ECAI 2023
K. Gal et al. (Eds.)

doi:10.3233/FAIA230298

Selective Learning for Sample-Efficient Training in Multi-Agent Sparse Reward Tasks

Xinning Chen[a;*], **Xuan Liu**[a;**], **Yanwen Ba**[a], **Shigeng Zhang**[b], **Bo Ding**[c] **and Kenli Li**[a]

[a]College of Computer Science and Electronic Engineering, Hunan University, China
[b]School of Computer Science and Engineering, Central South University, China
[c]School of Computer Science, National University of Defense Technology, China

Abstract. Learning effective strategies in sparse reward tasks is one of the fundamental challenges in reinforcement learning. This becomes extremely difficult in multi-agent environments, as the concurrent learning of multiple agents induces the non-stationarity problem and sharply increased joint state space. Existing works have attempted to promote multi-agent cooperation through experience sharing. However, learning from a large collection of shared experiences is inefficient as there are only a few high-value states in sparse reward tasks, which may instead lead to the curse of dimensionality in large-scale multi-agent systems. This paper focuses on sparse-reward multi-agent cooperative tasks and proposes an effective experience-sharing method, **M**ulti-**A**gent **S**elective **L**earning (MASL), to boost sample-efficient training by reusing valuable experiences from other agents. MASL adopts a retrogression-based selection method to identify high-value traces of agents from the team rewards, based on which some recall traces are generated and shared among agents to motivate effective exploration. Moreover, MASL selectively considers information from other agents to cope with the non-stationarity issue while enabling efficient training for large-scale agents. Experimental results show that MASL significantly improves sample efficiency compared with state-of-the-art MARL algorithms in cooperative tasks with sparse rewards.

1 Introduction

Deep reinforcement learning (DRL) has shown its advantage in sequential decision-making control tasks by employing neural networks as approximators, such as Atari games, game theory and robot control [6, 8, 13, 18]. However, recent success depends heavily on a well-formed reward function that provides explicit feedback to each agent at each step. For some complex tasks with sparse reward, such as autonomous driving and robotic control, learning an optimal policy becomes extremely difficult for agents due to the lack of feedback signals. Sparse rewards are delayed, which provide feedback to agents only in a few states (e.g., when the agent reaches a goal), while in most cases, agents are not rewarded. Discovering high-value states in sparse reward tasks is a hard exploration problem for agents, which has not been well studied in the RL domain.

For single-agent environments, previous studies on sparse reward have improved exploration efficiency by using additional supervised signals, such as expert demonstrations [37, 40], intrinsic rewards [1, 29, 38]. However, without considering the complex interactions between multiple agents, these methods cannot be directly applied to multi-agent settings. In multi-agent tasks, the sparse reward challenge is aggravated by the need for policy coupling and the non-stationarity of environments.

To promote cooperation in large state space, existing multi-agent reinforcement learning (MARL) algorithms take advantage of experience sharing. The centralized training with decentralized execution (CTDE) paradigm is widely used to enable agents to share their experience during centralized training [9, 19, 24]. However, fully sharing experience among agents may lead to the curse of dimensionality as the joint state-action space grows exponentially with the number of agents [27]. Recent works have tried to simplify the learning process by adopting techniques of inverse kinematic [17, 30], attention mechanism [14, 15, 23], mean field theory [39] and dropout [16]. However, useful information may be overlooked in the process of reducing interaction. How to maximize the valuable experiences while simplifying the interaction remains a question, which is particularly important in the tasks with sparse reward.

In this paper, we focus on sparse reward in cooperative multi-agent scenarios, where both the challenges brought by sparse reward and inherent non-stationarity greatly reduce learning efficiency. We propose a method called multi-agent selective learning (MASL) to achieve sample-efficient training. The key idea of MASL is to select only valuable experiences of other agents instead of the whole huge trajectory space to accelerate the learning process. First, inspired by the advanced idea of using a backtracking model to improve sample efficiency[10], we introduce a centralized backtracking model for multiple agents, which generates recall traces from high-value samples. More importantly, each agent not only speeds up learning by imitating its own recall traces, but also shares the traces with other agents for aiding effective exploration. Second, we propose to selectively use information of the related agents but not all, which effectively mitigates the non-stationarity of the multi-agent environments and enhances the scalability of our approach in scenarios with more agents. Last but not least, considering that it is difficult to identify high-value experiences when all agents obtain a shared team reward, we specifically consider fully cooperative tasks with shared team reward and design a retrogression-based selection method to overcome the difficulty of recognizing contributors from the shared reward.

The contribution of this paper is summarized as fourfold:

1) We propose multi-agent selective learning (MASL), an effective method to boost learning efficiency in multi-agent cooperative tasks

* Email: chenxinning@hnu.edu.cn
** Corresponding Author. Email: xuan_liu@hnu.edu.cn

with sparse rewards.

2) We introduce a centralized backtracking model to guide cooperative exploration and exploit a retrogression-based selection method to extract high-value agent trajectories from the team's success.

3) We propose a selector to selectively use information from other agents based on their relevancy to improve the training speed of centralized learning.

4) We conduct several cooperative multi-agent tasks with sparse reward setting to evaluate the performance of MASL. Experiment results show that MASL achieves higher sample efficiency than the baselines, especially in sparse tasks with large-scale agents.

2 Background

2.1 Multi-agent Reinforcement Learning

Multi-agent tasks are generally modeled as a partially observable Markov Game (POMG)[21]. When agents face a cooperative task with a shared reward function, the POMG is then known as decentralized Partially Observable Markov decision process (DEC-POMDP). It can be formally defined as a tuple $\langle S, A, O, Z, P, r, \gamma \rangle$, where S is the set of environment state s. Each agent $i \in \{1 \cdots N\}$ chooses its own action $a_i \in A$, forming a joint action $\boldsymbol{a} \in \mathbf{A} \equiv A^n$ which induces a state transition according to the state transition function $P(s'|s, \boldsymbol{a}) : S \times \mathbf{A} \times S \to [0, 1]$. Each agent receives a partial observation $o_i \in Z$, which is drawn from the observation function $O(s, i) : S \times N \to Z$. Each agent i obtains a reward as a function of the state and agent's action $r_i : S \times \mathbf{A} \to \mathbb{R}$.

In cooperative tasks, each agent of a team always shares the same reward, i.e., $r_1 = r_2 = \cdots = r_N$. The shared reward may lead to the credit assignment problem, which is a key problem that restricts agents from identifying and reusing their high-value experiences in sparse reward tasks. The agent aims at learning a policy $\pi_i(o_i)$ that maximizes the accumulation of expected future rewards $\mathbb{E}[R_i]$ where R_i is the discounted reward defined as $R_i = \sum_{t=0}^{T} \gamma^t r_i^t$ and T is the time horizon. $\gamma \in [0, 1)$ is the discount factor. We denote joint quantities over agents in bold.

