applied sciences

MDPI

Article

Learning Ad Hoc Cooperation Policies from Limited Priors via Meta-Reinforcement Learning

Qi Fang, Junjie Zeng, Haotian Xu, Yue Hu and Quanjun Yin *

College of Systems Engineering, National University of Defense Technology, Changsha 410073, China; fangqinudt@nudt.edu.cn (Q.F.); zengjunjie13@nudt.edu.cn (J.Z.); xuhaotian@nudt.edu.cn (H.X.); huyue11@nudt.edu.cn (Y.H.)

* Correspondence: yinquanjun@nudt.edu.cn

Abstract: When agents need to collaborate without previous coordination, the multi-agent cooperation problem transforms into an ad hoc teamwork (AHT) problem. Mainstream research on AHT is divided into type-based and type-free methods. The former depends on known teammate types to infer the current teammate type, while the latter does not require them at all. However, in many real-world applications, the complete absence and sufficient knowledge of known types are both impractical. Thus, this research focuses on the challenge of AHT with limited known types. To this end, this paper proposes a method called a Few typE-based Ad hoc Teamwork via meta-reinforcement learning (FEAT), which effectively adapts to teammates using a small set of known types within a single episode. FEAT enables agents to develop a highly adaptive policy through meta-reinforcement learning by employing limited priors about known types. It also utilizes this policy to generate a diverse type repository automatically. During the ad hoc cooperation, the agent can autonomously identify known teammate types followed by directly utilizing the pre-trained optimal cooperative policy or swiftly updating the meta policy to respond to teammates of unknown types. Comprehensive experiments in the pursuit domain validate the effectiveness of the algorithm and its components.

Keywords: ad hoc teamwork; meta-reinforcement learning; self-play; agent types; online adapting

check for updates

Citation: Fang, Q.; Zeng, J.; Xu, H.; Hu, Y.; Yin, Q. Learning Ad Hoc Cooperation Policies from Limited Priors via Meta-Reinforcement Learning. *Appl. Sci.* **2024**, *14*, 3209. https://doi.org/10.3390/app14083209

Academic Editors: He Cai and Maobin Lv

Received: 19 March 2024
Revised: 5 April 2024
Accepted: 8 April 2024
Published: 11 April 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

In traditional multi-agent cooperation problems, prior to being deployed, the participating agents can engage in multiple episodes of coordination to establish a commonly accepted protocol, thereby enhancing their task performance. Numerous studies have been conducted in this regard, such as QMIX [1], QTRAN [2], and MADDPG [3]. However, as the number of intelligent agents increases and the demand for collaboration among them expands, situations arise where agents lack the time for prearranged coordination and must collaborate with unfamiliar teammates. Imagine a scenario where a sudden outbreak of an epidemic occurs in a certain area, requiring urgent mobilization of medical personnel from other regions to assist in overcoming this crisis. In such emergencies, the temporarily deployed personnel may not have sufficient time to communicate and develop a comprehensive plan. Moreover, unfamiliarity exists between the local medical staff and the externally deployed personnel. In the absence of prior coordination, agents required to collaborate with unknown teammates are termed ad hoc agents, and the problem of cooperation with unknown teammates is referred to as an ad hoc teamwork (AHT) problem [4]. This presents one of the current challenges faced by multi-agent systems, with implications for emergency response, disaster management, and other dynamic environments.

Existing AHT methods are broadly categorized into two categories, i.e., type-based and type-free approaches. Type-based approaches typically assume that agents have knowledge about a set of known types of teammates, referred to as a type repository. Agents can acquire

optimal reactive policies by pre-training with these teammates, enabling them to infer the types of current teammates during online cooperation based on information such as their trajectories. Consequently, they select cooperative policies accordingly. Representative works in this area include the tactical adaptation method proposed by Bowling et al. [5,6] for the CMDragons robot soccer competition. Since 2010, some type-based approaches have also addressed scenarios where the current teammate is not included in the type repository, such as HBA [7], E-HBA [8], PLASTIC [9], and TEAMSTER [10]. Their focus lies in learning the policies of unknown teammates and planning the actions of the ad hoc agent based on these policies. To accurately learn these policies and effectively plan behaviors, a comprehensive environmental model or multiple episode samples are generally required. However, in most real-world scenarios, achieving this is not feasible.

The type-free approach argues that providing a set of known teammate types is impractical and ensuring the diversity of manually collected teammate types is challenging. Furthermore, if the type of a teammate is not included in the type repository, ad hoc cooperation performance tends to decline. Therefore, the focus of type-free approaches is to address this challenge by directly adapting to unknown teammates online, such as OPAT [11]; or by equipping agents with diversified behaviors to adapt to them, as demonstrated in EDRQN [12]; or by automatically generating a diverse range of teammate types for the agents, as seen in methods like BRDiv [13], L-BRDiv [14], and the research of Canaan et al. [15]. However, in practical scenarios, completely lacking prior knowledge of teammate types is also unrealistic. For instance, in the mentioned case of a pandemic outbreak, temporarily deployed personnel typically possess some level of medical treatment experience to fulfill their roles. These ad hoc agents may have prior collaboration experience with other agents, thereby mastering partial cooperative expertise. Utilizing the prior knowledge of a few known teammate types can also enhance the performance of the ad hoc agent in task completion.

Therefore, this paper focuses on investigating a problem setting of more practical interests than both type-based and type-free assumptions, i.e., scenarios where priors about a limited number of teammate types can be leveraged by the ad hoc agent and only a single episode is allowed for adapting to an unknown teammate. Meanwhile, to more accord with real-world application requirements, the agent is provided with sparse rewards but no environmental model. As a result, we present a method that relies on meta-reinforcement learning and self-play to utilize the prior knowledge about a small set of known types for adaptive cooperation with various teammates, called a Few typE-based Ad hoc Teamwork via meta-reinforcement learning (FEAT). The general motivation of this paper is to achieve a combination of (a) offline pre-training for acquiring a repository of diverse teammate types and a highly adaptive policy and (b) online fast adaptation for performing agile cooperation with encountered teammates of known or unknown types.

For offline pre-training, we treat the collaboration of the ad hoc agent with different known types of teammates as distinct tasks. Based on this idea, a meta-reinforcement learning (Meta-RL) policy that exhibits sensitivity to task parameters and rapid adaptability to new tasks is acquired. By employing the policy in self-play within environments characterized by stochastic reward perturbations, FEAT produces an extensive repository of clustered usable types along with corresponding collaborative policies, thereby avoiding inefficient online zero-shot adaptation. This reduces the probability of encountering an unknown teammate type during the testing phase, with which the ad hoc agent has no satisfactory way to cooperate. During online inference, if the agent can identify the actual type of the current teammate by leveraging prior knowledge of known teammate types, the agent can swiftly utilize the optimal cooperative policy. Otherwise, due to the high adaptability of the Meta-RL policy, the ad hoc agent can quickly update their policy within a single episode by imitating teammate behaviors without the need for environmental rewards and models. Experimental results in a pursuit domain have confirmed the effectiveness of our approach when there is already only a limited number of known teammate types.

To sum up, the main contributions of this paper are as follows. First, we propose a method that maximizes the utilization of a few known teammate types to develop a highly adaptive ad hoc agent policy. This enhances the efficiency of the ad hoc agent in adapting to unknown teammates. Second, we utilize this policy to automatically generate a diverse type repository, thereby improving the usage efficiency of known types and expanding the coverage of the repository for test teammates. Finally, we introduce a training approach for the ad hoc agent based on environmental characteristics, which does not rely on environmental models or rewards, and we demonstrate its effectiveness in a pursuit domain. The organization of this paper is as described below. Section 2 introduces relevant research related to this work. Section 3 covers the background knowledge involved in this work. Section 4 presents the primary methods of this work. Section 5 conducts benchmark experiments and ablation studies to explore the effectiveness of the algorithm, and Section 6 concludes the paper.

2. Related Work

This section presents an overview of related work on ad hoc teamwork, covering both type-based and type-free approaches.

2.1. Type-Based Ad Hoc Teamwork

In current research on AHT, studies focusing on type-based approaches are the most prevalent. This line of approaches typically involves preparing a range of potential teammate types for ad hoc agents. Key aspects of their investigation include (a) how to infer the actual type of teammates from already known types based on a limited set of observations and (b) how to devise cooperative policies for ad hoc agents based on known teammate types.

For key point (a), the repository is typically composed of manually designed teammate types. For example, Bowling et al. [5] developed a comprehensive tactical library. Also, Barrett et al. [9] utilized externally created soccer robot teams as their known teammate types. Moreover, some studies acknowledge the absence of explicit teammate types in the repository, thus requiring learning teammate types online. For instance, Barrett et al. [16] proposed the TwoStageTransfer [17] algorithm to learn unknown teammate types based on weighted models from existing manually designed types. This method, utilizing the C4.5 decision tree learner, although straightforward, is computationally intensive. PLASTIC-Model [9] learns models of teammate types from scratch via supervised learning, while PLASTIC-Policy [9] employs Deep Q-Networks (DQN) [18] to learn ad hoc agent policies through online trajectory sampling; both exhibit low utilization rates of known teammate types. Moreover, PLASTIC-Policy often requires multiple episodes to learn effective cooperative policies in sparse-reward environments, whereas our approach only needs a single episode. Additionally, the CTCAT [19] algorithm addresses the issue of mixed types in the repository, which differs from the focus of this study.

