Proceedings of the 4th International Conference on Signal Processing and Machine Learning
DOI: 10.54254/2755-2721/54/20241447

# The impact of Q-learning parameters on robot path planning problems in different complex environment

**Zihan Wang**

School of Future Science and Engineering, Soochow University, Suzhou, Jiangsu Province, 215006, China

2129401045@stu.suda.edu.cn

**Abstract.** With the development of reinforcement learning algorithms, its efficient problem-solving model and the universality of the method have been favoured by many scholars. More and more robot path planning problems are solved using reinforcement learning methods. This article focuses on the Q-learning algorithm, uses MATLAB to study the impact of Q-learning parameters on robot path planning problems in different complex environments, and tries to find the optimal solution to the parameters. Research has found that environmental complexity has a significant impact on the speed at which robots solve path planning problems, and the optimal solutions to problem parameters in different environments require detailed analysis of specific problems.

**Keywords:** Reinforcement Learning, Robot Path Planning, Q-Learning, Learning rate.

## 1. Introduction

The creation of completely autonomous beings that interact with their environment to learn the best behaviours and eventually improve them through trial and error is one of the main objectives of the area of artificial intelligence (AI). Making AI systems that are responsive and effectively learn has long been a challenge, from robots that can sense and react to their surroundings to purely software-based agents that can interact with plain English and multimedia.

In the field of robotics, path planning problems are often discussed. How to efficiently plan the target path is a question that scholars have always been concerned about. Currently, many algorithms have been used to solve robot path planning problems. The Artificial Potential Field Approach for Mobile Robot Navigation was introduced by Khatib in 1986 [1]. Choset found the intended path for the robot using the roadmap technique [2]. Holland used genetic algorithms in computer science, and they are widely used in many branches of technology and research, including robot navigation [3].

A rigorous mathematical framework for experience-driven autonomous learning is reinforcement learning (RL). Although RL has had some historical triumphs, earlier methods lacked scalability and were essentially constrained to rather low-dimensional issues. Since RL algorithms are not immune to these problems, they are confined in this way. This is because RL algorithms share the same complexity difficulties as other algorithms, such as memory difficulty, computational complexity, and in the case of machine learning techniques, sample complexity. Academics now have new methods to handle these issues as a result of the recent development of deep learning, which depends on the potent function approximation and representation learning features of deep neural networks. [4].

With the recent rapid development of machine learning, data-driven technology has gradually been applied to the autonomous navigation problem of robots. C. Sun use the Multi-agent Particle Environment (MPE) cooperative navigation scenario as an experimental benchmark job to evaluate the effectiveness of their suggested multi-agent Reinforcement Learning (MARL) algorithms [5]. MIT's ACL laboratory researchers M. Everett have made significant advancements in the decentralized RL-based MRMP (DecRL-MRMP) methods sector [6]. Amit Konar used Q-learning to improve mobile robot path planning problem [7]. There are significant advantages to using reinforcement learning to deal with robot path planning problems. By engaging in interactive learning with a variety of scenarios throughout the training stage, it can become independent of previous map data and develop higher generalisation abilities.

Among reinforcement learning algorithms, Q-learning has attracted the attention of many researchers. Q-learning uses the ε-greedy strategy to maintain a balance between exploration and exploitation. This allows the algorithm to continuously explore trade-offs between new behaviours and optimal behaviours during the learning process, thereby gradually optimizing the strategy. At the same time, the algorithm is relatively simple and easy to implement. This makes it one of the introductory algorithms for learning reinforcement learning, allowing researchers and developers to get started quickly. However, it usually requires a large amount of training data and time to obtain a high-quality policy and may converge slowly in some cases. Plenty of research has been done on parameters' influence on Q-learning. The convergence rate and learning rate employed in Q-learning have an intriguing connection [8]. Two key factors in Q-learning, the learning rate and the discount factor, are explored in detail with an eye toward application [9]. This article attempts to determine the impact of parameters and environmental complexity on the robot's learning rate through Q-learning, so that Q-learning can be used to solve the robot's path planning problem with higher efficiency. Few studies have comprehensively considered the impact of these aspects on Q-learning. This article comprehensively considers the impact of these factors to find out the conditions that can make Q-learning the fastest learning rate under different conditions. This article uses MATLAB to conduct simulation experiments, collect data, and analyse it.

