Deep Skill Chaining with Diversity for Multi-agent Systems*

Zaipeng Xie[1,2(✉)], Cheng Ji[2], and Yufeng Zhang[2]

[1] State Key Laboratory of Hydrology-Water Resources and Hydraulic Engineering, Hohai University, Nanjing, China
[2] College of Computer and Information, Hohai University, Nanjing, China
{zaipengxie,chengji,yufengzhang}@hhu.edu.cn

Abstract. Multi-agent reinforcement learning requires the reward signals given by the environment to guide the convergence of individual agents' policy networks. However, in a high-dimensional continuous space, the non-stationary environment may provide outdated experiences that lead to the inability to converge. The existing methods can be ineffective in achieving a satisfactory training performance due to the inherent non-stationary property of the multi-agent system. We propose a novel reinforcement learning scheme, MADSC, to generate an optimized cooperative policy. Our scheme utilizes mutual information to evaluate the intrinsic reward function that can generate a cooperative policy based on the option framework. In addition, by linking the learned skills to form a skill chain, the convergence speed of agent learning can be significantly accelerated. Hence, multi-agent systems can benefit from MADSC to achieve strategic advantages by significantly reducing the learning steps. Experiments are performed on the SMAC multi-agent tasks with varying difficulties. Experimental results demonstrate that our proposed scheme can effectively outperform the state-of-the-art methods, including IQL, QMIX, and hDQN, with a single layer of temporal abstraction.

Keywords: Reinforcement learning · Multi-agent systems · Temporal abstraction · Mutual information · Skill discovery

1 Introduction

As one of the most promising technologies to realize general AI, reinforcement learning has become one of the main focuses of multi-agent system research. Unlike the typical supervised or unsupervised learning with data set, reinforcement learning algorithms are learned via interactions. The agent continuously learns knowledge according to the rewards obtained, making it more adaptable to the environment. However, in reality, many complex problems cannot be modeled as a single agent interacting with the environment. Instead, they should be

*Supported by The Belt and Road Special Foundation of the State Key Laboratory of Hydrology-Water Resources and Hydraulic Engineering under Grant 2021490811, and the National Natural Science Foundation of China under Grant No. 61872171.

L. Fang et al. (Eds.): CICAI 2022, LNAI 13606, pp. 208–220, 2022.
https://doi.org/10.1007/978-3-031-20503-3_17

modeled as multi-agent collaboration or competition problems [1]. As a popular research area in distributed AI, a multi-agent system (MAS) can solve many complex real-time dynamic problems, such as network routing collaboration [2], trash recycling problems [3], and urban traffic control [4].

Recently, the state-action space's nonstationary property and its exponential growth have received much attention [5]. Several works have proposed multi-agent reinforcement learning (MARL) algorithms to improve policy effectiveness and convergence speed. However, these methods can be incompetent when a large number of agents learn simultaneously.

The basic paradigm of deep reinforcement learning (DRL) is a two-stage rule: *the Evaluation Stage* and *the Improvement Stage* [6]. On this premise, the agent's policy network needs to continuously improve the accuracy driven by the reward signals from the environment. However, in a collaborative MAS, we consider how each agent's policy network converges to the optimal state while the collaborative policies between different agents are still suboptimal. This is often difficult because the input to the policy networks of the agents may not be a direct global state. All the agents are simultaneously interacting with and learning from the environment. For an individual agent, its fellow agents can also be a part of the environment, leading to constant changes in the environment [7], i.e., one's best policy may change as the other agents' policies update.

We propose a method named Multi-agent Deep Skill Chaining (MADSC) that can effectively enable agents to extend their actions via temporal abstraction. Furthermore, we utilize the intrinsic reward to inspire cooperation and guide the agents' exploration trajectories. A neural network is developed based on the value decomposition method to aggregate the policy functions of each agent. Hence, our proposed algorithm can significantly improve the convergence speed and performance of cooperative policy networks in MAS.

