Structured Diversification Emergence via Reinforced Organization Control and Hierarchical Consensus Learning

Wenhao Li
East China Normal University
Shanghai, China
52194501026@stu.ecnu.edu.cn

Xiangfeng Wang*
East China Normal University and SRIAS, Shanghai, China
xfwang@cs.ecnu.edu.cn

Bo Jin*
East China Normal University and SRIAS, Shanghai, China
bjin@cs.ecnu.edu.cn

Junjie Sheng
East China Normal University
Shanghai, China
52194501003@stu.ecnu.edu.cn

Yun Hua
East China Normal University
Shanghai, China
52194501002@stu.ecnu.edu.cn

Hongyuan Zha
School of Data Science and AIRS, The Chinese University of Hong Kong, Shenzhen, China
zhahy@cuhk.edu.cn

Corresponding authors

*Corresponding authors

ABSTRACT
When solving a complex task, humans will spontaneously form teams and to complete different parts of the whole task, respectively. Meanwhile, the cooperation between teammates will improve efficiency. However, for current cooperative MARL methods, the cooperation team is constructed through either heuristics or end-to-end blackbox optimization. In order to improve the efficiency of cooperation and exploration, we propose a structured diversification emergence MARL framework named Rochico based on reinforced organization control and hierarchical consensus learning. Rochico first learns an adaptive grouping policy through the organization control module, which is established by independent multi-agent reinforcement learning. Further, the hierarchical consensus module based on the hierarchical intentions with consensus constraint is introduced after team formation. Simultaneously, utilizing the hierarchical consensus module and a self-supervised intrinsic reward enhanced decision module, the proposed cooperative MARL algorithm Rochico can output the final diversified multi-agent cooperative policy. All three modules are organically combined to promote the structured diversification emergence. Comparative experiments on four large-scale cooperation tasks show that Rochico is significantly better than the current SOTA algorithms in terms of exploration efficiency and cooperation strength.

KEYWORDS
Cooperative MARL; Diversification; Organization Control

1 INTRODUCTION
Multi-agent reinforcement learning (MARL) has been widely used and achieve fantastic performance in many application fields, like multiplayer games [21, 31], swarm robot control [27] etc. Most of the current MARL algorithms follow the centralized training and decentralized execution (CTDE, [29]) framework. In the centralized training phase, a decentralized policy needs to be learned for each agent through sharing local observations, parameters, or gradients among agents. However, these CTDE-based MARL algorithms have to consider each agent as an independent individual during the training procedure. Although these individuals can transmit information explicitly or implicitly to achieve collaboration, most of them are usually learned through an end-to-end blackbox scheme, which raises the difficulty to obtain meaningful communication protocols [45]. This makes it difficult for multi-agents to explore and collaborate effectively.

When humans perform tasks in an unknown environment, diverse teams instead of individuals, are usually used as the basic unit to make up for limited individual abilities. As for the nonstationary of the external environment and the difficulty of the related task, the team can be restructured accordingly. In terms of MARL, this means that the agents must have the ability to dynamically team-up. To improve the efficiency of exploration in an unknown environment, the behavior of different teams need to be sufficiently diverse; Further considering the capacity limitations of a single agent, the agents from the same team need to cooperate closely to improve the efficiency of task completion. In this paper, we call this inter-team diversification and intra-team cooperation ability the structured diversification emergence.

The core assumption is that the behavior of an agent is determined by both the team goal and its perception of the environment, while diverse team goals and consistent environmental perception can lead to structured diversification emergence. Designing an efficient algorithm for the agent to learn structured diversification emergence ability, we need to answer the following important questions: 1) Adaptive organization control: how do the agents form teams spontaneously, and the team composition can dynamically change to adapt to the external environment? 2) Structured diversification: How to maintain diversity in the behavior of different teams and to form tight cooperation between the agents within the team? 3) Behavior emergence: How to combine the above two processes organically? Let us conduct an in-depth analysis.

Adaptive organization control. In a cooperative task, before taking actions, an individual will first assess whether her ability is sufficient to complete the task based on her observations of the
This motivates us to impose consensus constraints on the agents’ behavior, but it only imposes perceptual consensus constraints on the organization control and hierarchical consensus learning. As shown in the MARL algorithm, so-called collaboration, which can’t lead to agents’ diverse behaviors.

Figure 1: The ROCHICO architecture. The algorithm is divided into three modules: organization control, hierarchical consensus, and decision. Each agent first makes a decision whether to team up with neighbors through the organization control module based on local observations. Next, the team intention generator and individual intention generator, trained based on the hierarchical consensus constraints, generate team intentions and individual intentions according to the teaming results. Finally, the decision-making module generates structured diversify policies based on team intentions and individual intentions.

Environment. If not, she will seek help from other individuals. Besides, since the environment is locally observable, a team would be more powerful by fetching the information from multiple individuals. Therefore, some existing works introduce teams in the MARL algorithms. However, these teams are constructed either through heuristics [31, 32] or through end-to-end blackbox optimization [4, 16, 39]. Compared with these methods, our work can be summarized as task allocation [6, 24, 33], relation adaption [9, 10, 19] and coalition formation [2, 26, 48]. For the reason that these methods are not RL-based and we will not expand here, more details can be found in Ye et al.

Figure 1. ROCHICO consists of three modules: organization control, hierarchical consensus and decision. First, the organization control module models the multi-agent organization control problem as a partially observable stochastic game (POSG), and introduces independent MARL to obtain a dynamic and autonomous teaming strategy. Second, the hierarchical consensus module obtains team intentions and individual intentions through contrastive learning and unsupervised learning. Structured diversification emerges by imposing hierarchical consensus constraints on hierarchical intentions. Finally, the decision module outputs the diversify cooperative policies based on results of reinforced organization control and hierarchical consensus learning to the environment, and feeds back external rewards to the other two modules, and decision module itself, so that all modules can be combined organically.

Our contributions mainly consist of the following folds: 1) We model the organization control of the multi-agent as an independent learning task, which enables the agent to autonomously and adaptively team up based on environmental feedback. 2) We impose hierarchical consensus constraints on hierarchical intentions obtained by the novel introduced contrastive learning and unsupervised learning auxiliary tasks to encourage the structured diversification emergence. 3) Performance experiments on various large-scale cooperative tasks show that ROCHICO is significantly better than the current SOTA algorithms in terms of exploration and cooperation efficiency.

2 RELATED WORKS

2.1 Organization Control Mechanism

Organization control is defined as a mechanism or a process that enables a system to change its organization without explicit command during execution time [5]. The most relevant existing self-organization mechanisms to our work can be summarized as task allocation [6, 24, 33], relation adaption [9, 10, 19] and coalition formation [2, 26, 48]. For the reason that these methods are not RL-based and we will not expand here, more details can be found in Ye et al.
Many cooperative tasks require agents to take different behaviors to achieve higher degree of task completion. Behavior diversification can be handcrafted or emerged through multi-agent interaction. Handcrafted diversification is widely studied at task allocation or role assignment. Heuristics mechanisms \cite{3, 24, 36, 41} assign a specific task or a pre-defined role to each agent based on its goal (capability, visibility) or by searching. Shu and Tian \cite{40} establishes a manager to assign suitable sub-tasks to rule-based workers with different preferences and skills. All these methods require that the sub-tasks and roles are pre-defined, while the worker agents are rule-based at the same time.

