# Phased Continuous Exploration Method for Cooperative Multi-Agent Reinforcement Learning

Jie Kang[1], Yaqing Hou[1], Yifeng Zeng[2],
Yongchao Chen[3], Xiangrong Tong[4] Xin Xu[5], and Qiang Zhang [1]
[1]Dalian University of Technology, Dalian, China
[2]Northumbria University, Newcastle, UK
[3]The Institute of Effectiveness Evaluation of Flying Vehicle, Beijing, China
[4]Yantai University, Yantai, China
[5]Wuhan University of Science and Technology, Wuhan, China
kangj@mail.dlut.edu.cn, houyq@dlut.edu.cn, yifeng.zeng@northumbria.ac.uk,
ycchentg@126.com, xr_tong@163.com, xuxin0336@163.com, zhangq@dlut.edu.cn

***Abstract*—In multi-agent reinforcement learning, achieving effective exploration for agents remains challenging due to the non-stationarity of the environment and discrepancies between local and global information. In this paper, we propose a curiosity-driven phased continuous exploration method, termed PCE. We recognize that agents in different learning phases possess distinct knowledge and policies, allowing them to learn diverse knowledge and experiences from the same states. Therefore, we divide the training process of agents into different phases, employing a curiosity-driven method to explore independently within each phase. Simultaneously, addressing the characteristic of inconsistent local and global information in multi-agent systems, we strike a balance between exploration from local and global perspectives. Finally, we evaluate the proposed method in the popular multi-agent test task, StarCraft II. The results indicate that the method excels in enhancing the exploration capabilities of agents.**

***Index Terms*—Multi-agent systems, multi-agent reinforcement learning (MARL), exploration.**

## I. Introduction

In recent years, with the continuous development of deep reinforcement learning, significant progress has been made in multi-agent reinforcement learning (MARL). Indeed, MARL shows great promise in addressing various real-world problems involving multiple agents, demonstrating notable performance in areas such as traffic control [1], robot coordination [2], and game AI [3]. In MARL, multiple agents learn policies within the same environment, where the dynamic changes in the environment are influenced by all agents. Early MARL research tended to directly apply single-agent reinforcement learning algorithms to multi-agent scenarios [4], with each agent treating others as part of the environment and learning independently. However, this approach suffers from apparent non-stationarity issues.

To address these issues, the paradigm of centralized training with decentralized execution (CTDE) [5] has become popular in recent years. This paradigm allows agents to leverage additional global information during training but requires only local information during execution. For example, algorithms such as multi-agent deep deterministic policy gradient (MADDPG) [6], monotonic value function factorization (QMIX) [7], and multi-agent proximal policy optimization (MAPPO) [8] follow the CTDE paradigm and exhibit excellent performance. However, despite their impressive performance, these methods all employ classic noise-based exploration strategies, which tend to be slow and may lead to inadequate exploration and suboptimal performance [9].

In order to achieve effective exploration in MARL, some recent relevant works have been proposed. For instance, MAVEN [10] introduces a hierarchical control strategy by encouraging exploration by adjusting the agent's behavior on a shared latent variable controlled by a hierarchical policy. Wang [11] measures the influence of an agent's behavior on the actions of other agents and promotes coordinated exploration by encouraging agents to access critical states. In addition, curiosity is an exploration mechanism based on intrinsic motivation, which usually uses the prediction error of the future state to measure novelty and serves as a reward signal to encourage the agent to explore. Its essence is to measure the agent's familiarity with the state and encourage the agent to explore unfamiliar states. For example, EMC [12] adopts the prediction error of individual Q-values as an intrinsic reward for coordinated exploration and utilizes episodic memory techniques to leverage explored information to facilitate policy training. Although these works achieve promising performance, they evaluate exploration metrics by considering the training process as an indivisible whole. However, it is important to note that the agent is in a continuous learning process and has different policy networks at different learning phases. These policy networks may lead to different behavioral decisions when facing the same scenario, resulting in distinct outcomes. Therefore, the knowledge and experience that an agent can vary depending on the phase in which it learns from a given state.

