with online inference or train a single policy robust to diverse teammates, we dynamically leverage the entire library of pretrained policies at every time step by applying a GPI policy. We address the impact of the distribution shift induced by new teammates on the GPI policy through difference rewards. We empirically demonstrate that this more exhaustive use of prior knowledge improves performance in three simulated environments relative to baselines. We also include a multi-robot demonstration of our method. Future directions include theoretically analyzing our approach and expanding to the online adaptation setting by developing sample-efficient ways to update pretrained policies.

ACKNOWLEDGMENTS

This work was supported in part by ONR N00014-20-1-2249, a NASA grant awarded to the Illinois/NASA Space Grant Consortium, and a GAANN grant from the U.S. Department of Education.

REFERENCES

- [1] P. Stone, G. Kaminka, S. Kraus, and J. Rosenschein, “Ad hoc autonomous agent teams: Collaboration without pre-coordination,” *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 24, pp. 1504–1509, Jul. 2010.
- [2] R. Mirsky, I. Carlucho, A. Rahman, E. Fosong, W. Macke, M. Sridharan, P. Stone, and S. V. Albrecht, “A survey of ad hoc teamwork research,” in *Multi-Agent Systems* (D. Baumeister and J. Rothe, eds.), (Cham), pp. 275–293, Springer International Publishing, 2022.
- [3] S. Barrett, A. Rosenfeld, S. Kraus, and P. Stone, “Making friends on the fly: Cooperating with new teammates,” *Artificial Intelligence*, vol. 242, pp. 132–171, 2017.
- [4] H. Li, T. Ni, S. Agrawal, F. Jia, S. Raja, Y. Gui, D. Hughes, M. Lewis, and K. Sycara, “Individualized mutual adaptation in human-agent teams,” *IEEE Transactions on Human-Machine Systems*, vol. 51, no. 6, pp. 706–714, 2021.
- [5] M. Zhao, R. Simmons, and H. Admoni, “Coordination With Humans Via Strategy Matching,” in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 9116–9123, Oct. 2022. ISSN: 2153-0866.
- [6] H. Hu, A. Lerer, A. Peysakhovich, and J. Foerster, ““Other-Play” for Zero-Shot Coordination,” in *Proceedings of the 37th International Conference on Machine Learning*, pp. 4399–4410, PMLR, Nov. 2020. ISSN: 2640-3498.
- [7] R. Zhao, J. Song, Y. Yuan, H. Haifeng, Y. Gao, Y. Wu, Z. Sun, and Y. Wei, “Maximum Entropy Population-Based Training for Zero-Shot Human-AI Coordination,” June 2022. arXiv:2112.11701 [cs].
- [8] C. Yu, J. Gao, W. Liu, B. Xu, H. Tang, J. Yang, Y. Wang, and Y. Wu, “Learning zero-shot cooperation with humans, assuming humans are biased,” in *The Eleventh International Conference on Learning Representations*, 2023.
- [9] X. Wang*, S. Zhang*, W. Zhang, W. Dong, J. Chen, Y. Wen, and W. Zhang, “Zsc-eval: An evaluation toolkit and benchmark for multi-agent zero-shot coordination,” in *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024.
- [10] A. Barreto, D. Borsa, J. Quan, T. Schaul, D. Silver, M. Hessel, D. Mankowitz, A. Zidek, and R. Munos, “Transfer in deep reinforcement learning using successor features and generalised policy improvement,” in *Proceedings of the 35th International Conference on Machine Learning* (J. Dy and A. Krause, eds.), vol. 80 of *Proceedings of Machine Learning Research*, pp. 501–510, PMLR, 10–15 Jul 2018.
- [11] S. Proper and K. Tumer, “Modeling difference rewards for multiagent learning,” in *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, p. 2, 2012.
- [12] T. Ni, H. Li, S. Agrawal, S. Raja, F. Jia, Y. Gui, D. Hughes, M. Lewis, and K. Sycara, “Adaptive agent architecture for real-time human-agent teaming,” in *Proceedings of AAAI ’21 W20: Workshop on Plan, Activity, and Intent Recognition (PAIR ’21)*, February 2021.

