mathematics

MDPI

Article

Learning to Utilize Curiosity: A New Approach of Automatic Curriculum Learning for Deep RL

Zeyang Lin, Jun Lai *, Xiliang Chen *, Lei Cao and Jun Wang

Command Control Engineering College, Army Engineering University of PLA, Nanjing 210007, China; hunterlzy@aeu.edu.cn (Z.L.); feiyuewuxian2018@aeu.edu.cn (L.C.); wangjun920811@aeu.edu.cn (J.W.)
* Correspondence: zhangk@aeu.edu.cn (J.L.); lgd_chenxiliang@aeu.edu.cn (X.C.)

Abstract: In recent years, reinforcement learning algorithms based on automatic curriculum learning have been increasingly applied to multi-agent system problems. However, in the sparse reward environment, the reinforcement learning agents get almost no feedback from the environment during the whole training process, which leads to a decrease in the convergence speed and learning efficiency of the curriculum reinforcement learning algorithm. Based on the automatic curriculum learning algorithm, this paper proposes a curriculum reinforcement learning method based on the curiosity model (CMCL). The method divides the curriculum sorting criteria into temporal-difference error and curiosity reward, uses the K-fold cross validation method to evaluate the difficulty priority of task samples, uses the Intrinsic Curiosity Module (ICM) to evaluate the curiosity priority of the task samples, and uses the curriculum factor to adjust the learning probability of the task samples. This study compares the CMCL algorithm with other baseline algorithms in cooperative-competitive environments, and the experimental simulation results show that the CMCL method can improve the training performance and robustness of multi-agent deep reinforcement learning algorithms.

Keywords: deep reinforcement learning; automatic curriculum learning; curiosity; sparse reward

MSC: 68T07

Citation: Lin, Z.; Lai, J.; Chen, X.; Cao, L.; Wang, J. Learning to Utilize Curiosity: A New Approach of Automatic Curriculum Learning for Deep RL. *Mathematics* **2022**, *10*, 2523. https://doi.org/10.3390/math10142523

Academic Editors: Jiangping Hu and Zhinan Peng

Received: 4 July 2022
Accepted: 15 July 2022
Published: 20 July 2022

1. Introduction

Deep reinforcement learning [1] combines the perception ability of deep learning with the decision-making ability of reinforcement learning, and has been widely used in the processing of complex decision-making tasks [2], such as Atari games [3], complex robot action control [4,5], and the application of AlphaGo intelligence [6]. In 2015, Hinton, Bengio and Lecun, famous experts in the field of machine learning, published a review paper on deep learning in *Nature*, which considered deep reinforcement learning as an important development direction of deep learning [7].

However, there is a significant problem in the application of deep reinforcement learning algorithms in multi-agent systems [8]. With the increase in the number of agents and the increase in the complexity of the environment, the coordination and cooperation between agents becomes more difficult, which can easily cause a situation where the Reinforcement Learning (RL) algorithm does not converge or even cannot be trained [9,10].

Curriculum learning [11], as a hot field of current artificial intelligence research, was proposed by Bengio et al. at the International Conference on Machine Learning (ICML) in 2009. Bengio et al. pointed out that the curriculum learning method can be regarded as a special kind of continuous optimization method, which can start with smoother (i.e., simpler) optimization problems and gradually add rougher (i.e., more difficult) non-convex optimization problems, and finally optimize the target task. In curriculum reinforcement learning algorithms [12], manually set the tasks of different difficulty levels, and gradually add more difficult tasks to the simple reinforcement learning tasks, so that the knowledge of

the source tasks can be reused in the process of learning difficult tasks, thereby accelerating the convergence of model to the optimal policy.

The above-mentioned predefined curriculum learning methods need to be manually set in advance in the process of task generation and sorting, so the quality of the generated curriculums will be directly affected by the experience of experts. However, the learning method of pre-defined curriculums requires manual curriculum difficulty assessment and sorting, and lacks task versatility. The current curriculum learning field gradually adopts automatic curriculum learning (ACL) instead of predefined curriculum learning to train reinforcement learning agents.

Before the agent learns the whole task, the difficulty of the experience samples in the experience replay buffer is evaluated and sorted, and the experience samples are learned in order from easy to difficult, so automatic curriculum learning [13] can realize the learning of difficult tasks, shorten the training time, and improve the training performance of task learning.

Traditional automatic curriculum learning often uses the temporal-difference error method to evaluate and sort the difficulty of task samples, that is, to obtain the optimal policy by maximizing the external reward value that appears in the process of interacting with the environment, but in the reward sparsity environment, the agent is difficult to obtain environmental reward feedback in long-lasting time steps. The lack of reward signals will affect the iteration and update of the agent's action policy, so it is hard for the agent to learn an effective policy.

To solve the above problems, this paper proposes a curriculum learning method based on curiosity module (CMCL), adding curiosity intrinsic reward in curriculum sorting criteria, the curiosity reward value of the experience samples was evaluated to obtain the curiosity priority, and the curriculum sequence of the experience samples was sorted together with the temporal-difference error, and the selection progress of the curriculum difficulty was adjusted by setting the curriculum difficulty factor, so as to enhance the exploration and training performance of the curriculum reinforcement learning algorithm for the environment. The experimental results of two tasks in multi-agent particle environment show that the CMCL method proposed in this paper can greatly improve the processing performance of multi-agent tasks in sparse reward environments compared with the three baseline algorithms.

The contributions of this paper are as follows:

(1) This paper proposes a curriculum reinforcement learning method based on the curiosity module. By adding curiosity priority to the curriculum sorting criteria, it can enhance the exploratory and robustness of reinforcement learning agents and avoid the appearance of turn-in-place agent;
(2) This paper introduces a curriculum difficulty factor in the process of selecting the curriculum difficulty of the model, and dynamically adjusts the difficulty of the currently selected curriculum through the curriculum difficulty factor, so as to realize automatic curriculum learning from easy to difficult priority experience.

The rest of this paper is organized as follows. Section 2 introduces related work, Section 3 introduces the MADDPG algorithm and the theory of automatic curriculum learning, Section 4 introduces the CMCL algorithm in detail, Section 5 presents experimental results and analyzes them, Section 6 presents discussion and Section 7 draws some conclusions.

2. Related Work

How to reasonably arrange the sequence of curriculums and select curriculums in the process of curriculum learning is the main research problem of current automatic curriculum reinforcement learning research. Carlos Florensa et al. [14] used generative networks to propose tasks that the agent needs to implement to automatically generate curriculums capable of learning many types of tasks without requiring prior knowledge. Ren et al. [15] proposed an automatic curriculum reinforcement learning method that uses

a priority curriculum sorting method to extract experience samples from the experience replay buffer to achieve automatic curriculum learning. Jiayu Chen et al. [16] used the perspective of variational inference to automatically generate training curriculums for the task environment and the number of agents from two aspects of task expansion and agent expansion, which can be used to solve cooperative multi-agent reinforcement learning problems in difficult environments.

Curiosity-driven agent exploration is an important approach in reward function design for reinforcement learning. In supervised learning, curiosity is used to alleviate the problem of imbalanced representation and distributional bias among data [17,18]. Pathak et al. [19] used curiosity as an intrinsic reward value for agents, which can encourage the agent to explore new environmental states. Our method is derived from the curiosity mechanism of the human brain [20]. Curiosity is used as a reference standard for automatic curriculum learning's curriculum sorting, which can complement the priority experience replay algorithm (PER). The selection probability of novel samples is increased in the samples to balance the exploration of the uncertain state in the process of environmental exploration of multi-agent system.

The most important works related to our method include the self-adaptive priority correction algorithm proposed by Hongjie Zhang et al. [21], the High-Value Prioritized Experience Replay proposed by Xi Cao et al. [22], and the Curriculum Guided Hindsight Experience Replay proposed by Meng Fang et al. [4]. Hongjie Zhang et al. predicted the sum of the real Temporal-Difference error of all samples in the experience replay, and corrected it by an importance weight. Xi Cao et al. designed a priority experience replay method based on the combination of temporal-difference error and value for the sparse reward environment, Meng Fang et al. applied the curiosity mechanism to the Hindsight experience replay algorithm (HER), and learned successful experience from failure through the HER mechanism. Our method provides a further improvement on the basis of the above methods. As one of the curriculum sorting standards in the priority experience replay algorithm, the curiosity mechanism can compensate for the exploratory and randomness of the agent in the sparse reward environment, thereby improving the training performance and robustness of the algorithm.

3. Basic Concepts

This chapter will sequentially introduce some important concepts of Deep Reinforcement Learning, Multi-Agent Deep Deterministic Policy Gradient algorithms (MADDPG), and Automatic Curriculum Learning (ACL).

