Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnline CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 55-11 (2022) 19–24

Dynamic Path Planning for Mobile Robots with Deep Reinforcement Learning

Laiyi Yang * **Jing Bi** * **Haitao Yuan** **

* *(School of Software Engineering in Faculty of Information Technology, Beijing University of Technology, Beijing 100124, China (e-mail: yanglaiyi@emails.bjut.edu.cn; bijing@bjut.edu.cn).*

** *School of Automation Science and Electrical Engineering, Beihang University, Beijing 100191, China (e-mail: yuan@buaa.edu.cn)*

Abstract: Traditional path planning algorithms for mobile robots are not effective to solve high-dimensional problems, and suffer from slow convergence and complex modelling. Therefore, it is highly essential to design a more efficient algorithm to realize intelligent path planning of mobile robots. This work proposes an improved path planning algorithm, which is based on the algorithm of Soft Actor-Critic (SAC). It attempts to solve a problem of poor robot performance in complicated environments with static and dynamic obstacles. This work designs an improved reward function to enable mobile robots to quickly avoid obstacles and reach targets by using state dynamic normalization and priority replay buffer techniques. To evaluate its performance, a Pygame-based simulation environment is constructed. The proposed method is compared with a Proximal Policy Optimization (PPO) algorithm in the simulation environment. Experimental results demonstrate that the cumulative reward of the proposed method is much higher than that of PPO, and it is also more robust than PPO.

Keywords: Deep reinforcement learning, path planning, Soft Actor-Critic algorithm, continuous reward functions, mobile robots.

1. INTRODUCTION

With the emergence of Internet of Things, big data, and artificial intelligence technologies, mobile robots are widely used in automatic production, mining, household services, agriculture and other fields. Path planning of mobile robots provides optimized paths from initial positions to target ones in an environment while avoiding static and dynamic obstacles Patle *et al.* (2019). At present, mobile robots are evolving towards self-learning and adaptive intelligence. With the widespread use of mobile robots, designing intelligent, accurate and efficient path planning algorithms is one of urgent tasks Gasparetto *et al.* (2015). In recent years, researchers propose many methods to solve such problems. Classical methods include traditional algorithms, graphics ones and bionics intelligent ones Chen *et al.* (2019). Nevertheless, they are still insufficient to exploit intelligent path planning for mobile robots due to many challenges.

First, traditional methods mainly include simulated annealing (SA) , artificial potential field Chen *et al.* (2016) and fuzzy logic Song *et al.* (2020). These methods are easy to be implemented. However, their modeling process is complicated, and prior knowledge and global information cannot be effectively used. Besides, it is easy to fall into local optima and cannot find global ones. SA has high operating efficiency and requires few initial conditions. However, its convergence speed is too slow, and the parameter setting requires researchers to have practical experiences. The artificial potential field method is easy to be described, and extremely few computations. Thus, its search speed is very fast. However, its solution is often a locally optimal one. The fuzzy logic method conforms to thinking habits of human beings. It does not require mathematical modeling, and has good stability. However, its adaptability is often poor.

Second, graphics methods mainly include the A* algorithm Duchon *et al.* (2014), the Voronoi diagram method Candeloro *et al.* (2017). Compared with traditional methods, they are simple to be modeled. However, its low search efficiency makes it difficult to be widely used. The A* algorithm is suitable for static environments and known ones, yet it cannot be applied to unknown environments. The Voronoi diagram method can effectively avoid obstacles by enclosing them in elements. However, they are not suitable for large-scale dynamic environments.

Third, bionics intelligent algorithms mainly include genetic algorithm Lamini *et al.* (2018), colony algorithm Akka *et al.* (2018), and particle swarm optimization algorithm Li *et al.* (2018). Compared with the graphics methods, these algorithms are more intelligent and have higher search efficiency. However, it needs to carefully consider the relation between specific parameters to obtain the

⋆ This work was supported by the Educational and Teaching Research Project in Beijing University of Technology under Grant ER2022KCB05, the Educational Research Project in Beihang University under Grant 4003213, the Fundamental Research Funds for the Central Universities under Grant YWF-22-L-1203, and the National Natural Science Foundation of China (NSFC) under Grants 62073005 and 62173013.

