N-Agent Ad Hoc Teamwork

Caroline Wang
Department of Computer Science
The University of Texas at Austin
caroline.l.wang@utexas.edu

Arrasy Rahman
Department of Computer Science
The University of Texas at Austin
arrasy@cs.utexas.edu

Ishan Durugkar
Sony AI
ishan.durugkar@sony.com

Elad Liebman[*]
Amazon
liebelad@amazon.com

Peter Stone
Department of Computer Science
The University of Texas at Austin and Sony AI
pstone@cs.utexas.edu

Abstract

Current approaches to learning cooperative multi-agent behaviors assume relatively restrictive settings. In standard fully cooperative multi-agent reinforcement learning, the learning algorithm controls *all* agents in the scenario, while in ad hoc teamwork, the learning algorithm usually assumes control over only a *single* agent in the scenario. However, many cooperative settings in the real world are much less restrictive. For example, in an autonomous driving scenario, a company might train its cars with the same learning algorithm, yet once on the road, these cars must cooperate with cars from another company. Towards expanding the class of scenarios that cooperative learning methods may optimally address, we introduce N*-agent ad hoc teamwork* (NAHT), where a set of autonomous agents must interact and cooperate with dynamically varying numbers and types of teammates. This paper formalizes the problem, and proposes the *Policy Optimization with Agent Modelling* (POAM) algorithm. POAM is a policy gradient, multi-agent reinforcement learning approach to the NAHT problem that enables adaptation to diverse teammate behaviors by learning representations of teammate behaviors. Empirical evaluation on tasks from the multi-agent particle environment and StarCraft II shows that POAM improves cooperative task returns compared to baseline approaches, and enables out-of-distribution generalization to unseen teammates.

1 Introduction

Advances in multi-agent reinforcement learning (MARL) [3] have enabled agents to learn solutions to various problems in zero-sum games, social dilemmas, adversarial team games, and cooperative tasks [37, 8, 19, 21, 33]. Within MARL, cooperative multi-agent reinforcement learning (CMARL) is a paradigm for learning agent teams that solve a common task via interaction with each other and the environment [22, 28, 46]. Recent CMARL methods have been able to learn impressive examples of cooperative behavior from scratch in controlled settings, where all agents are controlled by the same learning algorithm [33, 38, 4]. A related paradigm for learning cooperative behavior is ad hoc

[*]Work was done while at SparkCognition.

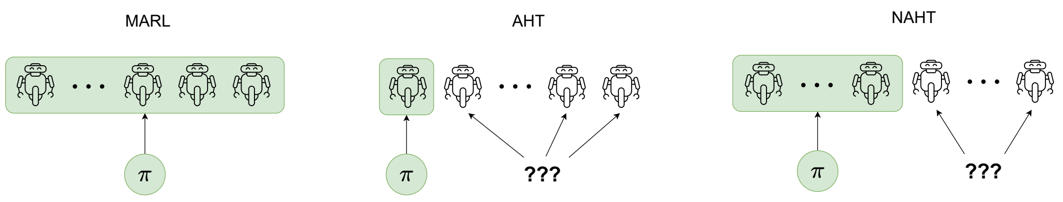

Figure 1: Left: CMARL algorithms assume full control over all M agents in a cooperative scenario. Center: AHT algorithms assume that only a single agent is controlled by the learning algorithm, while the other $M - 1$ agents are uncontrolled and can have a diverse, unknown set of behaviors. Right: NAHT, the paradigm proposed by this paper, assumes that a potentially varying N agents are controlled by the learning algorithm, while the remaining $M - N$ agents are uncontrolled.

teamwork (AHT). In contrast to CMARL, the objective of AHT is to create a single agent policy that can collaborate with previously unknown teammates to solve a common task [27, 39].

While a large and impressive body of work on CMARL and AHT exists, the current literature has largely examined scenarios in which either complete control over all agents is assumed, or only a single agent is adapted for cooperation [33, 38, 21, 4, 15]. Even learning methods for handling cooperative tasks in open multiagent systems [46], which encompass one of the most challenging settings where agents may enter or leave the system anytime, either operate assuming full control over all agents [17, 23] or only a single adaptive agent [31, 18, 32].

However, real-world collaborative scenarios—e.g. search-and-rescue, or robot fleets for warehouses—might demand agent *subteams* that are able to collaborate with unfamiliar teammates that follow different coordination conventions. Towards producing agent teams that are more flexible and applicable to realistic cooperative scenarios, this paper formalizes the problem setting of **N-agent ad hoc teamwork** (NAHT), in which a set of autonomous agents must interact with an uncontrolled set of teammates to perform a cooperative task. When there is only a single ad hoc agent, NAHT is equivalent to AHT. On the other hand, when all ad hoc agents are jointly trained by the same algorithm and there are no uncontrolled teammates, NAHT is equivalent to CMARL. Thus, the proposed problem setting generalizes both CMARL and AHT.

Drawing from ideas in both CMARL and AHT, we introduce Policy Optimization with Agent Modelling (POAM). POAM is a policy-gradient based approach for learning cooperative multi-agent team behaviors, in the presence of varying numbers and types of teammate behaviors. It consists of (1) an agent modeling network that generates a vector characterizing teammate behaviors, and (2) an independent actor-critic architecture, which conditions on the learned teammate vectors to enable adaptation to a variety of potential teammate behaviors. Empirical evaluation on multi-agent particle environment (MPE) and StarCraft II tasks shows that POAM learns to coordinate with a changing number of teammates of various types, with higher competency than CMARL, AHT, and naive NAHT baseline approaches. An evaluation with out-of-distribution teammates also reveals that POAM's agent modeling module improves generalization to out-of-distribution teammates, compared to baselines without agent modeling.

2 Background and Notation

The NAHT problem is formulated within the framework of **Decentralized Partially Observable Markov Decision Processes**, or Dec-POMDPs [7]. A Dec-POMDP consists of M agents, a state space $\mathcal{S}$, action space $\mathcal{A}$, per-agent observation spaces O_i, transition function $\mathcal{T} : \mathcal{S} \times \mathcal{A} \mapsto \Delta(\mathcal{S})$ [2], common reward function $r : \mathcal{S} \times \mathcal{A} \mapsto \Delta(\mathbb{R})$ (thus defining a cooperative task), discount factor $\gamma \in [0, 1]$ and horizon $T \in \mathbb{Z}$, which represents the maximum length of an interaction, or episode. Each agent observes the environment via its observation function, $\mathcal{O}_i : \mathcal{S} \times \mathcal{A} \mapsto \Delta(O_i)$. The state space is factored such that $\mathcal{S} = \mathcal{S}_1 \times \cdots \times \mathcal{S}_M$, where $\mathcal{S}_i$ for $i \in \{1 \cdots M\}$ corresponds to the state space for agent i. The action space is defined analogously. Denoting H_i as its space of localized observation and action histories, agent i acts according to a policy, $\pi_i : H_i \mapsto \Delta(\mathcal{A}_i)$. The notation $-i$ represents all agents other than agent i, and is applied throughout the paper to the mathematical

[2] $\Delta(S)$ denotes the space of probability distributions over set S.

objects introduced above. For example, the notation $\mathcal{O}_{-i}$ refers to the cross product of the observation space of all agents other than i. In the following, we overload r to refer to both the reward function, and the task defined by that reward function, whereas r_t denotes the reward at time step t.

3 NAHT Problem Formulation

Drawing from the goals of MARL and AHT [39], the goal of N-agent ad hoc teamwork is **to create a *set* of autonomous agents that are able to efficiently collaborate with both known and unknown teammates to maximize return on a task**. The goal is formalized below.

Let C denote a set of ad hoc agents. If the policies of the agents in C are generated by an algorithm, we say that the algorithm controls agents in C. Since our intention is to develop algorithms for generating the policies of agents in C, we refer to agents in C as *controlled*. Let U denote a set of *uncontrolled* agents, which we define as all agents in the environment not included in C.[3] Following Stone et al., we assume that agents in U are not adversarially minimizing the objective of agents in C.

We model an open system of interaction, in which a random selection of M agents from sets C and U must coordinate to perform task r. For illustration, consider a warehouse staffed by robots developed by companies A and B, where there is a box-lifting task that requires three robots to accomplish. If Company A's robots are controlled agents (corresponding to C), then some robots from Company A could collaborate with robots from Company B (corresponding to U) to accomplish the task, rather than requiring that all three robots come exclusively from A or B. Motivated thus, we introduce a *team sampling procedure* $X(U, C)$. At the beginning of each episode, X samples a team of M agents by first sampling $N < M$, then sampling N agents from the set C and $M - N$ agents from U. We restrict consideration to teams containing at least one controlled agent, i.e $N \geq 1$. We consider X a problem parameter that is not under the control of any algorithm for generating ad hoc teammates, analogous to the transition function of the underlying Dec-POMDP. A more explicit definition of X is provided in Appendix A.1.

Without loss of generality, let $C(\theta) = \{\pi_i^\theta(.|s)\}_{i=1}^N$ denote a *set* of M controlled agent policies parameterized by θ, such that a learning algorithm might optimize for θ. Let the $\boldsymbol{\pi}^{(M)}$ indicate a *team* of M agents and $\boldsymbol{\pi}^{(M)} \sim X(U, C)$ indicate sampling such a team from U and C via the team sampling procedure. The *objective* of the NAHT problem is to find parameters θ, such that $C(\theta)$ maximizes the expected return in the presence of teammates from U:

$$\max_\theta \left(\mathbb{E}_{\boldsymbol{\pi}^{(M)} \sim X(U, C(\theta))} \left[\sum_{t=0}^{T} \gamma^t r_t \right] \right). \tag{1}$$

Challenges of the NAHT problem include: (1) coordinating with potentially unknown teammates (*generalization*), and (2) coping with a varying number of uncontrolled teammates (*openness*).

4 The Need for Dedicated NAHT Algorithms

Having introduced the NAHT problem, a natural question to consider is whether AHT solutions may optimally address NAHT problems. If so, then there would be little need to consider the NAHT problem setting. For instance, a simple yet reasonable approach consists of directly using an AHT policy to control as many agents as required in an NAHT scenario. This section illustrate the limitations of the aforementioned approach by giving a concrete example of a matrix game where (1) an AHT policy that is learned in the AHT ($N = 1$) scenario is unlikely to do well in an NAHT scenario where $N = 2$, and (2) even an *optimal* AHT policy is suboptimal in the $N = 2$ setting.

Define the following simple game for M agents: at each turn, each agent a_i picks one bit $b_i \in \{0, 1\}$; at the end of each turn, all the bits are summed $s = \sum_i b_i$. The team wins if the sum of the chosen bits is exactly 1. We denote the probability of winning by $P(s = 1)$. Suppose the uncontrolled agents

[3] The original AHT problem statement used the terms "known" and "unknown" agents. However, it is common for modern AHT learning methods to assume some knowledge about the "unknown" teammates during training [27]. Thus, we instead employ the terms controlled and uncontrolled.

follow a policy that independently selects 1 with probability $p = \frac{1}{M}$.[4] In the following, we consider the three agent case, $M = 3$, for simplicity.

