Path Planning in Dynamic Environments Based on Q-Learning

Xiangqi Li*

School of Computer Science, Ocean University of China, Qingdao, China

*Corresponding author: lixiangqi@stu.ouc.edu.cn

Abstract. With the rapid progress of science and technology, the scope of applications for mobile robots is growing. Path planning in dynamic environment is always a challenging task for mobile robot, which shows significant impacts in the medical field and the military field. Q-learning, a model-free reinforcement learning algorithm, can recognize its surroundings and demonstrate a system making decisions for itself about how this algorithm learns to make accurate decisions about achieving optimal target. Therefore, a path planning algorithm was proposed for robots in dynamic environments based on Q-learning. This algorithm can successfully generate several viable paths in environments with both static obstacles, dynamic obstacles and target point. First, three environments with different levels of complexity were created. Afterwards, to generate several optimal paths, the path planning algorithm was conducted in multiple times in different environments. Finally, the experimental results were collected and visualized to illustrate information. The effectiveness of the proposed algorithm is validated by experiment results in solving problems of path planning in dynamic environments with required point.

Keywords: Q-learning, Path planning, Dynamic obstacle avoidance, Reinforcement learning.

1. Introduction

Path planning is always a challenging problem for its diversity of different problems and wide application in reality in robotic fields. For example, medical micro-surgical robots require the robot to avoid colliding with organs in the human body as much as possible and reach certain designated positions [1]. The garbage-clearing robot vehicle with automatic navigation should take into account the real state of the road, avoid running over pedestrians and other stationary objects, and pick up the trash in need [2]. In most situations, finding the optimal path between starting point and ending point is the main purpose for path planning problem.

A* algorithm and Dijkstra algorithm, two representatives of tradition path planning algorithms, were proposed to find shortest path [3]. A* algorithm and Dijkstra algorithm are both powerful approach in finding the shortest path in environments by estimating the minimum distance between two points.

However, there are two limitations for A* algorithm and Dijkstra algorithm. First, these conventional algorithms could only function effectively in static environments. Local and global path planning are two categories of path planning [4]. To create a new path in response to an environment change, the local sensor data acquired during navigation is employed for local path planning. On the other hand, if only static obstacles are present and localized before robot's exploration, a complete path can be created based on global path planning and its algorithms. Due to the inability of global path planning to handle uncertainty and a dynamic environment, this navigational approach has a limited range of applications. A* algorithm and Dijkstra algorithm are both informed search algorithms for global path planning that heavily rely on estimating the cost of movement between two given points. Consequently, the A* algorithm and Dijkstra algorithm may fail to find optimal paths in dynamic environments as they have little knowledge of the trajectory of dynamic obstacles. Second, A* algorithm and Dijkstra algorithm may lose competence when achieving some additional tasks in path planning process. Robots are usually expected to achieve some requirements by reaching some predetermined points in map in practical scenarios. In this case, the optimal path may not be the shortest path between starting point and ending point, because the optimal path sometimes sacrifices some advantages in length in order to reach the predetermined point. However, A*

algorithm and Dijkstra algorithm could only consider the minimum length for path, which make them incapable in meeting the balance between shortest path and predetermined requirements.

Q-learning is a reinforcement learning algorithm. By using Q-learning, an agent can learn to find actions with highest reward to reach the goal in dynamic environments [5,6]. Much attention has been spilled on Q-learning due to its ability to do self-learning without the need for an a priori environment model, which offers the chance to overcome the first constraint mentioned for the A* algorithm and Dijkstra algorithm [7]. Furthermore, the mobile robot also performs an action and is immediately rewarded since new information can be learned and updated by interacting with environments constantly [4]. In creating environments process, the reward values are predetermined to encourage robot to consider all additional requirements, which can overcome the second limitation successfully. Based on the obtained reward and the state action-value function, the Q-value is updated and subsequent actions are continually chosen [8]. Eventually, the optimal path is generated by considering Q Table the states with the highest Q-value stored in Q Table. With Q-learning, autonomous mobile robots can travel to their destination without colliding with other obstacles and achieving additional requirements through self-learning.