10.1016/j.ifacol.2022.08.042

optimal solution. In addition, these algorithms are prone to fall into local optima and suffer from slow convergence.

The research on above-mentioned path planning algorithms achieves great success. However, it is not adaptable to dynamically changing environments. It also lacks the ability of environment perception and learning, and it cannot process complex and high-dimensional environmental information. Therefore, it is difficult to perform high-quality path planning in the absence of prior knowledge and dynamic environments. The reinforcement learning algorithm does not need to establish such an environment model. Mobile robots continuously interact with the environment to generate data, and constantly learn by trials and errors, so as to learn strategies to realize path planning. The path planning algorithms of reinforcement learning include Q-learning Low *et al.* (2019); Konar *et al.* (2013); Das *et al.* (2016), SARSA Harwin *et al.* (2019); Xu *et al.* (2017), Deep Q-learning Network (DQN) Yang *et al.* (2020); Wen *et al.* (2020); Zhang *et al.* (2022). Path planning algorithms based on Q-learning and SARSA solve the problems of difficult modeling and insufficient prior knowledge. Yet they cannot solve high-dimensional problems. Although the DQN algorithm solves the problem of high dimensionality, it is only limited to problems with discrete action spaces, and it cannot solve that with continuous action spaces.

Therefore, it is necessary to propose an algorithm that does not depend on the prior knowledge, and has strong robustness and can be used for continuous action space problems. To achieve it, this work proposes an obstacle avoidance method for mobile robots based on the Soft Actor-Critic (SAC) algorithm. In summary, major contributions include:

(1) This work proposes an obstacle avoidance algorithm based on SAC, so that mobile robots can autonomously avoid static and dynamic obstacles with continuous numerical values without the prior knowledge.
(2) This work combines the potential energy function with the artificial potential field method to design an improved reward function, and it can guide mobile robots to successfully reach the target through autonomous learning.
(3) This work verifies that states and actions are normalized, and experiences are prioritized to provide faster robot decision-making that Proximal Policy Optimization (PPO).

2. PROPOSED ALGORITHM

SAC is a deep reinforcement learning algorithm that combines the actor-critic algorithm, the maximum entropy model and the offline policy. It is based on the Deep Deterministic Policy Gradient (DDPG), the stochastic strategy and the maximum strategy entropy. There are three advantages over other reinforcement learning algorithms such as PPO and DDPG in dealing with complex tasks.

(1) It has strong exploration ability. SAC can control the ratio of the maximum entropy to the reward, which is positively correlated with the exploration ability. Therefore, mobile robots can more easily find the optimal solution with multiple reward functions. In this work, a mobile robot is required to both reach the target and to avoid both static and dynamic obstacles.

(2) SAC's learnt strategies can be easily transferred to other scenarios. Strategies learned through the maximum entropy can be applied to solve similar types of tasks.

(3) SAC has strong robustness. SAC uses the stochastic sampling, and it can adjust more easily according to disturbances. To prevent mobile robots from falling into local optima and to encourage them to explore more spaces, SAC requires them to maximize both the cumulative reward and the maximum entropy. The mobile robots tend to explore by increasing the action entropy to avoid falling into a local optimum. By increasing the cumulative reward, mobile robots move towards their target.

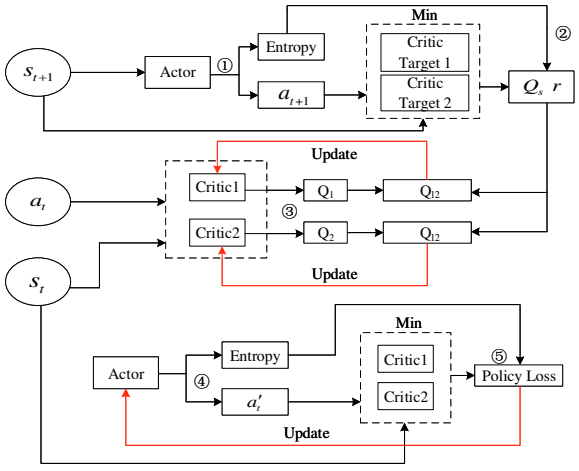

Fig. 1. Network structure in SAC

Then, Fig. 1 shows the network structure in SAC. As shown in the part ① in Fig. 1, the action entropy is derived from the output of the actor network, which is calculated as:

