Robot Path Planning based on an Improved Q-learning Method

Hu Xiaomei, Xu Jun
Shanghai Key Laboratory of Intelligent Manufacturing and Robotics, School of Mechatronic Engineering and Automation, Shanghai University, Shanghai 200072, China

Chai Jianfei*
Information Technology office, School of Mechatronic Engineering and Automation, Shanghai University, Shanghai 200444, China
E-mail:554763131@qq.com

***Abstract*—Path planning is one of the most basic and pivotal aspects in research of robots, which is to solve the walking problem of robots. As a widely used reinforcement learning method, Q-learning is employed when the robot has no prior knowledge of how its actions affect its environment. For Q-learning method, there is a problem of exploration-utilization in robot path planning. Therefore, robot path planning based on an improved Q-learning method is proposed. According to the environment in which the robot is located, a Markov decision model is established to design the reward-punishment mechanism and action strategy of the robot in the path planning. During the robot training process, a heuristic search function is defined and added to the value iteration algorithm in order to reduce the invalid path exploration in the environment. The experimental results show that the proposed method not only reduces the length of path and improves the efficiency of path planning, but also accelerates the speed of robot learning. This indicates the effectiveness of the proposed method.**

Keywords—Reinforcement learning, Q-learning, path planning, heuristic search

I. INTRODUCTION

Robot path planning is defined as a problem of finding a proper collision-free path for one or more robots from a start point to a goal point with regard to different evaluation criteria [1,2]. The number of feasible paths for a mobile robot to go from a start point to the goal point is often very large. Therefore, the path planning problem is one of the most challenging tasks in mobile robotics [1,3,4,5].

Reinforcement Learning (RL) is one of machine learning methods and the robot can seek an effective strategy to solve a continuous decision task based on RL [7]. Q-learning, a stochastic dynamic programming algorithm in RL, which is based on the theory of Markov Decision Process (MDP), does not require the interactive model of a machine-environment [8,9]. However, for traditional Q-learning, this is a blind searching for a target in a simple unknown region full of obstacles. Therefore, traditional Q-learning methods may not be suitable for a complicated environment. An improved method based on Q-learning is proposed in order to overcome the shortcoming of blind searching in this paper.

II. AN IMPROVED Q-LEARNING ALGORITHM

A. Algorithm principle

The Markov Decision Process (MDP) refers to a decision maker who makes decisions sequentially in a random dynamic system. That is, according to the state observed at each moment, the decision maker takes an action selected from the available actions to make a decision. Therefore, the entire path planning problem can be abstracted as an MDP, related parameters are defined.

1) Action set $A = \{a_i\}, i \in \{up, down, left, right\}$. A robot can randomly generate an action a_i;

2) State set $S = \{s_i\} = \{(x_i, y_i)\}, i = 1,2, \ldots.$

S records the positions of the robot during the movement. $s_i \in S$ represents the position of the robot in the i^{th} state. x_i, y_i represent respectively the coordinates of s_i. The initial position of the robot is the start state. Once the robot moves to the goal state, an episode ends and the robot will reselect the end position for the next training.

$R_{a_i}(t)$ represents that the robot takes the action a_i at the time t. $R_{s_i}(t)$ represents the robot's state $s_i \in S$ at the time t.

3) Reward function r: By taking an action based on a state in each episode, the robot will get an instant reward. For example, $r_{R_{s_i}(t)}^{R_{a_i}(t)} = E[r_{R_{s_i}(t+1)}^{R_{a_i}(t+1)} | R_{s_i}(t), R_{a_i}(t)]$ where $r_{R_{s_i}(t)}^{R_{a_i}(t)}$ represents that the robot obtains the desired reward r based on the state $R_{s_i}(t)$ and the action $R_{a_i}(t)$. According to the state $R_{s_c}(t)$ and $R_{s_g}(t)$, the reward function is set as Equation (1).

$$r = \begin{cases} 1, & R_{s_c}(t) = R_{s_g}(t) \\ -1, & R_{s_c}(t) = R_{s_o}(t) \\ 0, & others \end{cases} \quad (1)$$

Where, $R_{s_c}(t)$ represents the current state, $R_{s_g}(t)$ represents the goal state of a robot at the time t and $R_{s_o}(t)$ represents the robot moves to the obstacles at the time t in the environment.