The manually designed types and the types requiring online learning together form the type repository, provided to the ad hoc agent during the testing phase, known as the testing type repository. Based on it, Bowling et al. [5,6] proposed, in the context of small-scale robot soccer leagues, that ad hoc agents maintain a selection weight for three tactics and employ a weight update algorithm to choose the most suitable tactic dynamically. However, this approach relies on environmental feedback rewards, which may be very sparse in real-world cases. The AATEAM [20] algorithm employs attention mechanisms to measure the similarity between current teammate types and those in the testing types repository. However, this method requires a large amount of data to train the attention network, which is inefficient. Barrett et al. and Stone et al. [9,16,21] introduced the Bayesian theorem to compute beliefs that the actual teammate type belongs to the testing type repository, and they employed a polynomial weighting algorithm for updating. Albrecht et al. [7] modeled the AHT problem as stochastic Bayesian games (SBG) and utilized time-specific weighted posteriors for belief updates in the known type repository. All these methods require a

repository of a sufficient number of prepared types as support. Comparatively, in our study, there is no need to manually specify a sufficient type repository in advance.

Concerning essential point (b), Stone et al. [16,22] utilized the Monte Carlo Tree Search (MCTS) [23] method to strategize agent behavior in a pursuit domain, while PLASTIC-Model [9] and TEAMSTER [10] employ the Upper Confidence Bounds for Tree (UCT) [23] algorithm for planning ad hoc agent behavior. However, all these methodologies require accurate teammate and environmental models, a condition not stipulated in our setup. Meanwhile, PLASTIC-Policy [9] introduces the Fitted Q Iteration (FQI) [24] algorithm, which eliminates the necessity for an environmental model. However, it mandates multiple episodes for task completion, unlike our method, which accomplishes environmental adaptation within a single episode.

Charakorn et al. [25] introduced the Meta-RL algorithm into the domain of AHT problems, which is similar to our approach. However, their experimental setting is comparatively simplistic, with rewards provided at every step, facilitating the rapid adaptation of the Meta-RL policy. This is not applicable in our sparse-reward environment. Furthermore, they only employed the Meta-RL model during the testing phase, resulting in its underutilization. In contrast, we utilized the Meta-RL policy during the pre-training phase to rapidly generate a comprehensive range of the type repository, thereby notably improving the utilization of Meta-RL models and the efficiency in tackling AHT problems.

2.2. Type-Free Ad Hoc Teamwork

Research on AHT without reliance on priors of teammate types has gained traction in recent years. It eliminates the need to manually design a repository of extensive teammate types, instead opting for approaches such as online adaptation or automated generation of a repository with diverse types.

Among the first kind of solution, Wu et al. [11] treated AHT problems as a series of sequential games, utilizing UCT [23] for online planning of agent behaviors. However, this method entails an environmental model, which is not practical. The EDRQN [12] algorithm introduces the policy entropy to generate ad hoc agents with diverse behaviors, yet their approach generates black-box models, making it challenging to ensure adaptability to teammates in real-world environments.

As for the second kind of solution, Canaan et al. [15] proposed a population generation algorithm that leverages quality diversity to acquire a diverse population of Hanabi agents. However, their approach is limited by its reliance on domain-specific knowledge of Hanabi, hindering its applicability to other domains. The BRDiv [13] algorithm avoids generating teammate types with superficially different behaviors, automatically creating a diverse range of type repositories. Meanwhile, the L-BRDiv [14] algorithm introduces Minimum Coverage Sets (MCSs) in AHT problems to obtain the optimal response policy set against the policy of any teammate type. However, they did not employ any known types, which is a practice deemed unrealistic in this study. A small set of known types can significantly enhance the efficiency of generating diverse repositories. Therefore, this paper focuses on utilizing a limited set of known types to automatically generate teammate types and leveraging these to adapt to unknown teammates.

3. Preliminaries

This section presents our preliminary work. We formalize the problem of ad hoc teamwork, followed by an introduction to the meta-reinforcement learning algorithm called Model-Agnostic Meta Learning (MAML) [26].

3.1. Ad Hoc Teamwork

The ad hoc teamwork issue is a sequential decision problem. Sequential decision problems are typically solved based on the Markov Decision Process (MDP). Due to the partial observability of the ad hoc agent in our problem and the presence of multiple agents in the ad hoc team, we adopt the Decentralized Partially Observable Markov Decision

Process (Dec-POMDP) [27] to comprehensively represent the ad hoc team problem. Based on Dec-POMDP, the model of the AHT problem is as follows:

$$\langle n, \mathcal{I}, \Theta, \mathcal{S}, \mathcal{A}, P, R, Z, \mathcal{O}, \gamma \rangle,$$

where:

- n denotes the quantity of agents within the ad hoc team.
- $\mathcal{I}$ represents all agents in the ad hoc team, where $I^0 \in \mathcal{I}$ denotes the ad hoc agent, which is the agent under our control, and $I^{-0} \subset \mathcal{I}$ represents all agents in the ad hoc team except the ad hoc agent.
- Θ denotes the set of known teammate types, comprising m types. Θ^{-0} denotes the true types of all I^0's teammates, with no limitation that $\Theta^{-0} \subseteq \Theta$.
- $S_t \in \mathcal{S}$ represents some feature representation of the environmental state observed by all agents at time t.
- $A_t \in \mathcal{A}$ represents the joint action of all agents at time t.
- $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ denotes the transition function of the environment.
- $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ represents the reward function of the agents.
- $Z : \mathcal{S} \times \mathcal{A} \to \mathcal{O}$ denotes the observation function of all agents.
- $O_t \in \mathcal{O}$ represents the observation obtained by the agents based on the observation function $Z(S_t = \mathbf{s}, A_t = \mathbf{a})$.
- $\gamma \in (0, 1]$ represents the reward discount factor, which captures the uncertainty of the future.

Each agent $I \in \mathcal{I}$ has a policy $\pi : \mathcal{O} \times \mathcal{A} \to [0, 1]$ and interacts with the environment to generate a series of action–observation histories $\tau_t = \{O_0, A_0, \cdots, O_t, A_t\}$. Here, the policy of the ad hoc agent is denoted by π^0, and the policy set of all its teammates is denoted by π^{-0}. The task of the AHT problem is to enable agent I^0 to learn a policy π^0 that maximizes the collective reward through cooperation with teammates:

$$J(\pi^0, \pi^{-0}) = \mathbb{E}\left[\sum_{t=0}^{H} \gamma^t R_t\right], \tag{1}$$

where H classifies the maximum length of the ad hoc teamwork environment.

3.2. Model-Agnostic Meta Learning

MAML [26] is a Meta-RL method that does not require an environment or opponent model, making it applicable to various learning problems. The algorithm trains the initial parameters of the model through dual-level gradient updates. By updating the gradients of model parameters' gradients, the parameters can exhibit strong performance in a new task after one or more gradient steps. Inspired by MAML, this paper treats collaborating with different teammates in the ad hoc teamwork problem as distinct tasks, defined as follows:

$$\mathcal{T} = \{\pi^{-0}(A_t^{-0} = \mathbf{a}_t^{-0} | O_t^{-0} = \mathbf{o}_t^{-0})\}, \tag{2}$$

where π^{-0} represents the policies of teammates. Utilizing prior knowledge of teammate types refers to employing teammate policies in this context. All possible tasks constitute a task distribution $p(\mathcal{T})$.

A known set of teammate types Θ for each combination of teammate types $\{\Theta_i\}_{n-1} \subseteq \Theta$, corresponding to a task $\mathcal{T}_i \sim p(\mathcal{T})$, is given. The objective is to find a policy $\pi^0(\cdot|\theta)$ (denoted as $\pi^0(\theta)$) for the ad hoc agent that performs well on average across all tasks, where θ is the parameters of the policy. This is achieved by minimizing the cross-task loss function:

$$\mathcal{L}_{\mathcal{T}}\left(\pi^0(\theta)\right) = \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}\left(\pi^0(\theta')\right) = \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}\left[\pi^0(\theta) - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}\left(\pi^0(\theta)\right)\right], \tag{3}$$

where the loss function for each task is computed as:

$$\mathcal{L}_{\mathcal{T}_i}\left(\pi^0(\theta)\right) = -\mathbb{E}_{\mathbf{o}_t,\mathbf{a}_t\sim\pi^0(\theta),\pi_i^{-0}}\left[\sum_{t=1}^{H} R_i(O_t=\mathbf{o}_t, A_t=\mathbf{a}_t)\right]. \tag{4}$$

Updating the cross-task loss function through stochastic gradient descent:

$$\theta \leftarrow \theta - \beta\nabla_\theta \sum_{\mathcal{T}_i\sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}\left(\pi^0(\theta')\right), \tag{5}$$

which constitutes a double-loop gradient update process. α and β are hyperparameters.

4. Method

In this section, we will elaborate on the details of the FEAT algorithm. Section 4.1 presents an overview of the algorithm, while Section 4.2 describes the process of initializing the ad hoc agent's policy based on a small set of known types and obtaining the type repository. Section 4.3 elucidates how the ad hoc agent identifies teammate models during cooperation, and Section 4.4 delineates how the ad hoc agent adapts to the current teammate policy.

4.1. Overview of FEAT Algorithm

The overall framework of the algorithm is illustrated in Figure 1. Initially, we utilize the prior knowledge of a few known teammate types to derive a Meta-RL policy, which serves as the initial policy for the ad hoc agent. The objective is to enhance the efficiency of generating a diverse policy repository. Subsequently, the agent engages in self-play learning within a series of environments with perturbed reward functions. All policies obtained during this process are collected to form our policy repository, expanding the range of known teammate types for the ad hoc agent. Next, the types within the type repository are clustered based on their similarities, facilitating the identification of the most similar one to the current teammate among a large number of types. At the online stage, the ad hoc agent searches for the most similar type in the type repository based on the observations and actions of current teammates. If a similar type is identified, the most compatible policy from the repository will be directly employed for collaboration. Otherwise, the policy will be updated by imitating teammate behaviors to accomplish the current task within a single episode.

