Complex & Intelligent Systems (2023) 9:6211–6222
https://doi.org/10.1007/s40747-023-01071-x

ORIGINAL ARTICLE

Dynamic role discovery and assignment in multi-agent task decomposition

Yu Xia[1] · Junwu Zhu[1] · Liucun Zhu[1,2]

Received: 20 November 2022 / Accepted: 1 April 2023 / Published online: 26 April 2023

Abstract

Effective multi-agent teamwork can be facilitated by using personas to decompose goals into lower-level team subtasks through a shared understanding of multi-agent tasks. However, traditional methods for role discovery and assignment are not scalable and fail to adapt to dynamic changes in the environment. To solve this problem, we propose a new framework for learning dynamic role discovery and assignment. We first introduce an action encoder to construct a vector representation for each action based on its characteristics, and define and classify roles from a more comprehensive perspective based on both action differences and action contributions. To rationally assign roles to agents, we propose a representation-based role selection policy based on consideration of role differences and reward horizons, which enables agents to play roles dynamically by dynamically assigning agents with similar abilities to play the same role. Agents playing the same role share their learning of the role, and different roles correspond to different action spaces. We also introduce regularizers to increase the differences between roles and stabilize training by preventing agents from changing roles frequently. Role selection and role policy integrate action representations and role differences in a restricted action space, improving learning efficiency. We conducted experiments in the SMAC benchmark and showed that our method enables effective role discovery and assignment, outperforming the baseline on four of the six scenarios, with an average improvement in win rate of 20%, and is effective in hard and super hard maps. We also conduct ablation experiments to demonstrate the importance of each component in our approach.

Keywords Multi-agent · Task decomposition · Role discovery · Role assignment

Introduction

The current control strategies for multiple agents can be classified into three main paradigms: Centralized learning [1], Independent learning [2], and Centralized training distributed execution (CTDE) [3–5]. Centralized learning treats the entire system as a whole and uses single-agent reinforcement learning algorithms for training. Although this approach solves the problem of environmental non-smoothness, it requires global information, is not scalable, and cannot address issues of no communication, large scale, and large action space. Independent learning allows each agent to train its own strategy independently and achieves good performance in some cooperative tasks, but it ignores the multi-agent connections, which leads to unstable learning. Centralized training and distributed execution is a compromise approach that uses global information for training to improve learning efficiency, while allowing each agent to make independent decisions for execution. This approach has been shown to solve some of the multi-agent learning problems [6–8], but the optimal joint value function solution becomes complex as the number of agents increases.

In human societies, when dealing with complex tasks or problems, people tend to decompose them into different levels of subtasks, and individuals take on specific roles to deal with these subtasks [9]. This process is akin to each individual only exploring the constrained state action space associated with their assigned role. Thus, people will use roles associated with specific subtasks or strategies to deal with the

✉ Junwu Zhu
jwzhu@yzu.edu.cn

✉ Liucun Zhu
lczhu@bbgu.edu.cn

[1] School of Information Engineering, Yangzhou University, Yangzhou 225127, Jiangsu, China

[2] Advanced Science and Technology Research Institute, Beibu Gulf University, Qinzhou 535011, Guangxi, China

tasks. If we have a priori knowledge and domain expertise, we can create a set of predefined roles that decompose complex multi-agent tasks into simpler, low-level tasks [10]. However, in practice, predefined roles may not always meet the task requirements, and changes in task or team dynamics can affect the effectiveness of these roles. In some cases, team members need to take on new roles at the right time to adapt to evolving task dynamics.

Therefore, having a dynamic way of dividing and selecting roles is crucial. Existing approaches to dynamic roles require a lot of exploration in a large action space [11] and learning roles from scratch. Such an approach does not necessarily have advantages over role-free approaches. In a multi-agent environment, we can decompose tasks into small ones and gradually achieve the overall task goal. However, this approach requires a model that can effectively decompose multi-agent team tasks, posing a new challenge.

In this work, we propose a novel framework for learning dynamic role discovery and assignment (DRDA) in multi-agent team tasks. We use an action encoder to construct vector representations based on action attributes and combine action contributions to trip action representations. The results of action representations are used to cluster actions and eventually discover different roles. At each time step, the historical information of the agents is represented using a trajectory encoding network, the similarity with the role representation is computed to obtain the classification distribution of role selection, and the role selection strategy is learned. For learning strategies, agents playing the same role can share the learning process, and the network estimates the effects and contributions of the actions, weighting the Q-function values of the actions into the Q-functions of the roles. We also employ regularizers in the learning process of role discovery to better separate roles and increase the representational differences between them, and in the learning process of role selection strategy to avoid training instability caused by frequent agent changes.

In a cooperative task, we use action clustering to decompose the joint action space by learning action differences and action contributions. A role is defined by an action space, and each agent only needs to learn a subset of actions that correspond to their assigned function. For example, in a soccer game, players can be divided into goalkeepers, defenders, strikers, etc. Each player would be constrained in his choice of actions according to the role he plays, and the goalkeeper would only need to explore how to move when defending. We further differentiate and selection of roles by classifying them into long-term and short-term roles based on action differences and action contributions. We also investigate the losses incurred due to role differences through the reward horizon, which guides our role selection method. The role selection model coordinates role selection in the restricted action space of roles, and role policies are explored in this space and associated with action representations. We reduce the complexity of learning by spatially decomposing the multi-agent cooperation problem into several learning problems in the restricted space.

We conducted experiments to evaluate the performance of our proposed method in the SMAC [12] environment. The results demonstrate that our method consistently improves performance on all six maps tested. Our method performs well on both hard and super hard maps, achieving a win rate above the baseline, and also converges faster than the baseline method. Furthermore, we conducted ablation experiments to investigate the impact of three different components of our method on performance. These experiments revealed that the three components have varying degrees of impact on performance, providing insights into the effectiveness of our proposed approach.

Related work

CTDE

QMIX [5] is the most typical approach in the CTDE [13, 14] framework, which produces joint action valuations based on nonlinear combinations of individual agents. QTRAN [15] lifts the limitations of cumulativity and monotonicity to some extent, removing the structural constraints in the QMIX approach. However, QTRAN does not perform well in practice on subsequent SMAC tasks. Qatten [16] introduced a multi-headed attention mechanism to decompose the joint Q-values, compensating for the lack of theoretical support for the decomposition of joint Q-values in algorithms such as QMIX and VDN. QPLEX [17] introduces a duel structure in QMIX to generate action advantages. MADDPG [18] solves the MARL task for a continuous action space and stabilizes the training process by speculating the policies of other agents. For large-scale multi-agent tasks, Mean-Field [19] treats the agents around an agent as a whole with average properties, simplifying the interaction between agents. However, it lacks the details of individual agents, limiting its ability to cooperate on a large scale. Recently, QMIX-ME [20] was proposed to learn exploding strategies with maximum entropy while using the QMIX structure to solve the credit assignment problem. Additionally, optimizing the code level and enforcing monotonicity constraints for QMIX variants can improve the sample efficiency in SMAC and DEPP, according to recent studies [21].