3.1. Deep Reinforcement Learning

Reinforcement learning [23] consists of two parts: agents and environment. To maximize agents' total reward value, the agents observe the initial state in the environment, take actions from an action set, and the environment accepts the action and gives the agents a reward. This process can be modeled as a Markov decision quintuple (S, A, R, P, γ), where S represents the state space, A represents the action space, R represents the reward function, P represents the state transition function, and γ represents the discount factor. The schematic diagram of reinforcement learning is shown in Figure 1.

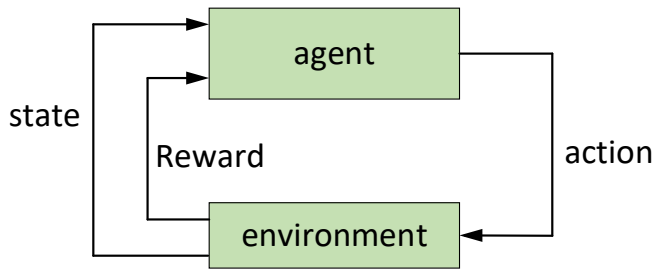


Figure 1. Schematic diagram of reinforcement learning.

Deep reinforcement learning approximates policy function and value function through a deep learning multi-layer neural network, thereby solving the high-dimensional mapping problem caused by continuous high-dimensional state-action pairs [24]. The goal of agents is to maximize expected reward $J(\pi_\theta) = E_{\tau\sim\pi_\theta}[R(\tau)]$ by continuously optimizing the policy π_θ, then the optimal policy is

$$\pi_\theta^* = \underset{\pi_\theta}{argmax} E_{\tau\sim\pi_\theta}(\sum_{t=0}^{\infty}\gamma^t r_t) \tag{1}$$

where r_t represents the reward of agents at time t.

Deep reinforcement learning algorithms can be divided into following three categories [25], deep reinforcement learning based on value function, deep reinforcement learning based on policy gradient, and deep reinforcement learning based on the actor-critic (AC) framework. The DRL algorithm based on the structure of the AC framework uses the error of the value function to guide the policy update and improve the performance of the algorithm training. The policy π_θ is updated by policy gradient $\nabla_\theta J(\pi_\theta)$ of expected reward, the formula is as follows:

$$\nabla_\theta J(\pi_\theta) = E_{\tau\sim\pi_\theta}[\sum_{t=0}^{T}\nabla_\theta \log\pi_\theta(a|s)R(\tau)] \tag{2}$$

where $\pi_\theta(a|s)$ represents the actor Function and $R(\tau)$ represents the critic Function.

3.2. MADDPG Algorithm

Multi-Agent Deep Deterministic Policy Gradient algorithm [26] (MADDPG) is an improved Multi-Agent Reinforcement Learning algorithm based on the AC network framework, which can be considered as an extended application of the DDPG algorithm in a multi-agent environment. To solve the problem of non-stationarity in Multi-agent Training Process [27], the MADDPG pioneered the principle of centralized training and distributed execution (CTDE), that is, in the training stage, the MADDPG algorithm allows the agents to obtain global information during learning, only local information is used in the decision execution. The AC training framework can be seen as an actor network for policy exploration, critic network as an evaluator to evaluate the policy, and obtain the current optimal policy. The algorithm structure consists of actor network, critic network, target actor network and target critic network. The training framework of the MADDPG algorithm is shown in Figure 2. The MADDPG algorithm stores experience tuples through the experience replay mechanism:

$$D_i = (o_1, \cdots, o_N, a_1, \cdots, a_N, r_1, \cdots, r_N, o_1', \cdots, o_N') \tag{3}$$

During the training process, experience tuples are stored in batches in the experience replay buffer, and the experience replay buffer extracts small samples of experience in stages and inputs them into the neural network for model training. This experience replay mechanism can reduce the degree of association between experience tuples, thus improving the neural network training efficiency. The MADDPG algorithm updates the action network of agents using the stochastic gradient descent method. The formula is as follows:

$$\nabla_{\theta^\pi} J = \frac{1}{K}\sum_{j=1}^{K}\nabla_{\theta^\pi}\pi(o,\theta^\pi)\nabla_a Q(s,a_1,a_2,\ldots,a_N,\theta^Q) \tag{4}$$

In the formula, o and a_i represent the observation value and action of the ith agent respectively; $\pi(o,\theta^\pi)$ represents the action of agent i obtained by inputting the observation value into actor network.

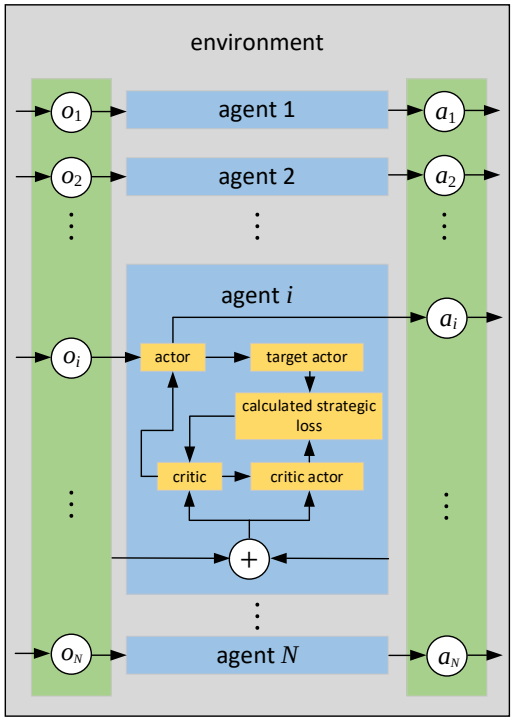


Figure 2. MADDPG algorithm training framework diagram.

The critic network of agents is iteratively updated as follows to minimize the loss function:

$$L = \frac{1}{K}\sum_{j=1}^{K}\left(y_j - Q(s_j, a_1, a_2, \ldots, a_N, \theta^Q)\right)^2 \tag{5}$$

In the formula, the function y represents the cumulative average reward of agent i in the target actor network.

The network parameters of target actor network and target critic network are replicated and updated in stages:

$$\theta_i' = \tau\theta_i + (1-\tau)\theta_i' \tag{6}$$

In the formula, τ represents the control parameter of the network parameter updating frequency, which can stabilize the parameter network update process. θ_i' represents the target network parameter of the ith agent, and θ_i represents the initial network parameter of the ith agent.

In view of the good stability and convergence of the MADDPG algorithm, it can be applied to various task scenarios such as cooperative, competitive and hybrid. The innovation and experimental verification of the algorithm in this paper are partly based on the MADDPG algorithm and its accompanying multi-agent particle environment (MPE).

3.3. Automatic Curriculum Learning

End-to-end deep reinforcement learning methods have led to breakthroughs in board games, real-time policy games, and path planning problems. However, reinforcement learning agents still face difficulties and challenges when dealing with many application scenarios [13]. The reason is that agents need to fully interact with the environment to obtain enough information to continuously modify its own policy, but the environment itself has the problems of reward sparseness, partial observability, delayed reward, and too high dimension of action space, which leads to the problem that the training time of the agent is too long or even unable to converge when dealing with difficult tasks.

In response to the above problems, Curriculum Learning (CL) can utilize knowledge from source tasks to speed up the learning of complex target tasks, thus improving the training performance of reinforcement learning agents on fixed task sets [28]. As an

important paradigm in the field of machine learning, curriculum learning can imitate the human learning sequence from easy to difficult. In the initial stage of reinforcement learning, the curriculum learning algorithm trains the model in a simple simulation environment (fewer obstacles and more reward values), and as the training progresses, the simulation environment is gradually added with more and more difficult (sparse reward values and more obstacles), and finally, the algorithm is validated in a full simulation environment.

Most traditional curriculum learning methods use predefined methods [13], that is, using expert experience to evaluate the difficulty of task curriculums and formulate curriculum plans from the perspectives of the number of agents, initial state distribution, reward function, goals, environment distribution, opponent policy, etc., such as tasks with a higher number of agents and more obstacles are generally considered more difficult training environments. Because the predefined curriculum learning method requires manual assessment and sorting of curriculum difficulties and lacks task versatility, the current curriculum learning field gradually adopts automatic curriculum learning instead of predefined curriculum learning to train reinforcement learning agents [29].

The current automatic curriculum learning process can be divided into curriculum sorting stage and curriculum selection stage [30]. The main idea is to construct a task curriculum sampler $q(n, \phi)$ based on the experience replay buffer, which can evaluate the difficulty of the transitions in the experience replay buffer and sort them from easy to difficult, and then the task $M(n, \phi)$ that is currently most suitable for agent training is extracted in real time from the experience replay to maximize the cumulative reward value of the reinforcement learning agent $J(\theta)$, ϕ represents the environmental factor variables that affect the difficulty of task curriculum.