Recently, the emergent diversification was introduced to single-agent RL \cite{7, 11–13} with the purpose to learn reusable diverse skills in complex and transferable tasks. In MARL, McKee et al. \cite{28} introduces diversity into heterogeneous agents to learn more generalized policies for solving social dilemmas. Wang et al. \cite{46} learns a role embedding encoder and a role decoder simultaneously. However, no mechanism guarantees the role decoder can generate different parameters and generate diversity policies accordingly, taking as input different role embeddings. Jiang and Lu \cite{17} establishes an intrinsic reward for each agent through a well-trained probabilistic classifier. The intrinsic reward makes the agents more identifiable and promote diversification emergence. Based on Eysenbach et al. \cite{7}, learning low-level skills for each agent in hierarchical MARL is considered in Lee et al. \cite{20}, Yang et al. \cite{47}. The high-level policy can utilize coordinated low-level diverse skills, but the high-level policy does not consider diversity.

3 PRELIMINARIES

Cooperative POSGs. POSG \cite{14} is denoted as a seven-tuple based on the stochastic game (or Markov game) as

\[
\langle X, S, (\mathcal{A}^i)_{i=1}^n, \{O^i\}_{i=1}^n, P, E, \{R^i\}_{i=1}^n \rangle,
\]

where \(n\) denotes agents total number; \(X\) represents the agent space; \(S\) contains a finite set of states; \(\mathcal{A}^i, O^i\) and denote a finite action set and a finite observation set of agent \(i\) respectively; \(\mathcal{A} = \mathcal{A}^1 \times \mathcal{A}^2 \times \cdots \times \mathcal{A}^n\) is the finite set of joint actions; \(P(s'|s,a)\) denotes the Markovian state transition probability function; \(O^i = O^1 \times O^2 \times \cdots \times O^n\) is the finite set of joint observations; \(E(a|s)\) is the Markovian observation emission probability function; \(R^i : S \times \mathcal{A} \times S \rightarrow R\) denotes the reward function of agent \(i\). The game in POSG unfolds over a finite or infinite sequence of stages (or timesteps), where the number of stages is called horizon. In this paper, we consider the finite horizon case. The objective for each agent is to maximize the expected cumulative reward received during the game. For a cooperative POSG, we quote the definition in Song et al. \cite{42},

\[
\forall x \in X, \forall x' \in X \setminus \{x\}, \forall \pi_x \in \Pi_x, \forall \pi_{x'} \in \Pi_{x'}, \frac{\partial R(x)}{\partial x} \geq 0,
\]

where \(x\) and \(x'\) are a pair of agents in agent space \(X\); \(\pi_x\) and \(\pi_{x'}\) are the corresponding policies in the policy space \(\Pi_x\) and \(\Pi_{x'}\) respectively. Intuitively, this definition means that there is no conflict of interest for any pair of agents.

QMI. The QMIX \cite{35} algorithm, which is the follow-up work of the VDN \cite{43}, is the current SOTA of cooperative MARL. QMIX claims that the the \(Q\)-value functions before \((Q_{tot})\) and after \((Q_a)\) decomposition should satisfy the following constraints:

\[
\frac{\partial Q_{tot}}{\partial Q_a} \geq 0, \forall a \in A.
\]

QMIX employs a hypernetwork-based mixing network to promote the two \(Q\)-function satisfying the above condition. Because of the non-linear mixing network, QMIX can outperform VDN.

4 ALGORITHMS

The overall framework of the proposed algorithm is shown in Figure 1, which can be divided into three modules. The organization control module receives local observations of all agents and makes adaptive teaming decisions use the traditional graph theory algorithm \cite{44}. The hierarchical consensus module will then generate the team intention and the individual intention based on the obtained teaming results. The hierarchical consensus constraints are established on the above hierarchical intentions to promote the structured diversification emergence. Finally, the decision module outputs the structured diversification policies through a cooperative MARL algorithm. However, it should be noted that the organization control process is not differentiable, so we cannot train the overall model by the end-to-end scheme. We draw on the communication MARL algorithms \cite{8, 18, 39}, while passing the external rewards to the organization control module and decision module separately, so as to realize the joint training.

4.1 Organization Control Module

If we consider all agents as a graph \(G(V, E)\), each agent is a node\(^1\) \(v \in V\) in the graph \(G\). The edge \(e \in E\) indicates whether the two agents connected by \(e\) belong to the same team. The main purpose of the organization control module is to determine the connections (edges) between agents. Then, we can naturally view the connected components as teams, while searching for connected components can be done efficiently by traditional graph theory algorithms \cite{44}.

In the beginning, if the problem in organization control module is modeled as a single-agent RL problem, \(O(n^2)\) edges need to be determined if the graph contains \(n\) nodes, and the size of the action space is \(O(2^{n^2})\). This makes it impossible to scale to a larger multi-agent scenario (e.g., tens of agents). Therefore, we model

\(^1\)Except for explicit emphasis, we will no longer distinguish between the two terms node and agent below.
the organization control problem as a MARL problem, while each node is considered as an agent. However, if each agent needs to determine its connections with all other nodes, the action space is still very large and be $O(2^n)$. Therefore, inspired by other team-based MARL algorithms, like [15, 16], we only consider the closest $m$ other agents. This idea is intuitive but effective, because it is reasonable to team up the agents according to the adjacency of the spatial position for most tasks.

Formally, the organization control problem can be modeled as a cooperative POSG, which is denoted as:

$$M_u := \left\{ N_u, S_u, \{ A_u^i \}_{i=1}^n, \{ O_u^i \}_{i=1}^n, \mathcal{P}_u, \mathcal{E}_u : \{ R_u^i \}_{i=1}^n \right\}.$$ 

We set the action of agent $i$, $a_u^i \in A_u^i$, as a $m$-dimension binary vector, denoting the connection action to $m$-nearest agents $\{v^j | j \in N_u(i)\}$ ($m \ll n$) according to its local observation $o_u^i$, where the subscript $u$ stands for ‘unorganized’. This kind of MARL problem will suffer that agent $i$ decides to connect with agent $j$ but agent $j$ does not want to connect with agent $i$ or vice versa, which means:

$$a_u^i[j] \neq a_u^i[i], \quad \text{with} \quad i \in N_u(j), \ j \in N_u(i).$$

Considering the positive effect of prosociality on promoting the cooperation of agents [32], we use the weakly connected graph $\mathcal{G}_u = (V_u, E_u)$ to form teams, which is established by converting directed edges into undirected edges, i.e.,

$$e(i, j) = a_u^i[j] \lor a_u^i[i], \quad \text{with} \quad i \in N_u(j), \ j \in N_u(i),$$

where $\lor$ denotes "OR" operation. Finally, the Tarjan’s algorithm is employed for searching weakly connected components with worst-case time complexity as $O(|V_u| + |E_u|)$ [44].

