Mobile robot navigation method based on improved Q-learning algorithm

1st Zhiyong Tan
School of Automation and Electrical Engineering
University of Science and Technology Beijing
Beijing, 100083, China
18701218065@163.com

2nd Tao Wang
School of Automation and Electrical Engineering
University of Science and Technology Beijing
Beijing, 100083, China
m17611259705@163.com

3rd Yao Yu
Key Laboratory of Knowledge Automation for Industrial Processes of Ministry of Education School of Automation and Electrical Engineering, University of Science and Technology Beijing, Beijing,100083, China
yuyao@ustb.edu.cn

2021 36th Youth Academic Annual Conference of Chinese Association of Automation (YAC) | 978-1-6654-3712-7/21/$31.00 ©2021 IEEE | DOI: 10.1109/YAC53711.2021.9486613

***Abstract*—In recent years, autonomous navigation of mobile robots has become one of the hot topics in research. An improved Q-learning method (IQL) combining action selection strategy and Q function update method is proposed. A new exploration strategy of staged ε greedy exploration is used in IQL. In addition, the idea of backtracking is used to update the Q value to speed up the convergence speed. Experiments conducted in a grid environment show that compared with the classic algorithm, the algorithm converges faster and the program execution time is shorter.**

Keywords—mobile robot; path planning; reinforcement learning; Q-learning;

I. INTRODUCTION

Currently, path planning is becoming more and more important in mobile robot navigation. The research interest in the perception and obstacle avoidance technology of intelligent robots has continued to grow [1]. The specific definition of path planning is that robots follow certain rules or requirements to plan an optimal or nearly optimal conflict-free path from the starting point to the target point in an environment with obstacles. [2]. The path planning problem of the robot [3] is to find the best path from the initial state to the target state under the constraints of some performance standards (such as using the shortest search time, obtaining the shortest path, or using the least turning operations) in the work. According to the degree of understanding of the environment, we can divide path planning into global path planning and local path planning. If the environment is known, path planning at this time is called global path planning. When the environment is complex, the calculation cost of the global path planning method is very high. If the environment information of the robot is not known or part of the terrain is unknown, the path planning at this time is called local path planning. The classic local path planning algorithms include simulated annealing method [4], particle swarm optimization (PSO) [5], ant colony algorithm (ASO) [6], reinforcement learning (RL), etc.

Q-learning is one of the classic reinforcement learning algorithms. This article applies Q-learning to the path planning problem of mobile robots. In [7], a heuristic Q-learning algorithm was proposed and applied to the path planning problem, but it is only suitable for simple static environments. In [8], a Q-learning method combined with deep learning is introduced to solve the local path planning problem. This method divides the learning process of the algorithm into two stages: the use of reinforcement learning algorithm and the training stage of neural network. However, in complex maps, the time complexity of this algorithm is very high.

This article proposes an improved Q-learning algorithm (IQL). Compared with the original algorithm, this algorithm has better performance in path planning problems. In IQL, in addition to updating the Q value after each step is completed, for each Q value in the state chain that successfully reaches the target point from the starting point, a reverse update is performed, so that the subsequent evaluation value can be timely feedback come back. Using ε-greedy exploration and some heuristic search strategies, a new staged exploration strategy is combined.

Using the idea of backtracking, a path planning attempt (a cycle of the robot from the starting point to the target state is called an attempt) is regarded as a series of data transmission. Each state experienced in the cycle is encapsulated together. A chain of states is formed, and each state forms a ring of the chain. Each step in the training cycle represents moving from the current state to the next state. In addition to updating the corresponding Q value for each step, it is judged whether the target point is reached. If the target point is reached, the Q value on the entire state chain is updated in reverse, from present to past.

The simulation experiment is designed and completed on win10 64-bit operating system, and the platform used in the experiment is python3.6. In this paper, the IQL is compared with the path planning methods of classical Q learning and SARSA

method, and the experiment is carried out in a grid environment and its performance is analyzed.

The rest of this paper is organized as follows. The Classic Q-learning is described in Section II. The IQL based path planning algorithm is listed in Section III. Section IV mainly includes experimental results and analysis.

II. CLASSIC Q-LEARNING

A. Q-Learning

The Q learning algorithm was proposed as early as the 1980s. The basic idea is that when the agent faces a target task, the agent will try different ways of movement, and constantly change the way of movement according to the response of the environment, until the target task is completed.