2 Related Works

In a large-scale MAS, it can be infeasible to train each agent separately. Existing multi-agent reinforcement learning approaches [8–10] often employ a centralized training, decentralized execution(CTDE) method, where centralized Q-networks are used for training the global network to stabilize the evaluation process. Meanwhile, a decentralized execution approach can be employed, allowing each agent to use their observations as a critical basis for decision-making during execution.

Rashid et al. [11] proposed an algorithm named QMIX to solve the credit assignment problem. By using the decomposition of the value function, QMIX can improve the efficiency of cooperation. However, evaluating the utility of each agent's behavior in MAS can be difficult as all agents are performing simultaneously. Bacon et al. [12] propose that the abstraction of learning tasks can be based on the option that is essentially a sequence of actions to complete the corresponding sub-tasks in a specific state subspace. The option itself can be considered a unique action that consists of an action set, including the primitive action. Tang et al. [13] propose hDQN, a hierarchical control structure is

constructed through the invocations between the upper and lower policy layers. In addition, various optimization algorithms have been proposed to improve the convergence speed. For example, Deep Skill Network (DSN) [14] combines the hierarchical scheme with the deep Q-learning algorithm to improve the reusability of existing trained skills. Hindsight Experience Replay method [15] integrates the experience pool design with the hierarchical approach to improve the experience sampling of the layered learning efficiency.

In the options framework, skill is often modeled as an option [16]. When an agent explores a particular environment state, its observations can directly determine the actions performed in the following multiple time steps at the macro level. Sharma et al. [17] propose a skill discovery method, using the mutual information between state sequences and potential skills to construct internal rewards that encourage agents to explore a collection of skills with diversity. Bagaria et al. [16] propose deep skill chaining that can autonomously discover skills in high-dimensional continuous domains. The algorithm constructs skills that implicitly represent relationships between several related skills, allowing an agent to execute the skill chain sequentially. However, the lack of an efficient exploration method may still lead to a decline in the algorithm's convergence performance.

3 MARL with Skill Discovery

A Markov Decision Process can be formally defined as (N, S, P, R, γ). When multiple agents are involved, MDP is no longer satiable to describe the environment because the agents' actions are strongly correlated with the overall environment state. Thus we can extend the definition of MDP into Markov games [18], also called stochastic games. The definition of a Markov game [18] can be described as $(N, S, \{A^{(i)}\}_{i\in N}, P, \{R^{(i)}\}_{i\in N}, \gamma)$, where N is the total number of agents, S is the overall state distribution, $\{A^{(i)}\}_{(i\in N)}$ represents the set of all agents, and P, R, γ are state transition, reward function, and discount factor correspondingly.

3.1 How Agents Learn their Policies

There are various methods [19–21] to learn a policy in an MDP. One popular method opts to learn an action-value function $Q^{\pi}(s_t, a_t)$, i.e., the sum of the cumulative rewards that the agent may achieve according to the current policy. And the goal for each agent is set to maximize the value of this action-value function give by $\pi(s_t) = \arg\max_a Q(s_t, a_t)$, where a_t and s_t denote the action and state at time t. The dynamic migration process of MAS can be described by $s_{t+1} = f_t(s_t, \boldsymbol{a}_t)$, where f_t is the environment-dependent deterministic function, and $\boldsymbol{a}_t$ denotes the joint-action. The reward function, as the extrinsic reward R_E, can be defined as a linear sum of the rewards of all the agents given by $R_E = \sum_{s_t, \boldsymbol{a}_t \in \tau^{(i)}} R(s_t, \boldsymbol{a}_t)$, where $\tau^{(i)}$ is the state trajectory of the i-th agent.

3.2 Option Framework

The concept of the option framework is derived from the temporal abstraction technique [22] in hierarchical reinforcement learning (HRL). An option can be formally defined as $\omega \in \Omega \triangleq (\mathcal{I}_\omega, \pi_\omega, \beta_\omega)$, where Ω denotes all the available options. All options consist of three parts: the set of initial states $\mathcal{I}_\omega$, the internal policy π_ω, and the set of termination states β_ω. The agent uses the option internal policy π_ω to select actions at low-level time steps to interact with the environment. The option internal policy can map the state s to the low-level action a.

