COORDINATION SCHEME PROBING FOR GENERALIZABLE MULTI-AGENT REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Coordinating with previously unknown teammates without joint learning is a crucial need for real-world multi-agent applications, such as human-AI interaction. An active research topic on this problem is ad hoc teamwork, which improves agents' coordination ability in zero-shot settings. However, previous works can only solve the problem of a single agent's coordination with different teams, which is not in line with arbitrary group-to-group coordination in complex multi-agent scenarios. Moreover, they commonly suffer from limited adaptation ability within an episode in a zero-shot setting. To address these problems, we introduce the Coordination Scheme Probing (CSP) approach that applies a disentangled scheme probing module to represent and classify the newly arrived teammates beforehand with limited pre-collected episodic data and makes multi-agent control accordingly. To achieve generalization, CSP learns a meta-policy with multiple sub-policies that follow distinguished coordination schemes in an end-to-end fashion and automatically reuses it to coordinate with unseen teammates. Empirically, we show that the proposed method achieves remarkable performance compared to existing ad hoc teamwork and policy generalization methods in various multi-agent cooperative scenarios.

1 INTRODUCTION

Multi-Agent Reinforcement Learning (MARL) (Gronauer & Diepold, 2021) holds promise in numerous cooperative domains, including resource management (Xi et al., 2018), traffic signal control (Du et al., 2021), and autonomous vehicles (Zhou et al., 2020). A number of methods (Lowe et al., 2017; Wang et al., 2020b; Papoudakis et al., 2019; Christianos et al., 2021) have been proposed to deal with joint policy learning and scalability issues. However, previous works commonly assume a fixed team composition, where agents only need to coordinate with training partners and do not consider generalization. Such a process is not in line with real-world applications that require agents to cooperate with unknown teammates whose coordination schemes may not be explicitly available. When coordinating with distributed teammates, co-trained agents may fail (Gu et al., 2022).

Another research domain dedicated to this need is ad hoc teamwork (Mirsky et al., 2022; Zhao et al., 2021), which stands at a single agent's perspective to adapt to different teams in a zero-shot fashion. Some approaches use domain randomization to expose the ad hoc agent to diverse partners during training (Strouse et al., 2021; Barrett et al., 2017; Chen et al., 2020). Thus, it will learn the commonality of different coordination schemes and avoid overfitting to a particular one. Another method utilizes Bayesian posteriors (Barrett & Stone, 2015; Stone et al., 2010) to continuously build awareness of teammates' types during coordination and adapt accordingly. However, there are three significant limitations: (1) The differences between coordination schemes could be very subtle and lie in only a few critical decisions. In this case, zero-shot coordination is not always feasible. As shown in the example in Fig. 1, teammate's behaviors are indistinguishable before the final critical action. (2) The information for identifying different teams is collected by the same policy that aims to maximize coordination performance. Thus, the exploration-exploitation dilemma will cause adverse effects on both sides. (3) Ad hoc teamwork stands at a single agent's perspective, which cannot deal with arbitrary group-to-group generalizations in complex multi-agent scenarios.

To overcome these limits and achieve generalizable coordination, we propose a multi-agent learning framework called Coordination Scheme Probing (CSP). Instead of doing zero-shot coordination

Figure 1: An example from Overcooked (Carroll et al., 2019a), where the chef in the blue hat needs to cooperate with chefs in other colors. Different chefs put the dishes in different positions after they are done cooking. For example, during training, the chefs in the green and orange hats put their dishes in the red and blue pentagrams as their last action, respectively. When a chef in the pink hat comes, how should the chef in the blue hat work with him? Since the first four actions for different chefs are indistinguishable, how to detect his type and coordinate with him is challenging.

like previous ad hoc methods, CSP tries to capture the unknown teammates' coordination scheme beforehand with limited pre-collected episodic data. Concretely, we start with generating a set of teams with high performance and diversified behaviors to discover possible coordination schemes for the task. After that, the scheme probing module learns to interact with teams in the training population to reveal their coordination schemes and represents them with dynamics reconstruction trick (Hospedales et al., 2021) at the team level. Finally, a multimodal meta-policy is trained with MARL, with each sub-modal acquiring a unique coordination scheme.

The whole learning framework of CSP learns multiple coordination schemes from a given task in an end-to-end fashion and automatically reuses the most suitable scheme recognized by the scheme probing module to coordinate with unknown partners. To validate our method, we conducted experiments on four challenging multi-agent cooperative scenarios and found that CSP achieves a more effective and robust generalization to unknown teammates compared to various baselines. Our visual analysis further confirms that CSP acquires multiple coordination schemes that are clearly distinguishable and can be precisely recognized. Meanwhile, our ablations show the necessity of probing unknown teammates in advance as well as using a multimodal policy for adaptation.

2 Related Work

Multi-Agent Reinforcement Learning (MARL). Fully cooperative MARL methods (Foerster et al., 2018) mainly focus on deploying a fixed team with centralized training and decentralized execution (CTDE) scheme (Oliehoek et al., 2008). Value factorization methods (Sunehag et al., 2018; Rashid et al., 2018; Wang et al., 2020a) are adopted widely in the CTDE process to solve the instability of teammates in the training process. Some works further use agent modeling to achieve stable training. ATT-MADDPG (Mao et al., 2019) uses attention-based centralized critic for value estimation, while LIAM (Papoudakis et al., 2021a) aims at estimating teammates' current actions and observations based on the learning agent's local history alone. Our work also utilizes the CTDE scheme to learn policies collaborating with others. However, such a learning process may converge to a fixed coordination scheme and may not coordinate well with unseen teammates following a different coordination mode. To jump out of the local optimal scheme, Tang et al. (2021) propose a method that could find a hard-to-search optimal coordination scheme by increasing diversity of population. In our work, we borrow the idea of diversity to mine various coordination schemes and simulate unseen coordination schemes.

Ad Hoc Teamwork. Ad hoc teamwork aims to learn a single autonomous agent that cooperates well with other agents without pre-coordination, such as joint training (Mirsky et al., 2022). There is a large body of work about ad hoc teamwork (Berger, 2007; Ravula et al., 2019; Mirsky et al., 2022). Among them, the top concerns are identification and specialization with unknown teammates in various information-limited situations (Stone et al., 2010). As for the first task, type inference methods use Bayesian belief update (Barrett et al., 2017; Albrecht et al., 2016) over the hypothesized type space of teammates. Experience recognition methods like PLASTIC (Barrett et al., 2017) and AATEAM (Chen

et al., 2020) take prior experience buffers into account to identify agents from trajectories rather than transitions. As for the latter task, planning approaches use bespoke planning (Bowling & McCracken, 2005; Ravula et al., 2019) or MCTS planning procedure (Kawaharazuka et al., 2021; Albrecht & Stone, 2017) to select actions suited to specific tasks. A recent work ODITS (Gu et al., 2022) introduces an information-based regularizer to automatically approximate the global information with a joint variational autoencoder (VAE). However, most ad hoc teammate works learn a single-agent ad hoc agent and do not consider a multi-agent control. And some teammates may not be distinguishable and adpatable within an episode. There still exists an urgent need to design a few-shot method to probe team features and makes control in multi-agent scenarios.

Population Based Training. It is widely accepted that one of the keys to generalization ability is the exposure to diverse partners during the training phase (Strouse et al., 2021). Population based training (PBT) (Jaderberg et al., 2017) maintains a set of diverse policies and samples from it to help train the generalization agent. One approach utilizes the stochastic nature of the environment and learns to train multiple policies independently with different initializations (Carroll et al., 2019a; Charakorn et al., 2021). Fictitious co-play (FCP) (Strouse et al., 2021; Gupta et al., 2021) further extends it by adding checkpoints from different training steps to simulate teammates with different strengths.

3 Background and Problem Formalization

Cooperative Multi-Agent Task. We consider a fully cooperative multi-agent task with partial observability, which can be modeled as a Dec-POMDP (Oliehoek & Amato, 2016) defined by a tuple $\mathcal{M} = \langle \mathcal{N}, \mathcal{S}, \{\mathcal{O}_i\}_{i=1}^n, \{\mathcal{A}_i\}_{i=1}^n, \Omega, P, R, \gamma, d_0 \rangle$, where $\mathcal{N} = \{1, \ldots, n\}$ is the set of agents, $\mathcal{S}$ is the set of global states, $\mathcal{O}_i$ and $\mathcal{A}_i$ are agent i's observation space, $\mathcal{A}_i$ is agent i's action space, $\gamma \in [0, 1)$ is the discount factor, and d_0 denotes the initial state distribution. At each timestep, agent $i \in \mathcal{N}$ acquires a local observation $o_i \in \mathcal{O}_i$ with observation function $\Omega(s, i)$ and chooses an action $a_i \in \mathcal{A}_i$ via individual policy. The joint action $\boldsymbol{a} = \langle a_1, \ldots, a_n \rangle$ taken by joint policy $\pi(\boldsymbol{a}|\boldsymbol{\tau})$ transitions the system to next global state s' according to the transition function $P(s' \mid s, \boldsymbol{a})$, and all agents get a shared reward $r = R(s, \boldsymbol{a})$.

