DiTAC: Discrete Teamwork Abstraction for Ad Hoc Collaboration

Jing Wang[a], Pengjie Gu[b], Mengchen Zhao[c,*], Guangyong Chen[d], Furui Liu[d] and Pheng-Ann Heng[a]

[a]The Chinese University of Hong Kong
[b]Nanyang Technological University
[c]South China University of Technology
[d]Zhejiang University

Abstract. Training autonomous agents to collaborate with unknown teammates in cooperative multi-agent environments remains a fundamental challenge in ad hoc teamwork research. Conventional approaches rely heavily on online interactions with arbitrary teammates under the assumption of full observability. However, in real-world scenarios, teammate policies are often inaccessible, making historical trajectory rollouts a more practical alternative. We propose DiTAC, a method that learns discrete teamwork abstractions for ad hoc collaboration by automatically extracting latent cooperation patterns from short trajectory segments and adapting effectively to diverse teammate behaviors. To mitigate the out-of-distribution challenge, we constrain learned representations within a discrete codebook. Furthermore, we employ a masked bidirectional transformer architecture to infer teammate behaviors from local observations, thereby relaxing the full observability assumption. Empirical results demonstrate that DiTAC significantly outperforms existing baselines and its variants across widely-used ad hoc teamwork tasks.

1 Introduction

The fast growth of autonomous agents, from robots to software, is opening up new chances and challenges in real-life applications. These agents often need to work together on the fly with others they haven't met before [1]. For instance, in disaster response, agents must collaborate rapidly despite unfamiliarity and privacy constraints. In gaming, virtual agents must be able to smoothly help various human players [35, 5, 9, 13], highlighting the need for rapid adaptation. These scenarios underscore the growing importance of ad hoc teamwork in autonomous systems research [26, 2, 17].

Existing research typically tackles this problem by employing an *online* learning paradigm, in which agents are trained with a pre-established set of teammates through environment interactions before they are deployed [1, 10, 20]. However, in numerous situations, this form of online interaction is impractical due to high associated costs and potential risks. These costs often involve extensive simulations or real-world testing, while risks might include damage to expensive equipment or hazards to human safety. Moreover, in complex domains that demand effective generalization, there is often a preference for leveraging large, pre-collected trajectory datasets. In light of these factors, our study addresses a subset of ad hoc teamwork challenges by exploring an *offline* data setting, where agents have access solely to previously collected trajectory data.

Addressing the problem of *offline ad hoc teamwork* poses three primary challenges: *i*) The first challenge lies in creating an offline learning framework that enables agents to generalize across a diverse range of teammates. This involves utilizing a substantial dataset collected beforehand to strengthen the agents' ability to adapt to various team dynamics. *ii*) The second challenge involves enabling agents to infer collaborative strategies from short trajectory segments, which encode recurring teamwork patterns (e.g., synchronized food collection in foraging tasks), even when cooperating with previously unseen teammates. *iii*) Lastly, handling teamwork situations that are not represented in the training dataset's distribution presents an additional problem. This out-of-distribution issue often markedly compromises performance in single-agent offline reinforcement learning. In the context of ad hoc teamwork [16, 11, 14], this issue becomes even more complex due to the multi-agent setting.

To tackle offline ad hoc teamwork challenges, we propose DiTAC, a transformer-based framework that extracts discrete teamwork abstractions from short trajectory segments. Unlike prior approaches relying on continuous latent variables [22, 33] or predefined roles [15, 32], DiTAC encodes multi-agent interactions into a finite set of interpretable codes. It leverages masked bidirectional transformers [31] and VQ-VAE [30] to infer latent cooperation patterns from partial trajectories. This grounding enables generalization to unseen teammates by matching observed behaviors to learned patterns, even under partial observability. DiTAC achieves strong generalization in ad hoc teamwork via three key innovations: discrete teamwork abstractions that compactly encode latent cooperation strategies, masked transformers that model partial observability, and decision grounding in discrete latent space for robustness to out-of-distribution scenarios. Extensive experiments validate DiTAC's effectiveness, showing improved generalization and robustness across prompt lengths and codebook sizes.

2 Related Works

2.1 Ad Hoc Teamwork and Trajectory-Based Behavior Modeling

Ad hoc teamwork focuses on enabling agents to collaborate with unknown teammates without prior coordination. Early approaches like

* Corresponding Author. Email: zzmc@scut.edu.cn

PLASTIC [1] defined finite teammate types but struggled in complex settings. Recent methods infer teammate behaviors from interactions: **BRDiv** [24] generates diverse teammate policies for robust training, while **AATEAM** [7] uses attention to model teammate types from states. However, these methods require *online interactions* and full observability. To address partial observability, **LIAM** [22] estimates teammate actions from local history, and **ODITS** [12] infers teamwork variables via information regularization. Unlike these works, we focus on *offline* ad hoc teamwork, where agents learn collaborative strategies from pre-collected trajectory segments. This aligns with hierarchical RL frameworks like **Option Discovery** [28], where sub-trajectories represent reusable skills. However, prior work lacks explicit modeling of *team-level* strategies (e.g., synchronized actions), which our method achieves through trajectory segmentation and discrete codebooks.

2.2 *Offline Multi-Agent Reinforcement Learning*

Offline MARL trains policies using pre-collected datasets to avoid risky online interactions. Methods like **BCQ** [11] and **CQL** [14] address distributional shifts in single-agent settings. In MARL, **MADT** [19] uses transformers for offline coordination but assumes known teammates during training. **RODE** [32] learns role embeddings from trajectories but requires predefined roles, limiting adaptability. Recent studies demonstrate that transformers [31] are capable of modeling complex, high-dimensional distributions of semantic concepts on a large scale. This includes achieving effective zero-shot generalization in language tasks [3] and generating images that are out-of-distribution [25]. And it is recently applied to solve RL problems for its efficiency and scalability when modeling long decision making. Decision Transformer [6] for offline RL treats learning a policy as a sequential modeling problem, which bridges sequence modeling and transformers with RL. Furthermore, instead of using prompts to extract knowledge from the pretrained model in NLP, the RL agent in Prompt-DT [34] is required to imitate the provided trajectory prompts to reproduce the policy that generates these trajectories. Our work bridges this gap by recasting offline ad hoc teamwork as *conditional sequence modeling*, inspired by **Decision Transformer** [6] and **Prompt-DT** [34]. Unlike these single-agent methods, we employ masked transformers to process partial trajectory segments and infer teamwork abstractions. This approach generalizes to unseen teammates without role predefinition, a key limitation of [32].

