Paper:

An Improved Q-Learning Algorithm for Mobile Robot Path Planning

Junkui Wang, Kaoru Hirota, Xiangdong Wu, Yaping Dai, Zhiyang Jia*

School of Automation, Beijing Institute of Technology
No.5 Zhongguancun South Street, Haidian District, Beijing 100081, China
E-mail: 3220180655@bit.edu.cn

[Received 00/00/00; accepted 00/00/00]

Abstract. An improved Q-learning (IQL) combined with Q-function initialization method, action selection strategy and reasonable reward function is proposed. The Q-function is initialized using the inverse Euclidean distance to update the entries in the Q-table efficiently and reduce the episodes significantly. And an improved ε-greedy exploration which combined with Boltzmann exploration and heuristic searching strategies is provided to reduce the search space and limit the variation range of orientation angle. In addition, a reasonable reward distribution is designed to reduce the torque generated by the robot in planning. Experiments undertaken on grid environment confirm that the Q-table obtained by the proposed initialization method outperforms both the classical initialization and the method based on neural network. The experimental results show that the proposed algorithm can keep a safe distance from obstacles and requires less torque in comparison to its classical counterpart.

Keywords: Mobile Robot, Path Planning, Reinforcement Learning, Q-Learning.

1. Introduction

According to the information of the environment, the path planning is divided into two categories, global path planning and local path planning. There are many local path planning methods, such as path planning methods based on fuzzy logic [1], neural networks (NN) [2], ant colony algorithms (ASO) [3] and reinforcement learning (RL) [4, 5].

RL is an important branch of Machine Learning (ML), which is mainly used to solve sequential decision-making problems. Q-learning (QL) is a type of reinforcement learning algorithms developed by Watkins. QL is used to solve the mobile robot path planning problems. In [6], a novel path planning method based on improved Q-learning algorithm and some heuristic searching strategies is proposed for mobile robots in a dynamic environment. In [7], an online pheromone stringency guiding heuristically accelerated Q-learning algorithm is presented to speed up the convergence rate. Q-learning can be improved through appropriate initialization of Q-values. Several methods have been used in this regard. Wiewiora use potential-based shaping function to initialize Q-values to update the Q-table [8]. Song et al. successfully improve the speed of convergence and stability of QL through applying dynamic wave expansion neural network into initialization of Q-values [9]. Simsek et al. introduce a new cost function for initialization of Q-table in order to overcome the slow convergence rate of QL [10]. Yan et al. initialize the Q-table using the inverse Euclidean distance between the current and the target position to accelerate the learning efficiency of QL [11].

In our research, we improve the classical Q-learning (CQL) algorithm, called the improved Q-learning (IQL) for increasing its performance in the path-planning problem. In IQL, the Q-function is initialized using the inverse Euclidean distance, which is efficiently used to update the entries in the Q-table, instead of repeatedly updating them like the CQL. And a new exploration strategy is proposed to select action at states, which combines ε-greedy exploration with Boltzmann exploration and some heuristic searching strategies. Also, a reasonable reward value distribution is designed to make the trajectory keep a safe distance from obstacles and requires less torque in comparison to its classical counterpart.

Different from the artificial potential field method, the inverse Euclidean distance used in the initialization method in this paper is used to construct the potential field. After establishing the potential field function for each state, we also use the improved Bellman equation to update the Q-value of each state-action pair in the Q-table, in order to apply the potential field function to the Q-table. At the same time, this paper combines a heuristic exploration strategy with random exploration, and combines it with Boltzmann exploration to form an improved ε-greedy strategy. In addition, for those actions that produce bad results, we also apply additional reward functions to them to reduce the probability of these actions being selected.

The simulation experiment is designed and completed on MATLAB. The IQL proposed in this paper will be compared with CQL, SARSA method and another path planning based on neural network to achieve Q learning in a grid environment.

The rest of this paper is organized as follows. The CQL is described in 2. The IQL based path planning algorithm list in 3. The experimental results and analysis are presented in 4.

