Intelligent Service Robotics (2020) 13:289–297
https://doi.org/10.1007/s11370-020-00313-y

ORIGINAL RESEARCH

Reinforcement learning path planning algorithm based on obstacle area expansion strategy

Haiyang Chen[1] · Yebiao Ji[1] · Longhui Niu[1]

Received: 23 August 2019 / Accepted: 13 January 2020 / Published online: 3 February 2020

Abstract

We improve the traditional Q(λ)-learning algorithm by adding the obstacle area expansion strategy. The new algorithm is named OAE-Q(λ)-learning and applied to the path planning in the complex environment. The contributions of OAE-Q(λ)-learning are as follows: (1) It expands the concave obstacle area in the environment to avoid repeated invalid actions when the agent falls into the obstacle area. (2) It removes the extended obstacle area, which reduces the learning state space and accelerates the convergence speed of the algorithm. Extensive experimental results validate the effectiveness and feasibility of OAE-Q(λ)-learning on the path planning in complex environments.

Keywords Path planning · Reinforcement learning · Eligibility trace · Concave obstacle · Obstacle area expansion

Abbreviation

OAE-Q(λ)-learning	The algorithm of Q(λ)-learning based on obstacle area expansion strategy

1 Introduction

The importance of robotic navigation technology has been increasingly emphasized due to the rise of unmanned driving [1]. Path planning is an important aspect of robot navigation technology [2–4]. It is defined as finding a collision-free path from the initial state to the target state according to some evaluation criteria in an environment with obstacles. Traditional path planning algorithms include artificial potential field method [5], genetic algorithm [6], ant colony optimization algorithm [7], etc. These methods need to model the environment in a certain space; therefore, there are some limitations in the complex and changeable environment. The reinforcement learning algorithm is a type of machine learning algorithm which has developed rapidly in recent years and is widely used in path planning [8, 9]. Its advantage is that it does not require accurate environmental models. Robot path planning methods using reinforcement learning algorithm include Q-learning [10], Sarsa [11], Q(λ)-learning [12], Sarsa (λ) [13], etc. However, the problem is that the more complex the environment is, the larger the learning state space will be, which will lead to a long learning time and slow convergence speed. In order to reduce the dimension of state space, many scholars have carried out a lot of research on this problem. Literature [14] proposed a method of abstracting state-action space, which reduces the dimension of the state space by utilizing the characteristics of the robot and the environment to generate a new state-action space. Literature [15] reduced the dimension of state space by the method of function approximation. It combined the reinforcement learning algorithm with the neural networks. Both methods of dimension reduction in the state space would cause errors because of the use of approximation algorithm.

This paper aims at the problem of huge state space of reinforcement learning in path planning under complex and unknown environment. Taking into account the concave obstacle areas in the environment, this paper introduces the idea named obstacle area expansion. Firstly, the concave obstacle areas are explored to avoid the agent falling into the concave obstacle area and causing a lot of invalid actions. Secondly, the concave obstacle areas are expanded and removed to reduce the state space dimension of the subsequent reinforcement learning. Combining the obstacle area expansion strategy with the Q(λ)-learning algorithm, we will propose an improved OAE-Q(λ)-learning path planning method. The simulation results show that this

✉ Yebiao Ji
jfj1968@163.com

[1] School of Electronic Information, Xi'an Polytechnic University, Xi'an 710000, People's Republic of China

method achieves path planning in an unknown environment of mobile robot and improves the convergence speed and learning efficiency of path planning.

2 Q(λ)-learning algorithm

An agent is rewarded by interacting with the environment, when using reinforcement learning algorithm for path planning in an unknown environment. The path with the highest reward is the optimal path. On the basis of the classical Q-learning, Q(λ)-learning combines the idea of instantaneous differential multi-step return [16], so the traditional single step update method is extended to multi-step update, which improves the update speed of the algorithm. The eligibility trace $E(s, a)$ records the motion path of the agent. Adding it to the update of Q value can reflect the update strength of Q value in different states. Therefore, the Q(λ)-learning algorithm combines the value function $Q(s, a)$ with the eligibility trace $E(s, a)$.

The traditional Q-learning updates its value function as follows:

$$Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma \max_{a'} Q(s', a') - Q(s, a)] \tag{1}$$

In the updating rule, $Q(s, a)$ is the Q value corresponding to action a in state s; $Q(s', a')$ is the Q value corresponding to action a' in state s'; r is the reward value when the environment changes from state s to state s'; α and γ are the learning rate and the discount factor, respectively.

