Collaboration with Dynamic Open Ad Hoc Team via Team State Modelling

Anonymous authors
Paper under double-blind review

Abstract

Open ad hoc teamwork presents the challenging problem of designing an autonomous agent that can rapidly adapt to collaborate with teammates without prior coordination in an open environment. Existing methods primarily rely on fixed, predefined teammate types, overlooking the fact that teammates may change dynamically. To address this limitation, we propose a novel reinforcement learning approach, the Open Online Teammate Adaptation Framework (Open-OTAF), which enables a controlled agent to collaborate with dynamic teammates in open ad hoc environments. To achieve this, the controlled agent employs a dual teamwork situation inference model to capture the current teamwork state, facilitating decision-making under partial observability. To handle the dynamic nature of teammate types, we first introduce a Chinese Restaurant Process-based model to categorize diverse teammate policies into distinct clusters, improving the efficiency of identifying teamwork situations. Next, to model heterogeneous agent relationships and accommodate a variable number of teammates, we represent the team as a heterogeneous graph and leverage heterogeneous graph attention neural networks to learn the representation of the teamwork situation. Extensive experiments across four challenging multi-agent benchmark tasks—Level-Based Foraging, Wolf-Pack, Cooperative Navigation, and FortAttack—demonstrate that our method successfully enables dynamic teamwork in open ad hoc settings. Open-OTAF outperforms state-of-the-art methods, achieving superior performance with faster convergence.

1 Introduction

Recently, there has been increasing interest in applying multi-agent reinforcement learning (MARL) to complex multi-agent tasks, such as navigating human-shared environments and real-time strategy games Roesch et al. (2020); Nguyen et al. (2020); Boldrer et al. (2022); Zhou et al. (2021); Zhang et al. (2018). The centralized training with decentralized execution (CTDE) framework has gained significant attention, enabling agents to train with global information in a centralized manner while making decisions based only on local observations during execution in a decentralized manner. Numerous MARL methods have been developed within the CTDE framework, primarily categorized into policy-based and value-based approaches Lowe et al. (2017); Rashid et al. (2018); Sunehag et al. (2017); Yu et al. (2022). Policy-based methods, such as MADDPG Lowe et al. (2017), COMA Foerster et al. (2018), and MAPPO Yu et al. (2022), leverage observations and actions of controlled agents to train policy networks. In contrast, value-based methods, exemplified by VDN Sunehag et al. (2017) and QMIX Rashid et al. (2018), construct a joint value function to optimize team performance. However, these approaches assume a fixed team configuration, where factors such as team size, formation, and goals remain unchanged, and teammate types are known and static.

In many real-world applications, agents must collaborate with unknown and diverse teammates in real-time, giving rise to the ad hoc teamwork (AHT) problem Stone et al. (2010). Existing research on AHT often assumes that all teammates' behaviours are predefined out of types, each corresponding to a specific coordination strategy Albrecht & Stone (2018); Barrett et al. (2017). However, in dynamic environments, teammates' strategies may evolve unpredictably, leading to severe miscoordination if assumed to be static and fixed Chen et al. (2020). A common approach to adapt to dynamic teammates is to leverage Bayesian posteriors of teammate types and incorporate them into reinforcement learning when learning policies Barrett et al. (2017); Ravula (2019); Chen et al. (2020). However, these methods require access to teammates' observations and actions to compute the posteriors, which is infeasible in partially observable environments. Recent efforts have attempted to address partial observability, but they primarily focus on closed environments where the number of agents remains fixed Gu et al. (2021); Ribeiro et al. (2022); Zhang et al. (2023); Rahman et al. (2024); Li et al. (2024); Chen et al. (2025). In contrast, many real-world scenarios require adaptation to a dynamically changing number of agents. For example, an autonomous vehicle must adjust its driving behaviour based on the number of surrounding vehicles, which could be driven by humans or produced by different manufacturers, each with distinct driving styles Barrett & Stone (2015); Albrecht et al. (2021). Therefore, achieving online adaptation with dynamic teammates in open AHT environments remains an open challenge.

This work aims to achieve online adaptation in open ad hoc teams with varying numbers of teammates in the partially observable environments. To this end, we present a novel reinforcement learning framework, the Open Online Teammate Adaptation Framework (Open-OTAF). The core idea of Open-OTAF is that teamwork performance is influenced not only by an autonomous agent's behaviour but also by the evolving dynamics of its teammates, which can be modeled as the online teamwork situation. By accurately identifying the current teamwork situation, the agent can select actions that enhance coordination, ultimately leading to improved teamwork performance.

In Open-OTAF, the ad hoc agent maintains a dual-structured model to learn the current teamwork situation. We represent the teamwork situation as a fixed-size learnable embedding. Since similar teammates often share common characteristics, independently learning a separated context for each teammate while ignoring their interrelationships can lead to redundant encodings. To address this, we employ a Chinese Restaurant Process (CRP) Blei & Frazier (2011) to cluster teammates based on their behavioural similarities. This clustering reduces the complexity of the context search space and enhances the efficiency of representation learning. The CRP takes teammate behaviours as input and outputs a probability distribution over teammate types, which serves as the ad hoc belief about the overall team composition. Furthermore, adapting to open environments requires the agent to handle changing team sizes and dynamic relationships with other agents, as these factors influence the agent's policy and role within the team. To model heterogeneous relationships among teammates while accommodating a variable team size, we represent the team as a heterogeneous graph and leverage a Heterogeneous Graph Attention Network (HAN) to compute the teamwork situation embedding. For coordinated policy training, we sample representative teammates and encode their identities into distinct contexts to enhance joint policy learning during centralized training. During decentralized execution, each agent approximates the global context using only its local observations. This process alternates iteratively, yielding a robust policy enabling adaptation to dynamic teammates. To summarize, we highlight the following contributions:

- To the best of our knowledge, this is the first work that considers the online adaptation with varying number of heterogeneous teammates in open ad hoc teamwork;
- We propose accommodating the heterogeneity of teammates in an open AHT setting and introducing the heterogeneous graph attention network to learn teamwork situations to facilitate teamwork effectively.
- We present a formal derivation, grounded in variational inference, that makes the local teamwork situation embedding informatively consistent with global context, bolstering the theoretical foundation of our Open-OTAF designs.
- We show that the proposed method achieves superior performance on a range of challenging multi-agent tasks, including the Level Based Foraging (LBF), Wolfpack, Penalized Cooperative Navigation (PCN), and the Fortattack Rahman et al. (2022). The extensive experiments confirm that Open-OTAF achieves dynamic teamwork for open ad hoc teams and delivers superior performance against state-of-the-art methods with a faster convergence speed.

The remainder of this paper is organized as follows. Section 2 provides a briefly review of the works in MARL, ad hoc teamwork, agent modelling, and graph neural networks. Section 3 introduces the preliminaries of the proposed method. Section 4 elaborates on the detailed designs of the Open-OTAF framework. Section 5 provides the evaluation experiments and ablation studies. Finally, Section 6 concludes the paper and presents future works.

2 Related work

Cooperative multi-agent reinforcement learning. Recently, the developments in MARL have led to remarkable progress in creating intelligent agents that can efficiently cooperate to solve complex tasks Hernandez-Leal et al. (2019); Zhang et al. (2021); Gronauer & Diepold (2022). MARL algorithms use RL techniques to train a fleet of agents in a multi-agent system. There are two main frameworks of MARL methods: centralized and decentralized learning. Centralized methods Claus & Boutilier (1998) learn a single policy to predict the joint actions of all agents directly, while decentralized learning Littman (1994) entails each agent independently. The framework of centralized training and decentralized execution (CTDE) bridges the gap between the two, which permits information sharing during training, while policies are only consuming the agents' local observations, enabling decentralized execution Lowe et al. (2017). The policy gradient methods are a sub-class of CTDE, wherein each agent consists of a decentralized actor and a centralized critic, which is jointly optimized with the shared information of the controlled agents Lowe et al. (2017); Foerster et al. (2018). The value decomposition methods represent the joint Q-function as a function of agents' local Q-functions Sunehag et al. (2017); Rashid et al. (2018); Son et al. (2019); Wang et al. (2020). However, these studies presume the team configuration (e.g., team size, teammate types, team formation, and goals) stays unchanged.

Ad hoc teamwork. In many real-world settings, agents are required to collaborate in diverse teams without previous joint training, which comes up with ad hoc teamwork (AHT) problems. There are three main assumptions that characterise AHT Mirsky et al. (2022). The primary assumption is the lack of prior coordination between agents, which means that the learner should be able to cooperate with the team on-the-fly, without the opportunity to rely on previously agreed collaboration strategies. The second assumption is that the learner lacks control over its teammates. Lastly, teammates are assumed to be collaborative, indicating that all agents in the team have a common goal and can take actions that will benefit the team. However, teammates might have additional objectives that may vary per teammate and even have different rewards. Early works in ad hoc teamwork operated under the assumption that the teammate's behaviour was known to the learning agent Stone & Kraus (2010); Agmon & Stone (2012). Other approaches have relaxed this assumption and assumed teammate behaviour to be unknown. These works Barrett et al. (2017) proposed methods that infer teammates' policy based on their displayed behaviour and utilize it for decision-making. Such methods typically utilise the concept of types, which encapsulates the important information determining an agent's behaviour. Existing type-based methods assume that the number of teammates is fixed and their behaviours are categorized into several known types Barrett & Stone (2015); Mirsky et al. (2020); Albrecht & Stone (2019); Chen et al. (2020); Ravula (2019). However, it is quite often that teammates with unknown types may enter and leave the environment without prior notification. The drawback of these approaches is that finite and fixed types might not cover all possible situations in complex environments Chandrasekaran et al. (2016); Rahman et al. (2021; 2022). Another strong assumption in these works is that the agent could access other teammates' observations and actions, which is unrealistic in partially observable environments. Though some previous arts have directed their attention toward addressing ad hoc teamwork under partial observability, they only considered closed environments in which the number of agents is fixed Gu et al. (2021); Ribeiro et al. (2022); Schäfer et al. (2022); Zhang et al. (2023); Rahman et al. (2023). However, it is quite often that agents may join and leave the team anytime, making learning in such open AHT environments still an open research challenge. Existing approaches neglect the heterogeneity (i.e., the varying types) among teammates and the varying team size. We first use HAN to tackle the potentially infinite team composition and generate better semantic embedding to fill this gap.

Agent modellingg. Agent modelling is a research topic that emerged alongside game theory Brown (1951). With the powerful representation capabilities of recent deep neural networks, the controlled agents can reason about other agents' unknown goals and behaviours Albrecht & Stone (2018); He et al. (2016); Albrecht & Stone (2019). For example, the recursive reasoning method, MeLIBA, conditions the ad hoc agent's policy on a belief over teammates, which is updated following the Bayesian rule Zintgraf et al. (2021). While these works assume that the teammates choose their actions independently. As a result, the Relational Forward Model (RFM) architecture was proposed, which can be used to model teammates' potential effects on each other Tacchetti et al. (2019). However, the aforementioned works commonly assume access to other agents' information. Recent work proposes to use VAE to learn teammate information under partial observability teamwork Gu et al. (2021). Inspired by these works, the agent modelling frameworks are introduced to predict the teammate information for open ad hoc teamwork. Motivated by this, we introduce a marginal utility network, that integrates joint action value estimates with action predictions learned using an agent model, to select optimal actions.

Graph neural network. Graph neural networks (GNNs) are a type of powerful neural architecture that can work with graph-structured data Wu et al. (2020), which have been exploited to learn factored action values to simplify the computation of optimal joint actions using coordination graphs (CGs) Böhmer et al. (2020); Naderializadeh et al. (2020). Unlike these methods that use CGs to model joint action values for fully cooperative setups, some works use them in open systems to model the impact of other agents' actions towards the learning agent's returns Rahman et al. (2021); Jiang et al. (2019) to handle dynamic input sizes. However, these works have overlooked the different relationships and importance among the agents introduced by the open environment. To solve this issue, we first introduce the Heterogeneous Graph Attention Network (HAN) to compute the ultimate teamwork situation embedding to facilitate decision-making by considering the potential effect of other agents. Together with previous works, we show the great potential of HAN, which could inspire further research.