The Q-learning algorithm builds a Q table based on the state space and action space generated by the agent to store the Q values of all state-action pairs. At the same time, a reward function is established according to the feedback information of the environment to achieve the purpose of training the agent to complete the target task. If the state-action pair is rewarded by the environment, its Q value will continue to increase; if the state-action pair is punished by the environment, its Q value will continue to decrease. Finally, the agent chooses actions according to the strategy according to the Q table, and completes the task in the best way.

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

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

where $Q(s_t,a_t)$ is the state-action pairs and their corresponding reward value, s_t represents the current state of the agent, a_t is the current action performed by the agent in the current environment, a_{t+1} is the follow-up action, s_{t+1} is the state reached by the agent after executing a_t, r_{t+1} is the immediate reward, $\alpha \in (0,1)$ is the learning rate, $\gamma \in (0,1)$ is the discount rate. When the value of α is relatively small, the Q value will gradually converge to the optimal Q value during the training process. When γ = 0, the agent only obtains the current reward, and the agent will gradually consider future rewards when γ is not far from 1. After many training processes, each state will get its own reward, that is to say, the value of each state converges to a certain value, the formula is as follows:

$$\max_{a_t \in A} Q(s_t,a_t) \to E\left\{\sum_{n=1}^{\infty}\gamma^{n-1}r_{t+n}\right\} \tag{2}$$

B. Q-learning in Path Planning Problem

Rasterized environmental space is one of the common prerequisites for solving path planning problems. Usually in the test environment, we assume that there are obstacles in certain locations. It is usually assumed that the robot is moving at a fixed speed, and the robot in the environment is regarded as a point, which can facilitate the subsequent training process.

The state is defined as the grid position in the rasterized environment, and the action is the behavior taken by the robot from one grid position to another grid position. The robot's action space is reasonably allocated to 8. The representation of the action is shown in Fig.1.

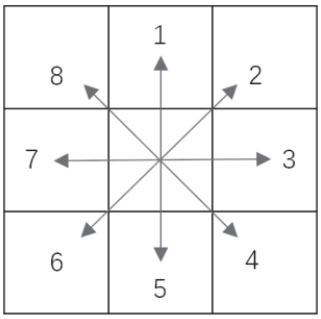

Fig. 1. 8 Discrete Actions.

Rewards and punishments are important factors affecting the efficiency of RL methods. In the reinforcement learning algorithm, the agent chooses an action in the current state and executes it, so as to reach the next state. In this process, it can obtain rewards from the environment. The agent is constantly training, hoping to maximize the cumulative expected return.

C. Problem Statement

In the early stages of training, the agent is very unfamiliar with the state of the environment, and randomly and blindly select actions from the action set. Every time the mobile robot performs an operation, it needs to use the action selection strategy of π. The common action selection strategies mainly include greedy strategy, ε_Greedy strategy and dynamic decline ε_Greedy strategy.

In the greedy strategy, each step chooses the greatest reward of action, and it's easy to have a local optimum. In response to this shortcoming, we often use ε_Greedy strategy, ε_Greedy strategy randomly choose action adopted ε as the probability. But the disadvantage of this method is that the parameters require multiple tests to determine the appropriate ones, which is time-consuming.

In the dynamic decline ε_Greedy strategy, the value of e gradually decreases with the episode, and finally reaches a certain value. This makes it possible to gradually shift from a tendency to explore to a tendency to exploit when making action choices. However, there may still be a useless exploration process in the process of use.

The classic Q-learning's Q value data transmission has a certain lag. In the traditional Q-learning training cycle, suppose there are three states s1, s2, and s3 that are arbitrarily connected. To obtain information feedback from the s3 state in the s1 state, we need to repeat the training cycle twice, that is

- The first cycle:

  $Q(s_2) \to Q(s_1) \to Q'(s_1)$

$$Q(s_3) \rightarrow Q(s_2) \rightarrow Q'(s_2)$$

- The second cycle:

$$Q'(s_2) \rightarrow Q'(s_1) \rightarrow Q''(s_1)$$

In the traditional algorithm, each update of the Q of the s1 state needs to be trained from the beginning, and the Q value corresponding to the s1 state is updated twice, and it needs to be trained twice, until then $Q(s_1)$ has the information of $Q(s_3)$. Through derivation, it is known that the data transmission of the classic Q-learning has a certain lag, and the first Q value needs to be iterated n times to obtain the feedback of the Q value of n states apart, that is, it needs to repeat the training n times [9].

