Effective and Stable Role-Based Multi-Agent Collaboration by Structural Information Principles

Xianghua Zeng[1], **Hao Peng**[1], **Angsheng Li**[1, 2]

[1] State Key Laboratory of Software Development Environment, Beihang University, Beijing, China;
[2] Zhongguancun Laboratory, Beijing, China.
{zengxianghua, penghao, angsheng}@buaa.edu.cn, liangsheng@gmail.zgclab.edu.cn.

Abstract

Role-based learning is a promising approach to improving the performance of Multi-Agent Reinforcement Learning (MARL). Nevertheless, without manual assistance, current role-based methods cannot guarantee stably discovering a set of roles to effectively decompose a complex task, as they assume either a predefined role structure or practical experience for selecting hyperparameters. In this article, we propose a mathematical **S**tructural **I**nformation principles-based **R**ole **D**iscovery method, namely **SIRD**, and then present a SIRD optimizing **MARL** framework, namely **SR-MARL**, for multi-agent collaboration. The SIRD transforms role discovery into a hierarchical action space clustering. Specifically, the SIRD consists of structuralization, sparsification, and optimization modules, where an optimal encoding tree is generated to perform abstracting to discover roles. The SIRD is agnostic to specific MARL algorithms and flexibly integrated with various value function factorization approaches. Empirical evaluations on the StarCraft II micromanagement benchmark demonstrate that, compared with state-of-the-art MARL algorithms, the SR-MARL framework improves the average test win rate by 0.17%, 6.08%, and 3.24%, and reduces the deviation by 16.67%, 30.80%, and 66.30%, under easy, hard, and super hard scenarios.

Introduction

Cooperative multi-agent reinforcement learning (MARL) has broadly been applied to a variety of complex decisions, such as games (Nowé, Vrancx, and Hauwere 2012; Vinyals et al. 2019), sensor networks (Zhang and Lesser 2011), social network (Peng et al. 2022), emergent tool usage (Baker et al. 2020), etc. There are two significant challenges in cooperative MARL: scalability that the joint state-action space grows exponentially with the number of agents (Yang and Wang 2020), and partial observability, which necessitates decentralized policies based on local action-observation histories because of communication constraints (Nguyen, Nguyen, and Nahavandi 2020). Hence, the paradigm of Centralized Training with Decentralized Execution (CTDE) (Oliehoek, Spaan, and Vlassis 2008; Kraemer and Banerjee 2016; Mahajan et al. 2019) is designed to deal with the above challenges. However, the centralized training requires searching agent policies in the joint state-action space, making the CTDE challenging to learn efficiently, especially when the number of agents is large (Samvelyan et al. 2019). A valid solution is integrating roles to decompose tasks in multi-agent systems, where each role is associated with a particular subtask and a role policy in the restricted state-action space (DeLoach and García-Ojeda 2010; Bonjean et al. 2014; Sun, Liu, and Dong 2020).

The key is bringing forward a set of roles to decompose the cooperative task effectively. The typical methods to predefine task decomposition or roles (Lhaksmana, Murakami, and Ishida 2018; Sun, Liu, and Dong 2020) require prior knowledge (subtask-specific rewards or role responsibilities) that is unavailable in practice. It is impractical to automatically learn an appropriate set of roles from scratch, equivalent to learning in the joint state-action space with substantial explorations (Wilson, Fern, and Tadepalli 2010; Wang et al. 2020). Instead of learning roles from scratch, RODE (Wang et al. 2021b) proposed to use the clustering technology to achieve role discovery in the joint action space. However, the performance of the RODE depends heavily on practical experience due to its high sensitivity to parameters of adopted clustering algorithms, such as the cluster number of K-means, the maximum and minimum cluster number of X-means, and the neighborhood radius and density threshold of DBSCAN [1]. Because practical task decomposition is not always well-defined a priori or changing, current role-based methods (Sun, Liu, and Dong 2020; Wang et al. 2020, 2021b) cannot guarantee effective role discovery without manual assistance.

This paper proposes a mathematical **S**tructural **I**nformation principles-based **R**ole **D**iscovery method, namely **SIRD**, and then presents a SIRD optimizing **MARL** framework, namely **SR-MARL**, for multi-agent collaboration. The crucial insight is that, instead of flat clustering, the SIRD transforms role discovery into hierarchical action space clustering. We firstly construct a weighted, undirected, and complete action graph, whose vertices represent actions, edges represent action correlations, and edge weight quantifies the functional correlation between actions with respect to achieving team targets. Secondly,

[1] Although recently proposed methods (Karami and Johansson 2014; Zhang et al. 2022), achieving automatical parameters search, introduces significant computational overhead, which is not conducive to real-time decision-making.

we leverage the one-dimensional structural entropy minimization principle (Li, Yin, and Pan 2016) to sparsify the action graph, to generate an initial encoding tree. Thirdly, we minimize the K-dimensional structural entropy (Li et al. 2018) of the sparse graph to get the optimal encoding tree, thereby achieving the hierarchical action space clustering. Furthermore, we take the hierarchical clustering on the optimal encoding tree as hierarchical abstracting of actions for role discovery. The SIRD is independent of manual assistance and flexibly integrated with various value function factorization approaches. Extensive experiments are conducted on the StarCraft II micromanagement tasks, including five easy maps, four hard maps, and four super hard maps. Comparative results and analysis demonstrate the performance advantages of our proposed role discovery method. All source code and data are available at Github[2].

The main contributions of this paper are as follows: 1) To the best of our knowledge, it is the first time to incorporate the mathematical structural information principles into MARL to improve effectiveness and stability under cooperative scenarios. 2) An innovative method, which transforms role discovery into a hierarchical abstracting on the encoding tree guided by one-dimensional and K-dimensional structural entropy, is proposed to optimize MARL for collaborative decisions. 3) Compared with the existing action space clustering, the hierarchical clustering achieved by the encoding tree guarantees more effective role discovery without manual assistance. 4) The extraordinary performance on challenging tasks shows that the SR-MARL achieves up to 6.08% average test win rate improvement and up to 66.30% deviation reduction than state-of-the-art baselines.