3 Problem fromulation

We aim to train a single autonomous agent, the ad hoc agent, that can cooperate with various teammates without pre-coordination in a partially observable open environment. The problem can be formulated by extending the framework of partially observable MDP (POMDP) Lowe et al. (2017) to an *Open POMDP*, which is defined by a tuple $\mathcal{M} = \langle \mathcal{N}, \mathcal{S}, \mathbb{O}, \mathcal{O}, \mathcal{A}, \mathcal{T}, \Gamma, \mathcal{R}, \gamma \rangle$. Specifically, $\mathcal{N}$ denotes the dynamic set of agents involved in the task. $\mathcal{S}$ is a set of states describing the possible configuration of all agents and the external environment, where the number and type of agents vary. Each agent k has its own observation space $\mathbb{O}_k \in \mathbb{O}$ with a corresponding observation function $\mathcal{O}_k : \mathcal{S} \to \mathbb{O}_k$ where $\mathcal{O}_k \in \mathcal{O}$. The agent types can be determined by their policies, which belong to a global policy set Γ.

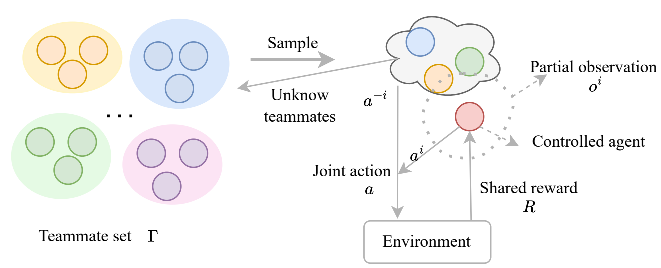


Figure 1: Visualization of the Dec-POMDP with teammate set.

Figure 1 shows the detailed schematics of this problem. We denote the policy of the ad hoc agent as π^i and the joint policy of all other agents as $\boldsymbol{\pi}^{-i}$. Each agent k selects an action $a_k \in A_k$ from its own action space $A_k \in \mathcal{A}$, giving rise to a joint action $[a_1, \cdots, a_n] \in A_1 \times A_2 \times \cdots \times A_n$ with n being the variable number of agent. The state transits by following a transition function $\mathcal{T} : \mathcal{S} \times A_1 \times \cdots \times A_n \to \mathcal{S}$. After each transition, agent k receives a new observation and obtains a scalar reward according to its reward function $r_k \in \mathcal{R} : \mathcal{S} \times A_k \to \mathbb{R}$. The initial state $s \in \mathcal{S}$ is determined by some prior distribution $p : \mathcal{S} \to [0, 1]$. The ad hoc agent's optimal policy π_i^* is defined by maximizing its own total expected return $R_i = \mathbb{E}_{r^t \sim r_i(s_t, a_t^i, \boldsymbol{a}_t^{-i}), (s_t, a_i^t, \boldsymbol{a}_t^{-i}) \sim \tau} [\sum_{t=0}^{T} \gamma^t r^t]$, where γ is the discount factor, $a_t^i \sim \pi^i$ and $\boldsymbol{a}_t^{-i} \sim \boldsymbol{\pi}^{-i}$.

Marginal Utility is defined to measure the contribution of an ad hoc agent to the whole team utility Genter et al. (2011). It represents the increase (or decrease) in a team's utility when an ad hoc agent is added to the team. Given teammates' actions $\boldsymbol{a}^{-i}$, there is a relationship between the marginal utility and the team utility (denoted by the joint action value) as $\arg\max_{a^i} u^i(s, a^i, \boldsymbol{a}^{-i}) = \arg\max_{a^i} Q^{\pi_i}(s, a^i, \boldsymbol{a}^{-i})$,

Table 1: Notations of algorithm

Notation	Description	Notation	Description
t	Time step	G	Integrating network
T_k	The k^{th} teammate	Φ	Set of Meta-path
$\mathcal{S}$	State space	E_{ω_1}	Trajectory encoder
$\mathcal{A}$	Action space	D_{ω_2}	Trajectory decoder
π^i	Policy of the controlled agent	$\bar{v}^m$	Mean value of the m^{th} cluster
$\boldsymbol{\pi}^{-i}$	Joint policy of all other agents	X	Set of nodes
a_i	Action of controlled agent	$\mathcal{E}$	Set of edges
$\boldsymbol{a}^{-i}$	Joint action of all other agents	$P(v_k^m \vert \tau_k)$	The probability of the k^{th} teammate
o_i	Local observation of agent i	α_{ij}	Weight for node pair i, j
u^i	Marginal utility of agent i	β^{Φ^i}	Weight for meta-path
B_t^i	Batch of local transition data	P	Number of meta-path
M	Number of clusters	T	Time horizon
τ_k	Trajectory which sampled from the interaction between the kth teammate and the environment		
v_k	The learned embedding represents the behavioural type of τ_k		
v_k^m	The embedding of the k th teammate belongs to the mth cluster based on its representation v_k		
n^m	The number of teammates belonging to the m^th cluster.		
C_t^i	Teamwork situation inferred by global information		
$e_{\Phi_p}^{ij}$	Importance of meta-path based node pair i, j		
Z_t^i	Teamwork situation inferred by local information		
$\mathcal{N}$	Dynamic set of agents involved in the task		

where $u^i(s, a^i, \boldsymbol{a}^{-i})$ denotes the marginal utility when the ad hoc agent chooses the action a^i under the state s. The agent chooses the action that maximizes the marginal utility to ensure the maximal team utility.

Definition 1. *(**Teamwork situation**) At each time step t, the teamwork situation $C_t^i \in C$, which is the current underlying teamwork state yielded by the environment state s_t and other teammates' actions $\mathbf{a}_t^{-i}$. It reflects the high-level semantics about the teammates' behaviours.*

Though different teammates generate diverse state-action trajectories, we assume that they may lead to similar teamwork situations at certain moments, and the ad hoc agent's action would affect their transitions. By accurately identifying the current teamwork situation, the ad hoc agent can choose the action accordingly to ensure online adaptation. In this work, the dynamic teammate mainly refers to the different combinations and numbers of teammates. At the beginning of each episode, the teammates' types are randomly sampled and kept fixed in that episode. However, the teammate combination and number are different across episodes. In addition, the set of behaviour types used for sampling is different for training and evaluation.

4 Methodology

4.1 The overall framework

Our training follows the CTDE setting Lowe et al. (2017), i.e., we can access the global state and other teammates' actions during the training phase but not during execution. Figure 2 depicts the overall framework of our Open-OTAF algorithm. We show the meaning of the notifications in Table 1.

As shown in Figure 2, in the training phase, we regard other teammates as a part of environmental dynamics perceived by the ad hoc agent. Different combinations of teammates lead to diverse and complex dynamics, we expect to learn a teamwork situation to describe the core information of teammates' behaviours implicitly. To effectively adjust to dynamic teammates, we design a teammate cluster model to distinguish teammates' behaviours and quantify the similarity between teammates via distribution (Section 4.2 and Figure 3(a)). Then, we introduce a global teamwork situation inference model that takes the controlled agent's state, the actions of its teammates, and the estimation of the team's composition as input. This inference model processes these inputs to infer and output the current teamwork situation C_t^i, enabling a more informed decision-making process. However, the size of the obtained embedding is changeable due to the variable

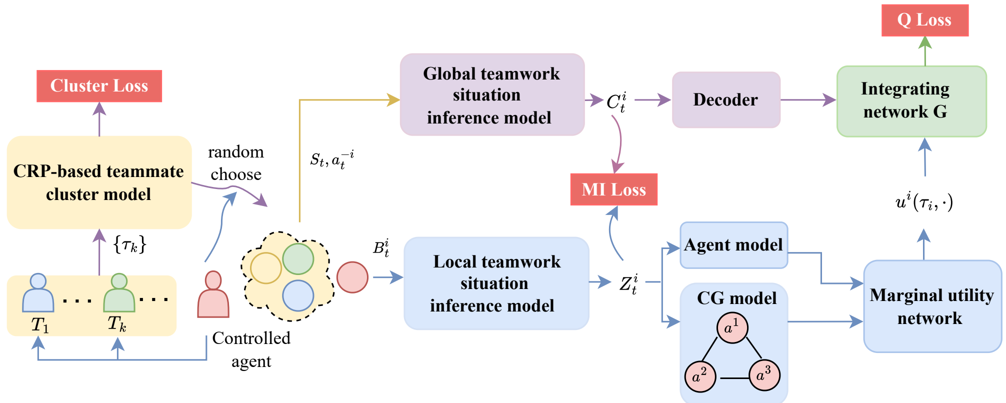


Figure 2: The overall framework of Open-OTAF. In the training phase, we introduce a global teamwork situation inference model to obtain C_t^i. Then, an integrating network G and a teamwork situation decoder are jointly proposed to regularize the information embedded in the learned variable. For the controlled agent, we introduce a local teamwork situation inference model to infer a proxy representation Z_t^i of C_t^i from local observation.

team size. To handle the dynamic team size and explicitly model the complex relationships among agents, we introduce HAN as an effective function approximation model to obtain the fixed-length embedding C_t^i, which can be optimized by the loss function (Section 4.3 and Figure 3(c)).

For the controlled agent, we expect to condition its policy on the current teamwork situation C_t^i. As the partial observability impedes the direct access to C_t^i, we introduce a local teamwork situation inference model to infer a proxy representation Z_t^i of C_t^i from local observation. We force Z_t^i to be informationally consistent with C_t^i by an information-based loss function (MI loss) to maximize the conditional mutual information. Then, we can then calculate the learner's marginal utility by simultaneously learning a factorized joint action value model (CG model) and an agent model. Then, we train a marginal utility network to estimate the ad hoc agent's conditional marginal utility (Section 4.4). For the controlled coordinating policy training, we sample representative teammates to coordinate with by capturing their identifications into distinguishing contexts to augment the joint policy during the centralized training phase. Each agent then utilizes its local information to approximate the global context information. The mentioned processes proceed alternately, and we can finally obtain a robust policy to adapt to any teammates gradually during the decentralized execution phase.

4.2 CRP-based teammate cluster model

In this section, we introduce the CRP-based teammate cluster model in detail. When identifying the teamwork situation, it is irrational and inefficient to treat each new teammate as a novel type ignoring their similarities. Accordingly, we aim to acquire clearly distinguishable boundaries of teammates' behaviours by applying a behaviour detection module to assign teammates with similar behaviours to the same cluster.