To prove that curriculum updating can increase the cumulative reward value of agents in the process of automatic curriculum learning, in this paper, the proof is performed as follows from the perspective of mathematics.

Proof. For a given number n of agents, $J(\theta)$ can be simplified as follows:

$$\begin{aligned} J &= \boldsymbol{E}_{\phi\sim p}[V(\phi,\pi)] = \boldsymbol{E}_{\phi\sim q}\left[\tfrac{p(\phi)}{q(\phi)}V(\phi,\pi)\right] \\ &= \boldsymbol{E}_{\phi\sim q}\left[V(\phi,\pi) + \left(\tfrac{p(\phi)}{q(\phi)} - 1\right)V(\phi,\pi)\right] \\ &\geq \underbrace{\boldsymbol{E}_{\phi\sim q}[V(\phi,\pi)]}_{J_1:policy\ update} + \underbrace{\boldsymbol{E}_{\phi\sim q}\left[V(\phi,\pi)log\frac{p(\phi)}{q(\phi)}\right]}_{J_2:curriculum\ update} \end{aligned} \tag{7}$$

□

In the formula, $p(\phi)$ represents the uniform distribution of ϕ in the range of possible values. For all ϕ, the inequality is due to $x - 1 \geq \log x$, the equal sign of the inequality holds if and only if $p(\phi) = q(\phi)$.

Through the simplification of the above equation, the cumulative reward value $J(\theta)$ can be composed of the policy update reward J_1 and the curriculum update reward J_2. The policy update reward J_1 represents that reinforcement learning agents update their own policy functions iteratively to maximize their reward value obtained from the environment, and the curriculum update reward J_2 represents the task curriculum sampling sorting and adjustment through the task curriculum sampler $q(n, \phi)$, which can improve the agent's ability to explore environment and the training performance of the model to maximize agents' cumulative reward value.

In traditional automatic curriculum learning algorithms, the ordering of task curriculums often takes the environmental reward value of agents as the reference standard, that is, it adjusts its own action policy according to the external reward value. However, in sparse reward environments, it is difficult for an agent to obtain positive or negative rewards from the environment during most of the exploration process. Under the framework of the

traditional automatic curriculum learning algorithm, selecting the task curriculum from low to high according to temporal-difference error can easily lead to overfitting of the model training, and agents stay in circles in the environment, making it difficult to train a good policy.

4. Curriculum Reinforcement Learning Based on Curiosity Model

This paper proposes a general automatic curriculum learning framework—curiosity module-based curriculum learning for deep RL (CMCL), which is divided into two stages: curriculum sorting and curriculum selection. For all reinforcement learning tasks, suppose $D = \{d_1, d_2, \cdots, d_j, \cdots, d_K\}$ represents the experience sample set in experience replay buffer, and the task curriculum sampler $q(n, \phi)$ is used to operate on experience sample set D. The first stage is to evaluate and sort the difficulty of the samples in experience sample set to generate a curriculum learning plan; the second stage selects curriculums according to the set ability evaluation rules according to the curriculum plan.

The core of the curriculum difficulty sorting is to define the difficulty of the task samples. To convert the task samples in the experience replay into a curriculum sequence, a curriculum index function (CI) needs to be defined to calculate the priority p_{d_j} of task sample d_j.

Definition 1. *Curriculum Index Function (CI).*

The function $CI(d_j) \to \mathbb{R}$ is used to define the curriculum sequence of the task sample d_j in the experience replay D. For the task sample d_i and d_j, if $CI(d_i) < CI(d_j)$, the curriculum sequence of task sample d_i is before the task sample d_j.

$$CI(d_j) = KP(c_j, \lambda) + \eta CP(d_j) \tag{8}$$

In this paper, the curriculum sequence function is divided into two parts: $KP()$ and $CP()$. $KP()$ represents K-fold-priority function, $CP()$ represents curiosity-priority function, and c_j represents the K-fold teacher model score of task sample d_i, λ represents the curriculum learning factor, η represents the hyperparameter, which is used to control the efficiency and exploration of sample learning.

4.1. K-Fold Priority Experience Replay

In this paper, the absolute value of the temporal-difference error of the neural network is used as a reference standard for the curriculum sequence function $CI(d_j)$, and the difficult task is defined as the task with a large weight correction value for the current neural network model. The reason is that tasks with large temporal-difference error may have an adverse effect on the improvement of training model ability. For example, 1. The random noise during the model training process is prone to data deviation, thereby affecting the training accuracy of model; 2. In the stochastic gradient descent process of deep neural network training, tasks with large temporal-difference error often require a small update step size to obtain a better model convergence effect.

In this paper, the K-fold cross-validation method is used to evaluate the difficulty of the samples in the experience replay buffer, and experience replay D is divided into K equal parts $\left\{\widetilde{D}_i : i = 1, 2, \ldots, K\right\}$, and trained separately to obtain K teachers Model network $\theta = \{\theta_1, \theta_2, \cdots, \theta_K\}$, since the experience replay D is divided, the obtained K teacher model networks are independent of each other. The training formula of the teacher model network is as follows:

$$\begin{gathered} \widetilde{\boldsymbol{\theta}}_i = \underset{\widetilde{\boldsymbol{\theta}}_i}{argmin} \sum_{d_j \in \widetilde{D}_i} L(d_j, \widetilde{\boldsymbol{\theta}}_i) \\ i = 1, 2, \ldots, K \end{gathered} \tag{9}$$

where L represents the loss function of the temporal difference error.

The K teacher models obtained are cross-validated. For example, if sample d_j belongs to teacher model i, then the sample d_j is scored on the $K-1$ teacher models other than its own teacher model i. The scoring process can be expressed as follows:

$$\begin{aligned} c_{ji} &= \left(y - Q^{\pi}_{teacher}(s, a_1, a_2, \ldots, a_N)\right)^2 \\ y &= r_j + \gamma Q^{\pi'}(s', a'_1, a'_2, \ldots, a'_N)\Big|_{a'_v = \pi'_v(o_v)} \end{aligned} \tag{10}$$

In the formula, c_{ji} represents the difficulty score of the teacher model i to the sample d_j, $Q^{\pi}_{teacher}$ represents the Q value obtained by inputting the state value s and the action value a into the value function network, and $Q^{\pi'}$ represents the Q value obtained after state s and the action values a are input into the policy function network, γ represents the discount factor, and the final difficulty score of the task sample d_j is the sum of the difficulty scores of all other teacher models:

$$c_j = \sum_{i \in (1,\ldots,K), i \neq k} c_{ji} \tag{11}$$

Definition 2. *K-Fold Priority Function (KP).*

The function $KP(c_j, \lambda) \rightarrow [0, 1]$ is used to define the K-fold priority of task sample d_j in experience replay D, c_j represents the final difficulty score of task sample d_j after K-fold cross-validation, λ represents the curriculum learning currently selected task curriculum difficulty factor. The K-fold priority function $KP(c_j, \lambda)$ is expressed as follows:

$$KP(c_j, \lambda) = \begin{cases} e^{c_j - \lambda} & , c_j \leq \lambda \\ \frac{1}{log(1-\lambda)} log(c_j - 2\lambda + 1), & \lambda < c_j < 2\lambda \\ 0 & , c_j \geq 2\lambda \end{cases} . \tag{12}$$

where 1. $KP(cj, \lambda)$ is monotonically decreasing when $c_j > \lambda$; 2. $KP(cj, \lambda)$ is monotonically increasing when $c_j < \lambda$; 3. $KP(c_j, \lambda)$ is the maximum value when $c_j = \lambda$.

The K-fold priority function outputs a scalar with a value range of $[0, 1]$ by inputting the difficulty score c_j of the task sample and the curriculum factor λ, thereby reflecting the sample priority of the task sample in the dimension of temporal-difference error. As the curriculum learning progresses, the curriculum factor λ can be gradually increased, thereby increasing the priority of the task curriculum with higher difficulty score c_j. Since the selection probability of task samples is proportional to the K-fold priority, agents can frequently select empirical samples which fit the current model capabilities. The graph of the K-fold priority function is shown in Figure 3, where $\lambda = 0.6$ is shown in the figure.

The framework of the K-Fold Cross-Validation method is shown in Figure 4.

4.2. Curiosity Exploration Rewards

In the K-fold priority function $KP(c_j, \lambda)$, we use the temporal-difference error as the reference standard for prioritization, which can improve the utilization efficiency of task samples and the robustness of training. However, in a multi-agent system, the traditional reinforcement learning algorithm uses extrinsic reward to guide agents to adjust their own policy. The agents take actions in environment to interact with the environment. When the policy is correct, it will get a positive reward value, otherwise it will get a negative reward value. This extrinsic reward method can achieve good performance in most RL environments, but in a sparse reward value environment, agents do not obtain immediate reward value most of the time they explore in the environment, and then agents are impossible to adjust their own policy according to their reward value, which will greatly reduce their convergence speed and training efficiency of the algorithm.