Task decomposition

Task decomposition breaks down a complex task into a set of subtasks with restricted action space. We classify task decomposition into two types: general [22] or domain-specific

[23–25], depending on whether the application domain is restricted or not. Manual task decomposition was the basis of much previous work [26, 27], and now, much research investigates how to automate decomposition tasks. Many approaches require a high level of domain knowledge to understand the relationships between subtasks, leading to limitations in their extension [28–30]. The M+ algorithm [22] decomposes tasks into more primitive subtasks through a task list based on the agent's specific skill set, making the subtasks executed by the agent more adapted to their own characteristics. Task tree and auction-based approaches [24, 31] produce more system-specific decompositions by providing feedback to improve agent-competency-based task decomposition. However, it can be more time-consuming when subtasks need to be adapted. Recently, a framework for human–agent cooperation [32] was proposed that describes the key components of teamwork.

Roles

Roles have been widely used in the design of multi-agent systems to reduce overall complexity by decomposing tasks and assigning agents with the same roles to handle the same subtasks [10, 33–39]. However, such systems rely heavily on prior knowledge [40] for defining roles and their related subtasks [41]. While predefinition can be effective in specific scenarios [42], the generalizability of prior knowledge is greatly limited. Consequently, there is a growing interest in investigating the generalizability of personas across different contexts. Wilson et al. [43] proposed a model-free actor discovery algorithm using a Bayesian policy search approach, while Wang et al. [11] designed a specialized goal to encourage actor emergence in a flexible, generalized manner. However, these approaches require a lot of exploration in the complete action space, which can result in inefficient behavior that wastes resources and increases the complexity of the mechanism. Moreover, recent works have used roles to measure the characteristics of different agents [11, 44–48]. However, accurately and completely defining roles remains a challenge. For example, Wang et al. [46] defined roles as high-level options in a hierarchical reinforcement learning framework [49], while Christianos et al. [44] defined roles based on the similarity of environmental impacts of stochastic policies. While these definitions capture some aspects of the differences in agents' behavior, they cannot measure them completely.

Our work builds upon the distinctions in action differences to classify actions based on their contributions, establishing a more comprehensive perspective for defining roles and measuring the role distance between agents. We categorize roles into long-term roles and short-term roles using the knowledge of action differences and contributions to enhance their differentiation. Studying role-induced losses through reward horizon in role selection makes role rewards more reasonable. We decompose the complete action space based on action differences and contributions, and our unfixed role discovery allows for dynamic role discovery.

Role-based task decomposition model construction

In this paper, we propose a novel approach for defining roles based on action differences and action contributions. We decompose and form different role action spaces based on the varying degrees of influence that different actions have on environmental changes and other agents. Role distances are then computed for different roles played by the agents. Agents are assigned roles by considering the role selector of the reward horizon, and policies are selected on the role action space using role policies. Our approach enhances the exploration process of the agent while providing a priori knowledge and narrowing the agent's action space. This is particularly effective in environments that require exploration.

We model the multi-agent cooperative task as a Decentralized Partially Observable Markov Decision Process (Dec-POMDP) [50, 51], which comprises the tuple $G = \langle N, S, A, P, R, \Omega, O, n, \gamma \rangle$. Here, N is a finite set of n agents, $s \in S$ is the true state of the environment, and at each time step c, each agent $i \in N := (1, \ldots, n)$ is an independent observation $o_i \in \Omega$ determined by the observation function $O(s, i) : S \times N \mapsto \Omega$. Agent i selects an action $a_i \in A$, forming a joint action space $a \in A^n$. The transition function $P\left(s'|s, a\right) : S \times A^N \times S \mapsto [0, 1]$ defines the next state s'. All agents share the same reward function $r = R(s, a)$, and $\gamma \in [0, 1]$ is the discount factor. Each agent has a local action–observation history $\tau_i \in T \equiv (\Omega \times A)^*$.

Based on the CTDE scheme, we design a role-based multi-agent team task decomposition framework to decompose multi-agent teamwork tasks into subtasks under different levels, each of which is associated with a role. Each agent plays a role in one time step, and agents playing the same role at the same time step can share information and jointly learn policies to solve the subtasks associated with the role. Breaking down team goals into subgoals at a high level is important because it is a general way for team members to be able to quickly understand how the team solves problems. Breaking down the main goal into its sub-goals through personas does not specify how these sub-goals are achieved or by whom. As an example, in a StarCraft II confrontation scenario, our objective is to eliminate members of the opposing team, and eliminating members of the opposing team would be broken down into subtasks such as move, surround, and attack.

In this paper, we give a multi-agent cooperation task G, a set of roles Ψ, role ρ_j, subtask g_j. $G = \langle N, S, A, P, R, \Omega,$

$O, n, \gamma\rangle$. The role $\rho_j \in \Psi$ is defined by a tuple $\langle j, r_{i,j}, g_j, \pi_{\rho_j}\rangle$, j is the identity of the role; $r_{i,j}$ is the reward function of the role; $\pi_{\rho_j} : T \times A_j \rightarrow [0, 1]$ is the role policy for roles associated with sub-task $g_j = \langle N_j, S, A_j, P, R, \Omega_j, O, \gamma\rangle$, $N_j \subset N, \cup_j N_j = N$, and agents can only play one role in time step c: $N_j \cap N_k = \emptyset$, $j \neq k$. A_j is the restricted action space of the role ρ_j. The action spaces of different roles can overlap: $A_j \subset A, \cup_j A_j = A, |A_j \cap A_k| \geq 0, j \neq k$.

We hope that we can dynamically discover a set of roles that maximize the global expected payoff function $Q^{\psi}(s_t, a_t) = \mathbb{E}_{s_{t+1:\infty}, a_{t+1:\infty}}\left[\sum_{i=0}^{\infty} \gamma^i r_{t+i} | s_t, a_t, \psi\right]$ of the role ψ^*. The global expected gain is influenced by both role division and role selection. Our mechanism is a dynamic grouping mechanism, neither role division nor role selection is fixed that can learn role policies by exploring the role-restricted action space. When an agent chooses a role, the policy to solve the corresponding subtask is determined simultaneously.

Methods

In team-based task completion, roles of individuals are not fixed, and the number of individuals in each role may vary in different states. For instance, during a search and rescue operation, all personnel start off as searchers to locate the search and rescue target. Once the target is located, some personnel become rescuers to carry out rescue operations while others continue searching for the next target. This approach enables agents to collaborate effectively in multi-agent cooperative tasks. Figure 1 shows the learning framework of Dynamic Role Discovery and Assignment (DRDA), which first involves how a set of roles can be discovered to decompose a multi-agent cooperative task based on actions. We then discuss how agents can select roles based on their fitness. Finally, after the agents have selected roles, we propose policies for these roles that are associated with action representations.

Role discovery based on action representations

The effectiveness of the mechanism is largely determined by the role classification scheme, thus, it is crucial to carefully determine how to classify the roles. While different agents output different actions at the same time period depending on the state, and these different actions can be represented as different roles, defining roles solely based on action differences cannot fully reflect the different visual distance, action space, and reward function of the roles. Therefore, action differences cannot be the only criterion for defining roles. To address this limitation, we introduce action values to define role type differences based on action differences, which are called action contributions. We estimate the Q-value or state value of each action based on the reward function, and the Q-value is the contribution of a single action, which is considered as the contribution of the action to the team.

In many cases, although the actions taken by the agents are not the same, the effects of the actions on the surrounding environment are highly similar. For example, in a soccer game, two players pass the ball to each other during an attack [52], and although the actions taken by the players at each time step will not be identical, the effects of the actions of the two players passing the ball to each other are highly similar before changing the action of passing the ball to each other, and the two players actually play very similar roles. As a result, the number of roles found based on time steps may be large, while the differences between roles [46] may be small. To maintain the differentiation between the roles, we propose a regularizer.