For instance, when a one-year-old human faces a book, they

This work was supported in part by the National Key Research and Development Program of China (No. 2021ZD0112400), the National Natural Science Foundation of China under Grant 62372081, the Young Elite Scientists Sponsorship Program by CAST under Grant 2022QNRC001, the NSFC-Liaoning Province United Foundation under Grant U1908214, the 111 Project, No.D23006, and the Fundamental Research Funds for the Central Universities under grant DUT21TD107, DUT22ZD214.

979-8-3503-5409-6/24/$31.00 ©2024 IEEE
DOI 10.1109/CAI59869.2024.00196

only learn the behavior of flipping through pages. At the age of five, facing the same book, the individual may search for interesting pictures, learning the behavior of looking for captivating images. By the age of fifteen, facing the same book, the person can read the text, acquiring knowledge from its contents. Therefore, during the learning process of the agent, even though certain states have been explored previously, due to the limited cognitive level of the agent at that time, it did not fully explore all the knowledge in those states. With the continuous updating of the policy network, the agent now has the ability to acquire new knowledge from past learned states. Thus, for the same state in a multi-agent environment, we should also give it certain exploration at different phases to encourage the agents to learn new knowledge from it. Specifically, when we evaluate the novelty of a state, we will not only consider the agent's familiarity with it but also consider the time dimension, that is, whether the state has been fully explored recently.

Based on the above considerations, this paper proposes a curiosity-driven phased continuous exploration method to encourage the agent to fully explore the environment and promote the learning of the agent's policy. First, we divide the training process into different phases and calculate the state novelty based on the agent's familiarity with the state features within each phase respectively. Then, we balance the exploration of both local and global perspectives in response to the characteristic of inconsistency between local and global information in multi-agent systems. Finally, we evaluate it in the popular multi-agent testing platform StarCraft II micromanagement benchmark tasks. The results are compared with other traditional methods and demonstrate the effectiveness of the proposed method.

## II. Background

In this section, we first introduce Markov decision processes in multi-agent reinforcement learning. Then, we introduce Random Network Distillation, an exploration method for reinforcement learning based on prediction errors. Finally, we also discuss the baseline algorithms used in this paper.

### A. Problem Formulation

This paper investigates fully cooperative multi-agent reinforcement learning tasks and formulates them as decentralized partially observable markov decision processes (Dec-POMDP). A Dec-POMDP can be defined as a tuple $G = \langle S, A, P, r, \Omega, O, n, \gamma \rangle$. Here, $S$ is the finite state space of the environment. For each time step $t$, every agent $i \in N \equiv \{1, \ldots, n\}$ chooses an action $a^i \in A$ which forms the joint action $\mathbf{a} \in \mathbf{A} \equiv A^n$. $P(s'|s, \mathbf{a}) : S \times \mathbf{A} \times S \to [0, 1]$ is the state transition function. $r(s; \mathbf{a}) : S \times \mathbf{A} \to \mathbb{R}$ is the reward function shared by all agents and $\gamma \in [0, 1)$ is the discount factor. In the present work, *partially observable* settings are considered, where an agent only has access to an observation $o_i \in \Omega$ drawn according to observation function $O(s, i) : S \times N \to \Omega$, not its true state $s^i$. The action-observation history for an agent $i$ is $\tau^i \in T \equiv (\Omega \times A)^*$ on which it can condition its policy $\pi^i(a^i|\tau^i) : T \times A \to [0, 1]$. The policies of all agents form a joint policy $\boldsymbol{\pi} = (\pi^1, \cdots, \pi^N)$, and the goal is to maximize the expected total reward:

$$\mathcal{J}(\boldsymbol{\pi}) \triangleq \mathbb{E}_{\rho_0, \boldsymbol{\pi}, P}[\sum_{t=0}^{\infty} \gamma^t r_t]. \tag{1}$$

where $\rho_0$ represents the initial state distribution, and $s_0 \sim \rho_0(s_0), s_{t+1} \sim P(\cdot|s_t, a_t), \mathbf{a}_t \sim \boldsymbol{\pi}(s_t)$.