After joining eligibility trace $E(s, a)$, the updating formula of Q(λ)-learning iteration is given by

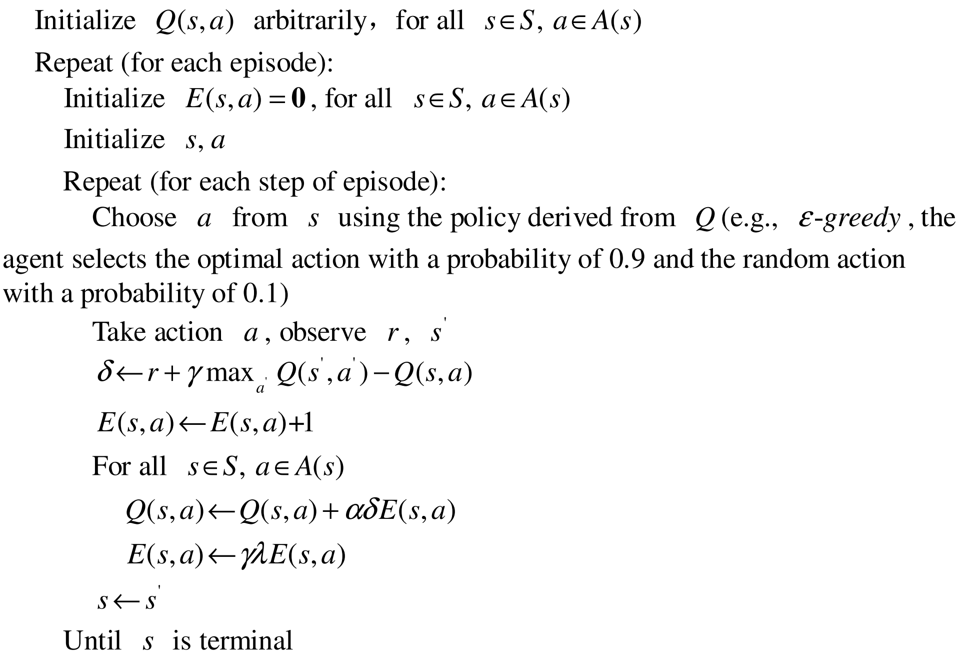
$$Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma \max_{a'} Q(s', a') - Q(s, a)]E(s, a) \tag{2}$$

In this rule, $E(s, a)$ is the eligibility trace. The initial value of the eligibility trace is 0. When the agent passes through a certain state, the eligibility trace value of that state needs to be increased by 1 at the moment. When performing subsequent actions, the eligibility trace $E(s, a)$ decreases according to the following rule:

$$E(s, a) \leftarrow \gamma\lambda E(s, a) \tag{3}$$

In this rule, λ is the eligibility trace attenuation factor. Each time an agent performs an action, the eligibility trace value at state s decreases once according to the above formula.

The flow of the algorithm:



```
Initialize $Q(s,a)$ arbitrarily, for all $s \in S$, $a \in A(s)$
Repeat (for each episode):
  Initialize $E(s,a) = \mathbf{0}$, for all $s \in S$, $a \in A(s)$
  Initialize $s, a$
  Repeat (for each step of episode):
    Choose $a$ from $s$ using the policy derived from $Q$ (e.g., $\varepsilon$-greedy, the agent selects the optimal action with a probability of 0.9 and the random action with a probability of 0.1)
    Take action $a$, observe $r$, $s'$
    $\delta \leftarrow r + \gamma \max_{a'} Q(s', a') - Q(s,a)$
    $E(s,a) \leftarrow E(s,a) + 1$
    For all $s \in S$, $a \in A(s)$
      $Q(s,a) \leftarrow Q(s,a) + \alpha\delta E(s,a)$
      $E(s,a) \leftarrow \gamma\lambda E(s,a)$
    $s \leftarrow s'$
  Until $s$ is terminal
```

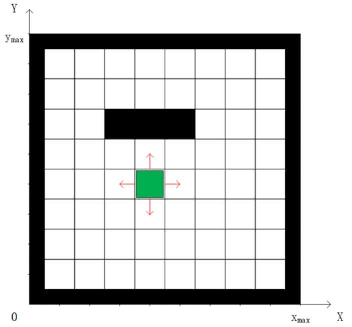


Fig. 1 Environmental model established by grid method: The green grid represents the location of the agent; the white grids represent the movable area; the black grids represent obstacles; the red arrows represent the directions of the agent's movement (color figure online)

3 Path planning based on OAE-Q(λ)-learning algorithm

This section introduces the idea of OAE-Q(λ)-learning algorithm. Firstly, we use the grid method to build the environmental model. Secondly, we introduce the specific implementation of the obstacle area expansion strategy. Finally, we combine the reinforcement learning algorithm with the obstacle area expansion strategy to illustrate the path planning method of OAE-Q(λ)-learning algorithm.

3.1 The idea of algorithm

The obstacle's state cannot be reached, when an agent uses reinforcement learning to plan a path in a complex and unknown environment. After exploring the obstacle for the first time, the subsequent learning need not plan this state anymore. Hence, every time the obstacle is explored, it can be removed from the environment model. According to this idea, the obstacle area expansion strategy was proposed: for the concave obstacle area in the environment, mark its internal state as immovable state and expand the obstacle area in the environment. The state space of reinforcement learning is reduced by removing the original obstacle state, and the extended immovable state in the environment model.