Many studies in using Q-Learning Algorithm have successfully solved path planning problem in robotic field. A path planning algorithm in static environments based on Q-learning was successfully achieved, which could search for the optimal path in environment with plenty of static obstacles [9]. To overcome some Q-Learning algorithm flaws like its sluggish convergence to ideal solutions, an Efficient Q-Learning (EQL) algorithm was proposed [10]. Dynamic and fast Q-learning (DFQL) was introduced to tackle the path planning problem for Unmanned Surface Vessels (USV), taking into account the possibilities of employing it in a practical situation [11]. Additionally, deep reinforcement learning is also promising in solving path planning problems [12]. Based on deep reinforcement learning algorithm, soft actor-critic (SAC), a path planning method for manipulator was proposed, which could avoid the dynamic obstacles and enable real-time planning [13]. Specifically, to realize dynamic obstacle avoidance, a novel function for calculating reward was designed.

A path planning algorithm based on Q-learning is proposed to help robots find optimal path in virtual dynamic environments. Find the optimal path with the ability to reach target point and ending point while avoiding collision with both static obstacles and dynamic obstacles is the main goal in environment maps of different complexity based on Q-learning Algorithm. Three environment maps of different complexity have been created first. In environment maps, the obstacles, starting point, target point and ending point are distinguished by different colors. Then the training process is carried for multiple times to generate Q-Tables and find different solutions in different environment maps. Finally, experiment results are recorded and visualized, including environment maps, optimal paths and detailed graphs to illustrate training progress.

2. Methodology

The implementation of Q-learning is introduced in this section. Three subsections are divided in this section: Develop Model, Create Environment, and Path Planning Algorithm based on Q-learning, which introduces the development of model, the creation of the environments in details, and specific description on implementation of Q-learning for the proposed path planning algorithm, respectively.

2.1. Develop Model

A garbage cleaning vehicle is modeled in this paper. The main task for garbage cleaning vehicle is to pick up the trashes and deliver them to trash bin while avoiding collision with existing obstacles, such as humans and roadblocks.

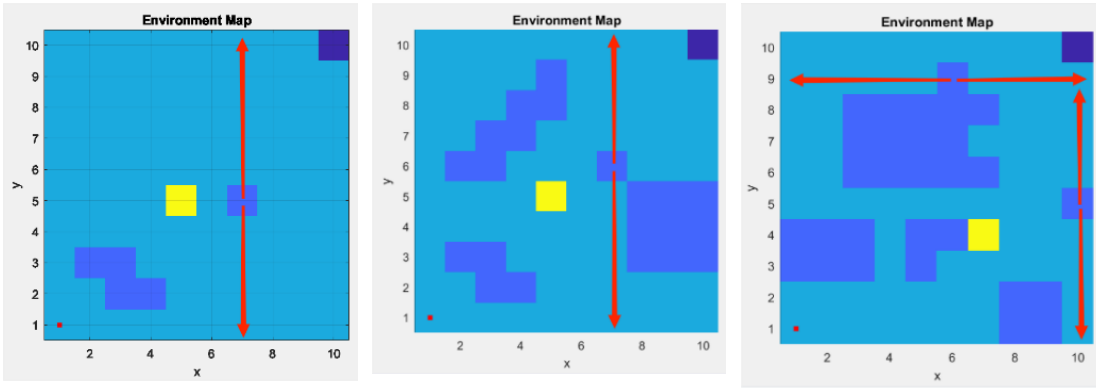
Virtual maps are constructed with different components to simulate the garbage cleaning vehicle. The starting point and ending point will denote the current location and location of trash bin, respectively. The trash point is set to model the location of trash, which is the required point for robot

to reach. Static obstacles and dynamic obstacles are used to simulate the humans and roadblocks in environments.

2.2. Create Environment

In this paper, some virtual obstacle environments in MATLAB are created for proposed algorithm. The empty environment should be constructed firstly with the starting point at (0, 0) in red. The ending point is located at (10, 10), which will be filled with navy blue. Some cells are picked as the trash points in yellow representing positive reward value when reaching these points. In terms of obstacles, both static and dynamic obstacles are in sky blue. The movement speed of dynamic obstacles and the movement speed of the car are always consistent. The red arrows in environment maps suggest the trajectory of dynamic obstacles. Other cells would be filled in cyan to suggest that the current cell is free of obstacles.