$$H(\pi(\cdot \mid s_{t+1})) = -\log t\pi(a_{t+1} \mid s_{t+1}) \tag{1}$$

where a_{t+1} denotes the action output of the actor network from the environment state s_{t+1}, and $\pi(\cdot \mid s_{t+1})$ is the probability of outputting a_{t+1} from s_{t+1}.

(1) Action entropy

From the part ② in Fig. 1, it is shown that the value estimation of the target network includes the action entropy, which is calculated as:

$$\begin{aligned} &Q_{\mathrm{s}}(r, s_{t+1}) = r + \lambda(V(s_{\mathrm{t+1}})) = \\ &r + \lambda(Q_\pi(s_{t+1}, a_{t+1}) + \alpha H(\pi(\cdot \mid s_{t+1}))) = \\ &r + \lambda(\min_{j=1,2} Q_{\varphi_{tj}}(s_{t+1}, a_{t+1}) - \alpha \log \pi(a_{t+1} \mid s_{t+1})) \end{aligned} \tag{2}$$

where $V_{s_{t+1}}$ denotes the state value of s_{t+1}, r is from the replay buffer, $Q_\pi(s_{t+1}, a_{t+1})$ is the estimation of the action value of a_{t+1} under s_{t+1}, α is the weight of entropy, φ_{tj} is the parameter of two critic target networks, and $\min_{j=1,2} Q_{\varphi_{tj}}$ indicates taking the minimum value of the output of two target networks, which can effectively prevent the overestimation.

(2) Critic network

From the part ③ in Fig. 1, it is known that SAC has two critic networks with the same structure as the critic target one. The loss function $L(\varphi_i, D)$ of two networks is given as:

$$L(\varphi_i, D) = E_{(s_t,r,s_{t+1},a_t)-D}[(Q_{\varphi_i}(s_t,a_t) - Q_s(s_{t+1}))^2] \quad (3)$$

where $E_{(s_t,r,s_{t+1},a_t)-D}$ means that (s_t, r, s_{t+1}, a_t) comes from the replay buffer D, and $Q_{\varphi_i}(s_t, a_t)$ is the Q-value estimate of the network with the weight φ_i for an agent performing action a_t when the environmental state is s_{t+1}.

(3) Actor network

The update formula for the actor-network is given as:

$$\max_{\theta} E_{s-D}[\min_{j=1,2} Q_{\varphi_j}(s_t, a'_t) - \alpha \log \pi_\theta(a'_t \mid s_t)] \quad (4)$$

where θ denotes the weight of the actor-network, α denotes the reward coefficient of entropy, which is the degree of importance of entropy.

(4) Critic target network

Then, we update each critic target network in proportion to the parameter ϕ, and this process is a sliding average update. The update is given as:

$$\phi_{ti} \leftarrow \rho\phi_{ti} + (1-\rho)\phi_i, \quad i \in \{1, 2\} \quad (5)$$

The general process of SAC can be summarized as follows:

(1) The current state s_t of each mobile robot is the input to the actor network, which calculates the mean and variance of the Gaussian distribution based on the input state, and samples the output action a_t according to the probability distribution.

(2) Each mobile robot executes the action a_t, and obtains the reward r_{t+1} from the environment. The state is changed to s_{t+1}. Then, the current experience is put into the replay buffer.

(3) When the number of experiences reaches a certain threshold, a batch of experiences are sampled from the replay buffer to update the critic network, the actor one and the critic target one according to (3), (4) and (5). The above process is repeated until the target network converges.