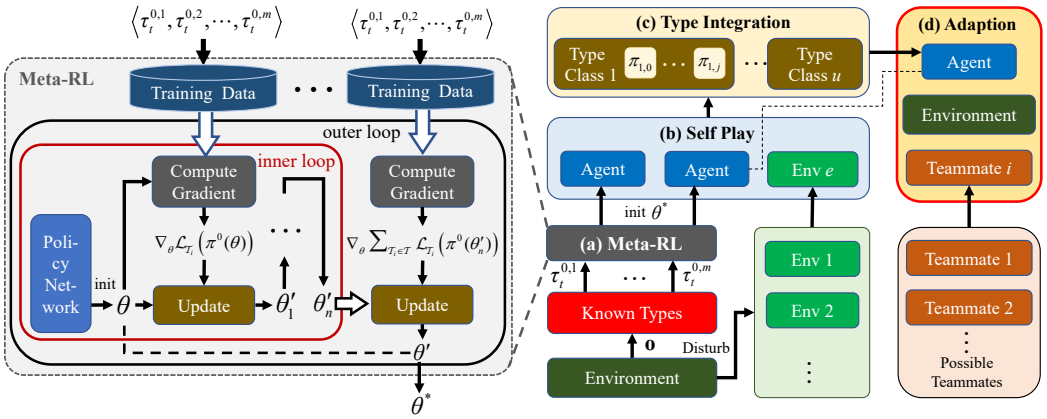


Figure 1. The framework diagram of the FEAT algorithm. (**a**) Acquisition of Meta-RL policy using a small number of teammate types. (**b**) Self-play generation of a diverse type repository by the Meta-RL policy in a perturbed environment. (**c**) Calculation of diversity metrics and cluster of the diverse type repository. (**d**) Online identification and adaptation of teammates during ad hoc agent cooperation.

4.2. Policy Repository Generation

The policies of known type repository Θ are denoted as $\Pi = \{\pi_1^{-0}, \cdots, \pi_m^{-0}\}$. The performance of different types varies, affecting the efficiency and behavior of the ad hoc agent when cooperating with them. Following the definition in Section 3.2, we have $\mathcal{T} = \{\mathcal{T}_1, \cdots, \mathcal{T}_m\}$, comprising a total of m cooperation source tasks, with the loss function defined across all tasks as follows:

$$\mathcal{L}_{\mathcal{T}}\left(\pi^0(\theta), \Pi\right) = \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}\left(\pi^0(\theta'), \pi_i^{-0}\right) = -\sum_{\mathcal{T}_i \sim p(\mathcal{T})} J\left(\pi^0(\theta'), \pi_i^{-0}\right). \tag{6}$$

Through stochastic gradient descent, the optimization of the above function yields the Meta-RL policy $\pi^0(\theta^*) = \arg\max \mathcal{L}_{\mathcal{T}}$ for the ad hoc agent, as detailed in Algorithm 1.

Algorithm 1 Obtaining Meta-RL policy

Input: $\mathcal{T}$: A few tasks of different known types
Input: N_{in}: The number of inner-loop iterations, N_{out}: The number of outer-loop iterations
Output: $\pi^0(\theta)$: The meta-RL policy of the ad hoc agent

```
Initialize ad hoc agent's policy π^0(θ)
for iteration n_o in N_out do
  for T_i in T do
    for iteration n_i in N_in do
      Use policy π^0(θ) to sample L trajectories τ_i in T_i
      Compute loss L_{T_i}(π^0(θ), π_i^{-0}) based on Equation (4)
      θ' ← θ − α∇_θ L_{T_i}(π^0(θ), π_i^{-0})
      Use policy π^0(θ') to sample L trajectories τ_i' in T_i
    end for
    Compute loss L_T(π^0(θ), Π) based on Equation (6)
    θ ← θ − β∇_θ L_T(π^0(θ), Π)
  end for
end for
```

In order to automatically generate a diverse type repository using the obtained Meta-RL policy, we introduce Gaussian noise $\epsilon \sim N(\mu, \sigma)$ to the environment reward function R, thereby inducing random perturbations and resulting in a series of perturbed reward functions $\tilde{R}$. Figure 2a illustrates the initial reward distribution in the environment, characterized by rewards of 1 at the four corners of the grid. Two essential properties must be preserved when perturbing the environment:

- **Consistency**: The agent's goals should remain within the original goals.
- **Reachability**: The agent should be able to achieve its goals.

To maintain consistency, a more significant value for the perturbation μ of rewards at the agent's original target states and a smaller value for σ are selected. For reachability, smaller values for both μ and σ are applied for perturbations at states other than the agent's original target states. Consequently, the perturbed reward function is defined as follows:

$$\tilde{R} = \begin{cases} R(s) + \epsilon_1, s \in \text{Goal} \\ R(s) + \epsilon_2, s \notin \text{Goal} \end{cases}. \tag{7}$$

Among these, we can take $\epsilon_1 \sim N(20, 5), \epsilon_2 \sim N(0, 1)$. The perturbation outcomes are illustrated in Figure 2b,c, where rewards at the four target states remain significantly higher than at other states, and the perturbed rewards are adjusted to prevent the agent from being trapped in local optima.

The obtained Meta-RL policy $\pi^0(\theta^*)$ then engages in self-play in a series of perturbed environments, with the loss function computed as follows:

$$\mathcal{L}_{SP}\left(\pi^0(\theta^*)\right) = -J\left(\pi^0(\theta^*), \pi^0(\theta^*)\right) = -\mathbb{E}_{\mathbf{o}_t, \mathbf{a}_t \sim \{\pi^0(\theta^*), \pi^0(\theta^*)\}}\left[\sum_{t=0}^{H} \gamma^t \tilde{R}_t(\mathbf{o}_t, \mathbf{a}_t)\right]. \quad (8)$$

+1	0	0	+1
0	0	0	0
0	0	0	0
+1	0	0	+1

(a)

+18.3	−0.94	0.71	+16.3
1.15	0.1	−0.05	−0.86
−1.88	0.09	1.09	1.07
+34.2	−0.90	0.15	+22.9

(b)

+29.7	0.55	0.18	+33.4
−1.51	0.43	0.17	−0.86
−0.46	−0.82	1.49	1.69
+20.9	−1.43	−1.41	+22.9

(c)

Figure 2. Visualization of the perturbed environment reward function. (**a**) Initial reward distribution in the environment, with four highlighted red areas representing the target regions for the agents to reach. (**b**) A situation of reward distribution after one perturbation. (**c**) Another situation of reward distribution after one perturbation.

After multiple training rounds, we generate a considerable number of effective policies, which augment the initial set of m policies to form the policy repository Π', also known as the type repository Θ'. Subsequently, to enhance the efficiency of identifying the teammate and obtaining the best-response cooperative policy, we evaluate the diverse policies within this repository from two perspectives $(\pi_i, \pi_j) \subseteq \Pi'$:

- **Similarity**. Since policies represent the probability distributions of different actions under specific observations, the comparison of the similarity between two policies involves the comparison of the similarity between two probability distributions. We chose the Jensen–Shannon (JS) divergence to measure the similarity between policies. Although JS divergence exhibits symmetry, it has a limitation in its inability to update gradients when two distributions do not completely overlap. This situation results in a constant divergence value regardless of the distance between their centers, leading to a gradient of zero and preventing gradient updates. However, this limitation does not influence our method, since we do not employ gradients. Therefore, the formula for computing the similarity of policy pairs is as follows, where $q = \frac{1}{2}(\pi_i + \pi_j)$:

$$M(\pi_i, \pi_j) := \frac{1}{2}\mathcal{D}_{KL}(\pi_i||q) + \frac{1}{2}\mathcal{D}_{KL}(q||\pi_j). \quad (9)$$

- **Compatibility**. This concept is derived from LIPO [28]. We extend its scope to measure the compatibility between pairs of policies, defining it in terms of task completion, task performance, and task completion time. The expressions are as follows:

$$C(\pi_i, \pi_j) := \frac{1}{K}\sum_{k=1}^{K} d(\pi_i, \pi_j)\left[\lambda_1 \sum_t R_t + \lambda_2(1 - N_{step}/N_{\max})\right]. \quad (10)$$

In this context, $d(\pi_i, \pi_j)$ measures the ability of the policy pair to complete the task, where $d = 1$ indicates successful completion and $d = 0$ denotes failure. $\sum_t R_t$ assesses the rewards gathered by the policy pair within a single episode, over a total of K episodes. The third term $(1 - N_{step}/N_{\max})$ measures the duration of task completion, where N_{step} signifies the number of steps required for task completion, and N_{max} represents the maximum number of steps permissible in the environment. In reinforcement learning training, a discount factor $\gamma \in (0, 1]$ is typically applied to the accumulated

rewards to regulate the agent's focus on short-term rewards. It also serves to minimize the number of steps required to complete the task during training; the more steps taken, the smaller the final returns $\gamma^t R_t$ accumulated. We aim to prevent the agent's returns from decreasing exponentially with an increase in steps; therefore, we compute the task completion reward and the completion steps independently, using λ_1 and λ_2 to manage their respective importance.

After calculating the similarity and compatibility for all policy pairs in Π', we obtain the similarity matrix **M** and the compatibility matrix **C**.

4.3. Teammate Identification

In Section 4.2, a diverse type repository has been provided for the ad hoc agent. If the ad hoc agent can identify the current teammate's policy based on limited observations and deduce the teammate's current actions, it can execute the task more effectively, as the Algorithm 2 shows.

