Solving Sparse Reward Tasks Using Self-Balancing Exploration and Exploitation

*Yan Kong[1], Junfeng Wei[1], Chih-Hsien Hsia[2,3]**

[1] *School of Computer and Software, Nanjing University of Information Science and Technology, China*
[2] *Department of Computer Science and Information Engineering, National Ilan University, Taiwan*
[3] *Department of Business Administration, Chaoyang University of Technology, Taiwan*
kongyan4282@163.com, 460749067@qq.com, hsiach@niu.edu.tw

Abstract

A core challenge in applying deep reinforcement learning (DRL) to real-world tasks is the sparse reward problem, and shaping reward has been one effective method to solve it. However, due to the enormous state space and sparse rewards in the real world, a large number of useless samples may be generated, leading to reduced sample efficiency and potential local optima. To address this issue, this study proposes a self-balancing method of exploration and development to solve the issue of sparse rewards. Firstly, we shape the reward function according to the evaluated progress, to guide the agent's learning of high-reward samples. Secondly, we construct a dual-trajectory exploration network, which provides intrinsic rewards based on the novelty of states and the trajectory difference of sibling agents to encourage the agent to explore and adjust the balance between exploration and exploitation. This method effectively prevents the generation of a large amount of useless training data during the interaction between the agent and the environment, resolves local optimal dilemmas through state novelty, and adjusts the strategy in a timely manner to solve sparse reward tasks. Our method outperforms basic reinforcement learning (RL) and curiosity-driven incentives in these experimental tasks. The self-balancing exploration and exploitation approach in our research provides a new perspective and effective solution for addressing the problem of sparse rewards, thereby advancing the application of DRL in real-world problems and achieving greater success.

Keywords: Deep reinforcement learning, Deep learning, Artificial intelligence, Sparse reward, Exploration and exploitation

1 Introduction

In the current wave of research in artificial intelligence (AI), there is a growing interest in enabling computers to perform tasks that previously relied solely on human intelligence. In this regard, reinforcement learning (RL) [1] has emerged as a unique paradigm. It involves training an agent to observe the states provided by the environment, make optimal decisions, analyze the consequences of its actions through interactions with the environment, and receives rewards accordingly. This progressive learning approach bears resemblance to human cognition. The essence of RL lies in maximizing the accumulated reward obtained by the agent during interactions with the environment while tackling specific tasks. The RL environment can be modeled as a Markov decision process [2], as shown in Figure 1. Specifically, at each time step t, the agent reaches a state s_t , takes an action a_t , receives a reward r_{t+1} from the environment, and iterates until the final task is completed. Since DeepMind introduced deep Q-networks (DQN) in 2013, by combining deep neural networks [3] with RL, deep reinforcement learning (DRL) [4] has revolutionized the field and become one of the widely applied AI algorithms, particularly in decision-making and control domains.

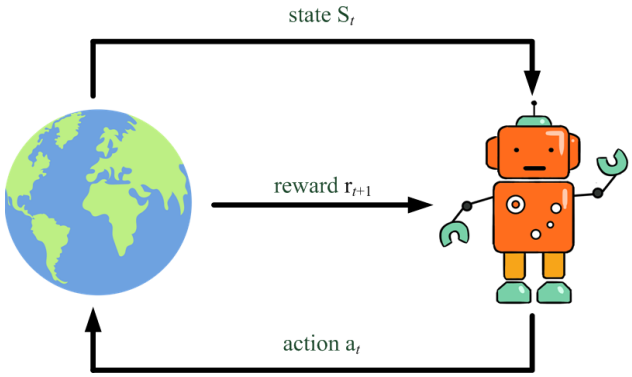


Figure 1. Markov decision process

Designing comprehensive reward functions for agents to learn from can be challenging in many real-world problems. Sparse rewards refer to the scenario where rewards are only provided when the task completion criteria are achieved, while many intermediate steps do not receive any reward signal. For example, the quality of each move is difficult to determine during the game of Go [5-6], so rewards are only provided based on the outcome. This approach to setting reward functions is crucial for solving complex problems, especially those where it is not possible to define a reward function explicitly. Overcoming the challenge of sparse rewards opens new possibilities for

*Corresponding Author: Chih-Hsien Hsia; E-mail: hsiach@niu.edu.tw
DOI: https://doi.org/10.70003/160792642025052603002

applying RL to practical problems, which is particularly important for tasks where modelling reward functions is difficult. Solving the problem of sparse rewards can help improve sample utilization efficiency. In DRL, the high cost lies not in the training process, but in the process of acquiring samples. Sample acquisition requires interaction between agents and the environment, and this interaction is costly not only in terms of time but also in terms of safety, controllability, recoverability, and many other aspects. Therefore, if we can to some extent solve the problem of sparse rewards, we can accelerate the learning process and reduce the number of interactions between agents and the environment.