Three different environment maps are illustrated in Figure 1, When the number of obstacles increases, the complexity of environments and difficulty of Q-Learning process may also grow correspondingly. The map of environment 1 shown in Figure 1a is composed of only one static obstacle and one dynamic obstacle. The motion range of the dynamic obstacle is from (7, 2) to (7, 10), starting at (7, 2). The map of environment 2 shown in Figure 1b is composed of two static obstacle and one dynamic obstacle. The motion range of the dynamic obstacle is from (7, 2) to (7, 10), starting at (7, 2). The trash points in these two environments are both located at (5, 5). The map of environment 3, which is also the most complex one, is shown in Figure 1c and composed of more static obstacles and more dynamic obstacles. The motion ranges of the dynamic obstacles are from (10, 1) to (10, 9) and from (1, 9) to (10, 9), starting at (10, 1) and (5, 9), respectively. The location of trash point is changed to (7, 4).



1a 1b 1c

Fig. 1 Map of Environment 1, 2 and 3

2.3. Path Planning Algorithm Based on Q-Learning

This section mainly presents the structure and details of proposed algorithm, including Q-learning Algorithm, Actions and Q-Table Initialization, Action Selection and Rewards, and Basic Flow of Algorithm. The Q-learning Algorithm part shows the basic ideas and parameter settings of Q-learning algorithm. Ulteriorly, more introduction about actions, Q-Table and rewards are illustrated in next two parts, namely Actions and Q-Table Initialization and Action Selection and Rewards. The basic flow of the algorithm is introduced eventually with the flow chart shown in Figure 2.

2.3.1 Q-learning algorithm

Q-Learning is a value-based algorithm. The primary notion of Q-learning is to construct Q-Table filled with Q values according to State and Action. Next actions which can reach the maximum rewards will be determined based on Q-Table. Employing the State-Action value function, Q values are calculated and stored in Q-Table. The function is shown in Equation (1). $Q(s_t, a_t)$ is the expected reward value when an action a_t is determined in state s_t.

$$\Delta Q_t(s_t,\, a_t) = \alpha[r_{t+1} + \gamma * (max) \; * \; Q_t(s_{t+1},\, a_t) - Q_t(s_t,\, a_t)] \qquad (1)$$

In equation (1), two parameters α and γ play significance roles in the training process. α is the learning rate, representing the coverage extent to which old information is overridden by new information [14]. γ is the discount factor, representing the importance of future rewards [14]. Both α and γ are adjusted in different environments to contribute favors for the training process. To simplify the problems, γ is set as a constant value at 0.25. α is initialized at 0.99 and decreased by 0.01 every 100 episodes.

2.3.2 Actions and Q-Table initialization

For actions, four actions are applied to environment 1 and environment 2, which only include up, down, left, and right to make simple motions. To prove the feasibility of Q-Learning algorithm in more complex condition, eight actions are applied to environment 3, including up, down, left, right, upper left, lower left, lower right, and upper right.

For Q-Table, it should be initialized in its size with only 0. Q-Table should be an array in three dimensions. The number of layers in Q-table and the number of actions set in current environment are consistent. For each layer, the Q-Table will restore a grid map in same size as the environment. Each number in Q-Table layers represents a specific state in environment maps. Considering different numbers of actions applied in different environments, the Q-Tables in environment 1 and environment 2 contains four layers. While for environment 3, the Q-Table has eight layers.

2.3.3 Action selection and rewards

Action selection is a significant part for conducting Q-Learning process. Basically, an action will be selected for next step with the highest Q value according to Q-Table established before. In proposed algorithm, if more than one actions has same highest Q value, then the action will be selected randomly to get highest Q value.

The setting of rewards is fundamental for Q-Learning algorithm. When the agent receives rewards, update will occur in Q-Table according to Equation (1). In environment 1 and environment 2, the reward values for different component in maps are consistent. Reaching obstacles or boundaries may result in -1 as reward value. While reaching trash point and ending point could achieve both 5 as reward values. To stress the differences of environment 3, the reward value for trash point is altered to 10, while other settings remain unchanged.

2.3.4 Basic flow of algorithm

The algorithm is mainly divided into three parts shown in Figure 2, including pre-processing, Q-learning training and results visualizing. In pre-processing part, environment maps are generated, and Q-Table is initialized with actions with relevant details shown in 3.2 and 3.3.2. After pre-processing, Q-learning training process is carried out. In every training episode, the dynamic environments are updated in the same pace of vehicle movement. The Q values are computed and updated in Q-Table following Equation (1) in 3.3.1 based on the reward settings introduced in 3.3.3. Additionally, if the robot hits obstacles or boundaries, it will receive -1 as reward value and terminate the current episode. The reward of ending point is only validated after gaining reward of trash point, which will result in the correct order for reaching trash point and ending point. Finally, the algorithm will stop the Q-learning training process when completing a certain number of iterations and visualized the experiment results.