4) Value function $V_\pi(R_{s_i}(t))$: At a certain time t, the robot observes the state of the environment $R_{s_i}(t)$, and then choose an action $R_{a_i}(t)$. After executing the action, the robot receives a reward $r_{R_{s_i}(t)}^{R_{a_i}(t)}$, which evaluates how great that action is. Then, the state of the robot will change into $R_{s_i}(t+1)$ and the robot will choose the next action $R_{a_i}(t+1)$ according to related knowledge. The goal of Q-

learning is to learn a mapping from states to actions. That is, the robot is to learn a policy π. The value function represents a prediction of future returns. In accordance with a certain strategy in the current state, the robot can obtain the expected value of cumulative returns. This value is used as an indicator to evaluate a state. The expression is shown in Equation (2).

$$V_\pi\left(R_{s_i}(t)\right) = E_\pi\left[r_{R_{s_i}(t+1)}^{R_{a_i}(t+1)} + \gamma V_\pi\left(R_{s_i}(t+1)\right)\middle| R_{s_i}(t), \pi\right] \quad (2)$$

Where, $0 \le \gamma \le 1$ is discount factor that keeps a trade-off between the importance of immediate and long-term rewards. According to Equation (2), value function can be expressed as Equation (3).

$$V_\pi\left(R_{s_i}(t)\right) = \sum_{a_i \in A} \pi\left(R_{s_i}(t), R_{a_i}(t)\right)\left[r_{R_{s_i}(t)}^{R_{a_i}(t)} + \gamma \sum_{R_{s_i}(t)} p_{R_{s_i}(t)R_{s_i}(t+1)}^{R_{a_i}(t)}\right] \quad (3)$$

In Equation (3),

$$p_{R_{s_i}(t)R_{s_i}(t+1)}^{R_{a_i}(t)} = Pr[R_{s_i}(t+1) | R_{s_i}(t), R_{a_i}(t)] \quad (4)$$

Where, $p_{R_{s_i}(t)R_{s_i}(t+1)}^{R_{a_i}(t)}$ represents the probability that the robot takes an action $R_{a_i}(t)$ and shifts to state $R_{s_i}(t+1)$. $\pi\left(R_{s_i}(t), R_{a_i}(t)\right)$ is the probability that the robot selects action $R_{a_i}(t)$ according to state $R_{s_i}(t)$ under policy π. Therefore, the optimal state-value function can be expressed as Equation (5).

$$V_\pi^*\left(R_{s_i}(t)\right) = max_{a_i \in A} r_{R_{s_i}(t)}^{R_{a_i}(t)} + \gamma \sum_{R_{s_i}(t+1)} p_{R_{s_i}(t)R_{s_i}(t+1)}^{R_{a_i}(t)} V_\pi^*(R_{s_i}(t+1)) \quad (5)$$

Each iteration process in the strategy iteration algorithm is usually composed of two parts: strategy evaluation and strategy improvement. In the strategy evaluation, the value function $V_\pi\left(R_{s_i}(t)\right)$ is calculated according to the current strategy. In the strategy improvement, the value function $V_\pi\left(R_{s_i}(t)\right)$ is maximized to $V_\pi^*(R_{s_i}(t))$ and the strategy is improved. The strategy iteration algorithm repeats each iteration process until the optimal strategy π^* is converged. According to Equation (4), the optimal strategy π^* can be expressed as Equation (6).

$$\pi^* = arg\max_\pi V_\pi\left(R_{s_i}(t)\right), \forall R_{s_i}(t) \in R_S(t) \quad (6)$$

