Multi-Robot Path Planning Method Based on Prior Knowledge and Q-learning Algorithms

Bo Li and Hongbin Liang
Department of Computer Science and Engineering, Chongqing University of Technology, 69 Red Light Street, Banan District, Chongqing, China

*Corresponding author email: libo@cqut.edu.cn

Abstract. The path planning method based on prior knowledge and Q-learning algorithm proposed in this paper solves the problems of low operating efficiency and slow learning speed in existing multi-robot coordination and collision avoidance. This method can be divided into two stages. Firstly, the improved Q-learning algorithm is used to plan the static collision free path of a single robot in the built model; After the Q-table is initialized with the prior knowledge obtained in the previous step, the Q-learning algorithm is used to achieve conflict-free motion among multiple robots. Finally, the simulation experiment shows that the proposed path planning method based on prior knowledge and Q-learning algorithm performs well in handling collision avoidance path planning of multiple robots.

1. Introduction
In recent years, with the rapid development of robotics, robots have entered more people's daily lives. In order to better meet the needs of people's production and life, the research of multi-robot systems has become a hot research topic, but multi-robot systems also face a series of problems such as complex environmental modeling and difficulty in coordination between multiple robots. How to design a safe and efficient robot path planning algorithm [1] coordinating multi-target collision-free motion has become one of the key problems faced by multi-robot systems. Path Planning is one of the main research directions of robot research. The path planning problem of the robot [2] is to search for an optimal path which from initial state to the target state under constrains of any performance standard (such as the least search time, the shortest path or the least steering operation) in working space that it can reach, while avoiding obstacles. Traditional robot path planning algorithms, such as A* algorithm, Genetic algorithm [3], Artificial Potential Field Method [4], etc., have their own limitations in the face of multi-robot path planning problems. The A* algorithm has too many turning times, and it is difficult for the robot to travel according to its prescribed route; the Genetic algorithm is still imperfect, and its planned route is greatly affected by human factors; the Artificial Potential Field Method is prone to jitter when the obstacles are dense and the steering space is narrow. At present, most of the researches on multi-robot path planning are developed from the existing single-robot path planning search algorithm, which can be divided into: traditional algorithm (such as A* algorithm, Artificial Potential Field Method.), intelligent optimization algorithm(such as Ant Colony Algorithm, Neural Network Algorithm) and other algorithm(Fuzzy control and taboo search ,etc.). Yin Cheng [5] proposes a CDRLOA(Concise Deep Reinforcement Learning Obstacle Avoidance Algorithm) to solve the obstacle avoidance for underactuated unmanned marine vessels; Chang Liu [6] use model predictive control to complete path planning algorithm; Shuhuan Wen [7] combined Slam to improve the autonomous navigation capability of the agent in a complex
environment. These algorithms do not perform well in the path planning of multi-robot systems and cannot meet our needs.

Reinforcement learning (RL) is one of the important branches of machine learning. Reinforcement learning algorithm usually carries out environment modeling based on Markov Decision Process\[8\], which has a complete theoretical basis. RL emphasizes immediate interaction with the environment, continually experimenting with various actions, and continuously optimizing action selection strategies through delayed rewards. The learning and thinking process of RL is similar to the human learning process, which can well explain the strategy selection of the robot. The robot subject also shows high adaptability and learning ability, and can independently choose actions according to the environmental information and complete the path planning task. But at the same time, the algorithm based on RL has also obvious disadvantages, that is, slow learning speed and long convergence time.