3.3 Problem Formulation

Consider a MAS with a CTDE architecture, and the option framework is employed based on HRL temporal abstraction for learning and updating the behavioral policies of each agent. From each agent's perspective, its option internal policy learns each primitive action $a_t^{(i)}$ on the low-level time scale and updates the policy parameters through a deep policy network. Our goal on the low-level time scale is to maximize the cumulative rewards of all the primitive actions. Meanwhile, on the high-level time scale, due to the use of temporal abstraction, each agent trains their option policy $\pi_\Omega^{(i)}$ to select the applicable option at the high-level temporal sequence.

We assume a greedy approach to measure the sum of rewards of all agents in a collaborative MAS and choose an option that maximizes the high-level Q-function. Our optimization goal can be formally described as:

$$\pi_\Omega^{(i)} \in \arg\max_{\pi_\omega} \mathbb{E}\left[\sum_{i=1}^{N}\sum_{t=1}^{T_{max}} R(s_t, a_t^{(i)})\right], \tag{1}$$

where T_{max} is the time budget available to the agent when an option is applied. If the time budget is reached, the control of the option is reclaimed, and the option policy π_Ω determines the next option candidate.

4 Methodology

4.1 Option Learning in MAS

All the agents can start constructing an option repository. The option repository within every agent is defined by

$$\mathcal{O} = \{\omega_k^{(i)} \mid (\mathcal{I}_{\omega_k^{(i)}}(s_t) = 1) \cap (\beta_{\omega_k^{(i)}}(s_t) = 0)\}, \tag{2}$$

where the $\mathcal{I}_{\omega_k^{(i)}}$ and $\beta_{\omega_k^{(i)}}$ denote the initiation set and termination set, $\omega_k^{(i)}$ represents the i-th agent's k-th option. We denote ω_G as the global option whose $T_{max} = 1$. As the agents explore the environment and exploit their experiences,

their option repository can be adaptively expanded. We use a simple binary classifier to train and predict the initiation set. If an agent reaches a specific target state and successfully triggers the $\beta(s_t)$ function of the option for K times, we train it with a sequence of state trajectories of length k as the input to the classifier $p(\omega_k^{(i)} \mid \tau_k^{(i)})$. During the classifier's training, its output determines how each agent may execute the option at an appropriate state trajectory.

In general, the independent Q-learning (IQL) method [23] can be adequate to fulfill the requirements of training policy networks with improved Q value prediction accuracy. However, the IQL method may not be able to promote collaborations among agents and lead to suboptimal results. Therefore, to mitigate the issue, we proposed to set up a Mixer network at the top layer of our proposed learning framework and the network employs a value decomposition algorithm similar to that in the QMIX [11] algorithm.

4.2 Skill Chaining for MARL

The option is considered a skill that can be combined to achieve improved results. Bagaria et al. [16] propose that skills chaining can produce an improved convergence performance in single-agent tasks. Since it is desired to improve training performance in MAS, we aim to build the skill chain for each option. However, establishing a skill chain can be hindered because the agent may not be able to reach the goal state due to the non-stationary problem, resulting in poor exploration efficiency and deteriorated convergence performance.

To solve the abovementioned performance bottleneck, we propose a DRL scheme, named multi-agent deep skill chaining (MADSC), to improve the convergence performance in MAS. The proposed MADSC scheme can optimize the MAS architecture based on the skill chaining technique.

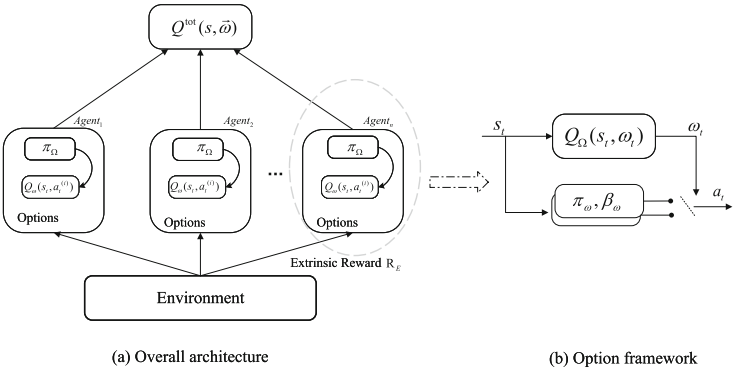

Fig. 1. Diagram of MADSC with the option framework

Figure 1 shows the overall diagram of the proposed MADSC scheme. Each agent has a complete repository of options and is independently selected and