5) Q value function $Q_\pi\left(R_{s_i}(t), R_{a_i}(t)\right)$: The value iterative algorithm continuously iterates the Q value, and finally all the Q values converge to an optimal value. $Q_\pi\left(R_{s_i}(t), R_{a_i}(t)\right)$ can be regarded as a Q function when an action $R_{a_i}(t)$ has been executed under the state $R_{s_i}(t)$, and the optimal Q value could be obtained. For Q-learning which learns the value function over state-action pairs, Q value function can be expressed as Equation (7).

$$Q_\pi^*\left(R_{s_i}(t), R_{a_i}(t)\right) = r_{R_{s_i}(t)}^{R_{a_i}(t)} + \gamma \sum_{R_{s_i}(t+1)} p_{R_{s_i}(t)R_{s_i}(t+1)}^{R_{a_i}(t)} \pi^* \quad (7)$$

Where, $Q_\pi^*\left(R_{s_i}(t), R_{a_i}(t)\right)$ stands for the value of taking action $R_{a_i}(t)$ in state $R_{s_i}(t)$ under policy π^*. The recursive definition of Q-function provides the basis for the Q-learning algorithm. The one-step update formula of Q-learning is Equation (8).

$$Q_\pi\left(R_{s_i}(t), R_{a_i}(t)\right) = (1-\alpha) Q_\pi\left(R_{s_i}(t), R_{a_i}(t)\right) + \alpha Q_\pi^*\left(R_{s_i}(t), R_{a_i}(t)\right) \quad (8)$$

Where, $0 \le \alpha < 1$ is the learning rate that defines the importance of recently obtained information compared to old information in updating Q value. Therefore, Q value function is a predictive function that estimates the expected return from the current state-action pair.

6) Heuristic function $H_\pi(R_{s_i}(t), R_{a_i}(t))$:

Set $R_{s_c}(t) = (x_c, y_c)$, where x_c, y_c represent respectively the coordinates of $R_{s_c}(t)$. $R_{s_{n_i}}(t)$ represents the robot's neighbor states at the time t, which are adjacent of the current state.

$$R_{s_{n_i}}(t) = \left(x_{n_i}, y_{n_i}\right), \mathrm{i} = \{\mathrm{up}, \mathrm{down}, \mathrm{left}, \mathrm{right}\} \quad (9)$$

Where, x_{n_i}, y_{n_i} represent respectively the coordinates of $R_{s_{n_i}}(t)$. Set $R_{s_g}(t) = (x_g, y_g)$, where x_g, y_g represent respectively the coordinates of $R_{s_g}(t)$. D represent the distance between two states, for example, $D(R_{s_c}(t), R_{s_g}(t))$ represents the distance between the current state $R_{s_c}(t)$ and the goal state $R_{s_g}(t)$.

According to the comparison of $D(R_{s_c}(t), R_{s_g}(t))$ and $D(R_{s_{n_i}}(t), R_{s_g}(t))$, $R_{s_{n_{up}}}(t)$ and $R_{s_{n_{left}}}(t)$ could be deleted in next path searching episode. Therefore, for the purpose of improving the learning efficiency, a suitable heuristic function $H_\pi(R_{s_i}(t), R_{a_i}(t))$ is set as Equation (10).

$$H_\pi(R_{s_i}(t), R_{a_i}(t)) = max(D(R_{s_c}(t), R_{s_g}(t)), D(R_{s_{n_i}}(t), R_{s_g}(t))) \quad (10)$$

Where, $D(R_{s_c}(t), R_{s_g}(t)) = \sqrt{(x_g - x_c)^2 + (y_g - y_c)^2}$,

$D(R_{s_{n_i}}(t), R_{s_g}(t)) = \sqrt{(x_g - x_{n_i})^2 + (y_g - y_{n_i})^2}$.