2.3 *Discrete Representation Learning for Collaboration*

Discrete latent spaces improve robustness to distribution shifts in RL. **VQ-VAE** [30] compresses continuous data into interpretable codes, while **MaskDP** [18] uses masking for scalable decision-making. In MARL, **LIAM** [22] models teammate behaviors as latent variables but lacks explicit strategy-level abstraction. Recent work explores discrete representations for multi-agent coordination: **Diverse Skill Discovery** [29] generates diverse agent behaviors via reward randomization, while **RODE** [32] decomposes tasks into role embeddings. Our method uniquely combines these ideas: we use VQ-VAE to map *team trajectory segments* to discrete codes representing teamwork situation. This approach avoids the combinatorial complexity of coordination graphs while grounding abstraction in observable trajectories, addressing partial observability and OOD challenges.

3 Preliminaries

We aim to train an offline robust *ad hoc* agent that could effectively interact with unknown teammates under partial observability without further joint learning.

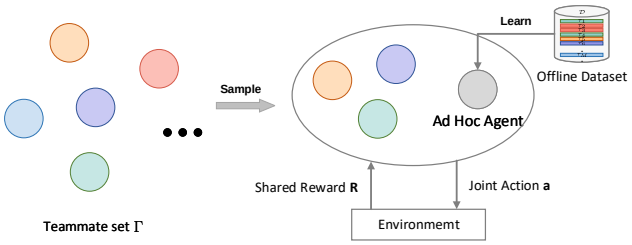

Figure 1: Illustration of the cooperation process among the ad hoc agent with random teammates.

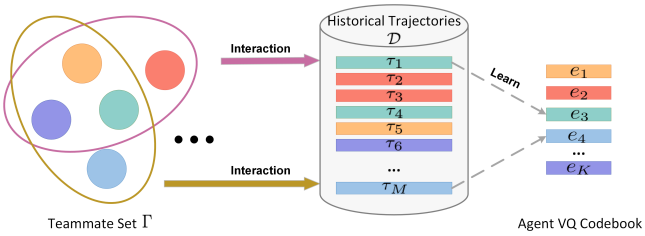

Figure 2: Illustration of learning discrete agent codebook from offline dataset.

3.1 *Dec-POMDP with an additional teammate set*

We model the problem as a decentralized Partially Observable Markov Decision Process (Dec-POMDP) [21] with an additional set of teammates' possible policies Γ [12], which can be represented as a tuple $\langle \mathcal{N}, \mathcal{S}, \mathcal{A}, \mathcal{O}, P, R, O, \Gamma \rangle$. It consists of a set of agents $\mathcal{N} = \{1, 2, \ldots, n\}$, a finite global states set $\mathcal{S}$, the joint action space $\mathcal{A} = \times_i \mathcal{A}_i$, where $a_i \in \mathcal{A}_i$ is the independent action of agent i, and the joint observation space $\mathcal{O} = \times_i \mathcal{O}_i$, where $o_i \in \mathcal{O}_i$ denotes the partial observation of each agent i can fetch in the partially observable environments. O denotes the observation function that is used to derive the joint observation $\boldsymbol{o} = O(\boldsymbol{s}, i)$, $P(\boldsymbol{s}', | \boldsymbol{s}, \boldsymbol{a})$ denotes the probability of state $\boldsymbol{s}'$ transited from the state $\boldsymbol{s}$ by taking the joint action $\boldsymbol{a}$, $R(\boldsymbol{s}, \boldsymbol{a})$ is the reward function that maps a state $\boldsymbol{s}$ and a joint action $\boldsymbol{a}$ to a immediate team reward $r \in \mathbb{R}$.

We denote the ad hoc agent under our control as the agent i, and all other agents as $-i$. Without loss of generality, the policy of the ad hoc agent is denoted by π_i, and the joint policy of other agents is denoted by π_{-i}. Γ includes a set of pretrained or predefined policies for teammate agents $-i$ to adopt for cooperation. We illustrate the problem in the Fig. 1, some arbitrary agents are sampled from unknown teammate set Γ. At each timestep t, all agents receive their partial information, take the joint action $\boldsymbol{a} = (a_i, a_{-i})$ where $a_i \sim \pi_i, a_{-i} \sim \pi_{-i}$, and receive a team-shared reward r. The goal of the ad hoc agent is to maximize the cumulative team-shared reward $\mathbb{E}_{\pi_i, \pi_{-i}} \left[\sum_{t=0}^{H-1} \gamma^t r_{t+1}^1 \right]$ when cooperating with $N-1$ arbitrary agents π_{-i} sampled from Γ, where γ is the discounted factor.

3.2 *Transformer-based Offline RL*

In previous works, ad hoc agent is trained online, where it cooperates with some pretrained teammates to iteratively take joint actions and receive feedback from the environment. However, this may

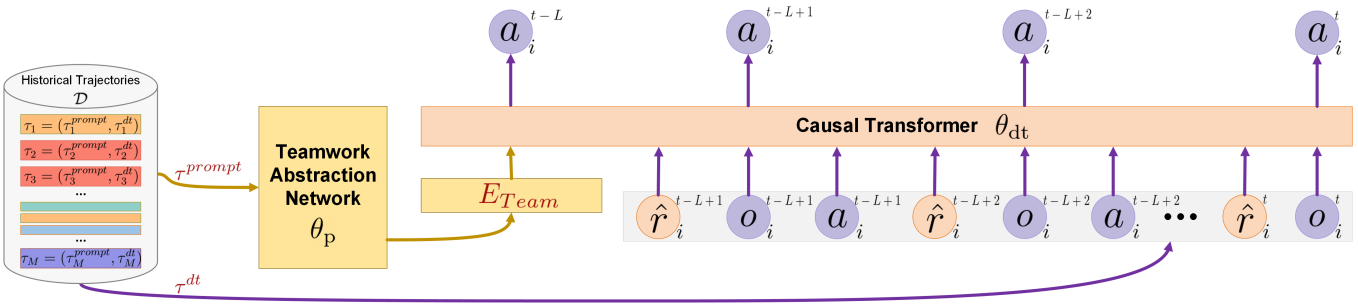

Figure 3: Framework of learning **Di**screte **T**eamwork abstraction for **A**d hoc **C**ollaboration (DiTAC). Trajectory segments τ^{prompt} are processed to extract teamwork situation abstractions, which are discretized into a codebook of reusable strategies.

