Intelligent Service Robotics (2024) 17:915–929
https://doi.org/10.1007/s11370-024-00544-3

ORIGINAL RESEARCH PAPER

ETQ-learning: an improved Q-learning algorithm for path planning

Huanwei Wang[1] · Jing Jing[1] · Qianlv Wang[1] · Hongqi He[1] · Xuyan Qi[1] · Rui Lou[1]

Received: 27 December 2023 / Accepted: 12 May 2024 / Published online: 26 June 2024

Abstract
Path planning algorithm has always been the core of intelligent robot research; a good path planning algorithm can significantly enhance the efficiency of robots in executing tasks. As the application scenarios for intelligent robots continue to diversify, their adaptability to the environment has become a key focus in current path planning algorithm research. As one of the classic reinforcement learning algorithms, Q-learning (QL) algorithm has its inherent advantages in adapting to the environment, but it also faces various challenges and shortcomings. These issues are primarily centered around suboptimal path planning, slow convergence speed, weak generalization capability and poor obstacle avoidance performance. In order to solve these issues in the QL algorithm, we have carried out the following work. (1) We redesign the reward mechanism of QL algorithm. The traditional Q-learning algorithm's reward mechanism is simple to implement but lacks directionality. We propose a combined reward mechanism of "static assignment + dynamic adjustment." This mechanism can address the issue of random path selection and ultimately lead to optimal path planning. (2) We redesign the greedy strategy of QL algorithm. In the traditional Q-learning algorithm, the greedy factor in the strategy is either randomly generated or set manually, which limits its applicability to some extent. It is difficult to effectively applied to different physical environments and scenarios, which is the fundamental reason for the poor generalization capability of the algorithm. We propose a dynamic adjustment of the greedy factor, known as the $\varepsilon - acc - increasing$ greedy strategy, which significantly improves the efficiency of Q-learning algorithm and enhances its generalization capability so that the algorithm has a wider range of application scenarios. (3) We introduce a concept to enhance the algorithm's obstacle avoidance performance. We design the expansion distance, which pre-sets a "collision buffer" between the obstacle and agent to enhance the algorithm's obstacle avoidance performance.

Keywords Q-learning · Path planning · Reinforcement learning · Reward mechanism · Greedy strategy

1 Introduction

In recent years, the mobile robotics industry has witnessed remarkable development, with an increasingly diverse range of applications. These applications include logistics and warehousing, healthcare, agriculture, construction, smart manufacturing and military fields. Additionally, the unmanned aerial vehicle (UAV) industry has flourished, encompassing areas such as surveillance, photography, agriculture, firefighting and, more, providing unique solutions for various industries.

Path planning plays a crucial role in a wide range of applications [1]. It is a core component of intelligent decision-making for mobile robots and UAV systems. Path planning algorithm allows robots and UAVs to plan reasonable paths in dynamic environments to accomplish tasks and avoid collisions.

There are various path planning algorithms, each suitable for different applications and scenarios. Based on their fundamentals and application fields, path planning algorithms can be categorized as follows: (1) Based on graph theory and search algorithms, including Dijkstra's algorithm and A* algorithm [2, 3]. They are suitable for shortest path planning in static environments. (2) Sampling-based algorithms, such as Rapidly exploring Random Tree (RRT) and Probabilistic Roadmap (PRM), which are suitable for high-dimensional environments and complex terrain. (3) Reinforcement learning algorithms, such as Q-learning (QL), which are designed for path planning in dynamic environ-

Huanwei Wang and Jing Jing have equally contributed to this work.

✉ Rui Lou
lourui@nudt.edu.cn

[1] PLA Information Engineering University, Science Ave 62, ZhengZhou 450001, Henan, China

ments. These algorithms enable intelligent agents to learn optimal path strategies through interaction with the environment, making them well suited for situations with high uncertainty and dynamics. Compared to other algorithms, QL algorithm can autonomously make decisions and learn online for path planning in uncertain and dynamic environments. It is capable of learning optimal path strategies without prior knowledge and has significant advantages when used to solve complex path planning problems.

In unfamiliar and complex environments, although QL algorithm has many advantages, it still has some challenges in complex environments, such as easy to fall into local optimal solutions, slow convergence speed and limited generalization capability. Researchers have already recognized these challenges and have attempted to address them [4–7], but their results have been unsatisfactory. To address these issues, we propose a combined reward mechanism of "static assignment + dynamic adjustment" to achieve the goal of planning the optimal path. We propose a dynamic adjustment strategy for the greedy factor, known as $\varepsilon - acc - increasing$ greedy strategy, to improve the algorithm's convergence speed and generalization capability. In addition, we design an expansion distance to improve the obstacle avoidance performance of the algorithm.

The primary contributions of this paper are summarized as follows.

(1) Improvement in the reward mechanism of QL algorithm. To the best of our knowledge, this is the first study to introduce a combination of "static assignment + dynamic adjustment" reward mechanism. This method is able to plan the optimal path to overcome the issues of the local optimal and the suboptimal paths.
(2) Improve the greedy strategy of QL algorithm. In traditional QL algorithm, the greedy factor is randomly generated and lacks universality. We first propose a novel $\varepsilon - acc - increasing$ greedy strategy with dynamic adjusted greedy factor, which effectively improves the generalization ability and efficiency of QL algorithm.
(3) Enhanced obstacle avoidance capability. We design an expansion distance, which presets a "collision buffer" between the obstacle and agent to enhance the algorithm's obstacle avoidance performance.

The remaining part of the paper proceeds as follows.

Section 2 discusses the related works. Section 3 covers the preliminary reward mechanism and greedy strategy of QL algorithm. In Sect. 4, we elaborate on the innovative design of the algorithm. The experimental results and discussion are presented in Sect. 5. Finally, conclusions are drawn in Sect. 6.

2 Related work

QL algorithm is a classical algorithm in reinforcement learning for learning how to make optimal decisions to maximize cumulative rewards in an unknown environment. Improving algorithm, path planning applications and reinforcement learning optimization are several widespread research directions for QL algorithms. Researchers seek to enhance QL algorithm to improve their performance and convergence speed in improving algorithm.

2.1 Improving greedy strategy

Improving QL algorithm is an active research field that involves exploring different learning rate strategies, exploration strategy (e.g., *e*-greedy strategy) and initialization methods. Variants of the algorithm, such as Dueling Q-learning [8] and Double Q-learning [9], have also been developed to further enhance performance.

Researchers improve the exploration strategy to explore the state space more efficiently, using uncertainty information, adaptive exploration strategy and information theory-based methods. Traditional greedy strategies usually adopt static and subjectively set greedy factors, which cannot effectively deal with the "exploration–exploitation" balance in different environments and can easily lead to a local optimal solution. In reference [10], a dynamic search factor is introduced to balance randomness and purposefulness by adjusting its size, and experimental results show that this effectively solves the problem of over-localized search. Literature [11] proposed an ε-DBE greedy strategy, in which the value of the exploration greedy factor ε decreases with the number of iteration steps, i.e., in the exploration phase, the greedy factor decreases as the number of iterations increases, and ultimately tends to be fully utilized, in order to avoid over-exploration and over-utilization. In contrast, literature [12] proposes a greedy strategy that combines the optimal path and the maximum reward value, which is ultimately used to plan out the fixed optimal path in unmanned ships.

Traditional ε-greedy strategies struggle to balance exploration and exploitation effectively. Noisy networks involve introducing noise into the weights of neural networks to enhance exploration strategy, to make exploration more random and effective, to make it more suitable for high-dimensional and continuous action space problems [13]. While noisy networks improve exploration, they may introduce a level of randomness that could lead to overly exploratory behavior. Balancing exploration and exploitation remains a challenge. Deep Deterministic Policy Gradients (DDPG) algorithm [14, 15] is designed for continuous action spaces. It improves exploration by using a deterministic policy and adding noise to explore the state space. In continuous action spaces, traditional stochastic policies may not be as

effective. DDPG uses a deterministic policy and adds noise to explore the state space more effectively, addressing the continuous action exploration challenge. DDPG provides better exploration in continuous action spaces but may still require careful tuning of noise parameters. Additionally, it could face issues with convergence and exploration in complex environments. Policy gradient methods can often lead to unstable exploration strategy updates. Trust Region Policy Optimization (TRPO) [16] uses trust region techniques to limit policy changes, ensuring robust and controlled improvements in exploration performance. TRPO achieves better stability and controlled exploration, but it can be computationally expensive and may require fine-tuning of hyperparameters. It doesn't fully address exploration in large and complex state spaces.