Algorithm 1 shows the pseudo codes of the proposed SAC algorithm. In Algorithm 1, the critic network is updated by (6), and the actor one is updated by (7).

$$\nabla_{\varphi_i} \frac{1}{|B|} \sum_{(s_t,a_t,r,s_{t+1})\in B} (Q_{\varphi_i}(s_t,a_t) - Q_s(r, s_{t+1}))^2 \quad (6)$$

$$\nabla_\theta \frac{1}{|B|} \sum_{s\in B} (\min_{j=1,2} Q_{\varphi_i}(s_t, a'_t) - \alpha \log \pi_\theta(a_t \mid s_t)) \quad (7)$$

Algorithm 1 Soft Actor Critic algorithm

```
Initialize parameters of SAC and that of each network
for each iteration do
    for each environment step do
        A mobile robot in the environment state s_t executes action a_t from the actor network and obtains a reward r_{t+1}. The state is changed to s_{t+1}. Put this experience (s_t, a_t, s_{t+1}, r_{t+1}) into the replay buffer.
    end for
    if it is time to update then
        for each gradient step do
            Gain a certain number of experiences B(s_t, a_t, s_{t+1}, r_{t+1})) from the replay buffer.
            The critic target network estimates Q values with (2)
            Update critic network by (6)
            Update actor network by (7)
            Update critic target network by (5)
        end for
    end if
end for
```

3. SYSTEM MODEL DESIGN

3.1 Three key factors of RL environments

(1) System state

In the Markov decision process (MDP), the state information represents environmental information and dynamic changes sensed by mobile robots, and it is basis for deep reinforcement learning (DRL) algorithms to make decisions and evaluate the cumulative reward. The quality of the state design directly determines the convergence speed and the final performance of DRL algorithms. The state information in this work includes real-time location information, velocity and moving direction of each mobile robot, location information of each dynamic obstacle, and that of the target. To reduce the dimension of neural networks, speed up the training of networks, and improve applicability of networks, the absolute location information of each obstacle is transformed into the relative distance between it and each mobile robot. Specifically, the system state is defined as:

$$s=\{l_{agent}, d_{obstacle}, d_{target}, v_{agent}, v_{obstacle}\} \quad (8)$$

where l_{agent} denotes the location vector of each mobile robot, $d_{obstacle}$ denotes the relative distance vector between each obstacle and each mobile robot, d_{target} denotes the relative distance vector between the target and each mobile robot, v_{agent} denotes the moving velocity of each mobile robot, and $v_{obstacle}$ denotes the moving velocity of each obstacle.

(2) System action

Traditional path planning algorithms in DRL usually transform a continuous action space into discrete one. They often divide the 360-degree direction into four ones, *i.e.*, east, west, south, and north. It is worth noting that this is quite different from the actual situation. In this work, the action space is designed as a continuous variable including the action speed and direction. Thus, the robot can move in any direction within 360 degrees, and it is more suitable for the actual situation. Let a denote a velocity vector. v denotes the amount of velocity, and d denotes the direction of velocity. Specifically, a=[v, d], $v\in$[0, 10], and $d\in$[0, 2π].

(3) Reward function

The reward function design is the most important part in DRL . The essence of a learning process of RL is the training of neural networks guided by the reward function. Thus, the design of the reward function determines whether mobile robots can learn the expected

policy, and directly affects the convergence speed and the final performance of DRL algorithms. This work designs the reward for mobile robots to find the target as the main reward. It is worth noting that the reward space is sparse, and mobile robots might have difficulty in learning due to the lack of feedback signals, and it easily fails to find the target. Therefore, this work adds other reward items or penalty ones to make the reward function denser, and guide mobile robots to explore more efficiently in the environment, thereby speeding up the convergence. Specifically, the reward signal is a scalar, with positive values representing rewards and negative ones representing penalties. r_{total} denotes the reward function in this work, which is obtained as:

$$r_{total}=r_1+r_2+r_3+r_4 \tag{9}$$

where r_1 denotes the reward for guiding mobile robots to learn correct angles, r_2 denotes that for guiding mobile robots to quickly find the target, r_3 denotes that for guiding mobile robots to avoid the collision with obstacles, and r_4 denotes that for reflecting the termination cases.