In the AHT problem setting, a learning algorithm assumes control of only a single agent. Let p_{aht} denote the probability with which the AHT agent selects 1. Given the aforementioned team of uncontrolled agents, we show that *any* value of p_{aht} results in the same probability of winning, which occurs because the probability of winning, $P(s = 1) = \frac{4}{9}$, is independent of p_{aht} (Lemma A.2).

Next, consider an NAHT scenario where a learning algorithm must define the actions of two out of three agents. Suppose that the same AHT policy is used to control both agents: both agents select 1 with probability p_{aht}. Above, we demonstrated that an AHT algorithm trained in the $N = 1$ scenario could result in learning any p_{aht}. However, in the $N = 2$ setting, we show that the optimal AHT policy $p_{\text{aht}} = \frac{1}{3}$ and the winning probability for this policy is $P(s = 1) = \frac{4}{9}$ (Lemma A.3).

Finally, we show there exists an NAHT policy that controls both agents and obtains a higher winning probability. Consider the policy where one controlled agent always plays 0, while the other plays 1 with probability p_{naht}. Lemma A.4 shows that the optimal $p_{\text{naht}} = 1$, and the probability of winning $P(s = 1) = \frac{2}{3} > \frac{4}{9}$. Thus, we have exhibited an NAHT scenario where an AHT policy that is optimal when $N = 1$, performs worse than a simple NAHT joint policy in the $N = 2$ setting. Empirical validation of the prior results are provided in Appendix A.5.1.

5 Policy Optimization with Agent Modeling (POAM)

This section describes the proposed Policy Optimization with Agent Modeling (POAM) method, which trains a collection of NAHT agents that can adaptively deal with different collections of unknown teammates. POAM relies on an *agent modeling network* to initially build an embedding vector characterizing teammates encountered during an interaction. Adaptive agent policies that can maximize the controlled agents' returns are then learned by training a *policy* conditioned on the environment observation and team embedding vector. To enable controlling a varying number of agents while learning in a sample-efficient manner, POAM adopts the independent learning framework with full parameter sharing. The training processes for agent modeling and policy networks are described in Sections 5.1 and 5.2 respectively, while an illustration of how POAM trains NAHT agents is provided in Figure 2.

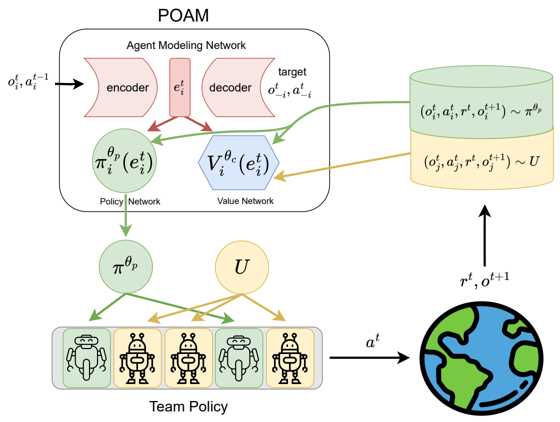

Figure 2: POAM trains a single policy network π^{θ_p}, which characterizes the behavior of all controlled agents (green), while uncontrolled agents (yellow) are drawn from U. Data from both controlled and uncontrolled agents is used to train the value network, $V_i^{\theta_c}$ while the policy is trained on data from the controlled agents only. The policy and value function are both conditioned on a learned team embedding vector, e_i^t.

5.1 Agent Modeling Network

Designing adaptive policies that enable NAHT agents to achieve optimal returns against any team of uncontrolled agents drawn from some set U, requires information on the encountered team's unknown behavior. However, in the absence of prior knowledge about uncontrolled teammates' policies, local observations from a single timestep may not contain sufficient information regarding the encountered team. To circumvent this lack of information, POAM's agent modeling network plays a crucial role in providing *team embedding vectors* that characterize the observed behavior of teammates in the encountered team.

We identify two main criteria for desirable team embedding vectors. First, team embedding vectors should identify information regarding the unknown state and behavior of other agents in the environment (both controlled and uncontrolled). Second, team embedding vectors should ideally

[4]Lemma A.1 shows that $p = \frac{1}{M}$ is the optimal value of p for a team composed of such agents.

be computable from the sequence of local observations and actions of the team. Fulfilling both requirements provides an agent with useful information for decision-making in NAHT problems, even under partial observability.

For each controlled agent, POAM produces informative team embedding vectors by training a model with an encoder-decoder architecture, illustrated by red components in Figure 2. For ease of presentation, the encoder-decoder models for controlled agent i will be referred to without the index i. The encoder, $f^{\text{enc}}_{\theta^e} : H_i \mapsto \mathbb{R}^n$, is parameterized by θ^e and processes the modeling agent's history of local observations and actions up to timestep t, $h^t_i = \{o^k_i, a^{k-1}_i\}^t_{k=1}$, to compute a team embedding vector of dimension n, $e^t_i \in \mathbb{R}^n$ that characterizes the modeled agents. This reliance on local observations helps ensure that the agent modeling network can operate without having access to the environment state that is unavailable under partial observability. The team embedding vector is decoded by two decoder networks: the observation decoder, $f^{\text{dec}}_{\theta^o} : \mathbb{R}^n \mapsto O_{-i}$ and the action decoder, $f^{\text{dec}}_{\theta^a} : \mathbb{R}^n \mapsto \Delta(A_{-i})$. The decoder networks are respectively trained to predict the observations and actions of all other agents on the team at timestep t, (o^t_{-i}, a^t_{-i}), to encourage e^t_i to contain relevant information for the current NAHT agent's decision-making process. While the observation decoder directly predicts the observed $-i$ observations, the action decoder predicts the parameters of a probability distribution over the $-i$ agents' actions, $p(a^t_{-i}; f^{\text{dec}}_{\theta^a}(f^{\text{enc}}_{\theta^e}(h^t_i)))$, where an appropriate distribution for p should be selected by the system designer.

Concretely, agent i's encoder-decoder model is trained to minimize a maximum likelihood loss over all teammates' observations and actions, given its own local observations and actions. As the experimental setting in this paper considers continuous observations and discrete actions, the observation loss is a mean squared error loss, while the action loss is the negative log likelihood of the $-i$ agents' actions, under the Categorical distribution.

$$L_{\theta^e,\theta^o,\theta^a}(h^t_i, o^t_{-i}, a^t_{-i}) = \Big(||f^{\text{dec}}_{\theta^o}(f^{\text{enc}}_{\theta^e}(h^t_i)) - o^t_{-i}||^2 - \log(p(a^t_{-i}; f^{\text{dec}}_{\theta^a}(f^{\text{enc}}_{\theta^e}(h^t_i))))\Big). \tag{2}$$

5.2 Policy and Value Networks

POAM relies on an actor-critic approach to train agent policies, where the policy and critic are both conditioned on the teammate embedding described in Section 5.1.

The policy network of agent i, $\pi^{\theta^p}_i : H_i \times \mathbb{R}^n \mapsto \Delta(A_i)$, is parameterized by θ^p, and uses the NAHT agent's local observation, o^t_i, and the team embedding from the encoder network, e^t_i, to compute a policy followed by the NAHT agents. Conditioning the policy network on e^t_i allows an NAHT agent to change its behaviour based on the inferred characteristics of encountered agents. When training the policy network, we also rely on a value (or critic) network, $V^{\theta^c}_i : H_i \times \mathbb{R}^n \mapsto \mathbb{R}$, parameterized by θ^c, which measures the expected returns given h^t_i, and e^t_i. The value network serves as a baseline to reduce the variance of the gradient updates, while conditioning on the learned teammate embeddings for similar reasons to the policy.

POAM then trains the policy and value networks using an approach based on the Independent PPO algorithm [45] (IPPO). IPPO is selected as the base MARL algorithm for two reasons. First, using an independent MARL method circumvents the need to deal with the changing number of agents resulting from environment openness. Second, IPPO has been demonstrated to be effective on various MARL tasks. To improve learning efficiency and enable information sharing between agents, full parameter sharing is employed for all neural networks. POAM trains the value network to produce accurate state value estimates by minimizing the following loss function:

$$L_{\theta^c}(h^t_i) = \frac{1}{2}\Big(V^{\theta^c}_i(h^t_i, f^{\text{enc}}_{\theta^e}(h^t_i)) - \hat{V}_i^{\,t}\Big)^2, \tag{3}$$

where $\hat{V}_i^{\,t}$ is the TD(λ) return. The policy network is analogously trained to minimize the PPO loss function [35], but where the policy additionally conditions on the team embeddings.

Leveraging data from uncontrolled agents In the NAHT setting, we assume access to the *joint* observations and actions generated by the current team deployed in the environment *at training time only*, where the team consists of a mix of controlled and uncontrolled agents. This assumption provides an opportunity to learn useful cooperative behaviors more quickly, by bootstrapping based on transitions from the initially more competent, uncontrolled teammate policies, as also observed by Rahman et al. [31].

POAM leverages data from both controlled and uncontrolled agents to train the value network—in effect, treating the uncontrolled agents as exploration policies. Note that this aspect of POAM is a significant departure from PPO, which is a fully on-policy algorithm. Since the policy update is highly sensitive to off-policy data, only data from the controlled agents is used to train the policy network.

6 Experiments and Results

This section presents an empirical evaluation of POAM and baseline approaches across different NAHT problems. We investigate three questions and foreshadow the conclusions:

Q1: Does POAM learn to cope with uncontrolled teammates with higher sample efficiency and asymptotic return than baselines? (Usually)

Q2: Does POAM improve generalization to previously unseen and out-of-distribution teammates, compared to baselines? (Yes)

Q3: Can we verify that the two key ideas of POAM—agent modelling and use of data from uncontrolled agents—work as desired and contribute positively towards POAM's performance? (Yes)

Full implementation details, including hyperparameter values and additional empirical results, appear in the Appendix. Our code is available at `https://github.com/carolinewang01/naht`.

6.1 Experimental Design

In the following, we summarize the experimental design, including the particular NAHT problem instance, training procedure, experimental domain, and baselines. Details on evaluation metrics are provided in Appendix A.3.3.

A Practical Instantiation of NAHT Similarly to AHT, the NAHT problem can be parameterized by the amount of knowledge that controlled agents have about uncontrolled agents, and whether uncontrolled agents can adapt to the behavior of controlled agents [5]. Furthermore, as a direct result of the fact that an NAHT algorithm may control more than a single agent, the NAHT problem may also be parameterized by whether the controlled agents are homogeneous, whether they can communicate, and what the team sampling procedure is. While the fully general problem setting allows for heterogeneous, communicating, controlled agents that have no knowledge of the uncontrolled agents, as a first step, this paper focuses on a special case of the NAHT problem, where agents are homogeneous, non-communicating, may learn about uncontrolled agents via interaction, and where the team sampling procedure consists of a uniform random sampling scheme from U and C (see Appendix A.3.1 for details). This case is designed primarily to assess whether controlled *subteams* may outperform independent controlled agents, when cooperating with multiple types of uncontrolled agents. We leave consideration of broader NAHT scenarios for future work.