These researches represent various approaches to enhance exploration strategies in QL, addressing issues such as instability, over-exploration or under-exploration in traditional exploration policies. These methods provide more effective and stable exploration strategies, thereby improving the performance of reinforcement learning algorithms.

2.2 Improving reward mechanism

The reward mechanism of the traditional QL algorithm is relatively straightforward and lacks clear guidance, which leads to the lack of smoothness in the planning path and the frequent occurrence of "jump points" and "return points," which makes it impossible to obtain the optimal path. To address this issue, many researchers optimize the reward mechanism to provide well-defined guidance, resulting in smoother optimal path planning. In literature [4], a distance-based reward mechanism is proposed, which adjusts the reward value according to the distance to the goal in the next step, and achieves significant improvements in UAV applications. Literature [17] proposes a reward mechanism based on the distance between the state node and the goal node, which helps to avoid blind searching and over-exploration by intelligent robots. On the other hand, literature [12] proposed a comprehensive incentive function integrating distance, safety and comfort, which was successfully applied to unmanned vessels with distinct guidance and obstacle avoidance capabilities.

Conventional strategy gradient methods may exhibit excessive updates, leading to instability and excessive exploration. Proximal Policy Optimization (PPO) [18] improves exploration strategies by limiting the range of strategy changes, enhancing stability and sample efficiency. PPO improves the stability of the strategy by applying the "clip surrogate objective" technique to the reward mechanism. However, PPO mitigates some exploration issues, but it doesn't fully address the challenge of handling high-dimensional state and action spaces. It also relies on finite difference approximations, which can be computationally expensive. Marcin Andrychowicz, et al. [19] present the "Hindsight Experience Replay" (HER) technique for improving QL by learning from failures and enhancing exploration. Tuomas Haarnoja et al. [20] introduce Soft Actor-Critic (SAC), a deep reinforcement learning algorithm that incorporates a maximum entropy reward mechanism to enhance exploration.

These studies represent innovative approaches on reward mechanism optimization for QL algorithm in recent years. They address issues in reward mechanism design, including sparse rewards, human feedback and self-expressive feedback. However, there are still some shortcomings such as hyperparameter tuning, computational resource requirements and dependence on external information. The role of reinforcement learning in the QL algorithm is primarily to assist an agent in learning how to make decisions in an environment to maximize cumulative rewards. It enables the agent to learn autonomously through interactions with the environment without the need for explicit supervision. In QL, reinforcement learning is used to learn the optimal action selection strategy, and it improves the quality of decision-making by continuously exploring the state space to attain higher rewards. Kumar et al. propose a reinforcement learning approach for self-expressive feedback that addresses issues in reward mechanism design, allows intelligences to learn task goals autonomously and improves generalizability [21, 22]. Allowing agent to receive additional reward signals from human feedback helps solve the sparse reward problem.

However, there are still some existing issues in current research. These methods require large amounts of training data and computational resources [14, 15, 23], and the algorithms are inefficient. Algorithms usually have difficulty generalizing the strategies learned in the training environment to different environments or tasks. This can lead to performance degradation in new environments.

2.3 Path planning

QL algorithm is employed to tackle path planning and obstacle avoidance for robots in dynamic environments. By learning Q-values, the robot can select optimal paths to navigate around obstacles and in dynamic settings. However, the method's applicability is constrained by the dimensionality of the state space, particularly in large environments, as it requires a large Q-table. Additionally, QL may demand substantial training data and may not be effective in rapidly adapting to changing environments [5]. Z Wang, et al. employ QL for path planning of unmanned aerial vehicles (UAVs). It addresses path planning and navigation for UAVs in different environments, aiding UAVs in finding optimal paths to reach their goals. This approach may require a considerable amount of training data and prior information about the

environment, which could limit its effectiveness in complex or partially known environments [24]. In addition, QL algorithms have a wide range of applications in path planning for UAVs [17, 25, 26].

Using the QL algorithm for path planning in robot navigation, S Li, et al. [27] introduce an adaptive path planning method that allows robots to dynamically adjust their paths based on changing environmental conditions, enhancing adaptability in navigation. The adaptability of this method is limited, especially in highly dynamic environments, and it may face challenges in such scenarios. QL is used to learn adaptive strategies for path planning and collision avoidance problems of robots in unknown environments [28]. However, this approach may have limited robustness in the face of highly dynamic and unknown environments, and further improvements are needed to enhance performance.

3 Preliminaries

In this section, we introduce the reward mechanism and ε-greed strategy of QL algorithm.

3.1 Reward mechanism

The reward mechanism is an important evaluating indicator used to evaluate the action of agents. A larger reward value indicates a better action selected by the agent. Otherwise, reward mechanism is one of the factors that determine the path planning results and algorithm efficiency. Reward mechanism of QL algorithm is shown in Eq. (1).

$$Q = \begin{cases} C1, & \text{state} = \text{goal} \\ -C1, & \text{state} = \text{obstacle} \\ 0, & \text{other} \end{cases} \tag{1}$$

In Eq. (1), Q represents the reward function value, state represents the state of the current node, goal represents the path destination, obstacle represents the obstacle node, and other represents other nodes except the destination and obstacle.$C1$ represents a positive integer. The reward value is set to $-C1$ when the agent encounters an obstacle. Reward value is set to $C1$ when the agent reaches the goal node, and reward values of other nodes are set to 0.

3.2 Greedy strategy

The principle of the ε-greedy strategy is as follows: QL algorithm introduces a random exploration factor ε during the initialization process, where $0 < \varepsilon \leq 1$. During the exploration phase, when selecting actions, the algorithm generates a random number between 0 and 1 and makes a decision. If the generated random number is Greater than ε, it will randomly choose an action. If this random number is less than ε, it will select the action that currently gets the maximum reward. After multiple iterations, the value of ε is gradually set to 1 and enters the full exploitation phase, where it consistently selects the action that can obtain the maximum reward until the iteration is completed. ε -greedy strategy is denoted by Eq. (2):

$$\varepsilon = \begin{cases} 1, & ep \geq pre_{ep} \\ r(0,1), & ep < pre_{ep} \end{cases} \tag{2}$$

where ε denotes the exploration factor, ep denotes the number of iterations, $r(0, 1)$ denotes a random number of (0,1) and pre_{ep} denotes the pre-set iteration threshold.

From the above introduction, it is evident that the ε-greedy strategy of the traditional QL algorithm has two issues.

1. ε is generated completely randomly in the initialization phase, which is not necessarily the best choice for the algorithm. If the initial value is set too large, it could lead to the loss of the exploration phase's significance, resulting in the Q-table not receiving effective updates, thus affecting the learning effect.
2. The preset value of pre_{ep} is usually based on experience or subjective preferences, which limits its applicability to some extent. It is challenging to effectively apply it in complex physical environments, which is the fundamental reason for the poor generalization ability of the QL algorithm. In real-world path planning tasks, the physical environment can vary significantly, leading to less than ideal performance when the algorithm is applied in different environments, indicating a lack of generalization capability.

4 Innovative design

4.1 Reward mechanism design

To overcome the shortcoming of QL algorithm reward mechanism, we redesign the reward mechanism and propose an innovative two-step assignment strategy of "static assignment + dynamic adjustment" reward mechanism.