Preliminaries

Dec-POMDP

In this work, we consider a fully cooperative multi-agent task M, which can be modeled as a Dec-POMDP (Oliehoek and Amato 2016). The Dec-POMDP is defined as a tuple $\langle N, S, A, P, \Omega, O, R, \gamma \rangle$, where $N \equiv \{n_1, n_2, \ldots, n_{|N|}\}$ is the finite set of agents, S is the finite set of global states, A is the action space, P is the global state transition function, Ω is the finite set of partial observations, O is the observation probability function, R is the joint reward function, and $\gamma \in [0, 1)$ is the discount factor. At each timestep, each agent $n_i \in N$ chooses an action $a_i \in A$ on a global state $s \in S$, and all chosen actions form a joint action $\boldsymbol{a} \equiv [a_i]_{i=1}^{|N|}$. The joint action $\boldsymbol{a}$ results in a joint reward $r = R(s, \boldsymbol{a})$ and a transition to the next global state $s' \sim P(\cdot \mid s, \boldsymbol{a})$. We consider the partially observable setting, where each agent n_i receives an individual partial observation $o_i \in \Omega$ according to the function $O(o_i \mid s, a_i)$. Each agent n_i has an action-observation history $\tau_i \in \mathcal{T}$ and constructs individual policy $\pi_i(a_i \mid \tau_i)$ to maximize team performance jointly.

Role-based Learning

The idea of role-based learning is to decompose a multi-agent cooperative task into a set of subtasks, where the role set specifies task decomposition and subtask policies (Wilson, Fern, and Tadepalli 2010; Wang et al. 2021b). Given a fully cooperative multi-agent task M, let Ψ be the role set. Each role $\rho_j \in \Psi$ is defined as a tuple $\langle t_j, \pi_{\rho_j} \rangle$, where t_j is a subtask defined as $\langle N_j, S, A_j, P, \Omega, O, R, \gamma \rangle$, $N_j \subset N, \cup_j N_j = N$, and $N_j \cap N_l = \varnothing, j \neq l$. A_j is the restricted action space of role ρ_j, $A_j \subset A$, $\cup_j A_j = A$, and $|A_j \cap A_l| \geq 0, j \neq l$. $\pi_{\rho_j} : \mathcal{T} \times A_j \mapsto [0, 1]$ is the role policy for the subtask t_j.

Structural Information Principles

In the structural information principles (Li and Pan 2016), the structural entropy measures the uncertainty of a graph under a strategy of hierarchical partitioning. And the dynamics measurement of the encoding tree is recently analyzed (Yang, Peng, and Li 2022). By minimizing the K-dimensional structural entropy, the optimal hierarchical structure of the graph is generated, namely the optimal partitioning tree, which we call an "encoding tree" in our article. Given a weighted undirected graph $G = (V, E, W)$, V is the vertex[3] set, E is the edge set, and $W : E \mapsto \mathbb{R}^+$ is the weight function. Let $n = |V|$ be the number of vertices and $m = |E|$ be the number of edges. For each vertex $v \in V$, its degree d_v is defined as the sum of the weights of its connected edges. Then, we formally give the definitions of the encoding tree, the one-dimensional, and K-dimensional structural entropy of the weighted undirected graph G.

Encoding Tree. The encoding tree of G is defined as a rooted tree T with the following properties: 1) For each node $\alpha \in T$, there is a vertex subset in G corresponding with α, $T_\alpha \subseteq V$. 2) For the root node λ, we set $T_\lambda = V$. 3) For each node $\alpha \in T$, its children nodes are marked as $\alpha^\wedge\langle i \rangle$ ordered from left to right as i increases, and $\alpha^\wedge\langle i \rangle^- = \alpha$. 4) For each node $\alpha \in T$, we suppose that L is the number of its children nodes; then all vertex subsets $T_{\alpha^\wedge\langle i \rangle}$ are disjointed, and $T_\alpha = \bigcup_{i=1}^{L} T_{\alpha^\wedge\langle i \rangle}$. 5) For each leaf node ν, T_ν is a singleton containing a single vertex in V.

One-dimensional Structural Entropy. The one-dimensional structural entropy of G is defined as follows:

$$H^1(G) = -\sum_{v \in V} \frac{d_v}{vol(G)} \cdot \log_2 \frac{d_v}{vol(G)}, \tag{1}$$

where $vol(G) = \sum_{v \in V} d_v$ is the volume of G.

K-dimensional Structural Entropy. Given an encoding tree T whose height is at most K, the K-dimensional structural entropy of G is defined as follows:

$$H^K(G) = \min_T \left\{ \sum_{\alpha \in T, \alpha \neq \lambda} H^T(G; \alpha) \right\}, \tag{2}$$

$$H^T(G; \alpha) = -\frac{g_\alpha}{vol(G)} \log_2 \frac{\mathcal{V}_\alpha}{\mathcal{V}_{\alpha^-}}, \tag{3}$$

where g_α is the sum of weights of all edges connecting vertices in T_α with vertices outside T_α. $\mathcal{V}_\alpha$ is the volume of T_α,

[2] https://github.com/RingBDStack/SR-MARL

[3] A vertex is defined in the graph and a node in the tree.

the sum of degrees of all vertices in T_α, and T ranges over all encoding trees of height at most K. The dimension K is also the maximal height of the encoding tree T.

The SIRD Optimizing MARL Framework

In this section, we present the overall framework of the SR-MARL and describe the detailed design of the structural information principles-based role discovery method SIRD.

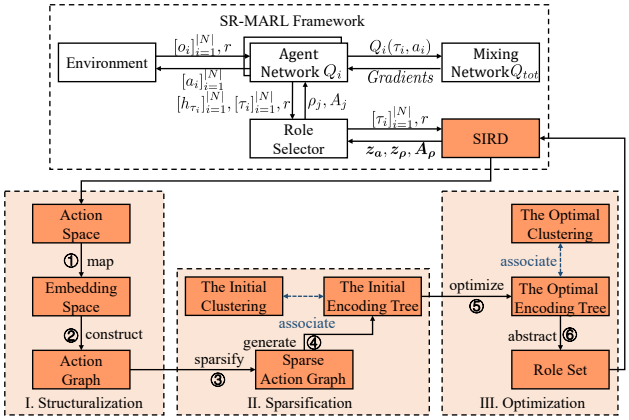


Figure 1: The overall framework of the SR-MARL.

