Prioritized Experience Replay in Multi-Actor-Attention-Critic for Reinforcement Learning

Sheng Fan¹, Guanghua Song¹, Bwei Yang¹ and Xiaohong Jiang²

¹School of Aeronautics and Astronautics, Zhejiang University, Hangzhou 310027, China
²College of Computer Science and Technology, Zhejiang University, Hangzhou 310027, China
Email: ghsong@zju.edu.cn

Abstract. Experience replay is a significant method of off-policy reinforcement learning (RL), which makes RL reuse the past experience and reduce the correlation between samples. Multi-Actor-Attention-Critic (MAAC) is a successful off-policy multi-agent reinforcement learning algorithm, due to its good scalability. To accelerate convergence, we use prioritized experience replay (PER) to optimize the experience selection in MAAC, and propose the PER-MAAC algorithm. In the PER-MAAC, the priority metric is based on the temporal-difference error during training. The algorithm is evaluated in the scenarios of Multi-UAV Cooperative Navigation and Rover-Tower. The experimental results show that PER-MAAC improves the speed of convergence.

1. Introduction

Deep reinforcement learning (DRL) has achieved great success on single-agent systems in recent years, such as playing Atari game [1] and Go [2, 3]. With the success of DRL, academics try to apply it to the multi-agent environment. However, it is challenging for multiple agents to learn policies effectively, because the policies of other agents are also part of the environment from the perspective of an individual agent. The multi-agent environment is non-stationary [4], thus it is not applicable to directly employ the single-agent algorithm.

To deal with this issue, several approaches of multi-agent have been proposed in recent years. In MADDPG [5], the authors adopt the framework of concentrated training and decentralized execution, where agents only make use of their own local information to get the best action during training while the centralized critic can concatenate all information from all agents. Moreover, there is no need to know the dynamic model of the environment and the special communication model. In addition, it can be used not only in a cooperative environment but also in a competitive environment.

Nevertheless, as the number of agents increases, the dimension of actions will grow exponentially, which is difficult to apply in the large-scale and complex environment. MAAC [6] follows the framework of centralized training and decentralized execution and utilizes the attention mechanism [7] to extract the correlation between agents. In particular, it has an advantage in a certain way because it avoids paying attention to all information, while selectively focus on some valuable information from other agents.

Experience replay is an important method for off-policy reinforcement learning, which mainly overcomes the problems of correlation data and non-stationary distribution of the experiences. In recent years, many scholars have made good progress in the experience replay [8-15]. De Bruin et al.
[8] use the prior knowledge to optimize the experience retention and sampling for robotic control. Hou et al. [9] focus on extending the prioritized experience replay mechanism [16] in reinforcement learning from the discrete control to the continuous control, and point out that this approach can significantly reduce the training time of the network and improve the stability of the training process and the robustness of the model. Wang et al. [15] combine experience reservoir retention with prioritized experience replay and apply them from single-agent to multi-agent system (MADDPG).

In our paper, we enhance the sampling method of MAAC to reduce the training time of the multi-agent system. On the basis of MAAC, our work combines prioritized experience replay to accelerate the convergence. We have evaluated our work in the scenarios of Multi-UAV Cooperative Navigation and Rover-Tower [6] in MAAC.

The rest of the paper is organized as follows. Section 2 introduces the related work of MAAC. Section 3 presents our methods. Section 4 presents simulation results for performance evaluation and finally, Section 5 concludes the paper.

2. Background

2.1. Markov Games
Littman [17] proposed the Markov games framework, which describes the Markov Decision Processes with $N$ agents. They are defined by a set of states $S$, observation sets $O_{1}, ..., O_{N}$, action sets $A_{1}, ..., A_{N}$, a state transition function defined by state for each agent, $T: S \times A_{1} \times \ldots \times A_{n} \to S$ and reward function of each agent $r_{j}: S \times A_{1} \times \ldots A_{n} \to \mathbb{R}$, $\forall j \in \{1, ..., N\}$. Each agent $i$ receives a private observation from global state, $o_{i}: S \to O_{i}$ and obtains reward as a function of the state and action. Each agent aims to learn a policy that maximizes its expected discounted return $R_{i} = \sum_{t=0}^{T} \gamma^{t} r_{j}$ where $\gamma$ is a discount factor to and $T$ is the time boundary.