trained by an option policy. Meanwhile, agents are coordinated through a Mixer network denoted as Q^{tot}. For each agent, its policy network π_Ω is updated through Q-learning,The update target y_t is given by

$$y_t = \sum_{t'=t}^{\tau} \gamma^{t'-t} r_{t'} + \gamma^{\tau-t} \cdot Q_\Omega(s_{t+\tau}, \arg\max_{\omega^{(i)}} Q(s_{t+\tau}, \omega)), \tag{3}$$

where τ denotes the trajectory of all time steps. Hence, the option policies of each agent is updated at each training step with the update target y_t.

The key to the skill chaining is to make the initiation set of the current option ω_i intersect with the termination set of the next option ω_{i-1}. The agents can naturally execute the next option when the option is executed to its termination set. Hence, it is desired that the learning of the initiation set $\mathcal{I}_{\omega_i}$ is continuously trained until it is accurate during the training process. For each agent, the condition $\beta_{\omega_{i+1}} = \mathcal{I}_{\omega_i}$ needs to be satisfied, so that the current option ω_i can be chained successfully to the next option. Figure 2 illustrates the diagram of the skill chaining process.

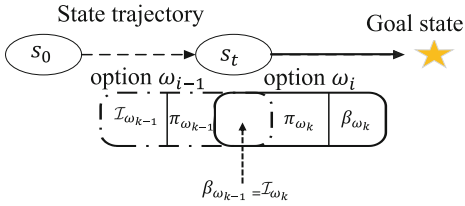

Fig. 2. Diagram of the skill chaining process

4.3 Mutual Information for Space Exploration

The mutual information is a general measure to describe the correlation between two random variables, and it is defined as the KL-diversity [24] between the joint distribution $p(s;\omega)$ and $p(s) \cdot p(\omega)$ given by

$$I(S;\Omega) = \iint p(s,\omega) \log \frac{p(s,\omega)}{p(s) \cdot p(\omega)} ds d\omega, \tag{4}$$

Maximizing mutual information of two random variables in deep learning training has been shown to perform well in many other domains [25–27]. A high mutual information between the state distribution S and the option sample distribution denotes that the uncertainty of the state distribution can be reduced when the option $\omega^{(i)}$ is fixed. Thus, we can effectively increase the diversity of the exploration with a maximized mutual information.

In our proposed MADSC method, the option's internal policy network is driven by the rewards from the environment. Meanwhile, the high-level cooperative policy is updated using a combination of intrinsic and extrinsic

rewards constructed through mutual information. We define the high-level value decomposition-based cooperative policy as π^{tot}. Therefore, our goal is to maximize the mutual information between the state trajectory distribution $p(\tau)$ and the sample distribution of option $\omega^{(i)}$. The resulting construction of the intrinsic reward function R_I for the i-th agent can be written as

$$R_I^{(i)} = I(S;\Omega) - \frac{1}{N}\sum_{i=1}^{N} I(S;\omega^{(i)}) = \log p(\Omega \mid S) - \frac{1}{N}\sum_{i=1}^{N} \log p(\omega^{(i)} \mid s_t) \quad (5)$$

The rewards function is implemented to promote efficient exploration in MDP and mitigate the performance bottleneck in the skill chain technique. Therefore, the reward that ultimately update the policy Q^{tot} is a mixture of extrinsic and intrinsic rewards given by

$$R^*(\tau,\omega) = \alpha \cdot \sum_{s_t,\boldsymbol{a}_t \in \tau^{(i)}} \gamma^t R(s_t,\boldsymbol{a}_t) + (1-\alpha)\cdot \sum_{i=1}^{N} R_I^{(i)}, \quad (6)$$

where $\sum_{s_t,\boldsymbol{a}_t \in \tau^{(i)}} \gamma^t R(s_t,\boldsymbol{a}_t)$ denotes the extrinsic rewards R_E and R_I represents the intrinsic reward, γ^t is the discount factor where lower values are placed on immediate extrinsic rewards. The parameter α is defined as a factor that may be used to regulate the contribution of the intrinsic reward. We incorporate the mixer network to process the Q-value generated by each agent so that each policy update can be coordinated and directed by the central mixer network.