3.2 Environmental model

The environmental model was established by the grid method, as shown in Fig. 1. The black area around the figure is a wall with a thickness of 0.5. The outer boundaries of the upper, lower, left and right walls are represented by $y = y_{\max}$, $y = 0$, $x = 0$ and $x = x_{\max}$, respectively.

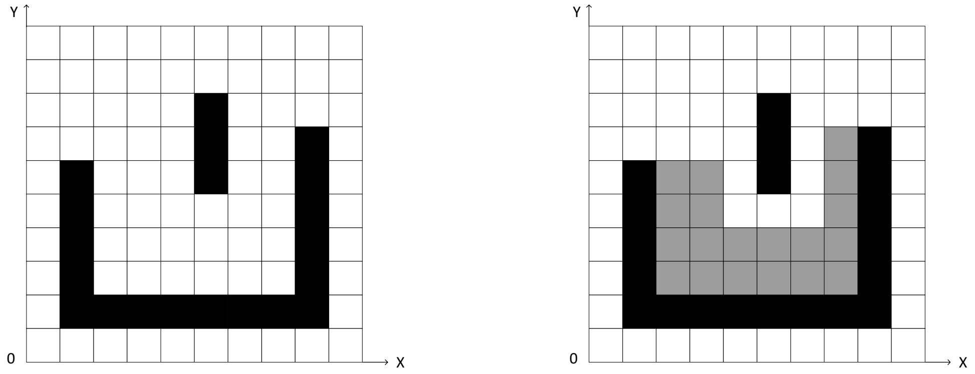
The size of the grid is a small square with the side length of 1. The size of the agent is one unit of the grid. Each state is an element of the environmental matrix (**EM**). Obstacles in the environment are randomly distributed. The white grid represents the movable state of the agent, and the element value of the corresponding **EM** is 0. The black grid represents the obstacles, and the corresponding element value of **EM** is 1. The environment matrix **EM** can be expressed as $\mathbf{EM} = \{\mathrm{em}_{ij} | \mathrm{em}_{ij} = 0 \,.\mathrm{or}\, 1,\ i, j \in N^{+}\}$.

The size of the agent is the size of a grid, and the position of the agent is represented by coordinates (x_0, y_0). The agent can take four possible actions: up, down, left and right. The discrete matrix $\mathbf{A} = [0, 1; 0, -1; -1, 0; 1, 0]$ is defined to represent the changes in the environment after the four actions are performed.

3.3 Obstacle area expansion strategy

The strategy of obstacle area expansion is proposed to deal with the concave obstacle area in environment. The expansion mode is transverse and vertical layer-by-layer expansion. Figure 2 is a comparison of the pre-expansion and post-expansion of the concave obstacle area. (To simplify the

Fig. 2 Contrast of obstacle area before and after expansion: The white grids represent the movable areas; the black grids represent obstacles; the gray grids represent expansible areas



(a) Obstacle area before expansion

(b) Obstacle area after expansion

environment model, the thickness of the wall is not shown here.) The black grids in the figure represent the concave obstacle area existing on the original environment, while the gray grids represent the expanded obstacle area. All the gray grids will be marked as *inf*, and these grids' values are changed from 0 to 1 in the environment matrix. Use the changed environment matrix to update the map environment before the learning of each episode. Therefore, the grids marked as *inf* in the previous episode will not be explored and planned by agents in the next learning.

The following definitions are given for the implementation of obstacle area expansion strategy.

Definition 1 Transversely, the relationship between the agent and the obstacle is $R_x(x,y)$. $R_x(x,y) > 0$ denotes that the transverse adjacent grid of agent at (*x*,*y*) has obstacles.

Definition 2 Vertically, the relationship between the agent and the obstacle is $R_y(x,y)$. $R_y(x,y) > 0$ denotes that the vertical adjacent grid of agent at (*x*,*y*) has obstacles.

Definition 3 Transversely and vertically, the relationship between the agent and the obstacle is $R_{xy}(x,y)$. $R_{xy}(x,y) > 0$ denotes that the transverse and vertical adjacent grids of agent at (*x*,*y*) have obstacles.

The implementation process of obstacle area expansion strategy is as follows:

Step 1 Confirm the existence of concave obstacle area. If $R_{xy}(x,y) > 0$, the current grid is marked as sus_1. The agent begins to search transversely and judges the values of $R_y(x,y)$ and $R_{xy}(x,y)$. When both $R_{xy}(x,y) > 0$ and $R_y(x,y) > 0$ are satisfied, the grid at (*x*,*y*) is marked as sus_2 and the opening direction of concave obstacle area is vertical. Then, the grids between sus_1 and sus_2 are marked as sus_d. If $R_y(x,y) < 0$ occurs, the agent goes back to sus_1 and begins to search transversely and judges the values of $R_x(x,y)$ and $R_{xy}(x,y)$. When both $R_{xy}(x,y) > 0$ and $R_y(x,y) > 0$ are satisfied, the grid at (*x*,*y*) is marked as sus_2 and the opening direction of concave obstacle area is transverse. Then, the grids between sus_1 and sus_2 are marked as sus_d. If $R_x(x,y) < 0$ occurs, the area is not a concave obstacle area and cannot be expanded. In this situation, the procedure goes to step 6.