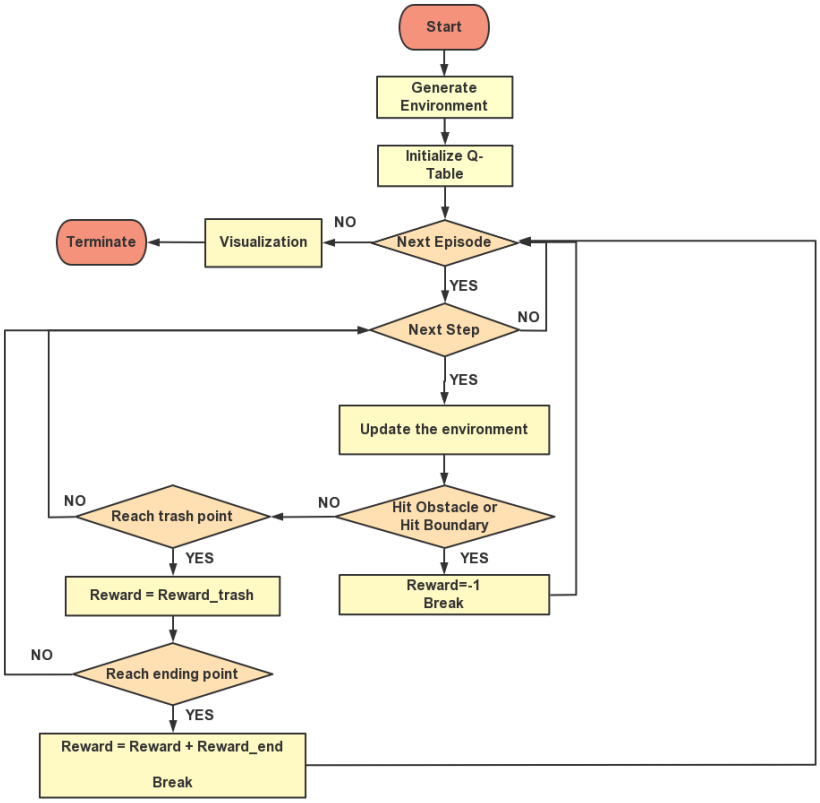


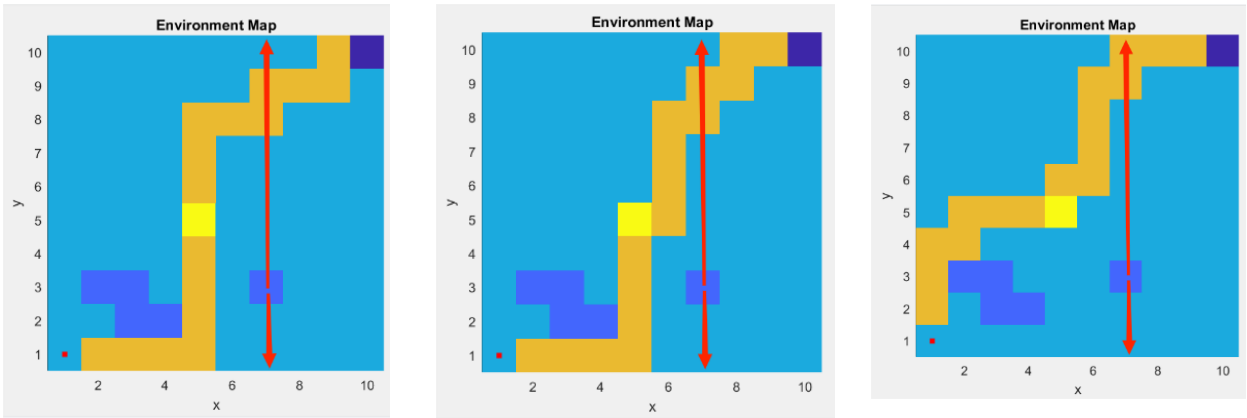
Fig. 2 The Flow Chart of Proposed Algorithm

3. Results

This section includes three different solutions for three environment maps of different complexity. Three solutions shown in 4.1, 4.2, 4.3 are generated from environment 1, 2, and 3, respectively. In each solution, the three most representative optimal paths are picked to illustrate experiment results. Additionally, three corresponding training graphs and action tables at (1, 1) are listed as the supplement of optimal paths.