III. IMPROVED Q-LEARNING

This section provides two strategies to improve the performance of the Q-learning algorithm in path planning problems. The specific instructions are as follows.

A. Action Selection Strategy

The performance of reinforcement learning algorithms is largely affected by the two stages of "exploration" and " exploitation". Exploration usually means that the agent randomly chooses any possible actions in the current state to explore the broader environment. The purpose of exploitation is to use the knowledge that the agent has learned, and to guide the agent to obtain good performance by choosing greedy actions.

According to experience, at the beginning of training, it is more inclined to explore, after learning a certain amount of knowledge, it slowly turned to use. Exploration means the choice of random actions, more states can be accessed, and the agent can obtain more environmental information in order to access all state-action pairs in the environment. Exploitation is to use the knowledge learned by the agent in the exploration stage to select appropriate actions so that the agent can get the greatest reward.

Based on the above reasons, we designed a new staged exploration strategy to solve the problem of balance between exploration and exploitation which combines the exploration of ε-greedy and some heuristic search strategies. It can avoid local optima and accelerate convergence.

1) Heuristic Searching Strategies: In order to speed up the learning process, some heuristic search strategies are left here. Random actions are required to satisfy the heuristic search strategy. Suppose (x_c, y_c) and (x_g, y_g) represent the current state and target state, respectively [10]. The details are as follows:

a) If $x_c < x_g$ *and* $y_c < y_g$, *a=rand(1,2,3); else if* $x_c > x_g$ *and* $y_c > y_g$, *a=rand(5,6,7).*

b) If $x_c < x_g$ *and* $y_c > y_g$, *a=rand(3,4,5); else if* $x_c > x_g$ *and* $y_c < y_g$, *a=rand(7,8,1).*

c) If $x_c = x_g$ *or* $y_c = y_g$, *a=rand(1,5) or a=rand(3, 7);*

d) Else a = a-1 / a / a+1;

We use strategies a)-c) to increase the probability of the robot reaching the target state, while strategy d) is used to reduce the direction angle of the path to avoid excessive turning angles of the agent.

2) Stop Strategies: On the issue of balancing exploration and exploitation, a judgment condition is added to judge whether the algorithm succeeds in finding a better path, so as to save redundant exploration and speed up the convergence of the algorithm. The specific judgment condition we adopted is that in 10 consecutive episodes, the path planned by the algorithm is the same and all successfully reached the target point.

Aiming at the problem of balance between exploration and exploitation, a new greedy strategy is designed in stages. In the first stage, when randomly selecting, priority is given to the set of actions that are close to the target point. In the second stage, the value of epsilon decreases with the cycle. In the third stage, when the number of consecutive repetitions of the path to the target point reaches the set value, the value of epsilon is set to a very low value unchanged, which can reduce unnecessary exploration process. The total action selection strategy is listed as Alg1.

Algorithm 1 Action Selection Strategy

Input: Control parameter ε, number of current episode i, random probability p, $p \in (0,1)$;

Output: action;

```
if path repeats ten times in a row & reaches the goal then
    action = argmax Q(s,a);
else
    if p > ε then
        action = argmax Q(s,a);
    else
        if state repeat time < 3 then
            action = Heuristic Searching Strategies ();
        else
            action = randomSelection [1,2,3,4,5,6,7,8];
        end if
    end if
    ε = ε / i
end if
```

B. Q-function update method

In order to improve the lag of data transmission in Q-learning and speed up the convergence speed, we designed a new Q value update method. The details are listed in Algorithm 2.

In the learning process of the Q-learning algorithm, a memory matrix is used to record the parameters used each time. When the agent successfully reaches the goal, iteratively transforms the Q values in the matrix according to the formula in Algorithm 2. As a result, each Q value under the path is affected by the reward from the goal, the update speed of the Q value becomes faster, the number of steps required for learning convergence is correspondingly reduced, and the convergence

speed is accelerated. Then clear the memory matrix and record again.