Step 2 Confirm the depth of concave obstacle area. The agent searches vertically from sus_2 and judges the values of $R_x(x,y)$ and $R_{xy}(x,y)$. When $R_{xy}(x,y) < 0$, the grid in the previous moment of (*x*,*y*) is marked as sus_3. Or when $R_{xy}(x,y) > 0$, the grid at (*x*,*y*) is marked as sus_3. Then, the agent goes back to sus_1 and begins to search vertically. Also, it judges the values of $R_x(x,y)$ and $R_{xy}(x,y)$. When $R_x(x,y) < 0$, the grid in the previous moment of (*x*,*y*) is marked as sus_4. Or when $R_{xy}(x,y) > 0$, the grid at (*x*,*y*) is marked as sus_4. (If the opening direction of concave obstacle area is transverse, the agent searches transversely from sus_2 and sus_1. Also, it judges the values of $R_y(x,y)$ and $R_{xy}(x,y)$.)

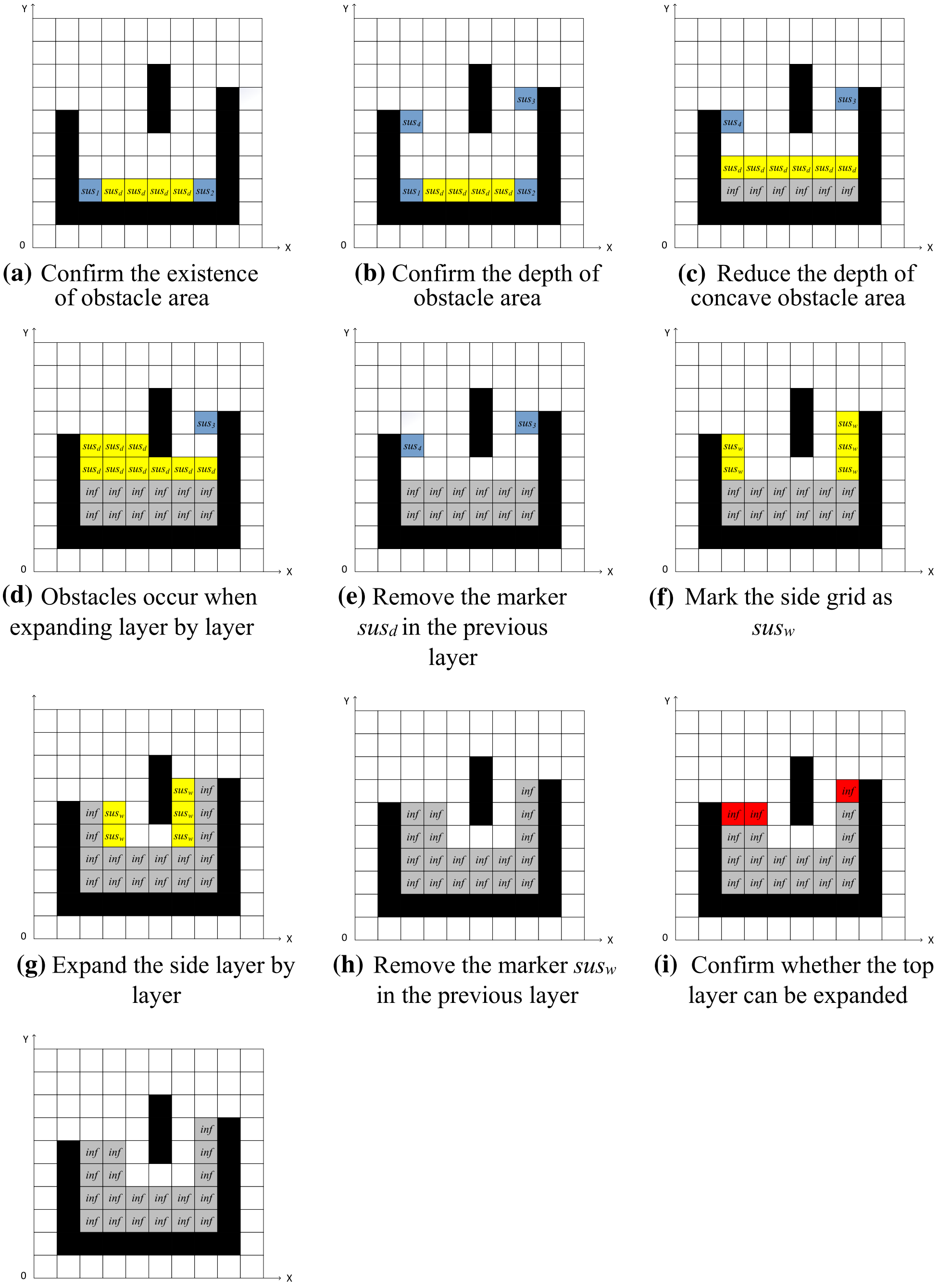
Step 3 Reduce the depth of concave obstacle area by expanding it from the bottom layer to the outside layer. The grids marked as sus_d are regarded as a layer unit and expand one layer in turn toward the opening of the concave obstacle area. If there is no obstacle in this layer, the grid marked as sus_d of the previous layer is converted to *inf,* and the grids of this layer are marked as sus_d. Then, the procedure repeats step 3. If there are obstacles in this layer, the marker of sus_d from the previous layer is canceled, and the procedure goes to step 4. If there is a starting point or an end point in the interior of this layer, the marker of sus_d in the previous layer is converted to marker *inf,* and the procedure goes to step 4.