Overall Framework of SR-MARL

The SR-MARL consists of five modules: Environment, Agent Network Q_i, Mixing Network Q_{tot}, Role Selector, and the role discovery module SIRD, as shown in Fig.1.

In the overall framework, each agent n_i makes decisions based on individual network Q_i which takes the partial observation o_i and joint reward r as inputs and is updated by the QPLEX-style mixing network Q_{tot} (Wang et al. 2021a). The mixing network O_{tot} has access to global information for centralized training. Apart from making decisions, the individual network Q_i encodes the local action-observation history τ_i into a d-dimensional hidden vector $h_{\tau_i} \in \mathbb{R}^d$, which is then fed into the role selector. The role discovery module SIRD described in the next subsection takes the action-observation histories of all agents $[\tau_i]_{i=1}^{|N|}$ and joint reward r as inputs. And the SIRD outputs action representations $\boldsymbol{z_a}$, role representations $\boldsymbol{z_\rho}$, and restricted action spaces $\boldsymbol{A_\rho}$ to the role selector for learning role policies. Inspired by the RODE (Wang et al. 2021b), the role selector assigns role $\rho_j \in \Psi$ and its associated action subspace $A_j \in \boldsymbol{A_\rho}$ to agent n_i, based on the dot product between the role representations $\boldsymbol{z_\rho}$ and the hidden vector h_{τ_i}. For coordinating the role assignment of all agents, we additionally apply a duplex dueling network of QPLEX (Wang et al. 2021a) to mix the dot products.

Role Discovery Module SIRD

As shown in Fig.1, the SIRD comprises Structuralization, Sparsification, and Optimization. In the structuralization, we map the action space to a fixed-dimensional embedding space and construct an action graph. In the sparsification, we sparsify the action graph and generate an initial encoding tree of the sparse graph. In the optimization, we optimize the encoding tree to realize the optimal hierarchical clustering, equivalent to a hierarchical abstracting of actions, and achieve role discovery on the optimal encoding tree.

Structuralization. Unlike the existing role-based methods (Sun, Liu, and Dong 2020; Wang et al. 2020, 2021b), the SIRD utilizes action correlations to construct an action graph, to improve the effectiveness of the role discovery. To this end, we learn action representations for achieving team targets, measure correlations between representations, and then construct the weighted, undirected, complete action graph based on the correlations.

Firstly, inspired by the RODE (Wang et al. 2021b), we similarly adopt the encoder-decoder structure (Cho et al. 2014) to learn the action representations, mapping the action space A to a d-dimensional embedding space. In the encoder, we encode each action $a \in A$ as an embedded representation $z_a \in \mathbb{R}^d$, as the step 1 in Fig.2[4]. For each agent n_i, the decoder decodes the representation z_{a_i} of its adopted action a_i to reconstruct its next partial observation o_i' and joint reward r. Given the action-observation histories of all agents $[\tau_i]_{i=1}^{|N|}$ and joint reward r, the encoder-decoder structure is trained end-to-end by minimizing the reconstruction loss. Secondly, for every two actions a_i and a_j with $a_i \neq a_j$, we measure their correlation $\mathcal{C}_{a_i,a_j} \in [-1, 1]$ through the Pearson Correlation Analysis:

$$\mathcal{C}_{a_i,a_j} = \frac{E\left((z_{a_i} - \mu_{z_{a_i}})(z_{a_j} - \mu_{z_{a_j}})\right)}{\sigma_{z_{a_i}} \sigma_{z_{a_j}}}, \tag{4}$$

where $\mu_{z_{a_i}}$ and $\sigma_{z_{a_i}}$ are the mean value and variance of the d-dimensional action representation z_{a_i}, respectively. Intuitively, the larger absolute value of $\mathcal{C}_{a_i,a_j}$ represents a more functional correlation between action a_i and action a_j on achieving team targets. Thirdly, we take each action in A as a vertex, connect any two vertices a_i and a_j, and assign $\mathcal{C}_{a_i,a_j}$ to edge (a_i, a_j) as weight w_{ij}, $w_{ij} = \mathcal{C}_{a_i,a_j}$, to construct the action graph G, as the step 2 in Fig.2.

Therefore, in the action graph G, vertices represent actions in the action space A, $V = A$, edges represent action correlations, and edge weight further quantifies the functional correlation between actions.

Sparsification. We implement sparsification of the action graph to reduce the computational cost and eliminate negative interference of trivial weights whose absolute values approach 0. Following the construction of cancer cell neighbor networks (Li, Yin, and Pan 2016), we sparsify the action graph G into a k-nearest neighbor (k-NN) graph G_k based on the one-dimensional structural entropy minimization principle. By minimizing the one-dimensional structural entropy of G_k, $H^1(G_k)$, we select the optimal number of neighbors k^*, as the step 3 in Fig.2. We summarize the sparsification step as Alg.1.

[4]For better understanding, we set $A = \{a_0, \ldots, a_{13}\}$ and differentiate actions by colors in the action embedding space as an example.

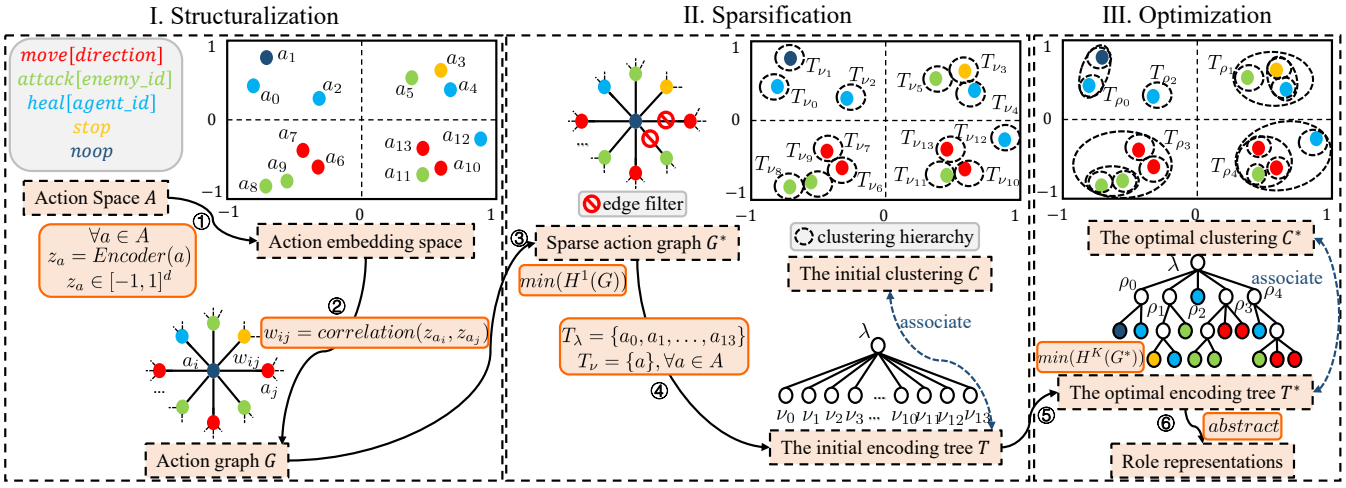


Figure 2: The structural information principles-based role discovery.