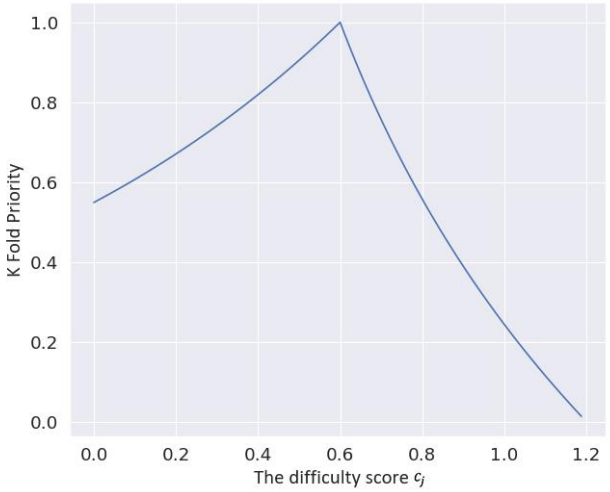


Figure 3. K-Fold priority function.

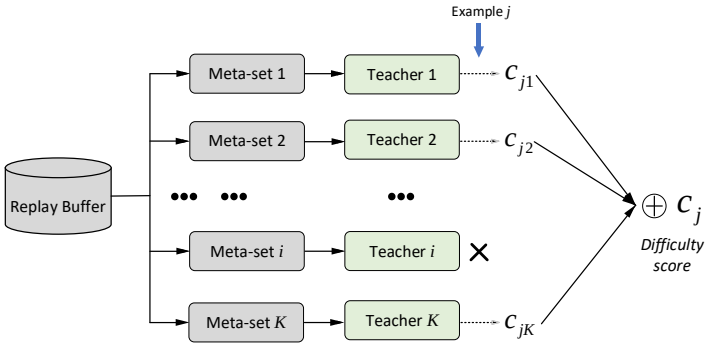


Figure 4. K-Fold Priority Cross Validation framework diagram.

Inspired by the theory of intrinsic motivation, based on the curiosity exploration mechanism [11], this paper uses the curiosity exploration reward as one of the reference standards of curriculum sequence function $CI(d_j)$ to enhance the agent's exploration of environment and avoid the over-fitting phenomenon of "turning in place" of agents.

The basic principle of curiosity exploration mechanism is that when the next state is inconsistent with the predicted state of policy network, the intrinsic reward of curiosity is generated. The greater the difference between actual state and predicted state, the greater the value of curiosity reward.

This curiosity-based mechanism is called the Intrinsic Curiosity Module (ICM), and the curiosity reward value is calculated through two sub-module networks. The first sub-module uses a feature convolutional neural network to extract the eigenvalues of the state s_t in experience samples, and encoded as $\phi(s_t)$, the second sub-module contains a forward neural network θ_F and an inverse dynamic network θ_I. The evaluation mechanism of curiosity reward value is shown in Figure 5.

In the ICM mechanism, the inverse dynamic network θ_I can estimate action value a_t through function g:

$$\hat{a}_t = g(s_t, s_{t+1}; \theta_I) \tag{13}$$

In the formula, a_t represents the actual action taken from state s_t to state s_{t+1}, $\hat{a}_t$ represents the estimated action of a_t, (s_t, a_t, r, s_{t+1}) experience tuple is obtained from the experience replay D, and the network parameters of reverse dynamic network θ_I are optimized by the following expressions:

$$\min_{\theta_I} L_I(\hat{a}_t, a_t) \tag{14}$$

where L_I represents the loss function between the predicted action value $\hat{a}_t$ and the actual action value a_t. The maximum likelihood estimates of the parameters θ_I of the inverse dynamic network can be obtained by minimizing L_I.

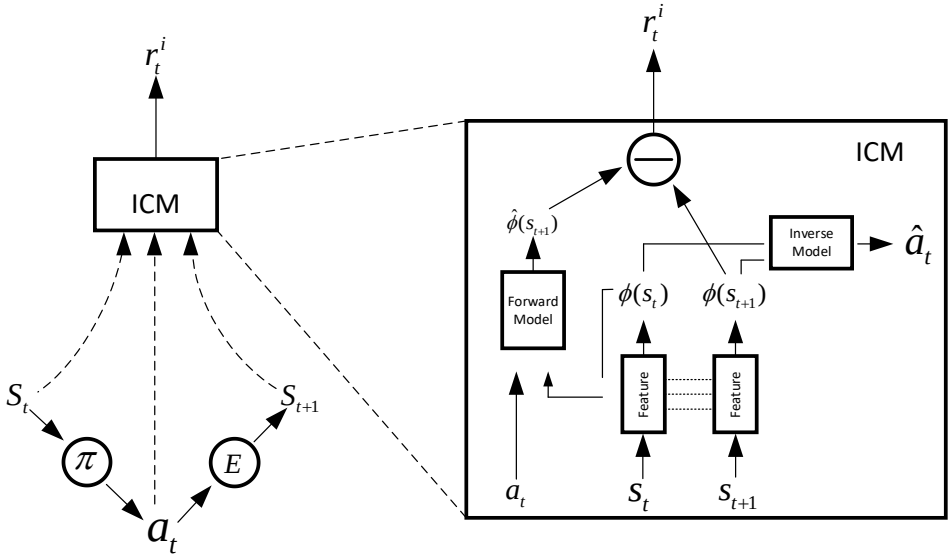


Figure 5. Curiosity reward evaluation mechanism.

For the forward neural network θ_F, the estimated state value a_t at the next time step $t+1$ can be obtained by inputting action value a_t and eigenvalue $\hat{\phi}(s_{t+1})$ of the state s_t.

$$\hat{\phi}(s_{t+1}) = f(\phi(s_t), a_t; \theta_F) \tag{15}$$

where the forward neural network parameter θ_F is optimized by the following loss function:

$$L_F(\phi(s_t), \hat{\phi}(s_{t+1})) = \frac{1}{2}\|\hat{\phi}(s_{t+1}) - \phi(s_{t+1})\|_2^2 \tag{16}$$

Then the overall optimization function learned by reinforcement learning agents is

$$\min_{\theta_P, \theta_I, \theta_F} \left[-\lambda E_{\pi(s_t;\theta_P)} \left[\sum_t r_t \right] + (1-\beta) L_I + \beta L_F \right] \tag{17}$$

In the formula, $0 \leq \beta \leq 1$ represents the weight parameter between the inverse dynamic network and the forward neural network, $\lambda > 0$ represents the weight parameter between the intrinsic curiosity reward value and the gradient descent loss function, and the available curiosity reward value is as follows:

$$r_t^i = \frac{1}{2}\|\hat{\phi}(s_{t+1}) - \phi(s_{t+1})\|_2^2 \tag{18}$$

Definition 3. *Curiosity Priority Function (CP).*

Function $CP(r_t^i(d_j)) \rightarrow [0,1]$ is used to define the curiosity priority of task sample d_j in experience replay D, $r_t^i(d_j)$ represents the curiosity reward value of the task sample d_j. The curiosity-priority function expression of $CP(r_t^i(d_j))$ is as follows:

$$CP(r_t^i(d_j)) = -e^{(-\frac{(r_t^i(d_j))^2}{10})} + 1 \tag{19}$$

where $CP(r_t^i(d_j))$ is a monotonically increasing function of $r_t^i(d_j)$.

From the above, the curriculum sequence function $CI(d_j) = KP(c_j, \lambda) + \eta CP(d_j)$ can be obtained, that is, the priority of each experience sample c_j in the experience replay D, then the sampling probability of each experience sample c_j is as follows:

$$P(d_j) = \frac{p_{d_j}^a}{\sum p_{d_j}^a} \tag{20}$$

In the formula, p_{d_j} represents the priority of the task sample d_j, and a represents the use degree of the priority p_{d_j}.

4.3. Algorithm Framework and Pseudocode

The CMCL algorithm proposed in this paper combines the K-fold priority function and the curiosity priority function in the curriculum sorting stage, so as to use temporal-difference error and curiosity reward to jointly sort curriculums. Adjusting the curriculum factor, the K-fold priority selection of task samples can be controlled to ensure that agents frequently select samples that are most suitable for the current training difficulty, and to improve the exploration of the environment by agents. The basic framework of the CMCL algorithm is shown in Figure 6, and Algorithm 1 describes the training process of the CMCL algorithm.

Algorithm 1: CMCL algorithm.