2.2 Multi-Agent Deep Deterministic Policy Gradient

Applying single-agent RL algorithms to the multi-agent case causes environment instability. To alleviate this problem, multi-agent DDPG (MADDPG[24] is proposed to learn a centralized critic for each agent, which allows agents to obtain the other agents' observations and actions during training.

Specifically, the centralized critic for agent i is represented by $Q_i^{\boldsymbol{\mu}}(s, \boldsymbol{a})$ taking the environment state $s = (o_1, \cdots, o_N)$, and the joint action $\boldsymbol{a} = (a_1, \cdots, a_N)$ as inputs, where $\boldsymbol{\mu} = \{\mu_1, \cdots, \mu_{\mathrm{N}}\}$ is the set of all agents' policies parameterized by θ_i. The centralized critic parameterized by ω_i is updated based on:

$$\mathcal{L}(\omega_i) = \mathbb{E}_{s,a,r,s' \sim \mathcal{D}} \left[\left(Q_i^{\boldsymbol{\mu}}(s, \boldsymbol{a}) - y_i \right)^2 \right], \tag{1}$$

where $y_i = r_i + \gamma Q_i^{\boldsymbol{\mu}'}(s', \boldsymbol{a}')|_{a'_{\mathrm{j}} = \mu'_{\mathrm{j}}(o_{j'})}$ and $\boldsymbol{\mu}' = \{\mu'_1, \cdots, \mu'_{\mathrm{N}}\}$ is the set of target policies with delayed parameters ω'_i.

Note that the centralized critic is only used in the training phase, and the policy is decentralized based on local observation which is updated by maximizing the objective $J(\theta_i) = \mathbb{E}\left[\sum_{t=0}^{T} \gamma^t r_i^t\right]$ as:

$$\nabla_{\theta_i} J(\theta_i) = \mathbb{E}_{s,a \sim \mathcal{D}} \left[\nabla_{\theta_i} \mu_i(o_i) \nabla_{a_i} Q_i^{\boldsymbol{\mu}}(s, \boldsymbol{a}) |_{a_i = \boldsymbol{\mu}_i(o_i)} \right]. \tag{2}$$

2.3 Backtracking Model

Training a well-performing policy in sparse reward tasks through random exploration is challenging, as only a few states yield high value and thus the sample efficiency is low. [4, 10] simultaneously propose to learn a backtracking model to generate traces that lead to high value states. The generated recall traces are used to improve the agents' policy by guiding the agent to effectively explore the high-value states.

More concretely, given a high-value state s_{t+1}, the backtracking model $B_\phi = q_\phi(s_t, a_t|s_{t+1})$ parameterized by ϕ predicts a previous state s_t and an action a_t that given state s_t can lead to state s_{t+1}, thus produces sequences of (s_t, a_t)-tuples forming recall traces. The policy π_θ is updated based on trajectories collected by RL algorithm. Agents imitate recall traces given by:

$$\mathcal{L}_{\mathcal{I}} = \sum_{t=0}^{H} \log \pi_\theta(a_t|s_t), \tag{3}$$

where (s_t, a_t)-tuples are sampled from the recall traces stored in buffer $\mathcal{B}$ and H is the time horizon.

At each iteration, the backtracking model is trained with agent trajectories filtered by returns. For training stability with continuous-valued states, backtracking model models the density of state variation $\Delta s_t = s_t - s_{t+1}$ rather than the raw s_t and is updated as:

$$\mathcal{L}_{\mathcal{B}} = \log \prod_{t=0}^{H} q(\Delta s_t, a_t|s_{t+1}) \tag{4}$$

To build an effective backtracking model, it is required to collected high value states, which can be easily selected in single-agent RL algorithms based on the received rewards or Q-values estimated by the critic. However, in multiple-agent scenarios where agents share a team reward, how to identify which agent has visited a high value state remains an issue.

3 Methodology

As discussed before, both the challenge of sparse reward and non-stationarity hinder the learning process. To address these challenges, MASL utilizes two techniques: 1) it backtracks the trajectories of high-value states to help agents avoid the unguided exploration process to speed up training. 2) it selects the information of other K agents based on the relevancy, rather than all agents, stabilizing the environment appropriately.

MASL is built on the framework of centralized training and decentralized execution. As shown in Fig.1, each agent i learns an independent deterministic policy $\mu_i(a_i|o_i; \theta_i)$ parameterized by θ_i and a Q-network $Q_i(o_1, \cdots, o_N, a_1, \cdot, a_N; \omega_i)$ parameterized by ω_i, where o_i and a_i represent observation and action, respectively. More importantly, the selectors for agents and a centralized backtracking model B_ϕ are used for selective learning in the training phase. During the execution, the backtracking model, selectors and Q-value networks will be removed, allowing each agent to act based on its local policy.

3.1 Backtracking Based on Optimal Trajectory Selection

To improve sample efficiency in sparse reward tasks, we introduce the backtracking model in RL to multi-agent tasks. However, it is inefficient to directly apply the model to MARL because of the need to identify high-value states and the neglect of collaboration. Therefore, we mainly deal with two key problem: 1) How to improve multi-agent cooperation using the backtracking model? 2) How to identify high-value trajectory when all agents share a reward?

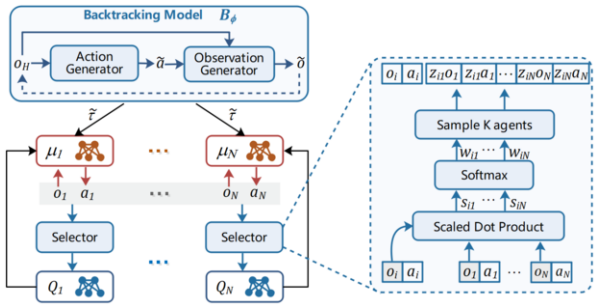


Figure 1. The architecture of MASL.

First, we design a global backtracking model that is shared among all agents to facilitate cooperative exploration. When an agent discovers a high-value state within the vast state space, we expect that all agents could benefit from the valuable experience. Instead of directly sharing past experiences, we utilize the high-value states to generate recall traces, which are then shared among agents to expedite the exploration process. An example is shown in Fig.2, where two agents are exploring the environment to reach the targets. When agent A2 discovers a target (e.g., a landmark) and successfully identifies a high-value state (i.e., observation in partially observable environments), A2 shares the recall traces generated from this high-value state to guide the other agent to reach the known high-value states from a new path. In this way, agent A1 can quickly recognize the target and acquire the individual ability to reach a certain target in the early stage. Second, we propose a retrogression-based selection method to select the optimal trajectories for backtracking.

Backtracking model for multi-agent scenarios. Consider a fully cooperative game with N agents with policies parameterized by $\boldsymbol{\theta} = \{\theta_1, \ldots, \theta_N\}$. Each agent i shares a team reward and only has a partial observation o_i, which is treated as the agent's private state.