Algorithm 2 Adapating to teammates

Input: Π': Automatically generated type repository
Input: $\pi^0(\theta^*)$: The Meta-RL policy
Initialize ad hoc agent policy $\pi^0(\phi) \leftarrow \pi^0(\theta^*)$
Initialize policy $\pi^0_* \leftarrow$ None
while not done **do**
  Get observation $\mathbf{o}_t^0$,$\mathbf{o}_t^{-0}$,$\mathbf{a}_{t-1}^{-0}$
  Add $\mathbf{o}_t^{-0}$, $\mathbf{a}_{t-1}^{-0}$ into the replay buffer $\mathcal{B}$
  bf_{inter}, $\mathbf{bf}_{inter} \leftarrow$ UpdateBelief(Π', $\mathcal{B}$)
  for Π'_i in Π' **do**
    if Π'_i satisfies the condition (14) with $\mathbf{bf}_{inter}$ **then**
      bf_{intra}, $\mathbf{bf}_{intra} \leftarrow$ UpdateBelief(Π'_i, $\mathcal{B}$)
      for $\pi'_{i,j}$ in Π'_i **do**
        if $\pi'_{i,j}$ satisfies the condition (14) with $\mathbf{bf}_{intra}$ **then**
          $\pi^0_* = \arg\max_{\pi^0 \in \Pi'} C(\pi^0, \pi'_{i,j})$
          $\pi^0(\phi) \leftarrow \pi^0_*$
        end if
      end for
    end if
  end for
  if π^0_* is None and len($\mathcal{B}$) $\leq N_B$ **then**
    Compute $\mathcal{L}_{IL}(\pi^0(\phi))$ based on Equation (16)
    Update $\phi \leftarrow \phi - \nabla \mathcal{L}_{IL}(\pi^0(\phi))$
  end if
end while
Function UpdateBelief (Π, $\mathcal{B}$):
  $n \leftarrow$ the size of Π
  Initialize belief distribution $bf \leftarrow [\frac{1}{n}, \cdots, \frac{1}{n}]_n$
  Initialize belief distribution list **bf**
  for Π_i in Π **do**
    for $(\mathbf{o}_{t-1}^{-0}, \mathbf{a}_{t-1}^{-0})$ in $\mathcal{B}$ **do**
      Update $P(\Pi_i)$ with $(\mathbf{o}_{t-1}^{-0}, \mathbf{a}_{t-1}^{-0})$ based on Equation (12)
      $bf[i] \leftarrow P(\Pi_i)$
      end for
  end for
  $bf = bf / \sum_i bf_i$
  Add bf in **bf**
return bf, **bf**

Before initiating collaboration with the teammate, we initialize the Meta-RL policy $\pi^0(\theta^*)$ (denoted as $\pi^0(\phi)$) as the initial policy for the ad hoc agent. We assume that the ad hoc agent receives the teammate's previous action $\mathbf{a}_{t-1}^{-0}$ and current observation $\mathbf{o}_t^{-0}$ at each time step and stores them in the replay buffer $\mathcal{B}$. Next, the process involves discerning among the policies in the policy repository based on the data in the replay buffer. However, as the types in the repository are automatically generated in an environment with random perturbations, resulting in a large number of generated types, we adopt the Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [29] algorithm to classify the type repository based on the similarity matrix $\mathbf{C}$, obtaining a total of u type clusters, the policy clusters of which are denoted as $\bigcup_{i=\{1,\cdots,u\}} \Pi_i' = \Pi'$.

For each sample pair $(\mathbf{o}_t^{-0}, \mathbf{a}_t^{-0}) \sim \mathcal{B}$ in the replay buffer, we compute the probability of each policy cluster $\Pi_i' = \{\pi_{i,1}, \cdots \pi_{i,l}\}$ in taking action $\mathbf{a}_t^{-0}$ under observation $\mathbf{o}_t^{-0}$ based on the policy repository.

$$P(\mathbf{a}_t^{-0}|\mathbf{o}_t^{-0}; \Pi_i') = \frac{1}{l}\sum_j \pi_{i,j}(\mathbf{a}_t^{-0}|\mathbf{o}_t^{-0}). \tag{11}$$

Using this likelihood, the posterior probability of the current teammate's policy in different policy classes is calculated as follows:

$$P\left(\Pi_i'|(\mathbf{o}_t^{-0}, \mathbf{a}_t^{-0})\right) = (1 - \eta \cdot \text{loss}) \cdot P\left(\Pi_i'|(\mathbf{o}_{t-1}^{-0}, \mathbf{a}_{t-1}^{-0})\right), \tag{12}$$

$$\text{wherer, loss } = 1 - P(\mathbf{a}_t^{-0}|\mathbf{o}_t^{-0}; \Pi_i'), \tag{13}$$

where $\eta = 0.5$, the prior probability $P\left(\Pi_i'|(\mathbf{o}_0^{-0}, \mathbf{a}_0^{-0})\right)$, determining the type class to which the teammate type belongs, is uniformly initialized. Following computation at each time step, the posterior probabilities of all type classes are normalized.

The teammate policy is considered to belong to policy class Π_i' at time step t when the following conditions are met. The first condition specifies that over h consecutive time steps, based on the current observation, the Π_i' obtained from Π' that is most similar to the current teammate policy remains consistent. The second condition indicates that over h successive time steps, the posterior probability of Π converges based on the current observation.

$$\begin{cases} \arg\max_{\Pi_i'} P\left(\Pi_i'|(\mathbf{o}_t^{-0}, \mathbf{a}_t^{-0})\right) = \cdots = \arg\max_{\Pi_i'} P\left(\Pi_i'|(\mathbf{o}_{t-h+1}^{-0}, \mathbf{a}_{t-h+1}^{-0})\right) \\ \sum_{j=1}^{h}\sum_{p=1}^{h} \left| P\left(\Pi_i'|(\mathbf{o}_{t-j}^{-0}, \mathbf{a}_{t-j}^{-0})\right) - P\left(\Pi_i'|(\mathbf{o}_{t-p}^{-0}, \mathbf{a}_{t-p}^{-0})\right) \right| \le \xi \end{cases}. \tag{14}$$

Subsequently, each policy within this policy class is subject to iterative updates of the previously mentioned posterior probability, utilizing observations and actions from the replay buffer. Ultimately, identify the policy $\pi_{i,j}^{-0} \in \Pi_i'$ that most closely resembles the teammate's policy within the policy repository. Following this, the ad hoc agent continues to explore the repository to identify the policy that is most compatible with the previously identified policy, serving as its current collaborative policy, denoted as $\pi_*^0 = \arg\max_{\pi^0 \in \Pi'} C(\pi^0, \pi_{i,j}^{-0})$.

4.4. Teammates' Policies Adaption

If the current teammate type is absent from the type repository, the ad hoc agent is unable to accurately determine the teammate type through updates of posterior probabilities. In the absence of knowledge regarding the teammate type, it is imprudent to blindly adopt behaviors. Additionally, in the context of a sparse-reward environment in which rewards are unattainable during task completion, the ad hoc agent is incapable of updating its policy using reinforcement learning methods within a single episode. Therefore, leveraging environmental characteristics is crucial to facilitate the updating of the policy for the ad hoc agent.

Nevertheless, it is noteworthy that should the environment be role-invariant, the ad hoc agent has the capability to update its policy using observations.

For an environment $\mathcal{E}$ comprising n agents, each with its own policy π^i, they together form a joint policy $\Pi_{\mathcal{E}} = \{\pi^1, \pi^2, \cdots, \pi^n\}$. The environment is deemed **role-invariant** provided there exists any permutation $pm : \{1, \cdots, n\} \to \{1, \cdots, n\}$ and its corresponding inverse permutation pm^{-1} such that $pm^{-1}(pm(\Pi_{\mathcal{E}})) = \Pi_{\mathcal{E}}$ and the following conditions are satisfied:

$$pm^{-1}(pm(\Pi_{\mathcal{E}})(\mathbf{o}_t)) = \Pi_{\mathcal{E}}(\mathbf{o}_t) = \{\pi^1(\mathbf{o}_t^1), \pi^2(\mathbf{o}_t^2), \cdots, \pi^n(\mathbf{o}_t^n)\}. \tag{15}$$

Considering an environment with two agents, at each time step, the ad hoc agent randomly selects b instances $(\mathbf{o}_i^i, \mathbf{a}_i^1)$ from the replay buffer. The loss function is computed as follows.

$$\mathcal{L}_{IL}\Big(\pi^0(\phi)\Big) = -\log \pi^0(\phi)(\mathbf{a}_i^1|\mathbf{o}_i^1). \tag{16}$$

Utilizing stochastic gradient descent, the loss function is updated to formulate a new policy for the ad hoc agent, facilitating improved cooperation with the teammate, as the Algorithm 2 shows. Furthermore, as the ad hoc agent is initialized using the Meta-RL policy, it displays high adaptability, leading to effective outcomes with limited imitation samples. This approach can also be applied to environments with multiple agents, in which a larger number of samples is added to the replay buffer at each time step, thereby providing more favorable conditions for training the ad hoc agent's policy.

5. Experiments

In this section, our experimental study is presented to demonstrate the effectiveness of FEAT. Section 5.1 outlines the experimental environment. Section 5.2 describes the primary settings and research objectives of the experiments. Section 5.3 presents the main experimental results. Section 5.4 showcases the ablation study results, primarily focusing on the design functionalities of various components within the algorithm.

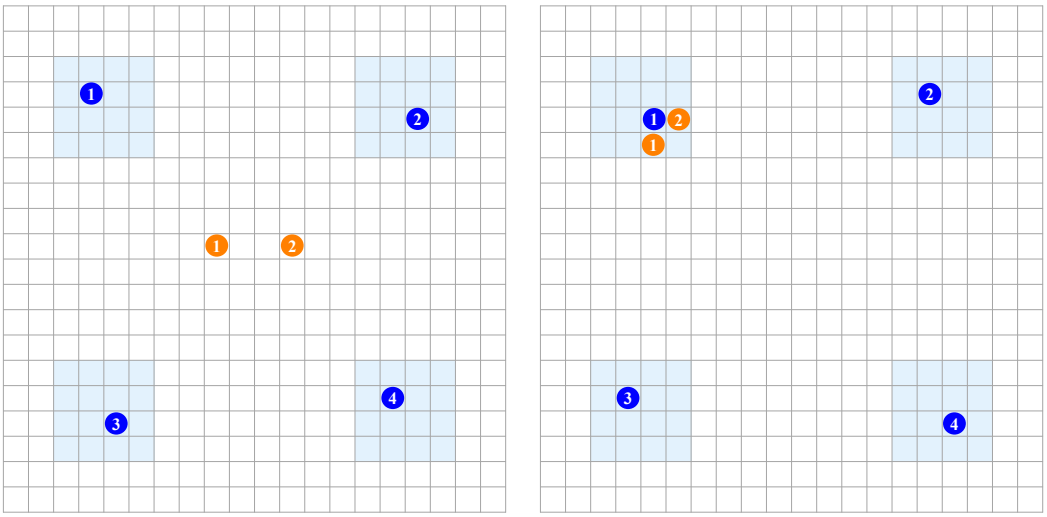
5.1. Domain Description

The experimental environment utilized in this study is identified as the Pursuit Domain, which is a setting where numerous studies about the AHT problem have been undertaken. A variant of the Pursuit Domain similar to that utilized by Xing et al. [12] is employed, with the distinction that in the present environment, a reward of 1 is awarded only when both predators capture the same prey simultaneously, contrasting with their setting, where a reward of 1 is given as soon as one predator captures the prey.