Input: experience replay buffer D, curriculum factor λ, curriculum stride μ, balance weight η, curriculum sequence vector $ci = [ci_1, ci_2, \cdots, ci_N]$
Output: The final policy π_θ
for episode = 1 to *max_episode* **do**
  Initialize a random process N for reinforcement learning action exploration
  Receive initial state s_0
  for t = 1 to *max_episode_length* **do**
    In state s_t, the agents select action a through policy network $\pi_\theta(s_t)$
    Obtain the reward r given by environment E
    Store (s_t, a, s_{t+1}, r) in experience replay buffer D
    $s_t \leftarrow s_{t+1}$
    The experience samples in D are sampled for K-level teacher model training $\left\{\widetilde{\theta}_i : i = 1, 2, \ldots, K\right\}$

$$\widetilde{\theta}_i = \underset{\widetilde{\theta}_i}{argmin} \sum_{d_j \in \widetilde{D}_i} L(d_j, \widetilde{\theta}_i)$$

    The score of experience sample d_j is evaluated by cross validation $c_j = \sum_{i \in (1,\ldots,N), i \neq k} c_{ji}$
    The K-fold priority $kp_j = KP(c_j, \lambda)$ can be obtained according to Equation (12)
    Calculate the curiosity reward $r_t^i = \frac{1}{2}\|\hat{\phi}(s_{t+1}) - \phi(s_{t+1})\|_2^2$
    The curiosity priority $cp_j = CP(r_t^i(d_j))$ can be obtained according to Equation (19)
    Update curriculum sequence function ci_j by $ci(d_j) = kp(c_j, \lambda) + \eta cp(d_j)$
    for agent $v = 1$ to *N_agent* **do**
      Sample a minibatch of transitions (s_t, a, s_{t+1}, r) from D according to the priority sampling probability

$$P(d_j) = \frac{p_{d_j}^a}{\sum p_{d_j}^a}$$

      The neural network parameter θ was updated by gradient descent algorithm
    end for
  Adjust curriculum factor λ based on current model capabilities $\lambda = \lambda + \mu$
  end for
end for

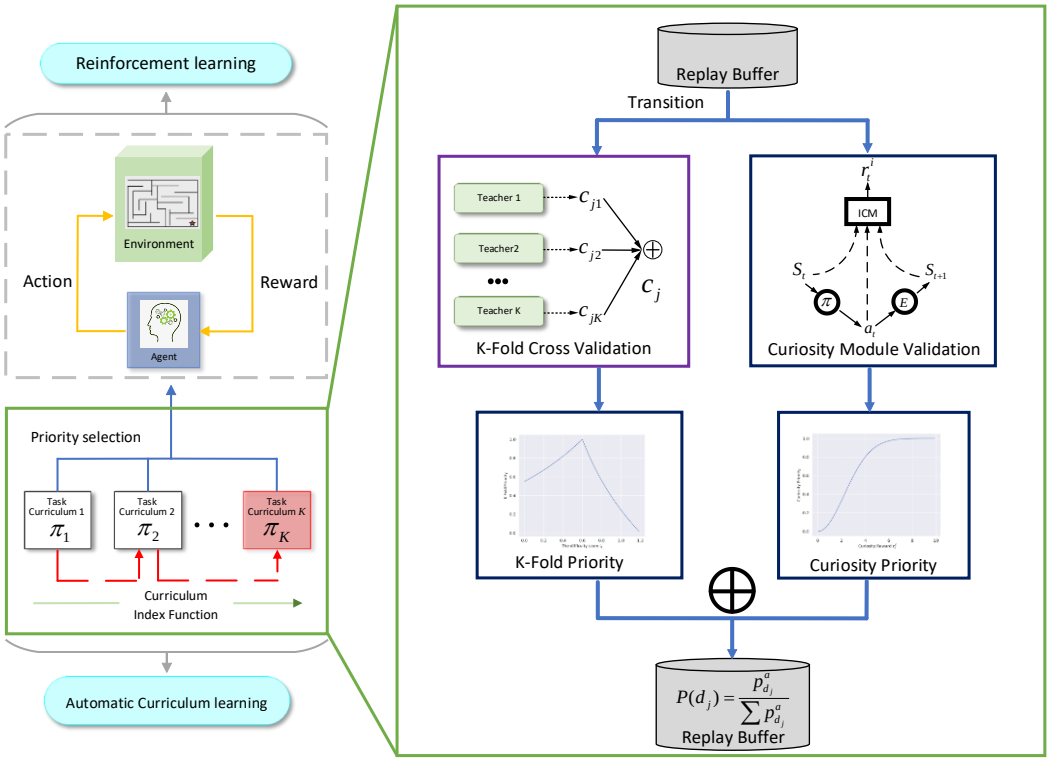


Figure 6. Framework diagram of curriculum reinforcement learning algorithm based on curiosity module.

5. Experiment

In this paper, the simulation verification of the CMCL algorithm is carried out in Multi-Agent Particle Environment [26] (MPE), and the multi-agent cooperative task and the competitive task are used as the target tasks. Based on the environment, a sparse reward value scenario is constructed to test the performance of the CMCL algorithm in teamwork and policy confrontation respectively. Each set of experiments is carried out in the experimental environment of Ubuntu18.04.3 + OpenAI + PyTorch, and adopts the hardware conditions of Intel Corei7-9700K + 64G + GeForceRTX2080. In our environment, the CMCL algorithm is compared with various baseline algorithms to demonstrate the effectiveness and feasibility of the CMCL algorithm. The key hyperparameters set for the RL training process are listed in Table 1. The state value and action value of the agents are input at the input end of the neural network, and the target Q value of the agents is obtained through the calculation of the neural network. The loss function is obtained by subtracting the original Q value, and the original Q value function is updated. Finally, the reinforcement learning algorithm is applied to the deep learning structure.

Table 1. Parameter setting of the DRL process.

Parameters	Values
Discount factor	0.99
Size of RNN hidden layers	64
Size of replay buffer	5000
Exploration	0.1
Initial curriculum factor λ	0.1
Batch size of replay buffer	128
Learning rate of actor network	0.001
Learning rate of critic network	0.001
Update rate of target network	0.01

5.1. Experimental Environment

5.1.1. Cooperative Experiment

The multi-agent cooperation experiment adopts the cooperative navigation experiment in the MPE environment. As shown in the Figure 7, N agents and N landmarks are randomly generated in a square two-dimensional plane with side length 1. The plane is surrounded by walls, and the agents can observe landmarks, but cannot observe the walls, and their missions are to reach landmarks in as few steps as possible and avoid collisions with other agents.

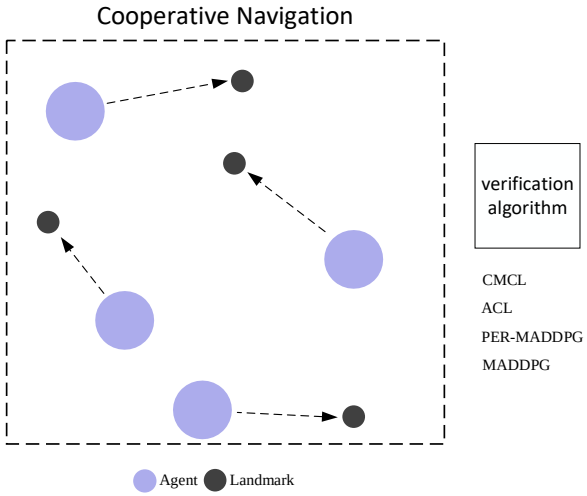


Figure 7. Cooperative navigation experimental environment.

Combined with the size of the two-dimensional plane, it is stipulated that when an agent enters an area with a radius of 0.1 around a landmark, the landmark is considered covered by the agent, and the cooperative navigation task is considered successful only when all landmarks are uniquely covered.

In the reward value setting of the experimental environment, to construct a sparse reward value scene, we cancel the dense reward function set according to the distance between the agent and the landmark in the original MPE environment. Therefore, the reward value obtained by each agent at each time step consists of only two parts, including 1. When there is a collision between the agents or the agent hits a wall, the environment gives a negative reward value, that is, agent collision reward value C_1; 2. When the agent covers the landmark, the environment gives a positive reward value, that is, the agent covers the landmark reward value C_2.