When applying DRL to address real-world problems, several challenges are typically encountered. Firstly, the environmental states in the real world are often large and complex [7]. For example, in the context of autonomous driving [8], the agent needs to consider information from various perspectives to make accurate decisions. However, the number of these states is enormous, and the data involved is also substantial. This impacts RL algorithms such as DQN [9] when dealing with large-scale state spaces because there is minimal repetition of states in the evaluation function, resulting in prolonged training periods for the agent [10]. To address this issue, prioritized experience replay (PER) [11] enhances the utilization and training speed of samples by changing the sampling probability of samples. In some specific scenarios, the priority of experience data is difficult to estimate in certain continuous decision-making problems because subsequent decisions related to each state are highly coupled, which leads to a performance degradation of the algorithm [12]. The second challenge lies in sparse rewards provided by the environment. Defining appropriate dense reward functions is often challenging in the real world, making it difficult for agent to learn effectively. In environments with sparse rewards, the rewards obtained by the agent from the environment are extremely limited, posing a significant difficulty for RL algorithms to improve performance. Addressing sparse reward problems often requires heuristic exploration strategies to assist the agent in discovering sparse rewards. The intrinsic curiosity module (ICM) [13] is one approach that encourages effectively the exploration of novel states by utilizing the differences between predicted and actual states to set intrinsic rewards. However, ICM also introduces another issue, as it fails to balance the exploration and exploitation strategies and does not provide reasonable reward values for correct samples [14-15], thereby hindering effective exploration. To addressing these challenges, various methods have been proposed in research, such as using more advanced deep learning models, introducing more sophisticated reward function designs, and employing more agent exploration strategies. These efforts aim to enhance the performance and efficiency of DRL in addressing real-world problems.

In this context, this study proposes a self-balancing exploration and exploitation method based on the DRL algorithm, as proximal policy optimization (PPO) [16] to address challenges in large-scale sparse reward environments. Firstly, to address the extensive state space in large-scale environments, we evaluate the current agent's progress in completing tasks based on external rewards and optimize its approach to exploration and exploitation. We shape the reward function to guide the agent in learning samples with high reward values, thereby reducing the frequency of interactions with the environment. Secondly, to address the sparse reward issue, we developed a dual-trajectory exploration network. Specifically, we constructed an internal reward function based on the novelty of states and the degree of trajectory similarity, balancing exploration and exploitation strategies. This study guided agents based on shaped rewards and used exploration to solve the problem of local optima that arise when learning high-reward samples, effectively addressing the problem of sparse rewards in large-scale environments. Our approach differs from other shaped reward methods in that we propose a self-balancing strategy that prioritizes shaped rewards while supplementing with exploration. This self-balancing method for exploration and exploitation enables us to address challenges posed by large-scale state spaces and sparse reward environments.

In summary, our contributions are as follows: 1) This study formulates the concept of progress based on the received external rewards and dynamically shapes the reward function to guide the agent in learning high-reward samples, determining the timing of exploration and exploitation, and reducing the frequency of interactions with the environment. 2) It constructs a dual-trajectory exploration network, utilizing two trajectories from sibling agents' interactions with the environment to calculate their similarity and self-adjust the balance between exploration and exploitation, thereby addressing the issue of sparse rewards.

2 Background

2.1 Reinforcement Learning

RL provides us with a solid framework in which an agent interacts with an environment to generate experience samples, aiming to learning from these samples and making optimal decisions in a specific environment. In RL, the agent is the entity that performs actions, while the environment responds to the agent's behavior, such as through game rules. The interaction process in RL can be represented by trajectories $\tau = \{(s_t, a_t, s'_t, r_t)\}$. In the case of periodic events, tasks will unfold sequentially in chronological order. The agent will be in some state $s_t \in S$, within the environment, and will select an action $a_t \in A$, based on a trained strategy. This action will interact with the environment, leading it to a new state s'_t, while the agent moves to the next time step $t+1$. The agent receives corresponding rewards r_t based on the task's reward function $r(s_t, a_t, s'_t)$. The agent's objective is to learn and implement an optimal strategy to maximize the expected cumulative reward $E_{\tau \sim p}[\Sigma_t \gamma^t r_t]$, thereby continuously optimizing its behaviors and decision-making processes. Through iterative interaction and trial and error with the environment, the agent can gradually adjust its policy to improve its decision-making capabilities.

2.2 Proximal Policy Optimization

The traditional RL algorithms face challenges when dealing with continuous action spaces and high-dimensional state spaces, while PPO [16] is one of the new methods proposed to address these challenges. The PPO algorithm is an improved policy gradient method introduced by OpenAI, aimed at resolving the optimization instability and low sampling efficiency issues in traditional policy gradient methods. Its core idea involves constraining the parameters of the new and old policy networks, enabling the new policy network to learn from data sampled by the old policy network and restricting the magnitude of policy updates to ensure stability and convergence of the policy. Specifically, the PPO algorithm introduces a constraint function to limit the difference between the new and old policies, and the formula for the constraint is as follows:

$$L^{CLIP}(\theta) = E_T^{'}[\min(r_t(\theta)A_t^{'}, clip(r_t(\theta),1-\varepsilon,1+\varepsilon)A_t^{'})] \quad (1)$$

within this context, Θ represents the policy parameters, $E_t^{'}$ indicating the expected experience over a time horizon. r_t denotes the probability ratio between the new and old policies, and $A_t^{'}$ is an estimate of the advantage value at time step t. ε is a hyperparameter, typically taking values of 0.1 or 0.2. The PPO algorithm has demonstrated outstanding performance in addressing RL problems in continuous action spaces and high-dimensional state spaces, and it has been widely applied in areas such as robot control, autonomous driving, and gaming strategies.

