2023 4th International Symposium on Computer Engineering and Intelligent Communications (ISCEIC) | 979-8-3503-0632-3/23/$31.00 ©2023 IEEE | DOI: 10.1109/ISCEIC59030.2023.10271221

Multi-robot Cooperative Navigation Method based on Multi-agent Reinforcement Learning in Sparse Reward Tasks

Kai Li
Marine Design & Research Institute of China
ShangHai, China
likaistar@126.com

Quanhu Wang
Shanghai Zhongchuan STD-NERC Co., Ltd.
ShangHai, China
wanggh521@sina.com

Mengyao Gong
Marine Design & Research Institute of China
ShangHai, China
782613935@qq.com

Jiahui Li
School of Computer Science
Northwestern Polytechnical University
Xi'an, China
2020262827@mail.nwpu.edu.cn

*Haobin Shi
School of Computer Science
Northwestern Polytechnical University
Xi'an, China
* Corresponding author: shihaobin@nwpu.edu.cn

***Abstract*—Multi-robot systems can collaborate to accomplish more complex tasks than a single robot. Cooperative navigation is the basis for multi-robot to complete rescue, reconnaissance, and other tasks in high-risk areas instead of human beings. Multi-agent reinforcement learning (MARL) is the most effective method to control multi-robot cooperation, but the sparsity of rewards limits its application in real scenarios. In this paper, a curiosity-inspired MARL approach which is called CIMADDPG is proposed to promote robot exploration. The global curiosity allocation mechanism is designed to determine each agent's contribution to the global reward. In addition, to ensure that the collaboration of agents is not lost during exploration, the dual critic network is designed to guide the update of the policy network jointly. Finally, the performance of the proposed method is verified in a multi-agent particle environment (MPE) and multi-robot (Turtlebot3) cooperative navigation simulation environment. The experimental results show that CIMADDPG improves the performance of SOTA by 23.53% ~ 48.84% and achieves a high success rate in multi-robot cooperative navigation.**

Index Terms*—*deep reinforcement learning; multi-robot; collaborative navigation; multi-agent reinforcement learning

I. Introduction

The early research of reinforcement learning (RL) focused on the field of single-agent. Still, there are also many multi-agent cooperative decision-making tasks in real life, such as recommendation system, traffic light control, automatic driving, etc. Researchers have gradually extended their focus from single-agent to multi-agent. Compared with single-agent tasks, multi-agent tasks face many new challenges: higher dimensional state and action space, non-stationary environment, and partial observation. To address these challenges, a framework called Centralized Training with Decentralized Execution (CTDE) has been proposed. It has become the mainstream framework of MARL. In CTDE, the critic learns a fully centralized value function based on joint states and actions and then uses it to guide the optimization of decentralized policies.

The CTDE-based MARL approach performs well on many tasks but poorly on complex scenarios with sparse reward settings. In the sparse reward scenarios, most actions of the agent cannot receive effective environmental feedback, resulting in slow policy learning or even failure to learn effective policies. These tasks require agents to explore the environment more efficiently, and most of the existing MARL algorithms based on CTDE adopt a simple exploration strategy based on ε-greedy or noise, which is challenging to meet the demand. If effective exploration strategies can be added to these existing MARL algorithms, they can be applied to complex scenarios with sparse reward settings.

This paper proposes a curiosity-inspired MARL approach to promote robot exploration. This method helps improve the application of such methods in sparse reward scenarios, such as multi-robot cooperative navigation. The structure of this paper is organized as follows: Sec. II presents the related work of this work. Sec. III explains our proposed MARL method in detail, including curiosity-inspired exploration, network architecture, and training process. Detailed experimental validation and application of multi-robot collaborative navigation are analyzed in Sec IV. Sec V concludes the paper and points out further work.

II. Related Work

The vigorous development of reinforcement learning in the multi-agent field cannot be separated from the theoretical basis of single-agent RL. Still, MARL is more complex than single-agent RL. Because in MARL, it is necessary to consider the interaction between the agent and the environment and the interaction between the agents.

One of the simplest MARL algorithms is Independent Q-Learning (IQL) [1] , which applies the DQN algorithm to every agent in the environment. Because the agents in the environment constantly change their strategies, the environment becomes non-stable for every agent. The algorithm is challenging to converge in complex multi-agent scenarios.