Generating Uncontrolled Teammates To generate a set of uncontrolled teammates U, the following MARL algorithms are used to train agent teams: VDN [41], QMIX [33], IQL [42], IPPO, and MAPPO [45]. We verify that the generated team behaviors are diverse by checking that (1) teams trained by the same algorithm learn non-compatible coordination conventions, and (2) teams trained by different algorithms also learn non-compatible coordination conventions (Appendix A.5.2). The emergence of diverse teammate behaviors from training agents using different MARL algorithms under different seeds aligns with the findings from Strouse et al. [40].

Let U_{train} denote the set of uncontrolled teammates used to train all (N)AHT methods. U_{train} consists of five teams, where each individual team is trained via VDN, QMIX, IQL, IPPO and MAPPO, respectively. U_{test} consists of a set of holdout teams trained via the same MARL algorithms, but that have not been seen during training. The experimental results reported in Sections 6.2 and 6.4 are computed with respect to U_{train} only, while the experimental results in Section 6.3 use U_{test}.

Experimental Domains Experiments are conducted on a predator-prey `mpe-pp` task implemented within the multi-agent particle environment [24], and the `5v6, 8v9, 10v11, 3s5z` tasks from the

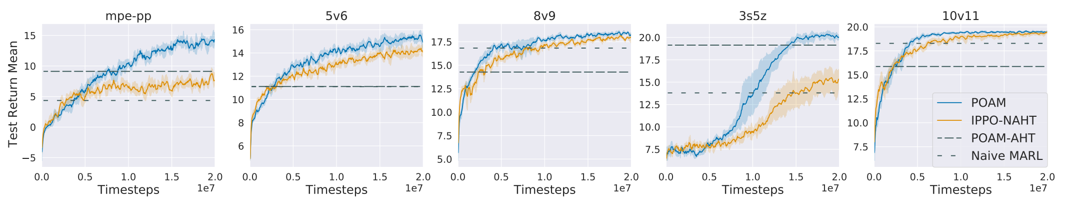

Figure 3: POAM consistently improves over the baselines of IPPO-NAHT, POAM-AHT, and the best naive MARL baseline in all tasks, in either sample efficiency or asymptotic return.

StarCraft Multi-Agent Challenge (SMAC) benchmark [34]. On the `mpe-pp` task, three predators must cooperatively pursue a pretrained prey agent. The team receives a reward of +1 per time step that two or more predators collide with the prey. On the SMAC tasks, a team of allied agents must defeat a team of enemy agents controlled by the game server. For each task, the first number in the task name indicates the number of allied agents, while the second indicates the number of enemy agents. The team is rewarded for defeating enemies, with a large bonus for defeating all enemies. See Appendix A.3.2 for full details.

Baselines As NAHT is a new problem proposed by this paper, there are no prior algorithms that are directly designed for the NAHT problem. Therefore, we construct three baselines to contextualize the performance of POAM. All methods employ full parameter sharing [30].

- *Naive MARL*: various well-known MARL algorithms are considered, including both independent and centralized training with decentralized execution algorithms [14]. The algorithms evaluated here include IQL [10], VDN [41], QMIX [33], IPPO, and MAPPO [45]. The MARL baselines are trained in self-play and then evaluated in the NAHT setting. In the following, only the performance of the *best* naive MARL baseline is reported.
- *Independent PPO in the NAHT setting* (IPPO-NAHT): IPPO is a policy gradient MARL algorithm that directly generalizes PPO [35] to the multi-agent setting. It was found to be surprisingly effective on a variety of MARL benchmarks [45]. In contrast to the naive MARL baselines, IPPO-NAHT is trained using the NAHT training scheme presented in Section 6.1. The variant considered here employs full parameter sharing, where the actor is trained on data only from controlled agents, but the critic is trained using data from both controlled and uncontrolled agents. The latter detail is a key algorithmic feature which POAM also employs, but is an extension from the most naive version of PPO (see Section 6.4). IPPO can be considered an ablation of POAM, where the agent modeling module is removed.
- *POAM in the AHT setting* (POAM-AHT): As considered in Section 4, a natural baseline approach to the NAHT problem is to use AHT algorithms that train only a single controlled agent, and copy these policies as many times as needed in the NAHT setting. To evaluate the intuition that AHT policies do not suffice for the NAHT problem setting, we consider an AHT version of POAM that is trained identically to POAM, but where the number of controlled agents is always one ($N = 1$) during training. Note that POAM-AHT is equivalent to the AHT algorithm introduced by Papoudakis et al. [29], LIAM.

6.2 Main Results

This section addresses Q1—that is, whether POAM learns to cope with uncontrolled teammates with greater sample efficiency or asymptotic returns, compared to baselines. Figure 3 shows the learning curves of POAM and IPPO-NAHT, and the test returns achieved by the best naive MARL baseline and POAM-AHT, on all tasks.[5] All learning curves consist of the mean test returns across **five** trials, while the shaded regions reflect the **95% confidence intervals**.

We find that for all tasks, POAM outperforms baselines in terms of asymptotic return for three out of five tasks (`mpe-pp, 5v6, 3s5z`). For all tasks, POAM's initial sample efficiency is similar to that of IPPO-NAHT for the first few million steps of training, after which POAM displays higher return. We attribute the initial similarity in sample efficiency to the initial cost incurred by learning team embedding vectors, which once learned, improves learning efficiency. IPPO-NAHT, which

[5]See the Appendix for tables providing the performances of all naive MARL methods, across all tasks.

can be viewed as an ablation of POAM with no agent modeling, is the next best performing method. Although IPPO-NAHT has generally poorer sample efficiency than POAM, the method converges to approximately the same returns on two out of five tasks (`8v9` and `10v11`). While the agent modeling module of POAM provides team embedding vectors to the policy learning process, the embeddings are themselves produced from each agent's own observation. Since no additional information is provided to POAM agents, it is unsurprising that IPPO-NAHT can converge to similar solutions as POAM, given enough training steps.

Finally, while the best naive MARL baseline and POAM-AHT learn good solutions on some tasks, neither method consistently performs well across all tasks. Overall, POAM discovers the most consistently performant policies compared to baseline methods, in a relatively sample-efficient manner. We conclude that (1) agent modeling improves learning efficiency, and (2) direct duplication of AHT-trained policies is less effective than methods that co-train agents for NAHT.

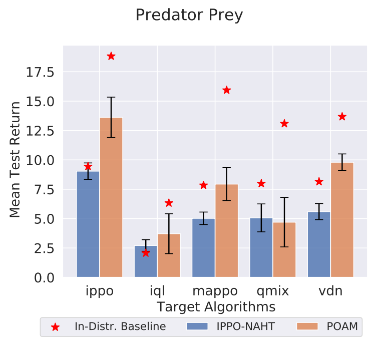

Figure 4: Test returns achieved by POAM and IPPO-NAHT, when paired with out-of-distribution teammates. POAM has improved generalization to OOD teammates, compared with IPPO-NAHT.

6.3 Out of Distribution Generalization

The AHT literature commonly assumes that uncontrolled teammates of interest may be interacted directly with during training [29, 6, 31]; experiments in the prior section were conducted under this assumption. However, in realistic scenarios, it may be challenging to enumerate all teammates likely to be encountered in the wild. This section examines the effectiveness of POAM under a true NAHT *evaluation* scenario, where POAM agents must coordinate with teammates that were not available at training time and are out-of-distribution (OOD) (Q2). Here, OOD teammates are created by running MARL algorithms with different random seeds than those used to generate train-time teammates.

Figures 4 and 10 show the mean and 95% confidence intervals of the test return achieved by POAM, compared to IPPO-NAHT, when the algorithm in question is paired with previously unseen seeds of IPPO, IQL, MAPPO, QMIX, and VDN. For each type of teammate, the performance of IPPO-NAHT/POAM against the exact teammates seen during training is shown as the in-distribution baseline. Both POAM and IPPO-NAHT consistently exhibit reduced performance against the OOD teammates, compared to their respective in-distribution performances. In three out of five tasks (`mpe, 5v6, 3s5z`), POAM has a significantly higher return than IPPO-NAHT, while the remaining two tasks exhibit a smaller improvement. In Appendix A.5.3, similar findings are presented with an alternative OOD teammate generation strategy, where the set of five MARL algorithms used to generate uncontrolled teammates (IPPO, IQL, MAPPO, QMIX, VDN) are divided into train/test sets.

6.4 A Closer Look at POAM

Two key aspects of POAM are the teammate modeling module, and the use of data from uncontrolled agents to train the critic (Q3). We study the impact of both aspects on POAM's sample efficiency, focusing on the `mpe-pp` and `5v6` tasks. Results on 5v6 may be found in Appendix A.4.

Teammate modeling performance In the NAHT training/evaluation setting, a new set of teammates is sampled at the beginning of each episode. Therefore, an important subtask for a competent NAHT agent is to rapidly model the distribution and type of teammates at the beginning of the episode, to enable the policy to exploit that knowledge as the episode progresses. This task is especially challenging in the presence of partial observability (a property of the SMAC tasks). To address the above challenges, POAM employs a recurrent encoder, which encodes the POAM agent's history of observations and actions to an embedding vector, and a (non-recurrent) decoder network, which predicts the egocentric observations and action distribution for all teammates.

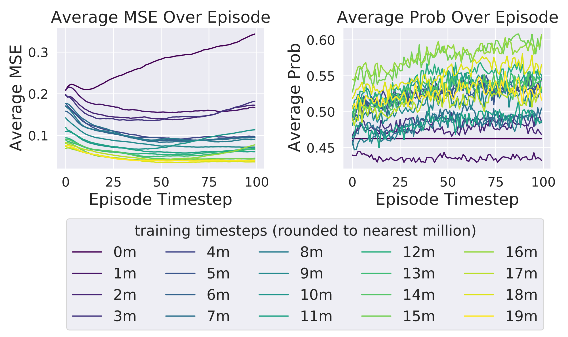

Figure 5: Evolution of a POAM agent's within-episode mean squared error (left) and within-episode probability of actions of modeled teammates (right), over the course of training on `mpe-pp`.

A natural question then, is whether the learned teammate embedding vectors actually improve over the course of an episode. Figures 5 and 8 depict the within-episode *mean squared error* (MSE) of the observation predicted by the decoder, and the within-episode *probability* of the action actually taken by the modeled teammates, according to the decoder's modeled action distribution. Each curve corresponds to a different training checkpoint of a single run.

For both tasks, we observe that the average MSE decreases over the course of training, while the average probability of the taken actions increases. For later training checkpoints, we observe that the average MSE actually decreases *within* an episode, while the average probability increases, reflecting the increased confidence of the agent modelling module as more data is observed about teammates. Thus, we conclude that POAM is able to cope with the challenges introduced by the sampled teammates and partial observability, to learn accurate teammate models.

Impact of data from non-controlled agents Recall that both POAM and IPPO-NAHT update the value network using data from both the controlled and uncontrolled agents. As Figures 6 and 9 show, this algorithmic feature results in a significant performance gain over training using on-policy data only, for both POAM and IPPO-NAHT.

7 Related Works

This section summarizes literature in areas most closely related to NAHT, namely, ad hoc teamwork, zero-shot coordination, evaluation of cooperative capabilities, agent modeling, and CMARL.