not always be feasible as the high cost of collecting diverse pre-trained teammates and RL algorithms may require a large number of training data due to low sample efficiency. We consider the offline RL setting [16], which aims to learn a policy from an offline dataset $\mathcal{D}$, where $\mathcal{D}$ owns a large number of trajectories that are pre-collected using some unknown joint behavior policies. A trajectory $\tau = \{\hat{r}^1, o_1^1, a_1^1, o_2^1, o_2^1, \ldots, \hat{r}^t \ldots o_{n-1}^t, a_{n-1}^t, o_n^t, o_n^t\}$ is sampled using a joint behavior policy in the environment, where $\hat{r}$ is the cumulative rewards (reward-to-go) from the current time step till the end of the episode. When training with offline collected data, $\hat{r}^t = \sum_{i=t}^{T} r^i$. During testing, $\hat{r}^t = R^\star - \sum_{i=0}^{t} r^i$ where $R^\star$ is the targeted total return for an episode. The ad hoc agent is expected to find the optimal policy using only the dataset $\mathcal{D}$ without interacting with any teammates in the cooperative environment.

4 Method

In this section, we introduce our proposed DiTAC model that automatically learns a discrete latent agent codebook and conditions only on partial observations for cooperating with unknown teammates to maximize the team-shared reward.

4.1 Overview

As shown in the Figure 2, we aim to learn a robust policy from an offline dataset $\mathcal{D}$, which contains a large number of trajectories pre-collected with some unknown teammate behavior policy $\pi \sim \Gamma_{train}$. Our approach models teamwork abstraction from short trajectory segments that encode recurring multi-agent behaviors. DiTAC captures teamwork abstraction from historical interactions to guide policy generation for zero-shot coordination.

4.2 Framework Details and Data Flow

Figure 3 illustrates the end-to-end architecture of DiTAC, which integrates a teamwork abstraction network θ_p and a causal transformer-based decision module θ_{dt}. The teamwork encoder processes masked trajectory segments τ^{prompt} to infer latent teamwork abstractions. Specifically, as presented in Figure 4, the encoder receives a segment of L^* steps of team interactions, where a subset of agents' observations (o) and actions (a) are randomly masked (e.g., replaced with zeros) in training stage. This masking simulates partial observability and forces the model to infer missing teammates' behaviors from contextual information. The encoder's bidirectional attention mechanism enables it to capture dependencies across all timesteps and agents within τ^{prompt}, generating a continuous latent representation z_i for each agent. These representations are then discretized via a vector-quantized codebook $\mathbb{E}$ to produce E_i, which encodes distinct agent behavior types. The teamwork abstraction E_{Team}, derived by averaging all E_i, is concatenated with the ad hoc agent's recent trajectory τ^{dt} and fed into the decision module. The causal transformer in θ_{dt} autoregressively predicts actions based on this combined input, guided by the inferred teamwork dynamics. This modular design ensures that DiTAC generalizes to unseen teammates by grounding cooperation strategies in discrete abstractions.

The input τ^{input} is a segment cropped from an episode trajectory τ and then separated into two parts to feed into the prompt and causal transformer respectively. At every time step, DiTAC takes τ^{input} as input, which contains the most recent $L^\star + L$ step trajectory information. In details, $L^\star$ steps teamwork interactions

$$\tau^{\text{prompt}} = \{\hat{r}^{\triangle+1}, o_1^{\triangle+1}, a_1^{\triangle+1}, o_2^{\triangle+1}, o_2^{\triangle+1}, \ldots, \hat{r}^{L^{\triangle+L^\star}} \ldots o_{n-1}^{\triangle+L^\star}, a_{n-1}^{\triangle+L^\star}, o_n^{\triangle+L^\star}, o_n^{\triangle+L^\star}\}$$

are fed into the prompt structure, and L steps ad hoc agent i personal trajectory

$$\tau^{\text{dt}} = \{\hat{r}^{t-L^\star+1}, o_i^{t-L^\star+1}, a_i^{t-L^\star+1}, \ldots, \hat{r}^{t-L^\star+L} o_i^{t-L^\star+L}, a_i^{t-L^\star+L}\}$$

are injected into the causal transformer structure, where $\triangle = (L^\star + L)$. The trajectory prompt τ^{prompt} contains the short trajectory segment, aiming to capture critical teamwork situation abstractions. For example, a 5-step segment might encode a 'flanking prey' strategy in predator-prey tasks. The trajectory input τ^{dt} of the specific ad hoc agent is utilized to model the sequential decision-making condition on its partial observation along with the teamwork abstraction. Then the two modules are jointly trained with reconstruction loss and action prediction loss. Then the two modules can be jointly trained with a reconstruction loss $\mathcal{L}_{\text{rec}}$ among prompt information and an action prediction loss $\mathcal{L}_{\text{CE}}$ among the partial observation.

4.3 Learning Teamwork Abstraction

We aim to learn the teamwork abstraction from the L-steps before interactions τ^{prompt} among all agents, up to horizon length $L^\star$, which encode collaborative strategies. These segments are compressed into discrete codes via a codebook, where each code represents a reusable teamwork pattern. We assume the existence of some discrete latent codebook that represents finite agent's behavior type, and at each time step, the latent embedding E_{Agent} contains information about the current environment dynamics perceived by the agent and the cooperation strategy with teammates. Hence, we employ the vector-quantization method VQ-VAE [30] to learn discrete abstractions of agent behavior type along with a masked bi-directional transformer.