Coordination Scheme. It describes unique high-level behaviors that lead to near-optimal returns of a task. The cooperative task itself determines the set of coordination schemes C, which reflects different ways to solve the task. To achieve good coordination, each agent should agree to the same coordination scheme $c \in C$ and stick to the policy π^c derived by the scheme throughout an episode. In the example of Fig. 1, there are two coordination schemes "Up" and "Down".

Problem Formalization. we aim to control a group of agents to coordinate with the rest of the teammate agents under coordination scheme c drawn from C. We denote the agent group under control by G^1 and the teammate group by G^{-1}, where $G^1 \cup G^{-1} = \mathcal{N}$ and $G^1 \cap G^{-1} = \emptyset$. For simplicity in notation, we treat each group as a single combined agent with joint observations $\boldsymbol{\tau} = \langle \boldsymbol{\tau}^1, \boldsymbol{\tau}^{-1} \rangle$ and joint actions $\boldsymbol{a} = \langle \boldsymbol{a}^1, \boldsymbol{a}^{-1} \rangle$. And their policies are defined as $\pi^i : \tau^i \to a^i, i \in \{1, -1\}$. The optimal policy π_θ^{1*} parameterized by θ for G^1 is to maximize the discounted cumulative reward: $\theta^* = \arg\max_\theta \mathbb{E}_{(\boldsymbol{a}^1, \boldsymbol{a}^{-1}) \sim (\pi_\theta^1, \pi^{-1}), c \sim C} \left[\sum_{t=0}^{H-1} \gamma^t r_{t+1} \,\middle|\, \pi^{-1} = (\pi^c)^{-1}, P, d_0 \right]$, where r_{t+1} is the shared reward at timestep $t+1$, and H is the episode length. Since we do not have access to the underlying scheme set C of the task, we use the policy set Π of G^{-1} as a surrogate and rewrite the objective as: $\theta^* = \arg\max_\theta \mathbb{E}_{(\boldsymbol{a}^1, \boldsymbol{a}^{-1}) \sim (\pi_\theta^1, \pi^{-1}), \pi \sim \mathcal{U}(\Pi)} \left[\sum_{t=0}^{H-1} \gamma^t r_{t+1} \,\middle|\, P, d_0 \right]$, where $\mathcal{U}(\cdot)$ denotes a uniform distribution.

4 Method

This section describes how the CSP framework addresses the generalizable coordination problem in an end-to-end manner (Fig. 2). When given a cooperative task, CSP learns with three stages: (1) It generates a diversified population of team policies to discover multiple feasible coordination schemes. (2) It trains a scheme probing module to efficiently reveal and represent coordination schemes of different teams by team-dynamics reconstruction. (3) It groups similar schemes to reduce learning complexity and solve coordination tasks with a multimodal meta-policy.

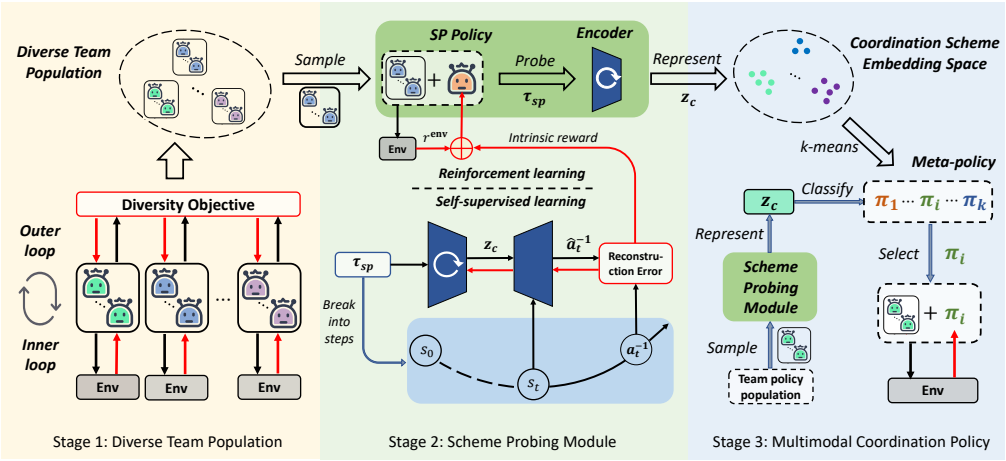


Figure 2: Overall framework of CSP, where SP is short for "scheme probing".

4.1 Diverse Team Population

We first need to use a set of diversified team policies to simulate the coordination schemes required for the training and evaluation phases. Instead of letting each team acquire diversity with different initializations (Carroll et al., 2019a; Strouse et al., 2021) alone, we propose the Soft-Value Diversity (SVD) objective and an alternate optimization process to gain diversity explicitly. We maintain a population with M independent multi-agent teams, each learning with an MARL method (e.g., QMIX (Rashid et al., 2018)). While each team performs independent "inner loop" learning, the population periodically performs centralized "outer loop" updates for all teams to maximize SVD:

$$J_{\text{SVD}}(\{\theta_i\}_{i=1}^{M}) = \mathbb{E}_{\boldsymbol{\tau},\boldsymbol{a}}\left[\sum_{i=1}^{M} \|\overline{D}(\boldsymbol{\tau},\boldsymbol{a}) - D_i(\boldsymbol{\tau},\boldsymbol{a})\|_2^2\right], \tag{1}$$

where θ_i denotes the Q-network's parameters for the i-th team's policy, $\boldsymbol{\tau}$ and $\boldsymbol{a}$ denote individual observation and action, $D_i(\boldsymbol{\tau},\boldsymbol{a}) = \frac{\exp\left(Q_{\theta_i}(\boldsymbol{\tau},\boldsymbol{a})\right)}{\sum_{\boldsymbol{a}'\in\mathcal{A}}\exp\left(Q_{\theta_i}(\boldsymbol{\tau},\boldsymbol{a}')\right)}$ is the normalized value estimation obtained by performing the Boltzmann softmax operator (Asadi & Littman, 2017), and $\overline{D}(\boldsymbol{\tau},\boldsymbol{a}) = \frac{1}{M}\sum_{i=1}^{M} D_i(\boldsymbol{\tau},\boldsymbol{a})$ is the average observation-action value estimation for all policies in the population Π. We use the Monte Carlo method to estimate the expectation based on recent collected trajectories in the replay buffer.

Maximizing SVD encourages teams within the population to have different value estimations of each observation-action pair by increasing the gap between their current values and the mean. This diversity in value estimation reflects various local optima caused by different joint decisions in multi-agent scenarios, where the same action will have different outcomes when teammates' actions change. The "inner" and "outer" loops in the framework are designed to improve individual team performance and diversity among different teams. We set a hyper-parameter to control the learning frequency of these two loops, which is a trade-off between the quality and diversity of policies in the population. We obtain Π_{train} and Π_{eval} independently via this process.

4.2 Scheme Probing Module

After having a diverse team population Π_{train}, we train the scheme probing module upon it to efficiently reveal and represent different teams' coordination schemes. It has two main parts. One is the scheme probing policy π_{sp} which interacts with a team for an entire episode and gathers the trajectory $\tau_{\text{sp}} = \{s_t, \boldsymbol{a}_t\}_{t=0}^{H}$ that reveals the current teammates' coordination scheme. The other is the team-dynamics autoencoder $\langle E_c, D_c\rangle$, which learns a representation z_c of this team's real coordination scheme c based on τ_{sp}.

Team-dynamics Autoencoder. The encoder E_c is parameterized as an LSTM (Hochreiter & Schmidhuber, 1997) which takes as input τ_{sp} and outputs an embedding vector z_c. The decoder D_c is a

feed-forward network that takes as input both z_c and current state s_t, and predicts teammates' next joint action distribution $\boldsymbol{a}_t^{-1}$. Formally,

$$z_c = E_c\left(\tau_{\text{sp}};\ \theta_c\right), \quad \hat{\boldsymbol{a}}_t^{-1} = D_c\left(\cdot \mid s_t, z_c;\ \phi_c\right). \tag{2}$$

The parameters θ_c and ϕ_c are jointly trained to minimize the cross entropy loss (i.e., reconstruction error in Fig. 2) of $\hat{\boldsymbol{a}}_t^{-1}$ and $\boldsymbol{a}_t^{-1}$ averaged over the entire trajectory τ_{sp}:

$$\mathcal{L}_{\text{tot}} = \frac{1}{H}\sum_{t=1}^{H}\mathcal{L}_{\text{pred}}(t) = -\frac{1}{H}\sum_{t=1}^{H}\log D_c(\boldsymbol{a}_t^{-1} \mid s_t, z_c;\ \phi_c). \tag{3}$$

We call this approach *team-dynamics reconstruction* for the following reasons. Firstly, D_c receives no historical information, so it cannot infer a team's behaviors based on the temporal ordering of states and joint actions. In this case, E_c is forced to embed information about this team's dynamics into z_c to make a good reconstruction. Additionally, since s_t is already input of D_c, E_c has no motive to embed any information about states. Thus, the embedding z_c contains only information about the policy and not the environment, making it a compressed and precise representation of a team.