The experimental environment is depicted in Figure 3a. The map comprises a 20×20 square grid, representing a toroidal world. Four light blue regions are present on the map, each hosting one prey that is restricted to move randomly within the confines of its respective region. Two predators are randomly placed at the center of each region with the objective of simultaneously capturing one of the four prey. If a predator occupies one of the four adjacent squares around a prey, the prey is considered captured by that predator, as depicted in Figure 3b. Subsequently, the predator and prey maintain their positions for the next time step.

The objective of this experimental setting is for two predators to capture a prey simultaneously. A successful capture yields a reward of 1, whereas capturing different prey or exceeding the maximum game steps results in a reward of -1. For all other scenarios, no reward is awarded, defining it as a sparse-reward environment. The task terminates when one of the following conditions is met:

- Both predators simultaneously capture the same prey;
- Both predators capture different prey;
- The task execution steps exceed the maximum step length of the environment.



(a) (b)

Figure 3. Illustration of the pursuit domain environment. (**a**) The initial state of the environment features two orange predators randomly placed in the central area, and four blue prey randomly placed in the surrounding light blue areas. (**b**) An illustration of prey capture, depicting both predators simultaneously positioned within the neighboring grid to the prey.

Predators have five potential actions: moving up, down, left, right, or staying stationary. Movement order for the predators is determined randomly, and should an obstacle obstruct the intended destination, they will remain stationary. Observations encompass their own positions, their teammate's position, and the position of the nearest prey.

5.2. Experiment Setup

To evaluate the effectiveness of our method, we select PLASTIC-Policy [9], a highly representative method among type-based approaches within the PLASTIC algorithm. We excluded PLASTIC-Model, another variant within the PLASTIC algorithm, due to its requirement for an accurate model of the environment, which is unavailable in our experiments. Subsequently, we selected the EDRQN [12] algorithm, representing type-free approaches. In brief, the configuration for the compared algorithms is established as follows:

- **FEAT**: The algorithm proposed in this paper;
- **PLASTIC-**Θ: An implementation of a typical type-based algorithm PLASTIC-Policy under the setting of leveraging a constrained set of types Θ as the known teammate types, on the online testing phase;
- **PLASTIC-**Θ': An implementation of a typical type-based algorithm PLASTIC-Policy under the setting of leveraging the diverse type repository Θ' obtained in Section 4.2 as the known teammate types, on the online testing phase;
- **EDRQN**: A representative type-free algorithm that the ad hoc agent learns a policy demonstrating diverse behaviors without utilizing any knowledge about known types in the pre-training phase. Subsequently, during the online testing phase, the agent directly cooperates with its teammates;
- **Random**: A baseline wherein the ad hoc agent selects actions randomly at each time step, establishing the lower bound for experimental performance.

This study supplies sets of known teammate types numbered 2, 3, and 4, with their performances documented in Table 1. At the end of each episode, the teammates collaborating with the ad hoc agent are switched. Additionally, during the online testing phase, we established three configurations for teammate scenarios:

- **In**Θ: Possible teammate types are within Θ;
- **In**Θ': Possible teammate types are within Θ';

- **NFInΘ'**: Possible teammate policies are not fully within Θ'.

The results of the experiments are detailed in Section 5.3.

Table 1. Performance metrics for the known teammate types. The success rates of completing tasks, the IDs of prey captured, and the average number of steps required to complete tasks are not completely identical across different types. The final column specifies the category of the teammate type repository to which each teammate type is affiliated.

	Win Rate (%)	Caught Prey ID	Average Number of Steps	Affiliated Known Repository
Type 01	92.00	0	58.35 ± 43.46	$N = 2, 3, 4$
Type 02	100.00	3	37.67 ± 20.76	$N = 2, 3, 4$
Type 03	81.00	2	13.67 ± 2.91	$N = 3, 4$
Type 04	57.00	1	93.53 ± 63.53	$N = 4$

5.3. Main Results

Leveraging prior knowledge from a range of known types, a total of 35 types were generated. The quantity of these types within the constructed type repository is detailed in Table 2a. Within this table, Θ denotes the known types, Θ' the type repository generated by the algorithm, and Θ_T the externally supplied types of teammates required for collaboration. During the testing phase for various algorithms, combinations involving the known teammate type repository and the required types for collaboration are detailed in Table 2b. It is noteworthy that neither the EDRQN nor the random algorithms require known teammate types. Additionally, the approach for selecting teammates during testing aligns with that of the aforementioned algorithms.

Table 2. Explanation of type repositories varies in different experimental settings. Table (**a**) delineates the number of types provided and the respective quantities of types contained within three distinct type repositories. Table (**b**) depicts the combinations of testing type repositories provided to the ad hoc agent and potential teammate types during the online testing phase for three algorithms across various teammate configurations.

(**a**) Quantity of types			
	Θ	**Θ'**	**Θ_T**
$N = 2$	2	37	67
$N = 3$	3	38	67
$N = 4$	4	39	67
(**b**) Composition of type repository			
	InΘ	**InΘ'**	**NFInΘ'**
FEAT	Θ', Θ	Θ', Θ'	Θ', Θ_T
PLASTIC-Θ	Θ, Θ	Θ, Θ'	Θ, Θ_T
PLASTIC-Θ'	Θ', Θ	Θ', Θ'	Θ', Θ_T

In the testing phase, we implemented a sampling method without replacement to select a type of teammate from Θ_T for collaboration in each episode. Each algorithm and teammate executed ten collaborative interactions, except for EDRQN, which participated in fifty. Based on this methodology, we calculated the win ratios of the ad hoc agent and the average number of steps required to successfully complete their tasks. The detailed results are presented in Tables 3 and 4.

As demonstrated in Tables 3 and 4, the FEAT algorithm exhibits superior performance and robust stability across various experimental setups. It facilitates efficient teamwork even without constraining the actions of teammates to achieve optimality.

Table 3. The average win rates and corresponding standard deviations for various algorithms throughout the experiments.

	$N = 2$ (%)			$N = 3$ (%)			$N = 4$ (%)		
	InΘ	**InΘ′**	**NFInΘ′**	**InΘ**	**InΘ′**	**NFInΘ′**	**InΘ**	**InΘ′**	**NFInΘ′**
FEAT	**100 ± 0.0**	**95.95 ± 2.18**	**93.73 ± 1.86**	**100 ± 0.0**	**95.53 ± 1.69**	**94.33 ± 2.48**	**95.00 ± 10.00**	**95.90 ± 3.28**	**94.63 ± 2.02**
PLASTIC-Θ	**100 ± 0.0**	69.46 ± 5.55	60.20 ± 1.86	96.67 ± 10.0	69.47 ± 6.47	61.64 ± 5.86	77.50 ± 17.5	71.54 ± 7.90	68.21 ± 4.12
PLASTIC-Θ′	100 ± 0.00	93.24 ± 3.47	83.13 ± 4.12	93.33 ± 13.33	93.16 ± 4.28	81.49 ± 3.61	75 ± 11.18	90.77 ± 5.15	83.43 ± 6.35
EDRQN	64.00 ± 34.70	84.05 ± 5.83	85.55 ± 4.80	74.00 ± 25.20	85.95 ± 5.27	86.09 ± 4.11	67.00 ± 21.47	83.19 ± 6.40	85.58 ± 3.48
Random	35.00 ± 32.02	31.62 ± 7.46	34.03 ± 5.28	26.67 ± 38.87	25.79 ± 5.98	34.03 ± 5.29	32.50 ± 25.12	31.28 ± 8.33	34.03 ± 5.29

Table 4. The average step lengths and associated standard deviations for successful task completions by various algorithms.

	$N = 2$			$N = 3$			$N = 4$		
	InΘ	**InΘ′**	**NFInΘ′**	**InΘ**	**InΘ′**	**NFInΘ′**	**InΘ**	**InΘ′**	**NFInΘ′**
FEAT	**52.05 ± 27.23**	**28.27 ± 3.22**	**26.59 ± 4.10**	**49.57 ± 30.55**	**27.94 ± 3.29**	**26.08 ± 2.00**	46.48 ± 15.58	**31.08 ± 5.14**	**27.97 ± 2.89**
PLASTIC-Θ	72.6 ± 49.96	51.66 ± 9.86	42.36 ± 5.08	60.83 ± 49.49	47.12 ± 8.13	43.10 ± 4.60	**43.52 ± 17.80**	44.16 ± 9.15	43.36 ± 7.46
PLASTIC-Θ′	80.80 ± 35.99	36.59 ± 5.91	28.14 ± 3.25	50.58 ± 24.02	33.96 ± 3.93	30.15 ± 2.37	60.69 ± 26.87	36.16 ± 6.70	29.95 ± 4.11
EDRQN	103.36 ± 58.87	70.63 ± 9.63	69.71 ± 6.56	72.48 ± 39.55	68.96 ± 9.32	71.51 ± 5.71	92.29 ± 36.26	70.91 ± 9.99	69.55 ± 6.76
Random	151.50 ± 14.73	117.35 ± 14.73	107.10 ± 11.73	149.75 ± 53.69	114.63 ± 17.97	107.104 ± 11.73	115.38 ± 35.84	107.15 ± 22.09	107.10 ± 11.73

The PLASTIC algorithm exhibits high performance when the actual types of teammates fall within a known type repository, as demonstrated by PLASTIC-Θ (InΘ) and PLASTIC-Θ' (InΘ, InΘ'). However, as the diversity of cooperating teammate types increases, there is a tendency for the performance of the PLASTIC algorithm to decline. This improvement in algorithmic performance from PLASTIC-Θ to PLASTIC-Θ' underscores the benefits of automatically generating a diverse type repository.