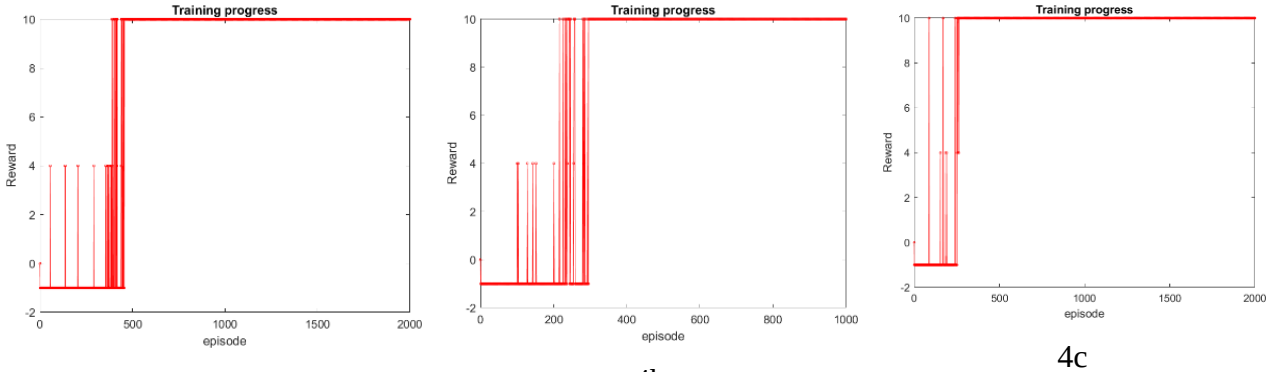
3.1. Solution 1

The environment map for solution 1 is illustrated in Figure 1a. Following the introduction in 3.3.2 and 3.3.3, three optimal paths numbered as 1-1, 1-2, and 1-3 are presented in orange line in Figure 3a, Figure 3b and Figure 3c, and the corresponding graph of training progress are illustrated in Figure 4a, Figure 4b and Figure 4c. The total lengths for three optimal paths are all 17 steps.



3a 3b 3c

Fig. 3 Three Optimal Path 1-1, 1-2 and 1-3



4a 4b 4c

Fig. 4 Training Progress Graph of Optimal Path 1-1, 1-2 and 1-3

From Figure 4, the convergence in environment 1 may fall within the episode interval [250, 500].

After training, Q-learning will generate Q-Table to store Q values. The Q Tables for three optimal path are all three dimensional array (10×10×4). In Q-Table, four actions including down, up, right and left are indicated as "1", "2", "3", and "4", respectively. Table 1 illustrates the Q-values of four actions at starting point (1,1).

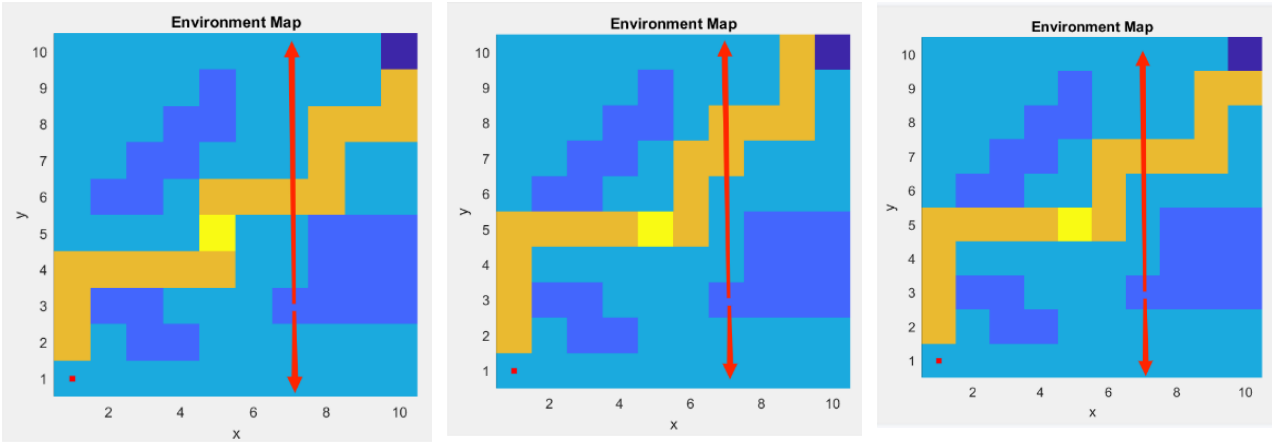
Table 1. The Q Values of Each Action at (1,1) in Solution 1

	Action 1	Action 2	Action 3	Action 4
Optimal path 1-1	-1.0000	0	0.0003	-1.0000
Optimal path 1-2	-1.0000	0	0.0003	-0.9999
Optimal path 1-3	-0.9900	0.0003	0	-1.0000

From Table 1, Optimal path 1-1 and 1-2 achieve highest value in choosing Action 3, which represents right direction. However, optimal path 1-3 reach highest value in choosing Action 2, which represents up direction. In Figure 3, all optimal paths are presented correct choice for highest value at the starting point (1, 1).