We first characterize and cluster the actions based on their attributes, determine the action spaces to which the actions belong, calculate the action contributions, and group the roles with similar contributions in the same space. Actions with different contributions are identified as outliers. We classify action spaces with similar action effects as the same roles and those with different action effects as different roles. By doing so, we determine the action spaces to which actions belong and the final role classification.

We present the two-layer linear network model depicted in Fig. 1a to acquire the action encoder $f_e(\cdot; \theta_e)$ with parameter θ_e. The action encoder $f_e(\cdot; \theta_e)$ maps one-hot actions a_i from $\mathbb{R}^{|A|}$ to $\mathbb{R}^m$, thereby generating the vector representation $x_{\phi_i} = f_e(a_i; \theta_e) \in \mathbb{R}^m$ for each action a_i. The action representation $z_a = Q_\pi(s, a_k)\, x_{\phi_i}$ of action a is then outputted in this space.

To calculate the Loss function value and update the network, we predict the value $\widetilde{o}_i'$, $\widetilde{r}_t$, and the difference between the true values o_i, r_t:

$$\mathcal{L}_e(\theta_e, \xi_e) = \mathbb{E}\left(o, a, r, o'\right) \sim \mathcal{D}\left[\sum_i ||\widetilde{o_i}' - o_i'||_2^2 + \lambda_e \sum_i \left(\widetilde{r_t} - r\right)^2\right] \tag{1}$$

the hyperparameter λ_e plays a crucial role in determining the training focus of the agent, and adjusting its value can alter the significance of predicting the next observation and the predicted gain. Finally, the replay buffer $\mathcal{D}$ is employed in the training process.

At the same time, if the roles do not differ, then a large number of roles with highly similar action effects will appear, which will increase the complexity of the framework. To prevent the emergence of duplicate roles resulting from highly similar action effects, we introduce a regularizer that maxi-

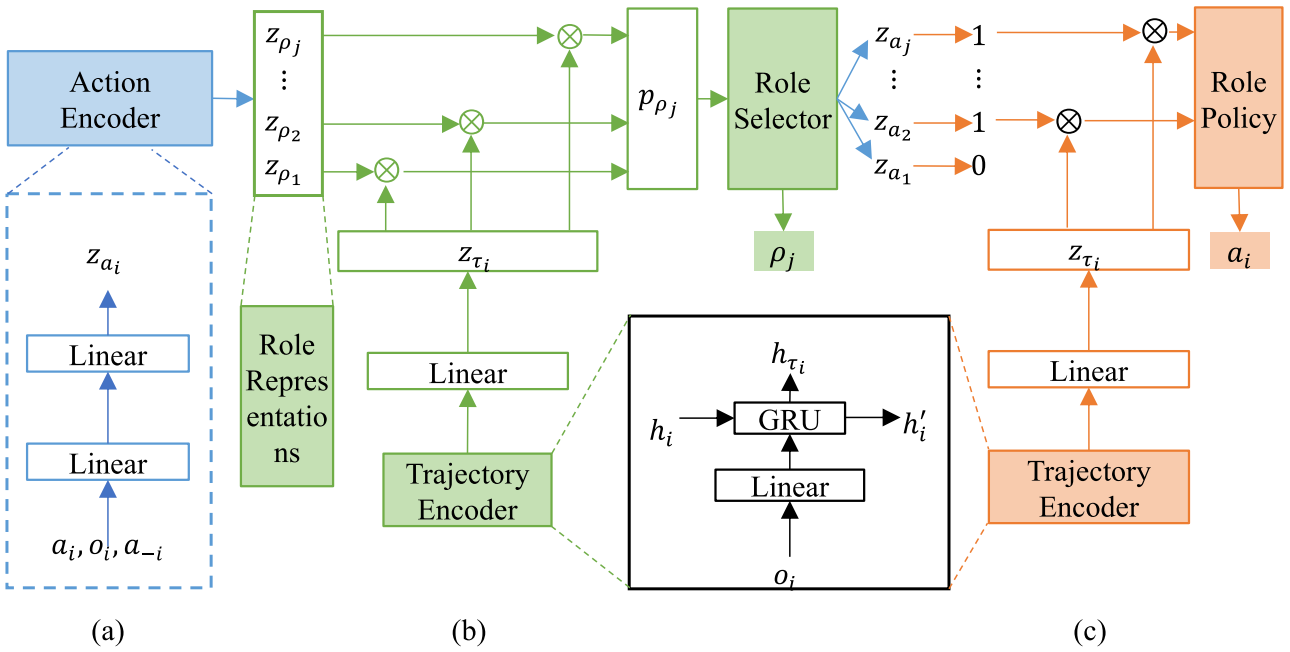


Fig. 1 The Overview framework of DRDA. **a** Action representation learning structure. **b** Role selection structure of the agent. **c** Policy learning structure of the role

mizes the L2 distance between role representations, thereby extracting the differences between roles and maintaining their differentiation:

$$\mathcal{L}_a(\theta_e) = \mathbb{E}_{\mathcal{D}}\left[-\sum_{j\neq k} ||z_{a_j} - z_{a_k}||^2\right] \tag{2}$$

We employ supervised learning on the samples (o, a, r, o') in the replay buffer, given K roles, i.e., K categories, with a complete action space for each role. After collecting samples and training the prediction model for a period, clustering is performed with k-means based on action representations, updating the cluster centers using the samples. Outliers are identified and marked as actions of outliers, and added to each role to update the clustering results, resulting in different categories representing different roles. The new action representations contain action effect information, and each clustering result represents a role, and the restricted action space of a role is the action in a category with the same action effect. Accordingly, we determine the action space to which the action belongs and the final role classification. Action effects can be measured by changes in action rewards and local observations. Through supervised learning, the hidden action representation z_a contains the same action effects (state transfer and reward values) as the original action a. During training, the action representation adapts to the dynamic changes of the environment by continuous learning, and after training is completed, the role and the corresponding action space are fixed.

Therefore, the overall optimization objective of role discovery is given by the equation:

$$\mathcal{L}_e(\theta_e, \xi_e) = \mathcal{L}_e(\theta_e, \xi_e) + \lambda_a \mathcal{L}_a(\theta_e) \tag{3}$$

where λ_a is the positive coefficient used to balance the regularizer.

Representation-based role selection

Taking into account the differences between various roles and the behavioral tendencies of agents, we aim to devise a role selection policy based on the similarity between the representations of roles and agents. Our roles are classified into two categories: short-term roles and long-term roles. We investigated the impact of the reward horizon, as reflected in the discount factor, on the loss of role reward. This is because certain roles in the team are oriented towards long-term rewards with no immediate gains, and it is not rational to use the same reward function for all roles. The reward horizon will vary for different types of roles: some favor immediate rewards (e.g., shooting immediate prey), while others prefer long-term rewards (e.g., guarding a camp). Our role selection policy is based on a traditional Q-network, where the local action observation history is fed as input, and the Q-value of the role is output. Every c steps, the agent chooses a role through the role selector, and each selection corresponds to determining the set of possible actions for the next c actions, where the Q value of the role is closely linked

to the role representation. The role will then act within the given time frame and receive different rewards based on the reward horizon and reward function.