Algorithm 2 Q-function update method

Input: Information of environment G, initial state s_i and goal state s_g, α, γ, ε, ξ and max iterations, $M=[]$ for all $s \in S$;

Output: optimal Q-table;

While num < max iterations **do**
  Initialize s_t, M;
  repeat
    ActionSelectionStrategy ();
    Observe the subsequent state s_{t+1};
    Receive an immediate r_t reward;
    $Q(s_t,a_t)=Q(s_t,a_t)+\alpha\left(r_{t+1}+\gamma\max Q(s_{t+1},a_{t+1})-Q(s_t,a_t)\right)$;
    Add a new row for memory matrix $M(t)\leftarrow[s,a,r]$;
    if s_{t+1} = goal **then**
      for k = t – 1 to 1 **do**
        $Q(s_t,a_t)=Q(s_t,a_t)+\alpha\left(r_{t+1}+\gamma\max Q(s_{t+1},a_{t+1})-Q(s_t,a_t)\right)$;
      end for
      update Q-value recorded in M;
    end if
  until s_{t+1} is terminal
end while

IV. Simulation

In this section, a comparative experiment will be conducted in the grid world to prove the feasibility and effectiveness of the proposed method to solve the problem of mobile robot path planning. In the map, the black parts represent static obstacles with predefined shapes, sizes and layouts. The CQL, SARSA method and the proposed IQL method are all applied to the map and their performance indicators are compared. SARSA is a temporary strategy difference method.

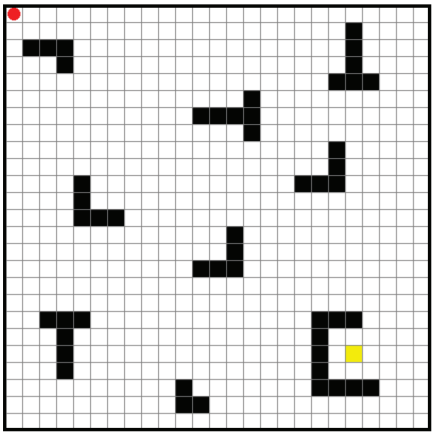

Fig. 2. 25x25 Grid World Map.

The 25x25 grid world map in the experiment is shown in Fig.2. The red dot represents the starting point, and the yellow square represents the goal point. Fig.3, Fig.4 and Fig.5 show the change in the number of steps of the mobile robot in each episode, from which we can judge the convergence of the algorithm. The experimental result of using the CQL to plan the route on the map is shown in Fig.3, and the experimental result of using SARSA to plan the route on the map is shown in Fig.4, and the experimental result of using the improved Q-learning to plan the route on the map is shown in Fig.5.

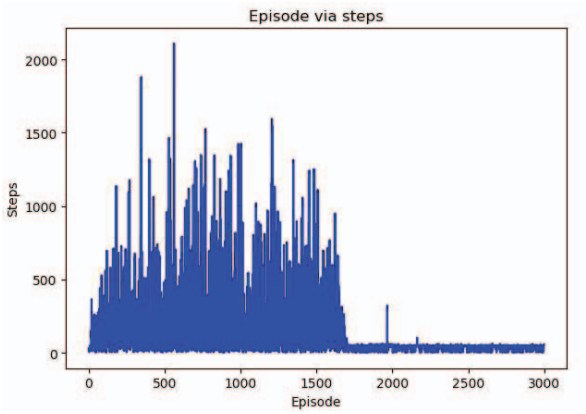

Fig. 3. Experimental Result Using Q-learning.

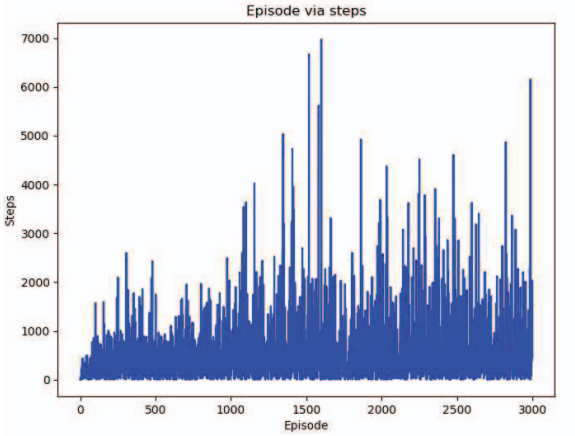

Fig. 4. Experimental Result Using SARSA.

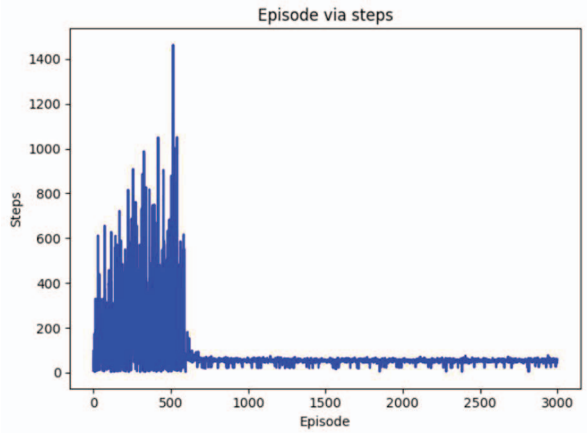

Fig. 5. Experimental Result Using The Improved Q-learning.