In the training phase, we maintain a centralized backtracking model B_ϕ shared by all agents. The backtracking model B_ϕ utilizes agents' good private states to generates traces for N agents to train their policies. As shown in Fig.1, B_ϕ is composed of a observation generator and an action generator. Given a high-value observation o_i^H of agent i, the action generator $q\left(\tilde{a}_i|o_i^H\right)$ generates an action $\tilde{a}_i$ that may cause the high-value state. The observation generator $q\left(\Delta o_i|o_i^H, \tilde{a}_i\right)$ outputs the observation variation $\Delta o_i = \tilde{o}_i - o_i^H$ according to o_i^H and $\tilde{a}_i$, by which we get a previous observation $\tilde{o}_i$ indirectly and then further obtain a sequence of $(\tilde{o}_i, \tilde{a}_i)$-tuples as recall trace $\tilde{\tau}_i$ for policy training.

To achieve cooperation exploration, a natural idea is knowledge sharing. Therefore, each agent shares their recall traces with the other agents. We omit the the subscript of recall traces in the following content. Given observation $\tilde{o}$, the policy μ_i parameterized by θ_i is updated to guide agents to take action $\tilde{a}$ as:

$$L\left(\theta_i\right) = \log \mu_i(\tilde{\tau}) = \sum\nolimits_{t=0}^{M} \log \mu_i(\tilde{a}|\tilde{o}) \tag{5}$$

Since the backtracking model B_ϕ predicts previous actions and observations based on agents' high-value observations, the distribution of recall traces should match the distribution of the high-value trajectory as closely as possible. Given a high-value trajectory $\tau = (o^1, a^1, r^1, \cdots, o^T, a^T, r^T)$, B_ϕ is trained by:

$$\begin{aligned} \mathcal{L}\left(\phi\right) &= \log \prod\nolimits_{t=0}^{T} q\left(\Delta o_t, a_t|o_{t+1}\right) \\ &= \sum\nolimits_{t=0}^{T} \log q\left(a_t|o_{t+1}\right) + \log q\left(\Delta o_t|a_t, o_{t+1}\right). \end{aligned} \tag{6}$$

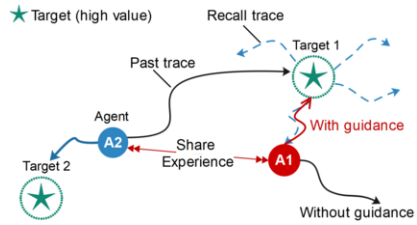


Figure 2. An example of exploiting optimal experience to provide guidance.

Note that agents only share the recall traces with other agents during training, and act independently during execution.

Retrogression-based trajectory selection. Effective identification of high-value agent trajectories is crucial for training both policies and backtracking models. When each agent receives an individual reward, it becomes easy to identify high-value states based on this reward. However, in fully cooperative environments where all agents receive the same reward, extracting high-value agent trajectories from the team's success poses a challenge. To address this, we propose a retrogression-based selection method for shared reward tasks to identify high-value agent trajectories. Inspired by the concept that one step backward for two steps forward, we deduce contributors by observing the change of shared reward after taking steps back.

When agents get a high value team reward on the journey of exploration, all the agents take a step backward in turn and observe how reward changes. Then, it is able to infer which agent contributes to the high-value reward (e.g., achieving the target). Specifically, according to the team reward obtained by agents, we define a dynamically changing baseline reward R^b as a high-value judgment criterion. Moreover, since it is always unrealistic to make the system state recovery, we use a rule-based strategy as an auxiliary policy for selecting the high-value trajectories. Given experiences $\left\{\left(o_i^t, a_i^t, o_i^{t+1}, r_i^t\right)\right\}_{i=1}^N$, if $r_i^t > R^b$, we test each agent in turn to identify the high value state in the set $\left\{o_i^{t+1}\right\}_{i=1}^N$: Agent i (from $i = 1$ to N) takes an action following a rule-based strategy $a_i = \pi_r(o_i)$, getting back to its previous position. And the other agents take actions to keep themselves staying on the spot. If the backward behavior of agent i leads to a reduction in the reward obtained, it is asserted that the agent i contributes to the team's reward. Therefore, we select the private state o_i^{t+1} of the agent i as a high value observation. In the end of each episode, the selected high-value agent trajectories are deposited to a high-value buffer $\tilde{B}$. Note that the rule-based strategy can be conducted based on the system principles or using some pre-training policies. Moreover, to maintain the stability of policy learning, the transitions collected by using the rule-based strategy will not be put into the replay buffer.

Considering that the policies of the agents co-evolve with training, it is inefficient to perform the selection process every time a high-value reward is obtained. Actually, with the improvement of policies, the team reward obtained by agents gradually increases. For a powerful team, setting a large baseline reward to promote them to explore a more optimal policy is necessary. Therefore, to make the retrogression-based trajectory selection process more efficient, we periodically update the baseline reward R^b to the highest reward obtained as a measure of learning progress. The retrogression-based trajectory selection process begins when $r_i^t > R^b$, and ends when all the contributors are identified ($r \leq 0$) or the game ends ($t \geq T$). Moreover, agents only execute the retrogression-based selection process once in an episode. Thus, it needs to expend N time steps (in the case that agent N is the contributor to great reward) at most interacting with the environment for selecting the high-value trajectories.

3.2 Training Based on Key Information Selection

Although sharing and imitating backtracking trajectories improves sample efficiency, agents still suffer from the unstable environment. Therefore, we allow agents to obtain information from the other agents for stable training, and select key information for learning efficiency and scalability.

Driven by an intuitive sense that, for an agent, there is little impact from the agents not urgently related. Instead of constantly paying attention to all teammates, agents should focus on agents who are correlated. It is important to note that limiting the information considered by agents can increase the environment's instability. To mitigate this and expedite training, we introduce a selector for each agent that learns to select K agents' information. Moreover, we measure agent correlation based on the similarity of their observations, driven by the intuition that neighboring agents are more likely to interact.

Specifically, the selector of agent i takes all agents' observations and actions $(\boldsymbol{o}, \boldsymbol{a}) = \{(o_i, a_i), (\boldsymbol{o^{-i}}, \boldsymbol{a^{-i}})\}$ as input, where $(\boldsymbol{o^{-i}}, \boldsymbol{a^{-i}}) = \{(o_j, a_j), j \in \{1, \cdots, N\}, j \neq i\}$. To learn which agents need to be coordinated with, the selector computes the relevancy weight w_{ij} for each agent j by matching their observations.