This paper uses the representative Q-learning algorithm \[9\] in RL, and proposes an improved method for the original Q-learning algorithm: the balance problem of exploration and utilization, the problem of maximizing deviation and the slow update speed. In the multi-robot system, in order to realize coordinated collision-free movement of multiple robots, a combined action is adopted (all robot’s action are combined into one motion vector), and also in order to ensure the overall migration of environmental information, a combined state is also adopted. This makes the Q-table’s value of each robot in the multi-robot system both a combined state and a combined action to a single mapping of Q values, ensuring coordinated motion of the multi-robot system. At the same time, the single robot path planning route obtained by the previously improved Q-learning algorithm is used as a priori knowledge to predict the next action of other robots in advance to reduce the learning space and speed up the learning. Finally, the simulation experiment on MATLAB proves the effectiveness of the improved algorithm, and the convergence time of the algorithm is significantly reduced. The robot can design a collision-free path faster than the original Q learning algorithm.

2. Background

2.1. Reinforcement Learning

Reinforcement Learning (also known as Evaluation Learning) realizes learning objectives through the interaction between agent and environment, agent try to get the cumulative value as much as possible while interacting with the environment.

The Markov Decision Process (MDP) can be defined as a quaternion relationship group:

\[
M = (S, A, \rho, f)
\]

In RL, the strategy \( \pi : S \rightarrow A \) is a mapping from the state space to the action space, indicating that agent performs the action \( a_t \) in the state \( s_t \) and moves to the next state \( s_{t+1} \) with the probability \( f(s_t, a_t, s_{t+1}) \), while getting the immediate reward \( r_t \) from the environment. The ultimate goal of RL is to find the optimal strategy \( \pi^* \) that can achieve the maximum expected reward.

2.2. Q-learning Algorithm

Q-learning is a model-free TD (0) RL\[10\] algorithm proposed by Watkins in 1989. The original algorithm’s flow is shown in figure 1:

\[
\begin{align*}
\text{Initialize } Q(s,a) \text{ arbitrarily; } \\
\text{Repeat (for each episode): } \\
\quad \text{Initialize } S \\
\quad \text{Repeat (for each step of episode): } \\
\quad \\
\quad \quad \text{Choose } A \text{ from } S \text{ using policy derived from } Q \text{ (e.g., } \epsilon\text{-greedy)} \\
\quad \\
\quad \quad \text{Take action } A, \text{ observe } R, S' \\
\quad \quad Q(S, A) \leftarrow Q(S, A) + \alpha[R + \gamma \max_a Q(S', a) - Q(S, A)] \\
\quad \quad S \leftarrow S' \\
\quad \quad \text{until } S \text{ is terminal }
\end{align*}
\]

Figure 1. Original Q-learning Algorithm
The action value function update formula of the Q-learning algorithm is as follows:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left( r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a) \right)$$

(1)

It can be seen that the original Q-learning algorithm simply selects the action according to the greedy strategy in the action selection, which causes the agent to not quickly acquire the surrounding environment in the early stage of training. In the later stage of training, the agent cannot plan the action smoothly; At the same time, always choose the action corresponding to the large Q value, which is easy to cause deviation; and every time an episode is experienced, the Q table is updated once, making the model fluctuate greatly, it is not easy to converge, the training is too slow and it is too sensitive to some abnormal situations. Robustness is not strong. Aiming at these three problems of original Q-learning algorithm, this paper proposes corresponding improved methods and proposes an improved algorithm based on Q-learning.

3. Improved Q-learning Algorithm

3.1. Improved Methods for Balancing Problems of Exploration and Utilization

The original Q-learning algorithm uses a simple $\varepsilon$-greedy strategy as an action choice. The $\varepsilon$-greedy strategy formula is expressed as follows:

$$\pi(a|s) = \begin{cases} \frac{\varepsilon}{m} + (1 - \varepsilon) & \text{if } a^* = \text{argmax}_{a \in A} Q(s, a) \\ \frac{m}{\varepsilon} & \text{else} \end{cases}$$

(2)