Step 4 Reduce the width of concave obstacle area by expanding the side of concave area. The grids between sus_1 to sus_4 and sus_2 to sus_3 are marked as sus_w and regarded as a layer unit. The grids expand one layer in turn to the interior of concave obstacle area. If there is no obstacle in this layer, the grid marked as sus_w of the previous layer is converted to *inf,* and the grids of this layer are marked as sus_w. Then, the procedure repeats step 4. If there are obstacles in this layer, the marker of sus_w from the previous layer is canceled, and the procedure goes to step 5. If there is a starting point or an end point in the interior of this layer, the marker of sus_w in the previous layer is converted to marker *inf,* and the procedure goes to step 5.

Step 5 Judge whether there are obstacles at the top of the expanded obstacle area. It judges the value of $R_y(x,y)$ which is marked as *inf* in the top layer. If the value of $R_y(x,y)$ on a grid satisfies $R_y(x,y) > 0$, it cancels the marker of three adjacent *inf* grids centered on (*x*,*y*). (If the opening direction of concave obstacle area is transverse, it judges the values of $R_x(x,y)$).

Step 6 Confirm the grids which can be expanded. The grids marked as *inf* on the environment are expandable grids.

Figure 3 shows the process of the expansion strategy of the obstacle area. Firstly, as shown in Fig. 3a, we mark sus_1 and start searching sus_2 transversely or vertically to confirm the existence of concave obstacle area. Secondly, as shown in Fig. 3b, we start searching for the height of the obstacle area to find sus_3 and sus_4. Next, as shown in Fig. 3c, we begin to expand from the bottom layer to the outside layer by layer. The expansion layer could be expanded to sus_3 or sus_4, if there was no obstacle, starting point or end point inside the concave obstacle area. Otherwise, as shown in Fig. 3d, e, the expansion layer could only be expanded to the previous layer of the obstacle layer. Then, we begin to expand layer by layer on both sides, as shown in Fig. 3f–h, to reduce the width of obstacle area. Finally, as shown in Fig. 3i, j, the expandable area has been determined by determining whether there are obstacles outside the top of the obstacle area.



(a) Confirm the existence of obstacle area

(b) Confirm the depth of obstacle area

(c) Reduce the depth of concave obstacle area

(d) Obstacles occur when expanding layer by layer

(e) Remove the marker sus_d in the previous layer

(f) Mark the side grid as sus_w

(g) Expand the side layer by layer

(h) Remove the marker sus_w in the previous layer

(i) Confirm whether the top layer can be expanded

(j) Confirm the expansion area

Fig. 3 The process of obstacle area expansion: The white grids represent the movable areas; the black grids represent obstacles; the yellow grids represent suspected expandable grids in the concave obstacle area; the blue grids represent the four edge points in the concave obstacle area; the gray grids represent expandable grids of the concave obstacle area; the red grids represent the top layer of the concave obstacle area (color figure online)

3.4 Path planning strategy

According to the principle of reinforcement learning and the idea of expanding obstacle areas, path planning subject to unknown environment is carried out. The planning steps are as follows:

Step 1 Initialize the data and start looping from the starting point to the end point.

Step 2 Update the environment according to the marked *inf* state.

Step 3 The agent is ready to start from the starting point.

Step 4 Obtain the Q values of the four adjacent states of the current state. Find the action corresponding to the maximum Q value through the greedy strategy and record the original state.

Step 5 Obtain the next states and the reward value. There are three cases of reward value. Case 1: The reward value