Scheme Probing Policy. Instead of collecting information passively, we expect π_{sp} to actively guide teammates' behaviors and reveal their coordination schemes. To this end, we introduce the reconstruction loss above at each timestep as an intrinsic reward for π_{sp}, which is added to the original environmental reward r_t^{env}:

$$r_t' = r_t^{\text{env}} + \alpha\mathcal{L}_{\text{pred}}(t), \tag{4}$$

where α is an adjustable hyperparameter to achieve a trade-off between coordination performance and information gain through probing. This additional term encourages π_{sp} to explore states with the large behavioral uncertainty across different teammates (reflected in high reconstruction loss), which is considered helpful in determining their identities and representing their coordination schemes.

4.3 Multimodal Coordination Policy

When the scheme probing module is well trained, it is fixed and used to guide downstream scheme-specific control. Common context-based methods (Hausman et al., 2018; Yang et al., 2020) use embeddings as augmentation of the agent's input space. However, in scenarios of coordination generalization, the behaviors under different coordination schemes can vary greatly and conflict with each other. Such processes require a single network to acquire multimodal behaviors as the context changes, which increases learning complexity and instability. As a comparison, we use embeddings to automatically group similar coordination schemes and solve each distinct group with an independent sub-policy to avoid conflicts.

Scheme Grouping. We first use the learned scheme probing module to probe and represent all teams in Π_{train} N times, which generates an embedding set Z of size $|Z| = NM$. Repeating it N times is to get the distribution of a team's representation rather than a single sample point when the environment and policy are stochastic. Then we perform k-means clustering based on Euclidean distances on Z to get k clusters with centers $\boldsymbol{\mu} = \langle \mu_1, \ldots, \mu_k \rangle$.

We use the Silhouette method (Rousseeuw, 1987) to automatically determine the most suitable k for the embedding set Z. The intuition is that a higher Silhouette value for a data point indicates that this point is placed in the correct cluster. Therefore, a cluster number k with the highest mean Silhouette value for all points in a dataset is desirable. Concretely, for the embedding set Z, the Sihouette value $SV(i)$ for each data point i that belongs to I-th cluster Z_I is defined as:

$$SV(i) = \begin{cases} \dfrac{d_{\text{out}}(i) - d_{\text{in}}(i)}{\max\{d_{\text{out}}(i), d_{\text{in}}(i)\}} & , \quad \text{if } |Z_I| > 1, \\ 0 & , \quad \text{if } |Z_I| = 1, \end{cases} \tag{5}$$

where $d_{\text{in}}(i)$ denotes the mean distance between data point i and other points in the same cluster, and $d_{\text{out}}(i)$ denotes the smallest mean distance of data point i to all points in any other cluster. They are defined as $d_{\text{in}}(i) = \frac{1}{|Z_I|-1}\sum_{j\in Z_I, i\neq j} d(i,j)$ and $d_{\text{out}}(i) = \min_{I\neq J}\frac{1}{|Z_J|}\sum_{j\in Z_J} d(i,j)$, where $d(i,j)$ is the Euclidean distance between data points i and j. We implemented an automatic k selection approach by linear searching for the maximum mean Silhouette value of all data points:

$$k^* = \arg\max_{k\leq M}\frac{1}{|Z|}\sum\nolimits_{i\in Z} SV(i). \tag{6}$$

The number of clusters k will be approximately equal to the number of environmental coordination schemes $|C|$ if Π_{train} already covers all possible schemes. So enlarging the size of Π_{train} will not infinitely increase the learning complexity.

Meta-Policy Learning. We initialize a multimodal meta-policy with k sub-policies, where each sub-policy takes local observation history as input:

$$\pi_{\text{meta}} = \{\pi_i(a \mid \tau) \mid i = 1, \ldots, k\}. \tag{7}$$

During training or deploying, when confronted with a team, we first let the scheme probing policy π_{sp} interact with it to get trajectory τ_{sp} and get the representation $z_c = E_c(\tau_{\text{sp}})$ of that teammate's coordination scheme. Then, we classify the teammate into one of the k classes according to the distance of z_c and all cluster centers $\boldsymbol{\mu}$. Once we have done the classification, we choose the corresponding sub-policy to do the reset of the control. Formally,

$$\pi_{\text{meta}}(a \mid \tau, \tau_{\text{sp}}) = \pi_{i^*}(a \mid \tau), \quad \text{where } i^* = \arg\min_i \|E_c(\tau_{\text{sp}}) - \mu_i\|_2^2. \tag{8}$$

This structural design takes advantage of multimodality to be highly expressive, allowing end-to-end learning of several vastly distinct coordination schemes simultaneously without affecting each other. Moreover, each sub-policy $\pi_i(a|\tau)$ only needs to acquire a unique and stationary coordination scheme. Compared to common context-based methods that use a single policy $\pi(a \mid \tau, z)$ to acquire all schemes, it reduces learning complexity and improves stability.

5 Experiments

In the section, we design experiments to answer the following questions: (1) How well does CSP perform when generalizing to unknown partners in multiple complex scenarios (Sec. 5.1)? (2) Can the proposed scheme probing process get meaningful and distinguishable representations (Sec. 5.2)? (3) Does the team population really have multiple coordination schemes (Sec. 5.3)? (4) What is the impact of each component of CSP (Sec. 5.4)?

We select four multi-agent cooperative environments with six scenarios as benchmarks: Level-based Foraging (LBF) (Papoudakis et al., 2021b) needs agents to find food randomly distributed on the map and eat it together. Predator-Prey (PP) (Böhmer et al., 2020) is a more challenging version of LBF that allows each prey to move randomly at each timestep. Overcooked (Strouse et al., 2021) requires two players coordinate to cook and deliver food to target locations. We use a standard layout *Coordination Ring* and a modified layout *Forced Coordination Hard* that enhances penalties for miscoordination. SMAC (Samvelyan et al., 2019) is a commonly used MARL benchmark that focuses on coordinated micromanagement. We use a standard map *1c3s5z* and a customized map *Fork* specially designed with multiple coordination schemes.

Five state-of-the-art approaches for ad hoc teamwork and MARL are chosen for comparison. PBT (Carroll et al., 2019b) uses domain randomization to train an ad hoc agent with a population of pre-trained partners. FCP (Strouse et al., 2021) draws on the practice of FSP (Heinrich et al., 2015) in games by constructing training sets with teammates' historical checkpoints. ODITS (Gu et al., 2022) applies a centralized "teamwork situation encoder" for end-to-end learning. LIAM (Papoudakis et al., 2021a) is an MARL method that predicts teammates' current behaviors based on local observation history, and FIAM (Papoudakis et al., 2021a) enhances LIAM by replacing local observations with global states. Notice that three of the five baselines (i.e., PBT, FCP, and ODITS) are ad hoc methods originally considered a single-agent setting. To make them compatible with multi-agent scenarios, we implement them upon an MARL framework VDN (Sunehag et al., 2018) that follows the CTDE paradigm to learn team policies for G^1. Other multi-agent methods (i.e., CSP, LIAM, and FIAM) also choose VDN as their base MARL framework for a fair comparison. Additionally, the original version of PBT, FCP, and ODITS use several self-play teams with different random seeds and initialization to construct a training population, which may be unstable and hard to compare. Instead, we reuse the same Π_{train} generated in CSP's stage 1 for all baselines, thus reducing the chance of outcome due to randomness of population generation. More details of each scenario and baseline are shown in Appendices B-C.