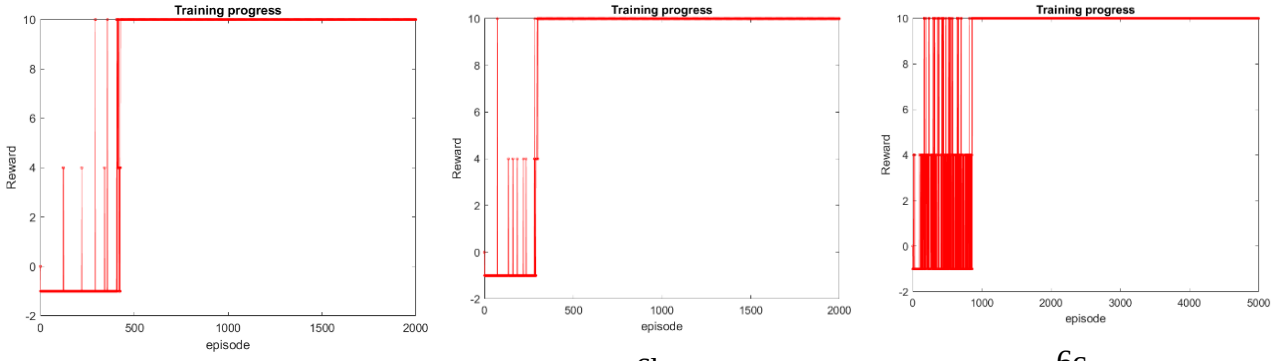
3.2. Solution 2

The environment map for solution 2 is illustrated in Figure 1b. Three optimal paths numbered as 2-1, 2-2, and 2-3 are presented in orange line in Figure 5a, Figure 5b and Figure 5c, and the corresponding graph of training progress are illustrated in Figure 6a, Figure 6b and Figure 6c. The total lengths for three optimal paths are all 17 steps.



5a 5b 5c

Fig. 5 Optimal Path 2-1, 2-2 and 2-3



6a 6b 6c

Fig. 6 Training Progress Graph of Optimal Path 2-1, 2-2 and 2-3

From Figure 6, the convergence in environment 2 may fall within the episode interval [250, 1000].

The Q Tables for three optimal paths in environment 2 are in the same settings as environment 1. Table 2 illustrates the Q-values of four actions at starting point (1,1).

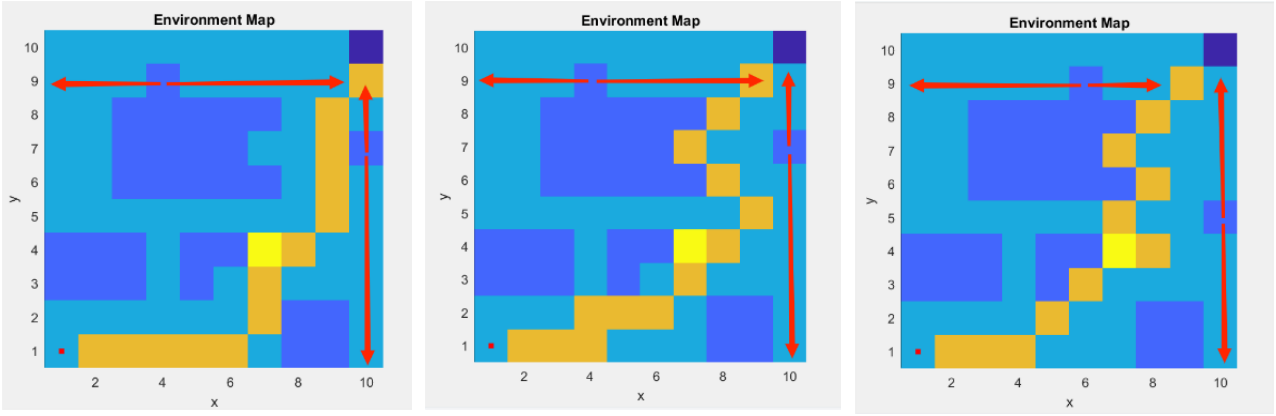
Table 2. The Q Values of Each Action at (1,1) in Solution 2