The $\varepsilon$ value is usually set very small at the beginning, the probability of 1-$\varepsilon$ is used to greedily select the current maximum action of Q value. The $\varepsilon$-greedy strategy has proven to be reliable and effective in the long-term use, but there are also some problems. At the beginning of the iteration, the agent should be biased towards exploring environmental information. When the environmental information is relatively complete, the agent should tend to be conservative, mainly based on greed, so that the algorithm can converge stably. Therefore, an improved method is proposed, which adds the discount factor $\varepsilon_{\text{decay}}$ of $\varepsilon$. After training a certain number of rounds $N$, $\varepsilon$ is changed to $\varepsilon = \varepsilon \times \varepsilon_{\text{decay}}$, and the balance agent chooses the exploration and utilization at different stages of training. The improved $\varepsilon$-greedy strategy is as follows:

$$\pi(a|s) = \begin{cases} \frac{\varepsilon}{m} + (1 - \varepsilon) & \text{if } a^* = \text{argmax}_{a \in A} Q(s, a) \\ \frac{\varepsilon}{m} & \text{else} \end{cases}$$

(3)

3.2. Improved Method for Maximizing Deviation Problems

In the traditional Q-learning algorithm, the update strategy is $\pi(s) \leftarrow \text{arg max}_a Q(s, a)$, that is, the action a corresponding to the maximum Q value is continuously selected, possibly because of some noise. The item thus affects the final result. For example, in the initial state $s_t^i$, a certain robot selects an action $a_t^i \in A$ according to a certain strategy, and moves to the state $s_{t+1}^i$, and an immediate return $r$ is obtained. That is:

$$Q(s_t^i, a_t^i) = r + \gamma \max_a Q(s_{t+1}^i, a_{t+1}^i)$$

(4)

For the above formula, in order to achieve the maximum long-term cumulative reward, the action $a_t^i$ which produces the maximum value of the current state Q value is repeatedly applied, but this result is only established when the Q value is very accurate. If a certain Q value in the currently generated Q value table is destroyed by a certain noise term (for example, the selection of $\alpha$ and $\gamma$ values), the value of $Q^i$ will be caused to fluctuate up and down, but each time in the Q learning algorithm Both use the
maximum Q value, then the value of \( Q^\pi \) is very likely to affect the correct Q value of an action. This effect will cause the performance of the Q learning algorithm to decrease, so that the optimal strategy cannot be learned. And when the robot system continuously adopts the action with the largest Q value, it is likely to be associated with the state of high Q in the early stage of training, so that the learning time converges too fast, thus missing some possible optimal strategies. For this problem, we can use the probability selection method instead of using the maximum value of Q to select the action, that is, when the robot selects the next state, use the probability to select:

\[
P(s|a_k) = \frac{Q(s|a_k)}{\Sigma_j Q(s|a_j)}
\]

(5)

The probability that the action is selected corresponds to the magnitude of the Q value. Therefore, the disadvantages of the above two points are effectively avoided.

### 3.3. Improved Method for Slow Update Issues

The learning process of the Q-learning algorithm is an iterative process that requires constant trial and error to gradually improve the mapping strategy from state to action. This requires the learning system to perform multiple rewards for each possible state action value through environmental feedback. Correction can get the optimal strategy \cite{11}. The Q-learning algorithm tends to be slow to update because each revision of the action selection strategy relies on feedback from the environment which causes a lot of unnecessary training expenses and low learning efficiency.

This paper proposes an improved method to increase the counting threshold \( h \) to solve the problem of slow updating speed in Q-learning. Count the number of visits to \( <s, a> \) state-action pairs during training, when the cumulative number of accesses reaches \( h \) times, the Q value of the state-action pair \( <s, a> \) is updated. This design not only enables the algorithm to have the ability of multi-step prediction, but also considers the influence of multiple future status-actions on the Q value, making the learning strategy more reasonable. At the same time, it does not need to increase the computational cost as the previous algorithms. At the same time, there is no need to calculates the qualification trace matrix like the original Q-learning algorithm which cause additional computational overhead.