### B. Random Network Distillation

Random Network Distillation (RND) [13] is a curiosity-driven method based on prediction error used in single-agent reinforcement learning to enhance the exploration efficiency of algorithms. The core idea of RND is to initialize two neural networks to generate state features: one is the target network $f$, and the other is the predictor network $\widehat{f}$. After random initialization, $f$ remains fixed, while $\widehat{f}$ learns to reduce the prediction error in comparison to the target network. Consequently, states that are frequently visited have small prediction errors, while states that are infrequently visited exhibit larger prediction errors. The prediction error values are then employed as intrinsic rewards for agents to enhance the exploration capabilities of the algorithm. RND, as a simple and effective exploration method, has made significant contributions to the research in reinforcement learning algorithms. In this paper, we propose enhancements to the RND algorithm and extend its application to the domain of multi-agent reinforcement learning settings.

### C. Multi-Agent Proximal Policy Optimization

Multi-Agent Proximal Policy Optimization (MAPPO) [8] is a multi-agent reinforcement learning proximal policy optimization algorithm designed to optimize the policies of agents in a multi-agent environment. Essentially an extension of the PPO [14] algorithm into the realm of multi-agent scenarios, MAPPO adheres to the paradigm of centralized training with decentralized execution. It achieves this by utilizing a global value function to guide the training of individual PPO agents. MAPPO follows common practices found in PPO implementations, including Generalized Advantage Estimation (GAE), observation normalization, gradient clipping, and others. Additionally, MAPPO introduces five crucial implementation details, namely value normalization, value function inputs, training data usage, policy and value clipping, and death masking. Moreover, MAPPO conducts a limited grid search on certain hyperparameters, ensuring its robust performance in complex multi-agent tasks.

## III. Methodology

In the following, we first introduce how to divide the phases, and then give the complete intrinsic reward calculation method. Finally, we summarize the algorithm flow.

### A. Exploring in Phases

In the exploration process of reinforcement learning, the states learned in the past still have new exploration value for the agent at the current phase. This is because the agent is

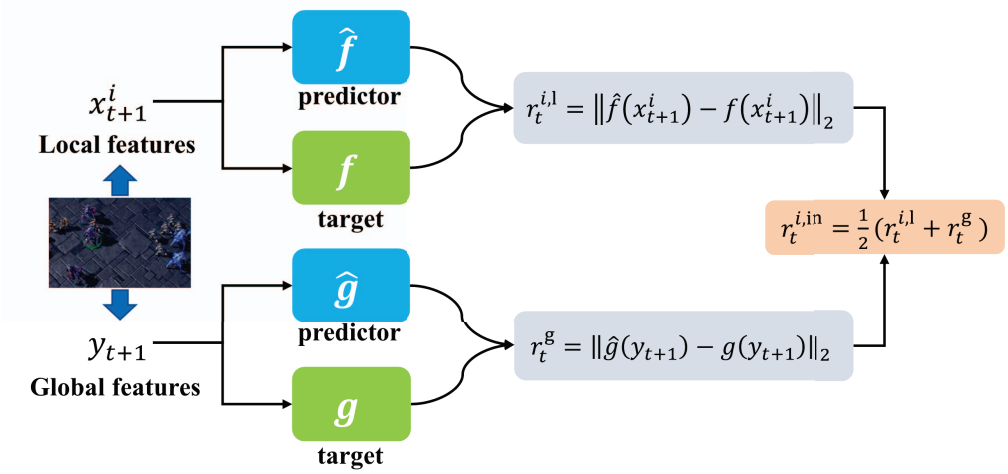



Fig. 1: Overall architecture of our method. Utilizing local observations and global states from the environment to generate local features and global features, respectively. Subsequently, local and global random network distillation models are employed separately to calculate novelty intrinsic rewards, and then combined to compute the final intrinsic reward.

constantly learning, and the policy network at each phase is different, that is, it has different knowledge and skills. Even when facing the same state, agents may make different decisions, allowing the exploration of diverse knowledge and experiences. Therefore, for states that have been familiar for a long time, agents should re-explore them at different phases, attempting to learn new experiential knowledge. Specifically, when we evaluate the novelty of a state, we consider not only the agent's familiarity with it but also whether the state has been sufficiently explored recently.