In addition to the external rewards $r_u^j$, from the environment, we also introduce additional intrinsic rewards to train the agents. Specifically, in order to strengthen training stability without causing excessive fluctuation on the graph structure, the novel structural consistency intrinsic reward $r_{ik}^i$ for each agent $i$ is defined by:

$$r_{ik}^i = \frac{1}{|N_u(i)|} \cdot \text{GED} \left( \mathcal{G}_u \ (N_u(i) \lor i), \ G'_u \ (N_u(i) \lor i) \right),$$

where $\text{GED}(\cdot, \cdot)$ represents the graph edit distance [37]. $\mathcal{G}_u \ (N_u(i) \lor i)$ and $G'_u \ (N_u(i) \lor i)$ represent the sub-graph only contains node $i$ and its $m$-nearest neighbors before and after take action $a_u^i$ respectively. The total reward for each agent $i$ is:

$$r_{ik}^i = r_{ik}^c + a_u^i r_{ik}^l,$$

where $a_u^i$ indicates the strength of constraint for structural consistency. For the organization control problem, the goal is to maximize the summation of all agents’ expected accumulated rewards, which can be denoted as follows:

$$\max \ J_u = \frac{1}{n} \sum_{i=1}^n \mathbb{E}_{a_u^i} \left[ r_{ik}^i + (\tau_u) \right].$$

Further to raise training efficiency, we use independent learning combined with DQN and parameter sharing mechanism. [30] has shown the significant performance of independent learning on multi-agent tasks. Specifically, we can minimize following TD(0) error for each agent $i$, i.e.,

$$L_i^i(\theta_u) = \mathbb{E}_{(a_u^i', o_u^i', r_u^i', o_u^i)}[Q_{\theta_u}(o_u^i, a_u^i) - y]^2.$$

where $y = r_{ik}^i + \gamma \max_{a_u'^i} Q_{\theta_u}(o_u^i', a_u'^i)$ and $\theta_u, \hat{\theta}_u$ parameterize $Q$ function and target $\bar{Q}$ function separately. Finally, the overall objective function is:

$$\min \ L_u^i(\theta_u) = \sum_i L_u^i(\theta_u).$$

### 4.2 Hierarchical Consensus Module

The goal of the hierarchical consensus module is to achieve efficient multi-agent exploration and cooperation, that is, structured diversification emergence. To this end, we put forward the concepts of hierarchical intentions, i.e., team intentions and individual intentions. Then the structured diversification emergence can be achieved through contrastive learning and hierarchical consensus learning. The hierarchical consensus module is composed of two sub-modules: team intention generator (Figure 2) and individual intention generator (Figure 3).

Figure 2: The network structure of the team intention generator in the training phase. Two teams are close to each other and one team is far apart. The team intention is based on the representations of all agents in the team, which are aggregated through a DeepSet network, and further is calculated through the intention encoder. The loss function consists of both contrastive loss and self-supervised loss.

#### 4.2.1 Team Intention Generator

The team intention should have the following characteristics: 1) the team intention must reflect the team behavior (or goal), so that it should be generated based on the joint observation of all agents within the team; 2) to improve the exploration efficiency of the team in an unknown environment, the team intentions among different teams must be diverse; 3) the team intention can reflect the team behaviors in a short period time, so we can predict the future observation of the agents within the team based on the team intention. In addition to the second one, the agents in each team make decisions independently, so they must access the local information of the other agents in the same team to achieve diverse behaviors. Considering the difficulty of communication learning [22], we use the global state of environment additionally to assist the generation of team intention, similar as Rashid et al. [34]. The global state is only used in the training phase since we use a CTDE framework.
Formally, in order to generate the team intention for each team, the state encoder \( f_G(\cdot) \) parameterized by \( \mu \) recovers the joint observation \( \bar{e}_k = (e_{t_s}^{k,1}, e_{t_s}^{k,2}, \ldots, e_{t_s}^{k,n_k}) \) of all \( n_k \) agents in team \( k \), and together with the global state \( s_t \) (such as the minmap of the environment) at timestep \( t \). The generated agent state embeddings of each team \( (e_{t_s}^{k,1}, e_{t_s}^{k,2}, \ldots, e_{t_s}^{k,n_k}) \) are fed into a DeepSet network [50], i.e., \( f_G(\cdot) \) parameterized by \( \nu \) to generate the team embedding \( e^k_t \):
\[
e^k_t = f_G \left( \nu \left( e_{t_s}^{k,1}, e_{t_s}^{k,2}, \ldots, e_{t_s}^{k,n_k} \right) \right).
\]
Finally, the team intention encoder \( f_\omega(\cdot) \) parameterized by \( \omega \) receives team embedding \( e^k_t \) and global state \( s_t \) again to generate the team intention \( c^k_t \):
\[
c^k_t = f_\omega \left( e^k_t, s_t \right).
\]
In order to generate diverse team intentions, we model the team intention generation as a contrastive learning problem. First, the average spatial position \( \langle x^k_t, y^k_t \rangle \) of the team \( k \) at each timestep \( t \) can be calculated based on the spatial position of the all agents within the team \( \langle x_{t_s}^{k,1}, y_{t_s}^{k,1} \rangle, \ldots, \langle x_{t_s}^{k,n_k}, y_{t_s}^{k,n_k} \rangle \). Combined with the current timestep, \( t_s \), a spatiotemporal feature representation \( e^{k,1}_{t_s} = \langle x_{t_s}^{k,1}, y_{t_s}^{k,1}, t_s \rangle \) of the team \( k \) can be obtained. The Euclidean distance in the spatiotemporal space is used to measure the distance between team \( k \) and \( l \), i.e.,
\[
d(k, l) = \| e^k_{t_s} - e^l_{t_s} \|^2_2.
\]
A simple version directed graph \( G_t((V_t, E_t)) \) can be established in team level, where \( V_t \) denotes the node set of \( G_t \) and each node \( v^k_t \in V_t \) represents a team \( k \). The \( E_t \) denotes the edge set and the two teams connected by edge \( e^k_t \in E_t \) have similar intentions. For each edge in edge set \( E_t \) of \( G_t \), we have
\[
e(k, l) = \begin{cases} 1 & \text{if } \min_{v \in v^k_t} d(k, v) = \min_{v \in v^l_t} d(l, v) \end{cases}.
\]
There is an edge connection between \( k, l \) two nodes if anyone is the nearest neighbor of the other. Similar to the organization control module, we use Tarjan’s algorithm to find all the weakly connected components in the directed graph \( G_t \). Teams that belong to the same weakly connected component will be assigned the same label. The team intention \( c^k_t \) is set to be the feature and the Tarhan’s algorithm labeled \( y^k_t \) is set to be the label, which constructs a supervised training set \( \{(c^k_t, y^k_t)\} \). After combining with the triplet loss [38], we could construct a contrastive learning problem by minimize following objective for each team \( k \):
\[
L^k_c(\phi, \psi, \varphi) = \mathbb{E} \left[ \max \left( 0, \| c^k_t - e^k_{t_s} \|^2_2 - \| c^k_t - c^l_t \|^2_2 + m \right) \right],
\]
where \( m \) is a margin parameter, \( k, u \) share the same label \( y^k_t = y^u_t \) and \( k, v \) have different labels \( y^k_t \neq y^v_t \), \( u \) and \( v \) are sampled from the weakly connected component same as \( k \) and different from \( k \) respectively.