The EDRQN algorithm displays a lack of stability, as evidenced by its performance in the test with N = 2(InΘ), where the standard deviation of its win rate reached 34.70%. The algorithm's performance deteriorates significantly when the collaborating agents' policies are suboptimal. For instance, in fifty collaborations with an agent whose average task completion steps amounted to 43.65, the win rate stood at merely 40%. However, as the diversity of tested teammate types increased, there was an improvement in the win rate. This improvement can be attributed to the increased presence of optimal policy-using teammates in the Θ_T, resulting in a higher win rate for the EDRQN agents. The findings indicate that the EDRQN algorithm tends to train policies that are responsive to optimal actions by teammates, resulting in poor performance when coupled with suboptimal or inferior teammates. Furthermore, the black-box form of the collaboration policies encountered by the ad hoc agents throughout the pre-training process complicates the comprehensive types of coverage assessment.

The random algorithm acts as the experimental performance baseline, where task completions occur coincidentally.

5.4. Ablation Results

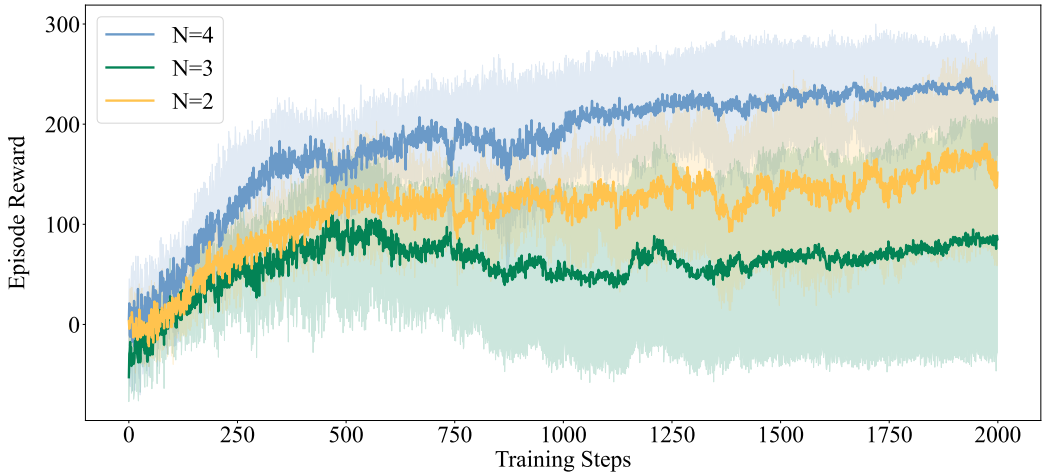
The ablation experiments in this section indicate that each module designed in our experiments contributes to the final experimental outcomes. The issues we aim to explore in the ablation experiments include the following:

1. What level of performance does the initially generated Meta-RL policy exhibit?
2. Does the type repository generated by the Meta-RL policy exhibit diversity and offer an advantage over self-play with the randomly initialized policy in terms of type generation?
3. What is the performance level of our method in the absence of automatic generation of a substantial type repository?
4. How significant is the contribution of the module that imitates the current behavior of teammates during an episode?

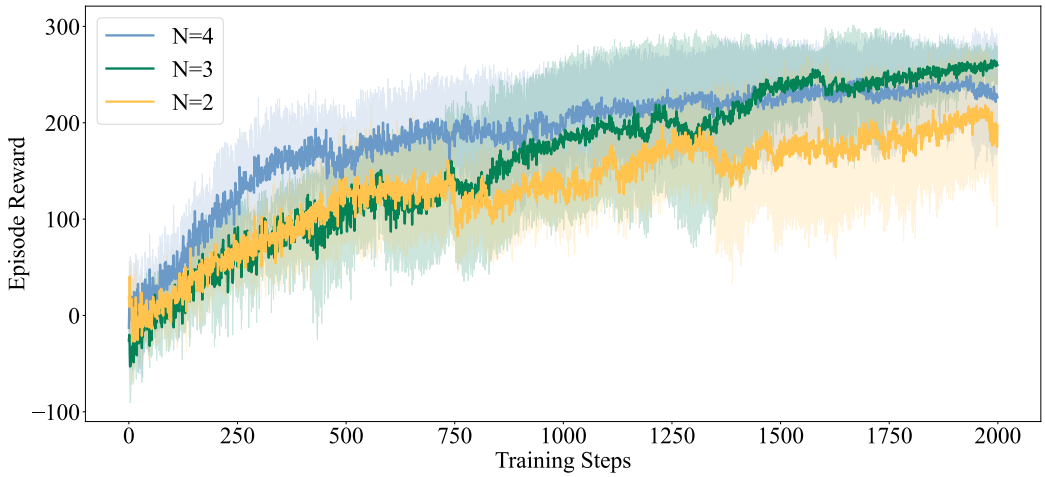
Initially, to better demonstrate the effectiveness of our experiment, we made tiny adjustments to the environment for this experiment. The environment does not terminate automatically when two predators capture different prey. For example, if both predators capture different prey at the 100th time step, a single predator would accumulate a reward of $\sum_t R_t = -201$. Likewise, if both predators capture different prey at the 50th time step, a single predator would accumulate a reward of $\sum_t R_t = 251$.

In question 1, we pre-trained the Meta-RL policy using prior knowledge about a predefined number of N types and evaluated them on a diverse group of 45 teammates with varying performance levels. The results are depicted in Figure 4. We set $N = 2$, $N = 3$, and $N = 4$, with teammate performance configurations matching those in Table 1.

As depicted in Figure 4, testing rewards steadily improve and converge as training steps increase. Obviously, there is an improvement in policy performance after a gradient descent. Figure 4b clearly indicates that the Meta-RL policy trained with $N = 4$ demonstrates superior performance compared to $N = 3$ and $N = 2$, with $N = 2$ showing the weakest performance. This suggests that the Meta-RL policy exhibits a parameter-sensitive characteristic, requiring minimal samples and iterations to collaborate effectively with unknown teammates. Furthermore, enhancing pre-training with a broader range of known teammate types can significantly enhance their performance in collaborating with unknown teammates.



(a) Playing with Testing Teammates



(b) Playing with Testing Teammates

Figure 4. The variation in average episode rewards with training steps during testing with ad hoc team members. (**a**) After each training step, we sampled three teammates and generated 20 trajectories for each teammate. The average reward of 60 trajectories was then plotted on the vertical axis as the average episode reward during testing. (**b**) After each training iteration, an evaluation of the current policy was conducted. Initially, a new teammate was randomly selected, and the current policy collaborated with this teammate to produce 20 trajectories. These trajectories were then utilized as samples to update the policy through a single round of gradient descent. Subsequently, the updated policy collaborated with the same teammate once more, generating an additional 20 trajectories. This procedure was repeated with three different teammates, resulting in a total of 60 trajectories, with the average trajectory rewards depicted on the y-axis.

In problem 2, using the policies obtained from $N = 4$, we acquired an additional 35 types to construct the type repository Θ'. We assessed the type repository using the similarity and compatibility metrics outlined in Section 4.2. The visualization of similarity and compatibility results is shown in Figure 5. Herein, the diagonal denotes the outcome of self-play within the same type, indicating the highest similarity and compatibility. The analysis reveals that the types exhibit a moderate level of similarity and substantial differences in compatibility. Furthermore, types with high compatibility tend to have greater similarity, as shown by $\mathbf{M}(\pi'_{30}, \pi'_{35})$ and $\mathbf{C}(\pi'_{30}, \pi'_{35})$. This demonstrates the diversity of the type repository.

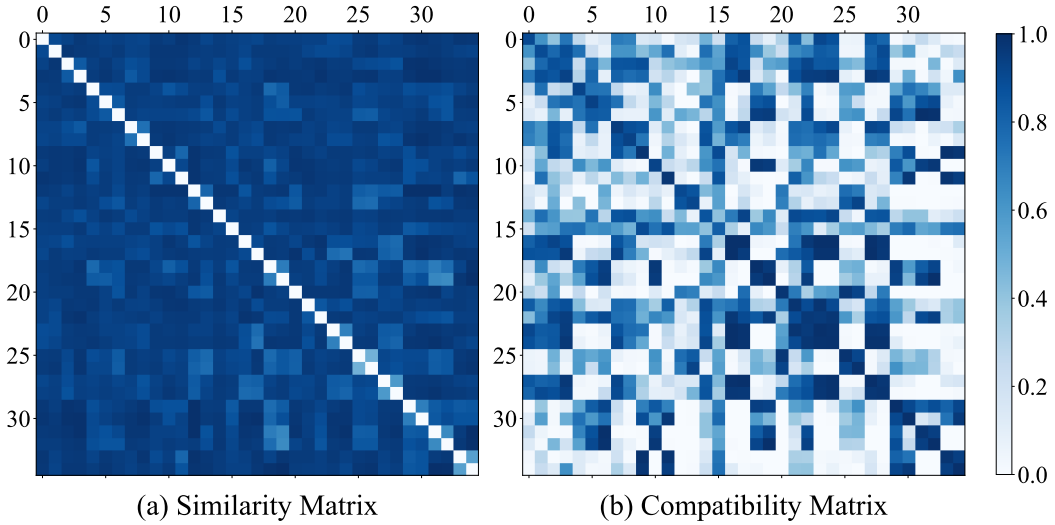


Figure 5. Description of the similarity and compatibility matrices of the type repository. (**a**) Each square depicts the similarity between corresponding policies of type pairs, as identified by the horizontal and vertical coordinates. (**b**) Each square depicts the compatibility between corresponding policies of type pairs, as determined by the horizontal and vertical coordinates.