Firstly, based on the mean value μ_w of edge weights in G, we introduce a factor δ, $\delta = \frac{1}{2|A|} \cdot \mu_w$, $|A|$ is the number of actions, to correct all weights in G (Lines 2-4 in Alg.1):

$$w_{ij} = w_{ij} + \delta. \tag{5}$$

For significant weights whose absolute values approach 1, the factor δ is relatively tiny, and the corrected weights are approximately equal to the original values. For trivial weights, δ can notably correct them to eliminate their negative interference. Secondly, for each k, we construct the graph G_k (Lines 5 and 8 in Alg.1) by solely retaining the most significant k edge weights for each vertex and then calculate the one-dimensional structural entropy $H^1(G_k)$ (Lines 6 and 9 in Alg.1). Thirdly, we find all local minimal structural entropies (Lines 10-11 in Alg.1), namely *LMSE*, select the minimum k^* from the *LMSE*, and output G_{k^*} as the sparse action graph G^* (Lines 12-14 in Alg.1).

Therefore, we generate an initial encoding tree T of G^*: 1) For the action space A, we generate a root node λ and set its corresponding vertex subset $T_\lambda = A$; 2) For each action $a \in A$, we generate a leaf node ν with $T_\nu = \{a\}$ and set ν as a child node of λ, $\nu^- = \lambda$, as the step 4 in Fig.2. Intuitively, the height of the initial encoding tree T is 1. The associated hierarchical clustering is initialized as a single-level hierarchy where each action is divided into a separate category, as the initial clustering C in Fig.2.

Optimization. To realize the optimal hierarchical clustering C^* of the action space A, we optimize the encoding tree T of the sparse action graph G^* from 1 layer to K layers. Firstly, we introduce two operators *merge* and *combine* from the deDoc (Li et al. 2018) to minimize the K-dimensional structural entropy of the sparse graph G^* for optimizing T, as the step 5 in Fig.2. In the encoding tree T, two nodes are brothers if they have a common father node. The *merge* and *combine* operators are defined on brother nodes and marked as T_{mg} and T_{cb} in our work. Secondly, based on the K-dimensional structural entropy minimization principle (Li et al. 2018), we utilize T_{mg} and T_{cb} to design an iterative optimization algorithm, Alg.2. At each iteration, Alg.2 traverses all brother nodes β_1 and β_2 in T (Lines 4 and 9 in Alg.2) and greedily selects operator T_{mg} or T_{cb} that reduces the structural entropy the most (Lines 5 and 10 in Alg.2) to execute (Lines 7 and 12 in Alg.2) on the premise that the tree height does not exceed K. When no brother nodes satisfy $\Delta SE > 0$, we terminate the iteration and output T as the optimal encoding tree T^*. In the encoding tree T^*, the root node λ corresponds to the action space A, $T_\lambda = A$, each leaf node ν corresponds to a singleton containing a single action $a \in A$, and other nodes correspond to clusters of different hierarchies. For each leaf node ν with $T_\nu = \{a\}$, we set its node representation $z_\nu = z_a$. Thirdly, we realize the optimal hierarchical action space clustering C^* according to T^*.

Furthermore, we take the hierarchical clustering C^* on

Algorithm 1: The Sparsification Algorithm

Input: The action graph G
Output: The sparse action graph G^*

1 $\delta \leftarrow$ initialize correction factor
2 **for** $i = 1, \ldots, |A|$ **do**
3 **for** $j = i + 1, \ldots, |A|$ **do**
4 $w_{ij} \leftarrow$ correct w_{ij} via Eq.(5)
5 $G_1, G_2 \leftarrow$ construct 1-NN graph and 2-NN graph
6 $H^1(G_1), H^1(G_2) \leftarrow$ calculate structural entropy via Eq.(1)
7 **for** $k = 2, \ldots, |A| - 1$ **do**
8 $G_{k+1} \leftarrow$ construct $(k+1)$-NN graph
9 $H^1(G_{k+1}) \leftarrow$ calculate structural entropy of G_{k+1} via Eq.(1)
10 **if** $H^1(G_k) < H^1(G_{k-1})$ & $H^1(G_k) < H^1(G_{k+1})$ **then**
11 *LMSE* $\leftarrow H^1(G_k)$
12 $k^* \leftarrow$ find minimial k from *LMSE*
13 $G^* \leftarrow G_{k^*}$
14 return G^*

the optimal encoding tree T^* as hierarchical abstracting of actions to discover roles, as the step 6 in Fig.2.