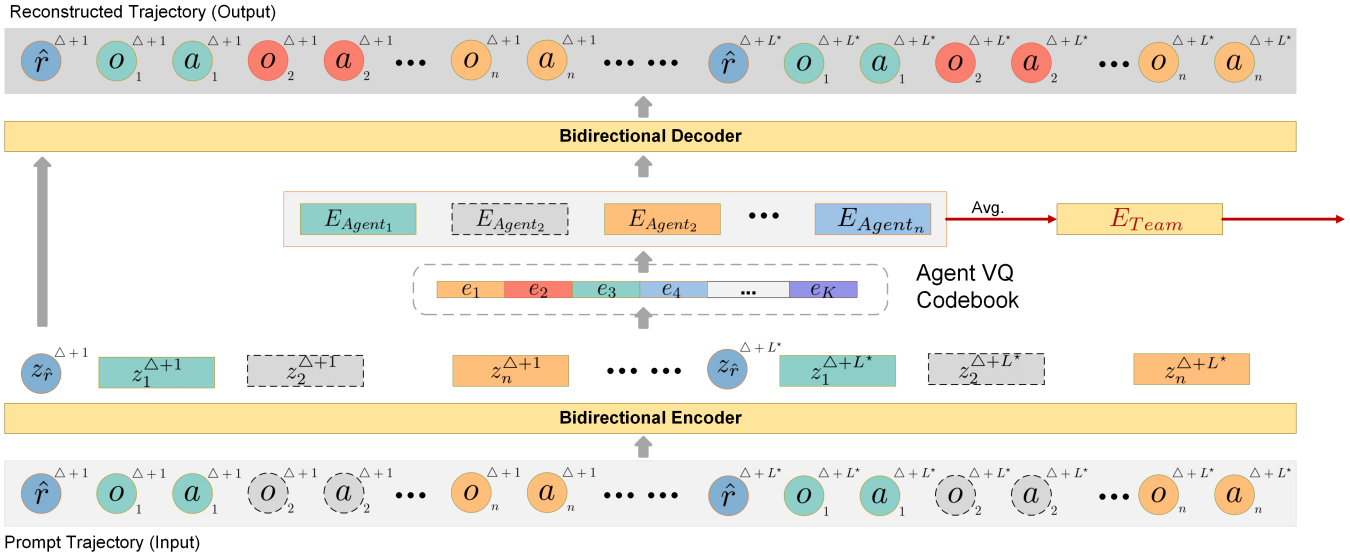

Figure 4: Details of learning discrete teamwork abstraction from bidirectional transformers via codebook integration and masking mechanism. To simplify the notation, we denote $t-(L^\star+L)$ by $\triangle$, which stands for the last timestep before the $(L^\star+L)$-step sampled trajectory segment respect to current timestep t. The grey shapes with dashed border represents the randomly masked agents' information. We aim to learn extract the discrete teamwork abstraction E_{Team} via this module and then inject it into the following causal transformer for decision making.

Algorithm 1 DiTAC Training

Require: Masked Bi-Directional Transformer θ_p, causal transformer θ_{dt}, overall network architecture θ, agent number N, offline dataset $\mathcal{D}$, batch size M, learning rate α.

for step in training steps **do**
  for $m=1$ **to** M **do**
    Sample a trajectory τ_m of length $L^\star+L$ from $\mathcal{D}$
    Process the trajectory τ_m into τ_m^{prompt} and τ_m^{dt} segments
  end for
  Get a batch $\mathcal{B}=\left\{\tau_m^{\text{prompt}},\tau_m^{dt}\right\}_{m=1}^{M}$
  $\tau_{rec}, E_{\text{Team}}=\theta_p\left(\tau^{\text{prompt}}\right), \forall \tau^{\text{prompt}}\in\mathcal{B}$
  $a^{\text{pred}}=\theta_{dt}\left(E_{\text{Team}},\tau^{\text{input}}\right), \forall \tau^{\text{input}}\in\mathcal{B}$
  Compute $\mathcal{L}_{rec},\mathcal{L}_{VQ-VAE},\mathcal{L}_{CE}$
  $\theta_p \leftarrow \theta_p-\alpha\nabla_{\theta_p}\left(\mathcal{L}_{rec}+\mathcal{L}_{VQ-VAE}\right)$
  $\theta \leftarrow \theta-\alpha\nabla_{\theta}\mathcal{L}_{CE}$
end for

During the training stage, we randomly mask some agents' information of the prompt input τ^{prompt}. For example,

$$masked(\tau^{\text{prompt}})=\{r^{\triangle+1},o_1^{\triangle+1},a_1^{\triangle+1},\_,\_,\ldots,$$
$$r^{L^{\triangle+L^\star}}\ldots o_{n-1}^{\triangle+L^\star},a_{n-1}^{\triangle+L^\star},o_n^{\triangle+L^\star},o_n^{\triangle+L^\star}\}$$

denotes masked $Agent_2$ information from τ^{prompt}. Masked inputs are processed by a bidirectional transformer to generate latent embeddings z that is fused with teammate information.

To capture each agent behavior type within $L^\star$ steps, we maintain a codebook $\mathbb{E}\subset\mathbb{R}^{K\times D}$, where K represents the size of the discrete latent space, and D denotes the dimension of the latent embedding vector. As shown in Figure 4, we discretize the latent embedding information of the $i-th$ agent $z_i=(z_i^{\triangle+1},z_i^{\triangle+2},\ldots,z_1^{\triangle+L^\star})\in\mathbb{R}^D$ into E_i by looking up the nearest neighbor embedding from the codebook:

$$E_i=\text{DISCRETIZE}(z_i)=e_j,$$

where $j=\arg\min_j\|z_i-e_j\|_2, e_j\in\mathbb{E}$. We feed all discrete agent latent embeddings E_i to the bidirectional decoder for further processing, which specifically involves reconstructing the teamwork trajectory τ^{prompt}. Through this process, we learn the discretized vector codebook that includes finite agent behavior types in Eq.(1) and Eq.(2):

$$\mathcal{L}_{rec}(\theta_p)=\mathbb{E}_{\tau^{\text{prompt}}\sim\mathcal{D}}\left(\tau^{\text{prompt}}-\tau_{rec}\right)^2, \tag{1}$$

where τ_{rec} is the reconstructed prompt trajectory.

$$\mathcal{L}_{\text{VQ-VAE}}(\theta_p)=\mathbb{E}_{\tau^{\text{prompt}}\sim\mathcal{D}}\left[\frac{1}{N}\sum_{i=1}^{N}\|\text{sg}(z_i)-E_i\|_2^2+\frac{\beta}{N}\sum_{i=1}^{N}\|z_i-\text{sg}(E_i)\|_2^2\right] \tag{2}$$

The function $\text{sg}(\cdot)$ refers to a stop-gradient operation that blocks gradients from flowing into the corresponding argument. The first term and second term in Eq.(2) is the codebook loss and commitment loss respectively, which encourages the discretized E_i and z_i to be mutually close to each other. The hyperparameter β controls the extent to which the code can change, and we set the value of β to 0.25 as the original VQ-VAE algorithm.

Codebook Integration and Masking Mechanism The discretization step (Figure 4, middle) plays a critical role in enforcing generalization. This process compresses diverse teammate behaviors into a finite set of discrete codes, mitigating out-of-distribution issues during inference. During training, random masking is applied to τ^{prompt} to simulate scenarios where the ad hoc agent lacks full observability (e.g., in disaster zones or competitive games). For instance, if Agent_2's observations and actions are masked, the encoder must reconstruct its behavior using correlations learned from other agents' unmasked data. The bidirectional transformer's self-attention layers enable cross-agent reasoning, allowing the model to fill in missing information by attending to contextual patterns. The codebook loss $\mathcal{L}_{\text{VQ-VAE}}$ further regularizes the latent space by aligning z_i with discrete codes, ensuring that similar teamwork scenarios map to neighboring embeddings. This combination of masking and discretization ensures robustness to both partial observability and novel teammates.