2. Classical Q-Learning

2.1. Q-Learning

The terms used in the Q-learning algorithm involve state, action, agent, and reward. All possible states of an agent and its possible actions in a given state are deterministically known. In other words, for a given agent A, let $S_1, S_2, ..., S_n$ be n possible states, where each state has m possible actions $a_1, a_2, ..., a_m$. At a particular state-action pair, the specific reward that the agent acquires is known as immediate reward. Let $r(S_i, a_j)$ be the immediate reward that the agent A acquires by executing an action a_j at state S_i. Agent learns an optimal strategy to maximize cumulative rewards obtained from the environment. The agent selects its next state from its current states by using a policy.

The Q-values of the QL algorithm is updated by solving the Bellman equations:

$$\begin{aligned} Q(s_t, a_t) =& (1-\alpha)Q(s_t, a_t) + \\ & \alpha(r_{t+1} + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1})) \end{aligned} \tag{1}$$

where $Q(s_t, a_t)$ is the state-action pairs and their corresponding reward value, s_t is the current state, a_t is the current action, a_{t+1} is the subsequent action, s_{t+1} is the state after performing a_t, r_{t+1} is the immediate reward. α is the learning rate, which determines the range of the latest received data that will override the previous value. When $\alpha = 0$, the agent will learn nothing, whereas the agent will only read the latest received information when $\alpha = 1$. According to Watkins, when the value of α is relatively small, Q-value in the function $Q(s, a)$ will eventually converge to optimal Q-value. Discount factor γ is a parameter to be considered in the QL to determine the type of reward that the agent receives. When $\gamma = 0$, the agent could only consider immediate reward, whereas the agent will consider future reward when γ approaches 1.

2.2. Q-Learning in Path Planning Problem

In solving the path planning problem, rasterizing the environment space is one of the most commonly used methods. Assuming that there are some static and indeterminate obstacles in the environment. In order to simplify the problem, assume that the robot moves with fixed speed and the robot could be regarded as a point every time in the experiment.

A state is the grid position where the agent defines its location in the environment, whereas an action is the movement that the agent takes to move from one state to another state. To decrease the distance of the path and the orientation angle of the robot, 8 actions are adopted in this paper. As shown in **Fig. 1**, action 1, 2, ... , 8 denote the up, right-up, ... , left-up, respectively.

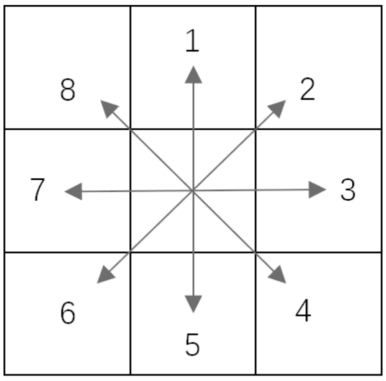


Fig. 1. 8 Discrete Actions.

For a reward, it is the positive value given to increase the Q-value when the correct action taken by the agent at the particular state, whereas it is the negative value given to decrease the Q-value when the wrong action taken by the agent.

2.3. Problem Statement

At the beginning of learning, the agent knows little about environment state and selects actions from action set randomly and blindly. The agent has no choice but to perform random selection during the early stage of learning since all Q-values are initialized to zeros in the classical Q-learning (CQL). Therefore, the calculation efficiency of CQL is relatively low.

In addition, CQL only sets rewards for performing actions, collisions, and reaching goal state. In order to reduce the execution of actions, CQL would learn the shortest path. However, in some environments with local minima, the shortest path may not be the optimal path. And the shortest path will be close to the obstacle most of the time, keeping an unsafe distance from the obstacle.

3. Improved Q-Learning

This section provides three strategies in order to improve the efficiency of CQL algorithm in the path planning problem. The details of the strategies are introduced as follows.

3.1. Q-function Initialization Method

Using priori knowledge to initialize Q-function is one of the important means to improve the convergence speed of QL algorithm [12].

In the artificial potential field method, the potential field is usually created based on the Euclidean distance between each position and goal position. Therefore, moving to the goal position can be thought as moving robot from a high-value state to a low-value state by following a “downhill” path. Mathematically speaking, this is the process of following gradient-descent. The initialization method proposed in this article draws on the idea of artificial potential field method. However, moving to the goal position in QL is to move the robot from a low Q-value to a high Q-value, which can be regarded as "uphill".