Role Discovery on the Optimal Encoding Tree. The optimal encoding tree T^* partitions the action space A into a 3-level abstracting hierarchy from up to bottom, including roles, sub-roles, and actions, as shown in Fig.3(a). In the 3-level hierarchy, children nodes of the root node are defined as roles $\Psi \equiv [\rho_i]_i^{|\Psi|}$, $\rho_i = \lambda^\wedge\langle i\rangle$, leaf nodes $[\nu_i]_i^{|A|}$ are defined as actions, and other nodes on the paths connecting roles and actions are defined as sub-roles $[\rho_i']_i$, as shown in Fig.3(b). For example, actions ν_{10}, ν_{11}, and ν_{13} are abstracted as a sub-role ρ_2', and then both ρ_2' and action ν_{12} are abstracted as a role ρ_4. To calculate role representation z_{ρ_i}, we need to hierarchically aggregate node representations from bottom to up in the subtree rooted by ρ_i. For example, we aggregate action representations $z_{\nu_{10}}$, $z_{\nu_{11}}$, and $z_{\nu_{13}}$ as a sub-role representation $z_{\rho_2'}$, and then aggregate $z_{\rho_2'}$ and $z_{\nu_{12}}$ as a role representation z_{ρ_4}. To realize aggregating from bottom to up, we propose using the structural entropy distribution as each node's weight. For each non-leaf node $\alpha \in T^*$, the aggregate function is defined to calculate its node representation z_α as follows:

$$z_\alpha = \sum_{i=1}^{L} \frac{H^{T^*}\left(G; \alpha^\wedge\langle i\rangle\right)}{\sum_{j=1}^{L} H^{T^*}\left(G; \alpha^\wedge\langle j\rangle\right)} \cdot z_{\alpha^\wedge\langle i\rangle}, \tag{6}$$

where L is the number of children nodes of α. In addition, the restricted action space of role ρ_i is defined as its corresponding vertex subset T_{ρ_i}, $A_i = T_{\rho_i}$.

Finally, we output action representations $\boldsymbol{z_a} \equiv \{z_a \mid \forall a \in A\}$, role representations $\boldsymbol{z_\rho} \equiv \{z_{\rho_i} \mid \forall \rho_i \in \Psi\}$, and restricted action spaces $\boldsymbol{A_\rho} \equiv \{A_i \mid \forall \rho_i \in \Psi\}$ to achieve the role discovery.

Time Complexity of SIRD. The overall time complexity of the SIRD is $O\left(n^2 + n + n \cdot \log^2 n\right)$, where n is the number of actions, $n = |A|$. Here, the time complexity of the structuralization is $O(n^2)$, which analyzes Pearson Correlation for every two actions. In the sparsification, the SIRD costs $O(n)$ time complexity to find the minimum $k^* \in \{1, \ldots, n\}$ from all local minimal structural entropies. In our work, the optimization of the encoding tree costs $O(n \cdot \log^2 n)$ time complexity by adopting a similar data structure (Clauset, Newman, and Moore 2004). Compared with flat clustering in the RODE (Wang et al. 2021b), the SIRD achieves role discovery in the same quadric time complexity and guarantees independency of manual assistance, which leads to effective and stable performance advantages.

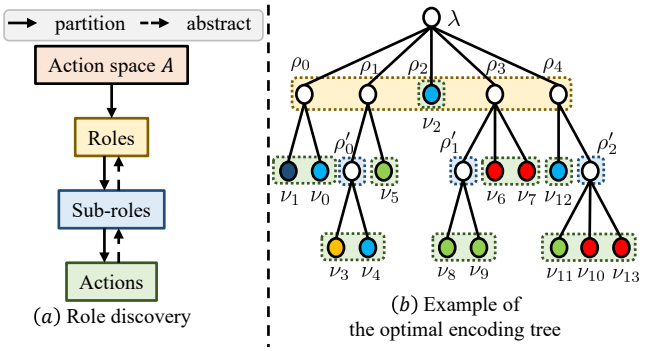
Experiments and Analysis

In this section, we conduct a large set of empirical and comparative experiments, aiming to verify the effectiveness and stability of the SR-MARL. Towards fair evaluation, all results are illustrated with the average and deviation of the performance testing with different random seeds, like in other works (Wang et al. 2021a,b).

Algorithm 2: The Iterative Optimization Algorithm

Input: The initial encoding tree T
Output: The optimal encoding tree T^*

$\Delta SE, \beta_1^*, \beta_2^* \leftarrow$ initialization
while *True* **do**
  $\Delta SE \leftarrow 0$
  for *each brother nodes* β_1 *and* β_2 *in* T **do**
    $\Delta SE, \beta_1^*, \beta_2^* \leftarrow$ maximize the reduction of the structural entropy caused by the *merge* operator via Eq.(2)
  if $\Delta SE > 0$ **then**
    $T \leftarrow T_{mg}(T; \beta_1^*, \beta_2^*)$
    Continue
  for *each brother nodes* β_1 *and* β_2 *in* T **do**
    $\Delta SE, \beta_1^*, \beta_2^* \leftarrow$ maximize the reduction of the structural entropy caused by the *combine* operator via Eq.(2)
  if $\Delta SE > 0$ **then**
    $T \leftarrow T_{cb}(T; \beta_1^*, \beta_2^*)$
  else
    Break
$T^* \leftarrow T$
return T^*



(a) Role discovery

(b) Example of the optimal encoding tree

Figure 3: The role discovery on the optimal encoding tree.

Experiment Setup

Datasets. We evaluate the SR-MARL on the StarCraft II micromanagement (SMAC) benchmark (Samvelyan et al. 2019), a mainstream benchmark of CTDE algorithms, of its rich environment and high control complexity. The SMAC benchmark includes five easy maps, four hard maps, and four super hard maps, where hard and super hard maps are typically hard-exploration tasks requiring agents to learn complex collaboration. As the SR-MARL is presented for improving multi-agent collaboration, we primarily focus on its performance on hard and super hard maps. In the micromanagement scenarios, each unit is controlled by an independent agent acting based on local observation, and a built-in AI controls all enemy units. At each timestep, each agent selects an action from the discrete action space, which consists of moving in four directions, stopping, taking no-op, and selecting an enemy/ally unit to attack/heal.

Categories	Easy	Hard	Super Hard
COMA	16.67 ± 22.73	$4.51 \pm \underline{9.58}$	-
IQL	52.50 ± 40.69	73.44 ± 24.85	10.55 ± 18.49
VDN	85.01 ± 17.22	71.49 ± 18.78	71.10 ± 27.23
QMIX	$\underline{98.44} \pm \underline{2.10}$	87.11 ± 18.58	70.31 ± 38.65
QTRAN	64.69 ± 36.79	58.20 ± 45.37	16.80 ± 20.61
QPLEX	96.88 ± 5.04	$\underline{89.85} \pm 11.35$	84.77 ± 10.76
RODE	93.47 ± 10.19	88.44 ± 20.96	$\underline{92.71} \pm \underline{9.20}$
SR-MARL	$\mathbf{98.61 \pm 1.75}$	$\mathbf{95.31 \pm 6.63}$	$\mathbf{95.71 \pm 3.10}$
Improvements(%)/Reductions(%)			
Average	$\uparrow 0.17$	$\uparrow 6.08$	$\uparrow 3.24$
Deviation	$\downarrow 16.67$	$\downarrow 30.80$	$\downarrow 66.30$

Table 1: Summary of the test win rates under different map categories: "average value $\pm$ standard deviation" and "improvements/reductions" (%). Bold: the best performance under each category, underline: the second performance.