a) To guide mobile robots to learn correct angles, the velocity direction of each mobile robot needs to be the same as the distance vector between it and the target. Thus, r_1 is defined as:

$$r_1=\cos(l_{vector}, v) \tag{10}$$

where l_{vector} denotes a vector from the location of each mobile robot to the target location, and v denotes a vector of each robot.

b) To guide mobile robots to quickly find the target, each mobile robot needs to obtain a positive reward when it moves towards the target, and a negative penalty when it moves away from the target. Thus, r_2 is defined as:

$$r_2=-\frac{d}{500}\times 0.4+1.0+\frac{pre_d-0.95\times d}{24} \tag{11}$$

where pre_d denotes the distance between each robot and the target in the previous iteration, and d denotes that between each mobile robot and the target in current iteration.

c) To guide mobile robots to avoid the collision with obstacles, each mobile robot needs to dynamically adjust its moving direction for avoiding the collision with obstacles in the moving process. For each mobile robot, if there are obstacles within its five meters, it is in a dangerous area; otherwise, it is in a safe one. Specifically, when each mobile robot finds that there are obstacles within five meters in its moving direction, the environment gives a negative reward of -1 to alert it to adjust its direction. In addition, when it moves from a dangerous area to a safe one, the environment gives it a negative reward of 0.5.

d) The termination cases include three results: first, the reward is -5 when a mobile robot reaches the target; second, the reward is -6 when a mobile robot collides with a obstacle; third, the reward is 100 when a mobile robot reaches a border of the search area.

4. PERFORMANCE EVALUATION

4.1 Simulation environment and setting

To evaluate the performance of the proposed method, this work constructs a simulation environment for obstacle avoidance based on Pygame, as shown in Fig. 2. The area of Fig. 2 is 400×400. A red square represents a mobile robot, a blue one represents a static obstacle, and a white one represents a moving obstacle, and a yellow one represents the target.

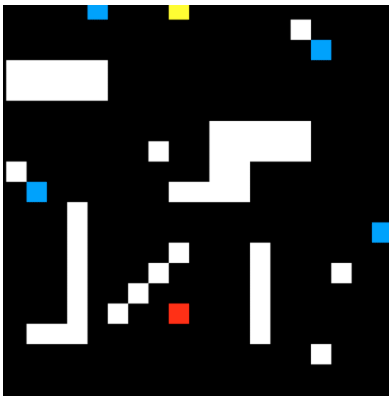

Fig. 2. Simulation environment.

This work adopts an open-sourced RLlib library to train our proposed method. This work adopts a fully connected layer with 256×256×256. Tanh is used as an activation function to transform an action space into the range of [-1, 1]. The operating system is Ubuntu1 16 running on a server with the Intel Xeon(R) CPU E5-2650 v4 at 2.20GHz. The work considers the reward in the future 20 steps. The setting of parameters of SAC is given in Table 1.

Table 1. Parameter setting of SAC

Parameter	Setting
Discount	0.95
Learning rate	0.0003
Number of samples per minibatch	256
Replay buffer size	10^7
Target smoothing coefficient	0.005
Initial value of α	1.0
Gradient steps	1
Target update interval	1
Exploration function	Stochastic sampling

4.2 Simulation results

The maximum cumulative reward value in each iteration for mobile robots is shown in Fig. 3. The purpose of mobile robots aims to improve the given reward value of the environment through continuous learning for obtaining the maximum reward. Therefore, the larger reward value, the better the learning effect of mobile robots. In Fig. 3, the maximum reward values of SAC and PPO show a gradually increasing trend. After about 40 iterations of SAC, its reward value is greater than 800. It indicates that mobile robots have learned a better policy. Although the reward value of PPO shows a gradual upward trend, its fluctuation is larger than that of SAC. More importantly, the maximum reward value of PPO is around 600, while that of SAC is as high as 820. Therefore, the learnt policy of SAC is better than PPO.