As shown in Figure 3(a), we have a batch of training teammates with the corresponding trajectories $\{\{\tau_1\}, \{\tau_2\}, \cdots, \{\tau_k\}, \cdots\}$ sampled from the interaction between the k^{th} teammate and the environment, where $\tau_k = (s_k^0, a_k^0, \cdots, s_k^T)$ and T demotes the time horizon. Given the high dimensionality of the trajectories, we use a trajectory encoder E_{ω_1} to extract features from the trajectory by $v_k = E_{\omega_1}(\tau_k)$, which can be used to represent its behavioural type, and $\bar{v}^m$ is the mean value of the m^{th} cluster. To calculate the teammate's distribution, we partition the trajectory τ_k into $\tau_k^s = (s_0, s_1, \cdots, s_{T-1}, s_T)$ and $\tau_k^a = (a_0, \cdots, a_{T-1})$. If M clusters are instantiated so far, the cluster that the k^{th} teammate belongs to will be inferred by $P(v_k^m|\tau_k) = P(v_k^m|\tau_k^s, \tau_k^a), m = 1, \cdots, M, M+1$. Let n^m be the number of teammates belonging to the m^{th} cluster. The v_k^m represents that the k^{th} teammate belongs to the m^{th} cluster based on its representation v_k, where $v_k^m = \frac{n^m \bar{v}^m}{n^m+1}$ when $m \le M$, else $v_k^m = v_k$ if $m = M+1$. Then, the cluster that the k^{th} teammate

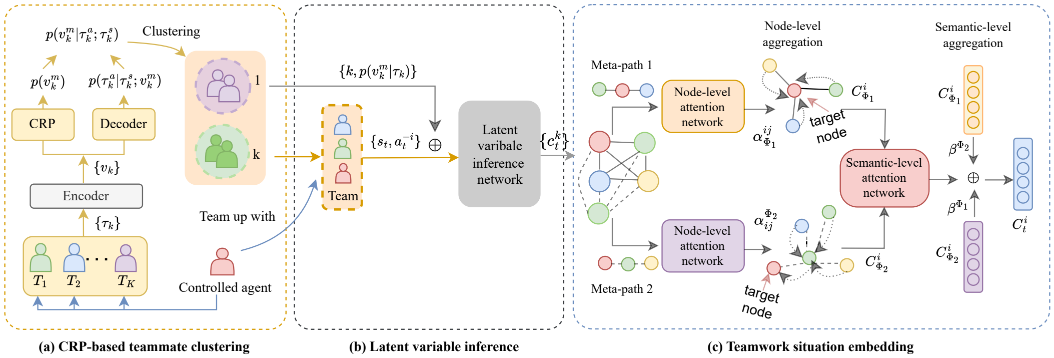


Figure 3: The global teamwork situation inference model: we first design a teammate cluster model to acquire clearly distinguishable boundaries of teammates' behaviours and formulate the distribution of dynamic teammates. Then, we introduce a variational auto-encoder to get the teamwork situation by taking the global state S and teammate actions a^{-i} as input. Thus, the current teamwork situation can be encoded by variable-size latent probabilistic variables $\{c_t^k\}$. Then, we introduce the HAN to learn a fixed-size embedding of the teamwork situation.

belongs to can be inferred as $P(v_k^m|\tau_k) = P(v_k^m|\tau_k^s, \tau_k^a)$. Based on the Bayesian rules Blei & Frazier (2011), the $P(v_k^m|\tau_k^s, \tau_k^a)$ can be decomposed as $P(v_k^m|\tau_k^s, \tau_k^a) \propto P(v_k^m)P(\tau_k^a|v_k^m; \tau_k^s)$.

The prior $P(v_k^m)$ is given by CRP, representing the probability of the k^{th} teammate belonging to m^{th} cluster without the teammate strategy information. If M clusters are formed so far, the prior $P(v_k^m) = n^m/(k-1+\lambda)$ when $m \leq M$, else $P(v_k^m) = \lambda/(k-1+\lambda)$ when $m = M+1$, where n^m denotes the number of teammate belonging to the m^{th} cluster, $\sum_{m=1}^{M} n^m = k-1$ and $\lambda > 0$ is a concentration hyperparameter that controls the probability of the formation of a new cluster. To estimate the conditional probability $P(\tau_k^a|\tau_k^s, v_k^m)$, we use a parametric network D_{ω_2} that take τ_k^s, v_k^m as input to estimate the likelihood of taking action τ_k^a as $P(\tau_k^a|v_k^m; \tau_k^s) = D_{\omega_2}(\tau_k^a|\tau_k^s; v_k^m)$. Combining the estimated prior distribution and predictive likelihood, we are able to decide which cluster the k^{th} teammate belongs to and their corresponding probability. After the assignment, the mean value of the m^{th} cluster, i.e., $\bar{v}^m$ will be updated. To force the learned representation v to capture the behavioural information of each teammate and estimate the predictive likelihood more precisely, the E_{ω_1} and D_{ω_2} can be optimized by:

$$\mathcal{L}_C(\omega) = \mathbb{E}_{\tau \sim \bigcup_{k=1}^{K} \mathcal{D}_k} - \log[D_{\omega_2}(\tau^a|\tau^s; E_{w_1}(\tau)), \tag{1}$$

where K is the number of teammates and $\omega = (\omega_1, \omega_2)$.

4.3 Learning to represent teamwork situation

After the teammates are divided into different clusters, we randomly choose multiple teammates to interact with the controlled agent. Despite the diversity and complexity of unknown teammates, we assume they can cause similar teamwork situations at some time. The controlled agent can choose an action based on the current teamwork situation accordingly. As shown in figure 3(b), we encode the teamwork situation in a stochastic embedding space via a latent variable inference network, which takes the concatenation of the state S, teammate actions a^{-i} and estimated teammate information $\{k, p(v_k^m|\tau_k)\}$ as input. Thus, the current teamwork situation can be encoded by latent probabilistic variables $\{c_t^k\}$ drawn from a multivariate Gaussian distribution $\mathcal{N}(\mu_{c_t}; \sigma_{c_t})$. Note that the size of $\{c_t^k\}$ corresponds to the number of teammates, which can change across episodes in the open ad hoc teamwork. Thus, we introduce the heterogeneous attention graph neural network, using the inherent ability of the graph neural network to transform the dynamic-size teammate's information into a fixed-size output.

We denote the heterogeneous graph as $H = (X, \mathcal{E})$, where $X = \{X_i|\forall i \in \{1, \cdots, n\}\}$ is a set of nodes representing the team agents, and $\mathcal{E}$ is the set of symmetric edges between every node pair. We use a heterogeneous graph attention neural network (HAN) with various types of edges to model open ad hoc teamwork. The reason is that the agent number is variable and the relationships among agents are different in open ad hoc teamwork. As shown in Figure 4, the controlled agents may not construct a cooperation relationship with the teammates planning to leave or just entering the environment, and the influence of teammates on the controlled agent may be different. We model the interaction relationship between the controlled agent and different teammates via different meta-paths, i.e., $\Phi = \{cooperate, no\ cooperate\}$.

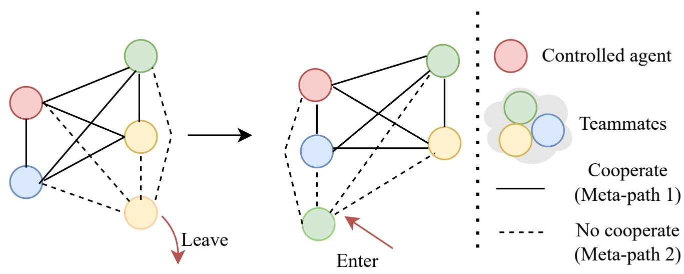


Figure 4: Heterogeneous graph representation of open ad hoc teamwork.

As shown in Figure 3(c), each meta-path has its own attention network and learns the attention weights $\alpha^{ij}_{\Phi_p}$ for the neighboring agents of the corresponding relationship. For instance, the "cooperate" meta-path only takes the node embedding of the neighboring agents which cooperate with the controlled agent as input. Thus, different meta-paths have diverse semantic information. Concretely, the importance of meta-path based node pair $\{i, j\}$ can be formulated as $e^{ij}_{\Phi_p} = att_{node}(c^i_t, c^j_t; \Phi_p)$, where att_{node} denotes the deep neural network which performs the node-level attention. After that, we normalize them to get the weight coefficient $\alpha^{ij}_{\Phi_p} = softmax_j(e^{ij}_{\Phi_p})$. Then, the learned embedding of node i for the meta-path Φ_p can be calculated as $C^i_{\Phi_p} = g(\sum_{j\in\mathcal{N}^i_{\Phi_p}} \alpha^{\Phi_p}_{ij} \cdot c^j_t)$, where $\mathcal{N}^i_{\Phi_p}(p \in \{1, 2\})$ denotes the meta-path p based neighbors of node i, and g denote the activate function. Then, the final teamwork situation embedding is aggregated by all semantic embeddings, where the semantic level attention network is used to learn the weight β^{Φ} of different meta-paths. We can fuse these semantic-specific embeddings to obtain the final embedding as $C^i_t = \sum_{p=1}^{2} \beta^i_{\Phi_p} \cdot C^i_{\Phi_p}$. Our model can get the optimal combination of neighbors and multiple meta-paths in a hierarchical manner, which enables C^i_t to better capture the complex structure and rich semantic information.

If C^i_t is able to capture the core knowledge of the current teamwork situation, we can predict the joint action value $Q^{\pi_i}(s_t, a^i_t, a^{-i}_t)$ according to C^i_t and the ad hoc agent's marginal utility u^i_t (Section 4.4). Thus, we use an integrating network G for generating the joint action value's estimation, i.e., $G(u^i_t, C^i_t) \approx Q_\pi(s_t, a^i_t, a^{-i}_t)$. Finally, we adopt a modified asynchronous Q-learning loss function (Q-loss) Mnih et al. (2016) as the optimization objective:

$$\mathcal{L}_Q = \mathbb{E}_{u^i_t, C^i_t, r_t \sim \mathcal{D}}[[r_t + \gamma \max_{a^i_{t+1}} \bar{G}(u^i_{t+1}, C^i_{t+1}) - G(u^i_t, C^i_t)]^2] \tag{2}$$

where $\bar{G}$ is a periodically updated target network. The expectation is estimated with uniform samples from the replay buffer $\mathcal{D}$.

4.4 Learning marginal utility function under partial observability

For ad hoc agents, we aim to model the marginal utility as a conditional function on the inferred teamwork situations. However, partial observability impedes the agent's access to C^i_t encoded from the global state-action trajectory. Thus, we introduce a local inference model to derive proxy representations of the teamwork situation from local observations. Concretely, the network architecture takes local trajectory B^i_t as input and outputs $z^i_t \sim \mathcal{N}(\mu_{\phi_i(B^i_t)}, \sigma_{\phi_i(B^i_t)})$. Then, with the predicted teammate type information, we obtain the final teamwork situation embedding Z^i_t via the heterogeneous attention graph neural network. To make Z^i_t informatively consistent with C^i_t, we introduce an information-based loss function $\mathcal{L}_{MI}$ to maximize the conditional mutual information $I(Z^i_t; C^i_t | B^i_t)$ between Z^i_t and C^i_t. Due to the difficulty and feasibility of estimating the mutual information, a variational distribution $q_\xi(Z^i_t | C^i_t, B^i_t)$ is used to approximate the conditional distribution $p_\xi(Z^i_t | C^i_t, B^i_t)$. Inspired by the information bottleneck Alemi et al. (2016), we derive a tractable lower bound for the MI as follows.

Theorem 4.1. *Let $I(Z_t^i; C_t^i | B_t^i)$ be the conditional mutual information between the local teamwork situation embedding Z_t^i of agent i and global embedding C_t^i. Then, the lower bound is given by:*

$$I(Z_t^i; C_t^i | B_t^i) \geq \mathbb{E}_{Z_t^i, C_t^i, B_t^i}\Big[\log \frac{q_\xi(Z_t^i | C_t^i, B_t^i)}{p(Z_t^i | B_t^i)}\Big]. \tag{3}$$

The lower bound can be rewritten as a loss function:

$$\mathcal{L}_{MI} = \mathbb{E}_{Z_t^i, C_t^i, \tau_t^i \sim \mathcal{D}}[D_{KL}[p(Z_t^i | B_t^i) || q_\xi(Z_t^i | C_t^i, B_t^i)]], \tag{4}$$

where $\mathcal{D}$ is the replay buffer.