4.1.1 Static assignment of reward mechanism

The basic principle of "static assignment" is as follows: the reward mechanism of QL algorithm is modified to effectively assign values only to obstacles and goal node. Inspired by the A * algorithm, it employs the distance relationship between the current node of the path and the goal node and devel-

ops the static assignment process according to the distance relationship. The implementation process is as follows:

1. Calculate the distance between the start node and goal node. Assuming the coordinate of start node are (x_0, y_0) and the coordinate of goal node are (x_1, y_1). The Euclidean distance is used to get the distance between the start node and goal node as D, as shown in Eq. (3).

$$D = \sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2} \tag{3}$$

2. Calculate the distance between each node and goal node. Assume the coordinates of the current node are (x_1, y_1); still using the Euclidean distance, the distance between the current node and goal node is D_i, as shown in Eq. (4).

$$D = \sqrt{(x_1 - x_i)^2 + (y_1 - y_i)^2} \tag{4}$$

3. Calculate the distance from each node to the diagonal. Assume the coordinates of the current node are (x_i, y_i). We can obtain the distance from the current node to the diagonal, denoted as d_i, as shown in Eq. (5).

$$d_i = |x_i - y_i| / \sqrt{2} \tag{5}$$

4. Calculate reward value of the node. D_i and d_i are employed in the reward mechanism. The design principle of reward mechanism is that the distance is far, but the reward value is small. According to the distance relationship, the reward mechanism we design is shown in Eq. (6) where q_i is the reward value and n is the map size.

$$q_i = -D_i - d_i + \frac{3\sqrt{2}}{4}n \tag{6}$$

5. Based on the above steps, the reward mechanism in the "static assignment" stage is shown in (7):

$$Q = \begin{cases} C1, & C1 > 0, state = goalnode \\ -C1, & state = obstacle \\ q_i, & other \end{cases} \tag{7}$$

The above steps are the design process of the "static assignment." The obstacle is still assigned $-C1$, and the goal node is still assigned $C1$ in the reward mechanism. Other nodes are no longer assigned to 0 but according to Eq. (7). In this way, each node has a relatively reasonable initial value when the algorithm is learning. After the design of static assignment, the learning process of QL algorithm is no longer completely random but can move along the direction with the maximum reward value and the direction closest to the goal node at the same time, so it has preliminary guidance.

4.1.2 Dynamic adjustment of reward mechanism

After completing the "static assignment," the reward mechanism starts to "Dynamic adjustment" the nodes in each iteration learning. The basic principle of "dynamic adjustment" is as follows: QL algorithm assigns values to nodes again in each iteration according to the assignment strategy, and the process is repeated. During the learning process of the algorithm, each node is dynamically assigned a reward value. And each assignment adjusts the reward value according to state and action. The dynamic assignment strategy is as follows.

1. If the next action of the agent makes it closer to the goal node without collision, the reward value Q_i of the original state will be adjusted, and the strategy is shown in (8):

$$Q_i = q_i + C3 \times \frac{D_i}{D} \tag{8}$$

2. If the next action of the agent moves it further away from the goal node without collision, the reward value Q_i of the original state will be adjusted, and the strategy is shown in (9):

$$Q_i = q_i - C3 \times \frac{D_i}{D} \tag{9}$$

3. Based on the above steps, the reward mechanism in the "Dynamic adjustment" stage is shown in (10):

$$Q = \begin{cases} C1, & state = goalnode \\ -C2, & state = obstacle \\ q_i + C3 \times \frac{D_i}{D}, & D_i < D_{i-1} \\ q_i - C3 \times \frac{D_i}{D}, & D_i > D_{i-1} \end{cases} \tag{10}$$

A comparison of the pre-improved reward mechanism and the improved reward mechanism is shown in Fig. 1.

The central idea of "dynamic adjustment" is to assign values according to the next action of the agent, that is, according to the distance between the node to be visited and the goal node. If it is close to the goal node, the reward value is assigned a positive value based on the "static assignment" or the previous assignment, and the reward value increases as the distance decreases. Conversely, if the agent is moving away from the goal, a negative reward value is assigned, and the reward value decreases as the distance increases.

We redesigned the reward mechanism in the proposed method. Through the dual design of "static + dynamic" reward mechanism and the employment of distance factor in the reward mechanism, the reward mechanism is more scientific and efficient. The advantages of this reward mechanism are mainly reflected in three aspects:

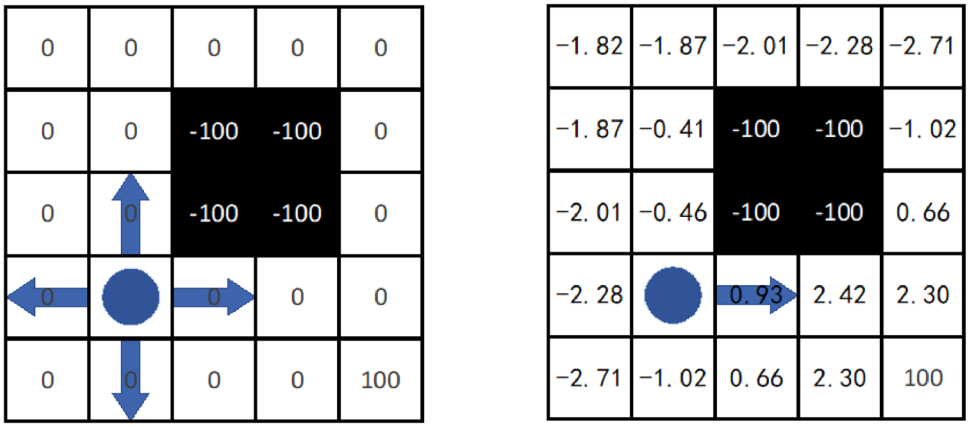


Fig. 1 Graph of pre-improved and improved reward mechanism

1. Guide the next action of the agent to always move toward the goal node, which means that the algorithm is always working toward the shortest and most reasonable path.
2. The reward mechanism designed by distance relationship gives different reward values to nodes, which guides the direction of the path and solves the problem of without guidance of the traditional reward mechanism.
3. The 8-direction omnidirectional search of path nodes is realized, which not only solves the right-angle turn problem of the 4-direction search planning path, but also solves the disorder of 8-direction search and the problem of jumping points.

4.2 Improving greedy strategy

How to achieve adaptive computation of the iteration threshold is the key to solve the problem of algorithm generalization ability. To address this key challenge, we propose an $\varepsilon - acc - increasing$ greedy mechanism to improve the algorithm. The idea of the mechanism is mainly reflected in two aspects: on the one hand, it promotes the greedy factor ε gradually increases in the exploration stage, so that it is sufficiently random in the exploration phase to fully update the Q-table. As the algorithm approaches the exploit phase, it shifts to full exploitation rather than random selection, thereby improving efficiency.

On the other hand, we adopt "experiment–fitting–verification" mode to automatically obtain the iteration threshold of the greedy strategy reaching the exploit stage in different environments. The algorithm adjusts the greedy factor before reaching the iteration threshold, so that the agent can fully explore and learn. After reaching the iteration threshold, the greedy factor is set to 1 and enters the exploitation stage. In the exploitation, it will longer time take time for random exploration, which is more scientific and efficient.

The core of $\varepsilon - acc - increasing$ greedy strategy is that the algorithm automatically obtains the threshold of reinforcement learning iterations, so as to achieve adaptivity in different maps. Whether the threshold can be automatically and accurately obtained is the key factor to determine for the success of the algorithm improvement. Through theory and analysis, we know that the factors affecting the iteration threshold are maps, specifically the map scale and the proportion of obstacles in the map. In this paper, the design process of $\varepsilon - acc - increasing$ greedy strategy is to use the methods of experiment, control variables, curve fitting, through the engineering mode of "experiment–fitting–verification" to achieve the strategy design, so as to improve the QL algorithm.

The specific steps are as follows:

Calculate the iteration threshold for a specific map. We collect each iteration time of QL algorithm on a specific map and curve fitting the data. In this experiment, the map scale is 10×10, and the ratio of obstacles is 40%. We design a functional relationship between QL algorithm iteration time and the iteration threshold, as shown in Eq. (11).

$$f(x) = \left(\frac{1}{19.947}\right)^3 e^{-5.273x} + \left(\frac{1}{3.415}\right)^3 e^{\left(\frac{1}{2.541}\right)^3 x} \tag{11}$$

In Eq. (11), x represents the number of iterations.