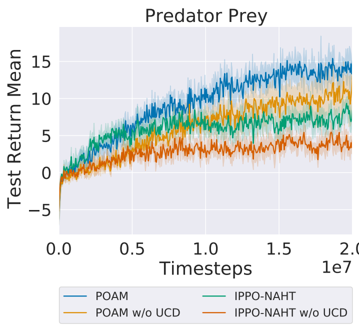

Figure 6: Learning curves of POAM and IPPO-NAHT, where the value network is trained w/w.o. uncontrolled agents' data (UCD).

Ad Hoc Teamwork & Zero-Shot Coordination. Prior works in ad hoc teamwork [39] and zero-shot coordination (ZSC) [16] explored methods to design adaptive agents that can optimally collaborate with unknown teammates. While they both highly resemble the NAHT problem, existing methods for AHT [31, 27] and ZSC [16, 25] have been limited to single-agent control scenarios. We argue that direct, naive applications of AHT and ZSC techniques to our problem of interest are ineffective—see the discussion in Section 4 and results in Section 6.

Recent research in AHT and ZSC utilizes neural networks to improve agent collaboration within various team configurations. These recent works mostly focus on two approaches. The first approach trains the agent to adapt to unknown teammates by characterizing teammates' behavior as fixed-length vectors using neural networks and learning a policy network conditioned on these vectors [31, 29, 47]. The second designs teammate policies that maximize the agent's performance when collaborating with diverse teammates [25, 31]. Our work builds on the first category, extending it to control multiple agents amid the existence of unknown teammates. While this also offers a potential path for robust NAHT agents, designing teammate policies for training will be kept as future work.

Evaluating Agents' Cooperative Capabilities. Beyond training agents to collaborate with teammates having unknown policies, researchers have developed environments and metrics to assess cooperative abilities. The Melting Pot suite [20, 1] mostly evaluates controlled agents' ability to maximize utilitarian welfare against unknown agents. However, this evaluation suite focuses on mixed-motive games where agents may have conflicting goals. This contrasts with our work's scope of fully cooperative settings, where all agents share the same reward function. MacAlpine et al. [26] also explored alternative metrics to measure agents' cooperative capabilities while disentangling the effects of their overall skills in drop-in RoboSoccer.

Agent Modeling. Agent modeling enables agents to characterize other agents based on their actions [2]. Such characterizations could attempt to directly infer modeled agents' actions, goals [13], or policies [29]. The modeled attributes have been used in cooperative, competitive, and general sum settings to inform update rules [11, 36] or to directly inform decision making [29]. POAM relies on agent modeling to provide important teammate information for decision-making when collaborating with unknown teammates.

Cooperative MARL (CMARL). CMARL explores algorithms for training agent teams on fully cooperative tasks. Some existing methods focus on credit assignment and decentralized control [12, 33]. Other works in CMARL also leverage parameter sharing and role assignment (e.g., [9, 44]) to decide an optimal division of labor between agents. However, these techniques assume control over all existing agents during training and evaluation, which limits their effectiveness in settings with unseen or uncontrolled teammates, as shown in prior work [43, 16, 31].

8 Conclusion

This paper proposes and formulates the problem of N-agent ad hoc teamwork (NAHT), a generalization of both AHT and MARL. It further proposes a multi-agent reinforcement learning algorithm to train NAHT agents called POAM, and develops a procedure to train and evaluate NAHT agents. POAM is a policy gradient approach that uses an encoder-decoder architecture to perform teammate modeling, and leverages data from uncontrolled agents for policy optimization. Empirical validation on MPE and StarCraft II tasks shows that POAM consistently improves over baseline methods that naively apply existing MARL and AHT approaches, in terms of sample efficiency, asymptotic return, and generalization to out-of-distribution teammates.

Limitations and Future Work. This paper addresses a special case of the NAHT problem, with homogeneous and non-communicating agents. POAM, which employs full parameter sharing, may not perform well in settings with heterogeneous agents, or in settings that require highly differentiated roles. POAM also does not leverage centralized state information or allow communication between controlled agents. Incorporating this information might enable learning improved NAHT policies. Further, POAM's actor update is purely on-policy, and therefore cannot leverage data generated by uncontrolled agents. Future work might consider employing off-policy methods to exploit the uncontrolled agent data. In addition to the directions suggested by POAM's limitations, algorithmic ideas from AHT, such as diversity-based teammate generation [25] and teammate-model-based planning methods [6], also suggest rich avenues for future work. Having introduced the NAHT problem in this work, we hope the community explores the many potential directions to design even better NAHT algorithms by considering advances in MARL, AHT, and agent modeling.

Acknowledgments and Disclosure of Funding

This work has taken place in the Learning Agents Research Group (LARG) at UT Austin. LARG research is supported in part by NSF (FAIN-2019844, NRT-2125858), ONR (N00014-18-2243), ARO (W911NF-23-2-0004, W911NF-17-2-0181), Lockheed Martin, and UT Austin's Good Systems grand challenge. Peter Stone serves as the Executive Director of Sony AI America and receives financial compensation for this work. The terms of this arrangement have been reviewed and approved by the University of Texas at Austin in accordance with its policy on objectivity in research.

References

[1] John P Agapiou, Alexander Sasha Vezhnevets, Edgar A Duéñez-Guzmán, Jayd Matyas, Yiran Mao, Peter Sunehag, Raphael Köster, Udari Madhushani, Kavya Kopparapu, Ramona Comanescu, et al. Melting pot 2.0. *arXiv preprint arXiv:2211.13746*, 2022.

[2] Stefano V Albrecht and Peter Stone. Autonomous agents modelling other agents: A comprehensive survey and open problems. *Artificial Intelligence*, 258:66–95, 2018.

[3] Stefano V. Albrecht, Filippos Christianos, and Lukas Schäfer. *Multi-Agent Reinforcement Learning: Foundations and Modern Approaches*. MIT Press, 2024. URL https://www.marl-book.com.

[4] Bowen Baker, Ingmar Kanitscheider, Todor Markov, Yi Wu, Glenn Powell, Bob McGrew, and Igor Mordatch. Emergent Tool Use From Multi-Agent Autocurricula. In *International Conference on Learning Representations*, September 2019. URL https://openreview.net/forum?id=SkxpxJBKwS.

[5] Samuel Barrett and Peter Stone. An analysis framework for ad hoc teamwork tasks. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pages 357–364, 2012.

[6] Samuel Barrett, Avi Rosenfeld, Sarit Kraus, and Peter Stone. Making friends on the fly: Cooperating with new teammates. *Artificial Intelligence*, 242:132–171, January 2017. ISSN 0004-3702. doi: 10.1016/j.artint.2016.10.005. URL https://www.sciencedirect.com/science/article/pii/S0004370216301266.

[7] Daniel S. Bernstein, Robert Givan, Neil Immerman, and Shlomo Zilberstein. The Complexity of Decentralized Control of Markov Decision Processes. *Mathematics of Operations Research*, 27(4):819–840, November 2002. ISSN 0364-765X. doi: 10.1287/moor.27.4.819.297. URL https://pubsonline.informs.org/doi/10.1287/moor.27.4.819.297.

[8] Noam Brown and Tuomas Sandholm. Superhuman ai for heads-up no-limit poker: Libratus beats top professionals. *Science*, 359(6374):418–424, 2018. doi: 10.1126/science.aao1733. URL https://www.science.org/doi/abs/10.1126/science.aao1733.

[9] Filippos Christianos, Georgios Papoudakis, Muhammad A Rahman, and Stefano V Albrecht. Scaling multi-agent reinforcement learning with selective parameter sharing. In *International Conference on Machine Learning*, pages 1989–1998. PMLR, 2021.

[10] Caroline Claus and Craig Boutilier. The dynamics of reinforcement learning in cooperative multiagent systems. In *Proceedings of the fifteenth national/tenth conference on Artificial intelligence/Innovative applications of artificial intelligence*, AAAI '98/IAAI '98, pages 746–752, USA, July 1998. American Association for Artificial Intelligence. ISBN 978-0-262-51098-1.

[11] Jakob Foerster, Richard Y. Chen, Maruan Al-Shedivat, Shimon Whiteson, Pieter Abbeel, and Igor Mordatch. Learning with opponent-learning awareness. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, page 122–130. International Foundation for Autonomous Agents and Multiagent Systems, 2018.

[12] Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[13] Josiah P Hanna, Arrasy Rahman, Elliot Fosong, Francisco Eiras, Mihai Dobre, John Redford, Subramanian Ramamoorthy, and Stefano V Albrecht. Interpretable goal recognition in the presence of occluded factors for autonomous vehicles. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7044–7051. IEEE, 2021.

[14] Pablo Hernandez-Leal, Bilal Kartal, and Matthew E. Taylor. A Survey and Critique of Multiagent Deep Reinforcement Learning. *Autonomous Agents and Multi-Agent Systems*, 33(6):750–797, November 2019. ISSN 1387-2532, 1573-7454. doi: 10.1007/s10458-019-09421-1. URL http://arxiv.org/abs/1810.05587. arXiv:1810.05587 [cs].

[15] Hengyuan Hu and Jakob N. Foerster. Simplified Action Decoder for Deep Multi-Agent Reinforcement Learning. September 2019. URL https://openreview.net/forum?id=B1xm3RVtwB.

[16] Hengyuan Hu, Adam Lerer, Alex Peysakhovich, and Jakob Foerster. "other-play" for zero-shot coordination. In *International Conference on Machine Learning*, pages 4399–4410. PMLR, 2020.

[17] Jiechuan Jiang, Chen Dun, Tiejun Huang, and Zongqing Lu. Graph convolutional reinforcement learning. In *International Conference on Learning Representations*, 2019.

[18] Anirudh Kakarlapudi, Gayathri Anil, Adam Eck, Prashant Doshi, and Leen-Kiat Soh. Decision-theoretic planning with communication in open multiagent systems. In *Uncertainty in Artificial Intelligence*, pages 938–948. PMLR, 2022.

[19] Raphael Koster, Miruna Pîslar, Andrea Tacchetti, Jan Balaguer, Leqi Liu, Romuald Elie, Oliver P Hauser, Karl Tuyls, Matt Botvinick, and Christopher Summerfield. Using deep reinforcement learning to promote sustainable human behaviour on a common pool resource problem, 2024. URL https://arxiv.org/ftp/arxiv/papers/2404/2404.15059.pdf.

[20] Joel Z Leibo, Edgar A Dueñez-Guzman, Alexander Vezhnevets, John P Agapiou, Peter Sunehag, Raphael Koster, Jayd Matyas, Charlie Beattie, Igor Mordatch, and Thore Graepel. Scalable evaluation of multi-agent reinforcement learning with melting pot. In *International conference on machine learning*, pages 6187–6199. PMLR, 2021.

[21] Fanqi Lin, Shiyu Huang, Tim Pearce, Wenze Chen, and Wei-Wei Tu. TiZero: Mastering Multi-Agent Football with Curriculum Learning and Self-Play. In *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '23, pages 67–76, Richland, SC, May 2023. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 978-1-4503-9432-1.

[22] Michael L Littman. Markov games as a framework for multi-agent reinforcement learning. In *Machine learning proceedings 1994*, pages 157–163. Elsevier, 1994.

[23] Bo Liu, Qiang Liu, Peter Stone, Animesh Garg, Yuke Zhu, and Anima Anandkumar. Coach-player multi-agent reinforcement learning for dynamic team composition. In *International Conference on Machine Learning*, pages 6860–6870. PMLR, 2021.