Notice that the entropy of our labels is independent of our optimization procedure and so can be ignored. The proxy encoder is conditioned on the transition data. Given the transitions, the distribution of the proxy representations $p(Z_t^i)$ is independent of the local histories. Thus, we have:

$$I(Z_t^i; C_t^i | B_t^i) \geq -\mathbb{E}_{C_t^i, B_t^i}[\mathcal{CE}[p(Z_t^i | C_t^i, B_t^i) \| q_\xi(Z_t^i | C_t^i, B_t^i)] dZ_t^i].$$

In practice, we sample data from the replay buffer, and the lower bound can be rewritten as a loss function to be minimized by:

$$\mathcal{L}_{MI} = \mathbb{E}_{Z_t^i, C_t^i, B_t^i}[D_{KL}[p(Z_t^i | B_t^i) \| q_\xi(Z_t^i | C_t^i, B_t^i)]]. \tag{5}$$

To achieve a robust policy, we use the inferred Z_t^i to estimate the agent's marginal utility $\mu(Z_t^i, a_t^i)$. Considering the influence of other teammates on the controlled agent, we adopt the coordination graphs (CGs) Rahman et al. (2021) to estimate the ad hoc agent's marginal utility. Given the embedding Z_t^i, the agent modelling module computes the likelihood of the agent's actions $p(a^j | Z_t^i)$. Then, the marginal utility of agent i can be calculated as:

$$\begin{aligned}\mu(Z_t^i, a_t^i) = Q^i(a_t^i | Z_t^i) + \sum_{j \in N_t, j \neq i} (Q^j(a_t^j | Z_t^i) + Q^{i,j}(a_t^i, a_t^j | Z_t^i)) p(a^j | Z_t^i) \\ + \sum_{j,k \in N_t; j,k \neq i} Q^{j,k}(a_t^j, a_t^k | Z_t^i) p(a_t^j | Z_t^i) p(a_t^k | Z_t^i).\end{aligned} \tag{6}$$

$Q^j(a_t^j | Z_t^i)$ represents $j's$ contribution towards the learner's returns by executing a^j, while $Q^{j,k}(a_t^j, a^k | Z_t^i)$ denotes j and $k's$ contribution towards the learner's returns by jointly choosing a^j and a^k. We implement $Q^j(a_t^j | Z_t^i)$ and $Q^{j,k}(a_t^j, a_t^k | Z_t^i)$ as multilayer perceptrons (MLPs) parameterized by β and δ to enable generalization across different numbers of the agents. Afterward, the marginal utility can be used to approximate the joint action-value function. Therefore, the overall optimization objective can thus be derived as $\mathcal{L}_{tot} = \mathcal{L}_Q + \mathcal{L}_{MI}$. During the training phase, the controlled agent interacts with different teammates to collect transition data into the replay buffer. Then, samples from $\mathcal{D}$ are fed into the framework for updating all parameters induced by the overall loss. During execution, the controlled agent conditions its behaviour on the inferred teamwork situations by choosing actions to maximize the utility function. We summarize our training procedure in Algorithm 1 in the Appendix A.1.

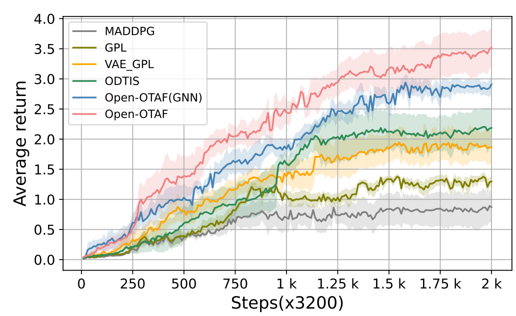
5 Experiment

To thoroughly assess the Open-OTAF algorithm, we evaluate our model in four widely-used partial observable multi-agent environments for various settings, including the Level-Based Foraging (LBF), Wolfpack, the Penalized Cooperative Navigation (PCN), and the Fortattack Rahman et al. (2022). From the experiments, we aim to answer the following questions: 1) Can Open-OTAF yield superior performance than SOTA baseline methods (Figure 5)? 2): Are the main components, i.e., the CRP, MI, and AM, necessary and effective (Figure 6, Figure A.5)? 3) Does Open-OTAF have generalization capacity with the varying teammate size and type (Figure A.6 and Table 2)?

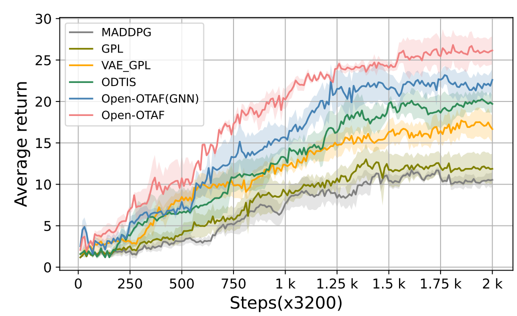
5.1 Experiment Setup and baselines

We select four multi-agent tasks as our environments, as shown in Figure A.1 in the Appendix A.2.1. Among them, Level Based Foraging (LBF), Wolfpack and Fortattack are three scenarios coming from Rahman et al. (2021). LBF is a cooperative grid world game with agents that are rewarded if they concurrently navigate to the food and collect it. In Wolfpack, multiple agents (predators) need to chase and encounter the adversary agent (prey) to win the game. The FortAttack environment defines a bounded two-dimensional space where agents are constrained within specific coordinate ranges. We also evaluate our method in the penalized cooperative navigation, coming from the MPE environment Lowe et al. (2017), where multiple agents are trained to move towards landmarks while avoiding collisions with each other. More details about the environments and aldorithm configuration are provide in the Appendix A.2.1 and A.3.

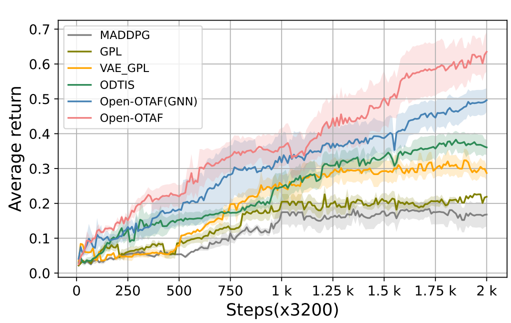
To illustrate the advancement of our method, we compare the proposed method with two type-based baselines, GPL and VAE-GPL, which can be used to solve open ad hoc teamwork under full and partial observability separately. Although GPL assumes full observability of the state, we apply GPL in our experiments by treating the learner's observations as input. ODITS Gu et al. (2021) applies a centralized "teamwork situation encoder" for end-to-end learning to adapt to arbitrary teammates in closed environments. As ODITS considers only a single ad hoc agent in the closed environment, we further extend the ODTIS to open environments by introducing a graph neural network. Lastly, to demonstrate the advantages of HAN, we also compare the algorithm Open-OTAF(GNN), where GNNs are introduced to produce the representations of teamwork situations. Table A.1 (in Appendix A.2.4) provides a summary of the different algorithms.



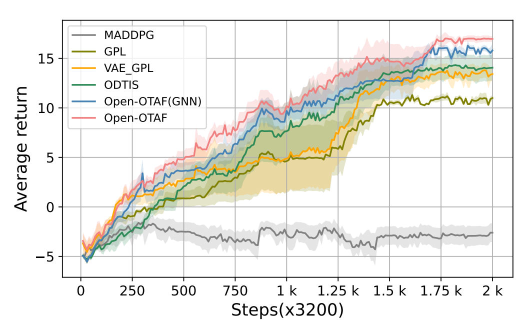
(a) LBF.



(b) Wolfpack.



(c) PCN.



(d) Fortattack.

Figure 5: Performance comparison across various scenarios. For LBF, Wolfpack, PCN and Fortattack, the training steps are 640, 0000 and we evaluate all the methods with 100 test episodes after every 3200 training steps. We report the performance in terms of average returns (solid line) and the standard deviation (shaded areas) with 7 random seeds.

5.2 Experiment results

5.2.1 Analysis of the average return

We first compare the overall performance of Open-OTAF with baselines. The results are summarized in Figure 5 and Table 2. From the curves, we observe that the returns of Open-OTAF consistently outperform the other four baselines with a faster convergence speed in all benchmark games while possessing a lower variance, which basically shows the effectiveness and high efficiency of the proposed method. In particular, the comparison with Open-OTAF(GNN) suggests the effectiveness of using HAN to produce representations of the teamwork situation. The utilization of HAN can boost the performance of open ad hoc teamwork than conventional GNN. Otherwise, the comparison between Open-OTAF and ODTIS shows that using the coordination graphs and agent model for optimal marginal utility estimation can boost performance as well. It enables our method to estimate the marginal utility by considering the effect of other agents, which leads to significantly better training. Also, ODITS shows superior performance than GPL and VAE-GPL, manifesting the necessity of utilizing global states to improve training efficiency. Furthermore, GPL performs poorly than other methods because the sequence of observations perceived by the learner contains the least useful information compared to other methods. These results show that our method can design an autonomous agent capable of robust teamwork under dynamic teammates without pre-coordination.

5.2.2 Ablation studies

To determine the importance of each component of Open-OTAF, we conduct an ablation study for the different components to see how it affects the training, and the results over 5 random seeds are shown in Figure 6. More ablation studies are provided in the Appendix A.4. First, to illustrate the importance of the CRP-based teammate cluster, we derive the method of w/o [1] CRP by removing the CRP process and taking each new teammate as a new cluster. Furthermore, we pick up w/o MI to investigate how maximizing mutual information between global and local information accelerates learning efficiency. Finally, w/o AM is introduced to illustrate the impact of other teammates in the calculation of the marginal utility. We only utilize a network to calculate the marginal utility without considering the effect of other agents.

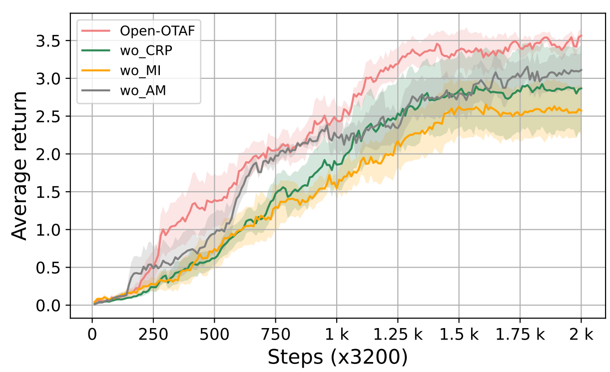


Figure 6: Ablations for different components.

In Figure 6, the comparison results show that w/o MI suffers the most severe performance degradation in all games, which demonstrates the necessity of utilizing global information to facilitate the learning of the local teamwork situation inference model. Furthermore, w/o CRP achieves lower performance than Open-ATAF, which manifests that the CRP-based teammate model helps the controlled agent adapts to dynamic teammates rapidly. We also find agent modelling helps learn more accurate marginal utility by considering the effect of other teammates, which can bring about a slight coordination improvement. It can also demonstrate that the cluster model can help to learn a robust ad hoc agent. The testing results are also provided in Figure A.5, where we provide the average return (bars) and the standard deviation (error bars) over different seeds. The results that the introduction of the CRP model for teammate types classification and the guidance of global information help agents to estimate more accurate teamwork situations.

5.2.3 The generalization capacity

To assess the generalization ability of Open-OTAF, we performed experiments involving different teammate numbers and types. First, we evaluate the generalization ability by changing the teammate types during

[1] We use the w/o in place of without for conciseness.

testing. Similar to the training procedure of Open-OTAF, we control one agent and test it with teammates randomly sampled from a pool consisting of 10 different policies, which were not seen during training. The generalization performance of Open-OTAF and the baselines is provided in Table 2. We show the average and 95% confidence bounds during testing utilizing 5 seeds. The data was gathered by averaging the returns at the checkpoint which achieved the highest average performance during training. We highlight in bold the algorithm with the highest average returns. The results show that our method significantly outperforms other baselines during testing. Unsurprisingly, methods that achieve low returns during training, will also achieve better performance when generalizing to different teammate types. Open-OTAF's performance in this diverse teammate-type setting was significant, outclassing the baseline performances by 5.1% $\sim$ 49.8%.