2.3 Intrinsic Curiosity Module

The curiosity-driven model (ICM) [13] primarily addresses environments with sparse rewards by guiding the agent to explore novel states through setting internal rewards. It provides high internal rewards for novel states and low internal rewards for non-novel states. ICM introduces two dynamic models: The first is the forward model, in which based on the feature space representation $\varphi(s_t)$ of the current state s_{t+1} and the current action a_t, the forward model obtains an estimated feature vector $\varphi^{'}(s_t)$ of the next moment's state s_{t+1}, and uses $\varphi^{'}(s_t)$ and the difference between s_{t+1} and the feature space representation $\varphi(s_t)$ as the internal reward value. The second one is referred to as the inverse model: This model is based on the current state s_t at time t and the next state s_{t+1} at time t+1, each is represented in the feature space as $\varphi(s_t)$ and $\varphi^{'}(s_t)$, respectively, through the feature model. Then, by passing through the inverse dynamics model, the estimated action value $a^{'}_t$ obtained is then compared with the true action a_t to calculate the difference between them. The reward function designed by ICM is:

$$\mathrm{r}_t = r_t^i + r_t^e \quad (2)$$

where A denotes the external reward value obtained by the agent interacting with the environment, while r_t^i is the internal reward value derived from the exploration model. In environments with sparse rewards, ICM can guide the agent to explore novel states through internal reward values.

2.4 Related Work

Reward Design and Learning. To address the issue of sparse rewards, an intuitive approach is to artificially shape the reward function, enabling the agent to receive denser rewards during interactions. The method of shaping the reward function [17-18] involves discussions on the definition of rewards, relevant empirical research results, and the impact on behavior. Additionally, Nair *et al*. [19] proposed an unsupervised learning approach based on the goal space, which allows the achievement of imagined target states by computing distances in the latent space [20] and shaping the reward function based on these distances. Through this method, it is possible to more effectively guide the agent's behavior, enabling it to learn more quickly and reach the desired target states.

Experience Replay. In order to more effectively utilize samples, researchers have proposed the PER method [11]. The core idea of PER is to prioritize the sampling of samples in the experience pool based on the size of the TD-error, in order to focus more on those samples that have a greater impact on training. Additionally, Bruin *et al*. [21] extended the framework of prioritized experience replay and proposed a unified experience selection mechanism to determine which samples should be stored in the experience pool and how to sample them. This improvement significantly increased the efficiency of sample utilization in robot arm experiments [22-23], making the training process more efficient. By prioritizing the sampling of samples with larger TD-errors, the robot arm can learn important knowledge more quickly, thereby enhancing its learning performance.

Exploration and Exploitation. In order to explore sparse rewards [24], Bellemare *et al*. [25] proposed a virtual count method that utilizes a probability generative model to measure the frequency of state occurrences and converts the frequency into virtual counts, serving as additional intrinsic rewards. To handle image states, Ostrovski *et al*. [26] emphasized the importance of probability models and proposed the use of PixelCNN as the choice for the probability generative model. This approach is more suitable for processing image states. Additionally, Pathak *et al*. [13] employed the intrinsic curiosity module (ICM) to obtain feature representations of states by calculating the disparity between predicted and actual states, and by excluding action-irrelevant components to enhance the effectiveness of intrinsic motivation. The improvements of these methods aim to enhance the agent's intrinsic drive to better address the challenges of sparse rewards. By introducing additional intrinsic rewards or using probability generative models to measure state frequency [27], agents can better explore the environment and learn more effective strategies.

Curriculum Learning. To gradually learn more complex tasks, agents can use the method of progressive learning. PowerPlay et al. [28] proposed to train a progressively more general problem solver by

continuously seeking the simplest yet unresolved problem, using a recursive self-improvement method to adjust its own structure and parameters to optimize problem-solving ability. Florsena et al. [29] proposed a reverse curriculum generation method that can generate task sequences suitable for the agent's ability level and task goals for reinforced learning agents. Their method first uses a pre-trained policy network to generate some tasks and then reversely generates suitable task sequences based on the difficulty of these tasks and the performance of the agents. This method can improve the learning efficiency and generalization ability of agents. Through progressive learning [30], agents can gradually improve their abilities and cope with more challenging task environments.

3 A Self-Balancing Approach to Exploration and Exploitation

In Figure 2, the self-balancing exploration and exploitation was presented. The dynamic shaping of the reward function based on the agent's progress in task completion is detailed in subsection 3.1, while section 3.2 is detailed a comprehensive description of the dual-path exploration network and the method of self-balancing exploration and exploitation.

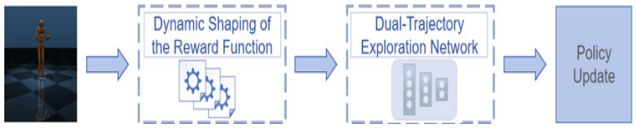


Figure 2. The process of our method

3.1 Dynamically Shaping Reward Functions

In the context of large-scale sparse reward environments, where the state space is vast and complex, the agent requires substantial amount of interaction experience with the environment to learn the optimal policy, severely impacting the performance of RL algorithms. In this section, we propose to compute the average of the external rewards obtained by the agent and use it as an evaluation of the agent's progress φ in task completion. We define the progress evaluation for the first assessment as φ_0. After this progress evaluation, if the external reward r_t obtained by the agent a_t time step t is not less than φ_0, it indicates that this sample is above the average level and is worth learning from.