As shown in Fig.1, the weight w_{ij} compares o_i with o_j by using the computationally efficient scale dot score function to get $s_{ij}(o_j, o_i) = \frac{o_j^{\mathrm{T}} o_i}{\sqrt{d}}$, where d is the dimensionality of the observation. Then the matching values are passed into a softmax function to obtain relevancy weights:

$$w_{ij} = \frac{\exp\left(s_{ij}\left(o_j, o_i\right)\right)}{\sum_{j \neq i} \exp\left(s\left(o_k, o_i\right)\right)}. \tag{7}$$

Then, we get a normalized weight vector $W_i \triangleq (w_{i1}, \cdots, w_{ij}, \cdots, w_{iN})$, $j \neq i$, which satisfies $\sum_{j \neq i} w_{ij} \equiv 1$. According to the weight vector W_i, we select K agents in a sample way and obtain an vector $\boldsymbol{z_i} = \{(z_{i1}, \cdots, z_{ij}, \cdots, z_{iN}), z_{ij} \in [0, 1], \}$, where

$$z_{ij} = \begin{cases} 1, & \text{if agent j is selected or } j = i \\ 0, & \text{if agent j is unselected} \end{cases}. \tag{8}$$

By combining $\boldsymbol{z_i}$, the other agents' information $(\boldsymbol{o^{-i}}, \boldsymbol{a^{-i}})$ are transfered to $(\hat{\boldsymbol{o}}^{\boldsymbol{-i}}, \hat{\boldsymbol{a}}^{\boldsymbol{-i}})$, given by

$$\begin{aligned} \hat{\boldsymbol{o}}^{\boldsymbol{-i}} &= \{z_{i1} \cdot o_{i1}, \cdots, z_{ij} \cdot o_{ij} \cdots, z_{iN} \cdot o_{iN}, j \neq i\} \\ \hat{\boldsymbol{a}}^{\boldsymbol{-i}} &= \{z_{i1} \cdot a_{i1}, \cdots, z_{ij} \cdot a_{ij} \cdots, z_{iN} \cdot a_{iN}, j \neq i\} \end{aligned}. \tag{9}$$

where the filtered out agents' information is replace by a zero vector. Finally, the selector outputs $(\hat{\boldsymbol{o}}, \hat{\boldsymbol{a}}) = \{(o_i, a_i), (\hat{\boldsymbol{o}}^{\boldsymbol{-i}}, \hat{\boldsymbol{a}}^{\boldsymbol{-i}})\}$.

Take the case of 3 agents as an example, we have the two possible output configurations for agent 1:

$$\{(o_1, a_1), (o_2, a_2), (\overrightarrow{0}, \overrightarrow{0})\}, \{(o_1, a_1), (\overrightarrow{0}, \overrightarrow{0}), (o_3, a_3)\}.$$

In off-line learning, networks are updated with mini-batch samples from the replay buffer $\mathcal{D}$. Therefore, the selector selects K agents based on the average weight of the mini-batch samples. Then, it can be seen as the information of the unselected agents is dropped out in current training round, thus narrowing the critic networks and yielding faster training.

With the selector applied, the centralized critic is trained to minimize the loss function:

$$\mathcal{L}(\omega_i) = \mathbb{E}_{\boldsymbol{o}, \boldsymbol{a}, r, \boldsymbol{o'} \sim \mathcal{D}}\left[\left(Q_i^{\mu}(\hat{\boldsymbol{o}}, \hat{\boldsymbol{a}}; \omega_i) - y_i\right)^2\right], \tag{10}$$

where $\hat{\boldsymbol{o}} = (\hat{o}_1, \cdots, \hat{o}_N)$ and $\hat{\boldsymbol{a}} = (\hat{a}_1, \cdots, \hat{a}_N)$ is the observations and actions selected by the selector, respectively. And target Q-value $y_i = r_i + \gamma Q_i^{\mu'}(\hat{\boldsymbol{o}}', \hat{\boldsymbol{a}}'; \omega_i')|_{a_j' = \mu_j'(o_j'; \theta_j')}$, where ω_i' and θ_i' are the parameters of target critic network and the target policy network for agent i respectively.

The policy for agent i is updated by maximizing the objective, $J(\theta_i) = \mathbb{E}[R_i]$, and the gradient is:

$$\nabla_{\theta_i} J(\theta_i) = \mathbb{E}_{\boldsymbol{o}, \boldsymbol{a} \sim \mathcal{D}}\left[\nabla_{\theta_i} \mu_i(o_i) \nabla_{a_i}(Q_i^{\boldsymbol{\mu}}(\hat{\boldsymbol{o}}, \hat{\boldsymbol{a}}; \omega_i))|_{a_i = \mu_i(o_i; \theta_i)}\right]. \tag{11}$$

Finally, the policy for each agent is updated by two terms, one over the trajectories collected by interaction as Eq.11 and another over the recall traces generated by the backtracking model as Eq.5.

4 Experiments

In this section, we perform experiments to investigate the effectiveness of our method on two continuous control tasks of resource collection and rover exploration, which are modified from the widely used multiple-particle environment (MPE) benchmark [24].

4.1 Setup

In this section, we evaluate the proposed algorithm against several MARL methods and answer the flowing question:1) Does MASL improve sample efficiency and yield faster learning? 2) Is it still available in large state space? 3) Does it achieve more advanced results than state-of-art algorithms? We compare MASL with several state-of-the-art solutions, including MADDPG [24], DDPG [20], MADDPG-MD [16], mean field reinforcement learning(MF) [39], multiple-actor-attention-critic (MAAC) [14] and independent generative adversarial self-imitation learning (IGASIL) [11].

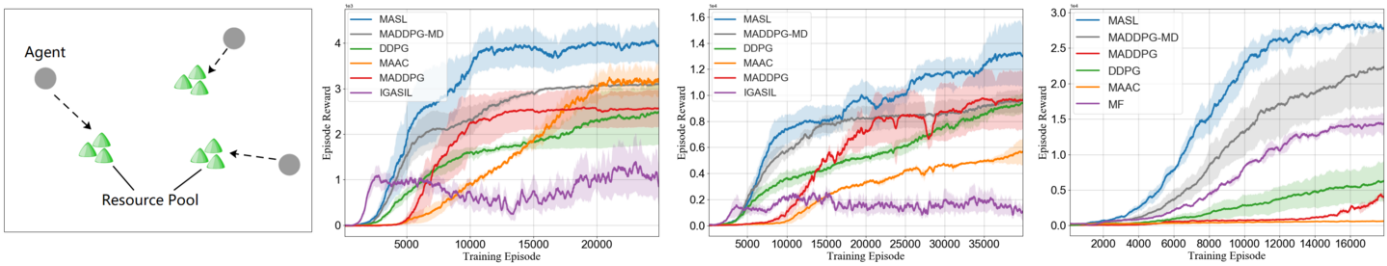
Metrics. Three metrics are used to measure the performance of an algorithm: 1) *Episode Reward*: the accumulated reward of each episode; 2) *Quantity*: the number of goals finished at the end of each episode; 3) *Distance*: the sum of the agent distance to the nearest goal at the final step of episode.