In addition, in order to enhance the impacts of team intention on the team behavior, besides using the team intention as the input of the following individual intention generator, we introduce another self-supervised task. The intention decoder \( f_x(\cdot) \) parameterized by \( \xi \) recieves team intention \( c^k_t \) of team \( k \) at current timestep \( t \) as input, and output the prediction of joint observation
\[
\hat{o}^k_{t+1} = (\hat{o}^k_{t+1,1}, \hat{o}^k_{t+1,2}, \ldots, \hat{o}^k_{t+1,n_k})
\]
at next timestep. Formally, we minimize following regression objective function for each team \( k \) so as to formulate the self-supervised task, i.e.,
\[
L^k_r(\xi) = \mathbb{E} \left[ \frac{1}{n_k} \sum_{i=1}^{n_k} \left( f_x(\xi) - o^k_{t+1,i} \right)^2 \right].
\]
The network structure is shown in Figure 2 and the overall problem of the team intention generator can be formulated as follows:
\[
\min \mathbb{E} \left[ \sum_{k} L^k_r(\xi) + \lambda_{tg} \cdot L^k_c(\phi, \psi, \varphi) \right].
\]

### 4.2.2 Individual Intention Generator
With the team intention as the guidance for the diverse behavior between teams, the hierarchical consensus module aims to realize the structured diversification emergence through individual intentions consensus. The key idea to achieve better cooperation within the team is that all agents in the same team should have a consistent cognition of the surrounding environment and the task. The individual intention generation is divided into the following three steps:

1. The individual encoder \( g_\phi(\cdot) \) parameterized by \( \phi \) encodes the initial observation \( o^k_{t-i} \) (agent \( i \) in team \( k \) at timestep \( t \)) into individual embedding \( h^k_{t-i} \) (yellow rounded square in Figure 3(b)), and extracts the individual cognition \( \chi^k_{t-i} \) (blue rounded square in Figure 3(b)) by
\[
\chi^k_{t-i} = g_\phi(h^k_{t-i})
\]

2. We regard agents in the same team as nodes in a new fully-connected graph. A GNN \( g_\psi(\cdot) \) parameterized by \( \psi \) then is introduced to further aggregate all individual embeddings \( h^k_{t-i} \) (blue rounded square in Figure 3(b)) through
\[
\hat{h}^k_{t} = g_\psi(\chi^k_{t-i}^k).
\]

3. The variational encoder \( g_\varphi(\cdot) \) (shown in Figure 3(c)) recieves the concatenated feature \( (\chi^k_{t-i}, \hat{h}^k_{t}) \) (consist of individual cognition \( \chi^k_{t-i} \) and team intention \( \hat{h}^k_{t} \)) as input, and output the individual intention \( \zeta^k_{t-i} \) (green rounded square in Figure 3(b)) through
\[
\zeta^k_{t-i} = g_\varphi(\chi^k_{t-i}^k, \hat{h}^k_{t}^k).
\]

Then, we impose hierarchical consensus constraints on the generated individual intentions, which leads to optimizing the following function for each agent \( i \) in team \( k \):
\[
L^k_i(\phi, \psi, \varphi) = \mathbb{E} \left[ \frac{1}{n_k} \sum_{j} KL \left[ q_{\phi}(\zeta^k_{t-i}^j | \zeta^k_{t-i}^k) \| q_{\phi}(\zeta^k_{t-i}^j | o^k_{t-i}) \right] \right] + KL \left[ q_{\phi}(\zeta^k_{t-i}^k | h^k_{t-i}) \| p(\zeta^k_{t-i}^k) \right].
\]
Recall the loss function of the variational autoencoder
\[
L^k_i(\varphi) = \mathbb{E} \left[ \| o^k_{t-i} - \hat{h}^k_{t} \|^2 + KL \left[ q_{\phi}(\zeta^k_{t-i}^k | h^k_{t-i}) \| p(\zeta^k_{t-i}^k) \right] \right].
\]
The prior distribution \( p(\zeta^k_{t-i}) \) in second term can be replaced by our above consensus constraints, which leads to
\[
L^k_i(\varphi) = \mathbb{E} \left[ \| o^k_{t-i} - \hat{h}^k_{t} \|^2 + \frac{1}{n_k} \sum_{j} KL \left[ q_{\phi}(\zeta^k_{t-i}^j | \zeta^k_{t-i}^k) \| q_{\phi}(\zeta^k_{t-i}^j | o^k_{t-i}) \right] \right].
\]

\footnote{Since the reparameterization trick is used in order to enable the backpropagation during implementation, it is reasonable to write the equal sign here.}
The overall problem of individual intention generation can be formulated as
\[
\min L^I_{\phi}(\phi, \psi, \varphi) := \sum_{t=1}^{n} L^T_{\phi}(\phi, \psi, \varphi).
\]

4.3 Decision Module
Before outputs the agents’ policies, it should be noted that the team intention to affect the final policies indirectly. To affect the agents’ behaviors more directly and efficiently, we introduce another intrinsic team reward and utilize the QMIX [34] technique to assign the rewards to all agents. Formally, the intrinsic team reward \( r^k_{t,i} \) is defined as:
\[
r^k_{t,i} = \sum_{j \in k} \|c^k_{t,i} - c^j_{t,i}\|_2^2.
\]
We additionally use the external reward \( r^e_t \) to learn the local Q function, as a result the local Q function will be trained based on two different reward signals \( r^k_t \) and \( r^e_{t,i} \). To make the behavior of the agents in each team have a certain diversity, we firstly combine the intrinsic reward \( r^k_{t,i} \) with the individual cognition \( \chi^k_{t,i} \) as the input of the QMIX algorithm, and generates the local Q-value. Further we optimize the local Q-value function by minimize following TD(0) error:
\[
L^I_{\phi}(\theta^I_{t}) = \mathbb{E}_{(\zeta^k_{t,i}, x^k_{t,i}, a^k_{t,i}) \sim D} \left[ (Q^k_{\theta^I_{t}}(\zeta^k_{t,i}, x^k_{t,i}, a^k_{t,i}) - y^I)^2 \right].
\]
where \( y^I = r^I_t + \gamma \max_{a^k_{t,i+1}} Q^k_{\theta^I_{t}}(\zeta^k_{t,i+1}, x^k_{t,i+1}, a^k_{t,i+1}) \) and \( \theta^I_{t}, \theta^I_{t+1} \) parameterize local Q function and local target Q function of agent \( t \) respectively. For each team \( k \), the team joint Q-value can be denoted as
\[
Q^k_{\theta^I_{t}}(\zeta^k_{t,i}, x^k_{t,i}, a^k_{t,i}) = Q^k_{\theta^I_{t}}(\zeta^k_{t,i}, x^k_{t,i}, a^k_{t,i}), \ldots, Q^k_{\theta^I_{t}}(\zeta^k_{t,i}, x^k_{t,i}, a^k_{t,i})
\]
and we also optimize the team joint Q-value function by minimize following TD(0) error:
\[
L^T_{\phi}(\theta^T_{t}) = \mathbb{E}_{(\zeta^k_{t,i}, x^k_{t,i}, a^k_{t,i}) \sim D} \left[ (Q^k_{\theta^T_{t}}(\zeta^k_{t,i}, x^k_{t,i}, a^k_{t,i}) - y^T)^2 \right].
\]
where \( y^T = r^T_t + \gamma \max_{a^k_{t,i+1}} Q^k_{\theta^T_{t}}(\zeta^k_{t,i+1}, x^k_{t,i+1}, a^k_{t,i+1}) \) and \( \theta^T_{t}, \theta^T_{t+1} \) parameterize joint Q-value function and joint target Q-value function of team \( k \) respectively. The overall learning problem of decision module is
\[
\min L^Q(\theta^I_{t}^I, \theta^T_{t}) = \sum_t L^I_{\phi}(\theta^I_{t}) + \lambda_{Qmix} \sum_k L^T_{\phi}(\theta^T_{t}).
\]

5 EXPERIMENTS
5.1 Environments
We evaluate the algorithm performances on four large-scale cooperative environments, including Pacmen, Block, Pursuit and Battle. More details can be found in appendix.