Therefore, the Q table can be initialized by inverse Euclidean distance. The potential function can be constructed as:

$$U(s) = U_{\text{att}}(s) + U_{\text{rep}}(s) \tag{2}$$

where the attractive potential function $U_{\text{att}}(s)$ is the inverse Euclidean distance between each position and goal position, and the repulsive potential function $U_{\text{rep}}(s)$ is the negative value of the inverse Euclidean distance to the nearest obstacle. The grid environment initialized with inverse Euclidean distance is shown in **Fig. 2**.

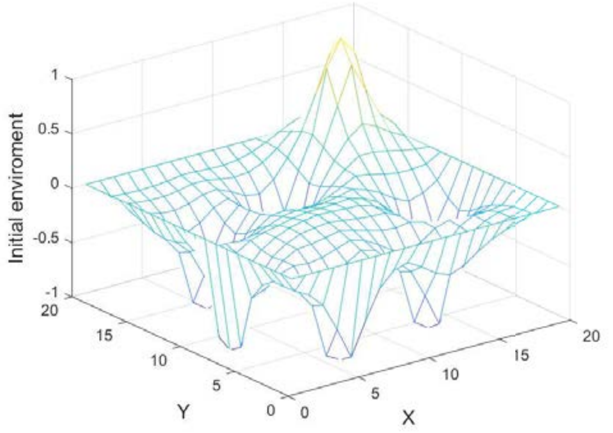


Fig. 2. Initial Grid Environment.

After obtaining the grid environment based on the inverse potential field, it is also necessary to initialize the Q table through the following expression:

$$Q(s_t, a_t) = r_{t+1} + \gamma U(s_{t+1}) \tag{3}$$

This initialization method can reduce the blindness in the learning and exploration, so as to accelerate the training efficiency of QL. As a result, the robot is no longer starting from zero information of its environment, but utilizing the priori knowledge to its exploration process, thus improving the computational complexity and convergence rate.

3.2. Action Selection Strategy

Performance of a reinforcement learning algorithm is greatly influenced by two important factors used in the control strategy of the algorithm, popularly known as "exploration" and "exploitation". Exploration usually refers to selecting any action with non-zero probability in every encountered state to learn the environment by the agent. Exploitation is targeted at employing the current knowledge of the agent to expect achieving good performance by selecting greedy actions [13]. In general, exploration is used at the beginning and exploitation is used at the end of the learning process. This is because exploration involves the selection of random actions, while the agent learns without considering the current state in order to visit all the state-action pair in the environment. Exploitation involves the knowledge from the agent to be applied in choosing actions to maximize the reward of the current state[14].

One classical method to balance exploration and exploitation in QL is ε-greedy exploration, where a parameter ε representing exploration probability is introduced to control the ratio between exploration and greedy action selection.

3.2.1. Heuristic Searching Strategies

During the initial stage of exploration, the motion of agent is completely random, resulting in wastage of computational effort, slower convergence rate and time-consuming. To accelerate the learning process, in this paper, we introduce some Heuristic Searching Strategies (HSS). Random action is required to satisfy the heuristic searching strategies. Assume that (x_c, y_c) and (x_g, y_g) denote the current state and goal state, respectively. The details of the heuristic searching strategies rules are listed as **Alg 1** .

Algorithm 1: Heuristic Searching Strategies

```
Input: current state $(x_c, y_c)$, goal state $(x_g, y_g)$ ;
Output: action ;
1  if $x_c == x_g$ and $y_c < y_g$ then
2      action = randSelection [8 1 2] ;
3  end
4  else if $x_c < x_g$ and $y_c < y_g$ then
5      action = randSelection [1 2 3] ;
6  end
7  else if $x_c < x_g$ and $y_c == y_g$ then
8      action = randSelection [2 3 4] ;
9  end
10 else if $x_c < x_g$ and $y_c > y_g$ then
11     action = randSelection [3 4 5] ;
12 end
13 else if $x_c == x_g$ and $y_c > y_g$ then
14     action = randSelection [4 5 6] ;
15 end
16 else if $x_c > x_g$ and $y_c > y_g$ then
17     action = randSelection [5 6 7] ;
18 end
19 else if $x_c > x_g$ and $y_c == y_g$ then
20     action = randSelection [6 7 8] ;
21 end
22 else if $x_c > x_g$ and $y_c < y_g$ then
23     action = randSelection [7 8 1] ;
24 end
```