Table 2: Results of generalization capacity under different teammate types.

Env	MADDPG	GPL	VAE_GPL	ODTIS	Open-OTAF(G)	Open-OTAF
LBF	$0.73_{(0.62)}$	$1.85_{(0.51)}$	$2.14_{(0.47)}$	$2.17_{(0.45))}$	$2.39_{(0.31)}$	$\mathbf{2.67_{(0.25)}}$
Wolf	$8.07_{(1.25))}$	$16.9_{(1.24))}$	$19.3_{(1.39)}$	$21.4_{(1.48)}$	$24.3_{(1.42)}$	$\mathbf{26.9_{(1.26)}}$
PCN	$0.11_{(0.15)}$	$0.43_{(0.12)}$	$0.52_{(0.11)}$	$0.69_{(0.11)}$	$0.77_{(0.10)}$	$\mathbf{0.86_{(0.09)}}$
Fortattack	$0.12_{(0.26)}$	$0.44_{(0.22)}$	$0.49_{(0.22)}$	$0.54_{(0.28)}$	$0.58_{(0.22)}$	$\mathbf{0.62_{(0.11)}}$

Moreover, to assess Open-OTAF's generalization ability further, we conduct an expanded experiment involving different teammate sizes. Open-OTAF was initially trained on 5 teammates with 10 different training policies and then tested on 3, 5, 6, 8 and 10 teammates with 10 different testing teammate policies. Results, summarised in Figure A.6 (in Appendix A.4), indicate a consistently superior performance of Open-OTAF. Experimental results show that, with desirable generalization to various teammate sizes, our method outperforms the state-of-the-art method. In summary, when encountering unknown teammates, the CRP-based module first classifies them into one of the 10 clusters (we set the maximum of cluster number as 10). Then, the attention network uses the clustering results as the semantic information to generate weights for better teamwork situation estimation. The teamwork situation is the input of the controlled agent's policy and affects its action selection. We admit that we may not train a best-response policy for unknown teammates. It is also impossible to train a best-response policy for every possible unknown teammate. However, the trained clustering ability of the CRP-based module and the generalization capability of the attention network allows the controlled agent to select reasonably good actions. The ablation studies, as shown in Figures A.6, and Table 2 demonstrate promising outcomes. It can be noted that the CRP-based teammate cluster module and the heterogeneous attention graph neural network (HAN) can improve generalization performance.

6 Conclusion

This work proposes a novel online adaptation reinforcement learning algorithm to address the challenging open ad hoc teamwork problem. In Open-OTAF, the CRP-based teammate cluster model aims to predict the probability distributions of the teammate's types, which can be considered as the agent's beliefs about the current status of its team. The inferred teammate types with environment state are fed into a representation learning model to encode the teamwork situation representations as latent probabilistic variables. With the assistance of the heterogeneous attention graph, we manage to infer the teamwork situation and adapt to the new teammate effectively. We also propose an information-based proxy encoder to infer the learned variables from local observations to overcome partial observability. With the guide of the global teamwork situation, the ad hoc agent adapts to new teammates' behaviours dynamically and quickly by conditioning its policy on the inferred variables. Extensive experiments demonstrate that Open-OTAF not only obtains a higher average return but also achieves a faster convergence speed. Our method can be seen as a primary attempt for online teammate adaptation with dynamic teammates in open ad-hoc teamwork, we sincerely hope it can be a solid foothold for applying RL to practical applications. For future work, we will relax the CTDE assumption to enhance the applicability of our approach, investigating a robust ad hoc agent within a fully decentralized framework.

References

Noa Agmon and Peter Stone. Leading ad hoc agents in joint action settings with multiple teammates. In *AAMAS*, pp. 341–348, 2012.

Stefano V Albrecht and Subramanian Ramamoorthy. A game-theoretic model and best-response learning method for ad hoc coordination in multiagent systems. *arXiv preprint arXiv:1506.01170*, 2015.

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

Stefano V Albrecht and Peter Stone. Reasoning about hypothetical agent behaviours and their parameters. *arXiv preprint arXiv:1906.11064*, 2019.

Stefano V Albrecht, Cillian Brewitt, John Wilhelm, Balint Gyevnar, Francisco Eiras, Mihai Dobre, and Subramanian Ramamoorthy. Interpretable goal-based prediction and planning for autonomous driving. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1043–1049. IEEE, 2021.

Alexander A Alemi, Ian Fischer, Joshua V Dillon, and Kevin Murphy. Deep variational information bottleneck. *arXiv preprint arXiv:1612.00410*, 2016.

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

Samuel Barrett, Peter Stone, and Sarit Kraus. Empirical evaluation of ad hoc teamwork in the pursuit domain. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pp. 567–574, 2011.

Samuel Barrett, Avi Rosenfeld, Sarit Kraus, and Peter Stone. Making friends on the fly: Cooperating with new teammates. *Artificial Intelligence*, 242:132–171, 2017.

David M Blei and Peter I Frazier. Distance dependent chinese restaurant processes. *Journal of Machine Learning Research*, 12(8), 2011.

Wendelin Böhmer, Vitaly Kurin, and Shimon Whiteson. Deep coordination graphs. In *International Conference on Machine Learning*, pp. 980–991. PMLR, 2020.

Manuel Boldrer, Alessandro Antonucci, Paolo Bevilacqua, Luigi Palopoli, and Daniele Fontanelli. Multi-agent navigation in human-shared environments: A safe and socially-aware approach. *Robotics and Autonomous Systems*, 149:103979, 2022.

George W Brown. Iterative solution of games by fictitious play. *Activity analysis of production and allocation*, 13(1):374–376, 1951.

Muthukumaran Chandrasekaran, Adam Eck, Prashant Doshi, and Leenkiat Soh. Individual planning in open and typed agent systems. In *Proceedings of the Thirty-Second Conference on Uncertainty in Artificial Intelligence*, pp. 82–91, 2016.

Feng Chen, Xinwei Chen, Rongjun Qin, Cong Guan, Lei Yuan, Zongzhang Zhang, and Yang Yu. Efficient multi-agent cooperation learning through team-mate lookahead. *Journal of Transactions on Machine Learning Research*, 2025.

Shuo Chen, Ewa Andrejczuk, Zhiguang Cao, and Jie Zhang. Aateam: Achieving the ad hoc teamwork by employing the attention mechanism. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 7095–7102, 2020.

Caroline Claus and Craig Boutilier. The dynamics of reinforcement learning in cooperative multiagent systems. *AAAI/IAAI*, 1998(746-752):2, 1998.

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.

Katie Genter, Noa Agmon, and Peter Stone. Role-based ad hoc teamwork. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 25, pp. 1782–1783, 2011.

Sven Gronauer and Klaus Diepold. Multi-agent deep reinforcement learning: a survey. *Artificial Intelligence Review*, pp. 1–49, 2022.

Pengjie Gu, Mengchen Zhao, Jianye Hao, and Bo An. Online ad hoc teamwork under partial observability. In *International Conference on Learning Representations*, 2021.

He He, Jordan Boyd-Graber, Kevin Kwok, and Hal Daumé III. Opponent modeling in deep reinforcement learning. In *International conference on machine learning*, pp. 1804–1813. PMLR, 2016.

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, 2019.

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

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR (Poster)*, 2015.

Yang Li, Shao Zhang, Jichen Sun, Wenhao Zhang, Yali Du, Ying Wen, Xinbing Wang, and Wei Pan. Tackling cooperative incompatibility for zero-shot human-ai coordination. *Journal of Artificial Intelligence Research*, 80:1139–1185, 2024.

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

Ryan Lowe, Yi I Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *Advances in neural information processing systems*, 30, 2017.

Reuth Mirsky, William Macke, Andy Wang, Harel Yedidsion, and Peter Stone. A penny for your thoughts: The value of communication in ad hoc teamwork. *Good Systems-Published Research*, 2020.

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 *Multi-Agent Systems: 19th European Conference, EUMAS 2022, Düsseldorf, Germany, September 14–16, 2022, Proceedings*, pp. 275–293. Springer, 2022.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pp. 1928–1937. PMLR, 2016.

Navid Naderializadeh, Fan H Hung, Sean Soleyman, and Deepak Khosla. Graph convolutional value decomposition in multi-agent reinforcement learning. *arXiv preprint arXiv:2010.04740*, 2020.

Thanh Thi Nguyen, Ngoc Duy Nguyen, and Saeid Nahavandi. Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications. *IEEE transactions on cybernetics*, 50(9): 3826–3839, 2020.

Arrasy Rahman, Ignacio Carlucho, Niklas Höpner, and Stefano V Albrecht. A general learning framework for open ad hoc teamwork using graph-based policy learning. *arXiv preprint arXiv:2210.05448*, 2022.

Arrasy Rahman, Elliot Fosong, Ignacio Carlucho, and Stefano V Albrecht. Generating teammates for training robust ad hoc teamwork agents via best-response diversity. *Transactions on Machine Learning Research*, 2023.

Muhammad Rahman, Jiaxun Cui, and Peter Stone. Minimum coverage sets for training robust ad hoc teamwork agents. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 17523–17530, 2024.

Muhammad A Rahman, Niklas Hopner, Filippos Christianos, and Stefano V Albrecht. Towards open ad hoc teamwork using graph-based policy learning. In *International Conference on Machine Learning*, pp. 8776–8786. PMLR, 2021.

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 *International Conference on Machine Learning*, pp. 4295–4304. PMLR, 2018.

Manish Chandra Reddy Ravula. *Ad-hoc teamwork with behavior-switching agents.* PhD thesis, 2019.

João G Ribeiro, Cassandro Martinho, Alberto Sardinha, and Francisco S Melo. Assisting unknown teammates in unknown tasks: Ad hoc teamwork under partial observability. *arXiv preprint arXiv:2201.03538*, 2022.

Martin Roesch, Christian Linder, Roland Zimmermann, Andreas Rudolf, Andrea Hohmann, and Gunther Reinhart. Smart grid for industry using multi-agent reinforcement learning. *Applied Sciences*, 10(19):6900, 2020.

Lukas Schäfer, Filippos Christianos, Amos Storkey, and Stefano V Albrecht. Learning task embeddings for teamwork adaptation in multi-agent reinforcement learning. *arXiv preprint arXiv:2207.02249*, 2022.

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 *International conference on machine learning*, pp. 5887–5896. PMLR, 2019.

Peter Stone and Sarit Kraus. To teach or not to teach?: decision making under uncertainty in ad hoc teams. In *AAMAS*, pp. 117–124. Citeseer, 2010.

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, pp. 1504–1509, 2010.

Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, et al. Value-decomposition networks for cooperative multi-agent learning. *arXiv preprint arXiv:1706.05296*, 2017.

A Tacchetti, H Francis Song, PAM Mediano, V Zambaldi, J Kramár, NC Rabinowitz, T Graepel, M Botvinick, and PW Battaglia. Relational forward models for multi-agent learning. In *7th International Conference on Learning Representations, ICLR 2019*. ICLR, 2019.

Jianhao Wang, Zhizhou Ren, Terry Liu, Yang Yu, and Chongjie Zhang. Qplex: Duplex dueling multi-agent q-learning. *arXiv preprint arXiv:2008.01062*, 2020.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24, 2020.

Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of ppo in cooperative multi-agent games. *Advances in Neural Information Processing Systems*, 35:24611–24624, 2022.

Jilie Zhang, Zhanshan Wang, and Hongwei Zhang. Data-based optimal control of multiagent systems: A reinforcement learning design approach. *IEEE transactions on cybernetics*, 49(12):4441–4449, 2018.