Algorithm 1: The MADSC process running on the central server

Init : Initialize with *env.reset*(), set parameters T_{max} for option framework
Global option for each agent: $\omega_G = (\mathcal{I}_{\omega_G^{(i)}}, \beta_{\omega_G^{(i)}}, T_{max} = 1)$
Initialize Agents' option repository $\mathcal{O} \leftarrow \{\omega_G\}$

```
while time steps t < Max training steps N^t do
    if (t%evaluate cycle==0) then
        forall Agents do
            Evaluate current policies and record win rates
    forall Agents do
        while t < T_max do
            Sample current option ω_t^(i) with π_{ω_k^(i)}(s_t) = arg max_{a_t^(i) ∈ a_t} Q_ω^π(s_t, a_t)
            Execute option ω_t^(i) and save experiences
            Update π_{ω_k^(i)}(s_t) using Q-learning and append it to O
    Generate intrinsic reward R_I using Eq. (5)
    Update Q^tot using R_I and extrinsic rewards R_E sampled from experiences
```

Algorithm 2: The MADSC process running on the individual agent

Init : Get state s_t, observation obs_t, $\beta_{\omega^{(i)}}(s_t) \leftarrow 0$
low-level time scale $t_0 = t$
T_{max} is the option's time budget
while s_t *is not the termination state* **do**
  while $(\beta_{\omega^{(i)}} == False)$ *and* $(t < T_{max})$ **do**
    $a_t^{(i)} \leftarrow \pi_{\omega^{(i)}}(obs_t)$
    $r_t, s_{t+1} \leftarrow env.step()$
    $t \leftarrow t + 1$
    $s_t \leftarrow s_{t+1}$
  Train option initiation set $\mathcal{I}_{\omega^{(i)}}$ with binary classifier

Since our MADSC scheme is developed for the CTDE architecture, it has two major processes: the first process is for centralized training, and the second process is for decentralized execution. Algorithm 1 summarizes the process running on the centralized server, where all the information required including the observations and state trajectories are collected and fed into the policy networks. Then the central server uses information to train those networks. Algorithm 2 describes the process running on an individual agent, where each agent utilizes its observation as an input to the policy network and subsequently selects an action and executes it.

5 Experimental Evaluations

In order to evaluate the performance of our proposed MADSC algorithm, we have performed multiple experiments using the StarCraft II (SMAC) [28] as the multi-agent test environment. In the experiments, we vary the difficulty of the SMAC maps and train our agents for three million time steps.

5.1 SMAC Environment Setup

We choose various maps of StarCraft II as the evaluation MAS environment. The main purpose is to train a group of agents for a cooperative goal, i.e., defeating the AI rival. Table 1 summarizes the information of the SMAC maps.

We first evaluate the performance by varying the value of α, and the results show that when $\alpha = 0.7$, we can achieve the best result. In addition, for the Option's time budget $T_{\max}$, we set $T_{\max} = 300$ as commonly used in most state-of-the-art work on SMAC.

Table 1. SMAC maps for our experimental evaluations

Name	Ally units	Enemy units	Map difficulty
2s3z	2 Stalkers, 3 Zealots	2 Stalkers, 3 Zealots	Easy
3m	3 Marines	3 Marines	Easy
1c3s5z	1 Colos,3 Stalkers,5 Zealots	1 Colos,3 Stalkers,5 Zealots	Normal
3s5z	3 Stalkers, 5 Zealots	3 Stalkers, 5 Zealots	Normal
8m vs 9m	8 Marines	9 Marine	Normal
8m	8 Marines	8 Marines	Normal
10m vs 11m	10 Marines	11 Marines	Hard
25m	25 Marines	25 Marines	Hard
Corridor	6 Zealots	24 Zerglings	Hard

5.2 Mutual Information Evaluation

Due to the inherent property of POMDP, the experiences can be outdated when the agents interact with the environment. Decision policies based on the invalidated samples may prevent the agents from obtaining the maximum expected return in the subsequent training process.