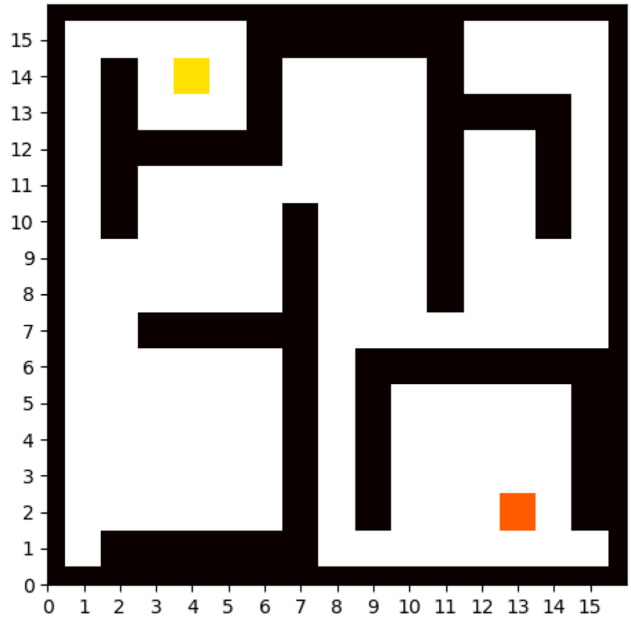


Fig. 4 Experimental environment: The white grids represent the movable areas; the black grids represent obstacles; the orange grid represents the starting point; the yellow grid represents the end point (color figure online)

is -1, when there is an obstacle in the next state. Mark the obstacle state as *inf* and judge the value of $R_{xy}(x, y)$. If $R_{xy}(x, y) > 0$, the agent implements the obstacle area expansion strategy. Otherwise, the agent continues to move in the environment. Case 2: The reward value is 0 and the agent enters the next state, when there is no obstacle in the next state. Case 3: The reward value is 1, when the next state is the end point. At last, the Q value is updated according to Eq. (2).

Step 6 Enter step 7 if the current position is the end point; otherwise, enter step 4.

Step 7 Enter step 8 if the current number of learning episodes meets the set number of learning episodes; otherwise, enter step 2.

Step 8 End the path-finding process.

The agent first judged its relationship with obstacle when it encountered obstacle. If only one side encountered obstacle, that is $R_x(x, y) > 0$ or $R_y(x, y) > 0$, it would mark the obstacle state as *inf* and then continue to search for the end point. If both sides encountered obstacles, that is $R_{xy}(x, y) > 0$, it would interrupt the search and begin to expand the obstacle area. These expansible obstacle states were marked as *inf*, and the agent did not explore these states again in the learning of this episode. The environment was updated before the next episode, and all states marked as *inf* were no longer iterated for Q value and planned path.

Table 1 Effect table of learning rate

Learning rate (α)	Number of convergence episodes	Program execution time	Average path length
0.1	420	169	46.36
0.3	238	89	42.28
0.5	186	68	40.92
0.7	115	55	40.81
0.8	96	49	40.92
0.9	67	47	40.71
0.95	76	48	41.44

Table 2 Effect table of discount factor

Discount factor (γ)	Number of convergence episodes	Program execution time	Average path length
0.5	68	51	41.63
0.7	75	49	41.44
0.8	67	47	40.71
0.9	75	48	40.96
0.95	85	48	40.91

Table 3 Effect table of exploration factor

Exploration factor (ε)	Number of convergence episodes	Program execution time	Average path length
0.5	–	73	–
0.7	118	56	50.05
0.8	78	52	44.34
0.9	67	47	40.71
0.95	70	49	41.02

Table 4 Effect table of eligibility trace attenuation factor

Eligibility trace attenuation factor (λ)	Number of convergence episodes	Program execution time	Average path length
0.7	77	49	41.42
0.8	74	48	42.44
0.9	67	47	40.71

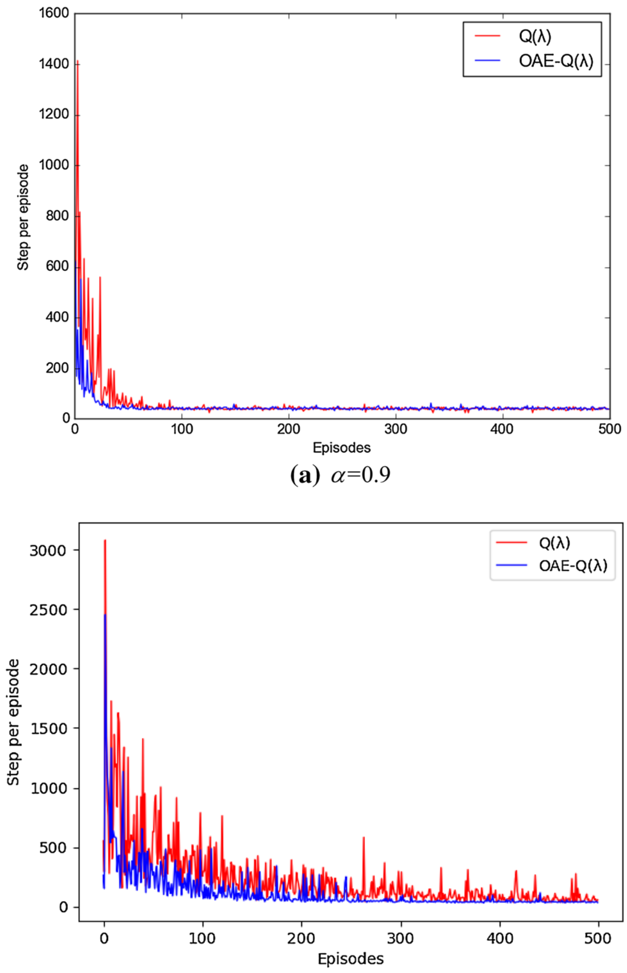
4 Experimental

The experimental environment is a grid world size of 15 * 15, so set $x_{\max} = 16$, $y_{\max} = 16$. As shown in Fig. 4, the orange grid is the starting point of the agent and the yellow is the end point. The black area is the obstacle.