Pacmen. 64 agents initialized at the maze center and 64 dots scatter randomly at four corners of the squared map. Agents get the reward by eat dots. The dots are distributed in different corners, the agent needs to team up and travel to different corners to eat more dots.

Block. There are 32 blockers and 32 blockees who have superior speed than the blockers. There also are 64 foods initialized at one side of the squared map. Blockers and blockees are only rewarded by eat foods. Since blockee runs faster than blocker, blocker needs to learn diverse policies to block blockees and eat food simultaneously.
**Pursuit.** There are 64 predators and 64 preys who have superior speed than the predators. Since the prey runs faster than the predators, the predators need to learn to round up through structured diversification policies.

**Battle.** 64 agents learn to fight against 64 enemies who have superior abilities than agents. As the hit point of enemy is 10 (more than single agent’s damage), agents need to continuously cooperate to kill the enemy. All environments are implemented by MAgent [53].

### 5.2 Baselines

IDQN is chosen as the baseline. Due to the connection between Rochico with QMIX and NCC-Q, we also compare these two methods as baselines. However, both QMIX and NCC-Q are not designed for large-scale scenarios. Therefore, we first randomly split all agents to multiple teams, and use QMIX and NCC-Q algorithms in each one. See Appendix for detailed hyperparameter settings.

### 5.3 Performance Comparison

**Pacmen** and **Block** need to pay more attention to the division of labor of agents than Pursuit and Battle environments, so the diversity of policies and the degree of collaboration between agents should have a greater impact on the final performance. As can be seen from Figures 4(a) and Figures 4(b), since IDQN does not take into account the cooperation between agents, the weak individual ability limits the overall task completion. Compared with IDQN, QMIX, and NCC-Q, which ensure the cooperation between agents explicitly, achieve a certain degree of diversification of policies with better performance. In addition, NCC-Q explicitly imposes consensus constraints on the policies within the team, thereby it can achieve better collaboration. However, QMIX and NCC-Q both use predefined teaming strategies, they cannot dynamically adapt to the non-stationary environment. **Rochico** outperforms all the other three methods: it can perform adaptive teaming because of reinforced organization control; and it can achieve tight cooperation even when the teams change dynamically because of the hierarchical consensus learning.

For **Pursuit** and **Battle** environments, more attentions are paid on the flexibility of agents’ collaboration policies. **Rochico** can achieve the best performance in these two more complex environments, as seen from Figure 4(c) and Figure 4(d). The ability of the individual agent is more important in Pursuit, as a result, the performance of IDQN is better than QMIX and NCC-Q: the flexible switching ability of the individual agent is more important in Battle, the performance of IDQN becomes poor.

### 5.4 Ablation Study

In this part, we first conduct an ablation analysis on the three important components of the Rochico algorithm: hierarchical consensus constraints, structural consistency intrinsic reward in the organization control module, and the intrinsic reward in the decision module. Here we choose the Pursuit environment because all comparisons have the most significant difference in this environment. The detailed comparisons can be summarized as follows:

1. We can see from Figure 4(e) that removing the structural consistency intrinsic reward (i.e., Rochico-c) has the least impact on the algorithm performance. This is because the edge between two nodes in the organization control module is determined by the OR operation, which makes it difficult for the edge existing at the previous timestep to disappear due to randomness at the next timestep. This indirectly realizes a certain degree of regularity for the stability of the graph structure.

2. It can be seen from the Figure 4(e) that after removing the hierarchical consistency constraint (Rochico-g, shown in Figure 3(a)), the performance of the Rochico algorithm has dropped significantly. Although IDQN performs better than QMIX and NCC-Q in Figure 4(e), this does not mean that cooperation is unnecessary in Pursuit environment. QMIX and NCC-Q use a predefined teaming strategy instead of the adaptive strategy in Rochico, which limits the capabilities of the agents.

3. After removing the intrinsic reward in decision module (Rochico-i, replaced by sum of local rewards of all agents in the team), the performance of Rochico-i has a great decline from Rochico. This shows that use of team intention to promote diversification indirectly is difficult. Through utilizing the intrinsic rewards based on intention difference between teams, the diversified policy emergence will become easier.

|                  | Rochico-C | Rochico-G | Rochico-I | Rochico-1 | Rochico-3 |
|------------------|-----------|-----------|-----------|-----------|-----------|
| Pacmen           | 951(±261) | 689(±37)  | 805(±43)  | 691(±52)  | 767(±74)  |
| Block            | 687(±45)  | 651(±38)  | 580(±52)  | 605(±56)  | 622(±44)  |
| Pursuit          | 2252(±420)| 2012(±385)| 1588(±394)| 2408(±403)| 2403(±345)| 2548(±380)
| Battle           | 2221(±24)| 199(±33)  | 169(±23)  | 194(±26)  | 180(±32)  | 232(±27)  |

**Table 1:** The average episode rewards of ablation algorithms in test environment. The mean and standard variance are calculated under 5 random seeds.

The ablation analysis on the decision of the organization control module is shown in Figure 4(f). The number suffix $k = {1, 2, 3}$ behind the Rochico indicates that each agent needs to decide at the same time whether to form a team with the nearest $k$ agents. $k = 2$ is the default setting of Rochico algorithm. It can be seen from Figure 4(f) that Rochico-2(Rochico) outperforms both Rochico-1 and Rochico-3. Rochico-1 has a smaller range which will weaken the connection between agents, and less effective collaboration between agents can be established; Rochico-3 has a larger range which will make the teams too fixed, and unable to flexibly adapt to the non-stationary environment. The average episode rewards of the above ablations in the test environment are shown in Table 1.

### 5.5 Emergence Behavior Analysis

Figure 5 shows the changing of the averaged team number in the training process. The curves corresponding to all environments show a downward trend in the training process, which means that the organization control module does learn teaming strategies that can promote cooperation. It can be seen from the Figure 5 that the Pursuit environment has the largest averaged team number. The Pursuit environment is more focused on the ability of the agent itself, which is also consistent with the previous analysis.

Further, we select the Pacmen environment to deeply analyze the teaming strategy of the organization control module and the individual intentions at different stages for task completion. For the convenience of the presentation, we conduct the experiment in the
Figure 4: The performance comparison and ablation study of the Rochico algorithm.

Figure 5: The averaged team number changing.

Pacmen with only 12 agents. The different colors in Figure 6 represent different teams (green represents food). The following two-dimensional scatter plot is the visualized result of the t-SNE [23] algorithm by reducing the dimensionality of the individual intentions at the corresponding timestep. It can be seen from Figure 6 that the points belonging to the same team show obvious aggregation, while the points belonging to different teams are far apart. As the task progresses, if the difference between the sub-tasks completed by different teams becomes larger, the corresponding points will be farther away.

Figure 6: The relationship between team pattern and team intention in Pacmen environment.