Ideally, we expect that agents can effectively converge to a target state using a skill chain. We also desire that the skill chains be constructed backward sequentially until the s_0 state of the MDP is included in the initiation set. However, in a large MAS, the states evolve rapidly as each agent interacts with the environment. The agents may not observe the global state directly. Thus, using skill chains may also lead to the trap of a local optimal due to the ineffective exploration, resulting in poor convergence of the policy networks. We propose to incorporate the mutual information between the state trajectory distribution and the option sampling distribution and the mutual information is used to construct the intrinsic rewards of the high-level strategy network. Figure 3 demonstrates the average win rate of our proposed MADSC with and without mutual information.

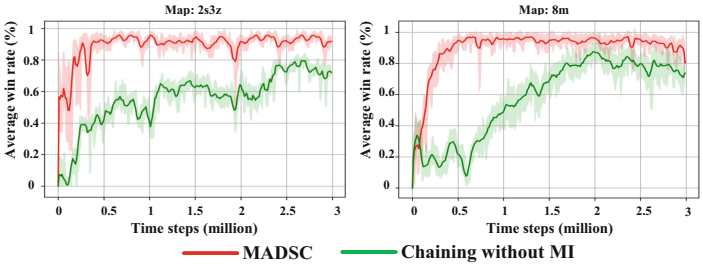

Fig. 3. Comparison of the mutual information in MADSC using intrinsic rewards

The construction of intrinsic rewards is based on the mutual information $I(S;\Omega)$ between the state distribution $p(s)$ and the ω sampling distribution $p(\omega)$. Hence, the convergence of exploration can be effectively secured against falling into the local optimum. We observe that, on the MAS tasks, including *8* m, *2s3z*, our method converges after 0.5 million time steps by utilizing the mutual information reward method. Meanwhile, the vanilla multi-agent skill chaining shows some difficulty reaching convergence even at three million steps.

5.3 Performance Evaluation

To evaluate the efficiency of our proposed MADSC scheme, We also compared the convergence performance of MADSC and some state-of-the-art MARL algorithms, including QMIX, IQL, and hDQN. The evaluation results are demonstrated in Fig 4.

A total of nine SMAC maps are utilized in the evaluation with various difficulties. The experiments are categorized into three difficulty levels, i.e., easy, normal, and hard. The main difference between each map exists in the number of agents and the difficulty variations in achieving the target state.

Our method can generally achieve the best performance among four cooperative MARL algorithms. For the easy-level maps, including *8* m, *2s3z*, because the action-state space is relatively small, both our MADSC method, QMIX, and