The MADDPG [2] algorithm adopts the CTDE framework where each agent trains two networks, namely the Critic network and Actor network. The Critic network can obtain global information during training to guide the Actor network update. MADDPG can better cope with the non-stationarity of the environment, but it has a credit distribution problem. COMA [3] uses a counterfactual approach to solve the credit allocation problem, allowing the agent to judge their contribution to the global reward. MAAC [4] introduces an attention mechanism into the centralized Critic network, which enables agents to selectively pay attention to information from other agents according to attention weights when learning value functions, thus improving the learning efficiency of agents and the scalability of algorithms. MAPPO [5] applies PPO to the multi-agent scenario and adopts the centralized value function mode.

III. Proposed Method

This paper's core work is designing a curiosity mechanism in MARL and using intrinsic reward. To distinguish the contribution of different agents to the global curiosity reward, we do not use the global curiosity reward directly but instead use the Global Curiosity Assignment Method (GCAM). GCAM assigns the global curiosity reward to each agent based on the individual curiosity reward of the agent. Since the size of the individual curiosity reward corresponds to the novelty of the agent's access state, our method can assign the intrinsic reward to the agent according to its contribution to the global curiosity reward.

A. Curiosity-Inspired Exploration

A Curiosity-Inspired Exploration (CIE) mechanism is designed to calculate the curiosity reward based on GCAM for multi-agent cooperation. CIE calculates the reward distribution weight according to the individual curiosity reward of the agent and assigns the global curiosity reward to each agent according to the weight. In CIE, both global curiosity and individual curiosity are calculated by the Random Network Distillation (RND) model [6], as shown in the orange box in Figure. 1. RND consists of a prediction network and a target network. The target network's weights are randomly initialized and fixed, and the prediction network is trained only by the input data. When there are *N* agents in the task, CIE uses a total of *N+1* RND models: one RND model to calculate the global curiosity reward and *N* RND models to calculate the individual curiosity rewards. When sampling from the experience replay buffer(o_t, a_t, r_t, o_{t+1}),

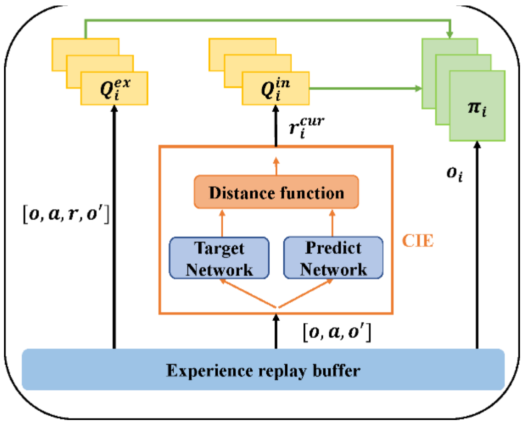


Figure 1. The network architecture of CIMADDPG.

global observations $o_{t+1} = [o_{t+1}^1, o_{t+1}^2, \cdots, o_{t+1}^N]$ can be fed into the global RND model to calculate the global curiosity reward r_{int}, and individual RND models for each agent to calculate the individual curiosity reward r_{int}^i.

After the individual curiosity reward of each agent is calculated, the softmax function is used to calculate the reward distribution weight of the agent, which can control the distribution of the global curiosity reward. The method of calculating the reward allocation weight P_i of agent i and the allocated curiosity reward r_i^{cur} is as follows:

$$P_i = e^{r_1^{int}} / (e^{r_1^{int}} + \cdots + e^{r_N^{int}}) \quad (1)$$

$$r_i^{cur} = P_i * r^{int} \quad (2)$$

CIE assigns larger curiosity rewards to agents visiting novel states to encourage them to explore unknown states and smaller curiosity rewards to agents visiting familiar states to prevent them from redundant exploration of familiar states.

B. Network Architecture

This section mainly considers how to use CIE in MARL algorithms so that each agent can effectively explore environments with sparse reward settings. Taking the MADDPG algorithm as an example, we propose a curiosity-inspired version, CIMADDPG. The CIMADDPG method trains two Critic networks for each agent: the external Critic network based on extrinsic reward and the internal Critic network based on intrinsic reward. It uses these two critics together to guide the update of the Actor network. The framework of CIMADDPG is shown in Figure 1.