By utilizing action representations, we construct the role representation z_{ρ_j} by averaging the representations of the action space contained in the role, which results in an average representation of the available actions of role ρ_j:

$$z_{\rho_j} = \frac{1}{|A_{\rho_j}|} \sum_{a_k \in A_{\rho_j}} z_{a_k} \tag{4}$$

z_{ρ_j} is the representation of role ρ_j, and A_{ρ_j} is the restricted action space of role ρ_j. We employ the structure shown in Fig. 1b to learn role selection. The observation o_i of agent i serves as the first input, which undergoes a shared trajectory encoder consisting of a shared linear layer and a GRU [53]. This encoder encodes the local action observation history τ into a vector h_τ that is parameterized by θ_{τ_β} for both networks. The role selector $f_\beta(h_\tau; \theta_\beta)$ is a fully connected neural network, parameterized by θ_β, which maps the vector h_τ to $z_\tau \in \mathbb{R}^d$ with the same length as the action representation. For the role selector, the expected payoff of agent i when choosing role ρ_j with observation state τ_i is denoted as $Q_i^\beta(\tau_i, \rho_j)$:

$$Q_i^\beta\left(\tau_i,\ \rho_j\right) = z_{\tau_i}^T z_{\rho_j} \tag{5}$$

Considering the similarity between the observation history and role representation of the agents, for each agent i, we compute the similarity between its action–observation history representation h_τ and the role representation $z_\rho := \left[z_{\rho_j}\right]_{j=1}^k$, i.e., the similarity $\left(h_\tau, z_{\rho_j}\right) = \left(z_{\tau_i}^T z_{\rho_j}\right) / \left(||h_\tau|| \, ||z_{\rho_j}||\right)$. Specifically, we approximate the cosine similarity using $Q_i^\beta\left(\tau_i,\ \rho_j\right)$ and apply the softmax function to obtain the categorical distribution of role selection $p\left(\rho|h_\tau, z_\rho\right) := \left[p\left(\rho_j|h_\tau, z_\rho\right)\right]_{j=1}^k$. Here, $p\left(\rho_j|h_\tau, z_\rho\right)$ is the probability that agent i selects role ρ_j with:

$$p\left(\rho_j|h_\tau, z_\rho\right) = \frac{\exp\left(Q_i^\beta\left(\tau_i,\ \rho_j\right)\right)}{\sum_{i=1}^n \exp\left(Q_i^\beta\left(\tau_i,\ \rho_j\right)\right)} \tag{6}$$

where $\exp(-)$ is the exponential function. To make the role selection process trainable, we use the Straight-Through Gumbel-Softmax Estimator to sample the role ρ_j, as drawing a role directly from a categorical distribution is indistinguishable.

To better coordinate role assignment and solve the role assignment problem in multi-agent learning, we use the QMIX method to estimate the global Q value of the joint action value function Q_{tot}^β using Q_i^β to generate the parameters ξ_β of the hybrid network using the current state as input and the hypernetwork [54]. We then use the return gain from the previous c-step $R_{LSTRR}^{(c)}$ plus the Q_{tot} value of the optimal role selection after step c, minus the Q_{tot} of the current prediction, squared about all historical experiences in the replay buffer, taking expectations as the loss function. The updated role selector f_β is learned by sampling multiple transitions from the replay buffer and minimizing the TD loss:

$$\begin{aligned}
&\mathcal{L}_\beta\left(\theta_{\tau_\beta}, \theta_\beta, \xi_\beta\right) \\
&= \mathbb{E}_D\left[\left(R_{LSTRR}^{(c)} + \gamma \max_{\rho'} \bar{Q}_{tot}^{\beta}\left(s_{t+c}, \rho'; \theta^-, \xi^-\right)\right.\right. \\
&\quad \left.\left. - Q_{tot}^\beta\left(s_t, \rho_t; \theta, \xi\right)\right)^2\right]
\end{aligned} \tag{7}$$

ρ in Eq. (7) refers to all the roles that can be chosen, while θ^- and ξ^- are target network parameters that are copied periodically from the current network and remain constant over multiple iterations. $\bar{Q}_{\text{tot}}^\beta$ is a target network with $\rho = \langle \rho_1, \rho_2, \ldots \rho_j \rangle$ as the joint set of roles for all agents, and the expectation is estimated using a uniform sample of the replay buffer $\mathcal{D}$:

$$\begin{aligned}
&Q_{\text{tot}}^\beta(s, \rho; \theta, \xi) = \sum_{i=1}^n Q_i^\beta(\tau_i, \rho_i; \theta_i) \\
&\frac{\partial Q_{\text{tot}}^\beta(s, \rho; \theta, \xi)}{\partial Q_i^\beta(\tau_i, \rho_i; \theta_i)} \geq 0, \forall i \in N
\end{aligned} \tag{8}$$

The monotonicity constraint in Eq. (8) ensures that maximizing joint Q_{tot}^β and maximizing individual Q_i^β are equivalent, meaning that the best individual action remains the same as the best joint action.

Since the role categories lead to role-specific reward ranges, a regularizer is available at each training step c to control the training stability:

$$\begin{aligned}
R_{LSTRR}^{(c)} = \frac{1}{k} \sum_{j=1}^{k} \left(Q_i^\beta\left(\tau_i, \rho_j\right) - \sum_{c=0}^{T-c} \mu_j^t r^{(t+c)}\right)^2; \\
\text{for } k \leq K.
\end{aligned} \tag{9}$$

where $LSTRR$ represents the long-term and short-term reward roles. Here $Q_i^\beta(\tau_i, \rho_j)$ is an estimate of the Q value associated with the role ρ_j, and $R_j = \sum_{c=0}^{T-c} \mu_j^t r^{(t+c)}$ is the discounted reward for the role ρ_j. This regularizer is used in the centralized training process while we know the rewards. Without loss of generality, we assume that $\mu_1, \mu_2, \ldots, \mu_k$ is a decreasing sequence (from long-term to short-term horizon).

At each time step, agent i will choose a role. When an agent frequently changes roles in adjacent time steps, it can cause instability during training. To smooth the role selection

of agents and stabilize training, we introduce regularizers that minimize the KL divergence between the role selection distributions of adjacent time steps:

$$\mathcal{L}_h\left(\theta_e, \theta_h\right) = \mathbb{E}_{\mathcal{D}}\left[\sum_a D_{KL}\left(p\left(\rho|h_\tau, z_\rho\right)||p'\left(\rho'|h'_\tau, z'_\rho\right)\right)\right] \tag{10}$$

where $p'\left(\rho'|h'_\tau, z'_\rho\right)$ is the role selection distribution for the next time step and $D_{KL}\left(\cdot||\cdot\right)$ is the KL divergence operator whose sum is carried over all agents.

The overall optimization objective for role selection is:

$$\mathcal{L}_\beta\left(\theta_{\tau_\beta}, \theta_\beta, \xi_\beta\right) = \mathcal{L}_\beta\left(\theta_{\tau_\beta}, \theta_\beta, \xi_\beta\right) + \lambda_h \mathcal{L}_h\left(\theta_e, \theta_h\right) \tag{11}$$

where λ_h is the positive coefficient of this regularizer.

Representation-related role strategy selection

After an agent chooses a role, it maintains the same role for the next c time steps, and its action space is restricted. Each role ρ_j is associated with a role policy $\pi_{\rho_j} : T \times A_j \rightarrow [0, 1]$, defined by the restricted action space. The policy parameters for each role are learned separately, as illustrated in Fig. 1c. Generally, agents playing the same role can learn policy parameters from each other, but the parameters are different for different roles.