5.1 Evaluation of Performance

We first investigate how CSP and other baselines perform when coordinated with unknown diverse teammates. Fig. 3 shows the average performance of coordinating with all teams in Π_{eval} during

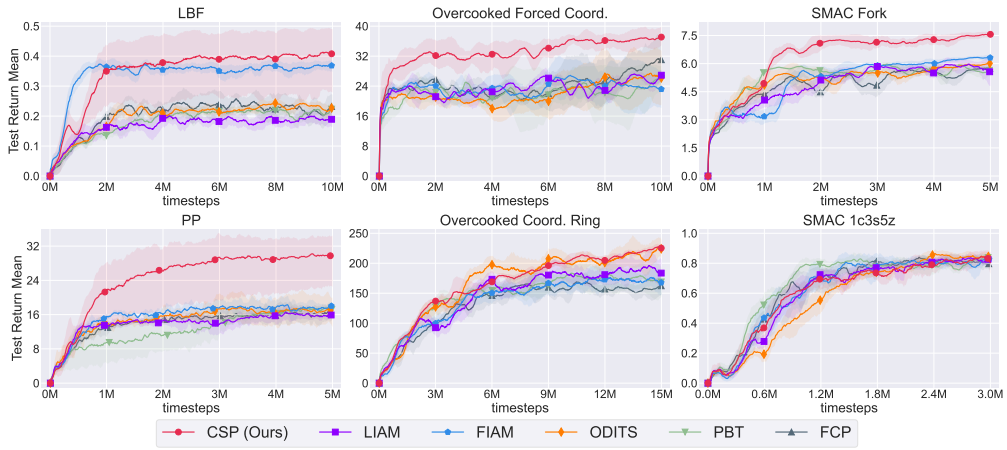


Figure 3: Mean generalization performance on Π_{eval} of all methods.

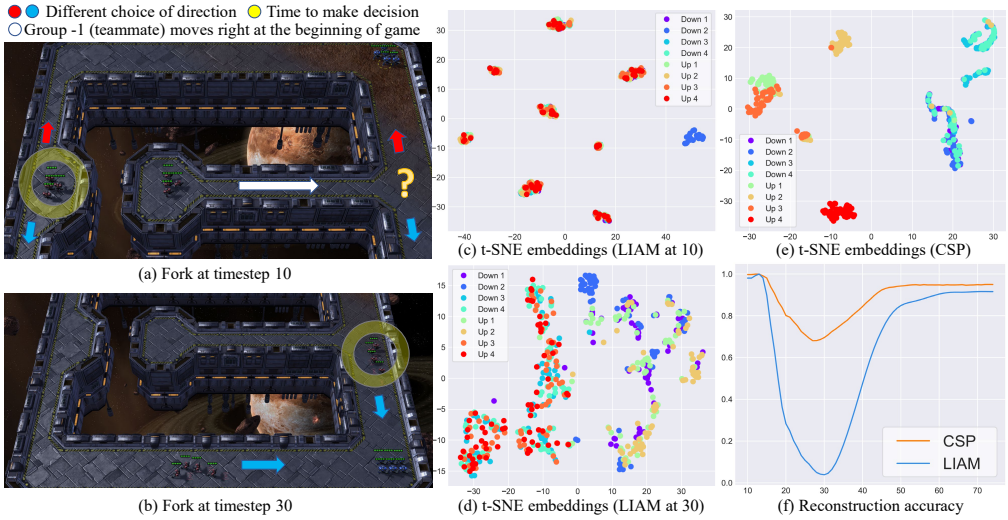
training. Since CSP always collects a probing trajectory τ_{sp} before adapted coordination, we use the average score of the two trajectories as CSP's performance metric for fair comparisons. All experiments are done with five random seeds, and all curves are marked with 95% confidence intervals in the shaded area.

In general, CSP outperforms PBT, FCP, and LIAM consistently and is better or at least comparable to FIAM and ODITS according to the tasks. In LBF, where each agent has a strictly limited observation range, FIAM has a clear advantage because it utilizes global information at each timestep to help make decisions. The comparable result of CSP shows that it is possible to achieve near-optimal control with pre-identified coordination scheme information alone, eliminating the need for global states throughout the entire episode. In PP, Overcooked, and SMAC's Fork, the cooperative tasks are challenging and contain several vastly different coordination schemes, which require the coordinator to change its behaviors aggressively and precisely. All baselines here have a clear gap with CSP. We believe the superiority comes from the fact that CSP uses multimodality to isolate the expression of different coordination schemes to avoid mutual interference. As a comparison, in SMAC's 1c3s5z, CSP and all baselines converge to roughly equivalent performance. The reason is that without a specially designed symmetry like Fork, the optimal policy in this task is reflected in focusing fire on the enemy and retreating when its health is low. The optimal decision under each state is sure with no ambiguity of different coordination schemes, and all methods acquire this optimal policy after training long enough. It is worth noting that CSP trains k sub-policies simultaneously but still has comparable or even better sample efficiency to baselines. We believe it is because CSP stabilizes training by combining the learning tasks of similar coordination schemes while avoiding conflict schemes from affecting each other.

5.2 Scheme Representation on SMAC

A meaningful and distinguishable representation of coordination schemes is the basis of CSP's adaptation ability. To demonstrate this, we visualize the embeddings of our scheme probing module with experiments on SMAC's Fork (Fig. 4). We let our scheme probing module interact with all teams in $\Pi = \Pi_{\text{train}} \cup \Pi_{\text{eval}}$ for 64 episodes. One of the baselines, LIAM, which does agent modeling at every timestep, also runs in parallel as a comparison.

Let's first make a brief understanding of the Fork task. There are two symmetrical points at *Up* and *Down* sides guarded by several enemies. If all the teammates choose to attack the same point, they will have enough force to eliminate all the enemies there and get a high reward. Otherwise, neither point will have a firepower advantage, and all the teammates will be destroyed and fail. Therefore, we can roughly claim that there are two basic coordination schemes $C = \{Up, Down\}$ in this task. Screenshots at timesteps 10 and 30 (Fig. 4a-b) indicate the early and middle stages of one episode. Asynchronous decision makings occur in (a) and (b) for G^1 and G^{-1} respectively to select a direction (*Up* or *Down*) to attack, and coordination succeeds if their choices are the same. Since G^{-1} may be controlled by various policies in Π, which will make different decisions, G^1 has to make the right choice based on the partners it meets.



(a) Fork at timestep 10
(b) Fork at timestep 30
(c) t-SNE embeddings (LIAM at 10)
(d) t-SNE embeddings (LIAM at 30)
(e) t-SNE embeddings (CSP)
(f) Reconstruction accuracy

Figure 4: (a)(b) SMAC's Fork at timesteps 10 and 30, when G^1 (left) and G^{-1} (right) make decisions respectively to attack enemies at point *Up* (red arrow) or *Down* (blue arrow). (c)(d) LIAM's embeddings of different teams at timesteps 10 and 30. (e) CSP's embeddings of different teams. (f) CSP and LIAM's mean reconstruction accuracy of G^{-1}'s joint action $\boldsymbol{a}_t^{-1}$ throughout one episode.

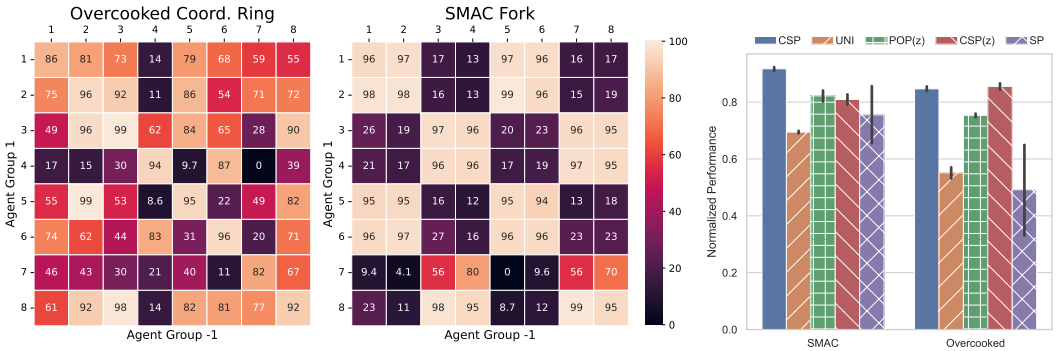
The t-SNE projection of CSP's scheme embeddings is shown in Fig. 4e, where different colors represent different teams in Π (named by their scheme observed) and each point represents a single run. There are two main phenomenons: (1) Each color forms a relatively compact cluster. (2) Clusters with similar colors (lighter or darker) tend to be close together, while deeper and lighter colors are farther apart. The former indicates that CSP's representations are highly consistent with low variance, which is beneficial for stabilizing downstream MARL learning. The latter shows that the embedding space holds semantically meaningful information, where teams with similar coordination schemes can be packed together. As a comparison, LIAM's embeddings at timesteps 10 and 30 are shown in Fig. 4c-d. We can observe that different teams are mixed up at the early stage, and a few local clusters emerge as time goes by but still cannot distinguish well between teams. Different from CSP's one-time probing and modeling approach, LIAM does modeling at each timestep to only capture teammates' temporary behaviors. As illustrated in the screenshots (white arrow), G^{-1} will always move right in the first few dozen steps, so the trajectories of different teams during this period are similar and hold insufficient information. It is impossible for LIAM to distinguish teams based on these trajectories, let alone make the right choice in the beginning. More results on other benchmarks are shown in App. D.1.