Baselines and Variants. To make the empirical results more convincing, we compare the SR-MARL, whose maximal encoding tree height is set as 2, with state-of-the-art MARL algorithms, including independent Q-learning method (IQL (Tampuu et al. 2017)), value-based methods (VDN (Sunehag et al. 2018), QMIX (Rashid et al. 2018), QPLEX (Wang et al. 2021a), QTRAN (Son et al. 2019)), actor-critic method (COMA (Foerster et al. 2018)), and role-based method (RODE (Wang et al. 2021b)). The implementations of the SR-MARL and baselines in our experiments are based on the PyMARL ((Samvelyan et al. 2019)), and the hyperparameters of the baselines have been fine-tuned on the SMAC benchmark. All experiments adopt the default settings and are conducted on 3.00GHz Intel Core i9 CPU and NVIDIA RTX A6000 GPU.

Evaluations

We evaluate the SR-MARL and state-of-the-art MARL algorithms under different map categories (easy, hard, and super hard) and summarize averages and deviations of test win rates in Table 1. Table 1 shows that under different categories, our framework achieves at most a 6.08% improvement in average value and at most a 66.30% reduction in deviation. The improvement in average value and reduction in deviation correspond to the performance advantages on effectiveness and stability of the SR-MARL, respectively. In terms of effectiveness, explorations in restricted action spaces specified by the role set discovered from the optimal hierarchical abstracting guarantee strong cooperative ability among agents. In terms of stability, the SR-MARL leverages the structural information principles to guide automatic role discovery and therefore avoids the selection of sensitive hyperparameters. The performance advantages on effectiveness and stability of the SR-MARL become significant under hard-exploration scenarios (hard and super hard maps).

On the other hand, we benchmark the SR-MARL and classical baselines on all 13 maps to evaluate the overall performance across the SMAC suite. Fig.4 shows each algorithm's average test win rate across all maps and the number of maps where it achieves the highest average test win rate at different stages. Fig.4(left) illustrates that compared with

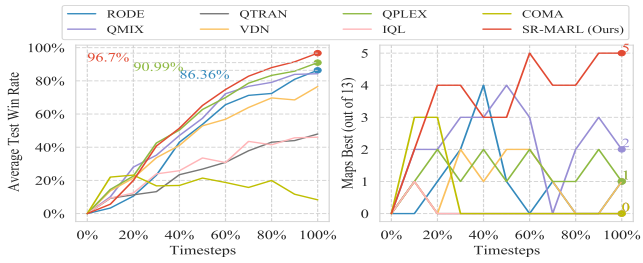


Figure 4: (left) The average test win rates across all 13 maps; (right) the number of maps (out of 13) where the algorithm's average test win rate is the highest.

baselines, the SR-MARL achieves outstanding overall performance and converges faster. In particular, the SR-MARL always maintains the highest average test win rate after 40% of the whole process until obtaining a 96.7% final average test win rate, which exceeds the second 90.99% (QPLEX) and the third 86.36% (RODE) by 5.71% and 10.34%. The advantages of the overall performance and learning efficiency can be attributed to the effective exploration of the action subspaces. Compared with the RODE, the SR-MARL accomplishes more effective role-based learning via transforming the role discovery into hierarchical action space clustering. From Fig.4(right), the SR-MARL finally performs best on almost half of all maps (5 of 13), much more than baselines.

In summary, the SR-MARL establishes a new state of the art on SMAC in terms of effectiveness and stability. Fig.5 show the learning curves of the SR-MARL and three representative baselines on each super hard map, respectively. The starting point of convergence and its variance are marked for each curve. The SR-MARL converges at 2056668 timestep and achieves a 94.6% average test win rate, with a variance of 0.0006, as shown in $27m\_vs\_30m$.

Integrative Abilities. The SIRD is agnostic to specific MARL algorithms and can be integrated with various value function decomposition approaches by replacing the mixing network Q_{tot} in Fig.1. We integrate the SIRD with the QMIX and QPLEX to get SR-QMIX and SR-QPLEX, respectively, and test their performance on map $2c\_vs\_64zg$. All integrated frameworks outperform the original approaches in effectiveness and efficiency, as shown in Fig.6. The comparative results indicate that the structural information principles-based role discovery method can significantly enhance multi-agent coordination.

Ablation Studies. We design ablation studies on map $2c\_vs\_64zg$ to evaluate the importance of structuralization and sparsification for performance advantages. To this end, we design two variants: ST-MARL and SP-MARL, degenerated frameworks of the SR-MARL without some functional modules. The ST-MARL variant without structuralization discovers roles from the joint action space via K-Means clustering instead of the SIRD module. The SP-MARL variant without sparsification directly optimizes the encoding tree of the complete action graph for role discovery. As

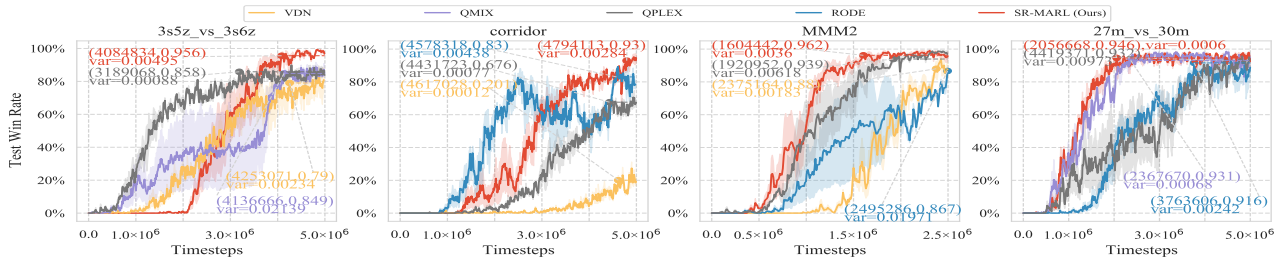


Figure 5: Average test win rates on four SMAC super hard maps.