After evaluating the progress, it is necessary to shape the reward function based on the progress φ. For samples that exceed the average level, it is important to assign a greater reward value to encourage the agent to learn from high-reward samples. The reward function shaped based on the latest progress φ_0 is given by the following equation:

$$r_t^{'} = \begin{cases} r_t^e, & r_t^e < \varphi_0 \\ \omega r_t^e \varphi_0, r_t^e \geq \varphi_0 \end{cases} \tag{3}$$

where r_t^e is the external reward obtained by the agent at time step t from interacting with the environment within an episode, and ω is a hyperparameter.

Based on the obtained external reward, it is used to determine the progress of the agent. If the obtained reward is greater than the current task completion progress φ, the reward will be significantly higher than the original external reward value, with ω ensuring that the reward does not fall below the original reward. This reward function is designed to guide the agent to learn from high-reward samples, discarding a large amount of useless interaction, and promoting the improvement of the performance of RL algorithms in large-scale environments.

Guiding the agent to learn from high-reward samples can increase the mean of the external rewards obtained by the agent, effectively enhancing the agent's progress in task completion, denoted as φ. To achieve this, a threshold β will be set to assess the degree of change in the latest progress φ. For the current evaluated progress $\varphi_1 = \varphi_0 + \beta$, the progress will be updated by φ to yield φ_1, where $\varphi_1 = \varphi_0 + \beta$. Furthermore, after the progress update, the reward function will be reshaped as follows:

$$r_t^{'} = \begin{cases} r_t^e, & r_t^e < \varphi_0 \\ \omega r_t^e \varphi_0, & \varphi_0 \leq r_t^e \leq \varphi_1 \\ \omega r_t^e \varphi_1, & \varphi_1 \leq r_t^e \end{cases} \tag{4}$$

For the aforementioned formula, given $\varphi_1 > \varphi_0$, it implies that the external rewards obtained by the agent at time step t fall into three intervals, wherein if r_t^e is greater than the latest progress φ_1, then the increment in rewards obtained is maximized.

The agent continues to learn and continuously updates the mean of the obtained external rewards. If the current evaluated progress is $\varphi = \varphi_i + \beta$, there is a need to update progress φ_i to φ_{i+1}, where $\varphi_{i+1} = \varphi_i + \beta$. As the progress of task completion φ is continuously updated to φ_n, the shaped reward function is given by:

$$r_t^{'} = \begin{cases} r_t^e, & r_t^e < \varphi_0 \\ \omega r_t^e \varphi_0, & \varphi_0 \leq r_t^e \leq \varphi_1 \\ \quad \cdots & \\ \omega r_t^e \varphi_i, & \varphi_i \leq r_t^e \leq \varphi_{i+1} \\ \quad \cdots & \\ \omega r_t^e \varphi_{n-1}, & \varphi_{n-1} \leq r_t^e \leq \varphi_{i+1} \\ \omega r_t^e \varphi_n, & \varphi_n \leq r_t^e \end{cases} \tag{5}$$

This reward function divides the progress of task completion φ into $n+1$ stages, where the rewards obtained by the agent from interacting with the environment all fall within stage $n+1$, with the increment in the original external reward values increasing from each stage. Assuming that the current progress φ of the agent in task completion is at φ_{n-1}, the external reward r_t^e obtained, regardless of how much greater than φ_{n-1}, remains at value $\omega r_t^e \varphi_{n-1}$. After the progress φ is updated to φ_n, the

reward value of sample $r_t^e \geq \varphi_n$ at the latest progress φ_{n+1} is smaller than the reward value of sample $r_t^e \geq \varphi_n$ at the latest progress φ_n, even though both samples receive the same external reward. Therefore, using progress to dynamically shape the reward function can incentivize the agent to effectively learn from high-reward samples, increase the mean of external rewards, and improve the progress of task completion.

In large-scale sparse reward environments, where the majority of samples have no rewards, it is challenging to effectively promote the learning of the agent. Therefore, we need to use intrinsic rewards to guide the agent to explore near high-reward samples, thus facilitating the learning of the agent.

3.2 Self-Balancing Exploration and Exploitation

Reasonably changing the exploration and exploitation strategies can effectively promote the learning of the agent. By dynamically shaping the reward function through progress φ, the agent is encouraged to learn from high-reward samples. After an update at progress φ, it can be indicated that the agent has achieved certain results in learning from high-reward samples, and under the current policy, continued interaction with the environment may yield even higher-reward samples, thus enhancing the progress of task completion φ. Therefore, at this point, the agent should lean more towards exploitation under the current policy to learn from similarly high-reward samples rather than exploring new samples. If there is no update in progress φ after a period of time, it can indicate that after learning from high-reward samples, the agent is in the vicinity of the high-reward samples being learned, without substantial progress. Therefore, it is necessary to encourage the agent to explore near the high-reward samples to prompt the agent to find even higher-reward samples. At this point, the agent's strategy should lean towards exploration.

The main idea of this approach is to adjust the exploration and exploitation strategies at the appropriate time. To this end, we have constructed a dual-trajectory exploration network as shown in the figure. The first part of the dual-trajectory exploration network employs an internal reward mechanism based on the ICM for the forward model. This involves predicting the next state based on the current state and action, and then comparing the predicted value with the actual state to derive the difference as the internal reward value. This model is used to assign intrinsic reward values to novel states and the loss of this model aligns with that of the ICM.