Its derivative function is shown in Eq. (12).

$$f'(x) = -\left(\frac{1}{2.726}\right)^5 e^{-5.273x} + \left(\frac{1}{2.946}\right)^6 e^{\left(\frac{1}{2.541}\right)^3 x} \tag{12}$$

2. We fixed the same proportion of obstacles in different maps, changed the map scale and obtained the relationship between the map scale and the iteration threshold. The same method is used to obtain the number of iterations when the algorithm iteration time is stable in other map scale scenarios. A total of 9 groups of data are collected and fitted. We design a functional relationship between map scale and the iteration threshold, as shown in Eq. (13).

$$f(s) = 1.131 * s^{1.411} + 25.15 \tag{13}$$

where s represents the map scale and $f(s)$ is the iteration threshold.

3. We fixed the map scale, changed the proportion of obstacles in the map and obtained the relationship between the proportion of obstacles and the iteration threshold. The same method is used to obtain the number of iterations when the algorithm iteration time is stable in other proportion of obstacles. A total of 9 groups of data are collected and fitted. We design a functional relationship between proportion of obstacles and the iteration threshold, as shown in Eq. (14).

$$f(r) = 398.5 * r^{0.614} + 75.09 \tag{14}$$

where r represents the proportion of obstacles and $f(r)$ is the iteration threshold.

4. As introduced above, the threshold is related to the map scale and the proportion of obstacles in the map. We weight Eqs. (13) and (14) to obtain the automatic calculation equation of the iteration threshold. During the experiment, it is found that the effect of the map scale on the iteration threshold is much greater than that of the obstacle proportion, so Eq. (13) should be assigned a larger weight as shown in Eq. (15).

$$pre_{ep} = n * f(s) + (1 - n) * f(r) \tag{15}$$

Taking all the above factors into consideration, the iteration threshold is automatically calculated as Eq. (16).

$$pre_{ep} = 1.0179 * s^{1.411} + 39.85 * r^{0.614} + 30.144 \tag{16}$$

5. Design a new "explore–exploit" mechanism function. After obtaining the iteration threshold of any map, we need to obtain a functional relationship that the exploration factor ε changes as the number of iterations increases. The number of iterations variable is denoted as ep. The function has two restrictions.

First, the variation range of the exploration factor ε is between 0.5 and 1, and the value range is selected according to the characteristics of QL algorithm. The algorithm is more greedy in the implementation process, and the selection of each step is in the direction of maximizing the reward value. Therefore, the variation range of ε can not only take into account the random effect in the early stage of exploration, but also fully reflect the greedy characteristics of QL algorithm.

Second, when ε changes from 0.5 to 1, the rate of change is first fast and then slow. Since the environment is completely unfamiliar and the experience gained after iteration is relatively rich, so the rate of change of ε is fast. However, as the number of iterations increases, the experience of QL algorithm gradually stabilizes, so the rate of change of ε is slow and gradually tends to 1. When the iteration threshold is finally reached, the exploration factor is set to 1.

We apply the logarithmic function for curve fitting and design a function that ε changes with episode, as shown in Eq. (17).

$$\varepsilon = 0.1 \cdot \left(\ln \left(\frac{ep}{pre_{ep}} + 0.9^7 \right) + \sqrt{33} \right) \tag{17}$$

Combining the above factors, the greedy strategy proposed in this paper is shown in Eq. (18). This greedy strategy is denoted $\varepsilon - acc - increasing$ greedy strategy.

$$\varepsilon = \begin{cases} 1, & ep \geq pre_{ep} \\ 0.1 \cdot \left(\ln \left(\frac{ep}{pre_{ep}} + 0.9^7 \right) + \sqrt{33} \right), & ep < pre_{ep} \end{cases} \tag{18}$$

6. To verify the scientific of $\varepsilon - acc - increasing$ strategy, the verification is carried out in a map scenario where no experimental data are collected. Comparing the theoretical value with the actual measured value, if there is no significant difference, it can be proved that the theory is scientific and reasonable. The experimental results show that the $\varepsilon - acc - increasing$ greedy strategy is correct and effective.

5 Experiment and discuss

5.1 Experimental setup

To test the universality of the improved QL algorithm in different physical scenarios, we design scenarios with different map scales and different obstacle density in experiments to verify the performance of the algorithm. Scenario 1, the map scale is 20×20, the proportion of obstacles to the map scale is 10%, the obstacles are sparse, and the task scenario is relatively simple. Scenario 2, the map scale is 50×50, and the proportion of obstacles to the map scale is 25%, the obstacles are denser, and the task scenario is more complex. Scenario 3, the map scale is 100×100, the proportion of obstacles to the map scale is 30%, the obstacles are very dense, and the task scenario is very complex.

In the section, all tests were performed on a PC with Windows 10 as OS with I7-8550U CPU 1.80GHz and 16GB RAM.

5.2 Iteration threshold experiment

How to automatically obtain the iteration threshold of QL algorithm? During the experiment, we collected the time of each iteration and took the number of iterations when the iteration time tended to be stable as the iteration threshold. This article takes a 10*10 scale map as the sample to describe

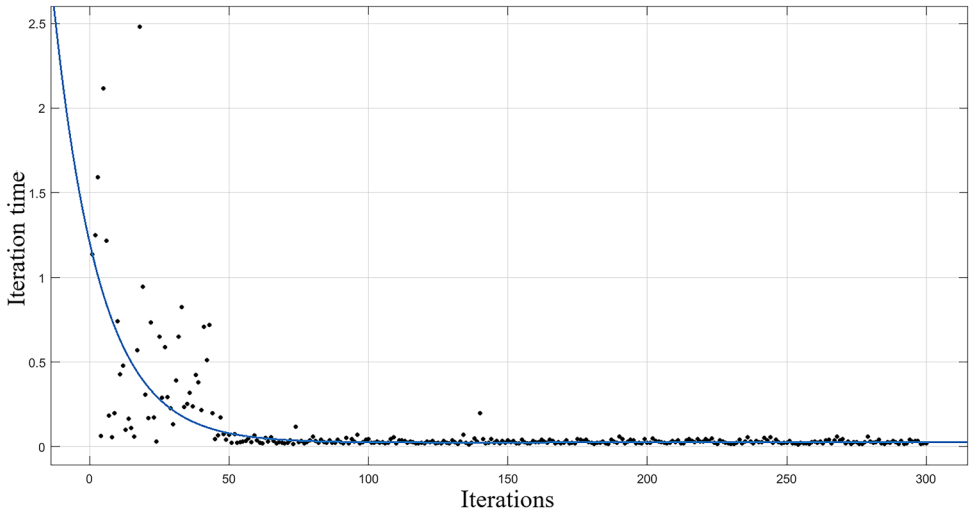


Fig. 2 Relationship between iteration time and iterations

Table 1 Correspondence of map size and iteration threshold

Map size	pre_{ep}
10*10	51
15*15	82
20*20	106
25*25	129
30*30	160
35*35	183
40*40	245

the automatic calculation process of the iteration threshold in detail.

We collect each iteration time and fit the data. The fitting curve is shown in Fig. 2, and the fitting function is shown in (12):

In this test environment, the iteration threshold of the algorithm is 51. According to the same theory, we calculate the iteration threshold for fitting maps of different scales. We analyzed the influencing factors of the iteration threshold and found that it is mainly related to the map size and the proportion of obstacles in the map.

1. We fixed the proportion of obstacles to 40%, changed the size of the map and obtained the data of the map size and iteration threshold through experiments, as shown in Table 1.

We perform curve fitting on the data to obtain the curve of map scale and iteration threshold as shown in Fig. 3. The curve function is Eq. (13).

2. We fixed the size of the map to 50×50, changed the proportion of obstacles and obtained the proportion of obstacles and iteration threshold through experiments, as shown in Table 2.