To achieve phased continuous exploration, we draw inspiration from the design principles of Random Network Distillation (RND) and incorporate temporal features into the familiarity measurement calculation. Specifically, we annotate the training time for each state data, referred to as *Phase*. We divide the entire MARL training process into $M$ phases, with each phase $p$ labeled by the set of numbers $\{1, 2, \cdots, M\}$. The formula for calculating the phase number is given by:

$$p = \left\lfloor \frac{k \cdot M}{K} \right\rfloor + 1 \tag{2}$$

where $K$ is the total number of training episodes, $k$ is the current episode index, and $k \in \{0, 1, \cdots, K-1\}$. We concatenate the one-hot encoding of the phase number with every state data generated during that time period as the input to the RND, which is used to compute the novelty and train the predictor network for the RND. As a result, similar states within the same phase remain deemed similar after the RND assessment. On the other hand, similar states across different phases have distinct phase number encodings (e.g., one might be *10000*, and another *00010*), introducing novelty errors. These novelty errors eventually serve as intrinsic motivation to encourage exploration by the agent. Consequently, due to the inherent differences in phase encodings, states that were thoroughly explored in previous phases continued to be encouraged for exploration in new phases, facilitating the acquisition of new knowledge and experiences.

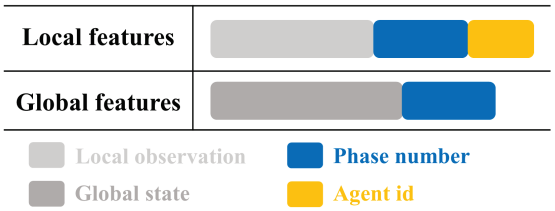



Fig. 2: Local and global RND inputs.

### B. Balancing Local and Global Novelty

Given the partially observable nature of multi-agent systems, we need to measure novelty from both a local and global perspective. To balance these aspects, we calculate novelty from a holistic perspective using the global state and, simultaneously, from an individual standpoint using the local observations of the agent. We then use the mean of these measurements as the final result for generating intrinsic reward signals, encouraging agents to explore. Equipping each agent with a dedicated RND mechanism for calculating local novelty is complex and lacks scalability. Therefore, we optimize this process by parameter sharing, where all agents share a common local RND. Each agent's local observations, along with the agent's ID code for differentiation, are jointly used as input.

In summary, the feature information for our method's input is illustrated in Figure 2, comprising both local and global features. The local features consist of local observation, phase number encoding, and agent ID encoding, used for calculating local novelty. Global features comprise global state and phase

number encoding, used for calculating global novelty. Additionally, the overall framework of our approach, as illustrated in Figure 1, involves two RND models that independently compute local and global novelties. The final novelty intrinsic reward is obtained by calculating their mean and is subsequently employed as intrinsic motivation to encourage agents for phased continuous exploration. Therefore, the specific calculation formula of the intrinsic reward $r_t^{i,\text{in}}$ of agent $i$ at time $t$ is as follows:

$$r_t^{i,\text{l}} = ||\widehat{f}(x_{t+1}^i) - f(x_{t+1}^i)||_2 \tag{3}$$

$$r_t^{\text{g}} = ||\widehat{g}(y_{t+1}) - g(y_{t+1})||_2 \tag{4}$$

$$r_t^{i,\text{in}} = \frac{1}{2}(r_t^{i,\text{l}} + r_t^{\text{g}}) \tag{5}$$

Among them, $x_{t+1}^i$ represents the local feature input of agent $i$, $y_{t+1}$ represents the global feature input, $r_{t+1}^{i,\text{l}}$ represents the local novelty intrinsic reward of agent $i$, and $r_{t+1}^{\text{g}}$ represents the global novelty intrinsic reward. Then, the intrinsic reward is combined with the extrinsic reward $r_t^{i,\text{ex}}$ provided by the environment to construct the overall reward. Therefore, the total reward $r_t^{i,\text{total}}$ of agent $i$ at time $t$ is expressed as follows:

$$r_t^{i,\text{total}} = r_t^{i,\text{ex}} + \alpha \times r_t^{i,\text{in}} \tag{6}$$

where $\alpha$ is a hyperparameter that balances extrinsic reward and intrinsic reward.