6 CONCLUSION

In this paper, in order to improve the efficiency of multi-agent exploration and collaboration in complex tasks, we propose a MARL framework Rochico based on reinforced organization control and hierarchical consensus learning. In the organization control module, we model the multi-agent organization control problem as a cooperative POSG and use an independent MARL algorithm to output an adaptive teaming strategy. In the hierarchical consensus module, based on the auxiliary tasks of contrastive learning and self-supervised learning, the exploration efficiency and the collaboration efficiency of multi-agents are improved through hierarchical consensus learning. The comparison between Rochico and current SOTA cooperative MARL algorithms in four large-scale cooperative multi-agent environments shows that our algorithm can complete complex tasks more efficiently through richer policies diversity and tighter agents collaboration.

7 ACKNOWLEDGE

This work was supported in part by National Key Research and Development Program of China (No. 2020AAA0107400), NSFC (No. 12071145), STCSM (No. 18DZ2270700 and 20511101100), the Open Research Projects of Zhejiang Lab (NO.2021KE0AB03) and a grant from Shenzhen Institute of Artificial Intelligence and Robotics for Society.
REFERENCES

[1] Sherief Abdallah and Victor R. Lesser. 2007. Multiagent reinforcement learning and self-organization in a network of agents. In AAMAS.
[2] Georgios Chalkiadakis, Edith Elkind, Evangelos Markakis, Maria Polukarov, and Nicholas R. Jennings. 2010. Cooperative Games with Overlapping Coalitions. J. Artif. Intell. Res. 39 (2010), 179–216.
[3] Mehdi Dastani, Virginia Dignum, and Frank Dignum. 2003. Role-assignment in open agent societies. In AAMAS.
[4] Christian Schroeder de Witt, Jakob Foerster, Gregory Farquhar, Philipp Torr, Wendelin Boehmer, and Shimon Whiteson. 2019. Multi-Agent Common Knowledge Reinforcement Learning. In NeurIPS.
[5] Giovanna Di Marco Serruggia, Marie-Pierre Gleizes, and Anthony Karageorgis. 2005. Self-organization in multi-agent systems. Knowledge Engineering Review 20, 2 (2005), 165–189.
[6] Daniela Scherer Dos Santos and Ana LC Bazzan. 2012. Distributed clustering for group formation and task allocation in multiagent systems: A swarm intelligence approach. Applied Soft Computing 12, 8 (2012), 2123–2131.
[7] Jiechuan Jiang, Chen Dun, Tiejun Huang, and Zongqing Lu. 2019. Graph Convolutional Reinforcement Learning. In NeurIPS.
[8] Eric A Hansen, Daniel S Bernstein, and Shlomo Zilberstein. 2004. Dynamic programming for partially observable stochastic games. In AAAI.
[9] Robin Glinton, Kate F. Sycara, and Paul Serrn. 2008. Agent Organized Networks Redux. In AAAI.
[10] Tuomas Haarnoja, Vitchyr Pong, Aurick Zhou, Muratza Dalal, Pieter Abbeel, and Sergey Levine. 2018. Composable deep reinforcement learning for robotic manipulation. In ICRA.
[11] Tuomas Haarnoja, Haoan Tang, Pieter Abbeel, and Sergey Levine. 2017. Reinforcement learning with deep energy-based policies. In ICLR.
[12] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 2018. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. In ICLR.
[13] Frans A Oliehoek, Matthijs TJ Spaan, and Nikos Vlassis. 2008. Optimal and approximate Q-value functions for decentralized POMDPs. Journal of Artificial Intelligence Research 32 (2008), 289–353.
[14] Georgios Papoudakis, Filippos Christianos, Lukas Schafer, and Stefano V Albrecht. 2016. A survey of self-organization and self-organization in a network of agents. In AAMAS.
[15] Kevin R McKeer, Ian Gemp, Brian McWilliams, Edgar A Duenez-Guzman, Edward Hughes, and Joel Z Leibo. 2020. Social Diversity and Social Preferences in Mixed-Motive Reinforcement Learning. In AAMAS.
[16] Kevin R McKeer, Ian Gemp, Brian McWilliams, Edgar A Duenez-Guzman, Edward Hughes, and Joel Z Leibo. 2020. Social Diversity and Social Preferences in Mixed-Motive Reinforcement Learning. In AAMAS.
[17] Alexander Pevsalski and Adam Lerrer. 2018. Prosocial Learning Agents Solve Generalized Stag Hunt Better than Selfish Ones. In AAMAS.
[18] Chongjie Zhang, Victor R Lesser, and Sherief Abdallah. 2010. Decentralized coordination in roborescue. Comput. J. 53, 9 (2010), 1417–1461.
[19] Peter Sunehag, G. Lever, A. Gruslys, W. Czarnecki, V. Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z. Leibo, K. Tuyls, and T. Graepel. 2018. QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. In NeurIPS.
[20] Alexander Pevsalski and Adam Lerrer. 2018. Prosocial Learning Agents Solve Generalized Stag Hunt Better than Selfish Ones. In AAMAS.
[21] Yuhang Song, Jianyi Wang, and Zhen Xiao. 2018. MAgent: A many-agent reinforcement learning platform for artificial collective intelligence. In AAAI.
SUPPLEMENTARY MATERIAL

A. Environments

**Pacmen.** This scenario is a fully cooperative task, where $N$ agents initialized at the maze center and $M$ dots scatter randomly at four corner rooms. Agents get the reward by eating dots. Each agent has a local observation that contains a circle view with a radius 7 centered at the agent itself. The moving or attacking range of the agent is the only 1 neighbor grid. The reward is $-0.01$ for moving, $+0.5$ for attacking the dot, $-0.1$ for attacking a blank grid, and $+5$ for eat a dot. Since the dots are distributed in different corners, the agent needs to be grouped automatically and travel to different corners to eat more dots.

**Block.** This scenario is a fully cooperative task, where $N$ blockers and $L$ blockees who have superior speed than the blocker. There also are $M$ foods initialized on one side of the map. Blockers and blockees are only rewarded by eating foods. The moving or attacking range of the blockers is the 5 neighbor grids and the blockees have larger local observed range. The blockers have very large hit points and therefore are considered indestructible. Blockees could be killed by blockers. The reward is 0 for moving, $-0.2$ for attacking, $-1$ for being killed, and $+5$ for eat one food. Since the blockee runs faster than the blocker, the blocker needs to learn to use diverse strategies to block blockees and eat food at the same time.

**Pursuit.** This scenario is a fully cooperative task, where $N$ predators and $L$ preys who have superior speed than the predators. The moving or attacking range of the agent is the 4 neighbor grids and the predator has a larger local observed range. The reward is 0 for moving, $-0.2$ for attacking the prey, 1 for killing, $-1$ for being killed, and $-0.2$ for attacking a blank grid. Since the prey runs faster than the predator, the predator needs to learn to round up through the division of labor and cooperation.

**Battle.** This scenario is a fully cooperative task, where $N$ agents learn to fight against $L$ enemies who have superior abilities than the agents. The moving or attacking range of the agent is the 4 neighbor grids, however, the enemy can move to one of 12 nearest grids or attack one of 8 neighbor grids. Each agent/enemy has 10 hit points. The reward is $-0.005$ for moving, $+5$ for attacking the enemy, $-0.1$ for being killed, and $-0.1$ for attacking a blank grid. As the hit point of the enemy is 10, agents have to continuously cooperate to kill the enemy. Therefore, the task is much more challenging than Pursuit in terms of learning to cooperate. The schematic diagrams of all environments are shown in Figure 7.