[24] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *Neural Information Processing Systems (NIPS)*, 2017.

[25] Andrei Lupu, Brandon Cui, Hengyuan Hu, and Jakob Foerster. Trajectory diversity for zero-shot coordination. In *International conference on machine learning*, pages 7204–7213. PMLR, 2021.

[26] Patrick MacAlpine and Peter Stone. Evaluating ad hoc teamwork performance in drop-in player challenges. In Gita Sukthankar and Juan A. Rodriguez-Aguilar, editors, *Autonomous Agents and Multiagent Systems, AAMAS 2017 Workshops, Best Papers*, pages 168–186. Springer International Publishing, 2017. URL http://nn.cs.utexas.edu/?LNAI17-MacAlpine.

[27] Reuth Mirsky, Ignacio Carlucho, Arrasy Rahman, Elliot Fosong, William Macke, Mohan Sridharan, Peter Stone, and Stefano V Albrecht. A survey of ad hoc teamwork research. In *European Conference on Multi-Agent Systems*, pages 275–293. Springer, 2022.

[28] Liviu Panait and Sean Luke. Cooperative multi-agent learning: The state of the art. *Autonomous Agents and Multi-Agent Systems*, 11:387–434, 2005. URL https://api.semanticscholar.org/CorpusID:19706.

[29] Georgios Papoudakis, Filippos Christianos, and Stefano V. Albrecht. Agent modelling under partial observability for deep reinforcement learning. In *Advances in Neural Information Processing Systems*, 2021.

[30] Georgios Papoudakis, Filippos Christianos, Lukas Schäfer, and Stefano V. Albrecht. Benchmarking Multi-Agent Deep Reinforcement Learning Algorithms in Cooperative Tasks. In *Advances in Neural Information Processing Systems*, volume 34, 2021.

[31] Arrasy Rahman, Niklas Höpner, Filippos Christianos, and Stefano V. Albrecht. Towards Open Ad Hoc Teamwork Using Graph-based Policy Learning. In *Proceedings of the 38 th International Conference on Machine Learning*, volume 139. PMLR, June 2021.

[32] Arrasy Rahman, Ignacio Carlucho, Niklas HÃ¶pner, and Stefano V. Albrecht. A general learning framework for open ad hoc teamwork using graph-based policy learning. *Journal of Machine Learning Research*, 24(298):1–74, 2023. URL http://jmlr.org/papers/v24/22-099.html.

[33] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*. PMLR, 2018.

[34] Mikayel Samvelyan, Tabish Rashid, Christian Schroeder de Witt, Gregory Farquhar, Nantas Nardelli, Tim G. J. Rudner, Chia-Man Hung, Philiph H. S. Torr, Jakob Foerster, and Shimon Whiteson. The StarCraft Multi-Agent Challenge. *CoRR*, abs/1902.04043, 2019.

[35] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *ArXiv*, abs/1707.06347, 2017. URL https://api.semanticscholar.org/CorpusID:28695052.

[36] Macheng Shen and Jonathan P. How. Robust opponent modeling via adversarial ensemble reinforcement learning in asymmetric imperfect-information games. In *Proceedings of the Thirty-First International Conference on Automated Planning and Scheduling*, 2021.

[37] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

[38] Kyunghwan Son, Daewoo Kim, Wan Ju Kang, David Earl Hostallero, and Yung Yi. QTRAN: Learning to Factorize with Transformation for Cooperative Multi-Agent Reinforcement Learning. In *Proceedings of the 36th International Conference on Machine Learning*, pages 5887–5896. PMLR, May 2019. URL https://proceedings.mlr.press/v97/son19a.html.

[39] Peter Stone, Gal Kaminka, Sarit Kraus, and Jeffrey Rosenschein. Ad Hoc Autonomous Agent Teams: Collaboration without Pre-Coordination. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 24, pages 1504–1509, July 2010. doi: 10.1609/aaai.v24i1.7529. URL https://ojs.aaai.org/index.php/AAAI/article/view/7529.

[40] DJ Strouse, Kevin McKee, Matt Botvinick, Edward Hughes, and Richard Everett. Collaborating with Humans without Human Data. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 14502–14515, 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/797134c3e42371bb4979a462eb2f042a-Paper.pdf.

[41] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z. Leibo, Karl Tuyls, and Thore Graepel. Value-decomposition networks for cooperative multi-agent learning based on team reward. In *Proceedings of the 17th International Conference on Autonomous Agents and Multi Agent Systems*, AAMAS '18, 2018.

[42] Ming Tan. Multi-agent reinforcement learning: Independent versus cooperative agents. In *International Conference on Machine Learning*, 1997. URL https://api.semanticscholar.org/CorpusID:267858156.

[43] Alexander Vezhnevets, Yuhuai Wu, Maria Eckstein, Rémi Leblond, and Joel Z Leibo. Options as responses: Grounding behavioural hierarchies in multi-agent reinforcement learning. In *International Conference on Machine Learning*, pages 9733–9742. PMLR, 2020.

[44] Mingyu Yang, Jian Zhao, Xunhan Hu, Wengang Zhou, Jiangcheng Zhu, and Houqiang Li. Ldsa: Learning dynamic subtask assignment in cooperative multi-agent reinforcement learning. *Advances in Neural Information Processing Systems*, 35:1698–1710, 2022.

[45] Chao Yu, Akash Velu, Eugene Vinitsky, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of mappo in cooperative multi-agent games. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, 2022.

[46] Lei Yuan, Ziqian Zhang, Lihe Li, Cong Guan, and Yang Yu. A survey of progress on cooperative multi-agent reinforcement learning in open environment. *arXiv preprint arXiv:2312.01058*, 2023.

[47] Luisa Zintgraf, Sam Devlin, Kamil Ciosek, Shimon Whiteson, and Katja Hofmann. Deep interactive bayesian reinforcement learning via meta-learning. *arXiv preprint arXiv:2101.03864*, 2021.

A Appendix

A.1 An Example of the Team Sampling Procedure

Let $g(k, \mu, s)$ represent a function for randomly selecting k elements from a generic set μ, conditioned on a state $s \in \mathcal{S}$. For instance, g might be the distribution, *Multinomial*$(k, |\mu|, \boldsymbol{\phi}(s))$, for selecting k elements with replacement from the set μ, parameterized by the state-dependent probability logits $\boldsymbol{\phi}(s) \in [0, 1]^{|\mu|}$. Allowing g to depend on an initial state enriches the representable open interactions. Returning to our robot warehouse example, a state-dependent g might enable us to model that robots suitable for heavy loads are more likely to be present near the loading dock area. Similarly, if the time is included in the state, we would be able to model dynamic characteristics, e.g. that certain types of robots are only available during regular working hours, when humans are available to supervise. Of course, in simple cases, g might be state-independent. For $s \in \mathcal{S}$, X is a sampling procedure parameterized by the tuple $(s, M, U, C, \boldsymbol{\phi}_M, g_U, g_C)$, where $\boldsymbol{\phi}_M \in [0, 1]^M$ represents the logits of a categorical distribution:

1. Sample an integer N from *Cat*$(\boldsymbol{\phi}_M)$.
2. Sample N controlled agents via g_U from U.
3. Sample $M - N$ uncontrolled agents via g_C from C.

Note that the agent sampling functions g_U, g_C could employ sampling with replacement to model scenarios where two robots may use the same policy, or sampling without replacement for scenarios where that is not possible (e.g. if all robots are heterogeneous).

A.2 Further Discussion of the Motivating Example

This section provides proofs for claims in Section 4, and further discussion in support of the motivating example.

Lemma A.1. *For a team of M agents independently and identically selecting actions with probability p, the p that maximizes the probability of winning is $p = \frac{1}{M}$.*

Proof. The probability of winning, $P(s = 1)$ may be computed as follows:

$$P(s = 1) = M * P(\text{agent } i \text{ chooses 1 and } i^{-} \text{ choose } 0) = Mp(1 - p)^{M-1}.$$

We wish to analytically compute the global maxima of this expression with respect to p, by considering the boundary points $p \in \{0, 1\}$ and the zeros of the first derivative.

Clearly, if all agents select 1 with probability 0, the team cannot win; thus, $p = 0$ is not a maxima. Similarly, if all agents select 1 with probability 1, the team also never wins; thus, $p = 1$ is also not a maxima. Next, we compute the derivative of $P(s = 1)$:

$$\begin{aligned}\frac{d[P(s = 1)]}{dp} &= M[(1 - p)^{M-1} - (M - 1)p(1 - p)^{M-2}] \\ &= M[(1 - p)^{M-2}[(1 - p) - (M - 1)p]].\end{aligned}$$

The zeros of the above expression occur at $p = 0$ and $p = \frac{1}{M}$. For $p = \frac{1}{M}$, note that $P(s = 1) = (\frac{M-1}{M})^{M-1} > 0$; thus, $p = \frac{1}{M}$ must be the global maximum. □

Lemma A.2. *In a team of three agents consisting of two uncontrolled agents who select 1 with probability $p = \frac{1}{3}$, and one controlled ad hoc agent who selects 1 with probability p_{aht}, the probability of winning is $P(s = 1) = \frac{4}{9}$.*

Proof. Let s_u denote the sum of the bits chosen by the two uncontrolled agents, and b_{aht} denote the bit value chosen by the controlled agent. The probability of winning can be computed by partitioning the winning outcomes by whether $s_u = 1$ and the ad hoc agent selects 0, or whether the ad hoc agent selects 1 and both uncontrolled agents select 0, $s_u = 0$.

$$\begin{aligned}
P(s=1) &= P(s_u = 1 \wedge b_{aht} = 0) + P(s_u = 0 \wedge b_{aht} = 1) \\
&= 2 \cdot \frac{1}{3} \cdot \frac{2}{3} \cdot \left(1 - p_{\text{aht}}\right) + p_{\text{aht}} \left(\frac{2}{3} \cdot \frac{2}{3}\right) \\
&= \frac{4}{9} - \frac{4}{9} \cdot p_{\text{aht}} + \frac{4}{9} \cdot p_{\text{aht}} \\
&= \frac{4}{9}.
\end{aligned}$$

□

Lemma A.3. *In a team of three agents where two ad hoc agents select 1 with probability p_{aht} and the remaining uncontrolled agent selects 1 with probability $\frac{1}{3}$, the maximizing $p_{aht} = \frac{1}{3}$ and the corresponding winning probability is $P(s = 1) = \frac{4}{9}$.*

Proof. Let s_{aht} denote the sum of the bits chosen by the two ad hoc agents, and b_u denote the bit value chosen by the uncontrolled agent. The probability of winning may be directly computed as follows:

$$\begin{aligned}
P(s=1) &= P(s_{aht} = 1 \wedge a_u = 0) + P(s_{aht} = 0 \wedge a_u = 1) \\
&= 2 \cdot p_{\text{aht}} \cdot (1 - p_{\text{aht}}) \cdot \frac{2}{3} + (1 - p_{\text{aht}})^2 \cdot \frac{1}{3} \\
&= (1 - p_{\text{aht}}) \left[\frac{4}{3} p_{\text{aht}} + \frac{1}{3}(1 - p_{\text{aht}})\right] \\
&= (1 - p_{\text{aht}}) \left(\frac{1}{3} + p_{\text{aht}}\right).
\end{aligned}$$

To determine the maximizing value, we compute $P(s = 1)$ at the boundary points $p_{\text{aht}} \in \{0, 1\}$ and at the zeros of the derivative of $P(s = 1)$. Note that for $p_{\text{aht}} = 0$, $P(s = 1) = \frac{1}{3}$, while for $p_{\text{aht}} = 1$, $P(s = 1) = 0$.