Similar to the role selector, we learn the role policy using a deep Q network that estimates the Q value of each action directly. Q values based on the effects of actions can leverage information about the differences and contributions of actions. The role policy $f_{\rho_j}(h_\tau; \theta_{\rho_j})$ is a fully connected network that maps h_τ to z_τ to obtain the observed representation $z_\tau \in \mathbb{R}^d$, and the Q value obtained by making an inner product of the observed representation z_τ and the representation z_{a_k} of the role is the agent i chooses the value of an original action a_k. The value function $Q_i(\tau_i, a_k)$ of the action a_k executed by an agent after selecting the role ρ_j when the observed state is τ_i:

$$Q_i\left(\tau_i, a_k\right) = z_{\tau_i}^T z_{a_k} \tag{12}$$

To learn Q_i by global reward, we again input the local Q values into a QMIX-style mixing network to estimate the global action values, $Q_{\text{tot}}(s, a)$. The parameters of the mixing network are denoted by ξ_ρ. We minimize the TD loss to learn the update role policy f_{ρ_j}. The Q_{tot} value of the optimal action selection is added to the return gain r, and the Q_{tot} of the current prediction is subtracted, and all historical experiences about the replay buffer are squared to take the expectation as the loss function. We minimize the TD loss to learn the update role policy f_{ρ_j}:

$$\begin{aligned}&\mathcal{L}_\rho\left(\theta_{\tau_\rho}, \theta_\rho, \xi_\rho\right)\\ &= \mathbb{E}_{\mathcal{D}}\left[\left(r(s_t, a_t) + \gamma \max_{a'} \bar{Q}_{\text{tot}}\left(s', a'; \theta^-, \xi^-\right)\right.\right.\\ &\quad \left.\left.- Q_{\text{tot}}(s, a; \theta, \xi)\right)^2\right]\end{aligned} \tag{13}$$

a in Eq. (13) represents all optional actions of agent i, $\bar{Q}_{\text{tot}}$ is a target network, θ_ρ are the parameters of all role policies, and the expectation is estimated using a uniform sample of the same replay buffer $\mathcal{D}$ as the role selector. Therefore, each agent only trains the policy parameters of its selected role. In this way, agents with similar abilities tend to choose the same roles, can share their experience, and speed up training, ultimately improving performance.

Experiment results

Details

We selected the SMAC benchmark [12] as our test environment, which is a widely used benchmark for MARL and leverages the StarCraft II Learning Environment (SC2LE) to provide a challenging platform for solving both competitive and cooperative multi-agent problems. The agent's action space in the SMAC benchmark includes four basic movement directions (up, down, left, and right), selecting an enemy to attack, stopping, and not acting at each time step. However, agents can only attack enemies within their shooting range. Therefore, if there are n_e enemies on the map, the action space for each allied unit consists of $n_e + 6$ discrete actions. Our goal is to maximize the win rate, which is defined as the ratio of games won to games played. Specifically, in our experiments, we measure the percentage of episodes in which our proposed method defeats all enemy units within the time limit, which we refer to as the test victory rate.

We use RMSprop for optimization, and all hyperparameters and structural details are presented in Table 1. The episodes are sampled from the replay buffer, and the role policy and role-selected mixing network share the same architecture. The parameters of the mixing network are set to the same values as QMIX [55]. We employ the default reward and observation settings of the SMAC benchmark in our experiments. The baseline uses the code provided by the authors, and we fine-tuned the hyperparameters on the SMAC benchmark. All experiments were conducted on NVIDIA RTX 2060 GPUs, and training times ranged from 12 to 48 h.

To determine the role action space, we utilize k-means clustering. The number of clusters, denoted as k, is considered as a hyperparameter. We set k to 3 for maps with

Table 1 Hyperparameter settings and structure details

Parameter	Value
Learning rate	$5 * 10^{-4}$
Scaling factor (λ_e)	10
Discounted discount (γ)	0.99
Exploration	ε-greedy
Buffer size (episodes)	50k
Number of batches	32
Target update coefficient (τ)	0.005
Number of units in hidden layer	64
GRU hidden state	64
Nonlinearity	ReLU
Hidden layer with ReLU activation	32
Length of the action representation (d)	20
Number of clusters	3 or 5

homogeneous enemies and 5 for maps with heterogeneous enemies.

Performance

For the purpose of evaluation, all experiments in this section were conducted using different random seeds, and the median performance was reported. The SMAC benchmark maps were classified into three difficulty levels: easy, hard, and super hard. In this study, we chose two maps from each difficulty level of SMAC, specifically, 3s5z and 10 m_vs_11m from the easy level, 2c_vs_64zg and 5 m_vs_6m from the hard level, and corridor and 27 m_vs_30m from the super hard level. The details of each map can be found in Table 2. The hard and super-hard maps are typical examples of challenging exploration tasks. We conducted experiments in six scenarios to evaluate the performance of our method in the SMAC experimental environment, and compared it with value-based multi-agent reinforcement learning (MARL) algorithms QMIX [55] and VDN [56]. The results are shown in Figs. 2, 3, 4. After 2 M training steps, our method outperformed all baselines by at least 5% in four out of six scenarios.

The experimental results for the easy maps are depicted in Fig. 2. In Fig. 2a, our method improves the win rate by approximately 10% compared to one baseline and performs similarly to another baseline, but our method does not converge as quickly as the baseline. In Fig. 2b, our method performs similarly to the baseline, with a win rate improvement of 2–5% over the baseline. Overall, on easy maps, our method performs comparably to the baseline, but requires more samples to achieve similar performance. We speculate that this is because easy maps do not require much exploration—allied agents simply engage in combat and destroy enemy units to win rewards, whereas our method still does a lot of exploration, and the benefits of the restricted action space are not apparent enough to outperform the baseline.

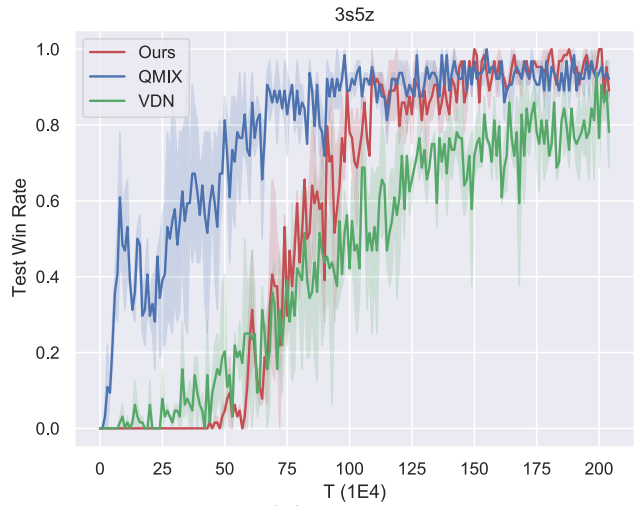
Figure 3 shows the performance of our method on hard maps. In both maps of Fig. 3a, b, our method outperforms the baselines. In Fig. 3a, our method improves the win rate by about 5% on average, and the convergence rate is even faster than the baseline method by about 28%. In Fig. 3b, our method outperforms the baseline method by approximately 25% on average, although the convergence speed is slower. Overall, on hard maps, our method improves the win rate by an average of 20% over the baseline.