![Figure 7: The simulated environments.](image)

B. Hyperparameters

The detailed hyperparameter settings of all algorithms are shown in Table 2, Table 3, Table 4 and Table 5.

C. Notations

The notations of three module in rochico are shown in Table 6, Table 7 and Table 8.
| Hyperparameter                          | Value                      |
|----------------------------------------|----------------------------|
| Batch Size                             | 512                        |
| Experience Replay Buffer Size          | 1,024,000                  |
| Target Network Update Frequency        | 1 times per 1000 samples   |
| Train Frequency                        | 4 times per batch          |
| Train Episodes                         | 500                        |
| Max Episode Length                     | 250                        |
| \( \epsilon \) Piecewise Decay        | \([0, 200, 400]\) episodes, \([1, 0.2, 0.05]\)     |
| Layer Number of CNN                    | 2                          |
| Kernel Size of Each Layer              | \(3 \times 3\)            |
| Kernel Number of Each Layer            | 32                         |
| Hidden Size of MLP                     | \([256, 512]\)             |
| Activation Function                    | ReLU                       |
| Double Q-Learning                      | True                       |
| Dueling Q-Learning                     | True                       |
| Learning Rate                          | 0.0001                     |
| \( \gamma \)                           | 0.99                       |
| Hidden Sizes of MLP in Team Intention Generator | \([32, 32, 32]\)             |
| Kernel Size of CNN in Team Intention Generator | \(3 \times 3\)            |
| Kernel Number of CNN in Team Intention Generator | 32                         |
| Hypernet Kernel Size of CNN            | \(3 \times 3\)            |
| Hypernet Layer Number of MLP           | 2                          |
| Hypernet Hidden Size of MLP            | 64                         |
| Layer Number of GCN                    | 3                          |
| Hidden Size of GCN                     | 32                         |
| Hidden Sizes of VAE Encoder            | \([32, 32]\)               |
| Hidden Sizes of VAE Decoder            | \([32, 32]\)               |
| Dimension of Hierarchical Intentions   | 32                         |

Table 2: The hyperparameters setting of **ROCHICO** algorithm.

| Hyperparameter                          | Value                      |
|----------------------------------------|----------------------------|
| Batch Size                             | 512                        |
| Experience Replay Buffer Size          | 4,096,000                  |
| Target Network Update Frequency        | 1 times per 1000 samples   |
| Train Frequency                        | 4 times per batch          |
| Train Episodes                         | 500                        |
| Max Episode Length                     | 250                        |
| \( \epsilon \) Piecewise Decay        | \([0, 200, 400]\) episodes, \([1, 0.2, 0.05]\)     |
| Layer Number of CNN                    | 2                          |
| Kernel Size of Each Layer              | \(3 \times 3\)            |
| Kernel Number of Each Layer            | 32                         |
| Hidden Size of MLP                     | \([256, 512]\)             |
| Activation Function                    | ReLU                       |
| Double Q-Learning                      | True                       |
| Dueling Q-Learning                     | True                       |
| Learning Rate                          | 0.0001                     |
| \( \gamma \)                           | 0.99                       |

Table 3: The hyperparameters setting of **IDQN** algorithm.
| Hyperparameter                     | Value                                               |
|-----------------------------------|-----------------------------------------------------|
| Batch Size                        | 512                                                 |
| Experience Replay Buffer Size     | 4,096,000                                           |
| Target Network Update Frequency   | 1 times per 1000 samples                           |
| Train Frequency                   | 4 times per batch                                   |
| Train Episodes                    | 500                                                 |
| Max Episode Length                | 250                                                 |
| epsilon Piecewise Decay           | [0, 200, 400] episodes, [1, 0.2, 0.05]              |
| Layer Number of CNN               | 2                                                   |
| Kernel Size of Each Layer         | $3 \times 3$                                       |
| Kernel Number of Each Layer       | 32                                                  |
| Hidden Size of MLP                | [256, 512]                                          |
| Activation Function               | ReLU                                                |
| Double Q-Learning                 | True                                                |
| Dueling Q-Learning                | True                                                |
| Learning Rate                     | 0.0001                                              |
| $\gamma$                          | 0.99                                                |
| Hypernet Kernel Size of CNN       | $3 \times 3$                                       |
| Hypernet Layer Number of MLP      | 2                                                   |
| Hypernet Hidden Size of MLP       | 64                                                  |

Table 4: The hyperparameters setting of QMIX algorithm.

| Hyperparameter                     | Value                                               |
|-----------------------------------|-----------------------------------------------------|
| Batch Size                        | 512                                                 |
| Experience Replay Buffer Size     | 1,024,000                                           |
| Target Network Update Frequency   | 1 times per 1000 samples                           |
| Train Frequency                   | 4 times per batch                                   |
| Train Episodes                    | 500                                                 |
| Max Episode Length                | 250                                                 |
| epsilon Piecewise Decay           | [0, 200, 400] episodes, [1, 0.2, 0.05]              |
| Layer Number of CNN               | 2                                                   |
| Kernel Size of Each Layer         | $3 \times 3$                                       |
| Kernel Number of Each Layer       | 32                                                  |
| Hidden Size of MLP                | [256, 512]                                          |
| Activation Function               | ReLU                                                |
| Double Q-Learning                 | True                                                |
| Dueling Q-Learning                | True                                                |
| Learning Rate                     | 0.0001                                              |
| $\gamma$                          | 0.99                                                |
| Hypernet Kernel Size of CNN       | $3 \times 3$                                       |
| Hypernet Layer Number of MLP      | 2                                                   |
| Hypernet Hidden Size of MLP       | 64                                                  |
| Layer Number of GCN               | 1                                                   |
| Hidden Size of GCN                | 32                                                  |
| Hidden Sizes of VAE Encoder       | [32, 32]                                            |
| Hidden Sizes of VAE Decoder       | [32, 32]                                            |
| Dimension of Latent Variable      | 32                                                  |