### 4. Multi-robot Path Planning Algorithm Based on Prior Knowledge and DQN

In the previous part of this paper, an improved Q-learning single robot path planning method is proposed. However, in daily production and life, most of the situations encountered require multiple robot coordination to complete the task. The Q-learning single robot path planning method obviously cannot meet the actual requirements. Therefore, based on this consideration, this paper proposes a multi-robot path planning algorithm based on prior knowledge and DQN. Multi-robot path planning problems often need to consider the following two issues \cite{12}: (1). collision avoidance path planning of a single robot in a multi-robot system; (2). Collision-free coordination between multiple robots.

#### 4.1. DQN

The Q-learning algorithm uses a table to store Q (s, a), but in a multi-robot system, because the number of robots increases, the dimension explosion causes the Q table cannot be created. The usual method at this time is to use a value function approximation. In 2013, DQN (Deep Q Network) algorithm was first proposed by DeepMind Technologies \cite{13}.

#### 4.2. Application of Prior Knowledge and Prior Rules

The application process of prior knowledge and prior rules is shown in Figure 2:
4.2.1. Prior Knowledge. The DQN algorithm is used to solve the multi-robot path planning problem. The goal is consistent with the Q-learning algorithm. It is also to learn how to interact with the environment continuously to select better actions in dynamic situations. However, at the beginning, the multi-robot system has little almost zero information about the environment, therefore, it is necessary to continuously learn through repeated trial and error, so the algorithm is not very efficient. If you can use the experience gained from previous execution of related tasks to make the multi-robot system have a certain understanding of the environment before learning and training, you can reduce the complexity of learning and the tedium of calculation. As mentioned above, there are two main problems that need to be solved in the problem of multi-robot path planning: 1. collision avoidance path planning of a single robot in a multi-robot system; 2. Collision-free coordination between multiple robots. This article uses prior knowledge to initialize the Q-table reasonably which can help agent predict subsequent action choices of individual robot, this method can effectively shorten the learning time.

4.2.2. Priori Rules. In the DQN algorithm, before the multi-robot system learns, a series of rules that affect the action selection can be artificially made. For example, in the process of robot collision avoidance path planning, when an obstacle appears on the left side of the current state of the robot, the next action of the robot cannot select to collide with the obstacle to the left, that is, when there is an obstacle on the left side, other actions should be selected. For example, turning right, moving forward or backward is a priori rule. The DQN algorithm also uses the $\epsilon$-greedy strategy to explore the environment. At the beginning of the learning, because the robot system knows nothing about the environment, it will set a larger value of the exploration factor $\epsilon$, making it more inclined to explore the environment. That is to say, in order to explore the environment, the learning system will have a large number of random selection actions, which means that a large amount of useless exploration will be generated, which increases the length of learning training. Adding a priori rule to control the robot's motion selection can solve this problem and improve the efficiency of the algorithm. Adding a priori rule has two advantages: First, after the special state $p_l$ occurs, the robot will select the next action according to the a priori rule $p_l \rightarrow \hat{a}_l$, which can reduce useless exploration; Second, after the a priori rule is added, if a special state does not occur, the environment will still be fully explored through the exploration strategy, and the learning and training process of the original DQN algorithm will not be affected.

4.3. Multi-robot Path Planning Algorithm Based on Prior Knowledge and DQN
In the improved DQN algorithm, this paper first uses the improved Q-learning single robot path planning method to obtain the static obstacle avoidance optimal path planning of each robot in the multi-robot system as a priori knowledge. The single robot tends to choose the optimal line to avoid collision with the static obstacle in the next action. When the resource conflict occurs between the

Figure 2. Application process of prior knowledge and prior rules
robots, the corresponding Q value is continuously updated according to the DQN algorithm. This article will use a priori knowledge to reasonably initialize the Q table to predict the subsequent action choices of each robot's tendency, and determine its own action at this time, thus shortening the learning process. Finally, simulation experiments were carried out on MATLAB, which further proved the superiority of the improved algorithm compared with original DQN algorithm.