Algorithm 2 DiTAC Testing

Require: Testing with teammates' behavioral policies $\pi_{-i} \sim \Gamma_{\textbf{Eval}}$.

Initialize trajectory segment τ with zeros, desired reward-to-go $\hat{r} = R^{\star}$
$t = 0$
while not done **do**
  if step $t > L^{\star} + L$ **then**
    Construct τ^{prompt} with $\tau_{t-(L^{\star}+L)} \sim \tau_{t-L}$ {Mask all teammates information with zero except the adhoc agent}
  else
    Construct τ^{prompt} with $\tau_0 \sim \tau_{L^{\star}}$
  end if
  Construct τ^{dt}
  Compute E_{Team} with $\theta_p(\tau^{\text{prompt}})$
  Compute a^{pred} with $\theta(E_{\text{Team}}, \tau^{\text{input}})$
  Step the joint action $(a^{\text{pred}}, a_{-i})$ { $a_{-i} \sim \pi_{-i}$ }
  Receive feedback from env: $o, a, r, \hat{r} \leftarrow \hat{r} - r$
  Append $[\hat{r}, o, a]$ to recent history τ
  $t \mathrel{+}= 1$
end while

We then average all agent embeddings to represent teamwork abstraction E_{Team} and pass them to the following causal transformer structure for further action prediction. In the evaluation stage, the masked bi-directional transformer structure allows us to infer the teamwork abstraction condition on only partial observations of the ad hoc agents directly, by applying masking patterns for the input requiring other teammates' information. Then the teamwork abstraction vector is injected with τ^{dt} into the causal transformer structure to guide the sequential decision-making.

4.4 Algorithms

DiTAC integrates two core modules: a bi-directional transformer-based teamwork abstraction network θ_p and a causal transformer-based decision module θ_{dt}. The ad hoc agent's policy π_i is instantiated via θ_{dt}, which generates actions autoregressively based on its recent trajectory τ^{dt} and a latent teamwork abstraction:

$$a_i \sim \pi_i(\tau^{\text{dt}}, E_{\text{Team}}; \theta_{\text{dt}}) \tag{3}$$

The training and inference procedures are summarized in Algorithm 1 and Algorithm 2, respectively. During training, the encoder θ_p receives fully observable trajectory segments τ^{prompt} with randomly masked agents to learn robust team representations. The decoder reconstructs the original input, optimized via a reconstruction loss $\mathcal{L}_{\text{rec}}$ and a vector quantization loss $\mathcal{L}_{\text{VQ-VAE}}$. Simultaneously, the decision module θ_{dt} is trained to predict actions using cross-entropy loss $\mathcal{L}_{\text{CE}}$:

$$\mathcal{L}(\theta) = \mathcal{L}_{\text{rec}}(\theta_p) + \mathcal{L}_{\text{VQ-VAE}}(\theta_p) + \mathcal{L}_{\text{CE}}(\theta) \tag{4}$$

At test time, only the ad hoc agent's observations remain unmasked, while teammate data is masked. The encoder infers a latent team abstraction from this partial input, which guides the decision module in generating context-aware actions. This design enables coordination with unseen teammates under partial observability.

5 Experiments

We evaluate DiTAC with evaluation teammate set on three distinct environments in varying agent number settings. We have chosen three multi-agent cooperative environments as benchmarks. We conduct each experiment with 4 different random seeds and then illustrate the average teamwork episode return with a 95% confidence interval over the standard deviation.

5.1 Environments

Level-based Foraging (LBF) [23] We inherit the setting used in the CSP [8] work: a 20×20 grid with 2 agents and 4 food items. Agents perceive a 5×5 local grid and execute discrete actions (move or collect). As modified in CSP, agents need to cooperate with each other to reach around a food and implement the action "collect" simultaneously to successfully capture the food, which is designed to satisfy the requirement for enhancing cooperation among agents. Different teammate policies can be reflected into the food collecting order, the challenge for training a robust ad hoc agent is learning to flexibly cooperate with various teammates to acquire the food.

Predator-Prey (PP) [4] This environment is a more complex 10×10 grid task requiring agents to collaboratively hunt moving prey. Specifically, each prey moves randomly during the game, which brings huge challenge for predators to learn to capture the prey collaboratively. Coordination within this environment is manifested through strategic prey chasing and the distribution of labor. We adopt three different scenarios to verify the ad hoc agent's ability to cooperate with different number of teammates, where xays stands for x predators y preys respectively.

SMAC Fork SMAC [27] is a widely used MARL benchmark that emphasizes coordinated micromanagement. Unlike standard maps emphasizing focus-fire and retreating, we use the custom Fork map [8]. This map features two symmetrical points at the *Up* and *Down* sides, guarded by several enemies. Among 8 agents, 4 are trained ad hoc agents. Teammates must coordinate their attacks on the same point to eliminate enemies and earn high rewards. Otherwise, neither point will achieve a firepower advantage, resulting in failures.

5.2 Offline Datasets

Instead of learning the model through extensive online interactions with various agents, we only have access to some pre-collected trajectories of unknown teammates in offline setting. Consequently, we construct a diverse teammate set comprising various behavioral policies from teammates and divide them for training and evaluation. Teammate policies are generated using CSP algorithm [8] that integrates a Soft-Value Diversity (SVD) objective with an alternating optimization mechanism. This process yields two policy populations, each comprising four distinct policies. Next, we sample corresponding models at three different checkpoints for each policy, resulting in a total of 12 distinct policies per population. We randomly select policies from one population as the training teammate set Γ_{Train} and use the remaining policies as the testing teammate set Γ_{Eval}. All data are derived from the training policy set, while testing is conducted with unseen policies to simulate authentic zero-shot collaboration settings. Detailed dataset statistics are presented in Table 1.