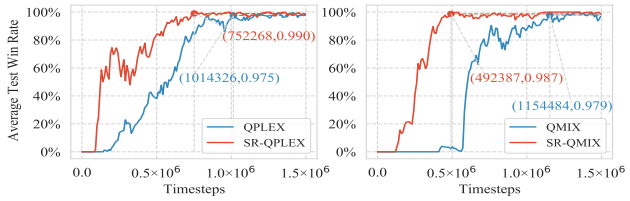


Figure 6: Average test win rates of the SR-MARL integrated with value decomposition methods QMIX and QPLEX.

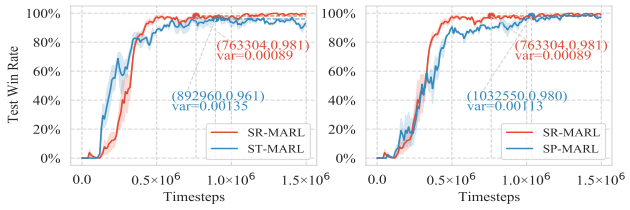


Figure 7: Average test win rates for ablation studies.

shown in Fig.7, the SR-MARL significantly outperforms the ST-MARL in the average and the deviation of the test win rates, which shows structuralization is the foundation of the SIRD and is crucial for the performance advantages on effectiveness and stability. The comparison between SR-MARL and SP-MARL indicates that sparsification remarkably boosts the learning process without affecting the performance advantages.

We discuss different maximal tree heights (2 and 3) and separately name associated frameworks SR-MARL-2 and SR-MARL-3. Fig.8 shows their learning curves on two different maps, $2c\_vs\_64zg$ and $MMM2$. In terms of effectiveness, stability, and efficiency, the SR-MARL-3 performs better on super hard map $MMM2$, while the SR-MARL-2 performs better on hard map $2c\_vs\_64zg$. The reason is maybe that super hard maps require a more hierarchical action space abstracting associated with a higher encoding tree to achieve complex multi-agent collaboration.

Related Work

Role-based Learning. In natural systems (Gordon 1996; Jeanson, Kukuk, and Fewell 2005; Butler 2012), the role is closely related to labor division and efficiency improvement. Inspired by the above benefits, multi-agent systems decom-

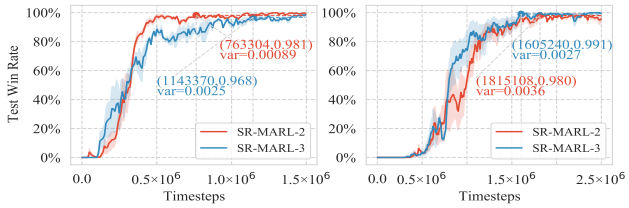


Figure 8: Average test win rates of the SR-MARL with the optimal encoding trees of different maximal heights (2 and 3) on two maps: (left) $2c\_vs\_64zg$ and (right) $MMM2$.

pose the task and specialize agents with the same role to subtasks for reducing the design complexity (Wooldridge, Jennings, and Kinny 2000; Bonjean et al. 2014). However, these methodologies use predefined task decomposition and roles that are unavailable in practice (Lhaksmana, Murakami, and Ishida 2018; Sun, Liu, and Dong 2020). Bayesian inference is introduced in MARL to learn roles (Wilson, Fern, and Tadepalli 2010), and ROMA encourages the emergence of roles by designing a specialization objective (Wang et al. 2020). Searching in the entire state-action space makes these methods inefficient. RODE achieves efficient role discovery by first decomposing joint action spaces (Wang et al. 2021b). Instead of flat clustering in RODE, the proposed SIRD utilizes hierarchical action space clustering to achieve more effective and stable role discovery.

Conclusion

This paper proposes a structural information principles-based role discovery method SIRD and presents a SIRD optimizing MARL framework SR-MARL for multi-agent collaboration. To the best of our knowledge, this is the first time that the mathematical structural information principles have been incorporated into MARL to improve performance under cooperative scenarios. Evaluations of challenging tasks in the SMAC benchmark demonstrate that the SR-MARL shows significant outperformance on effectiveness and stability over the state-of-the-art baselines and even can be flexibly integrated with various value function decomposition approaches to enhance coordination. For future work, we will conduct more analysis and explorations on the encoding tree under MARL scenarios.

Acknowledgments

The corresponding authors are Hao Peng and Angsheng Li. This paper was supported by the National Key R&D Program of China through grant 2021YFB1714800, NSFC through grant 61932002, S&T Program of Hebei through grant 20310101D, Natural Science Foundation of Beijing Municipality through grant 4222030, and the Fundamental Research Funds for the Central Universities.

References

Baker, B.; Kanitscheider, I.; Markov, T. M.; Wu, Y.; Powell, G.; McGrew, B.; and Mordatch, I. 2020. Emergent Tool Use From Multi-Agent Autocurricula. In *Proceedings of the International Conference on Learning Representations*, 1–28.

Bonjean, N.; Mefteh, W.; Gleizes, M.; Maurel, C.; and Migeon, F. 2014. *Handbook on Agent-oriented Design Processes*. Springer.

Butler, E. 2012. *The Condensed Wealth of Nations*. Centre for Independent Studies.

Cho, K.; Merriënboer, B. V.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; and Bengio, Y. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, 1724–1734.

Clauset, A.; Newman, M. E.; and Moore, C. 2004. Finding community structure in very large networks. *Physical Review E*, 70: 066111.

DeLoach, S. A.; and García-Ojeda, J. C. 2010. O-MaSE: a customisable approach to designing and building complex, adaptive multi-agent systems. *International Journal of Agent-Oriented Software Engineering*, 4(3): 244–280.

Foerster, J. N.; Farquhar, G.; Afouras, T.; Nardelli, N.; and Whiteson, S. 2018. Counterfactual Multi-Agent Policy Gradients. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2974–2982.

Gordon, D. M. 1996. The organization of work in social insect colonies. *Nature*, 380(6570): 121–124.

Jeanson, R.; Kukuk, P. F.; and Fewell, J. H. 2005. Emergence of division of labour in halictine bees: contributions of social interactions and behavioural variance. *Animal Behaviour*, 70(5): 1183–1193.