Kaiqing Zhang, Zhuoran Yang, and Tamer Başar. Multi-agent reinforcement learning: A selective overview of theories and algorithms. *Handbook of Reinforcement Learning and Control*, pp. 321–384, 2021.

Ziqian Zhang, Lei Yuan, Lihe Li, Ke Xue, Chengxing Jia, Cong Guan, Chao Qian, and Yang Yu. Fast teammate adaptation in the presence of sudden policy change. *arXiv preprint arXiv:2305.05911*, 2023.

Weigui Jair Zhou, Budhitama Subagdja, Ah-Hwee Tan, and Darren Wee-Sze Ong. Hierarchical control of multi-agent reinforcement learning team in real-time strategy (rts) games. *Expert Systems with Applications*, 186:115707, 2021.

Luisa Zintgraf, Sam Devlin, Kamil Ciosek, Shimon Whiteson, and Katja Hofmann. Deep interactive bayesian reinforcement learning via meta-learning. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 1712–1714, 2021.

A Appendix

A.1 Details about Derivation

Details about CRP and derivation of cluster assignment Chinese restaurant process (CRP) is a discrete-time stochastic process that defines a prior distribution over the cluster structures. It can be described simply as follows: as each customer enters a restaurant with an infinite number of tables, he chooses to sit down alone at a new table with a probability proportional to a concentration parameter or sits with other customers with a probability proportional to the number of customers sitting on the occupied table. Customers sitting at the same table will be assigned to the same cluster. The cluster assignment of the k^{th} teammate $P(v_k^m|\tau_k^s, \tau_k^a)$ can be decomposed as:

$$\begin{aligned}
&P(v_k^m|\tau_k^s, \tau_k^a) \\
&= \frac{P(v_k^m, \tau_k^s, \tau_k^a)}{P(\tau_k^s, \tau_k^a)} \\
&= \frac{P(\tau_k^s, \tau_k^a|v_k^m)P(v_k^m)}{P(\tau_k^s, \tau_k^a)} \\
&= \frac{P(\tau_k^a|v_k^m, \tau_k^s)P(\tau_k^s|v_k^m)P(v_k^m)}{P(\tau_k^s, \tau_k^a)} \\
&\propto P(\tau_k^a|v_k^m, \tau_k^s)P(\tau_k^s|v_k^m)P(v_k^m).
\end{aligned} \tag{7}$$

As τ_k^s is a set of states that is not determined by the behavioural type of the teammates if neglecting the correlation in time dimensionality, $P(\tau_k^s|v_k^m)$ is a constant. Accordingly, we can obtain that $P(v_k^m|\tau_k^s, \tau_k^a) \propto P(v_k^m)P(\tau_k^a|v_k^m; \tau_k^s)$.

Variational Bound of teammates context approximation To make context vector Z_t^i generated by local trajectory encoder informatively consistent with global context C_t^i, we propose to maximize the mutual information between Z_t^i and C_t^i conditioned on the agent i's local trajectory B_t^i. We draw the idea from variational inference Alemi et al. (2016) and derive a lower bound of this mutual information term.

Theorem A.1. *Let $I(Z_t^i; C_t^i|B_t^i)$ be the conditional mutual information between the local teamwork situation embedding Z_t^i of agent i and global embedding C_t^i. Then, the lower bound is given by:*

$$I(Z_t^i; C_t^i|B_t^i) \geq \mathbb{E}_{Z_t^i, C_t^i, B_t^i}\Big[\log \frac{q_\xi(Z_t^i|C_t^i, B_t^i)}{p(Z_t^i|B_t^i)}\Big]. \tag{8}$$

The lower bound can be rewritten as a loss function:

$$\mathcal{L}_{MI} = \mathbb{E}_{Z_t^i, C_t^i, \tau_t^i \sim \mathcal{D}}[D_{KL}[p(Z_t^i|B_t^i)||q_\xi(Z_t^i|C_t^i, B_t^i)]], \tag{9}$$

where $\mathcal{D}$ is the replay buffer.

Proof.

$$
\begin{aligned}
I(Z_t^i; C_t^i | B_t^i) &= \mathbb{E}_{Z_t^i, C_t^i, B_t^i}\Big[\log \frac{q_\xi(Z_t^i | C_t^i, B_t^i)}{p(Z_t^i | B_t^i)}\Big] \\
&= \mathbb{E}_{Z_t^i, C_t^i, B_t^i}\Big[\log \frac{q_\xi(Z_t^i | C_t^i, B_t^i)}{p(Z_t^i | b_t^i)}\Big] \\
&+ \mathbb{E}_{C_t^i, B_t^i}[D_{KL} p(Z_t^i | C_t^i, B_t^i) \| q_\xi(Z_t^i | C_t^i, B_t^i))] \\
&\geq \mathbb{E}_{Z_t^i, C_t^i, B_t^i}\Big[\log \frac{q_\xi(Z_t^i | C_t^i, B_t^i)}{p(Z_t^i | B_t^i)}\Big],
\end{aligned}
\tag{10}
$$

where the last inequality holds because of the non-negativity of the KL divergence. Then it follows that:

$$
\begin{aligned}
&\mathbb{E}_{Z_t^i, C_t^i, B_t^i}\Big[\log \frac{q_\xi(Z_t^i | C_t^i, B_t^i)}{p(Z_t^i | B_t^i)}\Big] \\
&= \mathbb{E}_{Z_t^i, C_t^i, B_t^i}[\log q_\xi(Z_t^i | C_t^i, B_t^i)] - \mathbb{E}_{Z_t^i, B_t^i}[\log p(Z_t^i | B_t^i)] \\
&= \mathbb{E}_{Z_t^i, C_t^i, B_t^i}[\log q_\xi(Z_t^i | C_t^i, B_t^i)] + \mathbb{E}_{B_t^i}[\mathcal{H}(Z_t^i | B_t^i)] \\
&= \mathbb{E}_{C_t^i, B_t^i}\Big[\int p(Z_t^i | C_t^i, B_t^i) \log q_\xi(Z_t^i | C_t^i, B_t^i) dZ_t^i\Big] + \mathcal{H}(Z_t^i | B_t^i)
\end{aligned}
\tag{11}
$$

□

To illustrate the flow of Open-OTAF, we show the training procedure in Algorithm 1. A set of teammates can be generated via heuristic and RL algorithms, and we store the small batch of trajectories into a replay buffer D. The encoder and decoder are trained to force the learned representation to precisely capture the behavioral information and precisely estimate the predictive probability (Line 6). Afterward, the CRP prior and predictive likelihood are calculated to determine the assignment of the teammates m (Line 7-9). Then, we update the existing cluster or instantiate a new cluster based on the assignment (Line 11-14). In the training, we first sample a teammate from the cluster and fix it in this episode. The teammates pair with the controllable agents,and they make decisions together. To train the agent policy networks and the context encoders, the moving average values of context vectors are updated, and the optimization objectives are calculated (Line 24-29).

A.2 Details about Environment setup and Baselines

A.2.1 Environment setup

Setup for LBF: In LBF, the learner retrieves objects that are positioned in an 8×8 grid world, as shown in Figure A.1(a). In the Open Dec-POMDP setting, one agent is controllable and will stay in the environment for the whole episode. The maximum number of teammates is 5, where the teammates can enter and leave the environment freely and their policy will change as well. Each agent, including the learner and their teammates, is assigned a numerical level, as are the objects themselves. The level of agents and the foods are randomly chosen from $\{1, 2, 3\}$ at the beginning of an episode. In the partially observable version, agents can only observe entities in a 5×5 grid surrounding them, including the positions and levels of all the entities. Any agent has six actions, i.e., [up, down, left, right, load, no-op], where the first four actions mean the agent moves towards the corresponding direction, load means loading food next to it, and a no-op action indicates doing-nothing during an entire episode. The agents' objective is to collect all the objects and the collection is only successful if the sum of the involved agents' levels is equal to or greater than the item level. Every agent that collects an object is given a reward equal to the level of the object. An episode terminates if all available objects are collected or after 50 timesteps.

Setup for Wolfpack: In Figure A.1(b), the multiple hunters attempt to catch randomly prey in a grid world of size a 10×10, where the "catching" means the prey is in the cardinal direction of at least two hunters. Partial observability is induced by limiting the learner's observation to entities within a Manhattan distance of 3 from itself. The observation of each agent is the coordinates of its location, its ID, and the

Algorithm 1 Training Procedure

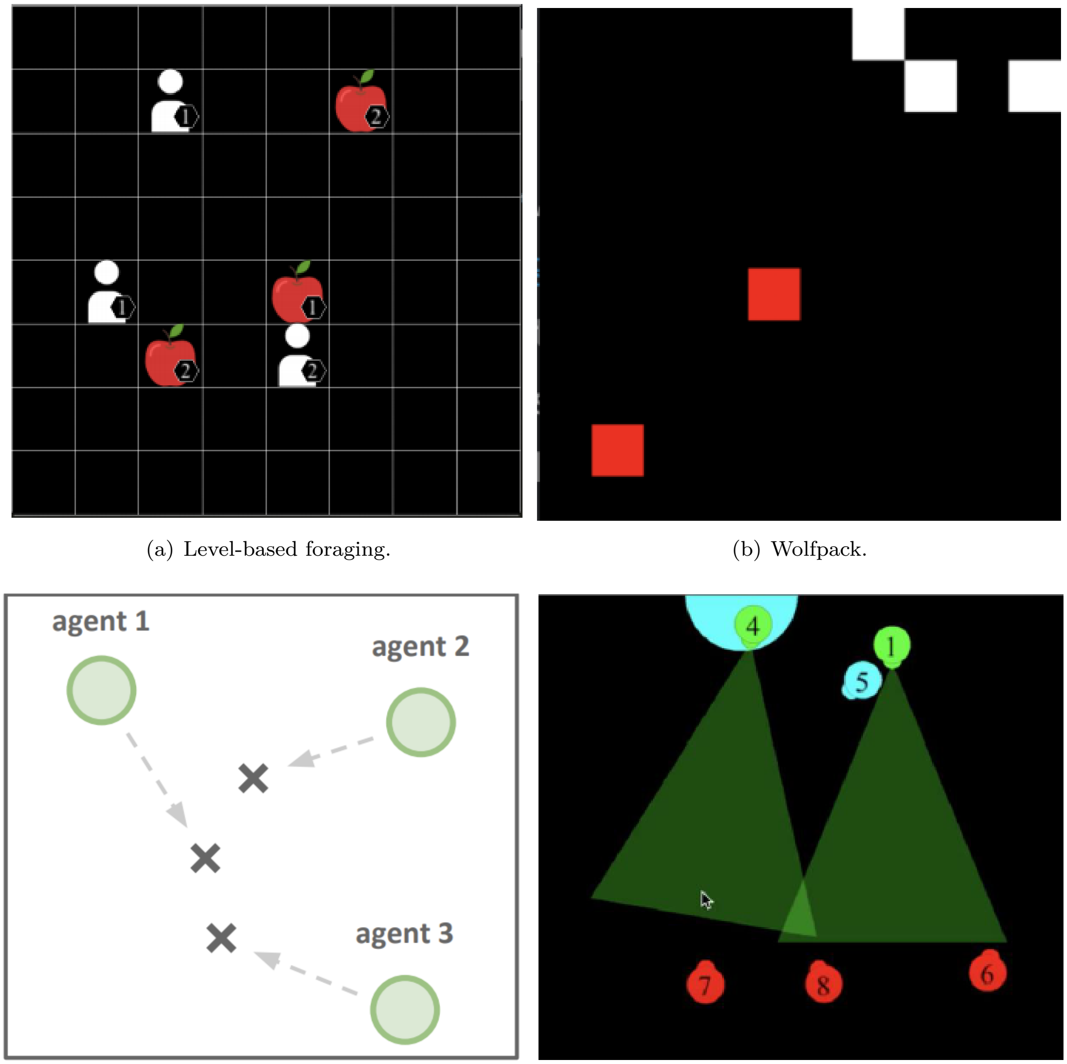
Require: Batch of training teammates' behavioural policies $\{\pi_j^{-i}\}_{k=1}^{K}$; learning rate α; scaling factor λ; concentration param α, initial clustering number $M = 1$.