The experimental results for super-hard maps are shown in Fig. 4. In both maps in Fig. 4a, b, our method performs significantly better than the baselines, especially in Fig. 4a. In Fig. 4a, our method improves the win rate by approximately 70% on average, and in Fig. 4b, our method improves the win rate by about 40%. Overall, on super hard maps, our algorithm improves the win rate by an average of 55% over the baseline algorithm.

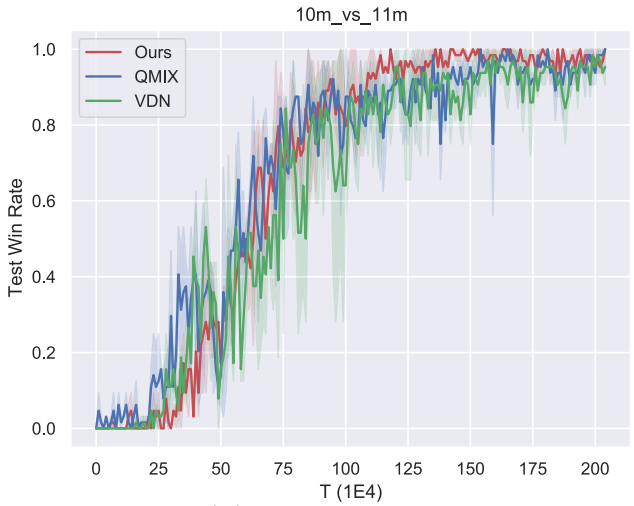
In summary, our method achieves good results on the SMAC benchmark and performs well in various scenarios, especially in the super hard and hard scenarios, where it surpasses all baselines. Our method outperforms the baseline by a wide margin on maps that require more exploration, such as a corridor. These results demonstrate that our method can effectively explore and solve complex tasks, as we anticipated.

Table 2 SMAC map details

Name	Ally Units	Enemy units	Type
3s5z	3 Stalkers and 5 Zealots	3 Stalkers and 5 Zealots	Heterogeneous and symmetric
10 m_vs_11m	10 Marines	11 Marines	Homogeneous and asymmetric
2c_vs_64zg	2 Colossi	64 Zerglings	Micro-trick: positioning
5 m_vs_6m	5 Marines	6 Marines	Homogeneous and asymmetric
27 m_vs_30m	27 Marines	30 Marines	Homogeneous and asymmetric
Corridor	6Z ealots	24 Zerglings	Micro-trick: wall off

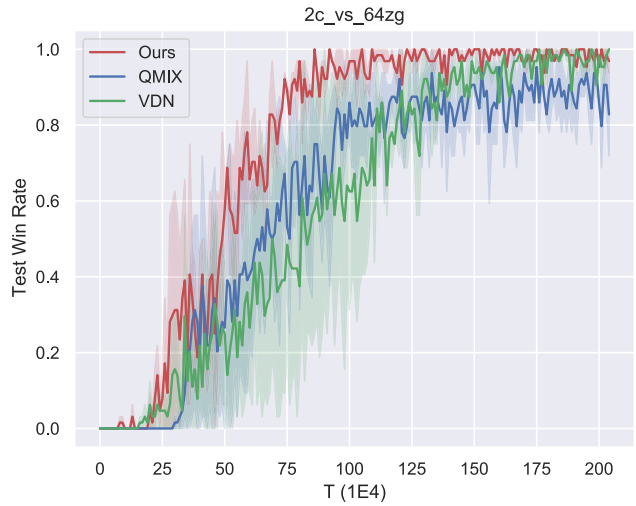


(a) 3s5z

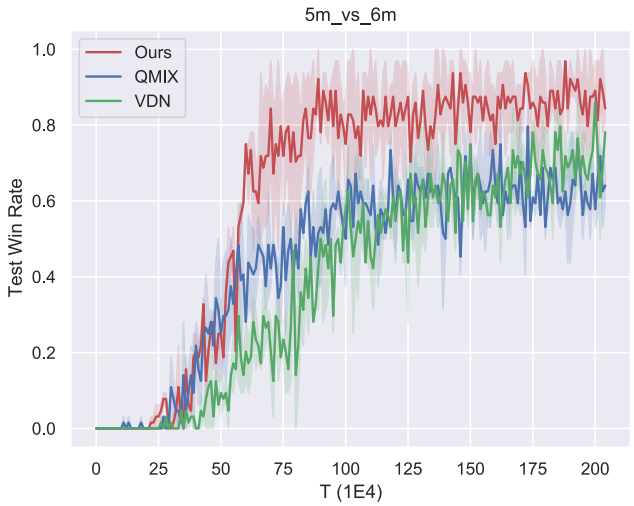


(b) 10m_vs_11m

Fig. 2 Performance comparison with baselines on easy maps



(a) 2c_vs_64zg



(b) 5m_vs_6m

Fig. 3 Performance comparison with baselines on hard maps

Ablation

Our approach consists of three components, namely (A) a restricted action space, (B) clustering using action differences and action contributions, and (C) integration of action representations into role selection and role policies. To test components A and B, we keep the rest of the framework unchanged and let the role action space become a space containing all actions or a space of random actions. For component B, we test the role action space in the case that it contains all actions. For component C, action representations are eliminated to learn role selection and role strategies using traditional Q-networks.

We chose one map from each of the three different difficulty levels for the ablation experiments, and the results are shown in Fig. 5. It can be seen that the performance using the traditional deep Q-network (No Action Repr.) method is closest to the original model, with an average performance reduction of 55%, outperforming the ablation without restricting the role action space (Full Action Spaces). It can be concluded that integrating action representations into role selection and role policy has a significant impact on the performance of our method, and the restricted action space is closely related to the performance of the new follow-through one. The performance of the method with Random Restricted Action Spaces (Random Restricted Action Spaces) is the worst of the three ablation experiments, with an average 60% reduction in performance compared to the original model, highlighting the importance of using action differences and action contributions to decompose the joint action space.