2.2. Soft Actor-Critic
Haarnoja et al. [18, 19] proposed an optimized algorithm called “Soft Actor-Critic” (SAC), which introduces a general maximum entropy objective to enhance exploration and robustness:

$$J(\pi) = \sum_{t=0}^{T} \mathbb{E}_{(s_{t}, a_{t}) \sim \pi(\cdot)}[r(s_{t}, a_{t}) + \alpha H(\pi(\cdot|s_{t}))]$$

The temperature parameter $\alpha$ determines the importance of entropy in rewards, affecting the distribution of stochasticity. $H$ is an entropy regularization term.

2.3. Attention
In recent years, attention [7] has achieved great success in computer vision and natural language processing. In particular, the self-attention function exacts the relative information between the embeddings. The output of the function is the weighted sum of the value vectors, where the weights are computed by the query and key vectors:

$$Attention(Q, K, V) = \text{softmax}(\frac{QK^{T}}{\sqrt{d_{k}}})V$$

where $d_{k}$ is the dimension of the query, key and value vectors. $Q$, $K$ and $V$ are all the vectors of the embedding. Moreover, this function utilizes the multi-head attention to stabilize the learning process.

2.4. Multi-Actor-Attention-Critic
Iqbal & Sha [6] took advantage of the attention mechanism to select relevant information for each agent, which made MAAC more scalable with the increase of the number of agents. The author used a centralized critic to incorporate the information of other agents. More concretely, considering a game
with $N$ agents with observation $o = (o_1, ..., o_{N})$ and actions $a = (a_1, ..., a_{N})$, the Q-value function $Q_i\Phi(o,a)$ for agent $i$, $i \in \{1, ..., N\}$:

$$Q_i\Phi(o,a) = f_i(g_i(o_i,a_i), x_i)$$

(3)

where $f_i$ is a two-layer multi-layer perceptron (MLP) and $g_i$ is a one-layer MLP. $x_i$ is the weighted sum of values from other agents through attention:

$$x_i = \sum_{j \neq i} \alpha_j v_j = \sum_{j \neq i} \alpha_j h(v_j(o_j,a_j))$$

(4)

where $h$ is a nonlinearity activation function (e.g. ReLU) and $V$ is a shared matrix to linearly transform the encoding embedding to generate the $v_j$. The attention weight $\alpha_j$ is the similarity between $e_i = g_i(o_i,a_i)$ and $e_j$, calculated by self-attention function:

$$\alpha_j \propto \exp(e_j^T W_k^i W_q e_i)$$

(5)

Due to the centralized critic, MAAC adopted a joint regression loss function to update critic by sharing parameter:

$$L_\Phi(\psi) = \sum_{i=1}^{N} \mathbb{E}_{(o,a,r,o') \sim D_i}[(Q_i\Phi(o,a) - y_i)^2], \text{where}$$

$$y_i = r_i + \gamma \mathbb{E}_{a' \sim \pi_{\theta}(a')}[Q_{\pi}(o',a') - \alpha \log(\pi_{\theta}(a_i | o_i))]$$

(6)

where $\pi$ and $\pi_{\theta}$ are the parameter of the target critics and target policies respectively. The $\alpha$ is the temperature parameter to control the stochasticity of the optimal policy.