Fig. 4f presents the mean reconstruction accuracy of G^{-1}'s joint actions throughout one episode. We can observe that CSP has consistently higher accuracy than LIAM, especially at timesteps close to 30. As described above, the observation segment before is not informative enough, in which case LIAM will fail to predict teammate behaviors when sudden uncertainty occurs. By contrast, CSP has a comprehensive view of the teammates it coordinates with after the probing phase in advance, so it can fully guarantee its scheme prediction throughout the coordination phase.

5.3 Scheme Diversity

To verify those team policies generated in stage 1 do hold multiple coordination schemes, we perform Cross-Play (Hu et al., 2020) experiments on Overcooked's Coordination Ring and SMAC's Fork (Fig. 5a). Teams in $\Pi = \Pi_{\text{train}} \cup \Pi_{\text{eval}}$ are paired to play the role of G^1 and G^{-1} for all combinations.

Firstly, we can find that values on the diagonal from the top left to the bottom right corner are generally larger than others. This indicates that each team coordinates well with itself, as they are always in the same coordination scheme. As a comparison, the relatively lower performance of other points indicates the inability to coordinate across different schemes. Interestingly, the performance on SMAC's Fork shows that it has two basic coordination schemes, where each team can coordinate



(a) Cross-Play performance of full population Π, normalized to $[0, 100)$.

(b) Ablations.

Figure 5: (a) The scores of cross-play on the Overcooked Coordination Ring and SMAC Fork benchmarks. (b) Final generalization performance of CSP and ablations for each component.

with exactly half of the teams in Π that have the same coordination scheme. This phenomenon is aligned with our understanding of the task, as described above in Sec. 5.2. More results on other benchmarks are provided in App. D.2.

5.4 Ablation Study

We perform ablations on SMAC and Overcooked to demonstrate the importance of CSP's different components: UNI removes multimodality and only has a single sub-policy $\pi(\tau)$; POP(z) uses a single context-based policy $\pi(\tau, z)$ instead of sub-policies; CSP(z) extends each sub-policy to be $\pi_i(\tau, z)$; SP does not use Π_{train} and learns with self-play alone. We report the mean generalization performance with Π_{eval} and error bars indicating the 95% confidence interval as shown in Fig. 5b.

Firstly, we can observe that SP generally does the worst and has a large variance. This means that without being exposed to diverse partners during training, the policy can only find a single coordination scheme and is hard to generalize. UNI has a lower variance but still performs poorly, which indicates that using domain randomization helps make the policy robust to partner changes, but it cannot be specialized to each scheme without a specially designed adaptability module. The relatively lower performance of POP(z) compared to CSP confirms our claim earlier that it is difficult to make adaptations within a single network based on different scheme embedding. In complex multi-agent scenarios, each coordination scheme can imply highly different and even opposite behaviors, where using multiple sub-policies helps to isolate these schemes and makes each consistent. Finally, CSP(z) does not outperform the original version. This phenomenon shows that our scheme grouping process already fully uses the information contained in the embedding. Further use of it as additional input no longer results in a boost but may increase learning difficulty.

6 Closing Remarks

To achieve generalizable coordination in complex multi-agent scenarios and address limitations of ad hoc teamwork, this paper considers learning a coordination scheme probing module for teammates recognition and a meta-policy consisting of multiple sub-policies for few-shot coordination with unseen diverse teams. With the help of this probing module, we can reduce few-shot coordination to a multi-task RL problem by clustering the representation space. A multimodal policy is then end-to-end trained to solve it directly. Sufficient experiments compared against strong baselines on various benchmarks validate the effectiveness of our proposed method. We point out two limitations and interesting future work: (1) Co-evolution of the population would be a more general interaction setting, such as Quality Diversity (Parker-Holder et al., 2020). (2) Effective knowledge transfer between submodules, such as Soft Modularization (Yang et al., 2020), could be considered to improve sample efficiency. Finally, instead of training alongside artificial agents, we also hope to study the human-in-the-loop setting to adapt to people's dynamic needs and preferences.

REFERENCES

Stefano V. Albrecht and Peter Stone. Reasoning about hypothetical agent behaviours and their parameters. In *Proceedings of the International Conference on Autonomous Agents and MultiAgent Systems*, 2017.

Stefano V Albrecht, Jacob W Crandall, and Subramanian Ramamoorthy. Belief and truth in hypothesised behaviours. *Artificial Intelligence*, 2016.

Kavosh Asadi and Michael L Littman. An alternative softmax operator for reinforcement learning. In *Proceedings of the International Conference on Machine Learning*, 2017.

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 Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

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

Ulrich Berger. Brown's original fictitious play. *Journal of Economic Theory*, 2007.

Wendelin Böhmer, Vitaly Kurin, and Shimon Whiteson. Deep coordination graphs. In *Proceedings of the International Conference on Machine Learning*, 2020.

Michael H. Bowling and Peter McCracken. Coordination and adaptation in impromptu teams. In *Proceedings of the National Conference on Artificial Intelligence*, 2005.

Micah Carroll, Rohin Shah, Mark K. Ho, Tom Griffiths, Sanjit A. Seshia, Pieter Abbeel, and Anca D. Dragan. On the utility of learning about humans for human-AI coordination. In *Advances in Neural Information Processing Systems 32*, 2019a.

Micah Carroll, Rohin Shah, Mark K. Ho, Tom Griffiths, Sanjit A. Seshia, Pieter Abbeel, and Anca D. Dragan. On the utility of learning about humans for human-ai coordination. In *Annual Conference on Neural Information Processing Systems*, 2019b.

Rujikorn Charakorn, Poramate Manoonpong, and Nat Dilokthanakul. Learning to cooperate with unseen agent via meta-reinforcement learning. *arXiv preprint arXiv:2111.03431*, 2021.

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*, 2020.

Filippos Christianos, Georgios Papoudakis, Muhammad A Rahman, and Stefano V Albrecht. Scaling multi-agent reinforcement learning with selective parameter sharing. In *Proceedings of the International Conference on Machine Learning*, 2021.

Xin Du, Jiahai Wang, Siyuan Chen, and Zhiyue Liu. Multi-agent deep reinforcement learning with spatio-temporal feature fusion for traffic signal control. In *Proceedings of the European Conference on Machine Learning*, 2021.

Jakob N. Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.

Sven Gronauer and Klaus Diepold. Multi-agent deep reinforcement learning: A survey. *Artificial Intelligence Review*, 2021.

Pengjie Gu, Mengchen Zhao, Jianye Hao, and Bo An. Online ad hoc teamwork under partial observability. In *Proceedings of the International Conference on Learning Representations*, 2022.

Abhinav Gupta, Marc Lanctot, and Angeliki Lazaridou. Dynamic population-based meta-learning for multi-agent communication with natural language. In *Advances in Neural Information Processing Systems 34*, 2021.

Karol Hausman, Jost Tobias Springenberg, Ziyu Wang, Nicolas Heess, and Martin Riedmiller. Learning an embedding space for transferable robot skills. In *Proceedings of the International Conference on Learning Representations*, 2018.

Johannes Heinrich, Marc Lanctot, and David Silver. Fictitious self-play in extensive-form games. In *Proceedings of the International Conference on Machine Learning*, 2015.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 1997.

Timothy M Hospedales, Antreas Antoniou, Paul Micaelli, and Amos J Storkey. Meta-learning in neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

Hengyuan Hu, Adam Lerer, Alex Peysakhovich, and Jakob N. Foerster. "other-play" for zero-shot coordination. In *Proceedings of the International Conference on Machine Learning*, 2020.

Max Jaderberg, Valentin Dalibard, Simon Osindero, Wojciech M Czarnecki, Jeff Donahue, Ali Razavi, Oriol Vinyals, Tim Green, Iain Dunning, and Karen Simonyan. Population based training of neural networks. *arXiv preprint arXiv:1711.09846*, 2017.

Kento Kawaharazuka, Manabu Nishiura, Shinsuke Nakashima, Yasunori Toshimitsu, Yusuke Omura, Yuya Koga, Yuki Asano, Kei Okada, Koji Kawasaki, and Masayuki Inaba. Stability recognition with active vibration for bracing behaviors and motion extensions using environment in musculoskeletal humanoids. In *Proceedings of the IEEE International Conference on Soft Robotics*, 2021.

Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems 30*, 2017.

Hangyu Mao, Zhengchao Zhang, Zhen Xiao, and Zhibo Gong. Modelling the dynamic joint policy of teammates with attention multi-agent DDPG. In *Proceedings of the International Conference on Autonomous Agents and MultiAgent Systems*, 2019.

Reuth Mirsky, Ignacio Carlucho, Arrasy Rahman, Elliot Fosong, William Macke, Mohan Sridharan, Peter Stone, and Stefano V Albrecht. A survey of ad hoc teamwork: Definitions, methods, and open problems. *arXiv preprint arXiv:2202.10450*, 2022.

Frans A Oliehoek and Christopher Amato. *A Concise Introduction to Decentralized POMDPs*. Springer, 2016.