Taking an environment with discrete action spaces as an example, the agent uses a stochastic policy instead of a deterministic one. We use $\pi = \{\pi_1, \cdots, \pi_N\}$ to represent the stochastic policies of *N* agents, and $\theta = \{\theta_1, \cdots, \theta_N\}$ to represent the parameters of the policy. After sampling from the experience replay buffer (o, a, r, o'), the external Critic in CIMADDPG is trained in the same way as the Critic in MADDPG is trained:

$$L(\omega_i) = E_{o,a,r,o' \sim D}[(Q_i^{ex}(o,a) - y_i^{ex})^2] \quad (3)$$

$$y^{ex} = r + \gamma * \overline{Q}_i^{ex}(o', a')|_{a' = \overline{\pi}(o')} \quad (4)$$

```
Algorithm 1 Curiosity-Inspired MADDPG
Init: The parameters for the external Critic, internal Critic, and Actor networks, $\omega, \eta, \theta$. The target network parameters are $\omega', \eta', \theta'$. The parameters for GCAM $\zeta$, learning rate $\alpha$, experience replay buffer $D$, max number of episodes $M$, and target network update interval $K$.
 1: for each episode = 1 to $M$ do
 2:   for t = 1 to $T$ do
 3:     Get the observation of the agent, $o^t = [o_1^t, \cdots, o_N^t]$
 4:     Select and execute $a^t$, get $[o^{t+1}, r^t]$
 5:     Save $[o^t, a^t, r^t, o^{t+1}]$, to $D$
 6:     if $|D| \geq batch\_size$
 7:       for agent $i = 1$ to $N$ do
 8:         Sample a batch form $D$
 9:         Calculate predict network parameter $\zeta$ from RND in GCAM by
            minimizing Eq. 8 and Eq. 9
10:         Calculate curiosity reward $r_i^{cur}$ by Eq. 2
11:         Update external critic $\omega$ by Eq. 3
12:         Update internal critic $\eta$ by Eq. 5
13:         Update Actor network $\theta_i$ by calculating Eq. 7
14:       end for
15:     end if
16:     if $t \% K == 0$ then
17:       $\theta' \leftarrow \tau\theta + (1-\tau)\theta'$
18:       $\omega' \leftarrow \tau\omega + (1-\tau)\omega'$
19:       $\eta' \leftarrow \tau\eta + (1-\tau)\eta'$
20:     end if
21:   end for
22: end for
```

Where Q_i^{ex} represents the value function fitted by the external Critic, r_i represents the external reward given by the environment, and $\overline{\pi} = \{\overline{\pi}_1, \cdots, \overline{\pi}_N\}$ represents the target policy of the agent.

After calculating the curiosity reward of the agent according to the CIE mechanism, the internal Critic r_i^{cur} is updated by minimizing the loss function $L(\eta_i)$.

$$L(\eta_i) = E_{o,a,r,o' \sim D}\left[\left(Q_i^{in}(o,a) - y_i^{in}\right)^2\right] \tag{5}$$

$$y^{in} = r_i^{cur} + \gamma * \overline{Q}_i^{in}(o', a')|_{a'=\overline{\pi}(o')} \tag{6}$$

Use the external Critic and the internal Critic together to guide the update of the Actor network, that is, update the Actor network according to the value function Q_i^{ex} and Q_i^{in} to calculate the following policy gradient:

$$\nabla_{\theta_i} J(\theta_i) = E_{o \sim D, a \sim \pi}\left(\nabla_{\theta_i} \log \pi_i(a_i|o_i) * \left(Q_i^{ex}(o,a) + Q_i^{in}(o,a)\right)\right) \tag{7}$$

C. Training of Proposed Algorithm

The Algorithm 1 presents the pseudocode of CIMADDPG.