### C. Algorithmic Summary

The details of our algorithm are discussed in Algorithm 1. Initially, we initialize the parameters of the local RND target network and predictor network, the global RND target network and predictor network, and the policy network. The parameters of the target networks are fixed after initialization (line 2). Subsequently, we conduct training for $K$ episodes, as shown in lines 3 to 20. In each episode, the agents first collect a set of trajectories through the joint policy $\pi$ (line 5). We then calculate the phase number $p$ of the current episode and obtain the global features $y_{t+1}$ by concatenating $p$ and the global state $s_{t+1}$ (lines 6-7). Next, we separately concatenate the local observations, phase number, and agent ID for each agent to obtain their local features $x_{t+1}$, and compute their intrinsic novelty rewards $r_t^i$ individually. The transitions are then stored in a buffer (lines 8-13). Following this, a batch of data is randomly sampled for training, and this data is used to update the policy networks of the agents (lines 15-16). Finally, the loss values for both local RND and global RND are computed using the sampled data, and their predictor networks are updated (lines 17-19). This process is repeated throughout the entire training cycle until completion.

## IV. Experiments

In this section, we first introduce the environment setup of the experiment. Subsequently, we present experimental results and analysis to verify the effectiveness of the proposed method.

**Algorithm 1** Phased Continuous Exploration (PCE)

1: **Input:** batch size $B$, number of agents $n$, episodes $K$, steps per episode $T$, phases $M$.
2: **Initialize:** Local RND target $\theta_f$ and predictor $\theta_{\widehat{f}}$, Global RND target $\theta_g$ and predictor $\theta_{\widehat{g}}$, Policy $\phi$, Replay buffer $\mathcal{B}$
3: **for** $k = 0, 1, \ldots, K-1$ **do**
4: // COLLECT TRANSITIONS
5: Collect a set of trajectories by running the joint policy $\boldsymbol{\pi}_\phi = (\pi_{\phi^1}^1, \ldots, \pi_{\phi^n}^n)$.
6: Compute phase number $p = \lfloor \frac{k \cdot M}{K} \rfloor + 1$.
7: Concatenate global features $y_{t+1}$.
8: **for** each agent $i$ **do**
9: Concatenate local features $x_{t+1}^i$.
10: Calculate intrinsic reward according to formula 5.
11: Calculate total reward according to formula 6.
12: Push $\{(o_t^i, s_t, a_t^i, o_{t+1}^i, s_{t+1}, x_{t+1}^i, y_{t+1}, r_t^{i,\text{total}})\}$ into $\mathcal{B}$.
13: **end for**
14: // UPDATE NETWORKS
15: Sample a random minibatch from $\mathcal{B}$.
16: Update policy $\phi$ with sampled data.
17: Compute $loss_{\text{local}} = ||\widehat{f}(x_{t+1}^i) - f(x_{t+1}^i)||_2$.
18: Compute $loss_{\text{global}} = ||\widehat{g}(y_{t+1}) - g(y_{t+1})||_2$.
19: Update local RND predictor $\theta_{\widehat{f}}$ and global RND predictor $\theta_{\widehat{g}}$.
20: **end for**

### A. Environment Setup

To evaluate our method, we considered the widely used StarCraft II [15] environment. The StarCraft II Multi-Agent Challenge (SMAC) platform is developed based on the StarCraft II game and serves as an open platform for studying Multi-Agent Reinforcement Learning (MARL). It provides a challenging multi-agent environment designed to evaluate the performance of agents in solving complex cooperative tasks. As illustrated in Figure 3, our experiments were conducted on two difficult maps and one extremely challenging map, with specific details outlined in Table I. These maps are carefully designed, requiring the learning of one or more micro-management techniques to defeat opponents. Each map involves a confrontation between two armies, with variations in initial positions, quantity, and unit types for each army. In each map, agents controlled by our algorithm engage in intense battles with enemy agents controlled by the built-in game AI. The end condition of the game is when all agents from any side are annihilated, and victory is achieved by eliminating all enemy agents.

### B. Overall Results

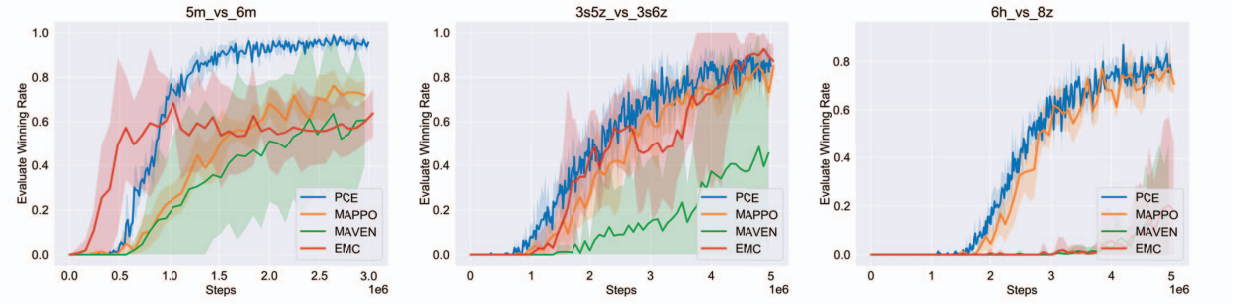
PCE is an exploration technique that can be flexibly applied to different MARL algorithms. To validate the adaptability and effectiveness of PCE, we apply it to the MAPPO [8] algorithm and test it on three SMAC maps: 5m_vs_6m (hard),