The derivative of the analytic expression of $P(s = 1)$ with respect to p_{aht} is:

$$\frac{d[P(s=1)]}{dp_{\text{aht}}} = (1 - p_{\text{aht}}) - \left(\frac{1}{3} + p_{\text{aht}}\right) = \frac{2}{3} - 2p_{\text{aht}}. \tag{4}$$

The zeros of the above expression occur at $p_{\text{aht}} = \frac{1}{3}$, which corresponds to $P(s = 1) = \frac{4}{9}$. □

Lemma A.4. *In a team of three agents where one agent always selects 0, one agent selects 1 with probability $\frac{1}{3}$, and the last agent selects 1 with probability p_{naht}, the optimal $p_{naht} = 1$ and results in a winning probability of $P(s = 1) = \frac{2}{3}$.*

Proof. Since one of the two controlled agents always plays 0, the game effectively becomes a two-player game instead.

Let a_u denote the action selected by the uncontrolled agent, and let a_{naht} denote the action of the controlled agent. The probability of winning may be directly computed as follows:

$$\begin{aligned}
P(s=1) &= P(a_u = 1 \wedge a_{naht} = 0) + P(a_u = 0 \wedge a_{naht} = 1) \\
&= \frac{1}{3} \cdot (1 - p_{\text{naht}}) + \frac{2}{3} \cdot p_{\text{naht}} \\
&= \frac{1}{3} + \frac{1}{3} \cdot p_{\text{naht}}.
\end{aligned}$$

It is clear that $p_{\text{naht}} = 1$ maximizes $P(s = 1)$, and the corresponding value of $P(s = 1) = \frac{2}{3}$. □

A.3 Experiment Details

A.3.1 Team Sampling Procedure Used in Experiments

Recall that U and C denote the sets of uncontrolled and controlled agent policies, respectively, while M denotes the team size for the task, and N the number of agents sampled from C. The experiments in the following consider an X_{train} consisting of sampling N uniformly from $\{1, \cdots, M-1\}$, sampling N agents from C and $M-N$ agents from U in a uniform fashion (Figure 7). The sampling procedure takes place at the beginning of each episode to select a team, which is then deployed in the environment. Data generated by the deployed team (e.g. joint observations, joint actions, and rewards) are returned to the learning algorithm. See Appendix A.3.3 for further details on the evaluation procedure.

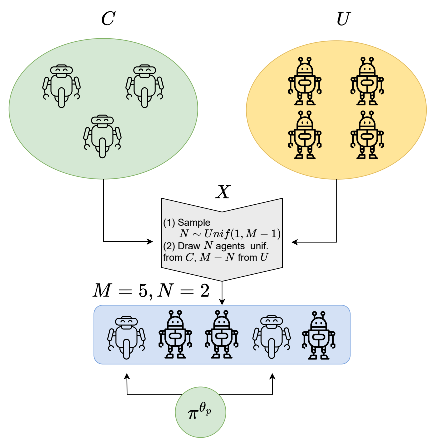
C
U
X
(1) Sample $N \sim Unif(1, M-1)$
(2) Draw N agents unif. from C, $M-N$ from U
$M=5, N=2$
π^{θ_p}

Figure 7: A practical instantiation of the NAHT problem.

A.3.2 Experimental Domains

MPE Predator Prey The MPE environment [24] (released under the MIT License) is a setting where particle agents interact within a bounded 2D plane, equipped with a discrete action space and continuous observation space. The observation space is 16-dimensional, and contains the agents' own position and velocity, relative positions/velocities of all other agents, landmarks, and prey. The action space consists of five discrete actions, corresponding to the four cardinal movement directions and a no-op action. The observation range is normalized to $[-1, 1]$, while the discrete action space is one-hot encoded.

The predator-prey task (`mpe-pp`) is a custom task implemented by the authors of this paper within the fork of the MPE environment released by Papoudakis et al. [30] (MIT License). In this task, three predators must cooperatively pursue a pre-trained prey agent. We use the pre-trained prey policy provided by the ePymarl MARL framework. The prey policy was originally trained by Papoudakis et al. [30], by using the MADDPG MARL algorithm to train both predator and prey agents for 25M steps, and generally attempts to escape approaching predators. The team receives a reward of 1 if two or more predators collide with the prey at a single time step, and no reward if only a single agent collides with the prey. A shaping reward consisting of 0.01 times the ℓ_2 distance between each agents in the team and the prey is provided as well. Since the prey policy is pre-trained and fixed, note that our predator-prey task is a fully cooperative task from the perspective of the predator (learning) agents. The maximum episode length is 100 time steps.

SMAC SMAC [34] (released under the MIT License) features a set of cooperative tasks, where a team of allied agents must defeat a team of enemy agents controlled by the game server. It is a partially observable domain with a continuous state space and discrete action space. For each agent, the observation space is continuous, and consists of features about itself, enemy, and allied agents within some radius. The action space is discrete, and allows an agent to choose an enemy to attack, a direction to move in, or to not perform any action. As the number of allies and enemies varies between tasks, the dimensionality of the observation and action spaces is particular to each task. The observation space is normalized to be between $[-1, 1]$, while the action space is one-hot encoded. The maximum number of time steps per episode is also task specific, although early termination occurs if all enemies are defeated. At each time step, the team receives a shaped reward corresponding to the damage dealt, and bonuses of 10 and 200 points for killing an enemy and winning the scenario by defeating all enemies. The reward is scaled such that the maximal return achievable in each task is 20.

The SMAC tasks considered in this paper are described in more detail below:

- `5v6`: stands for 5m vs 6m, five allied Marines versus six enemy Marines.
- `8v9`: stands for 8m vs 9m, eight allied Marines versus nine enemy Marines.
- `10v11`: stands for 10m vs 11m, ten allied Marines versus eleven enemy Marines.
- `3s5z`: stands for 3s vs 5z, three allied Stalkers versus five enemy Zealots.

A.3.3 Evaluation Details

This section provides details on how mixed teams are evaluated in the NAHT setting, and how cross-play scores, and self-play scores, and uncertainty measures are computed.

$M - N$ score. Given a set of controlled agents C, and a set of uncontrolled agents U, the goal of the $M - N$ score is to quantify the performance when these two teams must cooperate within the NAHT scenario. The $M - N$ score is computed in a deterministic and exhaustive fashion, by iterating over all possible values of N. Let N be the number of agents sampled from set C, such that $N < M$. For $N \in \{1, \cdots, M-1\}$, construct the joint policy $\boldsymbol{\pi}^{(M)}$ by selecting N agents uniformly from C and $M - N$ agents from U. Evaluate the resultant team on the task for E episodes. This results in $(M - 1) * E$ episode returns, which is averaged to form the $M - N$ score.

Cross-play scores. The cross-play scores reported in this paper are the average returns of teams generated by algorithm A, when coordinating with those generated by B. To compute the cross-play score, we first train multiple teams (by varying the seed) via both algorithms A and B. Next, the $M - N$ score is computed for random pairings of teams (seeds) from A and B, following the procedure specified above. Summary statistics can be computed over the set of all such NAHT returns.

For example: each algorithm (VDN, QMIX, IQL, MAPPO, IPPO) is run k times with different seeds, to generate k teams of agents that may act as uncontrolled teammates. For each pair of algorithms, we sample a subset of the possible seed *pairs*, and evaluate the teams that result from merging said seed pairs. If VDN and QMIX have seeds 1 2, and 3, the cross-play evaluation might consider the seed pairings (VDN 1, QMIX 2), (VDN 2, QMIX 3), (VDN 3, QMIX 1). Given a pair of seeds (e.g. (VDN 1), (QMIX 2)), the $M - N$ cross-play score is computed as the average return generated by sweeping $N \in \{1, \cdots, M-1\}$ and evaluating the merged team that consists of selecting N agents from the first team, and $M - N$ agents for E episodes. In our experiments, $E = 128$, and $k = 5$.

Self-play scores. The self-play scores reported in this paper are *model* self-play score, rather than *algorithm* self-play score [3]. The reason for this is that we are interested in the performance of agents when paired with *known teammates*.

Measuring uncertainty Means and 95% confidence intervals are computed over all team/seed pairings considered for any given scores, as we treat each $M - N$ evaluation as an independent trial. For most experiments, five seed pairs are considered, where seeds are paired to ensure that all seeds participate in at least one evaluation. This ensures that the computational cost of the evaluation remains linear in N, rather than quadratic. However, note that for the OOD experiments presented in Section 6.3, we consider all possible seed pairings for the most comprehensive evaluation.

A.3.4 Algorithm Implementation

The experiments in this paper use algorithm implementations from the ePyMARL codebase [30] (released under the Apache License). The value-based methods are used without modification (i.e. IQL, VDN, QMIX), but we implement our version of policy gradient methods (IPPO, MAPPO), based on the implementation of Yu et al. [45].

Parameter Sharing. All methods employ recurrent actors and critics, with full parameter sharing, i.e. all agents are controlled by the same policy, where the agent id is input to the actor and critic networks, to allow behavioral differentiation between agents. For POAM, which also maintains encoder-decoder networks for agent modeling, parameter sharing is used for the encoder and decoder networks, to improve training sample efficiency. Further, POAM's decoder, which predicts observations and actions for all $-i$ agents, employs parameter sharing across agent predictions, to prevent the target dimensionality from scaling with the number of teammates.

Optimizer and Neural Architecture. The Adam optimizer is applied for all networks involved. For policy gradient methods, the policy architecture is two fully connected layers, followed by an RNN (GRU) layer, followed by an output layer. Each layer has 64 neurons with ReLU activation units, and employs layer normalization. The critic architecture is the same as the policy architecture.

The value-based methods employ the same architecture, except that there is only a single fully connected layer before the RNN layers, and layer normalization is not used, following the ePyMARL implementation. Please consult the codebase for full implementation details.

Algorithm	Buffer size	Epochs	Minibatches	Entropy	Clip	Clip value loss
IPPO	128, **256**, 512	1, **4**, 10	1, **3**	0.01, 0.03, **0.05**	0.01, 0.05, **0.1**	**no**, yes
MAPPO	64, 128, **256**	4, **10**	**1**, 3	0.01, **0.03**, 0.05, 0.07	**0.05**, 0.1, 0.2	no, **yes**

Table 1: Hyperparameters evaluated for the policy gradient algorithms. Selected values are bolded

Hyperparameters. For the value-based methods, default hyperparameters are used. We tune the hyperparameters of the policy gradient methods on the `5v6` task, and apply those parameters directly to the remaining SMAC tasks. The hyperparameters considered for policy gradient algorithms are given in Table 1. POAM adopts the same hyperparameters as IPPO where applicable. We also tuned additional hyperparameters specific to POAM (Table 2).