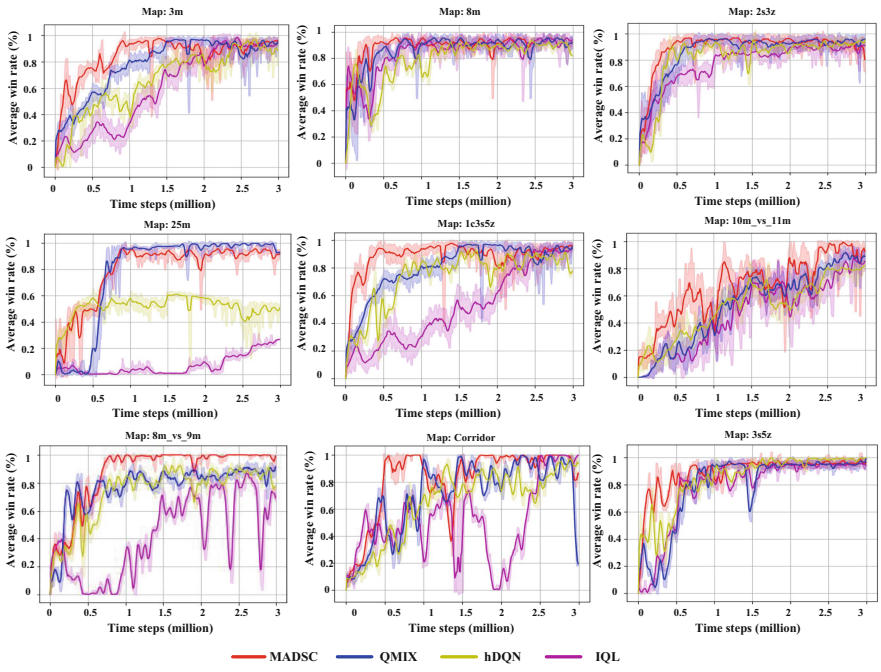

Fig. 4. Comparison of the win rates using various algorithms on SMAC tasks

hDQN can achieve an average 90% win rate after 0.5 million time steps. However, IQL does not perform well for map *2s3z*. For hard-level maps, e.g., *25* m, *10m_vs_11m*, *corridor*, our MADSC method can still perform well and converge to an average 90% win rate after 2.5 million time steps. The QMIX algorithm demonstrates a similar performance only on the *25m* task, but a deteriorated convergence performance on the *10m_vs_11m* and *corridor* tasks. However, due to the expanding size of the action-state space and the unpredictable transition function of the state trajectories, the hDQN and IQL cannot converge within the three million training time steps. For normal-level maps, such as *1c3s5z* and *8m_vs_9m*, our proposed MADSC method outperforms all three algorithms. This is because our MADSC scheme allows the agents to diversify their choices of the state trajectories by using the mutual information-based intrinsic rewards, leading to efficient space exploration. In addition, our quest for high cumulative reward expectations aims to find optimal strategies based on accurate prediction performance. The exploration efficiency can prevent the agent's policy from falling into a local optimal and may eventually allow the policy network to converge to high accuracy with a greater probability.

6 Conclusions

This paper proposes a novel MARL method, MADSC, for cooperative multi-agents in macro-action level POMDPs to mitigate the nonstationary problem and improve the convergence performance. Our approach introduces the temporal abstraction layer on MAS by building on HRL with the deep skill chaining method. The proposed MADSC method incorporates the mutual information to construct the intrinsic rewards that can prevent the policy networks from converging due to non-stationary. We evaluate our proposed method using various challenge difficulties in the SMAC multi-agent tasks. Experimental results demonstrate that our approach can effectively outperform some state-of-the-art methods with skill chaining.

References

1. Gronauer, S., Diepold, K.: Multi-agent deep reinforcement learning: a survey. Artif. Intell. Rev. **55**(2), 895–943 (2022)
2. Kang, Y., Wang, X., et al.: Q-adaptive: a multi-agent reinforcement learning based routing on dragonfly network. In: Proceedings of the 30th International Symposium on High-Performance Parallel and Distributed Computing, pp. 189–200 (2021)
3. Canese, L., Cardarilli, G.C., Di Nunzio, L., et al.: Multi-agent reinforcement learning: a review of challenges and applications. Appl. Sci. **11**(11), 4948 (2021)
4. Ma, J., Wu, F.: Feudal multi-agent deep reinforcement learning for traffic signal control. In: Proceedings of International Conference on Autonomous Agents and MultiAgent Systems, pp. 816–824. AAMAS (2020)
5. Su, J., Adams, S.C., et al.: Value-decomposition multi-agent actor-critics. In: Proceedings of the AAAI Conference on Artificial Intelligence, pp. 11352–11360 (2021)