5.3 Evaluation of Performance

Baselines

We evaluate our method against five baseline approaches, which consist of three variants of Decision Transformers and two state-of-the-art online MARL algorithms specifically designed for ad hoc

Table 1: Statistics of collected offline datasets

Environment	#Agents	#Samples	Mean Return	Max Return
Level-Based Foraging	2	60000	0.3752	0.5000
Predator Prey	2	60000	18.4973	40
	3	120000	22.2138	60
	4	120000	51.1635	80
	8	120000	89.4050	120
SMAC Fork	8	120000	3.2576	9.0698

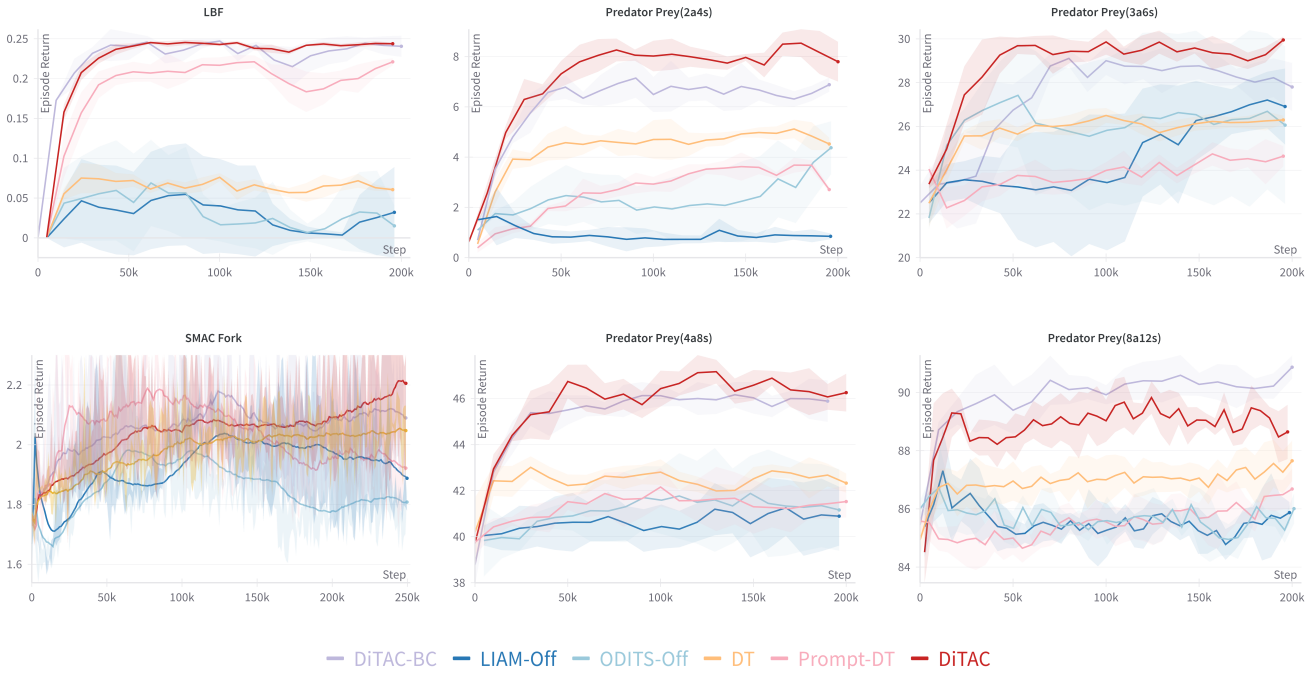

Figure 5: Performance comparison across different scenarios with various agent number settings among three environments.

agent learning. For fairness, all methods were tested in online multi-agent environments with diverse, unknown teammates. The performance trends in Figure 5 illustrate how each method evolves during training. Each data point corresponds to the average online interaction outcomes of the several trained models with unknown teammate policies $\pi_{-i} \sim \Gamma_{Eval}$, plotted against its respective training timestep.

- **Decision Transformer (DT)** [6]. We train a vanilla decision transformer for learning to cooperate with various policies by regarding the ad hoc learning as a normal offline single-agent learning task. The decision transformer has the same structure of the causal transformer we used in DiTAC but only eliminated the prompt structure, and is trained with the same process as the DiTAC but with only τ^{input}, which is designed to help ablate the effect of prompt.
- **Prompt Decision Transformer (Prompt-DT)** [34]. We adopt the Prompt-DT for learning to cooperate with different policies by regarding each cooperation as a single task. The Prompt-DT is trained with the input as same as the DiTAC that combines the τ^{prompt} and τ^{input}. This helps to discriminate the effect between additional information of prompt history and the discrete teamwork abstraction learning architecture.
- **DiTAC-based Behavior Cloning (DiTAC-BC)**.We omit DiTAC's reward-to-go tokens required in both τ^{prompt} and τ^{input}. We keep exactly the same model architecture of DiTAC for DiTAC-BC and mask all reward information in both training and evaluation stage to investigate the effect of reward-to-go tokens.
- **LIAM-Off; ODITS-Off**. Both **Local Information Agent Model** [22] and **Online aDaptation via Inferred Teamwork Situations** [12] learn a policy behavior representation during the online interactions with teammates originally, to ensure fairness, we conduct these two algorithms in an offline manner with our offline datasets. The offline variants use pre-collected data for training (no online interactions), while original **LIAM** and **ODITS** train via online interactions with some predefined teammates. These are helping to show the challenges exists in adapting the online algorithm to offline datasets.

We train our DiTAC model with prompt length $L^{\star} = 1$ and trajectory length $L = 10$ for all experiments presented in Figure5. To be consistent for two variants, we train DT with trajectories length $L = 10$ and train Prompt-DT with the same setting as DiTAC.

5.3.1 Performance comparison in three environments.

The performance results across three environments are presented in the Figure5. We first observe that our DiTAC and DiTAC-BC models outperforms all other baselines, which verifies the effectiveness. In the LBF environment, the LIAM-Off and ODITS-Off obviously barely cooperate with the teammate to acquire even one food, which shows that current state-of-the-art online algorithms are infeasible to address the challenges when the online interaction is disabled during the training process. While comparing with the results in PP, we can see the ODITS-Off presents the possible trend that adapting to cooperate with unseen teammates, which indicates that it indeed extracting teamwork situation knowledge from the offline dataset slightly. In contrast, the LIAM-Off shows slightly effectiveness in SMAC Fork.

Table 2: Ablation: The effect of prompt length on DiTAC's generalization ability.