To deal with the credit allocation, the author also introduced multi-agent advantage function, where the function uses a baseline to judge that whether the increase of reward is caused by the action of the agent itself or the other agents:

$$A_i(o,a) = Q_i\Phi(o,a) - b(o,a_i), \text{where}$$

$$b(o,a_i) = \mathbb{E}_{a_i \sim \pi_i(o_i)}[Q_i\Phi(o,(a_i,a_i))]$$

(7)

3. Method

Uniform sampling is a classic method in reinforcement learning, which extracts transition from experience replay uniformly. However, different experiences have different impacts on training results, even the same experience could produce different results in different episodes. Some experiences will can have significant influence on the current training, while others can not. Schaul et al. [16] proposed prioritized experience replay and have great performance on DQN. Inspired by this, we introduce the prioritized experience replay (PER) to MAAC. Similar to [16], temporal-difference (TD) error can be measured as importance to calculate the priority. In single-agent reinforcement learning, PER calculate the priority by the function as:

$$P(i) = \frac{p_i^\alpha}{\sum_{k=1}^{M} p_k^\alpha}, \text{where}$$

$$p_i = |\text{Loss}_i| + \varepsilon$$

(8)

Here $p_i$ is the priority of experience sample $i$ and $\alpha$ is the hyperparameter to control the proportion of priority. Usually, $\varepsilon$ is a small positive value to prevent the low priority from being drawn. Owing to the joint loss in the update of critic, it is beneficial to take advantage of the sum of TD-error of all agents as the priority. $p_i$ is the sum of absolute value of TD-error of all agents, thus it can be
\[ p_i = \frac{N}{\sum_{j=1}^{N} |\text{Loss}_j(j)| + \varepsilon} \]  

In MAAC, TD-error requires additional consideration of the entropy term, thus the TD-error of agent \( j \) is calculated by formula (10):

\[ \text{Loss}(j) = \mathbb{E}_{(o,a,r,s', o') \sim \mathcal{D}}(Q_j^o(o,a) - (r_j + \gamma \mathbb{E}_{o' \sim p_{o | s}}(Q_j^{o'}(o', a') - \alpha \log(\pi_{\alpha}(a_j' | o'))))) \]  

To reduce the complexity of sorting, we use the ‘sum-tree’ data structure to save the priority of all transition. Proportional prioritization sampling is a good choice in minibatch of size \( k \), where it divides the interval \([0, p_{\text{max}}]\) into \( k \) ranges and sampling uniformly from each range. This ensures low-priority sampling rate while avoiding overfitting. Taking into account the complexity of the algorithm and calculation, the loss in the selected minibatch experience is only updated during training, and the loss of other experiences in the experience replay will not be recalculated. We set the new transition as the highest priority to ensure they can be sampled preferentially.

**Algorithm 1: PER-MAAC Procedure**

1: **Input:** minibatch \( k \), exponent \( \alpha \), episode-size \( \eta \), step-size \( \delta \), num-updates \( \varphi \), replay period \( K \)
2: Initialize \( E \) parallel environments with \( N \) agents
3: Initialize experience replay \( D \), \( p_1 = 1 \)
4: \( T_{\text{update}} \leftarrow 0 \)
5: for \( i_{\text{ep}} = 1, \ldots, \eta \) do
6:   Reset environments, get initial observation \( o_j \) of each agent, \( j \)
7:   for \( t = 1, \ldots, \delta \) do
8:     Select actions \( a_j^e \sim \pi_j(c|o_j^e) \) for each agent, \( j \), in each parallel environment
9:     Send actions to other agents, get \( x_j \) for each agent, \( j \)
10: Store \((o,a,r,o')\) of all agents into \( D \) with maximal priority \( \max_{i<k} p_i \)
11: \( T_{\text{update}} \leftarrow T_{\text{update}} + E \)
12: if \( T_{\text{update}} > \text{min steps per update} \) then
13:   for \( j_{\text{update}} = 1, \ldots, \varphi \) do
14:     Divide \([0, p_{\text{max}}]\) into \( k \) ranges and sampling uniformly, \( B(k) \sim P(k) = p_k^\alpha / \sum_{i=1}^{K} p_i^\alpha \)
15:     Update Critic(\( B \))
16:     Calculate TD-error of all agents of \( B \), \( p_k = \sum_{j=1}^{N} |\text{Loss}_j(j)| + \varepsilon \)
17:     Update normalized \( P \) of each transition in \( D \), \( p_k \leftarrow p_k / \max_{i<k} p_i \)
18:   Update Policies(\( B \))
19: end for
20: Update target parameters:
21: \( \vartheta(\vartheta + (1-\tau)\vartheta) \)
22: \( \Theta(\Theta + (1-\tau)\Theta) \)
23: \( T_{\text{update}} \leftarrow 0 \)
24: end if
25: end for
4. Experiments