(a) 5m_vs_6m(hard) (b) 3s5z_vs_3s6z(super hard) (c) 6h_vs_8z(super hard)

Fig. 3: (a-c) The maps we considered in the StarCraft II Multi-Agent Challenge: 5m_vs_6m(hard), 3s5z_vs_3s6z(super hard), and 6h_vs_8z(super hard).

TABLE I: The SMAC maps considered in our experiments.

| Name | Ally Units | Enemy Units | Type |
|---|---|---|---|
| 5m_vs_6m | 5 Marines | 6 Marines | homogeneous & asymmetric |
| 3s5z_vs_3s6z | 3 Stalkers & 5 Zealots | 3 Stalkers & 6 Zealots | heterogeneous & asymmetric |
| 6h_vs_8z | 6 Hydralisks | 8 Zealots | micro-trick: focus fire |




(a) 5m_vs_6m (b) 3s5z_vs_3s6z (c) 6h_vs_8z

Fig. 4: Performance comparison between PCE, MAPPO, MAVEN, and EMC on three SMAC tasks; The x-axis is the number of environmental steps during training; The y-axis is the evaluation of winning rate during testing.

3s5z_vs_3s6z (super hard), and 6h_vs_8z (super hard). We adopt MAVEN, EMC, and the original MAPPO as comparison methods, which are related to our method and have advanced performance. Their descriptions are as follows:

1) MAVEN [10]: the method proposes to restrict the agent's behavior to a shared latent variable controlled by a hierarchical policy to improve exploration.
2) EMC [12]: the method enables curiosity-driven exploration by predicting individual Q-values and facilitates policy training through episodic memory technology.
3) MAPPO [8]: the method is a variant of the PPO algorithm in MARL using implementation tricks such as generalized advantage estimation and value normalization.

In Figure 4, we present the results for "evaluate winning rate" and "steps". The lines in the figure represent the average evaluation winning rate over five independent runs, and the shaded areas denote the 95% confidence interval. The results indicate that PCE outperforms the baseline MAPPO algorithm on all maps. In 5m_vs_6m, PCE shows the most significant improvement, achieving an average win rate of over 95%, a 23% increase over the baseline, and outperforming other methods in both convergence speed and final win rate. In 3s5z_vs_3s6z and 6h_vs_8z, PCE exhibits a noticeable improvement in convergence speed and a slight advantage in final win rate compared to MAPPO. In addition, the overall performance of PCE is better than that of MAVEN and EMC algorithms. These results demonstrate that PCE enhances the performance of the baseline algorithm in SMAC tasks by improving the exploration capability of the agents.

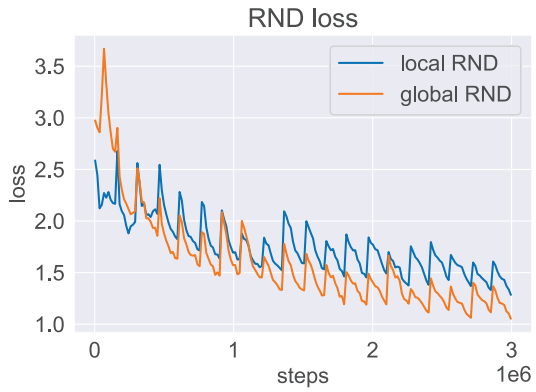



Fig. 5: Local and global RND loss.