L*	LBF	PP(2a4s)	PP(3a6s)	PP(4a8s)	PP(8a12s)	SMAC Fork
1	0.2438±0.0013	7.7972±0.7909	29.9510±0.1569	46.2561±0.8192	89.1335±0.5027	2.2056±0.3236
2	0.2310±0.0088	**8.4026±0.1478**	**30.2377±0.2152**	46.5892±0.6845	**89.5149±0.5001**	2.1333±0.0967
3	**0.2490±0.0013**	8.0023±0.4076	29.9599±0.1593	**46.9495±0.5146**	88.8402±0.3165	**2.2101±0.7325**
5	0.2345±0.0059	6.4059±0.4857	30.1508±0.3754	46.3828±0.3244	88.8307±0.5317	2.1908±0.2236

Table 3: Ablation: The effect of discrete codebook size on DiTAC's generalization ability.

K	LBF	PP(2a4s)	PP(3a6s)	PP(4a8s)	PP(8a12s)	SMAC Fork
32	0.2406±0.0205	7.3804±0.5756	29.6338±1.0405	**46.9982±0.6767**	89.0073±0.3190	2.2035±0.3948
64	0.2438±0.0013	7.7972±0.7909	29.9510±0.1569	46.2561±0.8192	89.1335±0.5027	**2.2056±0.3236**
128	0.2439±0.0087	7.3911±0.9622	30.0234±0.3989	45.9995±0.6381	**90.1249±0.2082**	2.1197±0.0962
256	**0.2487±0.0015**	**8.3958±0.3837**	**30.2096±0.4271**	46.9024±0.4976	89.6843±0.4828	2.1172±0.1255

DT presents similar performance among all baselines in three environments, indicating its ability to learn basic behavior patterns from offline trajectories. However, it is blind to cooperating with various types of teammates. Additionally, we observe that Prompt-DT demonstrates similar performance to our method in the LBF environment but loses its effectiveness in more complex environments such as PP and SMAC Fork. In fact, it performs even worse than vanilla DT, possibly because the prompt structure in Prompt-DT fails to extract effective team-level strategies information as the environments become more complex. This finding validates that specific design for capturing teamwork situation abstraction from trajectory segments τ^{prompt} is required. Furthermore, we hypothesize that the masked bi-directional transformer structure enforces the prompt to model teammate behavior policy types based on partial information from ad hoc agents. Hence, while providing prompt information directly can offer some benefits, if we cannot extract effective information from the prompt trajectory, it may even negatively impact the model. The masked prompt structure, combined with the learned codebook that grounds the learning of teammate abstractions in the space of (o, a), further enhances the generalization ability of our model.

5.3.2 Performance analysis as agent count increases.

We further extend the experiments to include settings with an increasing number of agents in the Predator-Prey environment and reported the results in the right part of Figure5. We further analyze how prompt trajectory segment length τ^{prompt} impacts the quality of learned teamwork situation abstractions. For instance, longer segments may capture phased strategies (e.g., 'scout-then-attack'), while shorter segments focus on atomic coordination patterns. Our DiTAC model achieves superior results in episode return across varying numbers of teammates, verifying the effectiveness and generalization ability of learned temwork abstraction. DiTAC-BC performs similarly to DiTAC in most scenarios and even better than DiTAC in the PP setting with 8 agents. Similar to the findings in [34], this improvement may be attributed to the high quality of historical trajectories collected with the training policies in PP(8a12s), as shown in Table 1, demonstrating that high-quality cooperation trajectories provide more collaboration-specific information than the reward-to-go tokens stored in the historical context. LIAM-Off and ODITS-Off continue to present low performance, with these algorithms showing higher variability compared to other baselines. This should be explained by the more serious out-of-distribution issue that arises when applying online algorithms within an offline training setting. DT is still lower than our model although it outperforms other baselines. However, Prompt-DT exhibits a continuous decline in performance as the number of agents increases. This indicates the importance of learning effective teamwork abstractions, and that invalid prompt teamwork abstraction may lead to serious miscoordination. In the scenario where 4 out of 8 agents are controlled as ad hoc agents in the SMAC Fork environment, our models DiTAC and DiTAC continue to present slight advantages compared to other baselines. This demonstrates the effectiveness of our models even as the number of ad hoc agents increases. We also note that the performance variation of all baselines is more pronounced than in other environments, which may be attributed to the low quality of the collected offline trajectories.

5.4 Ablation Study

Prompt Length We conduct experiments over all scenarios mentioned before with different prompt length. The default setting of our DiTAC model is $L^\star = 1$. As shown in the Table. 2, DiTAC achieves similar high performances with different prompt lengths, even when the prompt only contains one timesteps history information. Increasing the prompt length $L^\star$ does not apparently increase the performance. We conjecture that longer prompts of non-expert trajectories bring noise from irrelevant information. Shorter prompts, by contrast, reduce such interference and avoid input length mismatches, enabling better adaptability.

Discrete Codebook Size We train our DiTAC model with varying codebook sizes to evaluate the impact of the VQ-VAE training. The results are summarized in Table. 3, revealing that competitive performance can be achieved even with a smaller number of codes. This can be explained with finite teamwork abstractions required in current experiments environments. Therefore, once the codebook size is sufficiently large to contain teamwork abstractions, the models perform similarly. Overall, there are slight benefits associated with increasing the codebook size. We adopt $K = 64$ in our model training to balance computation and performance.

6 Conclusions and Limitations

In this work, we explored extracting teammate abstraction with a VQ-VAE codebook and leveraged it to train ad hoc agent for collaborating with previously unknown teammates in multi-agent environments. Experimental results show that our DiTAC model significantly outperforms various baselines and its variants in widely used ad hoc teamwork tasks. We also showed that DiTAC is robust to the prompt length and varying codebook size.

Our experiments demonstrate that teamwork situation abstractions derived from trajectory segments generalize effectively to small teams. As team size increases, the diversity of collaborative strategies within segments grows, necessitating larger codebooks to capture fine-grained patterns. Furthermore, the quality and diversity of pre-collected historical interaction trajectories involving unknown teammates significantly impact overall performance. In future research, we aim to investigate these issues clearly.

Acknowledgements

The work described in this paper was supported in part by the Research Grants Council of the Hong Kong Special Administrative Region, China, under Project CUHK 14200824; and by the Hong Kong Innovation and Technology Fund, under Project MHP/092/22. Additional support was provided by Guangdong Basic and Applied Basic Research Foundation (2025A1515010247) and the Fundamental Research Funds for the Central Universities (2024ZYGXZR069).

References

[1] S. Barrett and P. 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.

[2] M. Bowling and P. McCracken. Coordination and adaptation in impromptu teams. In *AAAI*, volume 5, pages 53–58, 2005.