- [13] J. Zand, J. Parker-Holder, and S. J. Roberts, "On-the-fly strategy adaptation for ad-hoc agent coordination," in *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '22, (Richland, SC), p. 1771–1773, International Foundation for Autonomous Agents and Multiagent Systems, 2022.
- [14] D. Strouse, K. McKee, M. Botvinick, E. Hughes, and R. Everett, "Collaborating with humans without human data," *Advances in Neural Information Processing Systems*, vol. 34, pp. 14502–14515, 2021.
- [15] K. Lucas and R. E. Allen, "Any-play: An intrinsic augmentation for zero-shot coordination," in *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*, pp. 853–861, 2022.
- [16] P. Gu, M. Zhao, J. Hao, and B. An, "Online ad hoc teamwork under partial observability," in *International Conference on Learning Representations*, 2022.
- [17] J. G. Ribeiro, C. Martinho, A. Sardinha, and F. S. Melo, "Assisting unknown teammates in unknown tasks: Ad hoc teamwork under partial observability," *arXiv preprint arXiv:2201.03538*, 2022.
- [18] R. Mirsky, W. Macke, A. Wang, H. Yedidsion, and P. Stone, "A Penny for Your Thoughts: The Value of Communication in Ad Hoc Teamwork," in *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, (Yokohama, Japan), pp. 254–260, International Joint Conferences on Artificial Intelligence Organization, July 2020.
- [19] H. Nekoei, X. Zhao, J. Rajendran, M. Liu, and S. Chandar, "Towards Few-shot Coordination: Revisiting Ad-hoc Teamplay Challenge In the Game of Hanabi," in *Proceedings of The 2nd Conference on Lifelong Learning Agents*, pp. 861–877, PMLR.
- [20] A. Rahman, N. Hopner, F. Christianos, and S. V. Albrecht, "Towards Open Ad Hoc Teamwork Using Graph-based Policy Learning," in *Proceedings of the 38th International Conference on Machine Learning*, p. 11, 2021.
- [21] C. Wang, M. A. Rahman, I. Durugkar, E. Liebman, and P. Stone, "N-agent ad hoc teamwork," *Advances in Neural Information Processing Systems*, vol. 37, pp. 111832–111862, 2025.
- [22] R. Paleja, M. Ghuy, N. R. Arachchige, R. Jensen, and M. Gombolay, "The Utility of Explainable AI in Ad Hoc Human-Machine Teaming," in *35th Conference on Neural Information Processing Systems (NeurIPS 2021)*, p. 14, 2021.
- [23] G. Tesauro, "TD-Gammon, a Self-Teaching Backgammon Program, Achieves Master-Level Play," vol. 6, no. 2, pp. 215–219, 1994.
- [24] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, L. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.
- [25] M. Carroll, R. Shah, M. K. Ho, T. L. Griffiths, S. A. Seshia, P. Abbeel, and A. Dragan, "On the utility of learning about humans for Human-AI coordination," in *33rd Conference on Neural Information Processing Systems (NeurIPS 2019)*, vol. 32, 2019.
- [26] C. Boutilier, "Planning, learning and coordination in multiagent decision processes," in *Proceedings of the Theoretical Aspects of Reasoning about Knowledge, TARK-96*, 1996.
- [27] A. Barreto, W. Dabney, R. Munos, J. J. Hunt, T. Schaul, H. P. van Hasselt, and D. Silver, "Successor features for transfer in reinforcement learning," in *Advances in Neural Information Processing Systems* (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds.), vol. 30, Curran Associates, Inc., 2017.
- [28] J. Castellini, S. Devlin, F. A. Oliehoeck, and R. Savani, "Difference rewards policy gradients," *Neural Comput & Applic*, Nov. 2022.
- [29] A. Barreto, S. Hou, D. Borsa, D. Silver, and D. Precup, "Fast reinforcement learning with generalized policy updates," *Proceedings of the National Academy of Sciences*, vol. 117, pp. 30079–30087, Dec. 2020. Publisher: Proceedings of the National Academy of Sciences.
- [30] S. Hansen, W. Dabney, A. Barreto, D. Warde-Farley, T. V. de Wiele, and V. Mnih, "Fast task inference with variational intrinsic successor features," in *International Conference on Learning Representations*, 2020.
- [31] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *CoRR*, vol. abs/1707.06347, 2017.
- [32] D. H. Wolpert and K. Tumer, "Optimal payoff functions for members of collectives," *Advances in Complex Systems*, vol. 4, no. 02n03, pp. 265–279, 2001.
- [33] M. A. Riedmiller, "Neural fitted q iteration - first experiences with a data efficient neural reinforcement learning method," in *European Conference on Machine Learning*, 2005.
- [34] G. Neumann and J. Peters, "Fitted q-iteration by advantage weighted regression," in *Proceedings of the 22nd International Conference on Neural Information Processing Systems*, NIPS'08, (Red Hook, NY, USA), p. 1177–1184, Curran Associates Inc., 2008.
- [35] D. Xing, P. Gu, Q. Zheng, X. Wang, S. Liu, L. Zheng, B. An, and G. Pan, "Controlling Type Confounding in Ad Hoc Teamwork with Instance-wise Teammate Feedback Rectification," in *Proceedings of the 40th International Conference on Machine Learning*, 2023.
- [36] R. E. Wang, S. A. Wu, J. A. Evans, J. B. Tenenbaum, D. C. Parkes, and M. Kleiman-Weiner, "Too many cooks: Bayesian inference for coordinating multi-agent collaboration," July 2020. arXiv:2003.11778 [cs].
- [37] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dornmann, "Stable-baselines3: Reliable reinforcement learning implementations," *Journal of Machine Learning Research*, vol. 22, no. 268, pp. 1–8, 2021.
- [38] R. Agarwal, M. Schwarzer, P. S. Castro, A. Courville, and M. G. Bellemare, "Deep reinforcement learning at the edge of the statistical precipice," *Advances in Neural Information Processing Systems*, 2021.