Next, we contrasted the performance of self-play in the environment using policies initialized with the Meta-RL policies and randomly initialized policies using the PPO algorithm, as shown in Figure 6. The reward configuration corresponds to the environment before adjustments were made. The graph clearly shows that the Meta-RL policy requires fewer steps and achieves higher efficiency in obtaining a reliable policy.

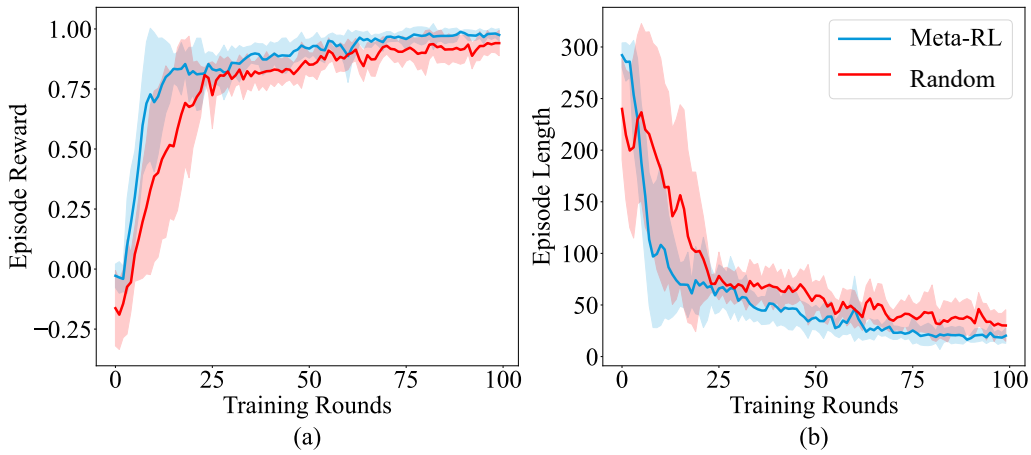


Figure 6. A comparative analysis of performance outcomes in self-play, which contrasts the effects of initializing with the Meta-RL policy that of random initialization. The horizontal axis denotes the number of training rounds. (**a**) This plots the average reward gained across 100 episodes following each training round. (**b**) This illustrates the average number of steps needed to complete a task across 100 episodes following each training round.

For problem 3, we did not utilize the automatically generated diverse type repository Θ' or utilize the Bayesian formula to identify the actual type of teammates during ad hoc collaboration. The performance of the algorithm is illustrated in Table 5. A total of 67 possible teammates are in Θ_T. Despite this, the algorithm achieves a high success rate and completes tasks in a relatively short number of steps. Moreover, as the number and diversity of known teammate types increase, the algorithm's performance gradually

improves. This improvement can be attributed to the enhanced generalization of the Meta-RL policy, enabling the algorithm to better adapt to unknown teammates.

Table 5. The efficacy of Meta-RL ad hoc agents, which are initialized with various known teammate types, in collaboration with teammates from Π_T without employing Bayesian inference or a diversity type repository.

	Win Rate (%)	Average Num of Steps
$N = 2$	74.33 ± 3.26	41.92 ± 4.80
$N = 3$	76.57 ± 1.77	50.75 ± 6.20
$N = 4$	83.28 ± 1.86	45.32 ± 5.08

Regarding problem 4, we initiated the ad hoc agent using known teammate types with $N = 3$. The agent collaborated with various teammates in the environment for ten episodes and we randomly sampled one of them, which had more than 50 time steps. Throughout this process, we did not utilize the teammate type identification module. Before each episode, the ad hoc agent's policy was reinitialized using the Meta-RL policy. We sampled 20 trajectories of the ad hoc agent collaborating with the current teammate at each time step and calculated the average reward. Figure 7 shows the results of the ad hoc agent collaborating with six different types of teammates, and Table 6 details the performance of each teammate.

Table 6. Diverse teammate performances are depicted in Figure 7, as well as their compatibility with the Meta-RL policy.

	Win Rate (%)	Caught Prey Id	Average Num of Steps	Compatibility
Figure 7a	93.00	0	97.68 ± 68.58	0.67
Figure 7b	97.00	1	66.30 ± 44.18	0.74
Figure 7c	98.00	2	11.95 ± 2.33	0.23
Figure 7d	97.00	3	93.53 ± 63.53	0.41
Figure 7e	95.00	3	25.00 ± 8.72	0.53
Figure 7f	100.00	3	11.99 ± 2.23	0.24

From Table 6, it is clear that the compatibility between the teammate depicted in Figure 7c and the current Meta-RL policy is minimal. Consequently, as depicted in Figure 7c, their average cooperation reward is approximately -100, which indicates a significant challenge in completing tasks. However, after imitation learning, their rewards experience a rapid and substantial improvement, highlighting the sensitivity of the Meta-RL policy to parameters and the significance of updating policies through imitation learning. In Figure 7b, teammates exhibit maximal compatibility with the Meta-RL policy, as evidenced by uniformly positive cooperation rewards and consistent task completion. The performance experienced a modest enhancement following imitation learning.

Further, we analyzed the impact of the cooperative teammates' performance on the joint outcomes, as depicted in Figure 7e,f. Notably, when teammates require fewer steps to complete tasks independently, as demonstrated in Figure 7f, there are fewer samples available for the ad hoc agent to learn from. Despite performance enhancements in the context of previously low compatibility, the limited number of training samples challenges the achievement of superior cooperative performance.

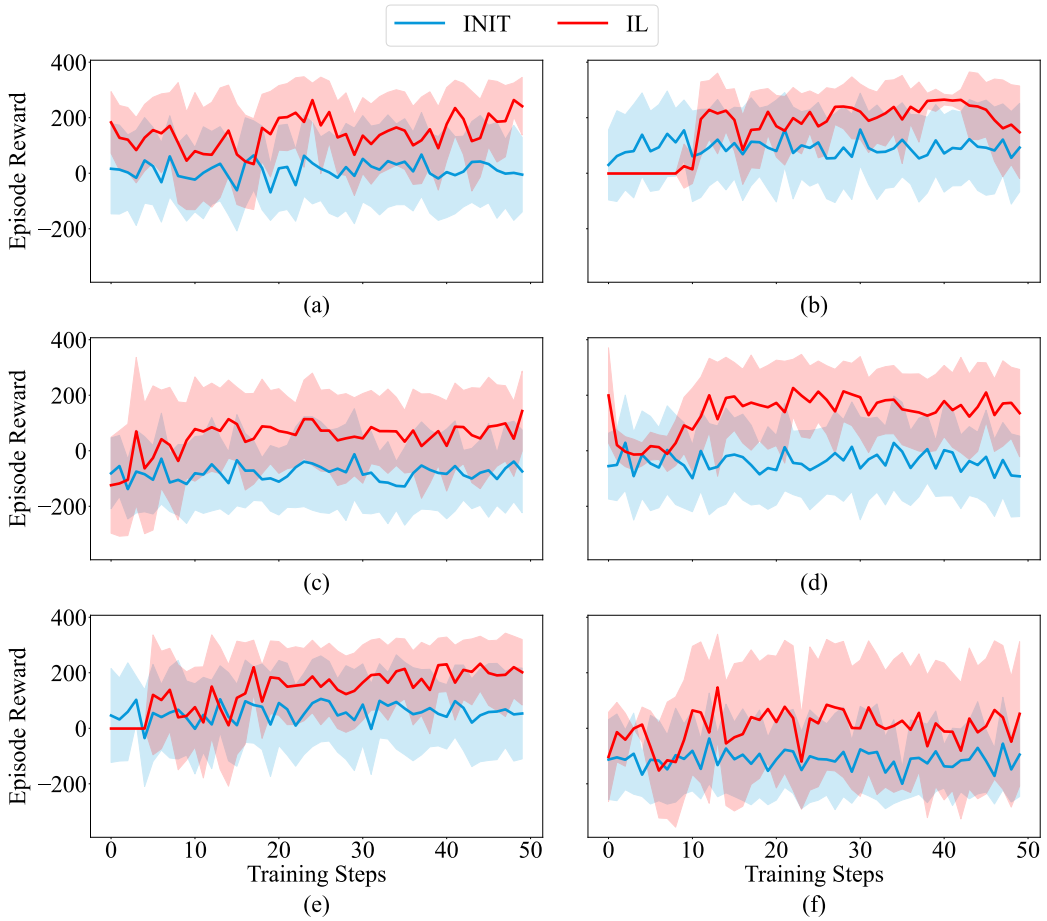


Figure 7. The variation in policy performance of ad hoc agents when cooperating with six distinct types of teammates, as shown in (**a**–**f**). The red line represents the performance of the ad hoc agent after updating its policy via imitation learning without inferring the teammate type.

6. Conclusions

In this paper, we introduced an algorithm termed FEAT for AHT, which is predicated on a small set of known teammate types. To optimally utilize prior knowledge of these types, FEAT employs meta-reinforcement learning algorithms to derive a Meta-RL policy that is highly responsive to new tasks. Subsequently, random perturbations were introduced to the environment's reward function, enabling the Meta-RL policy to participate in self-play within this perturbed setting and cultivate a diverse type repository. Based on this type repository, ad hoc agents initialized with the Meta-RL policy identify teammate policies while simultaneously imitating teammate behaviors to achieve effective cooperation. Comprehensive experimental analyses in the pursuit domain validated the effectiveness of the individual module designs within the FEAT algorithm, which outperforms representative methods such as PLASTIC [9] and EDRQN [12].