Nevertheless, there may be a local optimal region in an unknown obstacle environment. This situation will cause the exploration method which using heuristic search to fall into the local optimal region. Therefore, the heuristic search strategies needs to be combined with greedy search to be able to jump out of the local optimal region. Aiming at the above heuristic search strategy, this paper proposes an improved random exploration, as shown in **Alg 2**.

```
Algorithm 2: Improved Random Exploration
Input: current state ;
Output: action ;
1  repeat
2      if $repeat\ times \leq 3$ then
3          $action = Heuristic\ Searching\ Strategies()$ ;
4      end
5      else
6          $action = randSelection\ [1\ 2\ 3\ 4\ 5\ 6\ 7\ 8]$ ;
7      end
8      $next\ state = moveRobot(current\ state, action)$;
9      $CollisionCheck\ ()$ ;
10 until $next\ state \neq current\ state$;
```

3.2.2. Boltzmann Exploration

In exploration, Boltzmann exploration exploit all the information present in the estimated Q-values produced by the learning phase. Instead of always taking the optimal action, or taking a random action, this approach involves choosing an action with weighted probabilities. To accomplish this purpose we use a softmax over the networks estimates the value for each action. In this case the action which the agent estimates to be optimal is most likely (but is not guaranteed) to be chosen. In Boltzmann exploration all available actions are weighed by their relative value. The robot selects an action a_t with a probability $P_t(a_t)$ is shown as follows:

$$P_t(a_t) = \frac{exp(Q_t(s_t, a_t)/\tau)}{\sum_{i=1}^{n} exp(Q_t(s_t, a_i)/\tau)} \tag{4}$$

where temperature parameter τ is annealed over time. This parameter controls the spread of the softmax distribution, such that all actions are considered equally at the start of training, and actions are sparsely distributed by the end of training.

3.2.3. Total Action Selection Strategy

Different from the traditional ε-greedy strategy, the new strategy is capable of taking the trade-off between exploration and exploitation appropriately. The total action selection strategy is listed as **Alg 3**.

3.3. Reward Function

In the QL algorithm, the goal of robot is to obtain maximum cumulative rewards in the interaction with the environment (from initial state to target state). Therefore, it is important to construct a reasonable reward function in reinforcement learning.

```
Algorithm 3: Action Selection Strategy
Input: control parameter $\varepsilon$, number of current
       episode $i$, random probability $p$, $p \in (0,1)$ ;
Output: action ;
1 if $p > \varepsilon$ then
2     $action = Boltzmann\ Exploration()$ ;
3 end
4 else
5     $action = Improved\ Random\ Exploration\ ()$ ;
6 end
7 $\varepsilon = \varepsilon/i$ ;
```

In order to control the torque generated by the robot during the movement, it is necessary to minimize the 45° turn in the trajectory, and avoid turning more than 45°. In other words, actions that can produce turns over 45° should be punished, i.e., given a relatively low immediate reward. Therefore, all actions that can produce turns should be given additional rewards (i.e.,a negative cost).

Moreover, in order to make the movement of the robot safer, it should avoid being too close to obstacles while ensuring a relatively short trajectory. Therefore, all actions that enable the robot to move near obstacles should also suffer a certain negative cost. The reward function for the action of turning and the action of approaching obstacles is set as **Alg 4**.

```
Algorithm 4: Reward Function for Turns
Input: current reward, previous action, current
       action, current state, obstacle state ;
Output: updated reward ;
1  $Turning = abs(action - preaction)$ ;
2  if $Turning == 1$ or $Turning == 7$ then
3      $reward = reward + cost\ (turn\ 45°)$ ;
4  end
5  else if $Turning == 2$ or $Turning == 6$ then
6      $reward = reward + cost\ (turn\ 90°)$ ;
7  end
8  else if $Turning == 3$ or $Turning == 5$ then
9      $reward = reward + cost\ (turn\ 135°)$ ;
10 end
11 else if $Turning == 4$ then
12     $reward = reward + cost\ (turn\ 180°)$ ;
13 end
14 $next\ state = moveRobot(current\ state, action)$;
15 $dist = \|current\ state - obstacle\ state\|$ ;
16 if $dist < 2$ then
17     $reward = reward + cost\ (near\ obstacle)$ ;
18 end
```