Table 5: The hyperparameters setting of NCC – Q algorithm.
| Symbols | Descriptions |
|---------|--------------|
| $\mathcal{G}$ | The directed graph constructed by all agents in the environment. |
| $\mathcal{V}$ | The node set of graph $\mathcal{G}$ and each node represents a agent. |
| $\mathcal{E}$ | The edge set of graph $\mathcal{G}$ and edges are determined by agents’ policies. |
| $v$ | A node in node set $\mathcal{V}$. |
| $n$ | The mode of node set $\mathcal{V}$. |
| $\mathcal{M}_\text{org}$ | The POSG used to model organization control problem. |
| $X_u$ | The agent space of $\mathcal{M}_\text{org}$. |
| $S_u$ | The state space of $\mathcal{M}_\text{org}$. |
| $\mathcal{A}_i$ | The action space of agent $i$ belongs to agent space $X_u$. |
| $O_i^u$ | The observation space of agent $i$ belongs to agent space $X_u$. |
| $P_u$ | The transition model of $\mathcal{M}_\text{org}$. |
| $\mathcal{E}_u$ | The emission probability model of $\mathcal{M}_\text{org}$. |
| $R_i^u$ | The reward function of agent $i$ belongs to agent space $X_u$. |
| $a_{i, u}$, $a^\prime_{i, u}$ | Two consecutive actions belongs to action space $\mathcal{A}^u_i$. |
| $o_{i, u}$, $o^\prime_{i, u}$ | The consecutive observations belongs to observation space $O^u_i$. |
| $m$ | The upper bound of nearest neighbors of any agent in agent space $X_u$. |
| $N_m(i)$ | The $m$-nearest neighbors set of agent $i$ belongs to $X_u$. |
| $d(i)$ | The index of agent $i$ of action belongs to action space $\mathcal{A}^u_i$. |
| $\mathcal{G}_u$ | The corresponding undirected graph of $\mathcal{G}$. |
| $\mathcal{V}_u$ | The node set of graph $\mathcal{G}_u$ and each node represents a agent. |
| $e(i, j)$ | The edge between node $i$ and node $j$ in $\mathcal{G}_u$. |
| $r_{i, u}$ | The external reward of agent $i$. |
| $r_i^{\prime, u}$ | The structural consistency intrinsic reward of agent $i$. |
| $r_i^{\prime, u}$ | The total reward of agent $i$. |
| $\alpha_u$ | The strength of constraint for structural consistency. |
| GED(·) | The graph edit distance |
| $\mathcal{G}_u(N_m(i) \lor i)$ | The sub-graph only contains node $i$ and its $m$-nearest neighbors before take action $a_{i, u}$. |
| $\mathcal{G}_u(N_m(i) \lor i)$ | The sub-graph only contains node $i$ and its $m$-nearest neighbors after take action $a^\prime_{i, u}$. |
| $J_u$ | The optimization goal of organization control module. |
| $\tau_u$ | The sample trajectory. |
| $L_u^i(·)$ | The TD(0) error of agent $i$. |
| $\theta_u, \bar{\theta}_u$ | The parameters of $Q$ function and target $Q$ function. |
| $D$ | The experience replay buffer. |
| $y$ | The TD backup. |
| $\gamma$ | The discount factor. |
| $L_u^\theta(\theta_u)$ | The overall objective function of organization control module. |

Table 6: Notations in the organization control module.
| Symbols | Descriptions |
|---------|--------------|
| $f_\mu(\cdot)$ | The state encoder parameterized by $\mu$. |
| $k, u, v$ | The team indicators. |
| $n_k$ | The agent number of team $k$. |
| $o^{k,t}_i$ | The observation of agent $i$ in team $k$ at timestep $t$. |
| $o^t$ | The joint observation of all agents in team $k$ at timestep $t$. |
| $e^{k,t}_i$ | The observation embedding of agent $i$ in team $k$ at timestep $t$. |
| $f_\nu(\cdot)$ | The DeepSet network parameterized by $\nu$. |
| $e^k_t$ | The team embedding of team $k$ at timestep $t$. |
| $f_\omega(\cdot)$ | The team intention encoder parameterized by $\omega$. |
| $s_t$ | The global state of timestep $t$. |
| $c^k_t$ | The team intention of team $k$ at timestep $t$. |
| $(x^{k,t}_i, y^{k,t}_i)$ | The spatial position of agent $i$ in team $k$ at timestep $t$. |
| $[\bar{x}^k_t, \bar{y}^k_t]$ | The average spatial position of team $k$ at timestep $t$. |
| $t^k$ | The timestep of team $k$. |
| $e^{k,t}$ | The spatiotemporal feature representation of team $k$. |
| $d(k, l)$ | The Euclidean distance in the spatiotemporal space between team $k$ and $l$. |
| $G_t(V_t, E_t)$ | The undirected graph constructed by team (similar as undirected graph in organization control module). |
| $e(k, l)$ | The edge between team $k$ and team $l$. |
| $y^k_t$ | The label of team $k$. |
| $L^{k,\ell}_k(\mu, \nu, \omega)$ | The contrasive learning objective function of hierarchical consensus learning module. |
| $m$ | The margin parameter in contrasive learning objective function. |
| $f_\xi(\cdot)$ | The team intention decoder parameterized by $\xi$. |
| $\hat{o}^{k,t+1}_i$ | The reconstructed observation by team intention decoder of agent $i$ in team $k$ at timestep $t$. |
| $\hat{o}^{k,t+1}$ | The joint reconstructed observation by team intention decoder of all agents in team $k$ at timestep $t$. |
| $L^{k,\ell}_k(\xi)$ | The prediction loss of team intention decoder. |
| $L^{q,\ell}_g(\mu, \nu, \omega, \xi)$ | The loss function of team intention generator. |
| $\lambda_{tg}$ | The temperature parameter of loss function of team intention generator. |
| $g_\phi(\cdot)$ | The individual encoder parameterized by $\phi$. |
| $h^{k,t}_i$ | The individual embedding of agent $i$ in team $k$ at timestep $t$. |
| $g_\psi(\cdot)$ | The graph convolutional network parameterized by $\psi$. |
| $h^k_t$ | The joint embedding of all agents in team $k$ at timestep $t$. |
| $\varphi^{k,t}_i$ | The individual cognition of agent $i$ in team $k$ at timestep $t$. |
| $g_\varphi(\cdot)$ | The variational encoder parameterized by $\varphi$. |
| $\zeta^{k,t}_i$ | The individual intention of agent $i$ in team $k$ at timestep $t$. |
| $L^{\ell}_k(\phi, \psi, \varphi)$ | The hierarchical consensus loss of hierarchical consensus module. |
| $L^{\ell}_g(\phi)$ | The loss function of variation autoencoder. |
| $p(\zeta^{k,t}_i)$ | The prior distribution of individual intention of agent $i$ in team $k$ at timestep $t$. |
| $L^{\ell}_g(\phi, \psi, \varphi)$ | The loss function of individual intention generator. |

Table 7: Notations in the hierarchical consensus module.
| Symbols          | Descriptions                                                                 |
|------------------|-----------------------------------------------------------------------------|
| $r_{t, t}$       | The intrinsic reward of team $k$ at timestep $t$.                           |
| $L_{t}^{i}(\theta_{t}^{i})$ | The TD(0) error of local $Q$ function of agent $i$ in team $k$ at timestep $t$. |
| $Q_{\theta_{t}^{i}}(\cdot), Q_{\tilde{\theta}_{t}^{i}}$ | The local $Q$ function and local target $Q$ function of agent $i$ in team $k$, parameterized by $\theta_{t}^{i}$ and $\tilde{\theta}_{t}^{i}$ respectively. |
| $y_{t}^{i}$      | The TD backup of agent $i$ in team $k$.                                    |
| $L_{t}^{k}(\theta_{t}^{k})$ | The TD(0) error of joint $Q$ function of team $k$ at timestep $t$.        |
| $Q_{\theta_{t}^{k}}(\cdot), Q_{\tilde{\theta}_{t}^{k}}$ | The joint $Q$ function and joint target $Q$ function of team $k$, parameterized by $\theta_{t}^{k}$ and $\tilde{\theta}_{t}^{k}$ respectively. |
| $y_{t}^{k}$      | The TD backup of team $k$.                                                 |
| $L_{t}^{Q}({\theta_{t}^{i}}, {\theta_{t}^{k}})$ | The loss function of decision module.                                     |
| $\lambda_{QMI}$  | The temperature parameter of loss function of decision module.            |

Table 8: Notations in the decision module.