Our approach represents the first explicit algorithm designed for AHT, which focuses on a limited number of known types. The algorithm shows its capability to adapt to teammates within a single episode in partially observable and sparse-reward environments. However, prior related studies usually needed multiple episodes for successful collaboration with teammates. Moreover, based on the foundational assumptions of our algorithm, we suggest several promising directions for future research. Firstly, we assume a role-invariant experimental environment, which allows us to utilize teammate behavior imitation for adaptation. To address environments with variable roles, integrating opponent and environmental modeling with the Meta-RL policy seems to provide a feasible solution. Secondly, we assume that ad hoc agents have access to teammate observations and actions. To relax this assumption, we recommend exploring the concept of ODITS [30]. Finally,

despite creating a diverse type repository, the main purpose of it within the algorithm is to identify teammate policies. To further enhance this capability, we propose integrating transfer learning to infer teammate policies using the type repository. It is important to mention that our experimental setup, which involves only two agents, can be adapted to environments with different numbers of agents without impacting the performance of the algorithm.

Author Contributions: Conceptualization, Q.F. and J.Z.; methodology, Q.F. and H.X.; validation, Q.F. and H.X.; formal analysis, Q.F.; investigation, Q.F. and J.Z.; data curation, Q.F.; writing—original draft preparation, Q.F.; writing—review and editing, Q.F., J.Z. and Y.H.; visualization, J.Z.; supervision, Y.H., J.Z. and Q.Y.; funding acquisition, Y.H. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Natural Science Foundation of China (grant number 62306329, 62103420) and the Natural Science Foundation of Hunan Province of China (grant number 2023JJ40676).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The code of EDRQN can be found at https://github.com/dxing-cs/EDRQN, accessed on 23 October 2023. The source code of PLASTIC can be found at https://github.com/jmribeiro/PLASTIC-Algorithms, accessed on 12 October 2023. And the pursuit environment is contained at https://github.com/dxing-cs/EDRQN, accessed on 23 October 2023.

Acknowledgments: We thank Xing et. al. and João Ribeiro for the opensource code.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Rashid, T.; Samvelyan, M.; De Witt, C.S.; Farquhar, G.; Foerster, J.; Whiteson, S. Monotonic value function factorisation for deep multi-agent reinforcement learning. *J. Mach. Learn. Res.* **2020**, *21*, 1–51.
2. Wjkde, H.; Son, K.; Kim, D.; Qtran, Y. Learning to factorize with transformation for cooperative multi-agent reinforcement learning. In Proceedings of the 31st International Conference on Machine Learning, Proceedings of Machine Learning Research, PMLR, Long Beach, CA, USA, 9–15 June 2019.
3. Lowe, R.; Wu, Y.; Tamar, A.; Harb, J.; Pieter Abbeel, O.; Mordatch, I. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. In Proceedings of the Advances in Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R., Eds.; Curran Associates, Inc.: New York, NY, USA, 2017; Volume 30.
4. Stone, P.; Kaminka, G.; Kraus, S.; Rosenschein, J. Ad hoc autonomous agent teams: Collaboration without pre-coordination. In Proceedings of the AAAI Conference on Artificial Intelligence, Atlanta, GA, USA, 11–15 July 2010, Volume 24; pp. 1504–1509.
5. Bowling, M.H.; Browning, B.; Veloso, M.M. Plays as Effective Multiagent Plans Enabling Opponent-Adaptive Play Selection. In Proceedings of the ICAPS, Whistler, BC, Canada, 3–7 June 2004; pp. 376–383.
6. Bowling, M.; McCracken, P. Coordination and adaptation in impromptu teams. In Proceedings of the 20th National Conference on Artificial Intelligence, AAAI'05, Pittsburgh, PA, USA, 9–13 July 2005; AAAI Press: Washington, DC, USA, 2005; Volume 1, pp. 53–58.
7. Albrecht, S.V.; Ramamoorthy, S. A game-theoretic model and best-response learning method for ad hoc coordination in multiagent systems. In Proceedings of the 2013 International Conference on Autonomous Agents and Multi-Agent Systems, AAMAS '13, Richland, WA, USA, 6–10 May 2013; pp. 1155–1156.
8. Albrecht, S.V.; Crandall, J.W.; Ramamoorthy, S. E-HBA: Using action policies for expert advice and agent typification. In Proceedings of the Workshops at the Twenty-Ninth AAAI Conference on Artificial Intelligence, Austin, TX, USA, 25–26 January 2015.
9. Barrett, S.; Rosenfeld, A.; Kraus, S.; Stone, P. Making friends on the fly: Cooperating with new teammates. *Artif. Intell.* **2017**, *242*, 132–171. [CrossRef]
10. Ribeiro, J.a.G.; Rodrigues, G.; Sardinha, A.; Melo, F.S. TEAMSTER: Model-based reinforcement learning for ad hoc teamwork. *Artif. Intell.* **2023**, *324*, 104013–104038. [CrossRef]
11. Wu, F.; Zilberstein, S.; Chen, X. Online Planning for Ad Hoc Autonomous Agent Teams. In Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI), Barcelona, Spain, 16–22 July 2011; pp. 439–445. [CrossRef]

12. Xing, D.; Liu, Q.; Zheng, Q.; Pan, G. Learning with Generated Teammates to Achieve Type-Free Ad-Hoc Teamwork. In Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21, Montreal, QC, Canada, 19–27 August 2021; Zhou, Z.H., Ed.; Main Track; International Joint Conferences on Artificial Intelligence Organization: San Francisco, CA, USA, 2021; pp. 472–478. [CrossRef]
13. Rahman, A.; Fosong, E.; Carlucho, I.; Albrecht, S.V. Generating Teammates for Training Robust Ad Hoc Teamwork Agents via Best-Response Diversity. *Trans. Mach. Learn. Res.* **2023** . [CrossRef]
14. Rahman, A.; Cui, J.; Stone, P. Minimum coverage sets for training robust ad hoc teamwork agents. *arXiv* **2023**, arXiv:2308.09595.
15. Canaan, R.; Gao, X.; Togelius, J.; Nealen, A.; Menzel, S. Generating and Adapting to Diverse Ad Hoc Partners in Hanabi. *IEEE Trans. Games* **2023**, *15*, 228–241. [CrossRef]
16. Barrett, S.; Stone, P.; Kraus, S.; Rosenfeld, A. Learning Teammate Models for Ad Hoc Teamwork. In Proceedings of the AAMAS Adaptive Learning Agents (ALA) Workshop, Valencia, Spain, 4–9 June 2012.
17. Pardoe, D.; Stone, P. Boosting for regression transfer. In Proceedings of the 27th International Conference on International Conference on Machine Learning, ICML'10, Madison, WI, USA, 21–24 June 2010; pp. 863–870.
18. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G.; et al. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533. [CrossRef]
19. Xing, D.; Gu, P.; Zheng, Q.; Wang, X.; Liu, S.; Zheng, L.; An, B.; Pan, G. Controlling Type Confounding in Ad Hoc Teamwork with Instance-wise Teammate Feedback Rectification. In Proceedings of the 40th International Conference on Machine Learning, PMLR, 23–29 July 2023; Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., Scarlett, J., Eds.; The International Machine Learning Society: Stroudsburg, PA, USA, 2023 ; Volume 202, pp. 38272–38285.
20. Chen, S.; Andrejczuk, E.; Cao, Z.; Zhang, J. AATEAM: Achieving the Ad Hoc Teamwork by Employing the Attention Mechanism. *Proc. Aaai Conf. Artif. Intell.* **2020**, *34*, 7095–7102. [CrossRef]
21. Barrett, S.; Stone, P. Cooperating with Unknown Teammates in Complex Domains: A Robot Soccer Case Study of Ad Hoc Teamwork. *Proc. Aaai Conf. Artif. Intell.* **2015**, *29*, 2010–2016. [CrossRef]
22. Barrett, S.; Stone, P.; Kraus, S. Empirical evaluation of ad hoc teamwork in the pursuit domain. In Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems, AAMAS '11, Richland, WA, USA, 2–6 May 2011; Volume 2, pp. 567–574.
23. Kocsis, L.; Szepesvári, C. Bandit based monte-carlo planning. In *Lecture Notes in Computer Science, Proceedings of the European Conference on Machine Learning, Berlin, Germany, 18–22 September 2006*; Springer: Berlin/Heidelberg, Germany, 2006; pp. 282–293.
24. Ernst, D.; Geurts, P.; Wehenkel, L. Tree-based batch mode reinforcement learning. *J. Mach. Learn. Res.* **2005**, *6*, 503–556.
25. Charakorn, R.; Manoonpong, P.; Dilokthanakul, N. Learning to Cooperate with Unseen Agents Through Meta-Reinforcement Learning. In Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS '21, Richland, WA, USA, 3–7 May 2021; pp. 1478–1479.
26. Finn, C.; Abbeel, P.; Levine, S. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. In Proceedings of the 34th International Conference on Machine Learning, PMLR, Sydney, Australia, 6–11 August 2017; Precup, D., Teh, Y.W., Eds.; The International Machine Learning Society: Stroudsburg, PA, USA, 2017 ; Volume 70, pp. 1126–1135.
27. Bernstein, D.S.; Zilberstein, S.; Immerman, N. The complexity of decentralized control of Markov decision processes. In Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence, UAI'00, San Francisco, CA, USA, 30 June–3 July 2000; pp. 32–37.
28. Charakorn, R.; Manoonpong, P.; Dilokthanakul, N. Generating Diverse Cooperative Agents by Learning Incompatible Policies. In Proceedings of the Eleventh International Conference on Learning Representations, Kigali, Rwanda, 1–5 May 2023.
29. Ester, M.; Kriegel, H.P.; Sander, J.; Xu, X. A density-based algorithm for discovering clusters in large spatial databases with noise. *KDD* **1996**, *96*, 226–231.
30. Gu, P.; Zhao, M.; Hao, J.; An, B. Online Ad Hoc Teamwork under Partial Observability. In Proceedings of the International Conference on Learning Representations, Virtual, 25–29 April 2022.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.