4. Simulation and Analysis

In this section, two experiments with different world maps are carried out to demonstrate the feasibility and effectiveness of the proposed method applied for mobile robot path planning problem. We consider an environment of 20×20 grids, where each grid represents a state. The black areas denote the predefined static obstacles in different shapes, sizes and layout.

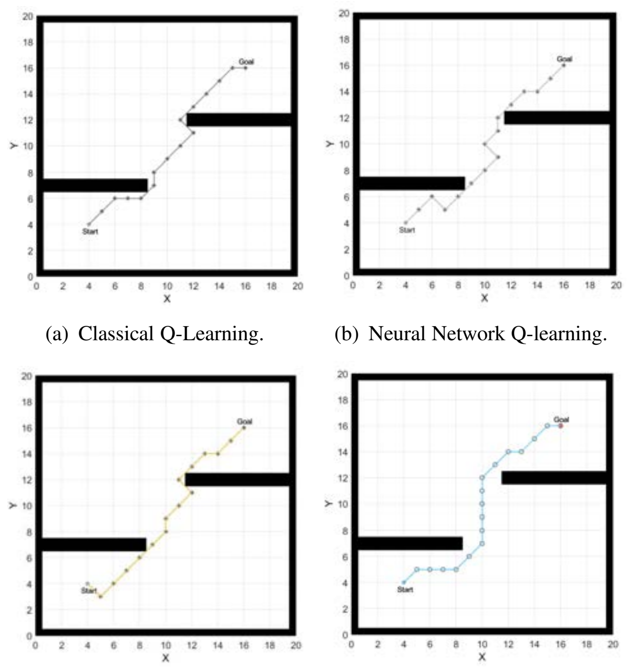
The performance metrics used here include the number of episode for each algorithm, the standard deviation of iterations for each episode, total travelled distance and the number of 90°/45° rotations (i.e., the smoothness of path) involved to completely execute the plan.

The simulation is repeated for 30 runs and subsequently, CQL, neural network Q-learning (NNQL) and SARSA method are compared with the proposed IQL. NNQL is a method of initializing the Q value table using neural network. SARSA is an on-policy Temporal-Difference method.

Experiment 1: The first experiment is carried out on a world map with two collateral walls to test whether robot can find a better path and escape the traps.

After exploration and exploitation, CQL can plan a shortest path. However, as shown in **Fig. 3(a)**, it is obvious that the path is very close to the obstacle. This will lead to a dangerous situation while robot is moving. Moreover, The path has generated some 90° turns, which may make it difficult for the robot to execute the trajectory properly due to the dynamic constraint and non-holonomic constraints. Also, it will cause great energy consumption. Similarly, the NNQL in **Fig. 3(b)** performs poorly in both the safety and smoothness of the path, as mentioned above. As shown in **Fig. 3(c)**, although SARSA can successfully plan the path, the path SARSA plan is also not ”safe” enough and will produce some 90° turns.

Under the influence of the repulsion potential function $U_{\text{rep}}(s)$ and the reward function designed to stay away from obstacles, the path planned by IQL is safer than the other method as shown in **Fig. 3(d)**. On the premise that the total travelled distance will not be too large, the path maintains a safe distance from the obstacle. Since negative rewards are set for all actions that produce large turns, the path will generate fewer 90° turns. This result can be obtained by observing **Table 1**. Compared to the other two algorithms, IQL generates the least 90° turns. This will help the robot to generate less torque during the movement, so as to reduce energy consumption. It can be noticed that the path generated by IQL has the most 45° turns, and its path length is the longest. Since the robot has to take a few more steps to ensure the safety of the robot. The shortest path is not always the optimal path. Ensuring safety and avoiding obstacles are the most important criteria.