We perform curve fitting on the data to obtain the curve of proportion of obstacles and iteration threshold as shown in Fig. 4. The curve function is Eq. (14).

3. We find that as the map scale increases, the effect of iteration threshold becomes more significant. However, as the obstacle proportion increases, the impact on the iteration threshold becomes smaller. According to the theory we proposed, we can derive function of the algorithm's iteration threshold concerning map scale and obstacle proportion, as shown in Eq. (16).

4. According to Eq. (16), it can calculate the iteration threshold of any map. To verify the correctness of this function, we choose a strange map as the verification sample. The map scale is 100×100, of which the proportion of obstacles is 40%. We can find that the effect of two thresholds can be regarded as the same under the experimental conditions. Therefore, we can draw conclusions that the fitting function can adapt to map changes well.

5.3 Simulation experiment

In all the maps of the experiment, black squares indicate obstacles, white squares indicate blank area, blue square indicates the starting point of the path, the green square indicates the end point of the path, the grey squares indicate the expansion distance, and the red squares indicate the path node. Obstacles are randomly generated on maps based on the specified proportion.

In this article, we call the QL algorithm after the improvement in the reward mechanism and "explore–exploit" mechanism as the TQ-learning algorithm. The TQ-learning algorithm which increases the expansion distance strategy is called ETQ-Learning algorithm. In this section, the traditional QL algorithm, TQ-Learning algorithm and ETQ-Learning algorithm are carried out to verify the performance and universality of the algorithm in different task scenarios.

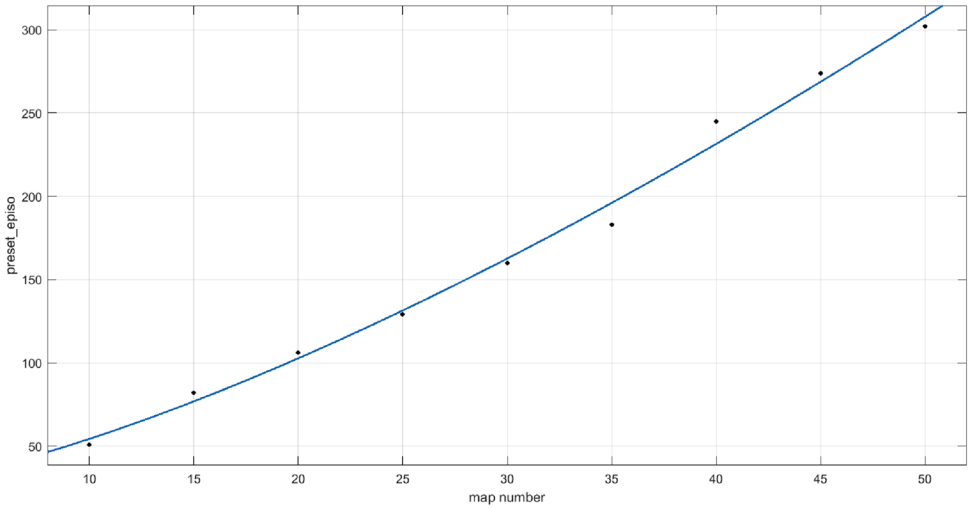


Fig. 3 Relationship curve of map scale and iteration threshold

Table 2 Correspondence of obstacle proportion and iteration threshold

Obstacle proportion	pre_{ep}
0.10	172
0.15	200
0.20	223
0.25	243
0.30	269
0.35	283
0.40	302

5.3.1 Simulation experiment results

(1) algorithm testing in scenario 1

Iterations of each algorithm are set to 300. The path planning results of three algorithms in the Scenario 1 are shown in Fig. 5:

In Fig. 5a, the path of QL algorithm has a relatively obvious "round-trip" phenomenon, and the length of the path increases significantly and obviously does not conform to the optimal characteristic. In Fig. 5b, the path of TQ-learning algorithm is always along the shortest path to the end point. In Fig. 5c, the expansion distance is designed in the algorithm, and ETQ-learning algorithm sets up a "collision buffer " around obstacles, which reduces the risk of collision during the robot's travel. The reason for the "round-trip" phenomenon shown in Fig. 5a is that the search process of QL algorithm is random and disordered in the exploration phase. The TQ-learning algorithm and the ETQ-learning algorithm design a distance-guided reward mechanism in the exploration phase, which makes the exploration factor change regularly and makes the planning path orderly and smooth.

The iteration time curves of three algorithms are shown in Fig. 6. The abscissa indicates the number of iterations, and the ordinate indicates the iteration time. In Fig. 6, the green curve indicates iteration time of the QL algorithm, the orange curve indicates the iteration time of TQ-learning algorithm, and the blue curve indicates the iteration time of ETQ-learning algorithm (the same below).

The test data of three algorithms in scenario 1 are shown in Table 3. Among them, the iteration time represents the average time of a single iteration, which indicates the efficiency of the algorithm. The number of path nodes represents the length of the planned path. The number of critical nodes represents the number of nodes adjacent to obstacles in the path.

(2) algorithm testing in Scenario 2

Iterations of each algorithm are set to 800. The path planning results of three algorithms in the scenario 2 are shown in Fig. 7:

In Fig. 7a, the path of QL algorithm still has a relatively obvious "round-trip" phenomenon, while the paths of TQ-learning algorithm and ETQ-learning algorithm belong to the shortest and optimal paths. The iteration time curves of three algorithms are shown in Fig. 8. The test data of three algorithms in Scenario 2 are shown in Table 4.

(3) algorithm testing in Scenario 3. Iterations of three algorithms are set to 1500 times. The path planning results of three algorithms in the scenario 3 are shown in Fig. 9:

In the more complex Scenario 3, the characteristics of the three algorithms are the same as those of Scenario 1 and Scenario 2. The iteration time curves of three algorithms are shown in Fig. 10. The test data of three algorithms in Scenario 3 are shown in Table 5.

5.3.2 Analysis and discussion

By testing the QL algorithm, TQ-learning algorithm and ETQ-learning algorithm in three scenarios, we obtain the iteration time, the path length and the number of critical nodes. Analyzing these results will provide insights into the performance of these algorithms.

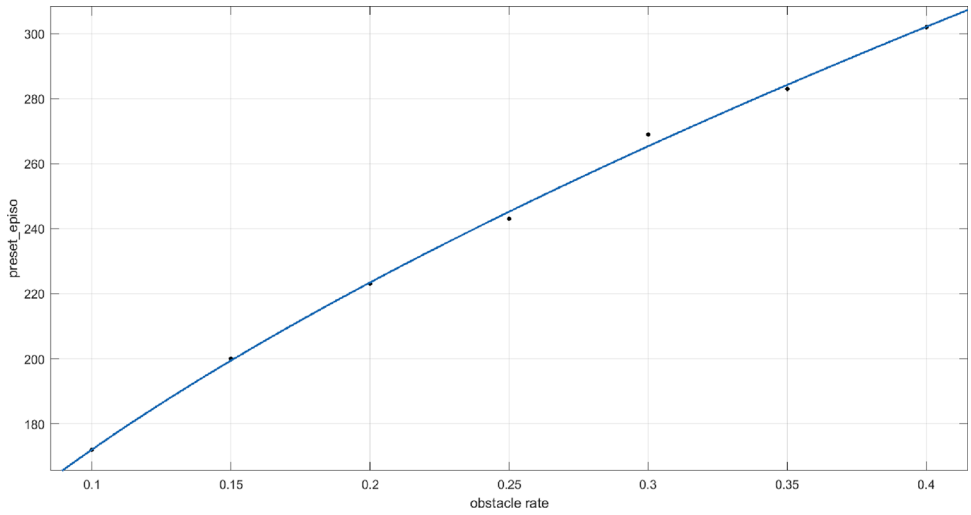
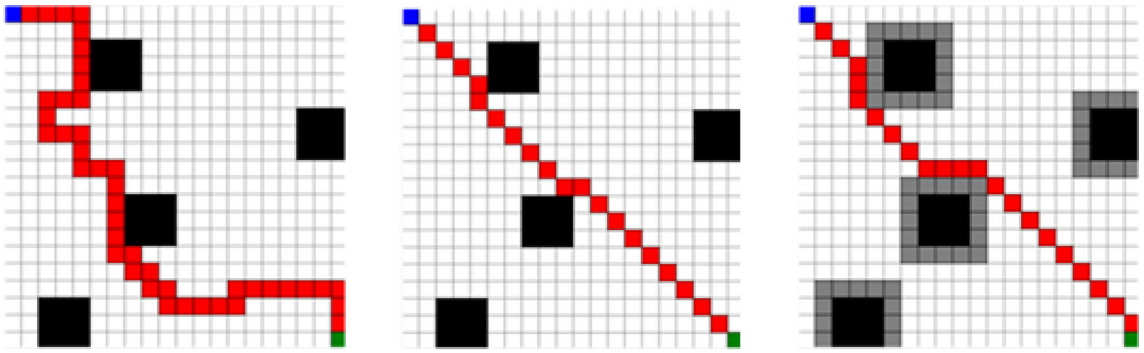