Karami, A.; and Johansson, R. 2014. Choosing DBSCAN parameters automatically using differential evolution. *International Journal of Computer Applications*, 91(7): 1–11.

Kraemer, L.; and Banerjee, B. 2016. Multi-agent reinforcement learning as a rehearsal for decentralized planning. *Neurocomputing*, 190: 82–94.

Lhaksmana, K. M.; Murakami, Y.; and Ishida, T. 2018. Role-based modeling for designing agent behavior in self-organizing multi-agent systems. *International Journal of Software Engineering and Knowledge Engineering*, 28(01): 79–96.

Li, A.; and Pan, Y. 2016. Structural Information and Dynamical Complexity of Networks. *IEEE Transactions on Information Theory*, 62: 3290–3339.

Li, A.; Yin, X.; and Pan, Y. 2016. Three-dimensional gene map of cancer cell types: Structural entropy minimisation principle for defining tumour subtypes. *Scientific Reports*, 6: 1–26.

Li, A.; Yin, X.; Xu, B.; Wang, D.; Han, J.; Wei, Y.; Deng, Y.; Xiong, Y.; and Zhang, Z. 2018. Decoding topologically associating domains with ultra-low resolution Hi-C data by graph structural entropy. *Nature Communications*, 9: 1–12.

Mahajan, A.; Rashid, T.; Samvelyan, M.; and Whiteson, S. 2019. Maven: Multi-agent variational exploration. *Advances in Neural Information Processing Systems*, 32: 7611–7622.

Nguyen, T. T.; Nguyen, N. D.; and Nahavandi, S. 2020. Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications. *IEEE Transactions on Cybernetics*, 50(9): 3826–3839.

Nowé, A.; Vrancx, P.; and Hauwere, Y.-M. D. 2012. Game Theory and Multi-agent Reinforcement Learning. In *Reinforcement Learning*, 441–470. Springer.

Oliehoek, F. A.; and Amato, C. 2016. *A Concise Introduction to Decentralized POMDPs*. Springer.

Oliehoek, F. A.; Spaan, M. T. J.; and Vlassis, N. 2008. Optimal and approximate Q-value functions for decentralized POMDPs. *Journal of Artificial Intelligence Research*, 32: 289–353.

Peng, H.; Zhang, R.; Li, S.; Cao, Y.; Pan, S.; and Yu, P. 2022. Reinforced, incremental and cross-lingual event detection from social messages. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Rashid, T.; Samvelyan, M.; de Witt, C. S.; Farquhar, G.; Foerster, J. N.; and Whiteson, S. 2018. QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. In *Proceedings of the International Conference on Machine Learning*, 4292–4301.

Samvelyan, M.; Rashid, T.; de Witt, C. S.; Farquhar, G.; Nardelli, N.; Rudner, T. G. J.; Hung, C.; Torr, P. H. S.; Foerster, J. N.; and Whiteson, S. 2019. The StarCraft Multi-Agent Challenge. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems*, 2186–2188.

Son, K.; Kim, D.; Kang, W. J.; Hostallero, D.; and Yi, Y. 2019. QTRAN: Learning to Factorize with Transformation for Cooperative Multi-Agent Reinforcement Learning. In *Proceedings of the International Conference on Machine Learning*, 5887–5896.

Sun, C.; Liu, W.; and Dong, L. 2020. Reinforcement learning with task decomposition for cooperative multiagent systems. *IEEE Transactions on Neural Networks and Learning Systems*, 32(5): 2054–2065.

Sunehag, P.; Lever, G.; Gruslys, A.; Czarnecki, W. M.; Zambaldi, V. F.; Jaderberg, M.; Lanctot, M.; Sonnerat, N.; Leibo, J. Z.; Tuyls, K.; and Graepel, T. 2018. Value-Decomposition Networks For Cooperative Multi-Agent Learning Based On Team Reward. In *Proceedings of the International Conference on Autonomous Agents and MultiAgent Systems*, 2085–2087.

Tampuu, A.; Matiisen, T.; Kodelja, D.; Kuzovkin, I.; Korjus, K.; Aru, J.; Aru, J.; and Vicente, R. 2017. Multiagent Cooperation and Competition with Deep Reinforcement Learning. *PloS One*, 12: e0172395.

Vinyals, O.; Babuschkin, I.; Czarnecki, W. M.; and etc. 2019. AlphaStar: Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, 575(7782): 350–354.

Wang, J.; Ren, Z.; Liu, T.; Yu, Y.; and Zhang, C. 2021a. Qplex: Duplex Dueling Multi-Agent Q-Learning. In *Proceedings of the International Conference on Learning Representations*, 1–27.

Wang, T.; Dong, H.; Lesser, V. R.; and Zhang, C. 2020. ROMA: Multi-Agent Reinforcement Learning with Emergent Roles. In *Proceedings of the International Conference on Machine Learning*, 9876–9886.

Wang, T.; Gupta, T.; Mahajan, A.; Peng, B.; Whiteson, S.; and Zhang, C. 2021b. Rode: Learning Roles to Decompose Multi-Agent Tasks. In *Proceedings of the International Conference on Learning Representations*, 1–24.

Wilson, A.; Fern, A.; and Tadepalli, P. 2010. Bayesian policy search for multi-agent role discovery. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 624–629.

Wooldridge, M. J.; Jennings, N. R.; and Kinny, D. 2000. The Gaia methodology for agent-oriented analysis and design. *Autonomous Agents and Multi-Agent Systems*, 3(3): 285–312.

Yang, R.; Peng, H.; and Li, A. 2022. Dynamic Measurement of Structural Entropy for Dynamic Graphs. *ArXiv*, abs/2207.12653.

Yang, Y.; and Wang, J. 2020. An overview of multi-agent reinforcement learning from game theoretical perspective. *arXiv preprint arXiv:2011.00583*.

Zhang, C.; and Lesser, V. R. 2011. Coordinated Multi-Agent Reinforcement Learning in Networked Distributed POMDPs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 764–770.

Zhang, R.; Peng, H.; Dou, Y.; Wu, J.; Sun, Q.; Li, Y.; Zhang, J.; and Yu, P. S. 2022. Automating DBSCAN via Deep Reinforcement Learning. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, 2620–2630.