The heuristic function $H_\pi(R_{s_i}(t), R_{a_i}(t))$ is applied to $Q_\pi\left(R_{s_i}(t), R_{a_i}(t)\right)$, Equation (8) could be rewritten by Equation (11).

$$Q_\pi\left(R_{s_i}(t), R_{a_i}(t)\right) = (1-\alpha) Q_\pi\left(R_{s_i}(t), R_{a_i}(t)\right) + \alpha H_\pi(R_{s_i}(t), R_{a_i}(t)) \quad (11)$$

Equation (11) not only considers the Q value under random environment, but also accelerates the convergence of Q value.

B. Algorithm description

In order to improve the efficiency of Q-learning, it is worthy of exploring how to utilize reinforcement learning without prior knowledge. A mechanism of search-space reduction was proposed in section 3.1, in which a robot could reduce the state space gradually. The nature of the algorithm is giving up some states during the path searching. In the proposed algorithm, Open_list is used to store the current state and its neighbor states. Closed_list is used to store the left states after each iteration. According to Q value function $Q_\pi\left(R_{s_i}(t), R_{a_i}(t)\right)$, Q table stores Q value after each episode. When fewer states are deposited in the state space, the algorithm efficiency will be promoted.

A heuristically accelerated Q-learning algorithm solves an MDP problem with explicit use of a heuristic function $H_\pi(R_{s_i}(t), R_{a_i}(t))$ for influencing the choice of actions by the learning robot. The algorithm is described.

Initialize Open_list, Closed_list and Q table;

Repeat (for each episode):

a) **Initialize** the parameters γ, α;

b) **Repeat** (for each step of episode):

Open_list← $R_{s_c}(t), R_{s_{n_i}}(t)$;

Calculate $D\left(R_{s_c}(t), R_{s_g}(t)\right)$ according to Equation (10);

If $D\left(R_{s_c}(t), R_{s_g}(t)\right)$=0:

Break;

Else:

$R_{s_c}(t) \leftarrow R_{s_{n_i}}(t)$;

For each $R_{s_{n_i}}(t)$:

Calculate $D(R_{s_{n_i}}(t), R_{s_g}(t))$ according to Equation (10);

If $D(R_{s_{n_i}}(t), R_{s_g}(t)) < D(R_{s_c}(t), R_{s_g}(t))$:

Set $H_\pi(R_{s_i}(t), R_{a_i}(t)) \leftarrow D(R_{s_c}(t), R_{s_g}(t))$;

Delete $R_{s_{n_i}}(t)$, add $R_{s_{n_i}}(t)$;

State S ←Closed_list;

Else:

Set $H_\pi(R_{s_i}(t), R_{a_i}(t)) \leftarrow D(R_{s_{n_i}}(t), R_{s_g}(t))$;

Delete $R_{s_c}(t)$, add $R_{s_c}(t)$

State S ←Closed_list;

Update Open_list, Closed_list and State S;

Endif

Endfor

Endif

Execute $R_{a_i}(t), r, R_{s_i}(t+1)$

Calculate $\pi^*$from $V_\pi\left(R_{s_i}(t)\right)$

Update Q table: e.g., ε−greedy

Calculate $Q_\pi\left(R_{s_i}(t), R_{a_i}(t)\right)$ according to Equation (11);

$R_{s_i}(t) \leftarrow R_{s_i}(t+1)$;

Until $R_{s_i}(t)$ is terminal;

Until the learning process ends.

III. Experiment Result and Analysis

In the experiment, a complicated packaging workshop environment is built up with the setting of 30×30 (grid representation) to test the feasibility of the proposed method. In this packaging environment, a pink block represents the start position, a red block represents the end position, the sequential black blocks are obstacles, the white blocks are efficient points that robot could be arrived in path planning. The start state and the goal state are $R_{s_s}(t) = (23,22)$ and $R_{s_g}(t) = (9,3)$, respectively. The parameter setting for the learning algorithms is as follows: exploration policy $\varepsilon - greedy$, $\varepsilon = 0.1$, discounted factor $\gamma = 0.9$, learning rate $\alpha = 0.01$, all the Q values are initialized at 0.