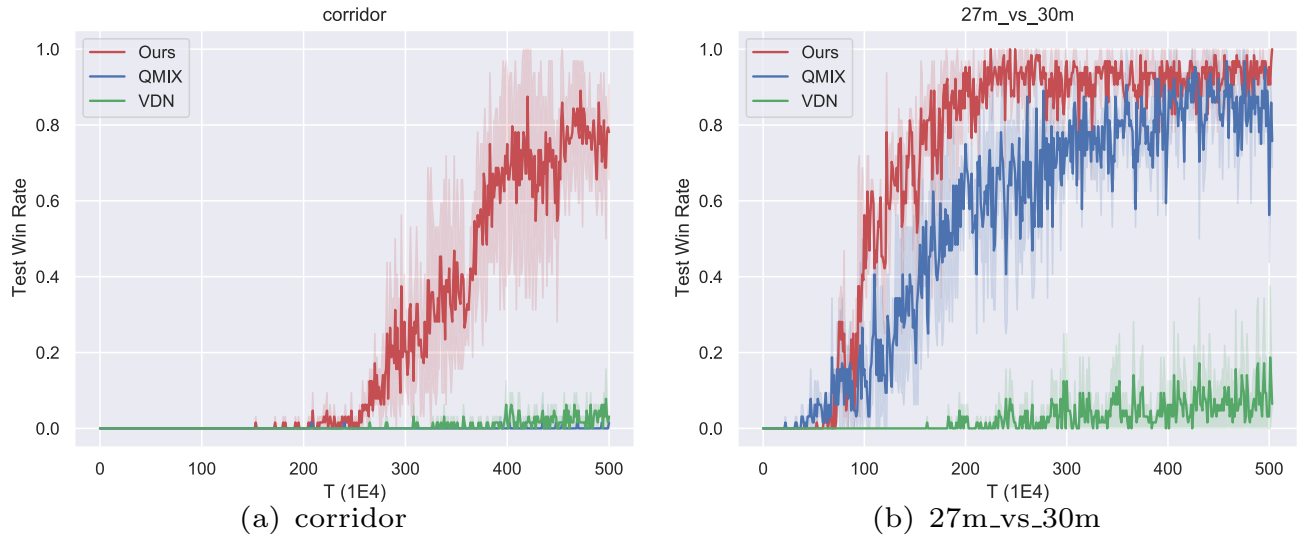


Fig. 4 Performance comparison with baselines on super hard maps

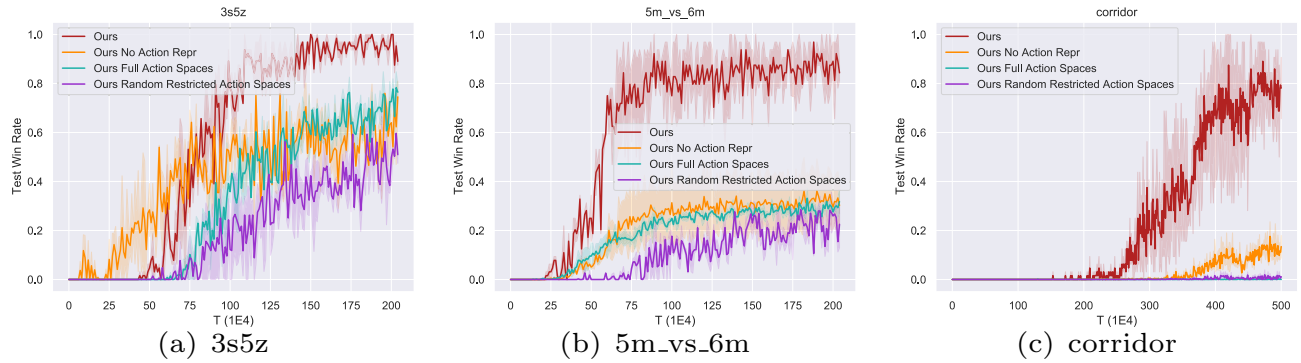


Fig. 5 Ablation studies regarding each component of ours

In conclusion, all three methods that ablate different components perform worse than the unablated methods in terms of performance. It can be seen that limiting the role action space and integrating action representations into roles and strategies play a key role in the performance of our methods.

Conclusion

Discovering a set of roles that effectively decompose tasks is a thorny problem that hinders the scalability of using roles to decompose team tasks. In this paper, we propose to divide roles to decompose team tasks, decompose the action space based on action differences and action contributions, and role selection enables agents to play roles dynamically based on reward horizons. Experimental results on SMAC show that our method has significant performance improvement over other MARL methods, with an average 20% improvement in win rate, especially excellent performance on hard and super hard maps. In addition, ablation experiments were conducted on three components of our method to verify the impact of different components of our method. The results of the experiments showed that the win rate decreased by 55–60% on average after ablating the components, which proves that three different components play a key role in the performance of our method. However, our framework may not perform optimally under certain conditions. Specifically, our method may not be effective in scenarios where task decomposition is unnecessary and victory can be achieved through simple strategies that do not require extensive exploration. In such cases, our method may not yield significant performance improvements. Our role selection and strategy selection are based on the traditional QMIX method, which can theoretically be replaced with other QMIX variant methods, and we hope to be able to achieve better performance with different combinations in the future. Our approach provides new insights into building effective multi-agent team task decomposition in dynamic environments.

Acknowledgements This research was funded by the National Project of Foreign Experts (No. G2022033007L), The Bagui Scholars Program

of Guangxi Zhuang Autonomous Region(No. 2019A08), Postgraduate Research & Practice Innovation Program of Jiangsu Province (KYCX22_3504).

References

1. Claus C, Boutilier C (1998) The dynamics of reinforcement learning in cooperative multiagent systems. In: AAAI/IAAI, vol 2, pp 746–752
2. Tan M (1993) Multi-agent reinforcement learning: independent vs. cooperative agents. In: Proceedings of the tenth international conference on machine learning, pp 330–337
3. Foerster J, Assael IA, De Freitas N, Whiteson S (2016) Learning to communicate with deep multi-agent reinforcement learning. In: Advances in neural information processing systems, vol 29
4. Gupta JK, Egorov M, Kochenderfer M (2017) Cooperative multi-agent control using deep reinforcement learning. In: International conference on autonomous agents and multiagent systems. Springer, pp 66–83
5. Rashid T, Samvelyan M, Schroeder C, Farquhar G, Foerster J, Whiteson S (2018) Qmix: monotonic value function factorisation for deep multi-agent reinforcement learning. In: International conference on machine learning, PMLR, pp 4295–4304
6. Mahajan A, Rashid T, Samvelyan M, Whiteson S (2019) Maven: multi-agent variational exploration. In: Advances in neural information processing systems, vol 32
7. Das A, Gervet T, Romoff J, Batra D, Parikh D, Rabbat M, Pineau J (2019) Tarmac: targeted multi-agent communication. In: International conference on machine learning. PMLR, pp 1538–1546
8. Wang J, Ren Z, Han B, Ye J, Zhang C (2020) Towards understanding linear value decomposition in cooperative multi-agent q-learning
9. Butler E (2012) The condensed wealth of nations. Centre for Independent Studies, NSW
10. Bonjean N, Mefteh W, Gleizes M-P, Maurel C, Migeon F (2014) Adelfe 2.0. Handbook on agent-oriented design processes. Springer, Berlin
11. Wang T, Dong H, Lesser V, Zhang C (2020) Roma: multi-agent reinforcement learning with emergent roles. arXiv preprint arXiv:2003.08039
12. Samvelyan M, Rashid T, De Witt CS, Farquhar G, Nardelli N, Rudner TG, Hung C-M, Torr PH, Foerster J, Whiteson S (2019) The starcraft multi-agent challenge. arXiv preprint arXiv:1902.04043
13. Kraemer L, Banerjee B (2016) Multi-agent reinforcement learning as a rehearsal for decentralized planning. Neurocomputing 190:82–94
14. Oliehoek FA, Spaan MT, Vlassis N (2008) Optimal and approximate q-value functions for decentralized pomdps. J Artif Intell Res 32:289–353
15. Son K, Kim D, Kang WJ, Hostallero DE, Yi Y (2019) Qtran: learning to factorize with transformation for cooperative multi-agent reinforcement learning. In: International conference on machine learning. PMLR, pp 5887–5896
16. Yang Y, Hao J, Liao B, Shao K, Chen G, Liu W, Tang H (2020) Qatten: a general framework for cooperative multiagent reinforcement learning. arXiv preprint arXiv:2002.03939
17. Wang J, Ren Z, Liu T, Yu Y, Zhang C (2020) Qplex: duplex dueling multi-agent q-learning. arXiv preprint arXiv:2008.01062
18. Lowe R, Wu YI, Tamar A, Harb J, Pieter Abbeel O, Mordatch I (2017) Multi-agent actor-critic for mixed cooperative-competitive environments. In: Advances in neural information processing systems, vol 30
19. Yang Y, Luo R, Li M, Zhou M, Zhang W, Wang J (2018) Mean field multi-agent reinforcement learning. In: International conference on machine learning. PMLR, pp 5571–5580
20. Guo F, Wu Z (2022) Learning maximum entropy policies with qmix in cooperative marl. In: 2022 IEEE 2nd international conference on electronic technology, communication and information (ICETCI). IEEE, pp 357–361
21. Hu J, Jiang S, Harding SA, Wu H, Liao S-W (2021) Rethinking the implementation tricks and monotonicity constraint in cooperative multi-agent reinforcement learning. arXiv e-prints, 2021
22. Botelho SC, Alami R (1999) M+: a scheme for multi-robot cooperation through negotiated task allocation and achievement. In: Proceedings 1999 IEEE international conference on robotics and automation (Cat. No. 99CH36288C), vol 2. IEEE, pp 1234–1239
23. Chen J, Yang Y, Wei L (2010) Research on the approach of task decomposition in soccer robot system. In: 2010 International conference on digital manufacturing & automation, vol 2. IEEE, pp 284–289
24. Zlot R, Stentz A (2003) Multirobot control using task abstraction in a market framework. In: Collaborative technology alliances conference
25. Dias MB, Zlot R, Kalra N, Stentz A (2006) Market-based multirobot coordination: a survey and analysis. Proc IEEE 94(7):1257–1270
26. Dorigo M, Floreano D, Gambardella LM, Mondada F, Nolfi S, Baaboura T, Birattari M, Bonani M, Brambilla M, Brutschy A et al (2013) Swarmanoid: a novel concept for the study of heterogeneous robotic swarms. IEEE Robot Autom Mag 20(4):60–71
27. Kiener J, Von Stryk O (2010) Towards cooperation of heterogeneous, autonomous robots: a case study of humanoid and wheeled robots. Robot Auton Syst 58(7):921–929
28. Li X, Dang S, Li K, Liu Q (2010) Multi-agent-based battlefield reconnaissance simulation by novel task decompositionand allocation. In: 2010 5th international conference on computer science & education. IEEE, pp 1410–1414
29. Cobo LC, Isbell CL Jr, Thomaz AL (2012) Automatic task decomposition and state abstraction from demonstration. Georgia Institute of Technology, Atlanta
30. Hu T, Messelodi S, Lanz O (2014) Dynamic task decomposition for probabilistic tracking in complex scenes. In: 2014 22nd International conference on pattern recognition. IEEE, pp 4134–4139
31. Zlot R, Stentz A (2006) Market-based multirobot coordination for complex tasks. Int J Robot Res 25(1):73–101
32. Evertsz R, Thangarajah J (2020) A framework for engineering human/agent teaming systems. In: Proceedings of the AAAI conference on artificial intelligence, vol 34, pp 2477–2484
33. Wooldridge M, Jennings NR, Kinny D (2000) The gaia methodology for agent-oriented analysis and design. Auton Agent Multi-Agent Syst 3(3):285–312
34. Omicini A (2000) Soda: societies and infrastructures in the analysis and design of agent-based systems. International workshop on agent-oriented software engineering. Springer, Berlin, pp 185–193