Fig. 4 Curve of obstacle proportion and iteration threshold



(a) Q-learning path (b) TQ-learning path (c) ETQ-learning path

Fig. 5 Paths of algorithm planning in scenario 1

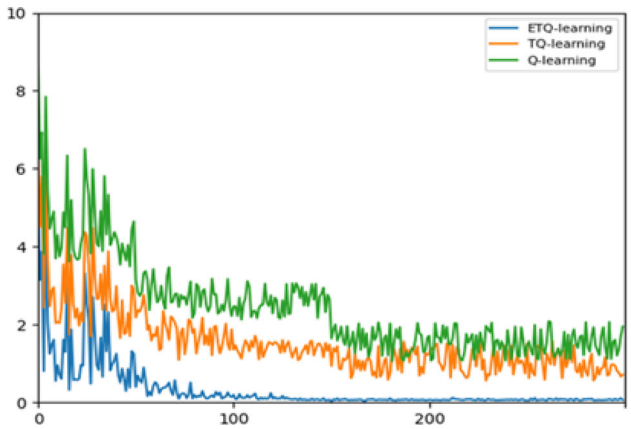


Fig. 6 Graph of algorithms iteration time in scenario 1

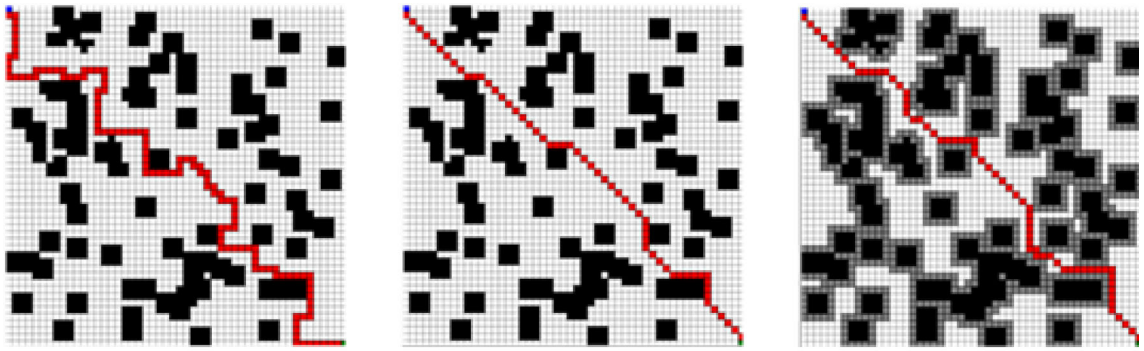
1. In the experiment of Scenario 1, the path of QL algorithm has a significant "round-trip" phenomenon, meaning that following this path would lead to a detour. This phenomenon is attributed to the non-guided nature of the reward mechanism in the QL algorithm, resulting in the occurrence of "return points" and "jump points" within the path. To address this issue, we redesigned the reward mechanism and applied it to both the TQ-learning algorithm and ETQ-learning algorithm. These enhanced algorithms guide the agent's next action to consistently search for the goal node, thereby planning the optimal path with the shortest and minimal right-angle turns.

Table 3 reveals that the path length of TQ-learning algorithm is 44.18% of that of QL algorithm, and the path length of ETQ-learning algorithm is merely 48.84% of the QL algorithm's path length. Consequently, whether considering the "straight line" effect of the path or the path's overall length, both the path of TQ-learning algorithm and ETQ-learning algorithm are significantly better than QL algorithm. This superiority is indicative of a more global optimal solution. Notably, this characteristic is consistently verified in Scenario 2 and Scenario 3.

2. In the experiment of Scenario 1, the efficiency of TQ-learning algorithm is 159% of that of QL algorithm. We redesign the reward mechanism and the " explore–exploit " mechanism (ε-acc-increasing greedy strategy), which is the reason for improving the efficiency of TQ-learning algorithm. This mechanism promotes TQ-learning algorithm to

Table 3 Statistics of algorithms in Scenario 1

Name	Iterations	Time/s	Number of path nodes	Number of critical nodes
Q-learning	300	3.484	43	6
TQ-learning		2.184	19	2
ETQ-learning		1.047	21	0



(a) Q-learning path (b) TQ-learning path (c) ETQ-learning path

Fig. 7 Paths of algorithm planning in scenario 2

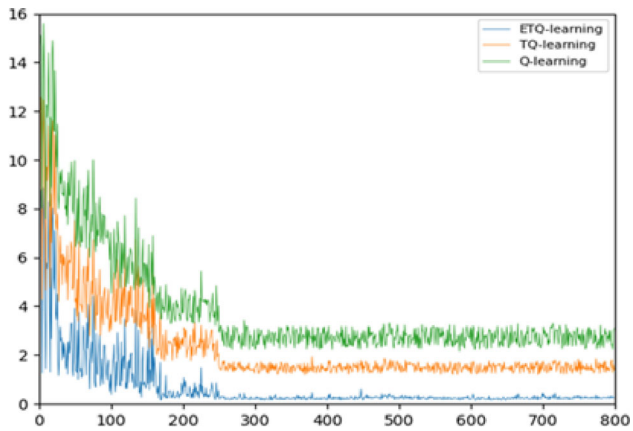


Fig. 8 Graph of algorithms iteration time in scenario 2

gradually increase the exploration factor ε from small to large in the exploration phase, so that it is sufficiently random in the exploration phase to fully update the Q-table. As the algorithm approaches the exploit phase, it shifts to full exploitation rather than random selection, thereby improving efficiency.

It's worth noting that the efficiency of TQ-learning algorithm is only improved by 59% in scenario 1, because the map scale is small and the limited number of explorable nodes. However, as the map scale gradually increases, the more nodes can be explored, and the ε-acc-increasing greedy strategy significantly improves efficiency. In Scenario 2, the efficiency of TQ-learning algorithm improves by 101%, and in Scenario 3, the efficiency of TQ-learning algorithm improves by 173%. These experimental results verify the scientific of the algorithm optimization theory.

3. The experiment results of scenario 1 are shown in Fig. 5c. After utilizing the expansion distance, ETQ-learning algorithm generates a "collision buffer" between the path and obstacles. The statistical analysis shows that the critical node count for the ETQ-learning algorithm has reduced to 0, which can effectively reduce the risk of collision. This finding is consistent and validated in Scenarios 2 and 3.