[3] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[4] W. Böhmer, V. Kurin, and S. Whiteson. Deep coordination graphs. *arXiv preprint arXiv:1910.00091*, 2019. URL https://arxiv.org/abs/1910.00091.

[5] D. Charles and M. Black. Dynamic player modeling: A framework for player-centered digital games. In *Proc. of the International Conference on Computer Games: Artificial Intelligence, Design and Education*, pages 29–35, 2004.

[6] L. Chen, K. Lu, A. Rajeswaran, K. Lee, A. Grover, M. Laskin, P. Abbeel, A. Srinivas, and I. Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *arXiv preprint arXiv:2106.01345*, 2021. URL https://arxiv.org/abs/2106.01345.

[7] S. Chen, E. Andrejczuk, Z. Cao, and J. Zhang. Aateam: Achieving the ad hoc teamwork by employing the attention mechanism. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 7095–7102, 2020.

[8] H. Ding, C. Jia, C. Guan, F. Chen, L. Yuan, Z. Zhang, and Y. Yu. Coordination scheme probing for generalizable multi-agent reinforcement learning. *OpenReview*, 2023. URL https://openreview.net/forum?id=PAKkOriJBd. Submitted to ICLR 2023.

[9] A. Drachen, A. Canossa, and G. N. Yannakakis. Player modeling using self-organization in tomb raider: Underworld. In *2009 IEEE symposium on computational intelligence and games*, pages 1–8. IEEE, 2009.

[10] I. Durugkar, E. Liebman, and P. Stone. Balancing individual preferences and shared objectives in multiagent reinforcement learning. International Joint Conference on Artificial Intelligence, 2020.

[11] S. Fujimoto, D. Meger, and D. Precup. Off-policy deep reinforcement learning without exploration. In *International conference on machine learning*, pages 2052–2062. PMLR, 2019.

[12] P. Gu, M. Zhao, J. Hao, and B. An. Online ad hoc teamwork under partial observability. In *International conference on learning representations*, 2021.

[13] R. Hare and Y. Tang. Player modeling and adaptation methods within adaptive serious games. *IEEE Transactions on Computational Social Systems*, 10(4):1939–1950, 2022.

[14] A. Kumar, A. Zhou, G. Tucker, and S. Levine. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33:1179–1191, 2020.

[15] N. Lau, L. S. Lopes, G. Corrente, N. Filipe, and R. Sequeira. Robot team coordination using dynamic role and positioning assignment and role based setplays. *Mechatronics*, 21(2):445–454, 2011.

[16] S. Levine, A. Kumar, G. Tucker, and J. Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

[17] H. Li, T. Ni, S. Agrawal, F. Jia, S. Raja, Y. Gui, D. Hughes, M. Lewis, and K. Sycara. Individualized mutual adaptation in human-agent teams. *IEEE Transactions on Human-Machine Systems*, 51(6):706–714, 2021.

[18] F. Liu, H. Liu, A. Grover, and P. Abbeel. Masked autoencoding for scalable and generalizable decision making. *Advances in Neural Information Processing Systems*, 35:12608–12618, 2022.

[19] L. Meng, M. Wen, Y. Yang, C. Le, X. Li, W. Zhang, Y. Wen, H. Zhang, J. Wang, and B. Xu. Offline pre-trained multi-agent decision transformer: One big sequence model tackles all smac tasks. *arXiv preprint arXiv:2112.02845*, 2021.

[20] R. Mirsky, W. Macke, A. Wang, H. Yedidsion, and P. Stone. A penny for your thoughts: The value of communication in ad hoc teamwork. International Joint Conference on Artificial Intelligence, 2020.

[21] F. A. Oliehoek, M. T. Spaan, N. Vlassis, and S. Whiteson. Exploiting locality of interaction in factored dec-pomdps. In *Int. Joint Conf. on Autonomous Agents and Multi-Agent Systems*, pages 517–524, 2008.

[22] G. Papoudakis, F. Christianos, and S. V. Albrecht. Agent modelling under partial observability for deep reinforcement learning. *arXiv preprint arXiv:2006.09447*, 2020. URL https://arxiv.org/abs/2006.09447.

[23] G. Papoudakis, F. Christianos, L. Schäfer, and S. V. Albrecht. Benchmarking multi-agent deep reinforcement learning algorithms in cooperative tasks. *arXiv preprint arXiv:2006.07869*, 2020. URL https://arxiv.org/abs/2006.07869.

[24] A. Rahman, E. Fosong, I. Carlucho, and S. V. Albrecht. Generating teammates for training robust ad hoc teamwork agents via best-response diversity. *Transactions on Machine Learning Research*, 2023.

[25] A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, and I. Sutskever. Zero-shot text-to-image generation. In *International conference on machine learning*, pages 8821–8831. Pmlr, 2021.

[26] M. Rovatsos and M. Wolf. Towards social complexity reduction in multiagent learning: the adhoc approach. In *Proceedings of the 2002 AAAI Spring Symposium on Collaborative Learning Agents*, pages 90–97, 2002.

[27] M. Samvelyan, T. Rashid, C. S. De Witt, G. Farquhar, N. Nardelli, T. G. Rudner, C.-M. Hung, P. H. Torr, J. Foerster, and S. Whiteson. The starcraft multi-agent challenge. *arXiv preprint arXiv:1902.04043*, 2019.

[28] R. S. Sutton, D. Precup, and S. Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999.

[29] Z. Tang, C. Yu, B. Chen, H. Xu, X. Wang, F. Fang, S. Du, Y. Wang, and Y. Wu. Discovering diverse multi-agent strategic behavior via reward randomization. *arXiv preprint arXiv:2103.04564*, 2021.

[30] A. Van Den Oord, O. Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.

[31] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[32] T. Wang, T. Gupta, A. Mahajan, B. Peng, S. Whiteson, and C. Zhang. Rode: Learning roles to decompose multi-agent tasks. *arXiv preprint arXiv:2010.01523*, 2020.

[33] A. Xie, D. Losey, R. Tolsma, C. Finn, and D. Sadigh. Learning latent representations to influence multi-agent interaction. In *Conference on robot learning*, pages 575–588. PMLR, 2021.

[34] M. Xu, Y. Shen, S. Zhang, Y. Lu, D. Zhao, B. J. Tenenbaum, and C. Gan. Prompting decision transformer for few-shot policy generalization. In *Thirty-ninth International Conference on Machine Learning*, 2022.

[35] G. N. Yannakakis. Game ai revisited. In *Proceedings of the 9th conference on Computing Frontiers*, pages 285–292, 2012.