(a) Classical Q-Learning. (b) Neural Network Q-learning.

(c) SARSA. (d) Improved Q-Learning.

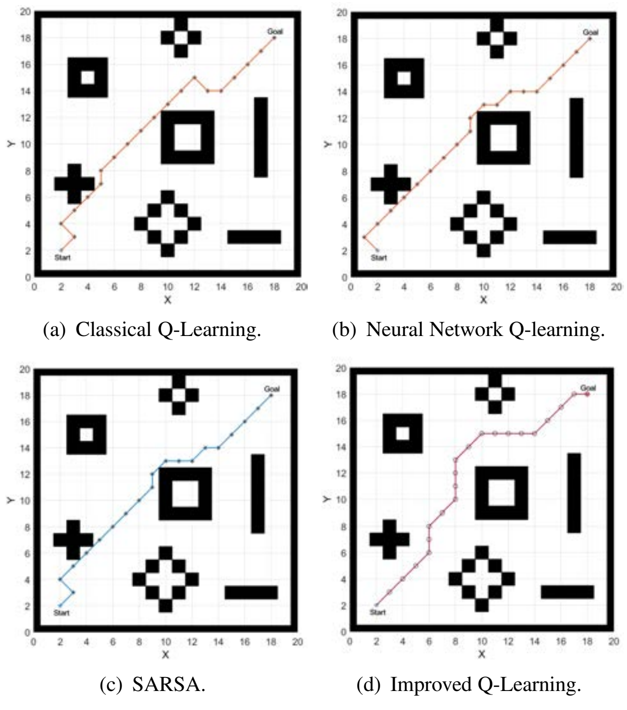
Fig. 3. World map 1 with two collateral walls.

Experiment 2: The second is concerned with training in the mix obstacles world map. Similar to the results in Experiment 1, the paths planned by CQL, NNQL and SARSA are not ”safe” enough and will produce some 90° turns. And the proposed IQL can learn a path that can not only maintain a safe distance from obstacles but also requires a small torque in a complex environment.

According to **Fig. 5(a)**, the number of IQL episodes is much less than CQL, and the number of iterations in each episode is relatively small. **Fig. 5(b)** compares the standard deviations of IQL and CQL to illustrate that the Q function initialization method based on inverse Euclidean distance and the action selection strategy can significantly improve the computational efficiency.

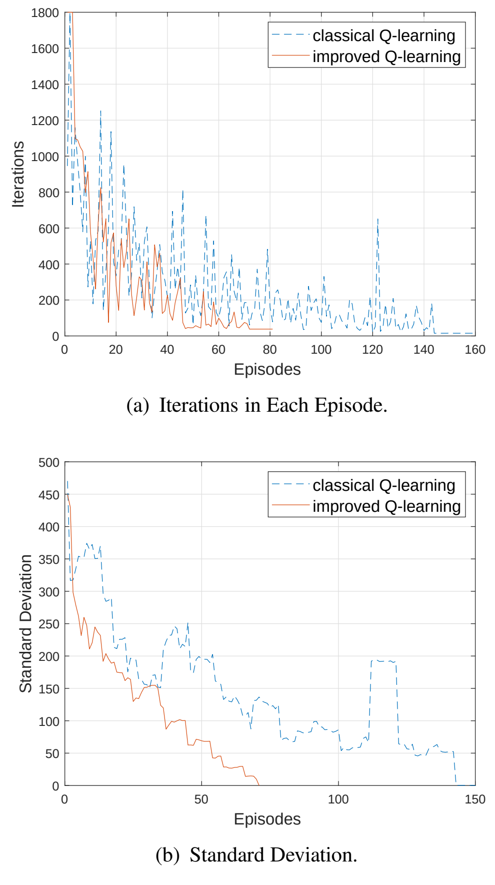
Table 1. Performance comparison between IQL, NNQL, SARSA and IQL based on the average result of 30 simulation experiments.