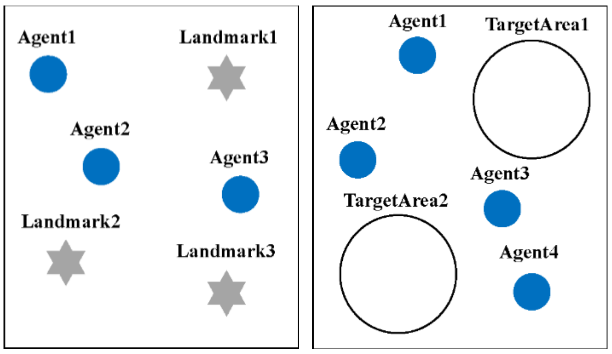


Figure 2. The experimental environments based on MPE. (a) *sparse spread.* (b) *sparse domain.*

The RND model's prediction and target networks' outputs are $\phi(\cdot)$ and $\psi(\cdot)$, respectively. In addition, the global RND model and the individual RND model's prediction network are trained by minimizing the loss function $L_p(\zeta)$ and $L_p(\zeta_i)$, respectively. The detailed calculation is shown as follows:

$$L_p(\zeta) = \frac{1}{b}\sum_b ||\phi(o^{t+1}) - \psi(o^{t+1})||_2 \tag{8}$$

$$L_p(\zeta_i) = \frac{1}{b}\sum_b ||\phi(o_i^{t+1}) - \psi(o_i^{t+1})||_2 \tag{9}$$

IV. Exprimental Analysis

A. Experimental Settings

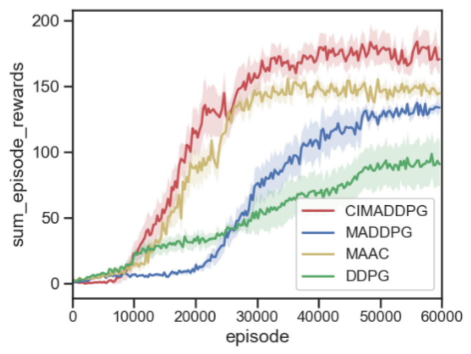
In order to verify the performance of the CIMADDPG algorithm proposed in this paper, we conduct a detailed investigation in two environments of cooperative navigation and cooperative occupation under a sparse reward setting.

Cooperative navigation. Cooperative navigation is a classic task in the MPE [2], which is also the experimental task used by MADDPG. As shown in Figure 2. (a), each agent needs to cover a landmark separately, avoiding collisions with other agents. We changed the original cooperative navigation task with dense rewards to one with sparse rewards and named it *sparse_spread*.

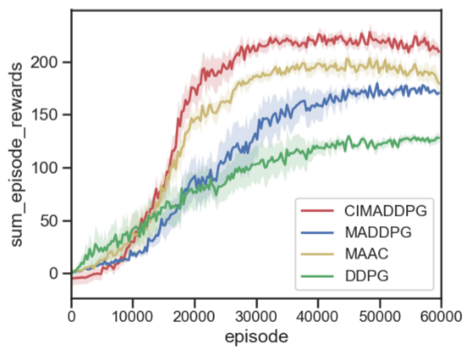
Cooperative occupation. In order to further test the coordination between agents based on the cooperative navigation task, we design another cooperative occupation task with sparse rewards, which is named *sparse_domain*. Figure 2. (b) shows the environment. In *sparse_domain*, every two agents must occupy a target area, and four agents must work in pairs to occupy two target areas within 45 steps to avoid a collision.

To verify the effectiveness of the proposed algorithm, CIMADDPG was compared with three different types of baseline algorithms: DDPG [7], MADDPG [2], and MAAC [4]. They correspond to the MARL algorithm of independent learning, centrally training decentralized execution, and SOTA.

The hyperparameters of the baselines are set to be consistent to ensure the fairness of the comparison. In addition, all algorithms are run five times using different random seeds to ensure the repeatability of experimental results.



(a)



(b)

Figure 3. The experimental results in MPE. (a) *sparse_spread*. (b) *sparse_domain*. The curves and shadows represent the mean and standard deviation of 5 experiments, respectively, and the horizontal and vertical coordinates represent the number of training rounds and the cumulative reward of rounds, respectively.