### C. Visual Analysis

To illustrate the effect of phased continuous exploration, we conduct a visual analysis of the impact on local and global novelty. Specifically, we analyze the variation curves of the loss functions for local and global RND models. As shown in Figure 5, the curves for local and global variations are quite similar. In terms of the overall trend, their loss function values decrease from high to low. This is because, through continuous exploration and training, the agents become quite familiar with the majority of state features, leading to a gradual reduction in the loss function values for the RND models. On a microscopic level, their loss function values exhibit a periodic pattern of highs and lows. This is attributed to the fact that PCE utilizes phase numbers to label the training process into different phases, allowing agents to independently explore state features within each phase. Consequently, the loss function values within each phase also undergo a process of decreasing from high to low. Through this phased exploration technique, agents can effectively explore and learn from the task environment at different phases, preventing them from falling into suboptimal policies.

## V. Conclusion

In this paper, we present a curiosity-driven phased continuous exploration method to address the exploration deficit in multi-agent systems. We recognize that agents possess distinct knowledge at different learning phases, leading to diverse decisions when confronted with the same state, consequently resulting in varied knowledge acquisition. Therefore, we partition the training process into different phases, employing a curiosity-driven approach for exploration within each phase. Simultaneously, considering the partially observable characteristics of multi-agent systems, we strike a balance between exploration from local and global perspectives. Finally, we evaluate the proposed method for the StarCraft II task, demonstrating its effectiveness in enhancing the agents' exploratory capabilities.

In future work, we will delve into dynamically and judiciously defining different training phases and provide effective solutions. In addition, we will try to extend the proposed method to a wider range of multi-agent task scenarios with practical significance to fully demonstrate its practical application potential, such as robot control, etc.

## References

[1] D. Chen, M. R. Hajidavalloo, Z. Li, K. Chen, Y. Wang, L. Jiang, and Y. Wang, "Deep multi-agent reinforcement learning for highway on-ramp merging in mixed traffic," *IEEE Transactions on Intelligent Transportation Systems*, 2023.

[2] S. Gu, J. G. Kuba, Y. Chen, Y. Du, L. Yang, A. Knoll, and Y. Yang, "Safe multi-agent reinforcement learning for multi-robot control," *Artificial Intelligence*, vol. 319, p. 103905, 2023.

[3] W. Du and S. Ding, "A survey on multi-agent deep reinforcement learning: from the perspective of challenges and applications," *Artificial Intelligence Review*, vol. 54, pp. 3215–3238, 2021.

[4] A. Tampuu, T. Matiisen, D. Kodelja, I. Kuzovkin, K. Korjus, J. Aru, J. Aru, and R. Vicente, "Multiagent cooperation and competition with deep reinforcement learning," *PloS one*, vol. 12, no. 4, p. e0172395, 2017.

[5] L. Kraemer and B. Banerjee, "Multi-agent reinforcement learning as a rehearsal for decentralized planning," *Neurocomputing*, vol. 190, pp. 82–94, 2016.

[6] 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.

[7] T. Rashid, M. Samvelyan, C. Schroeder, G. Farquhar, J. Foerster, and S. Whiteson, "Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning," in *International Conference on Machine Learning*. PMLR, 2018, pp. 4295–4304.

[8] 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.

[9] J. Hao, T. Yang, H. Tang, C. Bai, J. Liu, Z. Meng, P. Liu, and Z. Wang, "Exploration in deep reinforcement learning: From single-agent to multiagent domain," *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

[10] A. Mahajan, T. Rashid, M. Samvelyan, and S. Whiteson, "Maven: Multi-agent variational exploration," *Advances in neural information processing systems*, vol. 32, 2019.

[11] T. Wang, J. Wang, Y. Wu, and C. Zhang, "Influence-based multi-agent exploration," *arXiv preprint arXiv:1910.05512*, 2019.

[12] L. Zheng, J. Chen, J. Wang, J. He, Y. Hu, Y. Chen, C. Fan, Y. Gao, and C. Zhang, "Episodic multi-agent reinforcement learning with curiosity-driven exploration," *Advances in Neural Information Processing Systems*, vol. 34, pp. 3757–3769, 2021.

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

[14] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[15] M. Samvelyan, T. Rashid, C. S. De Witt, G. Farquhar, N. Nardelli, T. G. Rudner, C.-M. Hung, P. H. Torr, J. Foerster, and S. Whiteson, "The starcraft multi-agent challenge," *arXiv preprint arXiv:1902.04043*, 2019.