In Figure 3, dual-trajectory exploration network, we utilize sibling agents to interact with the environment under the same initial state s_t through the current policy $\pi_t(a|s)$, resulting in two sets of trajectories $\tau^1(s_t \ldots s_{t+k}, a_t \ldots a_{t+k}, r_t \ldots r_{t+k}, s_{t+1} \ldots s_{t+k+1})$ and $\tau^2(s_t \ldots s_{t+k}, a_t \ldots a_{t+k}, r_t \ldots r_{t+k}, s_{t+1} \ldots s_{t+k+1})$, where k denotes the number of steps in which the trajectories interact with the agent. All states from the samples are stored in separate lists, and then fed into a feature model to obtain $\varphi^1(s_{t+1} \ldots s_{t+k+1}; \theta)$ and $\varphi^2(s_{t+1} \ldots s_{t+k+1}; \theta)$, respectively. The feature vectors can be used to compare the states. Subsequently, we calculate the differences between the feature vectors of the two trajectories, which is used to evaluate the extent to which the current policy π balances exploration and exploitation. In conclusion, the dual-trajectory exploration network helps the agent calculate the novelty of states and also assesses the similarity between trajectories of sibling agents to determine the balance between exploration and exploitation under the current policy.

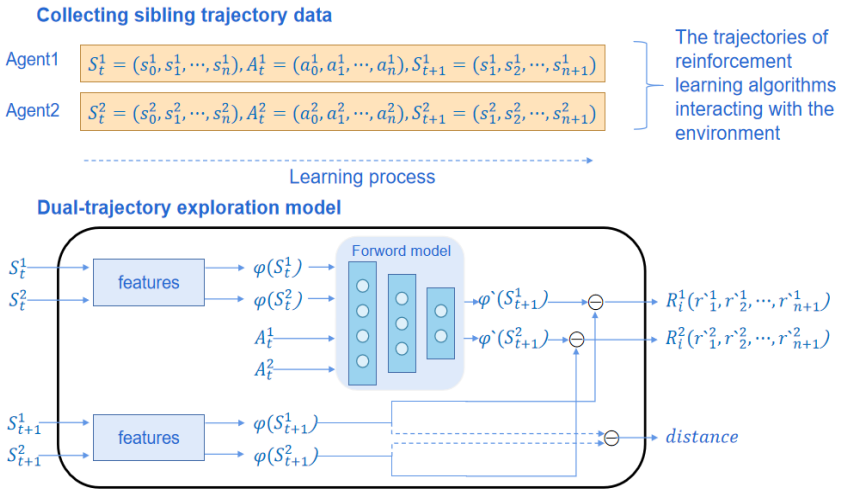


Figure 3. Dual-trajectory exploration model

We define the intrinsic reward function as:

$$r_{inner} = r_{in} * sig \tag{6}$$

where r_{in} denotes the discrepancy value between the predicted state and the actual state obtained through the forward model, and sig is determined based on the progress update. The formula for sig is as follows:

$$sig = \begin{cases} -y, & \delta \text{ is updated} \\ sig + \alpha * distance * y, & \delta \text{ has not been updated} \end{cases} \quad (7)$$

The variables α and y are hyperparameters, typically positive integers. The term *distance* represents the similarity between trajectories of neighboring agents, which reflects the balance between exploration and exploitation in the current policy. Calculating the similarity between two trajectories helps assess the true extent of exploration by the current policy. In the given equation, when there is an update in variable δ, the current policy needs to learn more from high-reward samples. In this case, the policy should lean towards exploitation to sample more high-reward instances. Therefore, the parameter *sig* is set as a negative value, resulting in negative intrinsic rewards within variable r. This indicates that the more novel a state is, as reflected by a larger value in variable r_{in}, the smaller the intrinsic reward received. This can be used during the exploitation phase to discourage the agent from accessing new states. On the other hand, when there is no update in variable δ, it implies that the current policy needs to obtain higher reward samples. In this scenario, the policy should lean towards exploration. Therefore, we gradually increase the value of *sig*, when *sig* becomes positive, it indicates that the agent has explored more novel states, and the intrinsic reward value is larger, which encourages the agent to visit new states.

The difference value between sibling trajectories can represent the balance between exploration and exploitation in the agent's current policy. Exploitation allows the agent to dynamically adjust its exploration and exploitation strategies, aiming to achieve self-balancing exploration and exploitation.

In large-scale sparse reward environments, we evaluate the agent's progress in completing tasks using metric δ and dynamically shape rewards to encourage the agent to learn from high-reward samples. This approach significantly reduces the number of interactions between the agent and the environment, allowing for more efficient learning from valuable data samples. To mitigate the impact of the lack of rewards in sparse reward environments on agent learning, we employ an intrinsic reward value through a dual-trajectory exploration network. By leveraging metric δ effectively, the agent achieves a self-balancing exploration and exploitation, leading to improved performance of the DR algorithm.

4 Experimental Results

4.1 Experimental Environment

MuJoCo is a software platform used to achieve high-performance dynamic simulation and control, for achieving widely applied in research and development of robot learning, RL, and control algorithms. The platform provides efficient dynamic simulators and diverse physical environments for evaluating and validating the performance of various robots and control algorithms. MuJoCo is commonly used for training and evaluating RL algorithms, particularly for controlling robot motions, complex object manipulation, and other tasks requiring highly realistic physical simulation. Within the MuJoCo environment, one can utilize scenarios such as Ant, HalfCheetah, Walker2D, Humanoid, and others. The observation spaces and action spaces of these environments consist of vectors composed of physical information, with the action space typically being a continuous action space of size N. As for rewards, corresponding game scores are defined based on the specific task, with significant variations in the rewards for each specific game. MuJoCo's environments offer a rich variety of scenarios and physical parameters, providing an important platform for training and evaluating RL algorithms.