4.1. Setup

In our experiment, we add connectivity constraint to the scenario of Liu et al. [20] and modify some reward set about coverage and fairness. Therefore, we design the cooperative navigation scenario of multi-UAV in the multi-agent particle environment framework introduced by Lowe et al. [5].

As shown in figure 1, N agents (UAVs) and M landmarks (base stations) are randomly generated in a certain area initially. Each agent has a uniform communication range. To simplify the environment, if landmark falls into the coverage range of any agents, it means that this landmark is covered. Meanwhile, we also consider fairness and collision, such as Jain's fairness index [20]. To define connectivity, we introduce the maximal connected subgraph in our environment in order to evaluate the current connectivity in each timeslot. If UAV is within the communication range of any other UAVs, we think they can communicate with each other. So the reward of each agent is defined as follows:

$$\text{reward} = \frac{N_{\text{covered}}^k}{N_{\text{landmark}}} * R_{\text{fairness}} * R_C - C_{\text{collision}}$$

where $N_{\text{covered}}$ and $N_{\text{landmark}}$ are the numbers of covered and total landmarks, respectively. $k$ is the hyperparameter to make the reward of covered non-linear. $R_{\text{fairness}}$ is the reward of Jain's fairness index to make the coverage timeslots of all UAVs as fair as possible. $R_C$ is the proportion of the maximal connected subgraph in all agents. Finally, $C_{\text{collision}}$ is a fixed penalty to prevent collision between agents.

Otherwise, we test our algorithm in the other environment. The Rover-Tower contains 4 “rovers” and 4 “towers”, and they are all agents. The rovers only know where they are but not destinations. The towers will tell the paired rovers about their destinations by one of five discrete communication messages. The pair is negatively rewarded by the distance of the rover to its destination.

Our experiments run with Pytorch 1.4 and Python 3.7 on Ubuntu 16.04 with 2*NVIDIA TITAN RTX GPUs. The attention network uses a hidden dimension of 128. The temperature parameter $\alpha$ in the policy gradient is 0.01 and the size of experience replay is $10^6$. Finally, we use 4 attention heads in the experiments.

![Figure 1](image.png)

**Figure 1.** Multi-UAV Cooperative Navigation. Gray circles are UAVs and purple circles are the coverage ranges. Their goal is to fairly cover as many base stations as possible and maintain some connectivity.

4.2. Results and Analysis

For $N = 5$ and $M = 20$ on Multi-UAV Cooperative Navigation (figure 2), we found that two algorithms will achieve the same result in the end, but MAAC reaches convergence around 5000 episodes while PER-MAAC only needs 2000 episodes. This means that PER-MAAC has accelerated training efficiency in the early stage of training due to more choices of valuable experiences.
Table 1 shows the average coverage, fairness and connectivity after 800 rounds of training. There is no significant improvement in coverage, but there are 2.8% and 8.2% of improvements in fairness and connectivity respectively.

| Algorithm   | Average coverage | Average fairness | Average connectivity |
|-------------|------------------|------------------|----------------------|
| MAAC        | 0.756            | 11.784           | 0.466                |
| PER-MAAC    | 0.756            | 12.119           | 0.504                |

Figure 3 illustrates the average rewards per 100 episodes in the scenario of Rover-Tower. We use Adam [21] as the optimizer for both with a learning rate of 0.004. In the initial 4000 episodes, the effect of prioritized experience replay is not obvious due to the relatively complex scene. As the number of experiences increases, the average reward of PER-MAAC starts to be gradually higher than MAAC and has a better stability. Both algorithms can reach the same reward in the end. This also reflects the effectiveness of PER-MAAC.