	Action 1	Action 2	Action 3	Action 4
Optimal path 2-1	-1.0000	0.0003	0	-1.0000
Optimal path 2-2	-1.0000	0.0003	0	-1.0000
Optimal path 2-3	-0.9999	0.0003	0	-0.9900

From Table 2, Optimal path 2-1, 2-2 and 2-3 achieve highest value in choosing Action 2, which represents up direction. In Figure 5, all optimal paths are presented correct choice to go upward for highest value at the starting point (1, 1).

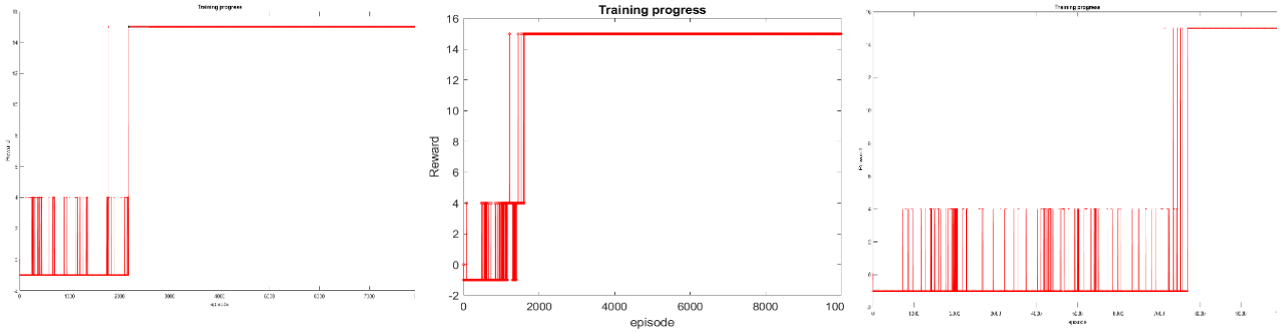
3.3. Solution 3

The environment map for solution 3 is illustrated in Figure 1c. Three optimal paths numbered as 3-1, 3-2, and 3-3 are presented in orange line in Figure 7a, Figure 7b and Figure 7c, and the corresponding graph of training progress are illustrated in Figure 8a, Figure 8b and Figure 8c. The total lengths for 3-1 and 3-2 are 14 steps, while the total length of optimal path 3-3 is 12 steps.



7a 7b 7c

Fig. 7 Optimal path 3-1, 3-2 and 3-3.



8a 8b 8c

Fig. 8 Training progress graph of optimal path 3-1, 3-2 and 3-3.

From Figure 8, the convergence in environment 3 may fall within the episode interval [1500, 8000].

The Q-Tables for three optimal paths are all three-dimensional array. However, due to the additional four actions, Q-Tables in environment 3 contains 8 layers (10×10×8). In Q-Tables, eight actions including down, up, right, left, left-up, right-up, left-down and right-down are indicated as "1", "2", "3", "4", "5", "6", "7", and "8", respectively. Table 3 illustrates the Q-values of eight actions at starting point (1,1).

Table 3. The Q Values of Each Action at (1,1) in Solution 2

	Action 1	Action 2	Action 3	Action 4	Action 5	Action 6	Action 7	Action 8
Optimal Path 3-1	-0.9900	0	0.0001	-1.0000	-1.0000	-0.9900	-1.0000	0
Optimal Path 3-2	-0.9900	0	0.0001	-1.0000	-1.0000	-1.0000	-0.9999	-0.9900
Optimal Path 3-3	-1.0000	0	0.0016	-0.9999	-1.0000	-0.9900	-1.0000	-0.9900

From Table 3, Optimal path 3-1, 3-2 and 3-3 achieve highest value in choosing Action 3, which represents right direction. In Figure 7, all optimal paths are presented correct choice to go rightward for highest value at the starting point (1, 1).

4. Discussion

The proposed algorithm can successfully solve the problem of finding multiple optimal paths in dynamic environment with special requirements. From the experiment results, the vehicle can seek for optimal and collision-free path to reach both trash point and ending point to achieve highest reward value in different environment maps.