Table I. COMPARISON OF OUR METHOD (CIMADDPG) WITH BASELINES IN *sparse_spread*

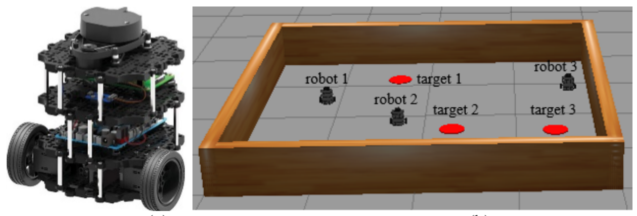
Algorithm	Average coverage		Distance	
CIMADDPG	**2.63**	**100%**	**0.17**	**100%**
MADDPG	2.28	-13.31%	0.57	-235.29%
MAAC	2.42	-7.98%	0.21	-23.53%
DDPG	1.91	-27.38%	0.86	-405.88%

Table II. COMPARISON OF OUR METHOD (CIMADDPG) WITH BASELINES IN *sparse_domain*

Algorithm	Average coverage		Distance	
CIMADDPG	**1.52**	**100%**	**0.43**	**100%**
MADDPG	1.09	-28.29%	1.26	-193.02%
MAAC	1.33	-12.50%	0.64	-48.84%
DDPG	0.87	-42.76%	1.89	-339.53%

B. Experimental Results and Analysis

Figure 3. (a) shows the experimental results of all algorithms in the *sparse_spread* task. It can be seen that our proposed CIMADDPG not only has faster convergence but also has better policy advancement. In the early stage of training, the MAAC needs to spend more time learning the allocation of attention weights. Still, in the later stage of training, the agent can selectively focus on important information to obtain better performance. DDPG has a faster learning speed than MADDPG in the early stage of training because, in the sparse reward environment, it can be regarded as a radical way to explore the environment to make decisions based on its information without considering the information of other agents. However, the performance of DDPG was the worst among all the algorithms in the late training period. This is attributed to the fact that the agents in DDPG can only access local information, which is not conducive to cooperation between agents. While other algorithms follow the paradigm of CTDE, and the agents can obtain global information, so they can better cooperate with other agents.



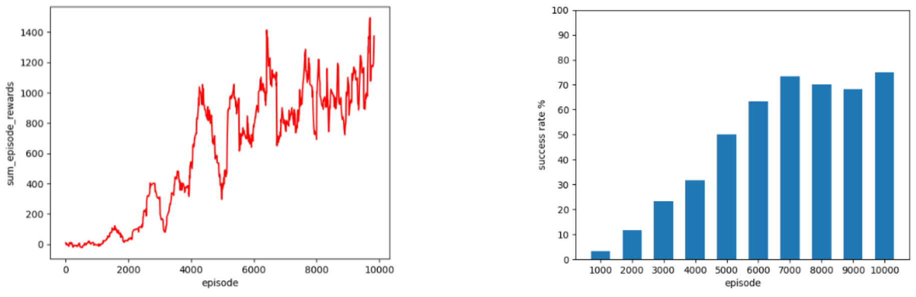
(a) (b)

Figure 4. The simulation environment of multi-robot cooperative navigation. (a) The TurtleBot3 wheeled robot is used in this experiment. (b) The simulation environment of multi-robot built-in Gazebo.

Two critical metrics are counted and analyzed: the number of landmarks the agent covers at the end of the episode and the sum of the distance between the agents and the nearest landmark. After the model is converged, the trained Actor network is tested for 1000 episodes, and the average value is calculated. As shown in Table I. , CIMADDPG performs better than other baselines, covering more landmarks on average and minimizing the sum of distances between each agent and the nearest landmark.

Figure 3. (b) shows the experimental results of all algorithms in the *sparse_domain* task. It can be seen that the CIMADDPG outperforms the baselines in both learning speed and final performance. The fact can be indicated that the curiosity-inspired exploration approach can help agents learn this policy of occupying areas more quickly. Because agents in MAAC can selectively pay attention to the agents that may enter the same area with them through the attention mechanism, MAAC can also achieve better performance. In the later stage of training, the performance of DDPG is very poor. Because in this task, only when two agents enter the area at the same time can they get a larger reward, and the agents in DDPG cannot obtain the information of other agents, it isn't easy to learn the cooperative behavior to enter the area with other agents. More obvious experimental results are recorded in Table II.