5. Conclusion
In this paper, we employ prioritized experience replay on MAAC and compare the experimental results with uniform experience replay. Intuitively, prioritized experience replay prefers to extract a more valuable experience according to TD-error to shorten the training time. We evaluate PER-MAAC algorithm in our experiments and show that it does indeed increase the convergence speed. Our environment of Multi-UAV Cooperative Navigation also makes sense for UAV simulation.

However, more issues should be further explored in the future. It would be meaningful to apply other methods of experience selection to MAAC. Furthermore, the Multi-UAV Cooperative Navigation scenario will further take into account the real physical factors.

Acknowledgement
This paper was partially supported by the National key R & D program of China sub project “Emergent behavior recognition, training and interpretation techniques” under grant No. 2018AAA0102302 and the project sponsored by the Ministry of Education of China under Grant No. 6141A02011803.
References

[1] Mnih V, et al. 2015 Human-level control through deep reinforcement learning Nature 518 (7540) 529-533.
[2] Silver D, et al. 2016 Mastering the game of Go with deep neural networks and tree search Nature 529 (7587) 484-489.
[3] Silver D, et al. 2017 Mastering the game of go without human knowledge Nature 550 (7676) 354-359.
[4] Tan M 1993 Multi-agent reinforcement learning: Independent vs. cooperative agents Proceedings of the Tenth International Conference on Machine Learning pp 330-337.
[5] Lowe R, et al. 2017 Multi-agent actor-critic for mixed cooperative-competitive environments Advances in Neural Information Processing Systems pp 6379-6390.
[6] Iqbal S and Sha F 2019 Actor-attention-critic for multi-agent reinforcement learning International Conference on Machine Learning.
[7] Vaswani A, et al. 2017 Attention is all you need Advances in Neural Information Processing Systems pp 5998-6008.
[8] De Bruin T, et al. 2018 Experience selection in deep reinforcement learning for control The Journal of Machine Learning Research 19 (1) 347-402.
[9] Hou Y, et al. 2017 A novel DDPG method with prioritized experience replay IEEE International Conference on Systems, Man, and Cybernetics (SMC) pp 316-321.
[10] Horgan D, et al. 2018 Distributed prioritized experience replay Proceedings of the International Conference on Learning Representations (ICLR).
[11] Andrychowicz M, et al. 2017 Hindsight experience replay Advances In Neural Information Processing Systems pp 5048-5058.
[12] Xu Z, et al. 2018 Experience-driven networking: A deep reinforcement learning based approach IEEE INFOCOM 2018-IEEE Conference on Computer Communications pp 1871-1879.
[13] Liu R and Zou J 2018 The effects of memory replay in reinforcement learning 56th Annual Allerton Conference on Communication, Control, and Computing (Allerton) pp 478-485.
[14] Foerster J, et al. 2017 Stabilising experience replay for deep multi-agent reinforcement learning Proceedings of the 34th International Conference on Machine Learning (ICML) pp 1146-1155.
[15] Wang Y and Zhang Z 2019 Experience selection in multi-agent deep reinforcement learning IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI) pp 864-870.
[16] Schaul T, et al. 2016 Prioritized experience replay International Conference on Learning Representations (ICLR).
[17] Littman M L 1994 Markov games as a framework for multi-agent reinforcement learning Machine Learning Proceedings pp 157-163.
[18] Haarnoja T, et al. 2018 Soft actor-critic algorithms and applications arXiv preprint 1812.05905.
[19] Haarnoja T, et al. 2018 Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor International Conference on Machine Learning (ICML) pp 1856-1865.
[20] Liu C H, et al. 2018 Energy-efficient UAV control for effective and fair communication coverage: A deep reinforcement learning approach IEEE Journal on Selected Areas in Communications 36 (9) 2059-2070.
[21] Kingma D P and Ba J 2015 Adam: A method for stochastic optimization International Conference on Learning Representations (ICLR).