5. Experiment

5.1. Introduction to Experimental Platform

The specific experimental environment is shown in table 1:

| experimental platform | Environment configuration |
|-----------------------|---------------------------|
| OS                    | Windows 10                |
| CPU                   | Inter Core i5-8400 2.80GHz|
| GPU                   | NVIDIA GeForce GTX 1060 6G |
| memory                | 16GB                      |
| Programming language  | Python/Matlab 2016a       |
| Deep learning framework| TensorFlow               |

5.2. Environmental Modeling

This article will use the grid method to describe the environment, the workspace of the robot system is divided into small grids, each of which represents a state of the robot system. On the map, a white grid indicates a secure area and a black grid indicates the presence of obstacles. The target state and obstacle in the environment are stationary, and the obstacle and boundary position in the environment are unknown to the robot. In subsequent experiments, the robot's workspace was either a 10x10 or a 20x20 grid map. In the simulation process, the robot's movement path and initial state are represented by red grid, and the target state is represented by yellow grid. As shown in figure 3, 4:

![Figure 3. Experimental Scene 10x10](image1)
![Figure 4. Experimental Scene 20x20](image2)

5.3. Experimental Results and Analysis

Experiment 1: Improved path planning algorithm for single robot

In the experiment 1, the grid map of 10x10 was adopted, the position of obstacles was randomly given, the initial state of the robot was (0,0), and the target state was (7,7). The Q-learning algorithm and the improved Q-learning algorithm are respectively used for the simulation experiment of path planning of single robot. Figure 5, 6 shows the planning path obtained by algorithm.
The red grid represents the walking path planned by the algorithm. The route planned by the original Q-learning algorithm has more turning points, and the route planned by the improved algorithm is smoother, indicating that the route planned by the improved Q-learning algorithm is more excellent. Figure 7,8 is a line graph of the time taken by the traditional Q-learning algorithm and the time it takes for the improved algorithm to converge during training.

As can be seen from the figure, the original Q-learning algorithm takes about 700 seconds to plan a qualifying line. The improved Q-learning algorithm only takes about 300 seconds to plan a qualified line. The traditional Q-learning algorithm can hardly find a path to the target state at the beginning of training, but the improved algorithm can find a path to the target state earlier.

In addition, it can be found that with the increase of training time before and after the improvement, the probability of the robot system successfully finding the path is gradually increasing, but the speed and times of improvement are obviously faster and more. The model before the improvement converges until about 900 seconds, and the improved model converges around the 500th second. The above two points can explain that the improved Q-learning algorithm greatly improves the efficiency of the algorithm compared with the original algorithm.

Experiment 2: Multi-robot path planning algorithm based on prior knowledge and DQN
The experimental comparison chart is shown in Figure 9,10.
As far as the convergence speed is concerned, the original DQN algorithm has not converge until 2500 steps, and the improved DQN algorithm has tended to converge completely around the 600th step. Therefore, we can find that the improved DQN algorithm has an obvious convergence speed of the algorithm. This is due to the rational application of prior knowledge, which increases the understanding of the environment of the robot system and speeds up the optimization of Q-Network. At the same time, it adds a priori rule, which reduces the number of useless random explorations and shortens during learning and training. The training duration makes the algorithm more intelligent and efficient.