Evaluation Metrics	Experiment 1				Experiment 2			
	CQL	NNQL	SARSA	IQL	CQL	NNQL	SARSA	IQL
No. of Episodes	175.8	155.0	165.4	**141.1**	264.1	263.7	265.3	**124.4**
No. of 90° turns	1.6	1.5	2.0	**0.8**	1.4	1.4	1.1	**0.4**
No. of 45° turns	7.0	6.1	6.4	**8.0**	6.6	6.8	7.2	**8.8**
Length(unit)	16.2	16.0	16.0	**19.0**	20.0	20.0	20.0	**21.9**


(a) Classical Q-Learning. (b) Neural Network Q-learning.

(c) SARSA. (d) Improved Q-Learning.

Fig. 4. World map 2 with mix obstacles.



(a) Iterations in Each Episode.

(b) Standard Deviation.

Fig. 5. The convergence behaviour of CQL and IQL.

5. Conclusion

The experimental results prove that the IQL proposed in this paper has improved computational efficiency, smoothness and path safety compared with CQL. Although the path of CQL is very short, it is very close to obstacles. And there are many turns in the path generated by CQL, and the smoothing performance is poor. On the premise that the total travelled distance would not be too large, the path generated by IQL maintains a safe distance from the obstacle and produces fewer 90° turns. According to the statistics results of the Experimental 1, the 90° turn generated by IQL is only 50% of CQL. And the computational efficiency of IQL has increased by 19.7% compared with CQL based on the number of episodes.

Since RL can be used to solve sequential decision-making problems, the next research work can be carried out in a unknown dynamic obstacle environment for some multi-agents decision-making and planning problems

Acknowledgements

This work is supported by the State Recruitment Programof Global Experts under Grant WQ20141100198.

References:

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

[2] M. Chen, D. Zhu, “Real-time path planning for a robot to track a fast moving target based on improved Glasius bio-inspired neural networks,” IJIRA, vol. 3, no. 2, pp. 186-195, 2019.

[3] H. Ali, D. Gong, M. Wang, X. Dai, “Path planning of mobile robot with improved ant colony algorithm and MDP to produce smooth trajectory in grid-based environment,” Frontiers Neurorobotics, vol. 14, pp. 44, 2020.

[4] R.S. Sutton and A.G. Barto, “Reinforcement Learning: An Introduction,” MIT Press, Cambridge, 1998.

[5] P. Wang, X. Li, C. Song, S. Zhai, “Research on dynamic path planning of wheeled robot based on deep reinforcement learning on the slope ground,” J. Robotics, vol. 2020, pp. 7167243:1–7167243:10, 2020.

[6] S. Li, X. Xu, and L. Zuo, “Dynamic Path Planning of a Mobile Robot with Improved Q-Learning Algorithm,” IEEE International Conference on Information and Automation. pp. 409-414, 2015.

[7] H. Wu, L. Cai, and X. Gao, “Online Pheromone Stringency Guiding Heuristically Accelerated Q-Learning,” Application Research of Computers, in press.

[8] E. Wiewiora, “Potential-based shaping and Q-value initialization are equivalent,” Journal of Artificial Intelligence Research. 2003.

[9] Y. Song, Y. Li, C. Li, G. Zhang, “An efficient initialization approach of Q-learning for mobile robots,” International Journal of Control Automation & Systems. vol. 10, pp. 166-172, 2012.

[10] M. Simsek, A. Galindo-Serrano, A. Giupponi, “Improved decentralized Q-learning algorithm for interference reduction in LTE-femtocells,” Wireless Advanced, 2011.

[11] C. Yan, X. Xiang, “A path planning algorithm for UAV based on improved Q-learning,” International Conference on Robotics and Automation Sciences, 2018.

[12] C. Hao, Z. Fang, and P. Li, “A 3-D Route Planning Algorithm for Unmanned Aerial Vehicle Based on Q-Learning,” Journal of Shanghai Jiaotong University, vol. 46, pp. 1931-1935, 2012.

[13] Y. Zhou, S. Zhang, Q. Luo, C. Wen, “Using flower pollination algorithm and atomic potential function for shape matching,” Neural Computing and Applications, vol. 29, no. 6, pp. 22-40, 2018.

[14] F. Lewis, D. Vrabie, “Reinforcement Learning and Approximate Dynamic Programming for Feedback Control,” IEEE Circuits and Systems Magazine, v. 9, pp. 3, 2013.