Obstacles and end point in the environment are static. The environment (that is, the location of obstacles, boundaries and targets) is unknown for the agent. It has four actions to choose from: up, down, left and right.

The effects of learning rate, exploration factor, discount factor and eligibility trace attenuation factor on the performance of the algorithm are compared by simulations. We test the algorithm in three aspects: number of convergence episodes, program execution time and average path length. Tables 1, 2, 3 and 4 show the test results.

From the above table data, it can be seen that the learning rate and the exploration factor have a greater impact on the performance of the reinforcement learning algorithm, while the discount factor and eligibility trace attenuation factor have a smaller impact on the performance of the algorithm. In this experiment, we choose $\gamma = 0.8$, $\varepsilon = 0.9$ and $\lambda = 0.9$. The learning rate A was chosen as 0.1 and 0.9, respectively. The number of learning episode is 500, and the reward function is designed as follows:



(a) α=0.9

(b) α=0.1

Fig. 5 Comparison of algorithm convergence:The red line represents the convergence speed of Q(λ)-learning algorithm, and the blue line represents the convergence speed of OAE-Q(λ)-learning algorithm (color figure online)

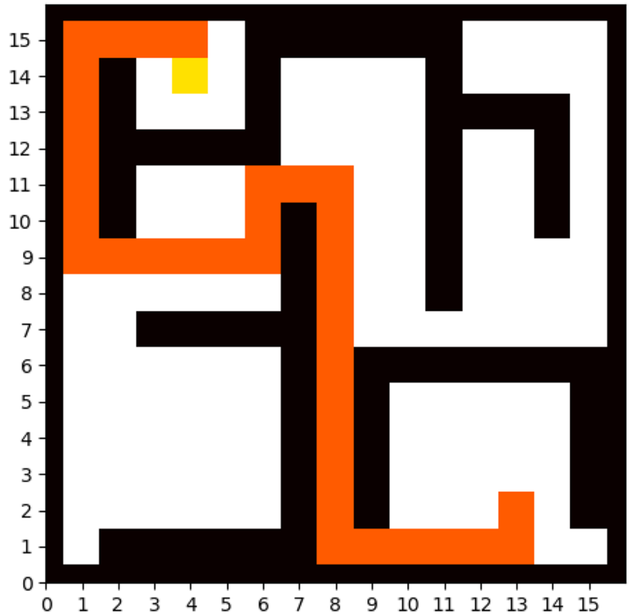


Fig. 6 Path planning results: The white grids represent the movable areas; the black grids represent obstacles; the yellow grid represents the end point; the orange grids represent the path planned by the agent (color figure online)

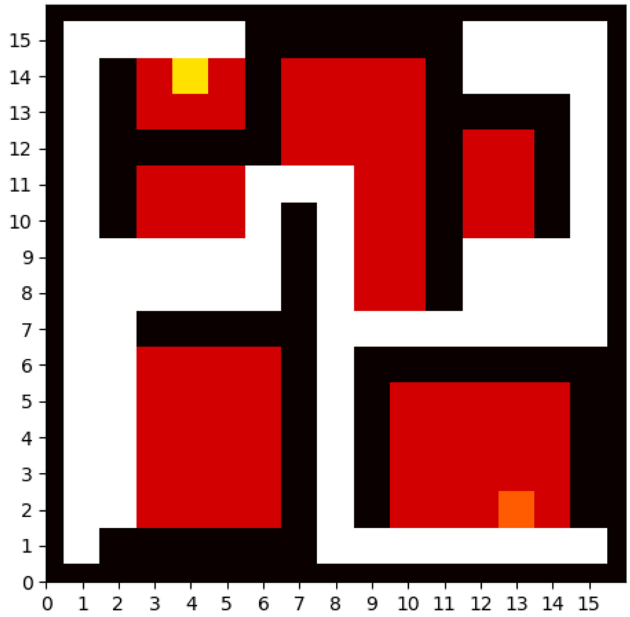


Fig. 7 Map environment after obstacle area expansion: The white grids represent the movable areas; the black grids represent obstacles; the orange grid represents the starting point; the yellow grid represents the end point; the red grids represent the expanded area (color figure online)

$$r = \begin{cases} 1, & \text{Reach the target} \\ -1, & \text{Encountering obstacles or boundarier} \\ 0, & \text{Other circumstances} \end{cases} \tag{4}$$

5 Results and discussion

The convergence rates of Q(λ) and OAE-Q(λ)-learning algorithms are compared under the above-mentioned experimental environment. Figure 5 is the comparison of the convergence rates of the two algorithms, Fig. 5 a is the convergence curve when the learning rate is 0.9, and Fig. 5 b is the convergence curve when the learning rate is 0.1. Figure 6 is the optimal path planned by OAE-Q(λ)-learning algorithm, and Fig. 7 is the environment after the expansion of the obstacle area.