According to the experimental results, we can clearly see that the mobile robot using the CQL converges in about 1650 episodes, and the mobile robot using SARSA still fails to converge effectively after 3000 episodes, and can only successfully plan to reach the target point occasionally. Path, the mobile robot applying the improved Q-learning successfully converges after 650 episodes, and the convergence speed is greatly improved.

In order to avoid the contingency of the experiment, the experiment was repeated 30 times, and three performance indicators including the path length of the final successful plan, the episodes used for the algorithm to converge, and the time required to complete the program execution were recorded. Details are shown in Table I.

From Table I, we can conclude that the final path length planned by the improved Q-learning is basically the same as CQL and Sarsa, but the program execution time is greatly shortened and the convergence speed is accelerated.

TABLE I. PERFORMANCE COMPARISON BETWEEN CQL, SARSA AND IQL BASED ON THE AVERAGE RESULT OF 30 SIMULATION EXPERIMENTS.

	Final path length	***Number of episodes at convergence***	***Time consumed***
The classic Q-learning	21.5	1654	545.2s
SARSA	23.2	failed	1532s
The improved Q-learning	21.4	650	205.3s

V. CONCLUSIONS

Aiming at the problems of slow convergence speed of traditional Q learning algorithms and easy falling into local optima, an improved Q learning method combining action selection strategy and Q function update method is proposed. Compared with the traditional Q-learning algorithm, the improved q-learning algorithm reduces the total number of steps in the later stages of learning, and the search results tend to be globally optimal. Therefore, the improved Q-learning algorithm overcomes the shortcomings of local optimization and slow convergence speed, and greatly improves the learning speed, optimization ability and adaptability to the environment. The strategy is simple, effective, and better applicable to various environments. Through the simulation of the use of mobile robots, the results show that the improved algorithm is effective and feasible. Therefore, the improved Q learning algorithm has good application prospects.

VI. ACKNOWLEDGMENTS

This work was supported by the Chinese National Natural Science Foundation Projects # 61931020, # 62033010.

REFERENCES

[1] S. Michal, C. Witold, G. Karolina. Sense and avoid for small unmanned aircraft systems: Research on methods and best practices[J]. Proceedings of the Institution of Mechanical Engineers, 2019, 233.16: 6044-6062.

[2] Z. Bien and J. Lee. A minimum-time trajectory planning method for two Robots[J]. IEEE Transactions on Robotics & Automation, 2002, 8(3):414-418.

[3] L. Bo, Y. Jie. The Time-Based Multi-Robot Coordinated Collision Avoidance Algorithm[J]. Journal of Chongqing University of Technology(Natural Science), 2019,33(3):91-97.

[4] H. Miao , Y. C. Tian. Dynamic robot path planning using an enhanced simulated annealing approach[J]. Applied Mathematics & Computation, 2013, 222(Complete):420-437.

[5] Y. Zhang, D. W. Gong, J. H. Zhang. Robot path planning in uncertain environment using multi-objective particle swarm optimization[J]. Neurocomputing, 2013, 103(MAR.1):172-185.

[6] Imen, Châari, Anis, et al. SmartPATH: An Efficient Hybrid ACO-GA Algorithm for Solving the Global Path Planning Problem of Mobile Robots[J]. International Journal of Advanced Robotic Systems, 2014.

[7] W. Adiprawita, A. S. Ahmad, J. Sembiring, et al. Reinforcement learning with heuristic to solve POMDP problem in mobile robot path planning[J]. Bandung, Indonesia 2011, 17-19.

[8] V. Ganapathy, S. C. Yun, H. K. Joe. Neural Q-Learning Controller for Mobile Robot[C]. Advanced Intelligent Mechatronics, 2009. IEEE/ASME International Conference on. IEEE, 2009.

[9] Y. Xu. Research on Path Planning for Mobile Robot Based on Reinforcement Learning[D]. Shandong University, 2013.

[10] S. Li, X. Xu, L. Zuo. Dynamic path planning of a mobile robot with improved Q-learning algorithm[C]. 2015 IEEE International Conference on Information and Automation, Lijiang, China, 2015, pp. 409-414.