Fig. 1 is the statistical analysis of steps per episode of the three methods mentioned above. The whole process of searching has been repeated over 5000 times. Meanwhile, for a clear comparison of the three path planning methods, path length is recorded every 200 episodes in Fig. 1. The result indicates that the proposed method not only reduces the length of path and improves the efficiency of path planning, but also accelerates the speed of robot learning.

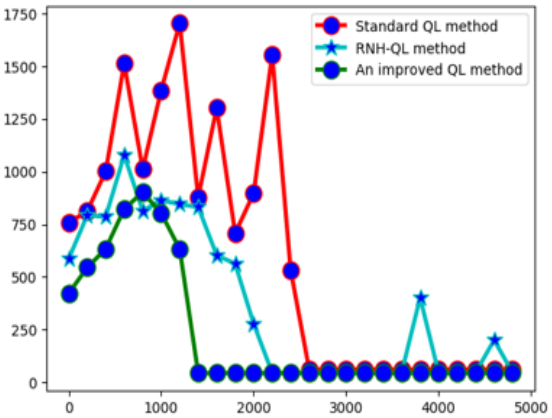


Fig.1. The performance analysis of three methods

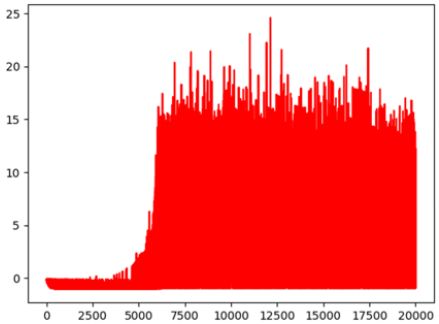
Cost value C is generated by the robot moving to the obstacle or the goal state. Meanwhile, cost value is calculated to evaluate the state space during each episode in robot's path planning. The robot will consume energy from state $R_{s_i}(t)$ to state $R_{s_i}(t+1)$ during each step in a navigation task. Cost value C is calculated according to the reward function r. The formula is shown in Equation (12).

$$C = r_{R_{s_i}(t)}^{R_{a_i}(t)} + \gamma r_{R_{s_i}(t+1)}^{R_{a_i}(t+1)} + \gamma^2 r_{R_{s_i}(t+2)}^{R_{a_i}(t+2)} + \cdots$$

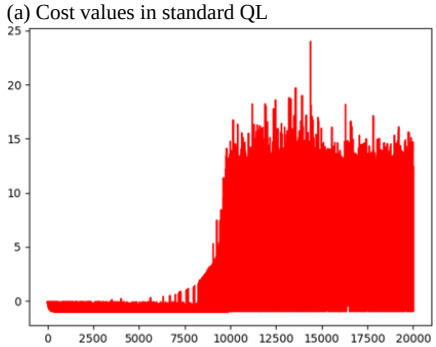
$$= \sum_{i=0} \gamma^i r_{R_{s_i}(t+i)}^{R_{a_i}(t+i)} \quad (12)$$

The three processes of path searching have been repeated 20000 times and the cost value per episode is shown in Fig. 2. As shown in Fig. 2(a), the cost value becomes stable at about 6000 rounds in standard QL algorithm. In Fig. 2(b), the cost

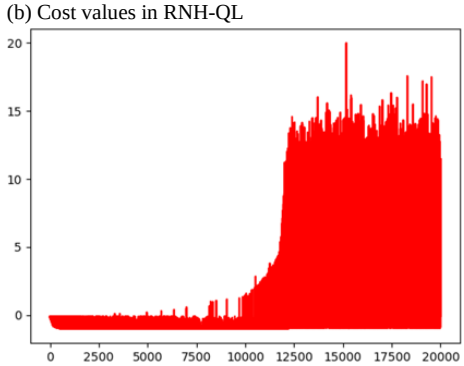
value of RNH-QL [9] algorithm becomes stable at about 9500 rounds. However, the cost value of an improved QL algorithm becomes stable at about 12000 rounds in Fig.2(c). Therefore, an improved QL algorithm performs a good result, which demonstrates that the robot has less state space in the process of path planning. On the other hand, the cost value in each episode also represents the energy consumption of the robot. In contrast, the result illustrates that the robot has the least energy consumption in an improved QL algorithm.