Frans A. Oliehoek, Matthijs T. J. Spaan, Shimon Whiteson, and Nikos A. Vlassis. Exploiting locality of interaction in factored dec-pomdps. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems*, 2008.

Georgios Papoudakis, Filippos Christianos, Arrasy Rahman, and Stefano V Albrecht. Dealing with non-stationarity in multi-agent deep reinforcement learning. *arXiv preprint arXiv:1906.04737*, 2019.

Georgios Papoudakis, Filippos Christianos, and Stefano V. Albrecht. Agent modelling under partial observability for deep reinforcement learning. In *Advances in Neural Information Processing Systems 34*, 2021a.

Georgios Papoudakis, Filippos Christianos, Lukas Schäfer, and Stefano V. Albrecht. Benchmarking multi-agent deep reinforcement learning algorithms in cooperative tasks. In *Advances the Neural Information Processing Systems 34*, 2021b.

Jack Parker-Holder, Aldo Pacchiano, Krzysztof Marcin Choromanski, and Stephen J. Roberts. Effective diversity in population based reinforcement learning. In *Advances in Neural Information Processing Systems 33*, 2020.

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

Manish Ravula, Shani Alkoby, and Peter Stone. Ad hoc teamwork with behavior switching agents. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 2019.

Peter J Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 1987.

Mikayel Samvelyan, Tabish Rashid, Christian Schroeder de Witt, Gregory Farquhar, Nantas Nardelli, Tim GJ Rudner, Chia-Man Hung, Philip HS Torr, Jakob Foerster, and Shimon Whiteson. The StarCraft multi-agent challenge. In *Proceedings of the International Conference on Autonomous Agents and MultiAgent Systems*, 2019.

Peter Stone, Gal A. Kaminka, Sarit Kraus, and Jeffrey S. Rosenschein. Ad hoc autonomous agent teams: Collaboration without pre-coordination. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2010.

DJ Strouse, Kevin R. McKee, Matt M. Botvinick, Edward Hughes, and Richard Everett. Collaborating with humans without human data. In *Advances in Neural Information Processing Systems 34*, 2021.

Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinícius Flores Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z. Leibo, Karl Tuyls, and Thore Graepel. Value-decomposition networks for cooperative multi-agent learning based on team reward. In *Proceedings of the International Conference on Autonomous Agents and MultiAgent Systems*, 2018.

Zhenggang Tang, Chao Yu, Boyuan Chen, Huazhe Xu, Xiaolong Wang, Fei Fang, Simon Shaolei Du, Yu Wang, and Yi Wu. Discovering diverse multi-agent strategic behavior via reward randomization. In *Proceedings of the International Conference on Learning Representations*, 2021.

Jianhao Wang, Zhizhou Ren, Terry Liu, Yang Yu, and Chongjie Zhang. QPLEX: Duplex dueling multi-agent q-learning. In *Proceedings of the International Conference on Learning Representations*, 2020a.

Yihan Wang, Beining Han, Tonghan Wang, Heng Dong, and Chongjie Zhang. DOP: Off-policy multi-agent decomposed policy gradients. In *Proceedings of the International Conference on Learning Representations*, 2020b.

Lei Xi, Jianfeng Chen, Yuehua Huang, Yanchun Xu, Lang Liu, Yimin Zhou, and Yudan Li. Smart generation control based on multi-agent reinforcement learning with the idea of the time tunnel. *Energy*, 2018.

Ruihan Yang, Huazhe Xu, Yi Wu, and Xiaolong Wang. Multi-task reinforcement learning with soft modularization. In *Advances in Neural Information Processing Systems 33*, 2020.

Rui Zhao, Jinming Song, Hu Haifeng, Yang Gao, Yi Wu, Zhongqian Sun, and Yang Wei. Maximum entropy population based training for zero-shot human-AI coordination. *arXiv preprint arXiv:2112.11701*, 2021.

Ming Zhou, Jun Luo, Julian Villella, Yaodong Yang, David Rusu, Jiayu Miao, Weinan Zhang, Montgomery Alban, Iman Fadakar, and Zheng Chen. Smarts: Scalable multi-agent reinforcement learning training school for autonomous driving. *arXiv preprint arXiv:2010.09776*, 2020.

A IMPLEMENTATION DETAILS OF CSP

A.1 ALGORITHMS

We now give the pseudocode corresponding to the three training stages and deploying stage of CSP. All these stages are fully decoupled and are shown in sequential order.

Algorithm 1 CSP: Training Stage 1

Initialize teammate policies $\Pi = \{\pi^1, \dots, \pi^M\}$
while not done **do**
  for inner loops **do**
    for $i = 1, \dots, M$ **do**
      $\pi^i \leftarrow \mathrm{MARL}(\pi^i)$
    end for
  end for
  for outer loops **do**
    Update all $\pi^i \in \Pi$ to maximize J_{SVD} (Eq. 1)
  end for
end while
return Π

Algorithm 2 CSP: Training Stage 2

Input: Teammate policy population Π_{train}
Initialize scheme probing policy π_{sp}, autoencoder $\langle E_c, D_c \rangle$, and replay buffer B
while not done **do**
  Sample teammate policy π^j from Π_{train}
  Generate $\tau_{\mathrm{sp}} = \{o_t, a_t\}_{t=0}^{H}$ by executing π_{sp} coordinating with π^j
  $B \leftarrow B \cup \{\tau_{\mathrm{sp}}\}$
  Sample trajectories $\mathcal{D}$ from B
  Compute $\{\mathcal{L}_{\mathrm{pred}}(t)\}_{t=0}^{H}$ based on $\mathcal{D}$ and update $\langle E_c, D_c \rangle$ (Eq. 3)
  $\pi_{\mathrm{sp}} \leftarrow \mathrm{MARL}(\pi_{\mathrm{sp}})$, with modified reward $r'_t = r_t^{\mathrm{env}} + \alpha \mathcal{L}_{\mathrm{pred}}(t)$ (Eq. 4)
end while
return π_{sp}, E_c

Algorithm 3 CSP: Training Stage 3

Input: $\Pi_{\mathrm{train}}, \pi_{\mathrm{sp}}, E_c$
Initialize team embedding set Z; multimodal meat-policy $\pi_{\mathrm{meta}} = \{\pi_1, \dots, \pi_k\}$
for $\pi^i \in \Pi_{\mathrm{train}}$ **do**
  for n in N **do**
    Generate $\tau_{\mathrm{sp}} = \{o_t, a_t\}_{t=0}^{H}$ by executing π_{sp} coordinating with π^i
    Get team embedding $z_c = E_c(\tau_{\mathrm{sp}})$
    $Z \leftarrow Z \cup \{z_c\}$
  end for
end for
Compute best k with the Silhouette method on Z (Eq. 6)
Compute centers $\boldsymbol{\mu} = \langle \mu_1, \dots, \mu_k \rangle$ with k-means clustering on Z
while not done **do**
  Sample teammate policy π^j from Π_{train}
  Generate $\tau_{\mathrm{sp}} = \{o_t, a_t\}_{t=0}^{H}$ by executing π_{sp} coordinating with π^j
  Get team embedding $z_c = E_c(\tau_{\mathrm{sp}})$
  Pick sub-policy π_{i^*} based on $i^* = \arg\min_i \|z_c - \mu_i\|_2^2$ (Eq. 8)
  Execute π_{i^*} to coordinate with π^j and update itself: $\pi_{i^*} \leftarrow \mathrm{MARL}(\pi_{i^*})$
end while
return $\boldsymbol{\mu} = \langle \mu_1, \dots, \mu_k \rangle, \pi_{\mathrm{meta}} = \{\pi_1, \dots, \pi_k\}$

Algorithm 4 CSP: Deploying

1: **Input:** $\pi_{\text{sp}}, E_c, \boldsymbol{\mu} = \langle \mu_1, \ldots, \mu_k \rangle, \pi_{\text{meta}} = \{\pi_1, \ldots, \pi_k\}$, new policy π_{new} to adapt to
2: Generate $\tau_{\text{sp}} = \{o_t, a_t\}_{t=0}^{H}$ by executing π_{sp} coordinating with π_{new}
3: Get team embedding $z_c = E_c(\tau_{\text{sp}})$
4: Pick sub-policy π_{i^*} based on $i^* = \arg\min_i \|z_c - \mu_i\|_2^2$ (Eq. 8)
5: Execute π_{i^*} to coordinate with π_{new}

A.2 Hyperparameters and Architecture

In Stage 1, "inner loops" and "outer loops" are set to 32 and 5, respectively. MARL refers to any multi-agent reinforcement learning method, and we choose VDN (Sunehag et al., 2018) here. Population size $M = |\Pi_{\text{train}}| = |\Pi_{\text{eval}}|$ is 4 for all scenarios but 1c3s5z, which is set to 5 instead. In Stage 2, α is set to 1×10^{-6} and $|\mathcal{D}|$ is set to 32. In Stage 3, N is set to 64.