C. Multi-Robot Cooperative Navigation Application



(a) (b)

Figure 5. The experimental results of CIMADDPG in the multi-robot cooperative navigation task. (a) The episode cumulative rewards during training. (b) The navigation success rate.

We use the TurtleBot3 wheeled robot on the robot simulation software Gazebo to conduct the multi-robot cooperative navigation simulation experiment. Its structure is shown in Figure 4. . The map is four meters long and four meters wide, and there are three TurtleBot3 robots on the map. A red circle represents the robot's target, and the robot needs to go to the target without colliding with other robots. The shape of the TurtleBot3 robot is similar to a cylinder, and its coordinate origin is located in the center of the chassis. When the distance between the robots is less than 0.2m, it can be regarded as a collision, and when the distance between the robots and the target is less than 0.2m, it can be considered as reaching the target. In this simulation environment, three positions are selected as the robot's starting position, and the target points are randomly generated for each robot.

The episode cumulative reward in the training process of CIMADDPG is shown in Figure 5. (a). Driven by the intrinsic reward of curiosity, robots will explore the novel state in the environment and produce coordinated exploration behavior. It reduces the probability of collision between robots. At the beginning of the training, the episode cumulative reward is negative, indicating a collision between the robots or between the robots and the wall. After 2000 episodes, the robots gradually learn to explore the environment without crashing and reach the target through exploration, so the episode cumulative reward increase progressively. The overall episode cumulative reward obtained by the robot shows a trend of fluctuation and growth.

In training, the model is saved once every 1000 episodes, and the average success rate of navigation is calculated through 20 rounds of tests. In the test, each robot has its target and does not need to assign a target dynamically. If a robot reaches its target within 500 steps, it completes the navigation task. The average success rate of navigation is the proportion of robots that complete the navigation task within 20 rounds in the total number of robots. The test results are shown in Figure 5. (b). It is not difficult to see that our method achieves a high navigation success rate since the 6000 rounds, which indicates that the CIMADDPG algorithm can be used to handle multi-robot cooperative navigation tasks under sparse rewards.

V. Conclusion

To solve the problem of poor performance of MARL in sparse reward navigation tasks, this paper proposes a curiosity-inspired exploration method to enhance the agent's exploration ability. In addition, we designed a dual-critic network to work together to guide the policy updates to ensure that collaboration is not lost during multi-agent exploration. The experimental results of MPE and multi-Turtlebot3 robot cooperative navigation simulation prove that the proposed method is helpful for MARL to be implemented in multi-robot cooperative navigation.

In order to more effectively induce cooperative behavior of agents while exploring, calculating influence rewards based on curiosity rewards, that is, the influence of agents on the state transition of other agents, is one of the interesting future research directions.

Acknowledgment

This work is supported in part by Major Research Project of National Natural Science Foundation of China under Grant 92267110, National Natural Science Foundation of China under Grant 62076202 and 61976178, Open Research Projects of Zhejiang Lab under Grant 2022NB0AB07, and Shaanxi Province Key Research and Development Program of China under Grant 2023-YBGY-354, and 2022GY-090.

References

[1] M. Tan, "Multi-agent reinforcement learning: Independent vs. cooperative agents," in Proceedings of the tenth international conference on machine learning, 1993, pp. 330–337.

[2] R. Lowe, Y. I. Wu, A. Tamar, J. Harb, O. Pieter Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," Advances in neural information processing systems, vol. 30, 2017.

[3] J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual multi-agent policy gradients," in Proceedings of the AAAI conference on artificial intelligence, vol. 32, no. 1, 2018.

[4] S. Iqbal and F. Sha, "Actor-attention-critic for multi-agent reinforcement learning," in International conference on machine learning. PMLR, 2019, pp. 2961–2970.

[5] C. Yu, A. Velu, E. Vinitsky, J. Gao, Y. Wang, A. Bayen, and Y. Wu, "The surprising effectiveness of ppo in cooperative multi-agent games," Advances in Neural Information Processing Systems, vol. 35, pp. 24 611–24 624, 2022.

[6] Y. Burda, H. Edwards, A. Storkey, and O. Klimov, "Exploration by random network distillation," arXiv preprint arXiv:1810.12894, 2018.

[7] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," arXiv preprint arXiv:1509.02971, 2015.