Initialize the replay buffer D
while *env* is not done **do** `// Generate data`
  for k =1, . . . , K **do**
    Sample data $D_k = \{(s_t, a_t^i, a_t^{-i}, r_t)\}_{t=1,\cdots,T}$ using π^i and π^{-i}.
    Sample small batch of trajectories τ_k.
    Update $E_{\omega_1}, D_{\omega_2}$ by minimizing $\mathcal{L}_C$.
    Extract the feature of the trajectory τ_k by $E_{\omega_1}(\tau_k)$.
    Calculate the CRP prior probability $P(v_k^m)$.
    Calculate the predictive likelihood $P(\tau_k^s|\tau_k^s; v_k^m)$.
    $m = \arg\max_m P(v_k^m)P(\tau_k^a|\tau_k^s; v_k^m)$.
    if $m \leq M$ **then**
      Assign the k^{th} teammate to the m cluster.
      Update the cluster center $\bar{v}^m = \frac{n^m \bar{v}^m + v_k}{n^m + 1}$.
      Update the number of cluster m: $n^m = n^m + 1$.
    else if
      then Update $M = M + 1$ and $n^{M+1} = 1$.
    end if
    $k \leftarrow k + 1$
  end for
  Add $\{K, P(v_k^m)\}$, D_k into $\mathcal{D}$.
  for steps in training steps **do**
    Sample one trajectory $D \sim \mathcal{D}$
    for $t = 1, \cdots, T - 1$ **do**
      Compute $(\mu_{c_t^i}, \sigma_{c_t^i}) = f(s_t, a_t^{-i}))$ and sample c_t^i.
      Compute $(\mu_{z_t^i}, \sigma_{z_t^i}) = f^*(B_t^i)$ and sample z_t^i.
      Compute C_t^i, Z_t^i using HAN.
      Computer $\mathcal{L}_Q, \mathcal{L}_{MI}$ and $\mathcal{L}_C$.
      Update G and global model by minimizing $\mathcal{L}_Q$.
      Update local model by minimizing $\mathcal{L}_{MI}$.
    end for
  end for
end while

coordinates of the prey k if observed. Agents in this environment have five actions, i.e. [up, down, left, right, no-op]. Every hunter in a pack that captured prey is given a reward of two times the size of the capturing pack. We penalize agents by -0.5 for positioning themselves next to prey without teammates positioned in other adjacent grids from the prey.

Setup for Penalized Cooperative Navigation (PCN): In PCN, multiple players must cooperate through physical actions to reach a set of landmarks as shown in Figure A.1(c). The agents need to learn to infer the landmark they must cover, and move there while avoiding other agents. The action space of each agent contains five discrete movement actions, i.e. [up, down, left, right, no-op]. Their objective is to simultaneously cover two destination grids to get a reward of 1. However, a penalty of -0.2 is imposed on the learner if it reaches a destination without the presence of teammates simultaneously covering the other destination. Similar to LBF, the learner can only see the destination grids or teammates if they are inside a 5×5 region surrounding the learner.

Setup for Fortattack: The FortAttack environment defines a bounded two-dimensional space where agents are constrained within specific coordinate ranges. In Figure A.1 (d), the horizontal axis is restricted to values between -0.8 and 0.8, while the vertical axis spans from -1 to 1. The fort is modeled as a semicircle with a center at $(0.0, 1.0)$ and a radius of 0.3. During initialization, the attackers are positioned with their vertical coordinates set between -1 and -0.8, ensuring they begin at a distance from the fort. This positioning allows the attackers to start from a location far enough to strategize their approach. Conversely, the defender's initial position is constrained to the vertical coordinate range between 0.8 and 1, placing them closer to the fort, thus providing an advantageous starting position for defense.



(a) Level-based foraging.

(b) Wolfpack.

(c) Penalized cooperative navigation.

(d) Fortattack.

Figure A.1: State visualization of benchmark environments.

A.2.2 Teammate Types

In open ad hoc teamwork, the primary challenge lies in the fact that the controlled agent lacks prior knowledge of its prospective teammate to interact with. Hence, we randomly choose multiple teammate types to contemplate all potentialities in simulating an open ad hoc teamwork environment where an agent may need to cooperate with unfamiliar counterparts. Therefore, in all environments, each type of teammate policy is implemented either via different heuristics or reinforcement learning-based policies. We vary the teammate's policies in terms of their efficiency in executing a task and their roles in a team. To create a diverse set of teammates for open ad hoc teamwork, we used the following mixture of heuristics along with RL-based models to control teammates during the training process. Our method aims to learn teamwork situations to cooperate with previously unknown teammates' behaviours. To investigate the policy transfer ability and generalization of the proposed methods, we evaluate the method by comparing the performance under different teammate types.

For each environment, we design 20 different policies, and we have one of 10 policies as the training set and the other 10 policies as the testing set. Specificity, in the Wolfpack environment, we used the 9 heuristic policies proposed by Barrett et al. (2011) along with an A2C algorithm Mnih et al. (2015) to control teammates. Similar to Wolfpack, we also create a diverse set of teammate types for LBF which requires agents to adapt their policies towards their teammates to achieve optimal performance. With LBF, we use 8 heuristics Albrecht & Stone (2019); Albrecht & Ramamoorthy (2015), an A2C algorithm Mnih et al. (2015) and DQN Mnih et al. (2016) as our teammate policies. During the testing, for Wolfpack and LBF, we use 10 RL-based policies, which were implemented by 5 different MARL-based to develop several teams of agents. To ensure diversity, we use different random seeds for each RL-based algorithm and save the corresponding models at 2 different checkpoints. On the other hand, for PCN, we utilize 5 different MARL algorithms, to develop 15 different policies showing distinct policy representations from all developed models. In the training, we adopt 5 heuristics Rahman et al. (2022) and random sample 5 different policies to control the teammates. The other 10 teammate policies as the testing set.

A.2.3 Environment Openness

We create an open process that determines how agents enter and leave during episodes. The number of timesteps an agent can exist within the environment is determined by sampling from a certain range of integers. After staying for the predetermined number of timesteps, agents are removed from the environment. After that, agents can reenter the environment after a specific period of waiting time. The type of an agent entering an environment is uniformly sampled from all available types.

In all environments, a teammate only exists in the environment for a certain number of timesteps. If a teammate has surpassed its designated temporal lifespan, it is immediately removed from the environment. A teammate that has been removed is allocated a waiting period, which is the duration before it is pushed into a reentry queue. Given a non-empty reentry queue, agents within the queue are reintroduced to the environment if the number of agents does not exceed the aforementioned upper limit. It is important to note that the reentry queue is randomized, thus inducing an aleatory team composition during learning. The maximum number of players for training is 3 and the maximum number of players for training is 5. For Wolfpack, teammates' lifetime is sampled uniformly between 25 and 35 timesteps while the waiting period is sampled uniformly between 15 and 25 timesteps. By contrast, in the LBF and Penalized cooperative navigation environment, the teammates' lifetime is sampled uniformly between 15 to 25 timesteps while the waiting period is sampled uniformly between 10 and 20 timesteps.

A.2.4 Baselines

MADDPG is a benchmark of MARL algorithm for closed environments. We employ MAPPDG to control the ad hoc agents, using the learner's observations as input to the policy. GPL and VAE-GPL are two type-based methods designed for addressing open ad hoc teamwork problems. Specifically, GPL is suited for scenarios with full observability, while VAE-GPL is applicable in settings with partial observability. Although GPL assumes full observability of the state, we apply GPL in our experiments by treating the learner's observations as input. ODITS improves zero-shot coordination performance in an end-to-end fashion. Two variational encoders are adopted to improve the coordination capability. As ODITS considers only a single ad hoc agent in the closed environment, we further extend the ODTIS to open environments by introducing a graph neural network. Moreover, to demonstrate the advantages of HAN, we also include a comparison with the Open-OTAF(GNN) algorithm, where GNNs are employed to produce the representations of teamwork situations.

A.3 Algorithm configuration

For the LBF, wolfpack, PCN and Fortattack, the training steps T are 640, 0000 and we evaluate all the methods with 100 test episodes after every 3200 training steps. The parameters of the networks are updated by `Adam` optimizer Kingma & Ba (2015) with a learning rate $1e-4$ for all environments. The discounting factor is $\gamma = 0.99$, and the concentration hyperparameter is $\alpha = 1$. Since they induce the best performance compared with other values. For exploration, we use ϵ greedy from 1.0 to 0.05. Batches of 128 episodes are sampled from the replay buffer, and all components in the framework are trained together in an end-to-end

Table A.1: The comparison between different algorithms.

Method	GNN	VAE	AM	MI	CRP	HAN
MADDPG						
GPL	√		√			
VAE-GPL	√	√	√			
ODTIS	√	√		√		
Open-OTAF(GNN)	√	√	√	√	√	
Open-OTAF		√	√	√	√	√

Table A.2: Hyperparameters in the experiments

Notation	**Meaning**
T	640,0000
α	1
γ	0.99
Optimizer	Adam
Batch size	128
Learning rate	$1e-4$
Evaluation interval	3200
Action selector	ϵ greedy
ϵ start	1
ϵ finish	0.05
Replay memory size	5000

fashion. More details of the hyper-parameters are provided in Table A.2. All experiments are carried out in a machine with Intel Core i9-10940X CPU and a single Nvidia GeForce 2080Ti GPU.

In the CRP-based teammate cluster model, we introduce an encoder E_{ω_1} to model the teammate trajectory into a latent space as their behaviour type, where E_{ω_1} is implemented by a long short-term memory (LSTM) network which takes τ_t as inputs and outputs 16-dimensional behavioural embeddings v. To predict the likelihood of the action, we use an RNN-based decoder D_{ω_2} that consists of a GRU cell whose hidden dimension is 16, takes τ_t^s and v as input and reconstructs the action a_t. we set the maximum of cluster number as 10 in all environments.

In the global teamwork situation inference model, the latent variable inference network is implemented as a 2-layer MLP and LSTM which receives the global trajectory as input. More details are provided in Figure A.2. It subsequently produces the mean and covariance matrix for the variational parametric distribution. Assuming that h_t and e_t are the hidden and cell states of the LSTM at timestep t, the LSTM updates the vectors following:

$$\mu_{c_t^i} = MLP_{\alpha^{\mu}}(e_t), \sigma_{c_t^i} = MLP_{\alpha^{\sigma}}(e_t), \tag{12}$$

where $e_t, h_t = LSTM_{\alpha}(H_t, e_{t-1}, h_{t-1})$ and $H_t = [s_t, a_t^{-i}, \{k, p(v_k^m|\tau_k)\}]$. The latent variable c_t vector will be sampled from the distribution $c_t \sim \mathcal{N}(\mu_{c_t}, \sigma_{c_t})$. Then we use the inherent ability of graph neural network to transform the dynamic-size team's information into a fixed-size output. As for the local teamwork situation inference model, it receives the ad hoc agent's preprocessed observation as input and outputs z_t.

To capture the heterogeneity among teammates and handle dynamic team sizes, we model the team as a heterogeneous graph and introduce a HAN to compute the final teamwork situation embedding. The global teamwork situation embedding c_t^i and predicted teammate cluster information will be concatenated as the node input of the HAN to calculate the fixed embedding C_t^i. To estimate the joint value function, we first map C_i^t into the parameters of G by a hyper-network. Then, G maps the ad hoc agent's utility u_i into the value estimation. This alternative design changes the procedure of information integration.