4.2 Experimental Reward Setting

The MuJoCo environment is well-known for its fast and highly realistic physical simulation, making it suitable for simulating various complex physical environments and robotic systems. Due to the relatively large state space of each environment, agents require more time to learn. To better approximate the setting of sparse reward environments, we set up the MuJoCo platform so that the agent can receive environment rewards when it completes every 200 steps or tasks. This adjustment allows the agent to more realistically coping with sparse reward environments and promotes its learning process.

4.3 Experimental Baseline

The PPO algorithm in RL has demonstrated strong solving capabilities in the MuJoCo environment. PPO addresses the issue of optimizing sample reuse by employing the clip method, enabling rapid learning of the optimal policy within a close-range reward. On the other hand, the ICM algorithm introduces a novel concept by providing intrinsic rewards based on the difference between predicted and actual states, effectively addressing exploration problems under sparse rewards, thereby enabling the agent to learn to complete tasks. In this experiment, these two algorithms are used as the experimental baselines.

4.4 Experimental Analysis

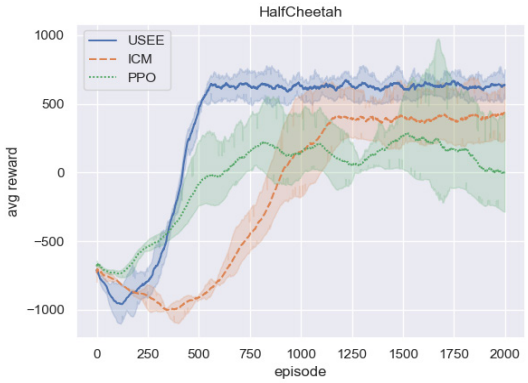
In the MuJoCo experimental environment, agents face a significantly large state space. In dense reward environments, agents often find it relatively easy to complete tasks; however, in sparse reward environments, extensive learning is required to achieve task proficiency. Our algorithms have demonstrated excellent performance across five experimental environments, enabling the exploration of more valuable experiences and resulting in higher scores for the agents within the environment.

In the Ant and HalfCheetah environments, the agents' goal is to coordinate their four legs to move forward, and the state space is not large. The agents can accomplish these tasks after a certain level of learning. However, within the MuJoCo setup, agents are prone to losing positive rewards due to death, and they only obtain positive rewards when they are able to move forward in a healthy manner. During the initial stages of training in these two environments, our algorithm's performance is poor due

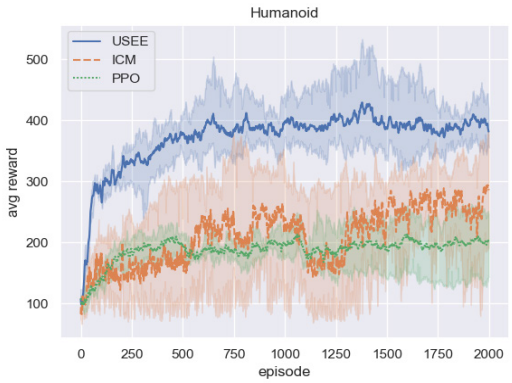
to frequent deaths resulting from extensive exploration, as shown in Figure 4(a) and Figure 4(b). However, once the exploration phase is completed, the agents can easily make correct movements in the environment, accomplish tasks successfully, and obtain high rewards. In contrast, although PPO algorithm tends to be stable but with lower average rewards due to inadequate exploration. The ICM algorithm focuses on exploration and can surpass PPO in terms of average rewards at later stages. However, due to its drawbacks, the ICM algorithm fails to properly balance between exploration and exploitation, resulting in a shortfall in average rewards.

In the Walker2d, Humanoid, and HumanoidStandup environments, which mimic human walking and standing, each agent has a large number of physically articulated connections. This results in an extremely vast state space that typically requires extensive training to learn how to complete tasks. As shown in Figure 4(c), Figure 4(d), and Figure 4(e), the PPO algorithm fails to effectively learn information in these three sparse reward environments, leading to a continual inability to improve average rewards. Due to its exploration capabilities, the ICM algorithm exhibits slightly inferior and unstable average rewards and even performs lower than PPO in some cases. However, our proposed algorithm (using self-balancing exploration and exploitation, USEE) effectively utilizes exploration by balancing exploration and exploitation strategies in later stages. This enables the agents to demonstrate outstanding performance with fast learning speed and stability in these three environments. As a result, our algorithm can fully leverage its capabilities in large-scale sparse reward environments for accomplishing the agent's tasks.