The average award in each iteration can reflect the stability of the proposed SAC algorithm. Fig. 4 shows the aver-

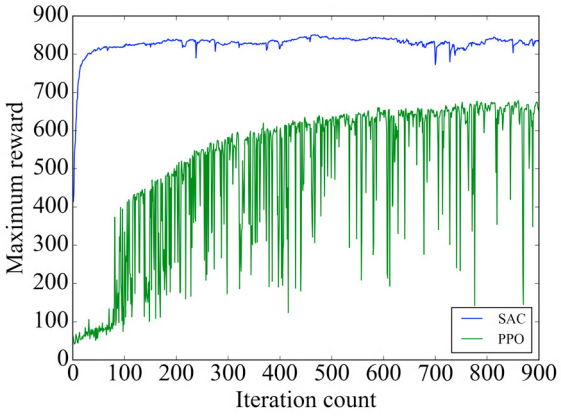

Fig. 3. Maximum reward with SAC and PPO

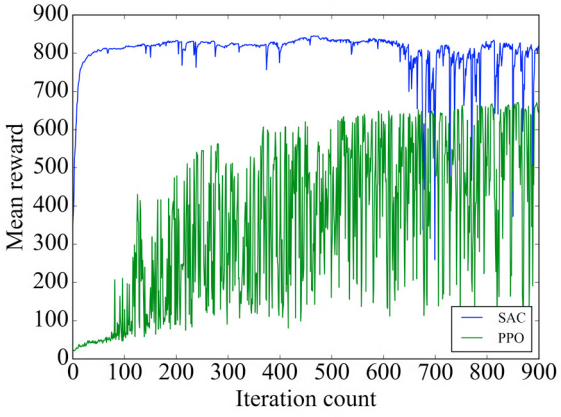

Fig. 4. Average reward with SAC and PPO

age award in each iteration. It is observed that the average award of SAC increases stably while that of PPO varies significantly. The result shows that SAC well achieves a better balance between exploration and exploitation, and it does not lose experiences with larger reward. In addition, SAC has larger average reward, and therefore, it has better robustness.

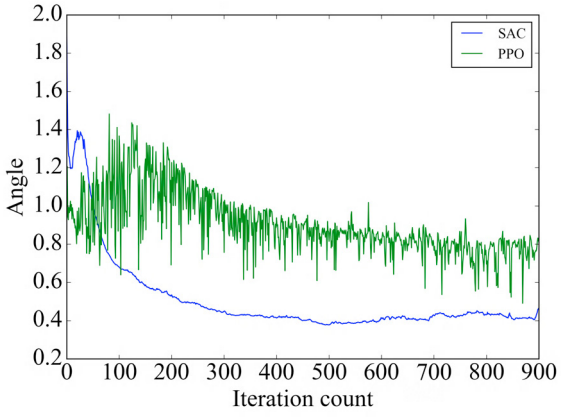

Fig. 5. Angles learnt with SAC and PPO

SAC also outperforms PPO with respect to the velocity, angle and the number of steps for each mobile robot. The angle shows the difference between the moving direction and that towards the target of each mobile robot. The