The agent collision reward value is as follows:

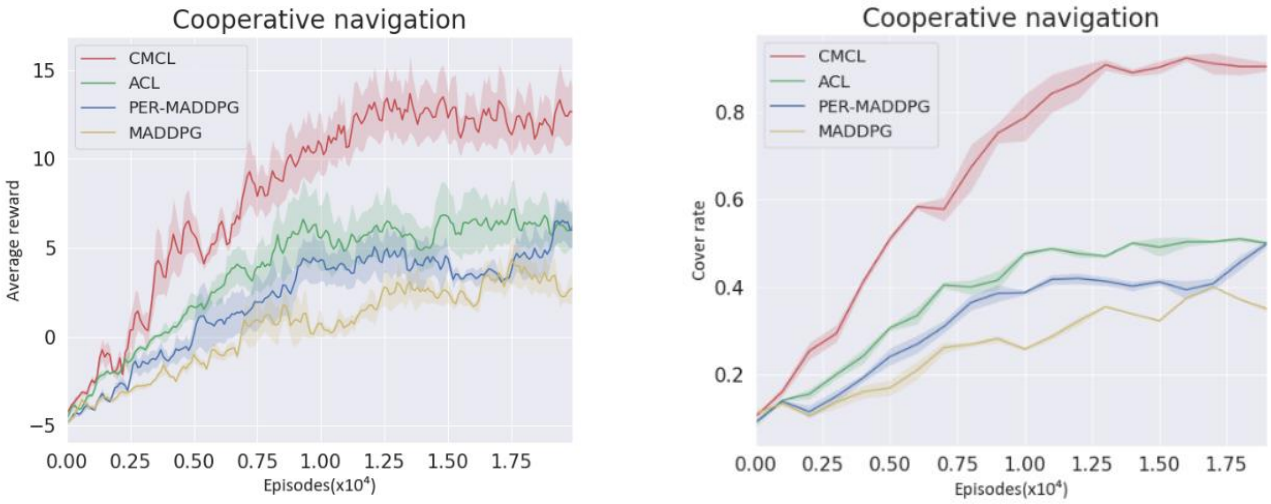
$$C_1 = \begin{cases} -1, if\ collided \\ 0, if\ not\ collided \end{cases} \tag{21}$$

The agent coverage landmark reward value is as follows:

$$C_2 = \begin{cases} +4, if\ covered \\ 0, if\ not\ coverd \end{cases} \tag{22}$$

As shown in Figure 7, in the $N = 4$ environment, the CMCL, ACL, PER-MADDPG, and MADDPG algorithms are used to control the movement of the agent. To prevent the agent from spinning in place or meaningless exploration, the episode duration is set to 30 steps, that is, when the agent finishes exploring after 30 steps, the environment is initialized to start a new episode of exploration.

Figure 8 shows the average reward value graph and the coverage graph obtained by the four algorithms after 20,000 episodes of training in the cooperative navigation environment. Figure 9 shows the bar graph of the average reward value of the four algorithms in 20,000 episodes, that is, the quotient of the total reward value obtained by the four algorithms in the whole training session and the number of sessions. As can be seen from the curve in Figure 8, at the beginning of the algorithm training, the agent is prone to colliding with other agents or with the wall. As the training progresses, agents gradually learn the policy of cooperatively covering landmarks. The curve of the CMCL algorithm oscillates slightly in the early training process, and gradually smooths in the later stage, and can obtain higher reward values and landmark coverage than other baseline algorithms, showing better training performance. Figure 10 shows the rendering of the agent training in cooperative navigation environment after the CMCL algorithm has been trained for 12,500 episodes. From the rendering, it can be seen that the agents can successfully approach and cover landmarks in the environment.



(**a**) (**b**)

Figure 8. Representation diagram of agents in cooperative environment. (**a**) Average reward in cooperative environment; (**b**) landmark cover rate in cooperative environment.

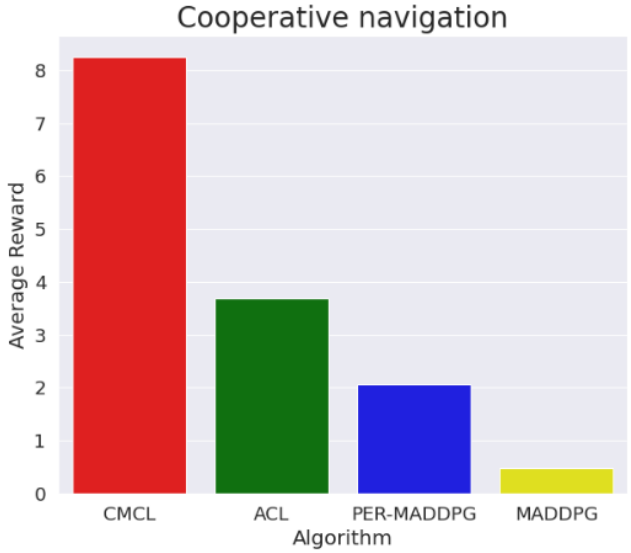


Figure 9. Bar chart of average reward value in cooperative environment.

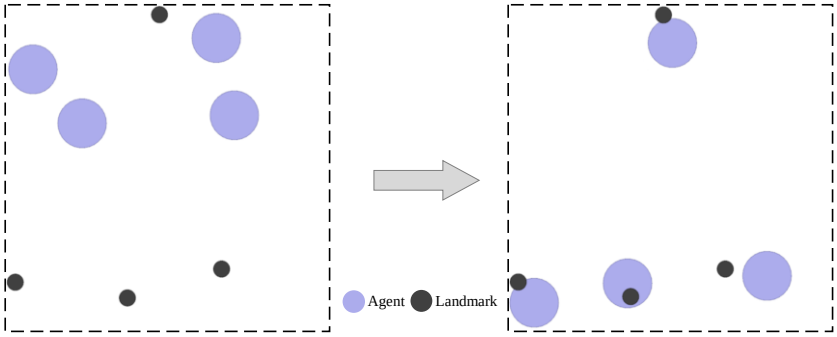


Figure 10. Diagram of the training effect of CMCL algorithm in cooperative environment.

5.1.2. Competition Experiment

In a cooperative training environment, agents share the observed value of the environment to maximize the total reward value, but in a multi-agent competition task, as training progresses, the policies of their opponents are constantly improved, resulting in the continuous fluctuation of the cumulative reward value. In addition to cooperating with other agents, the agent also needs to make policy corrections for the opponent's policy.

The multi-agent competition experiment uses the predator-prey experiment in the MPE environment. On a two-dimensional plane with side length 1, m predators and n prey are randomly generated, as well as three randomly generated obstacles, whose area is relatively large, which can prevent the intelligent body from observing and moving. The goal of predators is to capture prey as quickly as possible through team cooperation. During this process, the predators and the prey move randomly, and the prey move twice as fast as the predators. During the predation process, all predators form a team to hunt down the prey, and the capture is considered successful when the distance between the predator and the prey is less than the pursuit radius.

To construct the sparse reward scene of the predator-prey environment, the dense reward function set according to the distance between predator and prey is canceled. Therefore, the reward value obtained by the predator agent at each time step consists of two parts: 1. When the predator encounters the prey, it will receive a positive reward value, that is, the capture reward value D_1; 2. To prevent agents from escaping the boundary, when agent hits the wall, it will receive a negative reward value, that is, the collision reward value D_2.

The capture reward is as follows:

$$D_1 = \begin{cases} +5, if\ captured \\ 0, if\ not\ captured \end{cases} \tag{23}$$

This represents that when the predator captures the prey, it gets a positive large reward value, while the prey gets a large negative reward value.

The collision boundary rewards are as follows:

$$D_2 = \begin{cases} -1, if\ collided \\ 0, if\ not\ collided \end{cases} \tag{24}$$

This represents that the predator and prey get a negative reward when they collide with the boundary.

As shown in Figure 11, the CMCL, ACL, PER-MADDPG, and MADDPG algorithms are used to control the movements of predators and prey, respectively. To prevent the agent from spinning in place or performing meaningless exploration, the episode duration is set to 30 steps, which means the agent finishes the exploration after 30 steps and initializes the environment to restart the exploration.