6. Sarafian, E., Tamar, A., Kraus, S.: Constrained policy improvement for efficient reinforcement learning. In: International Joint Conference on Artificial Intelligence, IJCAI, pp. 2863–2871 (2020)
7. Terry, J.K., Black, B., Grammel, N., et al.: Pettingzoo: gym for multi-agent reinforcement learning. In: Advances in Neural Information Processing Systems, NeurIPS, pp. 15032–15043 (2021)
8. Liu, Y., Hu, Y., Gao, Y., et al.: Value function transfer for deep multi-agent reinforcement learning based on N-step returns. In: Proceedings of the 28th International Joint Conference on Artificial Intelligence, IJCAI, pp. 457–463 (2019)
9. Phan, T., Belzner, L., et al.: Resilient multi-agent reinforcement learning with adversarial value decomposition. In: Proceedings of the AAAI Conference on Artificial Intelligence, pp. 11308–11316 (2021)
10. Danassis, P., Wiedemair, F., et al.: Improving multi-agent coordination by learning to estimate contention. In: Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI, pp. 125–131 (2021)
11. Rashid, T., Samvelyan, M., et al.: QMIX: monotonic value function factorisation for deep multi-agent reinforcement learning. In: Proceedings of the International Conference on Machine Learning, ICML, vol. 80, pp. 4292–4301 (2018)
12. Bacon, P., Harb, J., Precup, D.: The option-critic architecture. In: Proceedings of the AAAI Conference on Artificial Intelligence, pp. 1726–1734 (2017)
13. Tang, H., Hao, J., Lv, T., et al.: Hierarchical deep multiagent reinforcement learning. CoRR abs/1809.09332 (2018)
14. Tessler, C., Givony, S., Zahavy, T., et al.: A deep hierarchical approach to lifelong learning in MineCraft. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 31 (2017)
15. Andrychowicz, M., Crow, D., Ray, A., et al.: Hindsight experience replay. In: Advances in Neural Information Processing Systems, NeurIPS, pp. 5048–5058 (2017)
16. Bagaria, A., Konidaris, G.: Option discovery using deep skill chaining. In: International Conference on Learning Representations, ICLR (2020)
17. Sharma, A., Gu, S., Levine, S., et al.: Dynamics-aware unsupervised discovery of skills. In: International Conference on Learning Representations, ICLR (2019)
18. Sayin, M., Zhang, K., Leslie, D., et al.: Decentralized Q-learning in zero-sum markov games. In: Advances in Neural Information Processing Systems, NeurIPS, vol. 34, pp. 18320–18334 (2021)
19. Engstrom, L., Ilyas, A., Santurkar, S., et al.: Implementation matters in deep RL: a case study on PPO and TRPO. In: International Conference on Learning Representations, ICLR (2020)
20. Osband, I., Blundell, C., Pritzel, A., et al.: Deep exploration via bootstrapped DQN. In: Advances in Neural Information Processing Systems, NeurIPS, pp. 4026–4034 (2016)
21. Mnih, V., Badia, A.P., Mirza, M., Graves, A., et al.: Asynchronous methods for deep reinforcement learning. In: Proceedings of the International Conference on Machine Learning, ICML, vol. 48, pp. 1928–1937 (2016)
22. Kulkarni, T.D., Narasimhan, K., et al.: Hierarchical deep reinforcement learning: integrating temporal abstraction and intrinsic motivation. In: Advances in Neural Information Processing Systems, NeurIPS, pp. 3675–3683 (2016)
23. Mnih, V., Kavukcuoglu, K., Silver, D., et al.: Human-level control through deep reinforcement learning. Nature **518**(7540), 529–533 (2015)

24. Kim, J., Park, S., Kim, G.: Unsupervised skill discovery with bottleneck option learning. In: Proceedings of the International Conference on Machine Learning, ICML, vol. 139, pp. 5572–5582 (2021)
25. Lin, Y., Gou, Y., Liu, Z., et al.: COMPLETER: incomplete multi-view clustering via contrastive prediction. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR, pp. 11174–11183 (2021)
26. Hjelm, R.D., Fedorov, A., Lavoie-Marchildon, S., et al.: Learning deep representations by mutual information estimation and maximization. In: International Conference on Learning Representations, ICLR (2019)
27. Bachman, P., Hjelm, R.D., Buchwalter, W.: Learning representations by maximizing mutual information across views. In: Advances in Neural Information Processing Systems, NeurIPS, pp. 15509–15519 (2019)
28. Samvelyan, M., Rashid, T., de Witt, C.S., et al.: The StarCraft Multi-Agent Challenge. CoRR abs/1902.04043 (2019)