Training Details. For all scenarios, the backtracking model generates trajectories of 3 steps for training. We set K=2,3,4 for the resource collection task with N=5,8 and 10 agents respectively. For the rover exploration task with N=6, 8 and 12 agents, we set K=4, 5, 6 respectively. More details can be found in Appendix A.2.[1]

4.2 Resource Collection Tasks

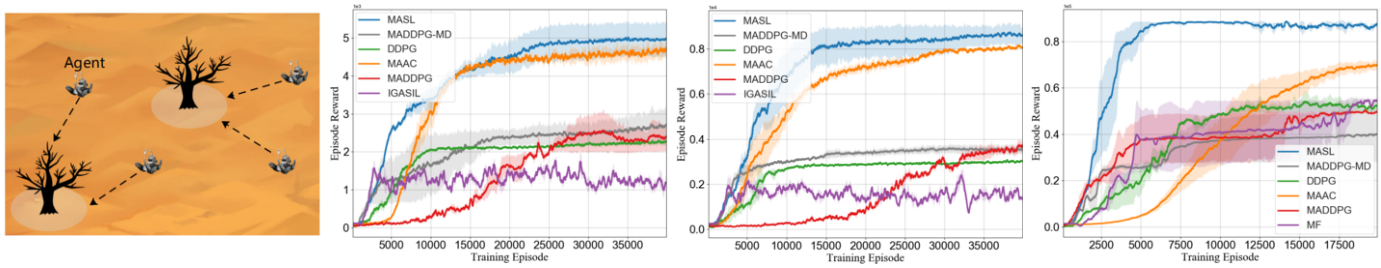
Environment Settings. The resource collection task requires agents to collect resources in multiple resource pools. There are N agents and L resource pools, as shown in Fig.3(a). The agents are generated at random locations and try to mine all the resource pools while avoiding collision. Each agent gets a positive reward related to the number of mining resource pools and a negative reward when collisions happen. In the fully shared reward setting, each agent shares a global reward at each time step, equal to the sum of each agent's individual reward. To evaluate the learning efficiency of our method, we first observe whether MASL learns faster compared to the baselines described above. Then we compare the final achievement to answer the second question. We evaluate the performance of MASL in the cases of $(N = 5, L = 5)$, $(N = 8, L = 8)$ and $(N = 10, L = 12)$, where we add the mean field reinforcement learning (MF) algorithm as a baseline in the more general case of (N=10, L=12).

[1] Appendix is available in https://github.com/CCConcerning/MASL.



(a) Resource collection (RC) (b) RC-5: $N = 5, L = 5$ (c) RC-8: $N = 8, L = 8$ (d) RC-10: $N = 10, L = 12$

Figure 3. Resource collection tasks: (a) Task description; (b-d) Episode rewards in the case of $(N = 5, L = 5)$, $(N = 8, L = 8)$ and $(N = 10, L = 12)$.



(a) Rover exploration (RE) (b) RE-6: $N = 6, L = 3$ (c) RE-8: $N = 8, L = 4$ (d) RE-12: $N = 12, L = 3$

Figure 4. Rover exploration tasks: (a) Task description; (b-d) Episode rewards in the case of $(N = 6, L = 3)$, $(N = 8, L = 4)$ and $(N = 12, L = 3)$.

Results. As shown in Fig. 3, where we plot the episode reward of different algorithms as the training progresses, MASL outperforms the other methods in all the cases. It is clear that MASL learns more rapidly and reaches higher episode rewards earlier than the baselines. In contrast, the other methods converge to suboptimal performance due to limited exploration. Moreover, it is noteworthy that as the number of agents increases, MASL maintains its superior performance in terms of learning speed and final reward attainment.

In the case of 5 agents, DDPG performs slightly worse than MADDPG. However, when the number of agents increases to 8, DDPG achieves similar performance as MADDPG. This result suggests that considering all other agents from the beginning might hinder the learning process, especially with a larger number of agents. The comparison between MASL and MAAC reveals that MASL's selection mechanism outperforms the soft attention method used in MAAC. The use of multiple attention heads in MAAC increases network complexity and requires significant time to learn attention distributions. In contrast, MASL, which considers K key agents based on raw observations, proves to be more effective. IGASIL exhibits poor performance as it fails to identify valuable experiences from shared rewards. Moreover, IGASIL incurs a training and running time that is more than triple that of MASL, due to its use of a discriminator for each agent. In contrast, MASL identifies high-value trajectories from the shared rewards and selectively shares experiences, thereby improving sample efficiency.

To test the final performance during execution, we evaluate the algorithms in the *Distance* and *Quantity* metrics. As shown in Table 1, MASL mines more resource pools and causes a smaller agent distance to each resource pool.

Table 1. Resource collection: The number of resource pools being mined and the sum of agents distance to the resource pools.

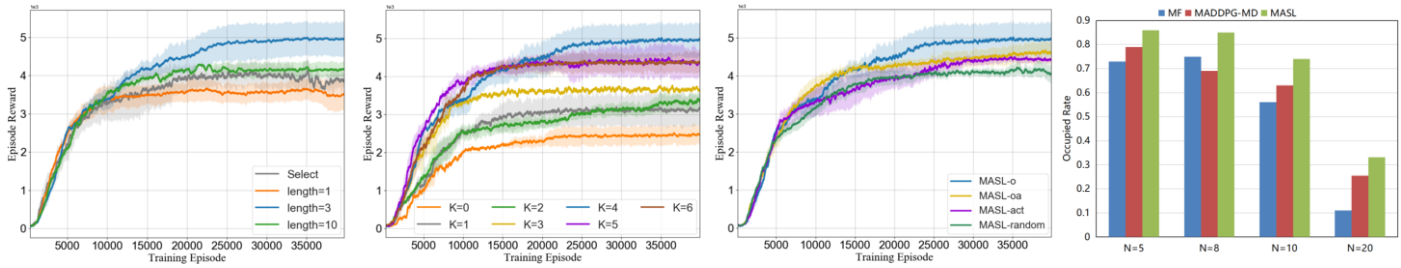
	Quantity		Distance	
	N=5	N=8	N=5	N=8
MASL	**4.32**	**6.81**	**0.46**	**0.91**
MADDPG-D	3.99	5.51	0.70	1.72
DDPG	3.52	5.48	1.17	1.85
MADDPG	3.66	5.50	1.17	1.60
MAAC	4.11	4.99	0.54	2.20
IGASIL	3.10	1.88	3.18	2.80

4.3 *Rover Exploration Tasks*

Environment Setting. The rover exploration task requires a team of agents to coordinately explore an area simultaneously, as shown in Fig.4(a). N agents learn to cooperatively explore different L target areas separately. The number of agents required to explore a target area is termed the coupling requirement, which is unknown for agents. Agents receive a shared reward related to the number of the explored areas. The rover exploration tasks bring more challenges to agents due to the requirement of deep cooperation. We evaluate MASL in the cases of $(N = 6, L = 3)$, $(N = 8, L = 4)$ and $(N = 12, L = 3)$.