35. Padgham L, Winikoff M (2002) Prometheus: a methodology for developing intelligent agents. International workshop on agent-oriented software engineering. Springer, Berlin, pp 174–185
36. Cossentino M, Gaglio S, Sabatucci L, Seidita V (2005) The passi and agile passi mas meta-models compared with a unifying proposal. International central and eastern European conference on multi-agent systems. Springer, Berlin, pp 183–192
37. Zhu H, Zhou M (2008) Role-based multi-agent systems. Personalized information retrieval and access: concepts, methods and practices. IGI Global, USA, pp 254–285
38. Spanoudakis N, Moraitis P (2010) Using aseme methodology for model-driven agent systems development. International workshop on agent-oriented software engineering. Springer, New York, pp 106–127
39. DeLoach SA, Garcia-Ojeda JC (2010) O-mase: a customisable approach to designing and building complex, adaptive multi-agent systems. Int J Agent-Oriented Softw Eng 4(3):244–280
40. Sun C, Liu W, Dong L (2020) Reinforcement learning with task decomposition for cooperative multiagent systems. IEEE Trans Neural Netw Learn Syst 32(5):2054–2065
41. Lhaksmana KM, Murakami Y, Ishida T (2018) Role-based modeling for designing agent behavior in self-organizing multi-agent systems. Int J Softw Eng Knowl Eng 28(01):79–96
42. Bresciani P, Perini A, Giorgini P, Giunchiglia F, Mylopoulos J (2004) Tropos: an agent-oriented software development methodology. Auton Agent Multi-Agent Syst 8(3):203–236
43. Wilson A, Fern A, Tadepalli P (2010) Bayesian policy search for multi-agent role discovery. In: Twenty-fourth AAAI conference on artificial intelligence
44. Christianos F, Papoudakis G, Rahman MA, Albrecht SV (2021) Scaling multi-agent reinforcement learning with selective parameter sharing. In: International conference on machine learning. PMLR, pp 1989–1998
45. Le HM, Yue Y, Carr P, Lucey P (2017) Coordinated multi-agent imitation learning. In: International conference on machine learning. PMLR, pp 1995–2003
46. Wang T, Gupta T, Mahajan A, Peng B, Whiteson S, Zhang C (2020) Rode: learning roles to decompose multi-agent tasks. arXiv preprint arXiv:2010.01523
47. Nguyen D, Nguyen P, Venkatesh S, Tran T (2022) Learning to transfer role assignment across team sizes. arXiv preprint arXiv:2204.12937
48. Hu S, Xie C, Liang X, Chang X (2022) Policy diagnosis via measuring role diversity in cooperative multi-agent rl. In: International conference on machine learning. PMLR, pp 9041–9071
49. Kulkarni TD, Narasimhan K, Saeedi A, Tenenbaum J (2016) Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In: Advances in neural information processing systems, vol 29
50. Ong SC, Png SW, Hsu D, Lee WS (2009) Pomdps for robotic tasks with mixed observability. Robot Sci Syst 5:4
51. Oliehoek FA, Amato C (2016) A concise introduction to decentralized POMDPs. Springer, Berlin
52. Kurach K, Raichuk A, Stańczyk P, Zajac M, Bachem O, Espeholt L, Riquelme C, Vincent D, Michalski M, Bousquet O, et al (2020) Google research football: a novel reinforcement learning environment. In: Proceedings of the AAAI conference on artificial intelligence, vol 34, pp 4501–4510
53. Hausknecht M, Stone P (2015) Deep recurrent q-learning for partially observable mdps. In: 2015 Aaai fall symposium series
54. Ha D, Dai A, Le QV (2016) Hypernetworks. arXiv preprint arXiv:1609.09106
55. Rashid T, Samvelyan M, De Witt CS, Farquhar G, Foerster J, Whiteson S (2020) Monotonic value function factorisation for deep multi-agent reinforcement learning. J Mach Learn Res 21(1):7234–7284
56. Sunehag P, Lever G, Gruslys A, Czarnecki WM, Zambaldi V, Jaderberg M, Lanctot M, Sonnerat N, Leibo JZ, Tuyls K, et al (2017) Value-decomposition networks for cooperative multi-agent learning. arXiv preprint arXiv:1706.05296