Task	Algorithm	ED epochs	ED Minibatches	ED LR
`5v6`	POAM	**1**, 5, 10	**1**, 2	**0.0005**, 0.005
`mpe-pp`	POAM	**1**, 5	**1**, 2	**0.0005**, 0.001

Table 2: Additional hyperparameters evaluated for POAM; note that ED stands for encoder-decoder. Selected values are bolded.

A.4 Supplemental Figures

This section contains additional figures referenced by the main paper. Please see Section 6 for the corresponding analysis and discussion.

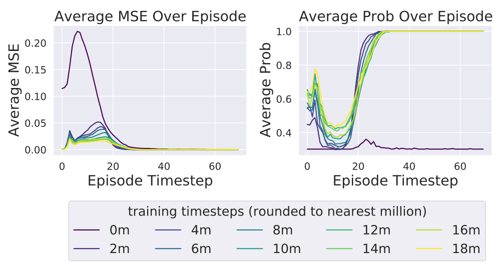

Figure 8: Evolution of the within-episode mean squared error (left) and probability of actions that were actually taken by modeled teammates (right), computed by the POAM agent over the course of training, on the `5v6` task. The agent modeling performance improves over the course of an episode, as more data about teammate behavior is observed.

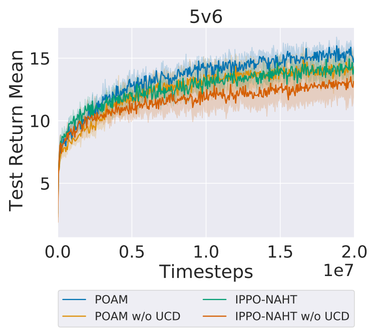

Figure 9: Learning curves of POAM and IPPO, where the value network is trained with and without uncontrolled agents' data (UCD) on `5v6`.

A.5 Supplemental Analysis

The following subsections contains secondary analysis and results, intended to support the primary analysis of the main paper. For all results, the mean and 95% confidence interval over five trials is reported. All reported returns are *test* returns.

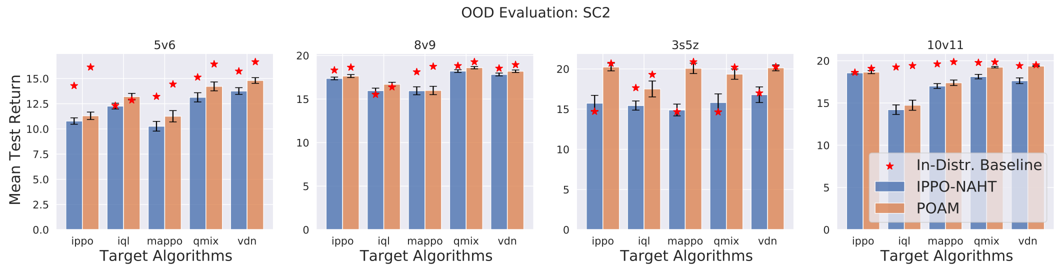

Figure 10: Returns achieved by POAM and IPPO-NAHT, when paired with out-of-distribution teammates. POAM shows improved generalization to OOD teammates, as compared to IPPO-NAHT, across all StarCraft tasks. For each type of teammate, the performance of IPPO-NAHT/POAM against the exact teammate seen during training is shown as the in-distribution baseline.

A.5.1 The Need for Dedicated NAHT Algorithms - Empirical Evidence

Section 4 argues theoretically that with a population of uncontrolled teammates that select one with probability 1/3 and zero otherwise, an optimal policy learned in the AHT setting would not be optimal in the NAHT setting. Here, we present experiments on the three agent bit matrix game that empirically verify the theory. Let N denote the number of controlled agents. The episode length is 25, and the observation for each agent consists of the agent index and the joint action at the previous time step. The team reward at each time step is $3 * \mathbb{1}_{\sum_i b_i = 1}$.

The optimal expected return in the $N = 1$ (AHT) setting is 33.333 (derived from Lemma A.3), and in the $N = 2$ (NAHT) setting is 50.0 (derived from Lemma A.4). The methods compared are POAM-AHT and POAM. Table 3 shows the expected returns achieved by POAM and POAM-AHT on the $N = 1$ and $N = 2$ scenarios. Since any policy is optimal in the $N = 1$ case, as expected, both methods achieve near-optimal returns. In the $N = 2$ case, as expected, POAM outperforms POAM-AHT by a large margin, achieving a near-optimal return of 48.858 ± 1.092.

	N=1 (AHT)	N=2 (NAHT)
POAM-AHT	33.441 $\pm$ 0.511	22.5 $\pm$ 8.250
POAM	33.483 $\pm$ 0.485	48.858 $\pm$ 1.092

Table 3: Returns of POAM-AHT versus POAM on the three agent bit matrix game, in the N=1 (AHT) and N=2 (NAHT) setting. POAM and POAM-AHT both achieve the optimal return for the $N = 1$ case, but POAM has a much higher return on the $N = 2$ case.

A.5.2 Validating the Existence of Coordination Conventions

The experimental procedure detailed in Section 6.1 generates diverse teammates in SMAC and MPE by using MARL algorithms to train multiple teams of agents. Two underlying assumptions of the procedure are that (1) teams trained by the same algorithm learn non-compatible coordination conventions, and (2) teams trained by different algorithms are not compatible. Both points are experimentally verified in this section.

Self-play with MARL algorithms. For the tasks under consideration, teams trained using the same MARL algorithm, but different seeds, can converge to distinct coordination conventions. Figure 11 demonstrates this by depicting the return of teams trained together (matched seeds) versus those of teams that were not trained together (mismatched seeds), across all naive MARL algorithms (IPPO, IQL, MAPPO, QMIX, VDN) and tasks considered. Overall, the returns of the teams that were trained together are higher than those not trained together.

The general phenomenon has been previously observed and exploited by prior works in ad hoc teamwork [40]. This paper takes advantage of this to generate a set of diverse teammates for the StarCraft II experiments. We select tasks where the effect is significant, to ensure that there are distinct coordination behaviors for POAM to model. Tasks that were considered but subsequently ruled out for this reason include `3m vs 3m, 8m vs 8m`, `6h vs 8z`, and the MPE Spread task.

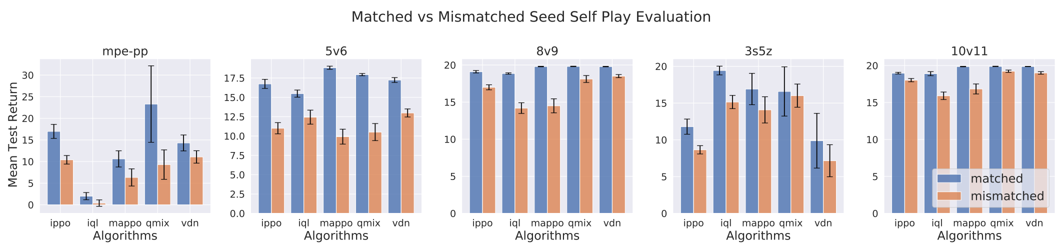

Figure 11: Agent teams that were trained together using the same algorithm (`matched` seeds) have higher returns than teams that were not trained together but were trained with the same algorithm (`mismatched` seeds).

	vdn	qmix	iql	ippo	mappo	poam-aht	mean xp score
vdn	14.305 +/- (1.844)	8.657 +/- (2.925)	4.423 +/- (1.677)	2.374 +/- (0.730)	1.912 +/- (1.394)	8.101 +/- (1.164)	4.342 +/- (1.244)
qmix	8.657 +/- (2.925)	23.297 +/- (8.845)	3.359 +/- (1.691)	2.692 +/- (1.522)	2.691 +/- (1.365)	10.287 +/- (1.297)	4.350 +/- (1.255)
iql	4.423 +/- (1.677)	3.359 +/- (1.691)	1.997 +/- (0.835)	1.520 +/- (0.820)	0.325 +/- (0.552)	4.809 +/- (1.924)	2.407 +/- (0.811)
ippo	2.374 +/- (0.730)	2.692 +/- (1.522)	1.520 +/- (0.820)	16.974 +/- (1.621)	8.776 +/- (1.244)	13.584 +/- (1.412)	3.840 +/- (1.056)
mappo	1.912 +/- (1.394)	2.691 +/- (1.365)	0.325 +/- (0.552)	8.776 +/- (1.244)	10.616 +/- (1.874)	8.809 +/- (2.012)	3.426 +/- (1.157)
poam-aht	8.101 +/- (1.164)	10.287 +/- (1.297)	4.809 +/- (1.924)	13.584 +/- (1.412)	8.809 +/- (2.012)	15.727 +/- (1.290)	9.118 +/- (1.068)

Table 4: Cross-play results for the `mpe-pp` task.

	vdn	qmix	iql	ippo	mappo	poam-aht	mean xp score
vdn	17.239 +/- (0.315)	12.164 +/- (1.142)	11.851 +/- (0.988)	9.437 +/- (0.955)	8.791 +/- (0.786)	11.521 +/- (1.278)	10.561 +/- (0.585)
qmix	12.164 +/- (1.142)	17.935 +/- (0.146)	9.857 +/- (0.672)	11.758 +/- (1.138)	10.623 +/- (1.188)	11.991 +/- (1.051)	11.101 +/- (0.565)
iql	11.851 +/- (0.988)	9.857 +/- (0.672)	15.482 +/- (0.462)	9.106 +/- (0.622)	8.657 +/- (0.747)	10.340 +/- (0.971)	9.868 +/- (0.469)
ippo	9.437 +/- (0.955)	11.758 +/- (1.138)	9.106 +/- (0.622)	16.737 +/- (0.587)	11.168 +/- (0.878)	11.373 +/- (1.154)	10.367 +/- (0.520)
mappo	8.791 +/- (0.786)	10.623 +/- (1.188)	8.657 +/- (0.747)	11.168 +/- (0.878)	18.826 +/- (0.205)	10.345 +/- (1.233)	9.810 +/- (0.518)
poam-aht	11.521 +/- (1.278)	11.991 +/- (1.051)	10.340 +/- (0.971)	11.373 +/- (1.154)	10.345 +/- (1.233)	10.502 +/- (0.585)	11.114 +/- (0.527)

Table 5: Cross-play results for the `5v6` task.

	vdn	qmix	iql	ippo	mappo	poam-aht	mean xp score
vdn	19.788 +/- (0.036)	18.631 +/- (0.207)	15.520 +/- (0.838)	16.899 +/- (0.571)	14.999 +/- (0.952)	14.330 +/- (1.002)	16.512 +/- (0.422)
qmix	18.631 +/- (0.207)	19.811 +/- (0.035)	15.754 +/- (0.819)	17.588 +/- (0.477)	15.338 +/- (0.799)	14.815 +/- (1.004)	16.828 +/- (0.385)
iql	15.520 +/- (0.838)	15.754 +/- (0.819)	18.854 +/- (0.093)	14.229 +/- (0.863)	12.995 +/- (0.878)	13.053 +/- (0.953)	14.625 +/- (0.463)
ippo	16.899 +/- (0.571)	17.588 +/- (0.477)	14.229 +/- (0.863)	19.093 +/- (0.163)	15.390 +/- (0.825)	15.197 +/- (0.922)	16.027 +/- (0.413)
mappo	14.999 +/- (0.952)	15.338 +/- (0.799)	12.995 +/- (0.878)	15.390 +/- (0.825)	19.795 +/- (0.037)	13.965 +/- (0.999)	14.680 +/- (0.462)
poam-aht	14.330 +/- (1.002)	14.815 +/- (1.004)	13.053 +/- (0.953)	15.197 +/- (0.922)	13.965 +/- (0.999)	12.370 +/- (0.469)	14.272 +/- (0.450)