Results. We first summarize the results of the $(N = 6, L = 3)$ and $(N = 8, L = 4)$ rover exploration scenarios, where the coupling requirement is 2. As shown in Fig. 4(b), the reward curve of MASL grows significantly faster than the other methods. MASL agents learn quickly to form a team of two to explore an area in RE-6 scenarios. Similar results can be found in RE-8 scenario $(N = 8, L = 4)$. As the number of agents increases, the size of joint state space increases rapidly, which slows down the training process. However, MASL still maintains an advanced performance (see Fig. 4(c)). We observe in the experiments that the independent DDPG agents quickly learn to reach the nearest target, while MADDPG agents learn to explore different areas but often fail to form a team. In contrast, once an agent reaches a target occasionally, MASL quickly learns to form teams to explore different targets.

To evaluate the performance of MASL in long-horizon tasks, we conduct experiment on the 100-step rover exploration task with 12 agents and 3 areas $(N = 12, L = 3)$, which require agents to explore 3 areas with a high coupling requirement of 4. As shown in Fig.4(d), MASL still achieves higher sample efficiency in the long-term games compared with the baselines of MADDPG-MD and MADDPG.



(a) (b) (c) (d)

Figure 5. Ablation results(a)-(c): (a) Results of setting different backtracking sizes. (b) Results of selecting different number of agents. (c) The results of different selection mechanisms. Scalability results (d): The target completion rates in the resource collection tasks with different number of agents.

Table 2. Rover Exploration: the number of fully explored targets and the sum of agents distance to target areas.

	Quantity		Distance	
	N=6	N=8	N=6	N=8
MASL	**2.14**	**3.05**	**2.61**	**3.13**
MADDPG-D	1.19	1.12	7.77	14.25
DDPG	1.11	1.0	10.21	18.79
MADDPG	1.23	1.34	8.10	6.83
MAAC	1.93	2.60	3.21	4.11
IGASIL	0.54	0.36	2.26	3.73

Table 2 demonstrates the number of fully explored targets by teams of two agents and the sum of agent distance to target areas in the tasks with coupling requirement of two. The superiority of our method shows that the proposed selective learning method improves sample efficiency and promotes effective exploration, while becomes more significant as the number of agents increases.

4.4 Ablation Study

We perform ablation experiments on a 6-agent rover exploration task(RE-6:N=6, L=3) to study the effectiveness of key components of MASL: 1) the backtracking model that generates recall traces and 2) the selector that selects the information of the other K agents.

Ablation on the backtracking model. We study the importance of learning from recall traces by comparing against MASL with backtracking removed, which refers to *Select*. To demonstrate the sensitivity of our algorithm to the parameter of backtracking length, we set the length to 1, 3 and 10 respectively. Fig. 5(a) shows the performance in the training phase, the x-axis is the training episode, and the y-axis is the accumulated reward of each episode. It's clearly shown that MASL strictly dominates *Select* in the final performance, which indicates that learning from recall traces yields deeper exploration. It is seen that different backtracking lengths may result in rather different results. A long backtracking length (10 steps) leads to insufficient exploration, while a short length (1 step) cannot provide enough guidance. MASL with backtracking 3 steps provides appropriate guidance for the rover exploration tasks.

Ablation on the hyperparameter K of the selector. We study the effectiveness of the selector by changing the number of selected agents K. The hyperparameter K can be seen as a trade-off factor between training speed and stability. Note that from the perspective of agents, the larger the K, the more stable the environment. However, the input space also grows quickly with the increase of K, leading to slow policy learning and sub-optimal performance. The case of K=0 equals to a variant of MASL called DDPG+BM, which applies the backtracking model to DDPG. The case of $K = 6$ equals to a MASL variant that removing the selector. As shown in the Fig.5(b), the setting of K = 4 is the best in this environment, which speeds up training while ensuring stationary to a certain extent. It suggests that the trade-off between efficient training and stability can be found by selectively considering the information of other agents.

Ablation on the similarity metrics of the selector. We further study the effectiveness of similarity metric based on observations. The key information selection mechanism selects the other related agents based on their observations. However, the measurement of the relevancy between agents is an open question. Therefore, since the observations and actions is shared between agents, we add two variants: 1) MASL-oa which select key information based on both observations and actions as the criterion; 2) MASL-act which measures relevance based on actions; 3) MASL-random that selects K agents randomly. The original version of MASL is denoted as MASL-o. We conduct experiments in the rover exploration task of 6 agents to verify the effectiveness of using the matching degree of observations as a correlation measurement between agents. The experimental result is shown in the Fig. 5(c), all of the selection mechanisms based on relevance outperform the random selection mechanism (MASL-random), which indicates that it is useful to focus on related agents. Compared with MASL-oa and MASL-act, MASL-o has better performance. Although we allow the agents to access all observations and actions during the training process, observations contain more information than actions and thus become a better measure of relevance. Actions may act as noise in the selection process, which affects the accuracy of related agents' information selection when MASL-oa agents learning.

4.5 Discussions

In this section, we study the scalability and applicability of MASL to best show its advantage in sparse reward tasks.

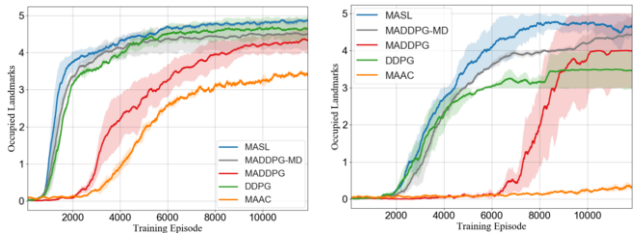
Scalability study. To evaluate the scalability of MASL, we train agents in the resource collection scenarios with $N = [5, 8, 10, 20]$ agents respectively, and then report the final target completion rates of MASL compared with MADDPG-MD and mean field reinforcement learning (MF) [39], which are designed to address large-scale agents. The MF algorithm is added as a baseline method, which approximates the interaction within agents using an average effect to enable coordination between large-scale agents. As shown in the Fig. 5(d), MASL achieves higher target completion rates than the other baselines, even in the large-scale scenario with 20 agents. We observe that MASL only sacrificed 0.93% of the computation time (see Appendix A.4 for the detailed computational complexity analysis) to improve the sample efficiency by 8.8% compared with MADDPG-MD when $N = 5$. The sample efficiency of MASL is improved

by 30.2% in the case of $N = 20$. We also study the scalability of the trained MASL agents by transferring the trained policies of 5 agents to 30 agents, and testing MASL in the environment of 30 agents ($N = 30, L = 30$). We observed that the trained decentralized policies of MASL in easy tasks can be directly scale to complex tasks with large-scale agents, while achieving a higher target completion rate of 36%, surpassing MADDPG-MD (24.7%). MASL improves the sample efficiency for sparse reward tasks, especially in large-scale scenarios.