For the ad hoc agent, we utilize coordination graphs and agent modelling to calculate the marginal utility by considering the effect of other agents as shown in equation 6 (in the main paper). More details are provided in Figure A.3. The single utility of $Q^i_\pi(a^i_t|Z^i_t)$ and pairwise utility $Q^{i,j}_\pi(a^i_t|Z^i_t)$ are implemented as multilayer perceptions (MLPs) parameterised by β and δ respectively. Concretely, given Z^i_t, the agent modelling module computes the likelihood of observed agents' actions q by multilayer perceptrons (MLPs). The agent marginal utility can be calculated by equation 6. After that, the marginal utility μ^i_t will be fed into the mixing network to calculate the joint action function finally. To maximize the mutual information between local and global teamwork situations embedding conditioned on the agent i's local trajectory, a variational distribution network, which is similar to the global inference network, is used to approximate the conditional distribution.

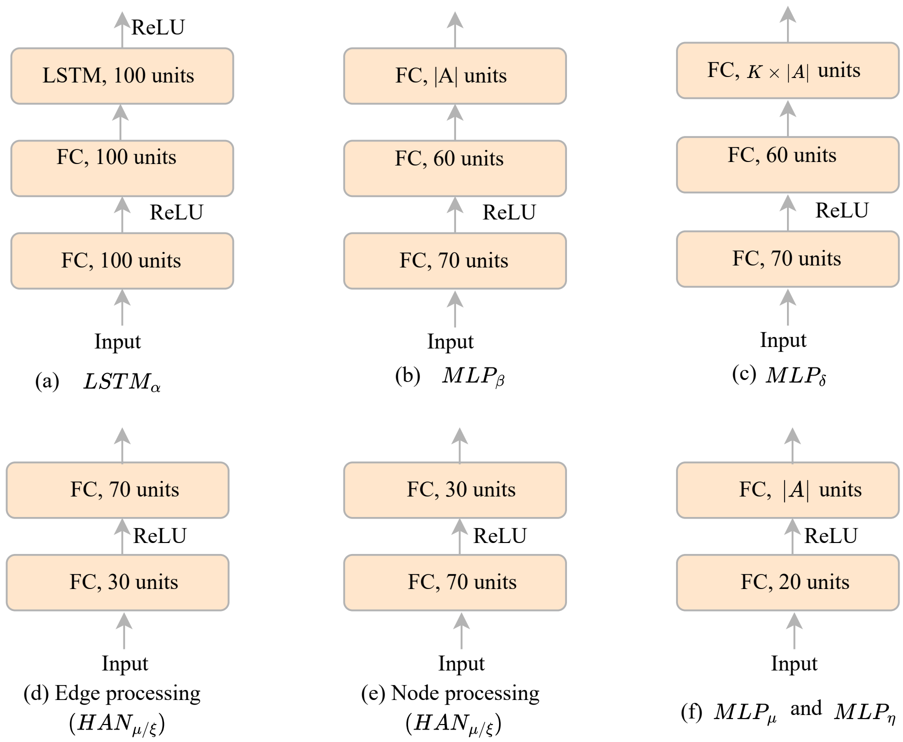
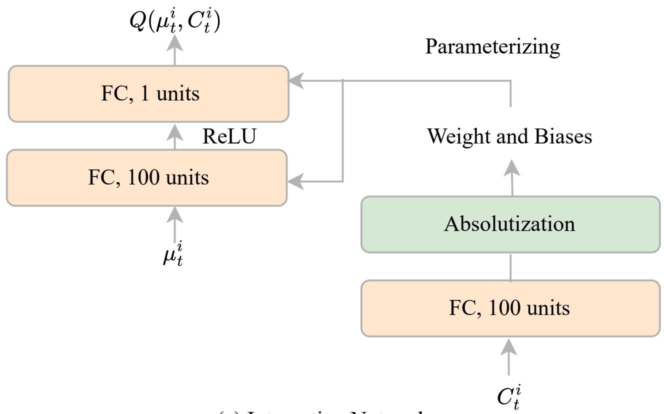


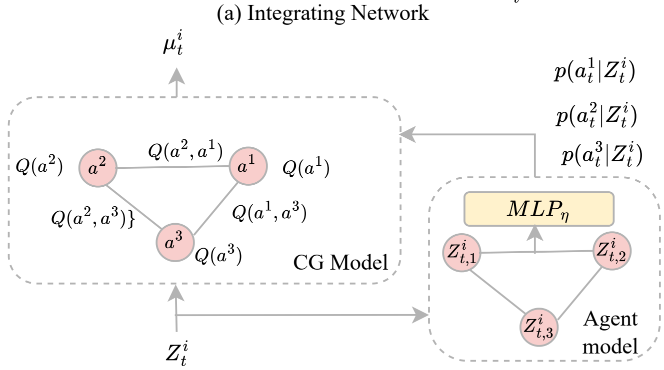
Figure A.2: Architecture details of teamwork situation inference model: (a) the architecture used in teamwork situation inference network; (b) singular utility computation; (c) pairwise utility computation; (d) edge embedding computation in the agent and auxiliary agent model; (e) node embedding computation in the agent and auxiliary agent model; (f) the MLP used by the agent and auxiliary agent model to process the resulting GNN node embeddings. FC denotes a fully connected layer, LSTM denotes an LSTM layer, and the accompanying number denotes the size of the layer. Labels on the arrows indicate the non-linear functions used between the layers while no labels indicate no non-linear functions being applied to the resulting output vectors.

A.4 Experiment Results

To determine the importance of each component of Open-OTAF, we also conducted an ablation study of the different components under different scenarios. The comparison results under Wolfpack, PCN and Fortattack environment are provided in Figure A.4. The results show that the w/o MI variant exhibits the most significant performance degradation across all games, highlighting the importance of incorporating

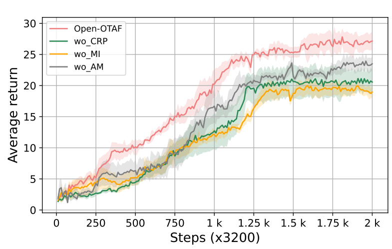


(a) Integrating Network

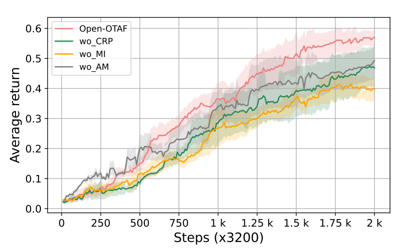


(b) Marginal Utility Network

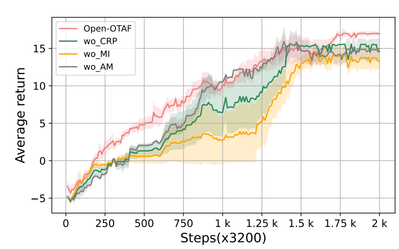
Figure A.3: Architecture details of interacting network and marginal utility network.



(a) Ablation study in Wolfpack.



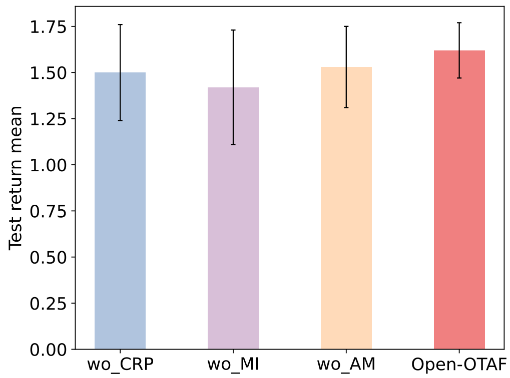
(b) Ablation study of PCN.



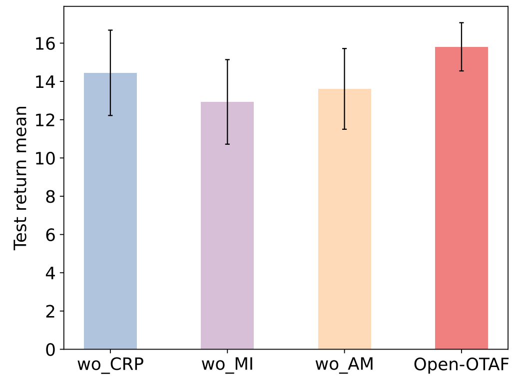
(c) Ablation study of Fortattack.

Figure A.4: Ablations for different components under Wolfpack, PCN and Fortattack environment.

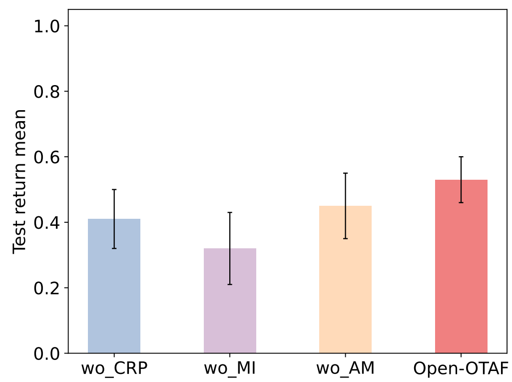
global information to support the learning of the local teamwork situation inference model. Additionally, *w/o* CRP performs worse than Open-ATAF, indicating that the CRP-based teammate model enables the controlled agent to adapt more effectively to dynamic teammates. Moreover, we observe that agent modeling contributes to learning more accurate marginal utilities by accounting for the influence of other teammates, leading to modest improvements in coordination.



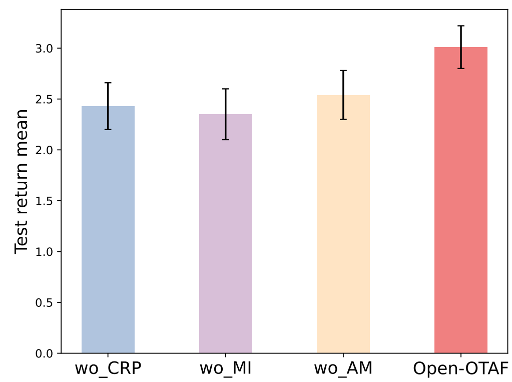
(a) LBF.



(b) Wolfpack.

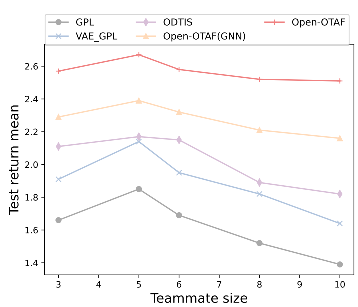


(c) PCN.

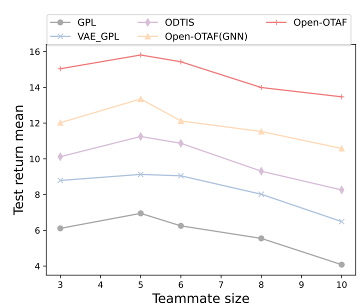


(d) Fortattack.

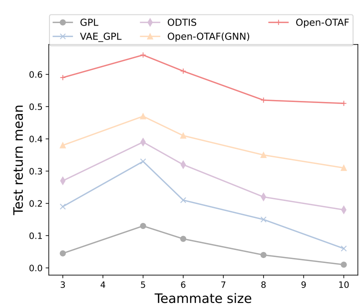
Figure A.5: Testing performance comparison across various scenarios for different components.



(a) LBF.



(b) Wolfpack.



(c) PCN.

Figure A.6: Generalization study with varying teammate size.

The testing results are also provided in Figure A.5, where we provide the average return (bars) and the standard deviation (error bars) over different seeds. The results show that w/o CRP and w/o MI suffer the most severe performance degradation in all environments when compared with Open-OTAF. It shows that the introduction of the CRP model for teammate types classification and the guidance of global information help agents to estimate more accurate teamwork situations.