Table 6: Cross-play results for the `8m` task.

	vdn	qmix	iql	ippo	mappo	poam-aht	mean xp score
vdn	9.879 +/- (3.722)	13.742 +/- (3.270)	13.345 +/- (2.114)	8.061 +/- (1.981)	8.689 +/- (1.618)	19.367 +/- (0.601)	10.959 +/- (1.416)
qmix	13.742 +/- (3.270)	16.582 +/- (3.350)	15.780 +/- (2.290)	11.130 +/- (3.400)	14.541 +/- (2.466)	19.333 +/- (0.814)	13.798 +/- (1.542)
iql	13.345 +/- (2.114)	15.780 +/- (2.290)	19.429 +/- (0.594)	12.757 +/- (1.224)	11.704 +/- (1.275)	17.131 +/- (0.895)	13.397 +/- (1.009)
ippo	8.061 +/- (1.981)	11.130 +/- (3.400)	12.757 +/- (1.224)	11.794 +/- (1.034)	11.291 +/- (1.297)	19.609 +/- (1.080)	10.810 +/- (1.203)
mappo	8.689 +/- (1.618)	14.541 +/- (2.466)	11.704 +/- (1.275)	11.291 +/- (1.297)	16.910 +/- (2.133)	20.257 +/- (0.793)	11.556 +/- (1.079)
poam-aht	19.367 +/- (0.601)	19.333 +/- (0.814)	17.131 +/- (0.895)	19.609 +/- (1.080)	20.257 +/- (0.793)	20.172 +/- (0.579)	19.139 +/- (0.480)

Table 7: Cross-play results for the `3s5z` task.

Cross-play with MARL algorithms Tables 4, 5, 6, 7, and 8 display the full cross-play results for all MARL algorithms and POAM-AHT, on all tasks. Note that the tables are reflected across the diagonal axis for viewing ease. The values on the off-diagonal (i.e., where the two algorithms are not the same) are the cross-play score, while the values shown on the diagonal are self-play scores,

	vdn	qmix	iql	ippo	mappo	poam-aht	mean xp score
vdn	19.868 +/- (0.021)	18.980 +/- (0.260)	18.276 +/- (0.274)	18.638 +/- (0.197)	16.453 +/- (0.864)	16.328 +/- (0.714)	18.087 +/- (0.280)
qmix	18.980 +/- (0.260)	19.884 +/- (0.034)	17.348 +/- (0.453)	18.193 +/- (0.311)	18.608 +/- (0.370)	16.395 +/- (0.684)	18.282 +/- (0.199)
iql	18.276 +/- (0.274)	17.348 +/- (0.453)	18.901 +/- (0.279)	16.610 +/- (0.547)	16.206 +/- (0.692)	14.826 +/- (0.650)	17.110 +/- (0.282)
ippo	18.638 +/- (0.197)	18.193 +/- (0.311)	16.610 +/- (0.547)	18.971 +/- (0.112)	17.048 +/- (0.559)	16.125 +/- (0.566)	17.622 +/- (0.247)
mappo	16.453 +/- (0.864)	18.608 +/- (0.370)	16.206 +/- (0.692)	17.048 +/- (0.559)	19.862 +/- (0.039)	15.670 +/- (0.708)	17.079 +/- (0.351)
poam-aht	16.328 +/- (0.714)	16.395 +/- (0.684)	14.826 +/- (0.650)	16.125 +/- (0.566)	15.670 +/- (0.708)	12.179 +/- (0.151)	15.869 +/- (0.308)

Table 8: Cross-play results for the `10v11` task.

computed as described in App. A.3.3. The cross-play and self-play scores displayed are means and 95% confidence intervals. The rightmost column reflects the row average and 95% confidence interval of the cross-play (XP) scores corresponding to the test set of VDN, QMIX, IQL, IPPO, and MAPPO, excluding the self-play score. Thus, the rightmost column reflects how well on average the row MARL/AHT algorithm can generalize to the test set used throughout this paper.

Overall, we find that MARL algorithms perform significantly better in self-play than cross-play. We note that there are a few exceptions (e.g., IPPO vs QMIX on `8m`), and that the cross-play and self-play scores are much closer on `10v11`, but overall the trend is consistent across tasks.

A.5.3 Generalization to Unseen Teammate Types

As discussed and experimentally verified in Section A.5.2, diverse coordination conventions in StarCraft and MPE Predator Prey can be generated by (1) running the same MARL algorithm with various random seeds, or (2) running various MARL algorithms. In Section 6.3, the out-of-distribution (OOD) teammates considered were generated by running MARL algorithms with different random seeds than those used to generate train-time teammates—in other words, generating diverse and unseen teammates using the first procedure specified above.

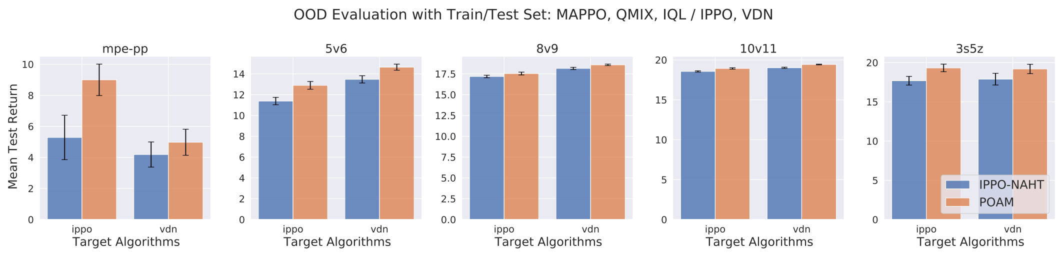

Figure 12: Returns achieved by POAM and IPPO-NAHT, when trained on MAPPO, QMIX, and IQL, and tested on IPPO, VDN.

Here, we generate OOD teammates using the second procedure. More precisely, the five MARL algorithms used to generate teammates may be divided into train/test sets. Figure 12 shows the results of such an experiment, where POAM and IPPO-NAHT are trained with MAPPO, QMIX, and IQL teammates, and tested with agents from IPPO and VDN. Similar to the results presented in Section 6.3, we find that POAM generally outperforms IPPO-NAHT for the unseen teammate types for all tasks.

A.5.4 Modeling Controlled and Uncontrolled Agents

POAM's encoder-decoder (ED) models both controlled and uncontrolled agents. Since the controlled agents are updated during the training process, the encoder-decoder must deal with a "moving target". Thus, in principle, the problem of modeling the controlled agents is more challenging than modeling uncontrolled agents.

Fig. 13 shows the probability of predicting the correct *action* for the uncontrolled agents and controlled agents separately, on the `mpe-pp` task. Note that the action probabilities shown in Fig. 5 (right) of the main paper would be the average of the two plots shown in Fig. 13.

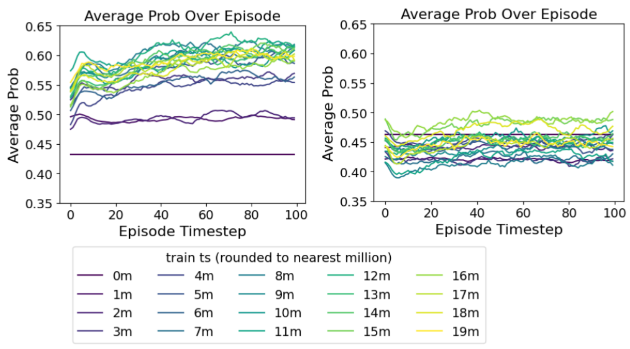

Figure 13: Loss of POAM's encoder-decoder on the `mpe-pp` task, separated by uncontrolled (left) and controlled agents (right).

For uncontrolled agents (Fig. 13, left), we observe that the accuracy of action predictions for the uncontrolled agents increases much more consistently as training goes on, and is higher than that for the controlled agents. As expected, the ED is able to model the uncontrolled agents more easily than the nonstationary controlled agents.

A.5.5 Performance as a Function of the Number of Controlled Agents

To provide more insight on how an NAHT team's performance changes as the number of controlled agents increases, Figure 14 displays the mean test returns of POAM and POAM-AHT as a function of the number of controlled agents, where the evaluation returns are averaged across five types of uncontrolled teammates: QMIX, VDN, IQL, MAPPO, and IPPO. Note that the self-play returns of POAM and POAM-AHT (i.e. where the number of controlled agents is maximal) are shown as horizontal lines, while the performances at $N = 0$ correspond to the averaged self-play returns of the five uncontrolled team types.

Recall that POAM-AHT was trained on the $N = 1$ scenario only, while POAM is trained on the full NAHT setting. As expected, POAM outperforms POAM-AHT for all values of $N > 1$, while for $N = 1$, the methods perform similarly. POAM-AHT's performance declines as the number of controlled agents increases, which likely occurs because the evaluation setting becomes further from the training setting as N increases.

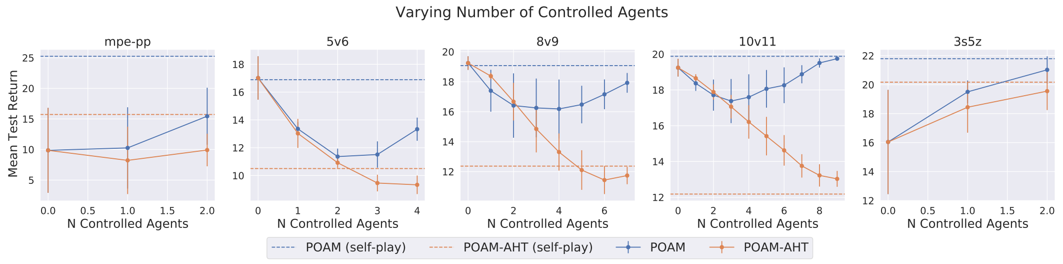

Figure 14: Comparing the performance of POAM versus POAM-AHT, as the number of controlled agents varies. POAM and POAM-AHT agents are evaluated with the following set of uncontrolled agents: QMIX, VDN, IQL, MAPPO, IPPO.

A.6 Computing Infrastructure

All value-based algorithms (e.g. QMIX, VDN, IQL) were run without parallelizing environments, while policy gradient algorithms were run with parallelized environments. All methods were trained for 20M steps on all tasks. Each run took between 12-48 hours of compute, and used less than 2gB of GPU memory. Runs were parallelized to use computational resources efficiently. The servers used for our experiments ran Ubuntu 20.04 with the following configurations:

- Intel Xeon CPU E5-2630 v4; Nvidia Titan V GPU.
- Intel Xeon CPU E5-2698 v4; Nvidia Tesla V100-SXM2 GPU.
- Intel Xeon Gold 6342 CPU; Nvidia A40 GPU.
- Intel Xeon Gold 6342 CPU; Nvidia A100 Gpu.