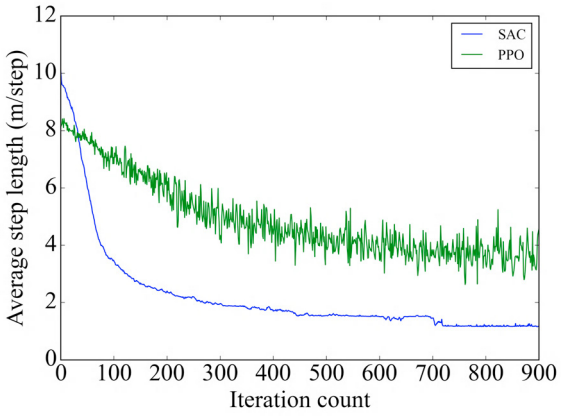

Fig. 6. Average length of each moving step with SAC and PPO

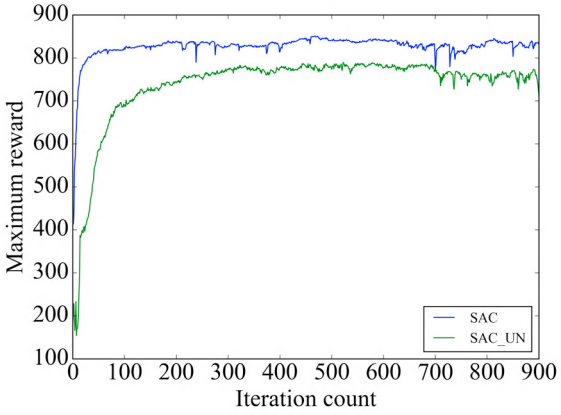

Fig. 7. Number of steps with SAC and PPO

learning of angles also shows the learning result of SAC. As shown in Fig. 5, the angle learnt by SAC is about 0.4 while that of PPO is about 0.8. Thus, the learning process of SAC is more stable, and learns the correct direction in a faster way than PPO. In addition, as shown in Figs. 6 and 7, the learnt speeds of SAC and PPO both decrease gradually. The learnt speed of SAC is about 1.5, which is lower than that (4) of PPO. Therefore, SAC needs more steps to finally arrive to its target. However, as shown in Figs. 5 and 6, the learnt policy of SAC is more stable than that of PPO.

The maximum cumulative reward reflects the best performance of the proposed SAC algorithm. As shown in Fig. 8, the effectiveness of algorithm learning is more influenced by normalizing the states and using the prioritization experiences. Here, SAC_UN means the original SAC without using state normalization and priority experiences. In the first 20 iterations, the maximum cumulative reward of SAC_UN fluctuates and only gradually increases. SAC adopts the state normalization and prioritized experiences, and it keeps the state vector normally distributed and yields experiences that have weights, and it achieves better learning results. It is also shown that there is a gradual increase from the very beginning, and there is there is a large fluctuation only until the 200th iteration. Therefore, SAC that adopts state normalization and prioritization

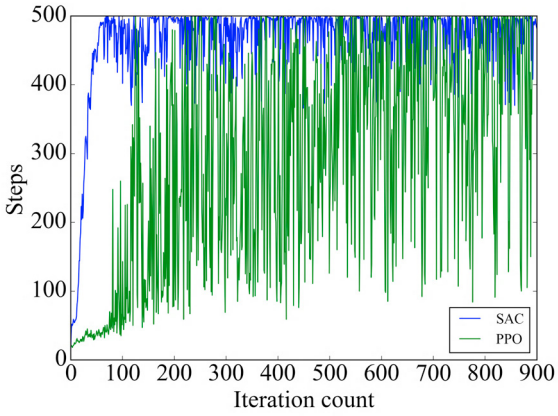

Fig. 8. Maximum reward value with SAC and SAC_UN

experiences learns better than SAC_UN that does not adopt these acceleration tricks.

5. CONCLUSION

This work proposes a path planning algorithm for mobile robots based on the Soft Actor-Critic (SAC) algorithm in a complex environment with static and dynamic obstacles. Mobile robots can continuously perform actions to avoid obstacles and finally reach the final target. Compared with the Deep Q-learning Network (DQN) method, the proposed method is more suitable in the complex environment and characteristics of mobile robots. According to the potential energy function of the artificial potential field method, a reasonable reward function is designed, and the obstacles are divided into static and dynamic categories. Then, this work considers the collision, steering and other situations to make the output action more reasonable and effective. To speed up the training, a Pygame-based simulation environment is constructed by adopting the priority replay buffer and the dynamic state normalization technology to verify the effectiveness of the proposed algorithm. Experimental results demonstrate that the proposed path planning performance for mobile robots outperforms that the proximal policy optimization, and the original SAC without using state normalization and priority experiences.

Our future work will focus on the following aspects. First, we will further consider more real factors such as acceleration speeds of mobile robots, and the friction of the environment. Second, we will further adopt convolutional neural networks to replace the fully connected network for yielding better path planning performance.