As can be seen from Fig. 5, the OAE-Q(λ)-learning algorithm with the obstacle area expansion strategy has a faster convergence rate than the traditional Q(λ)-learning algorithm. The agent has more exploration steps in the early stage, and the convergence curve will still be disturbed in the late learning stage when using the Q(λ)-learning algorithm in the path planning. The OAE-Q(λ)-learning algorithm basically converges after the 40th episode and the 250th episode, and the path steps are stable between the 35th and the 38th steps. The agent has little knowledge of the environment in the early stage of learning; therefore, the traditional algorithm would make the agent spend a lot of steps to escape from the concave obstacle area after they fell into it. Moreover, it was possible for agents to fall into it in the subsequent learning. The addition of obstacle area expansion strategy not only provides a way for the agent to get out of the concave obstacle area, but also avoids the situation that the agent enters the obstacle area for the second time. Figure 6 shows the path planned by the agent in the 500th episode. Figure 7 shows the map environment after the end of learning, and the red area is the expanded concave obstacle area. It can be seen that the original enhanced learning state space is significantly reduced after removing the obstacle area, which ensures the improvement of learning efficiency.

6 Conclusion

This paper presents a new method for path planning in complex environments. We expand the concave obstacle area in the environment in order to solve the huge problem of reinforcement learning state space. This method not only avoids the predicament that the agent falls into the concave obstacle area and wanders repeatedly while exploring the environment, but also reduces the dimension of the state space in reinforcement learning after removing the expanded obstacle area. Experiment shows that the optimal path is planned with less learning times than the traditional method. The proposed algorithm has great advantages for terrain with more concave obstacle areas.

Acknowledgements The research of this paper is supported by the National Natural Science Foundation of China.

Funding The authors are partially supported by NSFC (61573285).

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Galceran E, Cunningham AG, Eustice RM et al (2017) Multipolicy decision-making for autonomous driving via change point-based behavior prediction: theory and experiment. Auton Robots 41(6):1367–1382
2. Li Y, Li D, Maple C et al (2013) K-order surrounding roadmaps path planner for robot path planning. J Intell Robot Syst 75(3–4):493–516
3. Chen Y, Cheng L, Wu H et al (2015) Knowledge-driven path planning for mobile robots: relative state tree. Soft Comput 19(3):763–773
4. Hebecker T, Buchholz R, Ortmeier F (2015) Model-based local path planning for UAVs. J Intell Rob Syst 78(1):127–142
5. Chen YB, Luo GC, Mei YS et al (2016) UAV path planning using artificial potential field method updated by optimal control theory. Int J Syst Sci 47(6):14
6. Lee D, Shim DH (2018) A mini-drone development, genetic vector field-based multi-agent path planning, and flight tests. Int J Aeronaut Space Sci 19(3):785–797
7. Yue L, Chen H (2019) Unmanned vehicle path planning using a novel ant colony algorithm. EURASIP J Wirel Commun Netw 2019(1):136
8. Zhang B, Mao Z, Liu W et al (2015) Geometric reinforcement learning for path planning of UAVs. J Intell Rob Syst 77(2):391–409
9. Jiang J, Xin J (2019) Path planning of a mobile robot in a free-space environment using Q -learning. Progr Artif Intell 8(1):133–142
10. Haghzad Klidbary S, Bagheri Shouraki S, Sheikhpour Kourabbaslou S (2017) Path planning of modular robots on various terrains using Q-learning versus optimization algorithms[J]. Intel Serv Robot 10(2):121–136
11. Pakizeh E, Pedram MM, Palhang M (2015) Multi-criteria expertness based cooperative method for SARSA and eligibility trace algorithms. Appl Intell 43(3):487–498
12. Kim B, Pineau J (2016) Socially adaptive path planning in human environments using inverse reinforcement learning. Int J Social Robot 8(1):51–66
13. Martinez-Gil F, Lozano M, Fernández F (2014) MARL-Ped: a multi-agent reinforcement learning based framework to simulate pedestrian groups. Simul Model Pract Theory 47:259–275
14. Ito K, Takeuchi Y (2016) Reinforcement learning in dynamic environment: abstraction of state-action space utilizing properties of the robot body and environment]. Artif Life Robot 21(1):11–17

15. Yasini S, Naghibi Sitani MB, Kirampor A (2016) Reinforcement learning and neural networks for multi-agent nonzero-sum games of nonlinear constrained-input systems. Int J Mach Learn Cybernet 7(6):967–980
16. Yu T, Wang HZ, Zhou B et al (2015) Multi-agent correlated equilibrium Q(λ) learning for coordinated smart generation control of interconnected power grids. IEEE Trans Power Syst 30(4):1669–1679