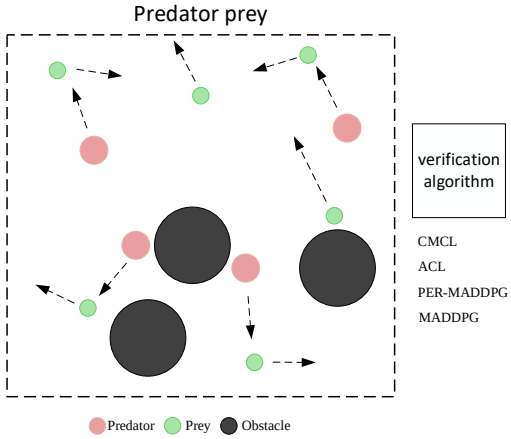
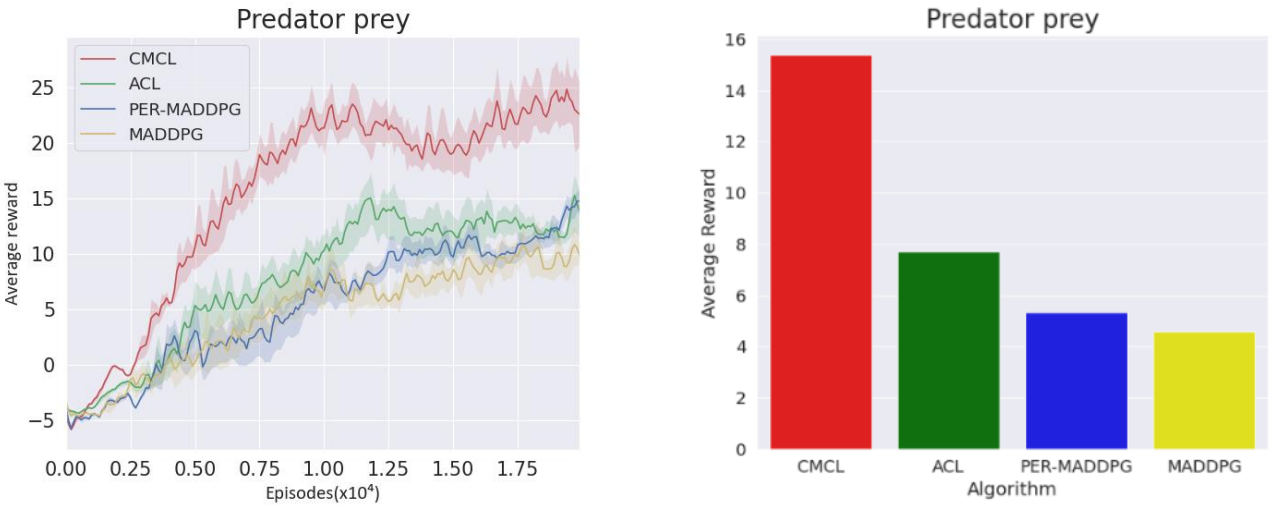


Figure 11. Schematic diagram of adversarial environment.

As shown in Figure 12, the predator agents are controlled by the CMCL, ACL, PER-MADDPG, and MADDPG algorithms respectively, and the prey agents are controlled by the MADDPG algorithm. The bar chart and the error band chart of the average reward value obtained after 20,000 episodes of training indicates that the average reward value in the bar chart is the quotient of the total reward value obtained during the whole training of the four algorithms and the number of episodes. As can be seen in the figure, as training progresses, predator agents controlled by the four algorithms gradually learn the cooperative hunting policy, which tends to stabilize after 10,000 episodes. Throughout the training process, the average reward value of the CMCL algorithm is generally higher than that of other baseline algorithms and is significantly higher than that of the other three algorithms after 10,000 episodes.

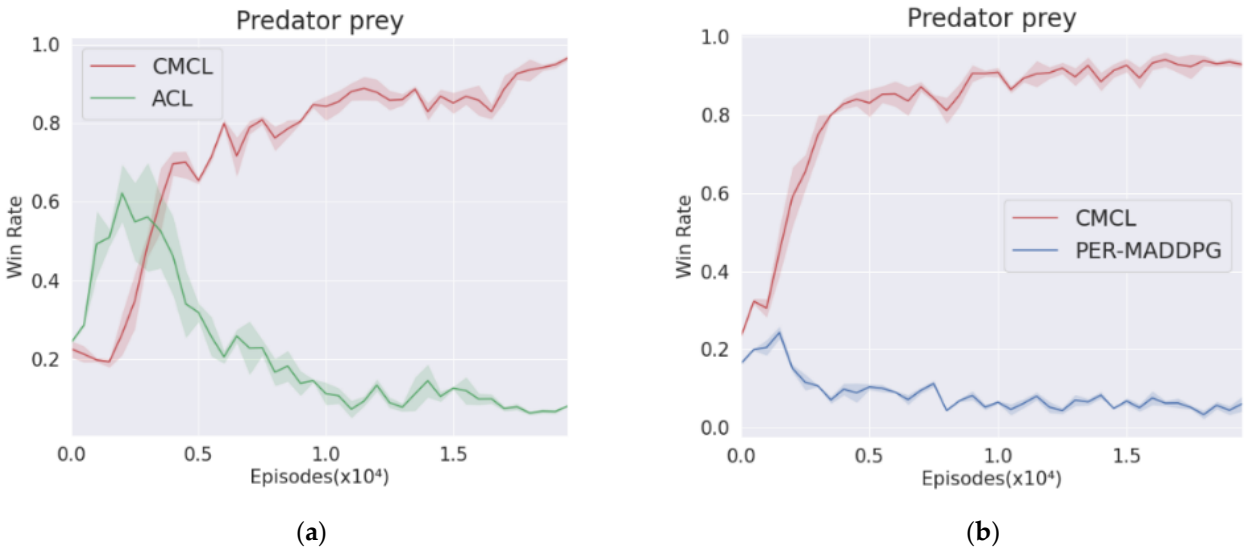


(**a**) (**b**)

Figure 12. Representation diaram of agents in cooperative environment. (**a**) Episode reward achieved in adversarial environment; (**b**) the average reward obtained by the four algorithms in the adversarial environment.

Figure 13a shows the win rate charts obtained by both agents in each round under the condition that the predator agents adopt the CMCL algorithm and the prey agents adopt the ACL algorithm. Figure 13b shows the win rate charts obtained by both agents in each round under the condition that the predator agent adopts the CMCL algorithm and the prey agent adopts the PER-MADDPG algorithm. It can be seen from the figure that when the predator agents controlled by the CMCL algorithm fight against the prey agents

controlled by the ACL algorithm, the two sides won and lost in the early stage. However, after a certain training period (5000 rounds), the predator agents controlled by the CMCL algorithm gain a significant advantage. Predator agents controlled by the CMCL algorithm can gain obvious advantages in a short period of time against the prey agents controlled by the PER-MADDPG algorithm, and the winning rate is above 0.85.



(**a**) (**b**)

Figure 13. The win rate of predator and prey using two algorithms respectively in the adversarial environment. (**a**) CMCL vs. ACL; (**b**) CMCL vs. PER-MADDPG.

Figure 14 shows the training effect diagram of the CMCL algorithm obtained after 10,000 episodes of training in a competitive environment. It can be seen from the effect diagram that the predator agent can learn the batch-hunting policy, that is, to round up the prey agents in two batches by rational use of terrain obstacles. It can be seen that the CMCL algorithm can achieve better training performance than other baseline algorithms in the multi-agent competitive environment.

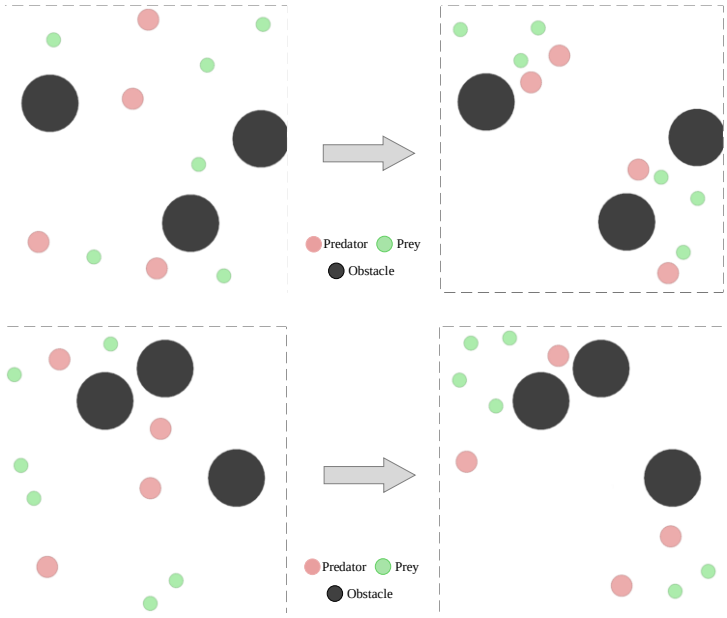


Figure 14. Training effect diagram of CMCL algorithm in competitive environment.

6. Discussion

On the basis of the analysis of the above two experimental environments, the overall performance of our proposed CMCL algorithm is better than that of the other three baseline algorithms, and the following experimental results can be obtained.

In the cooperative environment, the average reward value and landmark coverage of the CMCL algorithm are better than those of the ACL, PER-MADDPG and MADDPG algorithms. Combined with the screenshots of the actual performance of the agents in the experimental simulation environment, CMCL algorithm training in the cooperative environment can be performed. The agents can learn to execute policies dispersedly and cooperatively cover landmarks, avoiding collisions between agents or between agents and the wall.

In the competitive environment experiment, the average reward value of the CMCL algorithm is better than those of the ACL, PER-MADDPG and MADDPG algorithms, and when the predator agent controlled by the CMCL algorithm is confronted with the prey agent controlled by the ACL algorithm and the PER-MADDPG algorithm, after a period of training, a good win rate can be obtained. Combined with the actual performance screenshots of the agents in the experimental simulation environment, it can be concluded that the predator agents trained by CMCL algorithm in the competitive environment can learn to cooperate to surround the prey agents and group the prey agents to carry out the hunting strategy, and avoid the collision between agents or between agents and walls.