Details of neural network architectures used by CSP are provided in Fig. 6. Each policy used in CSP (i.e., teammate policy $\pi^i \in \Pi$, probing policy π_{sp}, and each sub-policy in $\{\pi_1, \ldots, \pi_k\}$) is a form of "GRU policy" with local observation history as input and outputs value estimation across its action space. The MARL framework VDN adds up all agents' local utility q_i to form Q_{tot}, which is updated to approximate the global discounted return.

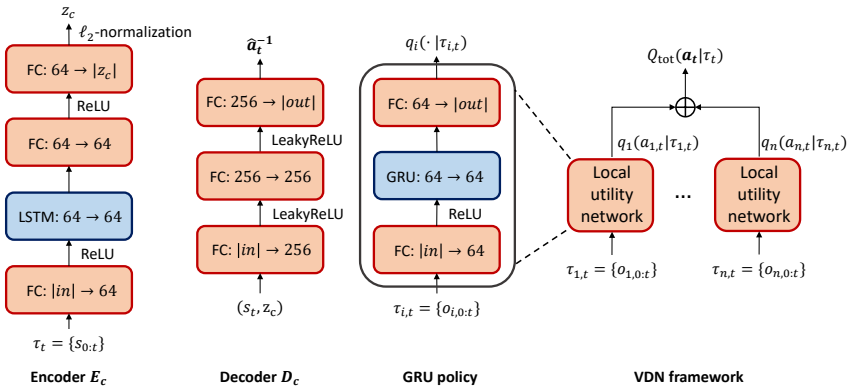


Figure 6: Network architectures of CSP, where $o_{i,t}$ and $a_{i,t}$ represent the local observation and action for agent i at timestep t, and $\tau_{i,t}$ is the trajectory of agent i's local observations until timestep t.

B Details of Environments

Level-based Foraging (LBF) (Papoudakis et al., 2021b) We use an instance of LBF, where the environment is a 20×20 grid world with 2 agents and 4 food. Each agent has a self-centered 5×5 observation range and a discrete action space for moving in four directions and collecting food. The goal of each agent is to collect all the food on the map. When food is collected, the environment returns a shared reward proportional to the food level with a total reward normalized to 1. An episode terminates if all the food is collected or reaches 50 total timesteps. To enhance the requirement for cooperation, we set the extra constraint that each food can only be collected if both agents are adjacent and perform the "collect" action simultaneously. We let CSP or baselines control one player and teammates from Π control another. The variability of coordination schemes in this environment is reflected in the order of eating all 4 food.

Predator-Prey (PP) (Böhmer et al., 2020) This environment can be considered a more complex version of LBF, where 2 predators with a 5×5 observation range are expected to hunt 4 prey in a 10×10 grid world. An episode ends when all prey is captured, or 200 timesteps have passed. The extra difficulty comes from the fact that each prey moves randomly throughout the game, so predators must constantly jointly chase the prey. We simplify the task by removing the "capture" action, and predators can capture prey with only a siege. Therefore, there will be no miss-capturing

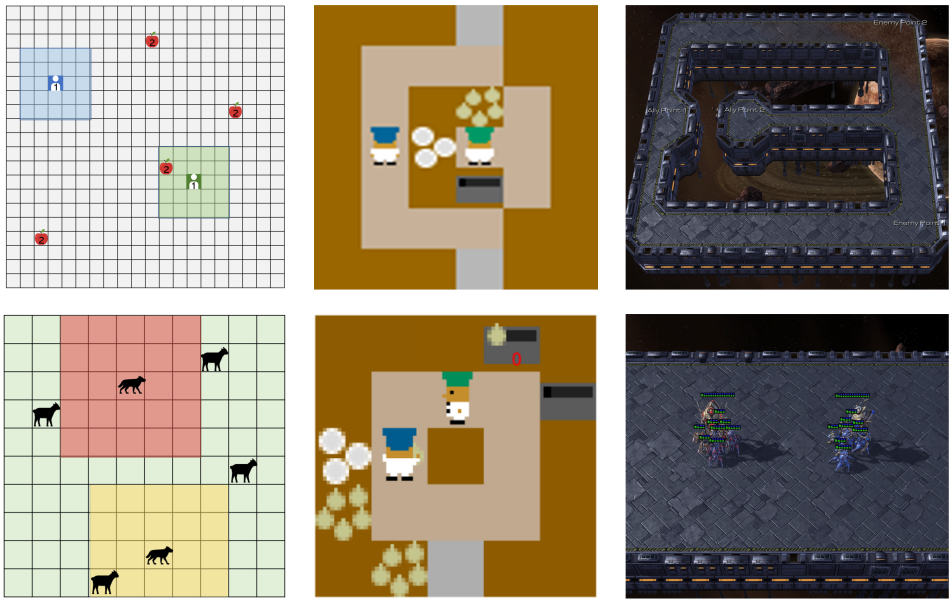


Figure 7: All the environments used in this paper. The first row from left to right: Level-based Foraging (LBF), Overcooked Forced Coordination Hard, SMAC Fork. The second row from left to right: Predator-Prey (PP), Overcooked Coordination Ring, SMAC 1c3s5z.

punishment. The coordination schemes in this environment are reflected in the order of chasing prey and the respective division of labor in the roundup of prey.

Overcooked (Strouse et al., 2021) There are 2 agents in the environment sharing a 6-dimensional discrete action space: moving in four directions, interacting with the object facing, and doing nothing. Cooking a dish requires a series of actions and a waiting period. Delivering a dish requires picking it up, moving to the correct delivery point, and putting it down. The goal of both agents is to complete as many delivery orders as possible within 400 timesteps. We set CSP or baselines to control the blue player and teammates from Π to control the green player. In the layout Coordination Ring (lower), the passage is narrow, and the two agents may clash in their pathfinding. In the layout Forced Coordination Hard (upper), only the green agent can touch the cookware, and only the blue agent can reach the delivery point. Coordination is forced in this layout since no agent can finish the task alone, and they have to adapt to teammates' preferences.

SMAC (Samvelyan et al., 2019) It is widely used as a multi-agent benchmark for its high complexity of control. Each agent can move in four cardinal directions, stop, do nothing, or select an entity to interact (heal or attack according to its type) at each timestep. Therefore, if there are n_a allies and n_e enemies in the map, the action space for each unit contains $n_a + n_e + 6$ discrete actions, and the joint action space size is $(n_a + n_e + 6)^{n_a}$. The map 1c3s5z is a standard map that requires control of 9 agents. We set CSP or baselines to control the first 4 agents, and teammates in Π control the reset 5 agents.

We specially designed a map called *Fork* (Fig. 8) for this work which requires strong coordination and has very different coordination modes. It has 2 ally spawn points on the left and middle left sides of the map and 2 enemy spawn points on the upper right and lower right corners. At the beginning of an episode, each ally spawn point generates 4 marines (long-range attack unit), and each enemy spawn point generates 6. We let CSP or baselines control allies at point 1 and teammates from Π to control allies at point 2. If both groups choose to attack the same group of enemies, it will be 8 versus 6 and is easy to win. By contrast, if two groups of teammates attack different groups of enemies, respectively, it will be 4 versus 6, and neither group will be able to defeat. Thus, intuitively there are two main kinds of cooperation modes, which we refer to as *Up* and *Down* in this work, indicating attack enemies in the corresponding direction first. The population Π is trained with CSP's stage 1. In order to make the population unbiased in attacking directions, we manually pick 8 policies from the original population with 4 *Up*s and 4 *Down*s to form the balanced population.

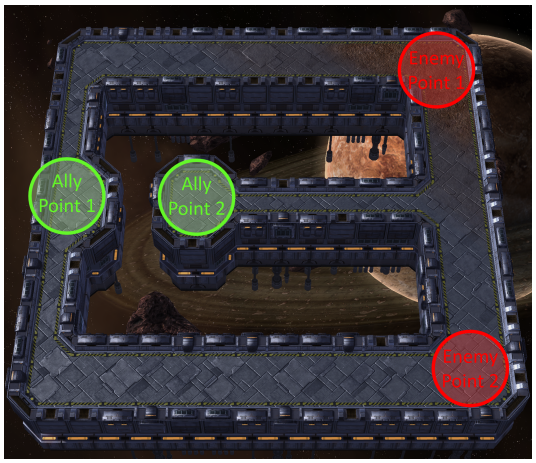


Figure 8: New SMAC map Fork.

C Details of Baselines

We compare CSP against 5 baselines. All of them are implemented with a similar GRU agent and VDN framework as CSP (Fig. 6), except that the input of GRU agents is $\tau_t = \{(o, z_c)_{0:t}\}$ which has additional embedding z_c at each timestep, so we mainly focus on how they build z_c with components shown in Fig. 9.