REFERENCES

B. K. Patle, A. Pandey, D. R. K. Parhi, and A. Jagadeesh. A review: On path planning strategies for navigation of mobile robot, *Advances in Enzymology*, 15(4), pp. 582–606, 2019.

A. Gasparetto, P. Boscariol, A. Lanzutti, and R. Vidoni. Path planning and trajectory planning algorithms: A general overview. *Motion and operation planning of robotic systems*, pp. 3–27, 2015.

C. Chen, X. Q. Chen, F. Ma, X. J. Zeng, and J. Wang. A knowledge-free path planning approach for smart ships based on reinforcement learning. *Ocean Engineering*, pp. 189, 106299, 2019.

Y. B. Chen, G. C. Luo, Y. S. Mei, J. Q. Yu, and X. L. Su. UAV path planning using artificial potential field method updated by optimal control theory. *Int. Journal of Systems Science*, 47(6), pp. 1407–1420, 2016.

Q. Song, Q. Zhao, S. Wang, Q. Liu, and X. Chen. Dynamic path planning for unmanned vehicles based on fuzzy logic and improved ant colony optimization. *IEEE Access*, 8, pp. 62107–62115, 2020.

F. Duchon, A. Babinec, M. Kajan, P. Beno, M.Florek, T. Fico, and L. Jurisica. Path planning with modified a star algorithm for a mobile robot. *Procedia Engineering*, 96, pp. 59–69, 2014.

M. Candeloro, A. M. Lekkas, Sørensen, and A. J. A Voronoi-diagram-based dynamic path-planning system for underactuated marine vessels. *Control Engineering Practice*, 61, pp. 41–54, 2017.

C. Lamini, S. Benhlima, and A. Elbekri. Genetic algorithm based approach for autonomous mobile robot path planning. *Procedia Computer Sci.*, 127, pp. 180–189, 2018.

K. Akka, and F. Khaber. Mobile robot path planning using an improved ant colony optimization. *International Journal of Advanced Robotic Systems*, 15(3), pp. 851–856, 2018.

G. Li, and W. Chou. Path planning for mobile robot using self-adaptive learning particle swarm optimization. *Science China Information Sci.*, 61(5), pp. 1–18, 2018.

E. S. Low, P. Ong, and K. C. Cheah. Solving the optimal path planning of a mobile robot using improved Q-learning. *Robotics and Autonomous Systems*, 115, pp. 143–161, 2019.

A. Konar, I. G. Chakraborty, S. J. Singh, L. C. Jain, and A. K. Nagar. A deterministic improved Q-learning for path planning of a mobile robot. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 43(5), pp. 1141–1153, 2013.

P. K. Das, H. S. Behera, and B. K. Panigrahi. Intelligent-based multi-robot path planning inspired by improved classical Q-learning and improved particle swarm optimization with perturbed velocity. *Engineering science and technology*, 19(1), pp. 651-669, 2016.

L. Harwin and P. Supriya. Comparison of SARSA algorithm and temporal difference learning algorithm for robotic path planning for static obstacles. In *2019 IEEE Third International Conference on Inventive Systems and Control (ICISC)*, pp. 472-476, 2019.

D. Xu, Y. Fang, Z. Zhang, and Y. Meng. Path planning method combining depth learning and Sarsa algorithm. In *2017 IEEE 10th Int. Symposium on Computational Intelligence and Design*, 2, pp. 77-82, 2017.

Y. Yang, L. Juntao, and P. Lingling. Multi-robot path planning based on a deep reinforcement learning DQN algorithm. *CAAI Transactions on Intelligence Technology*, 5(3), pp. 177-183, 2020.

S. Wen, Y. Zhao, X. Yuan,Z. Wang, D. Zhang, and L. Manfredi. Path planning for active SLAM based on deep reinforcement learning under unknown environments. *Intelligent Service Robotics*, 13(2), pp. 263-272, 2020.

F. Zhang, C. Gu, and F. Yang. An improved algorithm of robot path planning in complex environment based on double DQN. In *Advances in Guidance, Navigation and Control*, Springer, Singapore, pp. 303-313, 2022.