The current CMCL algorithm can achieve good training performance in the sparse reward value environment, but there are still two limitations:

1. The dimension explosion problem. A large number of agents in the reinforcement learning environment due to the excessively large state space and the action space, it is easy for the algorithm to fail to converge due to the explosion of dimensions.
2. The problem of reliability distribution. When multiple agents are trained in a reinforcement learning environment, the effective exploration of the environment by the agents can easily be affected due to the uneven distribution of reward functions, especially when multiple players are trained. This problem is more obvious.

7. Conclusions

To solve the problem that the training efficiency of the automatic curriculum reinforcement learning algorithm is not high in the scenario of sparse reward value, this paper adds a curiosity module on the basis of automatic curriculum learning, and uses the curiosity reward value and the temporal-difference error as the reference standard for curriculum sorting. The ICM module is used to evaluate the priority of curiosity, the curriculum factor is designed to control the selection of curriculum difficulty, and an automatic curriculum reinforcement learning algorithm based on the curiosity module is proposed, and the availability and superiority of the algorithm in sparse reward scenarios are verified by simulation experiments in cooperative and competitive environments. With the increase in the number of agents in multi-agent reinforcement learning, the input nodes of the neural network and the complexity of the neural network grow linearly, which can easily cause the problem of dimension explosion in the training process, which makes the algorithm difficult to converge. Methods that can be adopted include compression of state space and share parameters between agents. In the future, based on automatic curriculum reinforcement learning, further research will be conducted on how to reduce the time complexity of multi-agent reinforcement learning training under large-scale number conditions.

The main abbreviations are listed in Table 2.

Table 2. Main abbreviations.

Abbreviation	Explanation
RL	Reinforcement Learning
ACL	Automatic Curriculum Learning
DL	Deep Learning
MADDPG	Multi-Agent Deep Deterministic Policy Gradient
PER	Prioritized Experience Replay
ICM	Intrinsic Curiosity Module
MPE	Multi-Agent Particle Environment
CMCL	Curiosity Module-based Curriculum Learning
ICML	International Conference on Machine Learning
AC	Actor-Critic
KP	K-Fold Priority
CP	Curiosity Priority
CI	Curriculum Index
CTDE	Centralized Training and Distributed Execution

Author Contributions: Methodology, Z.L.; Software Z.L., J.L. and X.C.; Validation, Z.L., J.L. and X.C.; writing—original draft, Z.L., L.C. and J.W.; writing—review and editing, X.C. and J.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was partially supported by the National Natural Science Foundation of China (No. 61806221).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data used to support the findings of this study are available from the website https://github.com/openai/multiagent-particle-envs (accessed on 13 May 2022).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*; MIT Press: Cambridge, MA, USA, 2018.
2. Mnih, V.; Badia, A.P.; Mirza, M. Asynchronous methods for deep reinforcement learning. In Proceedings of the International Conference on Machine Learning (ICML), New York, NY, USA, 18–20 December 2016; pp. 1928–1937.
3. Foglino, F.; Christakou, C.C.; Gutierrez, R.L. Curriculum learning for cumulative return maximization. *arXiv* **2019**, arXiv:1906.06178.
4. Fang, M.; Zhou, T.; Du, Y. Curriculum-guided hindsight experience replay. *Adv. Neu. Infor. Pro. Sys.* **2019**, *19*, 12602–12613.
5. Gu, S.; Holly, E.; Lillicrap, T. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May–3 June 2017; pp. 3389–3396.
6. Silver, D.; Huang, A.; Maddison, C.J. Mastering the game of Go with deep neural networks and tree search. *Nature* **2016**, *529*, 484–489. [CrossRef] [PubMed]
7. Lecun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436–440. [CrossRef] [PubMed]
8. Singh, A.; Jain, T.; Sukhbaatar, S. Individualized controlled continuous communication model for multiagent cooperative and competitive tasks. In Proceedings of the International Conference on Learning Representations (ICLR), New Orleans, LA, USA, 6–9 May 2019; pp. 154–160.
9. Yang, Y.; Luo, R.; Li, M. Mean field multi-agent reinforcement learning. In Proceedings of the International Conference on Machine Learning (ICML), Stockholm, Sweden, 10–15 July 2018; pp. 5571–5580.
10. Liu, Q.; Cui, C.; Fan, Q. Self-Adaptive Constrained Multi-Objective Differential Evolution Algorithm Based on the State–Action–Reward–State–Action Method. *Mathematics* **2022**, *10*, 813. [CrossRef]
11. Bengio, Y.; Louradour, J.; Collobert, R. Curriculum learning. In Proceedings of the 26th Annual International Conference on Machine Learning (ICML), Quebec, MT, Canada, 14–18 June 2009; pp. 41–48.
12. Xue, H.; Hein, B.; Bakr, M. Using Deep Reinforcement Learning with Automatic Curriculum Learning for Mapless Navigation in Intralogistics. *Appl. Sci.* **2022**, *12*, 3153. [CrossRef]
13. Portelas, R.; Colas, C.; Weng, L. Automatic curriculum learning for deep rl: A short survey. *arXiv* **2020**, arXiv:2003.04664.

14. Florensa, C.; Held, D.; Geng, X. Automatic goal generation for reinforcement learning agents. In Proceedings of the International Conference on Machine Learning, Stockholm, Sweden, 10–15 July 2018; pp. 1515–1528.
15. Ren, Z.; Dong, D.; Li, H. Self-paced prioritized curriculum learning with coverage penalty in deep reinforcement learning. *IEEE Trans. Neu. Net. Learn. Syst.* **2018**, *29*, 2216–2226. [CrossRef]
16. Chen, J.; Zhang, Y.; Xu, Y. Variational Automatic Curriculum Learning for Sparse-Reward Cooperative Multi-Agent Problems. *Adv. Neu. Infor. Pro. Syst.* **2021**, *34*, 102–116.
17. Haibo, H.; Edwardo, A.G. Learning from imbalanced data. *IEEE Trans. Know. Data. Eng.* **2008**, *9*, 1263–1284. [CrossRef]
18. Geoffrey, E.; Hinton. To recognize shapes, first learn to generate images. *Pro. Bra. Res.* **2007**, *165*, 535–547.
19. Pathak, D.; Agrawal, P.; Efros, A.A. Curiosity-driven exploration by self-supervised prediction. In Proceedings of the International Conference on Machine Learning (ICML), Sydney, NSW, Australia, 6–11 August 2017; pp. 2778–2787.
20. Gruber, M.J.; Bernard, D.G.; Charan, R. States of curiosity modulate hippocampus-dependent learning via the dopaminergic circuit. *Neuron* **2014**, *84*, 486–496. [CrossRef]
21. Zhang, H.; Qu, C.; Zhang, J. Self-Adaptive Priority Correction for Prioritized Experience Replay. *Appl. Sci.* **2020**, *10*, 6925. [CrossRef]
22. Cao, X.; Wan, H.; Lin, Y. High-value prioritized experience replay for off-policy reinforcement learning. In Proceedings of the 2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI), Portland, OR, USA, 4–6 November 2019; pp. 1510–1514.
23. Li, Y. Deep reinforcement learning: An overview. *arXiv* **2017**, arXiv:1701.07274.
24. Lv, K.; Pei, X.; Chen, C. A Safe and Efficient Lane Change Decision-Making Strategy of Autonomous Driving Based on Deep Reinforcement Learning. *Mathematics.* **2022**, *10*, 1551. [CrossRef]
25. Grondman, I.; Busoniu, L.; Lopes, G.A.D. A survey of actor-critic reinforcement learning: Standard and natural policy gradients. *IEEE Trans. Syst. Man. Cyber.* **2012**, *42*, 1291–1307. [CrossRef]
26. Lowe, R.; Wu, Y.I.; Tamar, A. Multi-agent actor-critic for mixed cooperative-competitive environments. *Adv. Neu. Infor. Pro. Syst.* **2017**, *30*, 133–160.
27. Lei, W.; Wen, H.; Wu, J. MADDPG-based security situational awareness for smart grid with intelligent edge. *Appl. Sci.* **2021**, *11*, 3101. [CrossRef]
28. Wang, X.; Chen, Y.; Zhu, W. A survey on curriculum learning. *IEEE Trans. Pat. Ana. Mac. Intel.* **2021**, *37*, 362–386. [CrossRef]
29. Parker-Holder, J.; Rajan, R.; Song, X. Automated Reinforcement Learning (AutoRL): A Survey and Open Problems. *arXiv* **2022**, arXiv:2201.03916. [CrossRef]
30. Kumar, M.; Packer, B.; Koller, D. Self-paced learning for latent variable models. *Adv. Neu. Infor. Pro. Syst.* **2010**, *23*, 154–160.