6. Conclusions
In this paper, for single robot system and multi-robot system, we propose a single robot path planning algorithm based on Q-learning algorithm and a multi-robot system path planning algorithm based on prior knowledge and DQN. In the path planning problem of single robot, solutions are proposed respectively for the three problems existing in the traditional Q-learning algorithm: (1). how to keep a good balance between exploration and utilization. we use the greedy strategy with decreasing $\varepsilon$ to balance the need for exploration and utilization in different phases of training; (2). Maximum Deviation. We adopted the method of selecting action which the probability of being selected was determined according to the Q value, that can effectively avoid excessive deviation; (3). Slow update of algorithm. By updating in a certain step size, a large amount of unnecessary computational overhead of the algorithm during the training process is avoided. In the path planning algorithm of multi-robot system, we first use the improved Q learning algorithm to obtain the static collision-free motion sequence of a single robot, and then initialize the Q value table by using the prior knowledge obtained in the single robot path planning process. After the system has a certain understanding of the environment, the collision-free path planning of the multi-robot system is finally realized. In the path planning of a single robot, the training cost of the improved algorithm is significantly smaller than that of the original Q-learning algorithm, and the planning result is smoother; In the path planning of multi-robot systems, we have achieved collision-free path planning for multiple robots, and the training cost is smaller than the traditional Q-learning algorithm, which proves the superiority of the improved algorithm.

However, the improved algorithm in this paper requires a large amount of computation, it’s too difficult apply to large multi-robot system. In the following research, we can improve from the following aspects: 1. The neural network structure in DQN is too simple, and the algorithm performance can be improved by reasonably constructing the network structure; 2. In order to obtain an optimal solution for a multi-robot system, an optimization index can be added. Such as time and energy consumption; 3. The coordinated movement of multi-robot systems is not only avoiding collisions, but more coordination methods can be considered.
Acknowledgments
This work is supported by the Natural Science Foundation Project of Chongqing Science and Technology Commission. (Grant No. cstc2015jcyjBX0113), Chongqing higher education teaching reform research project(173155), Chongqing graduate education and teaching reform research project(yjg183114).

References
[1] ZHU Da-qi, YAN Ming-zhong. Survey on technology of mobile robot path planning[j]. Control and Decision,2010(07):4-10.
[2] Li-Bo, Yi Jie. The Time-Based Multi-Robot Coordinated Collision Avoidance Algorithm[J]. Journal of Chongqing University of Technology(Natural Science), 2019,33(3):91-97.
[3] MA Yong-jie, YUN Wen-xia. Research progress of genetic algorithm[j]. Application Research of Computers, 2012, 29(4):1201-1206
[4] Duan Jian-min, Chen Qiang-long. Research on path planning algorithm based on improved artificial potential field-genetic algorithm[J]. Foreign electronic measurement technology, 2019(3).
[5] Cheng, Yin & Zhang, Wei-dong. (2017). Concise deep reinforcement learning obstacle avoidance for underactuated unmanned marine vessels. Neurocomputing, 272. 10.1016/j.neucom.2017.06.066.
[6] C. Liu, S. Lee, S. Varnhagen and H. E. Tseng, "Path planning for autonomous vehicles using model predictive control," 2017 IEEE Intelligent Vehicles Symposium (IV), Los Angeles, CA, 2017, pp. 174-179, doi: 10.1109/IVS.2017.7995716.
[7] Wen, S., Zhao, Y., Yuan, X. et al. Path planning for active SLAM based on deep reinforcement learning under unknown environments. Intel Serv Robotics 13, 263–272
[8] Wikipedia F, Programming D, Processes M. Markov Decision Process[J]. European Journal of Operational Research, 1989, 39(1):1-16.
[9] Ben J. A. Kröse. Learning from delayed rewards[J]. Robotics and Autonomous Systems, 1995, 15(4):233-235.
[10] Sutton, R. (1988). Learning to predict by the methods of temporal differences. Machine Learning, 3 (1): 9–44.
[11] CAI Qing-Sheng, ZHUANG Bo, AN AGENT TEAM BASED REINFORCEMENT LEARNING MODEL AND ITS APPLICATION [J]. JOURNAL OF COMPUTER RESEARCH& DEVELOPMEN T, 2000, 37(9):1087-1093.
[12] ZHANG Ya-lin, LEI Xiao-yu. Path planning research for multi-robot[J]. APPLICATION RESEARCH OF COMPUTERS (09):12-15.
[13] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonooglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing atari with deep reinforcement learning.