(a) Cost values in standard QL



(b) Cost values in RNH-QL



(c) Cost values in an improved QL

Fig.2. Cost values in three methods

Fig. 3 shows the experimental results of path planning based on improved QL. In this complicated packaging workshop environment, the robot has to go through many fixed obstacles and some possible dynamic obstacles before it reaches the goal state $R_{s_g}(t)$. The result shows that the robot moves to the global goal without being trapped in a local minimum through learning. The path searching using the proposed algorithm makes the robot competent to navigate in a large unknown environment and to adapt to dynamic environments as well.

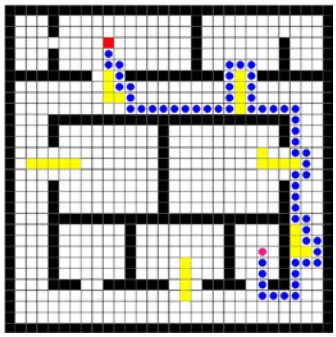


Fig.3. The performance of robot in an environment with more dynamic obstacles

IV. Conclusion

For standard Q-learning method and some existing improved methods, there is a problem of invalid path exploration in robot path planning. To support mobile robots to effectively make a better decision, a heuristic function is defined and it is applied to the value iteration algorithm in the path searching missions. The robot path planning algorithm based on the improved Q-leaning aims at speeding up the learning process and reducing the searching time. The experimental results show that the algorithm has the advantage of more adaptability to the environment for a robot in a complex environment. Meanwhile, it can help the robot reduce a large amount of state-action information by utilizing the heuristic function during the path planning process, which improves the efficiency of path planning.

References

[1] Tsai C-C, Huang H-C, Chan C-K (2011) Parallel elite genetic algorithm and its application to global path planning for autonomous robot navigation. IEEE Trans Ind Electron 58(10):4813-4821.

[2] Fetanat M, Haghzad S, Shouraki SB (2015) Optimization of dynamic mobile robot path planning based on evolutionary methods. In: IEEE AI & Robotics (IRANOPEN), 2015.

[3] Soltani AR et al (2002) Path planning in construction sites: performance evaluation of the Dijkstra, A*, and GA search algorithms. Adv Eng Inform 16(4):291-303.

[4] Zhang J Q, Hu X M, Kang J S, Xiong F, Zeng N (2017) Novel Cylinder Movement Modeling Method Based on Aerodynamics. Chinese Journal of Mechanical Engineering 30(5):1193-1202.

[5] Konar A et al (2013) A deterministic improved Q-learning for path planning of a mobile robot. IEEE Trans Syst Man Cybern Syst 43(5):1141-1153.

[6] Das P, Behera H, Panigrahi B (2015) Intelligent-based multi-robot path planning inspired by improved classical Q-learning and improved particle swarm optimization with perturbed velocity. Int J Eng Sci Technol 19:651-669.

[7] Banerjee D, et al (2012) Path-planning of mobile agent using Q-learning and real-time communication in an unfavorable situation. In: 2012 world congress on information and communication technologies (WICT).

[8] Li S, Xu X, Zuo L (2015) Dynamic path planning of a mobile robot with improved Q-learning algorithm. In: 2015 IEEE international conference on information and automation.

[9] Zhang Fengyun, Duan Shukai, Wang Lidan. Route searching based on neural networks and heuristic reinforcement learning. Cogn Neurodyn (2017)11:245-258