Applicability study. Since we propose MASL in the context of tasks with shared reward, we further study the applicability of MASL by considering tasks with reward forms of individual rewards and partially shared rewards. We change the reward function of the resource collection environment to use individual reward and partially shared reward respectively, where each agent gets a positive reward when it mines a resource pool in the individual reward setting. For the partially shared reward setting, agents obtain a reward related to the number of resource pools mined by the whole team and are punished when it collides with other agents. As shown in Fig. 6, MASL achieves faster learning efficiency than the baselines in both cases. We observe that the algorithms perform better in both cases compared to the fully shared reward setting. It suggests that different reward functions greatly affect the performance of the algorithms, while the shared sparse reward brings difficulties in credit assignment. When agents access individual rewards, the contribution of agents can be differentiated directly based on the rewards. In this case, MASL accelerates learning by utilizing the backtracking model and selectors without the need of performing the retrogression-based selection process. In the case of partially shared reward, the learning process of agents is similar to that of shared reward case, since the individual contributions cannot be clearly distinguished. MASL generally improves sample efficiency, especially in tasks with shared rewards. Moreover, MASL has broad applicability to sparse reward tasks beyond the shared reward settings.

5 Related Work

Sparse reward in reinforcement learning. To deal with the sparse reward challenge, an intuitive method is reward shaping [26], but it requires domain knowledge and lacks design generality and expandability. Return decomposition paradigm is proposed recently to redistribute sparse environmental feedback [5]. RRD [32] learns long-term reward redistribution via randomized return decomposition. Intrinsic motivation based methods have been widely studied to aid in exploration by adding intrinsic rewards [1, 29, 38]. Recent works have benefit from leveraging past experiences to promote sample efficiency [33, 28, 4, 10]. Self-imitation learning (SIL) [28] imitates the agent's own past good experiences. Some works [4, 10] generate traces from past high-value states by a learned backtracking model. GPRIL [34] reasons demonstrated states from expert demonstrations by using generative models. However, these methods usually cannot be directly applied to multi-agent environments with shared rewards, since they rely on the identifiability of high-value samples (high-reward transitions or expert demonstrations). There are some works making tentative exploration to solve sparse reward in multi-agent tasks. MAGAIL [36] extends GAIL [12] to multi-agent cases by using expert demonstrations. MERL [25] introduces evolutionary algorithm to maximize the sparse team-based objective under the assumption that dense agent-specific rewards are accessible. IGASIL[11] combines SIL with GAIL to guide agents to explore around high-reward regions. CMAE [22] learns coordinated exploration in restricted state spaces, which only works well on tasks with discrete state space. MAGIC [2] drives efficient exploration by learning consistent goal recognition.



(a) Individual reward (b) Partially shared reward

Figure 6. The number of resource pools in resource collection tasks with individual reward (a) and partially shared reward (b).

Multi-agent reinforcement learning. Policy learning in cooperative multi-agent tasks suffers from the non-stationary environment. To deal with the non-stationarity problem, MADDPG [24] extends DDPG [20] to multi-agent setting and proposes to use a centralized critic. Since MADDPG gets all agents' information as input, its training efficiency and scalability are limited by the dimension of state and action spaces. To improve the efficiency of centralized training, MADDPG-MD [16] introduces the dropout technique to drops out the received messages from other agents with a certain probability. Unlike MADDPG-MD that chooses agents randomly, our work selects agents based on relevancy by an effective selector. Mean Field method [39] reduces the input dimension by approximating the interactions within the population of agents with an average effect. MAAC [14] aggregates all weighted agent information by using attention mechanism. SePS [3] recognizes heterogeneous agents and shares parameters among them for improving computational efficiency. Some work has developed an attractive way to exploit centralized learning by learning a joint state-action value function and address the credit assignment problem to some extent [7, 9, 31, 35]. The proposed retrogression-based selection method addresses the credit assignment problem of sparse team reward tasks by interacting with the environment and reasoning from the changes in rewards.

None of the existing works explicitly solve the sparse reward problem and the scalability problem of centralized training. In stark contrast, MASL fully considers the cooperation among agents and improves sample efficiency by sharing recall traces. Moreover, MASL yields efficient learning of large input dimensions by selecting key information.

6 Conclusions

In this paper, we propose an efficient training method called multi-agent selective learning (MASL), to improve sample efficiency for multi-agent sparse-reward tasks. By using a centralized backtracking model, MASL learns not only from traces obtained by interacting with the environment but also from recall traces generated by the backtracking model. Moreover, we design a selector to improve learning efficiency while balancing stability. Experiments show that MASL significantly speeds up learning in several multi-agent sparse-reward tasks, especially in tasks with large-scale agents. Ablation studies show that both learning from recall traces and focusing on relevant agents jointly accelerate training and aid in exploration under sparse rewards.

Acknowledgements

This work was supported by the National Key Research and Development Program of China (2022YFC3400404), the National Science Foundation of China (62172154), the Hunan Provincial Natural Science Foundation of China under grant No. 2023JJ30702. Prof. Xuan Liu is the corresponding author of the paper.

References

[1] Marc G. Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Rémi Munos, 'Unifying count-based exploration and intrinsic motivation', in *Proc. Adv. Neural Inf. Process. Syst.*, pp. 1471–1479, (2016).

[2] Xinning Chen, Xuan Liu, Shigeng Zhang, Bo Ding, and Kenli Li, 'Goal consistency: An effective multi-agent cooperative method for multi-stage tasks', in *Proceedings of International Joint Conference on Artificial Intelligence*, pp. 172–178.

[3] Filippos Christianos, Georgios Papoudakis, Arrasy Rahman, and Stefano V. Albrecht, 'Scaling multi-agent reinforcement learning with selective parameter sharing', in *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139, pp. 1989–1998, (2021).

[4] Ashley D. Edwards, Laura Downs, and James C. Davidson, 'Forward-backward reinforcement learning', *CoRR*, **abs/1803.10227**, (2018).

[5] Yonathan Efroni, Nadav Merlis, and Shie Mannor, 'Reinforcement learning with trajectory feedback', in *Proc. AAAI Conf. Artif. Intell.*, pp. 7288–7295, (2021).

[6] David Silver et al., 'Mastering the game of go with deep neural networks and tree search', *Nat.*, **529**(7587), 484–489, (2016).

[7] Peter Sunehag et al., 'Value-decomposition networks for cooperative multi-agent learning based on team reward', in *Proc. AAMAS*, pp. 2085–2087, (2018).

[8] Volodymyr Mnih et al., 'Human-level control through deep reinforcement learning', *Nat.*, **518**(7540), 529–533, (2015).

[9] Jakob N. Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson, 'Counterfactual multi-agent policy gradients', in *Proc. AAAI Conf. Artif. Intell.*, pp. 2974–2982, (2018).

[10] Anirudh Goyal, Philemon Brakel, William Fedus, Soumye Singhal, Timothy P. Lillicrap, Sergey Levine, Hugo Larochelle, and Yoshua Bengio, 'Recall traces: Backtracking models for efficient reinforcement learning', in *Proc. ICLR*, (2019).