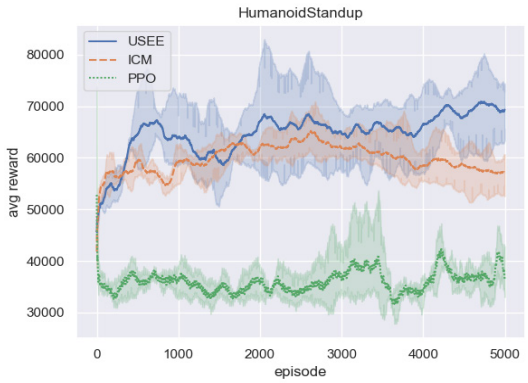
(a) Ant



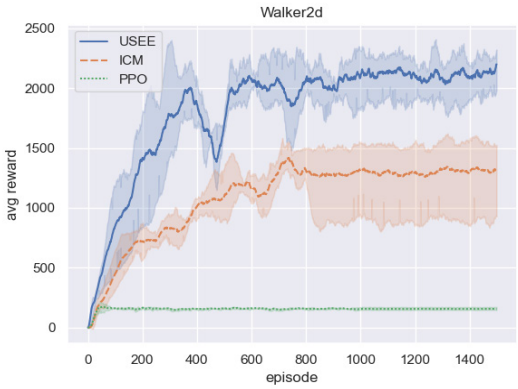
(b) HalfCheetah



(c) Humanoid



(d) HumanoidStandup



(e) Walker2d

Figure 4. Average rewards across different environments

5 Conclusion and Future Work

The main focus of this study is to address the issue of poor learning efficiency of agents in large-scale sparse reward environments. Firstly, this work evaluates the agent's progress in environments with vast state spaces and shapes rewards based on that progress to guide the agent in learning high-reward samples and reduce its interaction with the environment. Secondly, to tackle the problem of sparse rewards, this work constructs a dual-trajectory exploration network. The first part of the network calculates the novelty reward value based on the novelty of each state, while the second part evaluates the similarity between sibling trajectories. By self-balancing exploration and exploitation strategies through similarity at opportune times for both exploration and exploitation, effective exploration near high-reward samples is achieved. In our work, enhancing RL algorithms' performance in large-scale sparse reward environments is effectively accomplished by learning high-reward samples through shaping the reward function and self-balancing exploration and exploitation.

For future work, we will conduct further research in the following areas. Firstly, to address the sparse reward problem, we will not solely rely on exploration to obtain intrinsic rewards, as we will introduce inverse RL algorithms to shape more realistic intrinsic rewards to facilitate agent learning. Secondly, we plan to utilize other neural networks such as convolutional neural networks, long short-term memory, and ResNet. Thirdly, in the reward function shaping aspect, we intend to use more sophisticated approaches to replace the agent's task completion progress.

References

[1] L. P. Kaelbling, M. L. Littman, A. W. Moore, Reinforcement learning: a survey, *Journal of Artificial Intelligence Research*, Vol. 4, No. 1, pp. 237-285, January, 1996.

[2] R. Bellman, A Markovian decision process, *Journal of Mathematics and Mechanics*, Vol. 6, No. 5, pp. 679-684, 1957. https://www.jstor.org/stable/24900506

[3] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, *Nature*, Vol. 521, No. 7553, pp. 436-444, May, 2015. https://doi.org/10.1038/nature14539

[4] J. Zhang, C. Zhang, W.-C. Chien, Overview of deep reinforcement learning improvements and applications, *Journal of Internet Technology*, Vol. 22, No. 2, pp. 239-255, March, 2021. https://doi.org/10.3966/160792642021032202002

[5] Y. Kong, Y. Rui, C.-H. Hsia, A deep reinforcement learning-based approach in poker game, *Journal of Computers*, Vol. 34, No. 2, pp. 41-51, April, 2023. https://doi.org/10.53106/199115992023043402004

[6] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, D. Hassabis, A general reinforcement learning algorithm that masters chess, shogi, and go through self-play, *Science*, Vol. 362, No. 6419, pp. 1140-1144, December, 2018. https://doi.org/10.1126/science.aar6404

[7] T. Zahavy, M. Haroush, N. Merlis, D. J. Mankowitz, S. Mannor, Learn what not to learn: action elimination with deep reinforcement learning, *International Conference on Neural Information Processing Systems*, Montréal Canada, 2018, pp. 3566-3577.

[8] D. Isele, R. Rahimi, A. Cosgun, K. Subramanian and K. Fujimura, "Navigating occluded intersections with autonomous vehicles using deep reinforcement learning," *IEEE International Conference on Robotics and Automation*, pp. 2034-2039, 2018.

[9] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, M. Riedmiller, Playing atari with deep reinforcement learning, *arXiv preprint arXiv:1312.5602v1*, December, 2013. https://arxiv.org/abs/1312.5602

[10] L.-J. Lin, Self-improving reactive agents based on reinforcement learning, planning and teaching, *Machine Learning*, Vol. 8, No. 3-4, pp. 293-321, May, 1992. https://doi.org/10.1007/BF00992699

[11] Z. Yang, Y. Kong, C.-H. Hsia, DERLight: a deep reinforcement learning traffic light control algorithm with dual experience replay, *Journal of Internet Technology*, Vol. 25, No. 1, pp. 79-86, January, 2024. https://doi.org/10.53106/160792642024012501007

[12] Y. Hou, L. Liu, Q. Wei, X. Xu, C. Chen, A novel DDPG method with prioritized experience replay, *IEEE International Conference on Systems, Man, and Cybernetics*, Banff, AB, Canada, 2017, pp. 316-321, 2017. https://doi.org/10.1109/SMC.2017.8122622

[13] D. Pathak, P. Agrawal, A. A. Efros, T. Darrell, Curiosity-driven exploration by self-supervised prediction, *International Conference on Machine Learning*, Sydney, NSW, Australia, 2017, pp. 2778-2787.