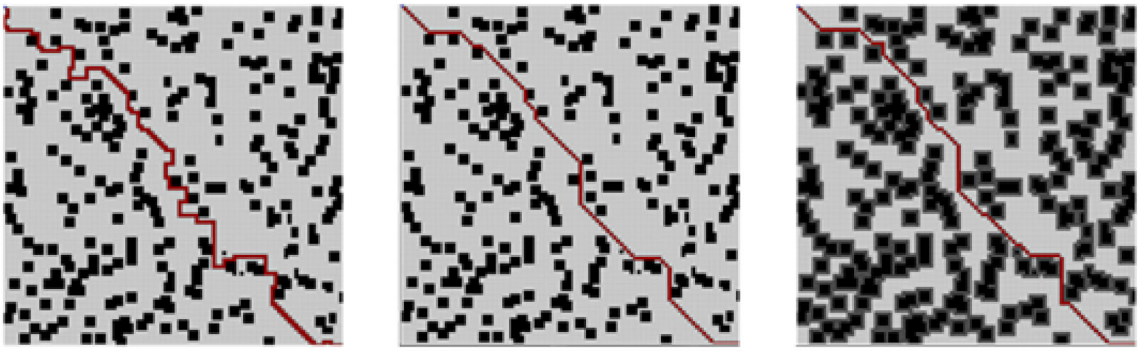
An additional advantage of employing the expansion distance is its positive impact on algorithm efficiency. In scenario 1, The efficiency of ETQ-learning algorithm is 233% higher than QL algorithm. In Scenario 2, the efficiency of ETQ-learning algorithm is 293% higher than QL algorithm, and in Scenario 3, this value is 351%. It can be seen that the above rules that as the proportion of obstacles in the map increases, expansion distance improves the algorithm efficiency more significantly.

To further validate the performance and generalization capabilities of the ETQ-learning algorithm, we conducted additional experiments using randomly generated maps in addition to the above experiments. Specifically, we randomly generated 5 maps for each scenario to simulate various complex physical environments. We conducted repeated tests on the 15 generated maps; the results were consistent with mentioned above.

Through testing in different scenarios, ETQ-learning algorithm has better adaptability to environments. From the above analysis and discussion, ETQ-learning algorithm significantly improves the performance of TQ-learning algorithm, which mainly focuses on three aspects. Firstly, it optimizes

Table 4 Statistics of algorithms in Scenario 2

Name	Iterations	Time/s	Number of path nodes	Number of critical nodes
Q-learning	800	6.821	117	22
TQ-learning		3.394	58	16
ETQ-learning		1.735	68	0



(a) Q-learning path (b) TQ-learning path (c) ETQ-learning path

Fig. 9 Paths of algorithm planning in Scenario 3

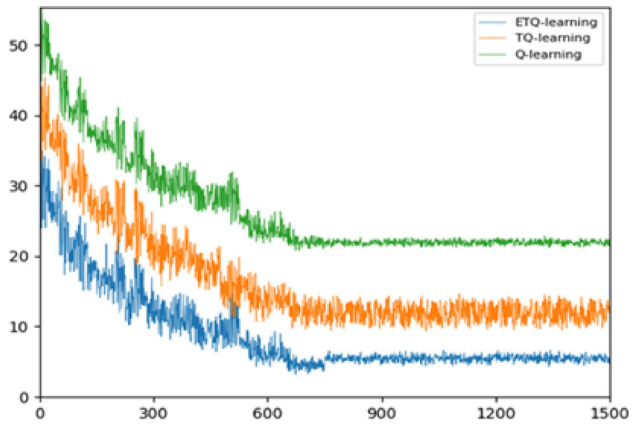


Fig. 10 Graph of algorithms iteration time in Scenario 3

the planning path of QL algorithm, which is the most basic and most important indicator of path planning. Secondly, it improves the efficiency generalization ability of QL algorithm, which is one of the important indicators to make agents work efficiently. Thirdly, the QL algorithm's collision avoidance performance has been improved to minimize collision risks during the agents' motion.

5.4 Comparison experiment

To verify the performance advantages of the ETQ-learning algorithm and other QL optimization algorithms, in this section, ETQ-learning algorithm is compared with AQ-learning algorithm [4], IQ-FA algorithm [5], YQ-learning algorithm [6] and EMQL algorithm [7].

1. Comparison of planning path

IQ-FA algorithm is an improved version of the QL algorithm introduced in [13]. However, [13] does not provide the planning path of IQ-FA algorithm. We analyze its optimization method and find that IQ-FA algorithm does not solve the issue of guiding the search, so the planned path still has "round-trip point" " and "jump point." AQ-learning algorithm is applied to three-dimensional map in [6]. Its path fluctuates in the empty space as shown in Fig. 11a, indicating that the path is not smooth. The planning path of YQ-learning is shown in Fig. 11b, and the planning path of EMQL algorithm is shown in Fig. 11c. Algorithm paths still directly exist the phenomenon of path round trip, so they are not global optimal solutions. Based on this analysis, the path planning effectiveness of ETQ-learning algorithm is superior.

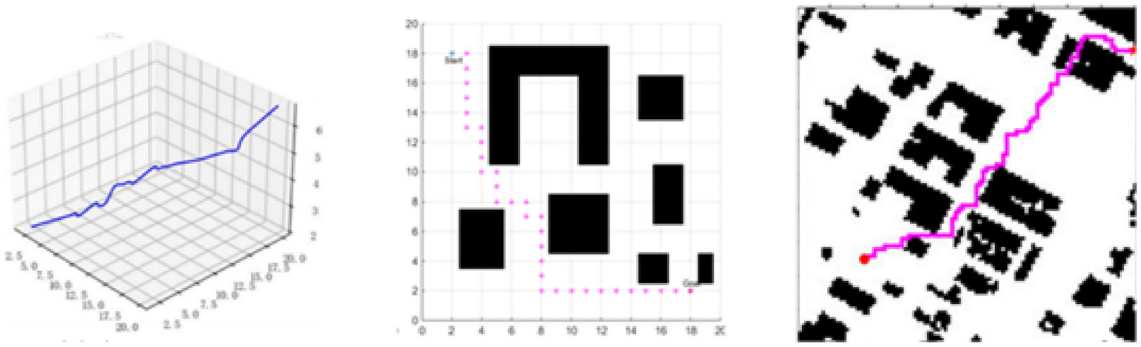
1. Comparison of algorithm efficiency

If absolute data are directly used for comparison, it may seem unacceptable because efficiency of the algorithm is affected by many factors, such as the programming language and computer performance. Therefore, it may be unscientific to use the absolute time. To reflect the scientific of comparison, we adopt the proportional transformation comparison method. This method is based on efficiency of the QL algorithm in other experiment and calculates the ratio of the two benchmarks: the efficiency of ETQ-learning algorithm and the ratio are multiplied to obtain the relative efficiency. The relative efficiency was used for comparison with other method.

According to [6], the average iteration time of AQ-learning algorithm is 5.31s. We select a group of data with the lowest efficiency of the ETQ-learning algorithm. The average

Table 5 Statistics of algorithms in Scenario 3

Name	Iterations	Time/s	Number of path nodes	Number of critical nodes
Q-learning	1500	26.046	227	41
TQ-learning		9.541	116	21
ETQ-learning		5.775	133	0



(a) AQ algorithm path (b) YQ-learning path (c) EMQL algorithm path

Fig. 11 Other algorithm paths

Table 6 Efficiency comparison between ETQ-learning and AQ-learning

	Q-learning	AQ-learning	ETQ-learning
Time	7.45	5.31	–
This experiment time	3.484	–	1.047
Ratio	2.138	1	2.138
Relative time	–	5.31	2.238

Table 7 Efficiency comparison between ETQ-learning and IQ-FA

	Q-learning	IQ-FA	ETQ-learning
Time	3.71	4	–
This experiment time	3.484	/	1.047
Ratio	1.064	1	1.064
Relative time	–	4	1.114

iteration time of the algorithm in Scenario 1 is 1.047s. Table 6 shows the comparison data between ETQ-learning algorithm and AQ-learning algorithm. The efficiency of ETQ-learning algorithm is 2.37 times that of AQ-learning algorithm, which is superior to AQ-learning algorithm.

Using the same comparison method, the comparison data of ETQ-learning algorithm and IQ-FA algorithm are shown in Table 7. The efficiency of ETQ-learning algorithm is 3.59 times that of IQ-FA algorithm, which is also superior to IQ-FA learning algorithm.