[11] Xiaotian Hao, Weixun Wang, Jianye Hao, and Yaodong Yang, 'Independent generative adversarial self-imitation learning in cooperative multiagent systems', in *Proc. AAMAS*, pp. 1315–1323, (2019).

[12] Jonathan Ho and Stefano Ermon, 'Generative adversarial imitation learning', in *Proc. Adv. Neural Inf. Process. Syst.*, pp. 4565–4573, (2016).

[13] Ionel-Alexandru Hosu and Traian Rebedea, 'Playing atari games with deep reinforcement learning and human checkpoint replay', *CoRR*, **abs/1607.05077**, (2016).

[14] Shariq Iqbal and Fei Sha, 'Actor-attention-critic for multi-agent reinforcement learning', in *Proc. Int. Conf. Mach. Learn.*, (2019).

[15] Jiechuan Jiang and Zongqing Lu, 'Learning attentional communication for multi-agent cooperation', in *Proceedings of Conference on Neural Information Processing Systems*, pp. 7265–7275, (2018).

[16] Woojun Kim, Myungsik Cho, and Youngchul Sung, 'Message-dropout: An efficient training method for multi-agent deep reinforcement learning', in *Proc. AAAI Conf. Artif. Intell.*, pp. 6079–6086, (2019).

[17] Daniel Kubus, Rania Rayyes, and Jochen J. Steil, 'Learning forward and inverse kinematics maps efficiently', in *International Conference on Intelligent Robots and Systems*, pp. 5133–5140. IEEE, (2018).

[18] Haoran Li, Qichao Zhang, and Dongbin Zhao, 'Deep reinforcement learning-based automatic exploration for navigation in unknown environment', *IEEE Trans. Neural Networks Learn. Syst.*, **31**(6), 2064–2076, (2020).

[19] Shihui Li, Yi Wu, Xinyue Cui, Honghua Dong, Fei Fang, and Stuart J. Russell, 'Robust multi-agent reinforcement learning via minimax deep deterministic policy gradient', in *Proc. AAAI Conf. Artif. Intell.*, pp. 4213–4220, (2019).

[20] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra, 'Continuous control with deep reinforcement learning', in *Proc. ICLR*, (2016).

[21] Michael L. Littman, 'Markov games as a framework for multi-agent reinforcement learning', in *Machine Learning, Proceedings of the Eleventh International Conference*, pp. 157–163, (1994).

[22] Iou-Jen Liu, Unnat Jain, Raymond A. Yeh, and Alexander G. Schwing, 'Cooperative exploration for multi-agent deep reinforcement learning', in *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139, pp. 6826–6836, (2021).

[23] Yong Liu, Weixun Wang, Yujing Hu, Jianye Hao, Xingguo Chen, and Yang Gao, 'Multi-agent game abstraction via graph attention neural network', in *Proc. AAAI Conf. Artif. Intell.*, pp. 7211–7218, (2020).

[24] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch, 'Multi-agent actor-critic for mixed cooperative-competitive environments', in *Proc. Adv. Neural Inf. Process. Syst.*, pp. 6379–6390, (2017).

[25] Somdeb Majumdar, Shauharda Khadka, Santiago Miret, Stephen McAleer, and Kagan Tumer, 'Evolutionary reinforcement learning for sample-efficient multiagent coordination', in *Proc. Int. Conf. Mach. Learn.*, volume 119, pp. 6651–6660, (2020).

[26] Andrew Y. Ng, Daishi Harada, and Stuart J. Russell, 'Policy invariance under reward transformations: Theory and application to reward shaping', in *Proc. Int. Conf. Mach. Learn.*, pp. 278–287, (1999).

[27] Thanh Thi Nguyen, Ngoc Duy Nguyen, and Saeid Nahavandi, 'Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications', *IEEE Trans. Cybern.*, **50**(9), 3826–3839, (2020).

[28] Junhyuk Oh, Yijie Guo, Satinder Singh, and Honglak Lee, 'Self-imitation learning', in *Proc. Int. Conf. Mach. Learn.*, pp. 3875–3884, (2018).

[29] Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell, 'Curiosity-driven exploration by self-supervised prediction', in *Proc. Int. Conf. Mach. Learn.*, pp. 2778–2787.

[30] Adolfo Perrusquía, Wen Yu, and Xiaoou Li, 'Multi-agent reinforcement learning for redundant robot control in task-space', *Int. J. Mach. Learn. Cybern.*, **12**(1), 231–241, (2021).

[31] Tabish Rashid, Mikayel Samvelyan, Christian Schröder de Witt, Gregory Farquhar, Jakob N. Foerster, and Shimon Whiteson, 'QMIX: monotonic value function factorisation for deep multi-agent reinforcement learning', in *Proc. Int. Conf. Mach. Learn.*, pp. 4292–4301, (2018).

[32] Zhizhou Ren, Ruihan Guo, Yuan Zhou, and Jian Peng, 'Learning long-term reward redistribution via randomized return decomposition', in *Proc. ICLR*, (2022).

[33] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver, 'Prioritized experience replay', in *Proc. ICLR*, (2016).

[34] Yannick Schroecker, Mel Vecerík, and Jonathan Scholz, 'Generative predecessor models for sample-efficient imitation learning', in *Proc. ICLR*, (2019).

[35] Kyunghwan Son, Daewoo Kim, Wan Ju Kang, David Hostallero, and Yung Yi, 'QTRAN: learning to factorize with transformation for cooperative multi-agent reinforcement learning', in *Proc. Int. Conf. Mach. Learn.*, volume 97, pp. 5887–5896.

[36] Jiaming Song, Hongyu Ren, Dorsa Sadigh, and Stefano Ermon, 'Multi-agent generative adversarial imitation learning', in *Proc. Adv. Neural Inf. Process. Syst.*, pp. 7472–7483, (2018).

[37] Bradly C. Stadie, Pieter Abbeel, and Ilya Sutskever, 'Third person imitation learning', in *Proc. ICLR*, (2017).

[38] Haoran Tang, Rein Houthooft, Davis Foote, Adam Stooke, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel, '#exploration: A study of count-based exploration for deep reinforcement learning', in *Proc. Adv. Neural Inf. Process. Syst.*, pp. 2753–2762, (2017).

[39] Yaodong Yang, Rui Luo, Minne Li, Ming Zhou, Weinan Zhang, and Jun Wang, 'Mean field multi-agent reinforcement learning', in *Proc. Int. Conf. Mach. Learn.*, pp. 5567–5576, (2018).

[40] Brian D. Ziebart, Andrew L. Maas, J. Andrew Bagnell, and Anind K. Dey, 'Maximum entropy inverse reinforcement learning', in *Proc. AAAI Conf. Artif. Intell.*, pp. 1433–1438, (2008).