Considering the three sets of solutions for different environment maps, extra attention should be spent in two aspects. First, the convergence rate decreases as the complexity of environment maps and the mobility of vehicle increases. This may cause additional time cost in completing the training process. The use of lightweight neural networks may help solve this problem, which shows great potential in deep reinforcement learning. In addition, the optimized strategy applied in selecting actions with same highest reward may help robot to achieve better trade-off between exploitation and exploration, which can result in faster convergence. Second, in the third environment map, the difference of total length may suggest that some redundant movements have been made. The vehicle can do eight actions in the third environment map including down, up, right, left, left-up, right-up, left-down and right-down. For example, in optimal path 3-2 shown in Figure 7b, when the optimal path marked by the orange line reaches (3, 1), the vehicle needs to first move rightward to reach (4, 1), and then move upward to reach (4, 2). However, this path can be simplified as moving to the upper right from (3, 1) to (4, 2) directly. This is mainly caused by the only standard, Q Table, when choosing next action. The vehicle will totally depend on Q Table in finding next action with highest Q value, which can imply the highest probability to achieve positive reward. Since Q learning could be implemented without the prior model of environment maps, this algorithm lacks the ability to do global path planning, which can concisely reach minimum length of optimal paths.

5. Conclusion

This paper proposes a novel path planning algorithm to seek for multiple collision-free paths based on Q-learning in three different dynamic environment maps with different number of obstacles. Planning optimal path in dynamic environments with numerous obstacles requires the creation of virtual environment maps and the implementation of Q-learning algorithm. The optimal paths in different environment maps are demonstrated and analyzed with the supplement of Q values at staring point. Additionally, the training progress graphs are created using the rewards of each state over the course of each episode. Experiment shows that the algorithm can successfully find multiple optimal paths in dynamic environment and reach the predefined scoring points. Future study will pay close attention to the possibility of increasing convergence rate in complex environments, which will show great potentials and promising futures in practical applications.

References

[1] R. D. Field, P. N. Anandakumaran, and S. K. Sia, "Soft medical microrobots: Design components and system integration," Applied physics reviews, vol. 6, p. 041305, 2019.

[2] L. Piardi, J. Lima, A. Pereira, and P. Costa, "Coverage path planning optimization based on q-learning algorithm," AIP Conference Proceedings, vol. 2116, 07 2019.

[3] Delling, Daniel, et al. "Engineering route planning algorithms." Algorithmics of large and complex networks. Springer, Berlin, Heidelberg, 2009. 117-139.

[4] E. S. Low, P. Ong, and K. C. Cheah, "Solving the optimal path planning of a mobile robot using improved q-learning," Robotics Auton. Syst., vol. 115, pp. 143–161, 2019.

[5] C. Watkins and P. Dayan, "Q-learning," Machine Learning, vol. 8, pp. 279–292, 1992.

[6] R. S. Sutton and A. G. Barto, "Reinforcement learning: An introduction," IEEE Transactions on Neural Networks, vol. 16, pp. 285–286, 2005.

[7] E. S. Low, P. Ong, C. Y. Low, and R. B. Omar, "Modified q-learning with distance metric and virtual target on path planning of mobile robot," Expert Syst. Appl., vol. 199, p. 117191, 2022.

[8] J. M. López-Guede, B. Fernández-Gauna, and M. Graña, "State-action value function modeled by elm in reinforcement learning for hose control problems," International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, vol. 21, pp. 99–116, 2013.

[9] Y. Hu, L. Yang, and Y. Lou, "Path Planning with Q-Learning," Journal of Physics: Conference Series, vol. 1948, p. 012038, June 2021. Publisher: IOP Publishing.

[10] A. Maoudj and A. Hentout, "Optimal path planning approach based on Q-learning algorithm for mobile robots," Applied Soft Computing, vol. 97, p. 106796, 2020.

[11] B. Hao, H. Du, and Z. Yan, "A path planning approach for unmanned surface vehicles based on dynamic and fast Q-learning," Ocean Engineering, vol. 270, p. 113632, 2023.

[12] Y. Li, "Deep reinforcement learning: An overview," ArXiv, vol. abs/1701.07274, 2017.

[13] P. Chen, J. Pei, W. Lu, and M. Li, "A deep reinforcement learning based method for real-time path planning and dynamic obstacle avoidance," Neurocomputing, vol. 497, pp. 64–75, 2022.

[14] M. L. Littman, "Markov games as a framework for multi-agent reinforcement learning," in International Conference on Machine Learning, 1994.