[14] A. P. Badia, P. Sprechmann, A. Vitvitskyi, D. Guo, B. Piot, S. Kapturowski, O. Tieleman, M. Arjovsky, A. Pritzel, A. Bolt, C. Blundell, Never give up: learning directed exploration strategies, *arXiv preprint arXiv:2002.06038*, February, 2020. https://arxiv.org/abs/2002.06038

[15] Y. Burda, H. Edwards, A. Storkey, O. Klimov, Exploration by random network distillation, *arXiv preprint arXiv:1810.12894v1*, October, 2018. https://arxiv.org/abs/1810.12894

[16] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, Proximal policy optimization algorithms, *arXiv preprint arXiv:1707.06347v2*, August, 2017. https://arxiv.org/abs/1707.06347

[17] A. Y. Ng, S. J. Russell, Algorithms for inverse reinforcement learning, *International Conference on Machine Learning*, Stanford, CA, USA, 2000, pp. 663–670.

[18] J. Asmuth, M. L. Littman, R. Zinkov, Potential-based shaping in model-based reinforcement learning, *AAAI Conference on Artificial Intelligence*, Chicago, Illinois, 2008, pp. 604-609.

[19] A. V. Nair, V. Pong, M. Dalal, S. Bahl, S. Lin, S. Levine, Visual reinforcement learning with imagined goals, *International Conference on Neural Information Processing Systems*, Montréal, Canada, 2018, pp. 1-10.

[20] A. Péré, S. Forestier, O. Sigaud, P. Y. Oudeyer, Unsupervised learning of goal spaces for intrinsically motivated goal exploration, *arXiv preprint arXiv:1803.00781v3*, October, 2018. https://arxiv.org/abs/1803.00781

[21] T. D. Bruin, J. Kober, K. Tuyls, R. Babuska, Experience selection in deep reinforcement learning for control, *Journal of Machine Learning Research*, Vol. 19, No. 9, pp. 1-56, 2018.

[22] D. Rolnick, A. Ahuja, J. Schwarz, T. Lillicrap, G. Wayne,

Experience replay for continual learning, *International Conference on Neural Information Processing Systems*, Vancouver, BC, Canada, 2019, pp. 350-360.
[23] H. van Hasselt, A. Guez, D. Silver, Deep reinforcement learning with double Q-learning, *AAAI Conference on Artificial Intelligence*, Phoenix, Arizona, 2016, pp. 2094-2100.
[24] P. Ladosz, L. Weng, M. Kim, H. Oh, Exploration in deep reinforcement learning: a survey, *Information Fusion*, Vol. 85, pp. 1-22, September, 2022. https://doi.org/10.1016/j.inffus.2022.03.003
[25] M. Bellemare, S. Srinivasan, G. Ostrovski, T. Schaul, D. Saxton, R. Munos, Unifying count-based exploration and intrinsic motivation, *International Conference on Neural Information Processing Systems*, Barcelona, Spain, 2016, pp. 1479-1487.
[26] G. Ostrovski, M. G. Bellemare, A. Oord, R. Munos, Count-based exploration with neural density models, *International Conference on Machine Learning*, Sydney, NSW, Australia, 2017, pp. 2721-2730.
[27] H. Tang, R. Houthooft, D. Foote, A. Stooke, X. Chen, Y. Duan, J. Schulman, F. De Turck, P. Abbeel, Exploration: a study of count-based exploration for deep reinforcement learning, *International Conference on Neural Information Processing Systems*, Long Beach California USA, 2017, pp. 2750-2759.
[28] J. Schmidhuber, Powerplay: training an increasingly general problem solver by continually searching for the simplest still unsolvable problem, *Frontiers in Psychology*, Vol. 4, Article No. 313, June, 2013. https://doi.org/10.3389/fpsyg.2013.00313
[29] C. Florensa, D. Held, M. Wulfmeier, M. Zhang, P. Abbeel, Reverse curriculum generation for reinforcement learning, *Annual Conference on Robot Learning*, *Proceedings of Machine Learning Research*, pp. 482-495, 2017.
[30] S. Narvekar, B. Peng, M. Leonetti, J. Sinapov, M. E. Taylor, P. Stone, Curriculum learning for reinforcement learning domains: a framework and survey, *Journal of Machine Learning Research*, Vol. 21, No. 1, pp. 7382-7431, 2020.

Biographies

Yan Kong received her Ph.D. degree in Computer Science from the University of Wollongong, Australia. Currently, she works as an Associate Professor in Nanjing University of Information, Science and Technology, China. Her research interests include Deep Reinforcement Learning, Multi-agent system, and Machine Learning. Moreover, her research focuses on the smart control on the traffic signal lights to alleviate the traffic congestion.

Junfeng Wei received his Master's degree in Software Engineering from Nanjing University of Information Science and Technology, China. His research interest is deep reinforcement learning. In addition, his research focuses on the problem of improving the learning efficiency of agents in sparse reward environments.

Chih-Hsien Hsia received the Ph.D. degree in Electrical and Computer Engineering from Tamkang University, and the second Ph.D. degree from National Cheng Kung University, Taiwan, respectively. He currently is a Full Professor and a Chairperson with the Department of Computer Science and Information Engineering, NIU. His research interests include DSP IC Design, AI in Multimedia, and Cognitive Engineering.