Similarly, the comparison data of ETQ-learning algorithm and YQ-learning algorithm are shown in Table 8. The efficiency of ETQ-learning algorithm is 1.78 times that of YQ-learning algorithm. The comparison data of ETQ-learning algorithm and EMQL algorithm are shown in Table 9. The efficiency of ETQ-learning algorithm is 2.434 times that of EMQL algorithm which is also superior to YQ-learning algorithm and EMQL algorithm.

In the comparison experiment, the data of ETQ-learning select the test data with the lowest efficiency. If it is in a larger map or a complex map with a higher proportion of obstacles, the algorithm efficiency of ETQ-learning is better. Through the above comparison, we can draw a conclusion that the efficiency of ETQ-learning algorithm is better than that of other QL optimization algorithms, which shows that ETQ-learning algorithm is excellent in terms of efficiency.

6 Conclusion

The ETQ-learning algorithm designed in this paper represents a significant advancement over the QL algorithm. We have redesigned the reward mechanism and greedy strategy and fundamentally solved the many deficiencies of the QL algorithm. We have focused on improving the algorithm's efficiency so that the agent performs tasks more efficiently. We address the lack of

algorithm's generalization to effectively apply to complex physical environments. We introduced expansion distance

Table 8 Efficiency comparison between ETQ-learning and YQ-learning

	Q-learning	YQ-learning	ETQ-learning
Time	4.45	2.24	–
This experiment time	3.484	–	1.047
Ratio	1.277	1	1.277
Relative time	–	2.24	1.285

Table 9 Efficiency comparison between ETQ-learning and EMQL

	Q-learning	EMQL	ETQ-learning
Time	7.0623	5.1649	–
This experiment time	3.484	–	1.047
Ratio	2.027	1	2.027
Relative time	–	5.1649	2.122

that increase the algorithm's obstacle avoidance performance. Furthermore, we conducted comprehensive comparisons between the ETQ-learning algorithm and other optimized QL algorithm and achieved significant breakthroughs in various aspects.

Author Contributions Conceptualization was performed by H.W.; methodology by H.W. and J.J.; software by Q.W.; validation by Q.W. and R.L.; formal analysis by H.H.; writing original draft by H.W.; writing review and editing by J.J. and R.L.; funding by X.Q.

Funding The government of Henan Province, China, supported the research in the form of the Henan Provincial Key R & D Special Funds(221111210300).

Data availability available in GitHub https://github.com/wanghw1003/ETQlearning.

Declarations

Conflict of interest The authors declare that they do not have any conflict of interest.

Ethical approval This article does not contain any studies with human participants or animals performed by any of the authors.

Consent to Participate All authors agreed to participate the research.

Consent for Publication All authors read and approved the final manuscript.

References

1. Costa MM, Silva MF (2019) A survey on path planning algorithms for mobile robots. In: 2019 IEEE international conference on autonomous robot systems and competitions (ICARSC), IEEE, pp. 1–7
2. Wang H, Lou S, Jing J, Wang Y, Liu W, Liu T (2022) The EBS-A* algorithm: an improved A* algorithm for path planning. PLoS ONE 17(2):e0263841
3. Wang H, Qi X, Lou S, Jing J, He H, Liu W (2021) An efficient and robust improved A* algorithm for path planning. Symmetry 13(11):2213
4. Li D, Yin W, Wong WE, Jian M, Chau M (2021) Quality-oriented hybrid path planning based on A* and Q-learning for unmanned aerial vehicle. IEEE Access 10:7664–7674
5. Wang B, Liu Z, Li Q, Prorok A (2020) Mobile robot path planning in dynamic environments through globally guided reinforcement learning. IEEE Robot Autom Lett 5(4):6932–6939
6. lipei S (2018) Research on intelligent vehicle dynamic path planning algorithm based on improved Q-learning
7. Zhao M, Lu H, Yang S, Guo F (2020) The experience-memory Q-learning algorithm for robot path planning in unknown environment. IEEE Access 8:47824–47844
8. Wang J, Ren Z, Liu T, Yu Y, Zhang C (2020) Qplex: duplex dueling multi-agent Q-learning, arXiv preprint arXiv:2008.01062
9. Hasselt H (2010) Double Q-learning, Advances in neural information processing systems. 23
10. guojun M, shimin G (2021) Improved Q-learning algorithm and its application to path planning. J Taiyuan Univ Technol 52(1):91
11. Yunjian P, Jin L (2022) Q-learning path planning based on exploration-exploitation trade-off optimization. Comput Technol Dev. 32(1–7)
12. chengbo W, zinyu Z, zhiqiang Z, shaobo W (2018) Path planning for unmanned vessels based on Q-learning. Ship Ocean Eng 47(5):168–171
13. Fortunato M, Azar MG, Piot B, Menick J, Osband I, Graves A, Mnih V, Munos R, Hassabis D, Pietquin O et al (2017) Noisy networks for exploration, arXiv preprint arXiv:1706.10295
14. Ates U (2020) Long-term planning with deep reinforcement learning on autonomous drones. In: Innovations in intelligent systems and applications conference (ASYU). IEEE 2020:1–6
15. Zijian H, Xiaoguang G, Kaifang W, Yiwei Z, Qianglong W (2021) Relevant experience learning: a deep reinforcement learning method for UAV autonomous motion planning in complex unknown environments. Chin J Aeronaut 34(12):187–204
16. Schulman J, Levine S, Abbeel P, Jordan M, Moritz P (2015) Trust region policy optimization. In: International conference on machine learning, PMLR, pp. 1889–1897
17. Zhang T, Huo X, Chen S, Yang B, Zhang G (2018) Hybrid path planning of a quadrotor UAV based on q-learning algorithm. In: 37th Chinese control conference (CCC). IEEE 5415–5419
18. Schulman J, Wolski F, Dhariwal P, Radford A, Klimov O (2017) Proximal policy optimization algorithms, arXiv preprint arXiv:1707.06347
19. Andrychowicz M, Wolski F, Ray A, Schneider J, Fong R, Welinder P, McGrew B, Tobin J, Pieter Abbeel O, Zaremba W (2017) Hindsight experience replay, Advances in neural information processing systems 30
20. Haarnoja T, Zhou A, Abbeel P, Levine S (2018) Soft actor-critic: off-policy maximum entropy deep reinforcement learning with a stochastic actor. In: International conference on machine learning, PMLR, pp. 1861–1870

21. Kumar A, Gupta A, Levine S (2020) Discor: corrective feedback in reinforcement learning via distribution correction. Adv Neural Inf Process Syst 33:18560–18572
22. Kong D, Yang L (2022) Provably feedback-efficient reinforcement learning via active reward learning. Adv Neural Inf Process Syst 35:11063–11078
23. Song Y, Steinweg M, Kaufmann E, Scaramuzza D (2021) Autonomous drone racing with deep reinforcement learning. In: 2021 IEEE/RSJ international conference on intelligent robots and systems (IROS), IEEE, pp. 1205–1212
24. Wang Z, Yang H, Wu Q, Zheng J (2021) Fast path planning for unmanned aerial vehicles by self-correction based on Q-learning. J Aerosp Inf Syst 18(4):203–211
25. Yan C, Xiang X (2018) A path planning algorithm for UAV based on improved q-learning. In: 2nd international conference on robotics and automation sciences (ICRAS). IEEE :1–5
26. de Carvalho KB, de Oliveira IRL, Villa DK, Caldeira AG, Sarcinelli-Filho M, Brandão AS (2022) Q-learning based path planning method for UAVs using priority shifting. In: 2022 International Conference on Unmanned Aircraft Systems (ICUAS), IEEE, pp. 421–426
27. Li S, Xu X, Zuo L (2015) Dynamic path planning of a mobile robot with improved Q-learning algorithm. In: IEEE international conference on information and automation. IEEE 409–414
28. Wang Y, Wang S, Xie Y, Hu Y, Li H (2022) Q-learning-based collision-free path planning for mobile robot in unknown environment. In: 2022 IEEE 17th conference on industrial electronics and applications (ICIEA), IEEE, pp. 1104–1109