PBT This baseline uses simple domain randomization to train G^1 against all the teams in Π_{train}. The training set Π_{train} is required from CSP's stage 1. We think tixing this population increases the fairness of the comparison.

FCP This baseline uses similar domain randomization approach like PBT to train G^1, except that the training population is an extended version of Π_{train}. Checkpoints at one-third and two-thirds of the total training timesteps are added to Π_{train}, indicating teammates with different levels of ability.

Local information agent model (LIAM) (Papoudakis et al., 2021a). This baseline equips each agent with an encoder-decoder structure to predict other agents' observations $\boldsymbol{o}_t^{-1}$ and actions $\boldsymbol{a}_t^{-1}$ at current timestep based on its own local observation history $\tau_t = \{o_{0:t}\}$. The encoder and decoder are optimized to minimize the mean square error of observations plus the cross-entropy error of actions. The original version of LIAM considers only a single controllable agent, and predictions are made upon this agent's local observation. To fit in MARL's centralized training, we let all controllable agents make predictions based on their observations and calculate their loss, and use the mean of their loss as the final loss.

Full information agent model (FIAM) (Papoudakis et al., 2021a). This baseline is a variant of LIAM by replacing the input trajectory of local observations $\tau_t = \{o_{0:t}\}$ with the trajectory of global states $\tau_t = \{s_{0:t}\}$.

Online adaptation via inferred teamwork situations (ODITS) (Gu et al., 2022). Unlike the previous two methods that predict the actual behaviors of teammate agents, ODITS improves zero-shot coordination performance in an end-to-end fashion. It has two variational autoencoder pairs, one global and one local. The global encoder takes in state trajectory $\tau_t = \{s_{0:t}\}$ and outputs the mean and variance of a Gaussian distribution. A vector z_e indicating "global teamwork situation" is then obtained by sampling from it. The global decoder uses z_e to build the parameters z_h of a hyper-network that maps the ad hoc agent's local utility Q_i into global utility Q_{tot} to approach the global discounted return. The local encoder has a similar structure as the global encoder, except that its input is replaced with local trajectory $\tau_t = \{o_{0:t}\}$. It is updated by maximizing the mutual information of its output $\hat{z}_e$ and the global z_e. The local decoder further maps $\hat{z}_e$ into a variable z_c used as the ad hoc agent's input. The whole training is end-to-end by maximizing global return and mutual information of z_e and $\hat{z}_e$. Similar to LIAM, ODITS only considers a single ad hoc agent

setting. We modified the loss to the mean of all controllable agents' individual loss to make it fit in MARL.

To ensure fairness in the use of hidden variables, we make CSP and all baselines have the same width for z_c in each environment. Concretely, $|z_c|$ is set to 8 for Overcooked, LBF, and PP, and 64 for Fork.

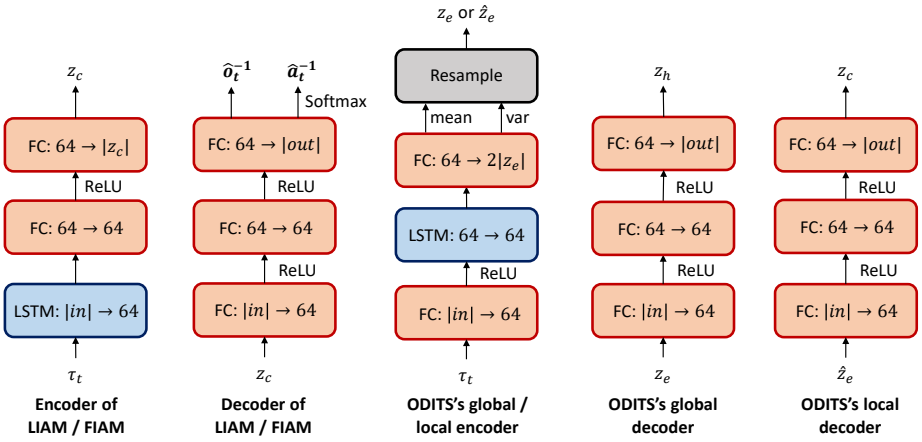


Figure 9: Network architectures of baselines.

D MORE EXPERIMENT RESULTS

D.1 EMBEDDINGS ON EACH ENVIRONMENT

To measure the quality of our scheme embedding, Fig. 10 adopts t-SNE to visualize the embedding of distinguished teams. Each color in four t-SNE sub-figures represents the embedding of distinguished teams. It shows that our scheme embedding can cluster the same coordination scheme and classify the different schemes, which shows a high quality property for team recognition and sub-policy selection.

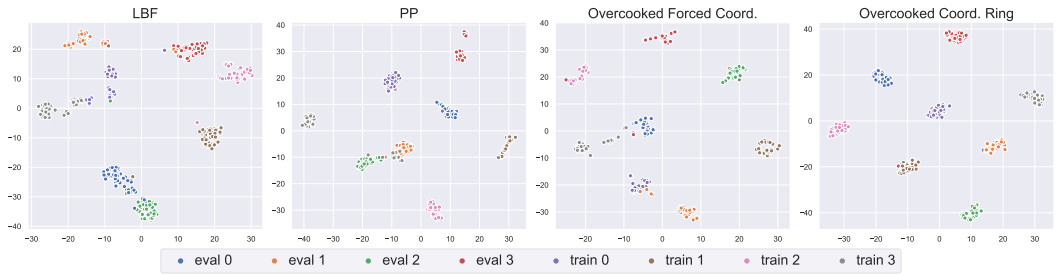


Figure 10: t-SNE of CSP's coordination scheme embeddings in different environments.

D.2 CROSS-PLAY OF DIVERSE TEAMMATES

To measure the quality and diversity of our SVD population generation in training stage 1. We concatenate all the teams in the training set Π_{train} and the evaluation set Π_{eval} together and make cross play with each other. In LBF, Overcooked, PP, and SMAC Fork benchmarks, the quantities of Π_{train} and Π_{eval} are both 4 while that of SMAC 1c3s5z is 5. We show the results of cross-play by heatmaps, where the front policies are from the training set and the latter are from the evaluation set. We normalize the score from 0 to 100 and the darker the color in heatmaps is, the lower the score of cross-play achieves. Results in Fig. 11 show that in most benchmarks agents in the population cannot work with others, as only the color of the diagonal is light, which means cross-play fails. The Fork mainly contains two coordination schemes *UP* and *Down*, and as a result, at least one team

can cross-play together with the population over 2. The heatmaps show that our SVD method can generate a diverse population to simulate the coordination scheme.

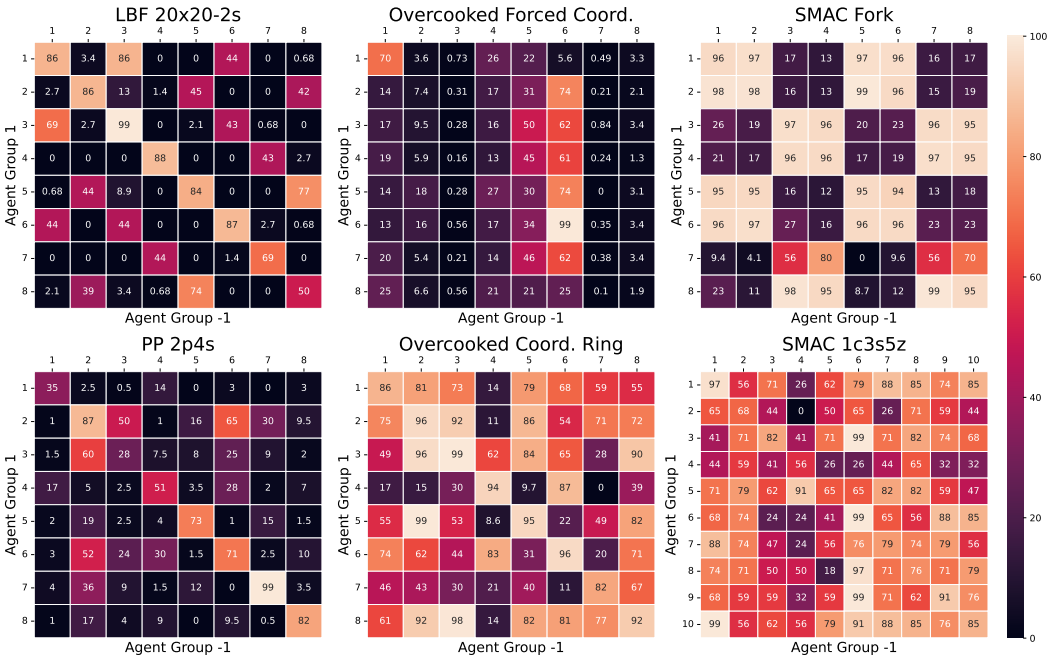


Figure 11: Cross-play performance of $\Pi = \Pi_{\text{train}} \cup \Pi_{\text{eval}}$ in all scenarios.