## 2. Using Q-learning algorithm to solve path planning problems

Finding a collision-free path between an initial (start) and end configuration (target) inside a given environment is the definition of the path planning issue [10]. In this article, the author uses the Cell decomposition (CD) approach to divide the entire space into small square spaces, using the centre of each square grid as a waypoint, and then connecting the waypoints to complete the path planning problem.

In this article, the author sets up a closed environment that contains both a starting point and a target point. There are obstacles between the two points. The complexity of the environment is defined based on the number of obstacles and the number of bifurcations. Between the beginning location and the finish point, there are several routes.

Since the Q-learning method is a model-free learning strategy and saves a lot of time when creating the environment, especially when moving between multiple mazes during the experiment, it is utilized to find the shortest path. The following Figure 1 illustrates the general Q-learning procedure.

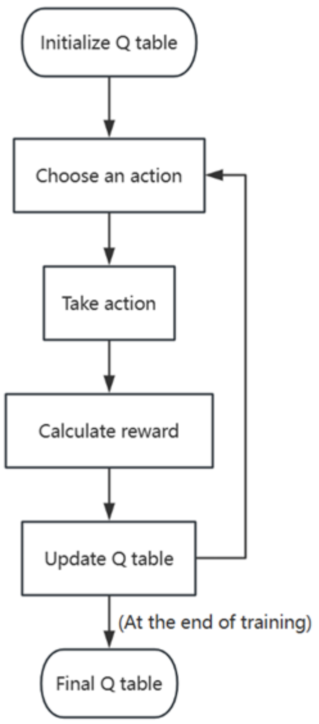

**Figure 1.** How general Q-learning works.
(Picture Credit to: Original)

In the general case, an equation is needed here to update the Q-table. Here Q-learning uses the equation below to update Q table:

$$Q^{new}(s,a) \leftarrow Q^{old}(s,a) + \alpha[r + \gamma * maxQ(s_{t+1}, a_{t+1}) - Q^{old}(s,a)] \quad (1)$$

In this equation, $\gamma$ is introduced as a discount factor ($\gamma \in [0,1]$). The proportion of long-term advantages at the present time is represented by the discount factor. When the value of the discount factor becomes close to zero, agents are more likely to focus on the near-term benefit and want to get the reward as soon as they can. The value of the discount factor approaches one when agents are more prepared to delay rewards, more concerned about the future, and more focused on future advantages. The learning rate is introduced as $\alpha(\alpha \in [0, 1])$. The learning rate is set, and the higher it is set and the nearer it is to 1, the faster the agents identify the shortest path. s represents the current state of the agent, a represents an action taken by the agent, and r represents the benefit obtained by the agent after taking an action.

The pseudocode for using Q-learning to complete path planning is shown in Algorithm 1.

**Algorithm 1. Pseudocode of Q-learning for path planning**

```
Initialize $\boldsymbol{Q(s,a)}$
Repeat many times
- Pick $s$ as start state
- Repeat each step to goal
   * Choose a based on $\boldsymbol{Q(s,a)}$ $\boldsymbol{\varepsilon}$-greedy
   * Do $a$, observe $r$, $s'$
   * $\boldsymbol{Q^{new}(s,a) \leftarrow Q^{old}(s,a) + \alpha[r + \gamma * maxQ(s_{t+1}, a_{t+1}) - Q^{old}(s,a)]}$
   * $s = s'$
- Until $s$ terminal
```

In this pseudocode, $Q(s,a)$ is first initialized, which is the matrix representing the Q-table. Then a loop is used to let the agent select each step, then calculate the reward and update the Q-table, and finally connect all steps to obtain the final path.

## 3. Results

### *3.1. Background setting*

This article will use the branches of the maze to represent the complexity of the maze. Firstly, the author randomly generates mazes with branches of 5, 6, and 7 through MATLAB Robotics Playground Expansion Pack. An example of the generated maze is shown in Figure 2(a)(b)(c). After obtaining the maze image, we set the maze environment, with a size of 40 * 40. The author also initialized the Q table with a value of -1 for obstacles, 1 for endpoints, and 0 for other open spaces.

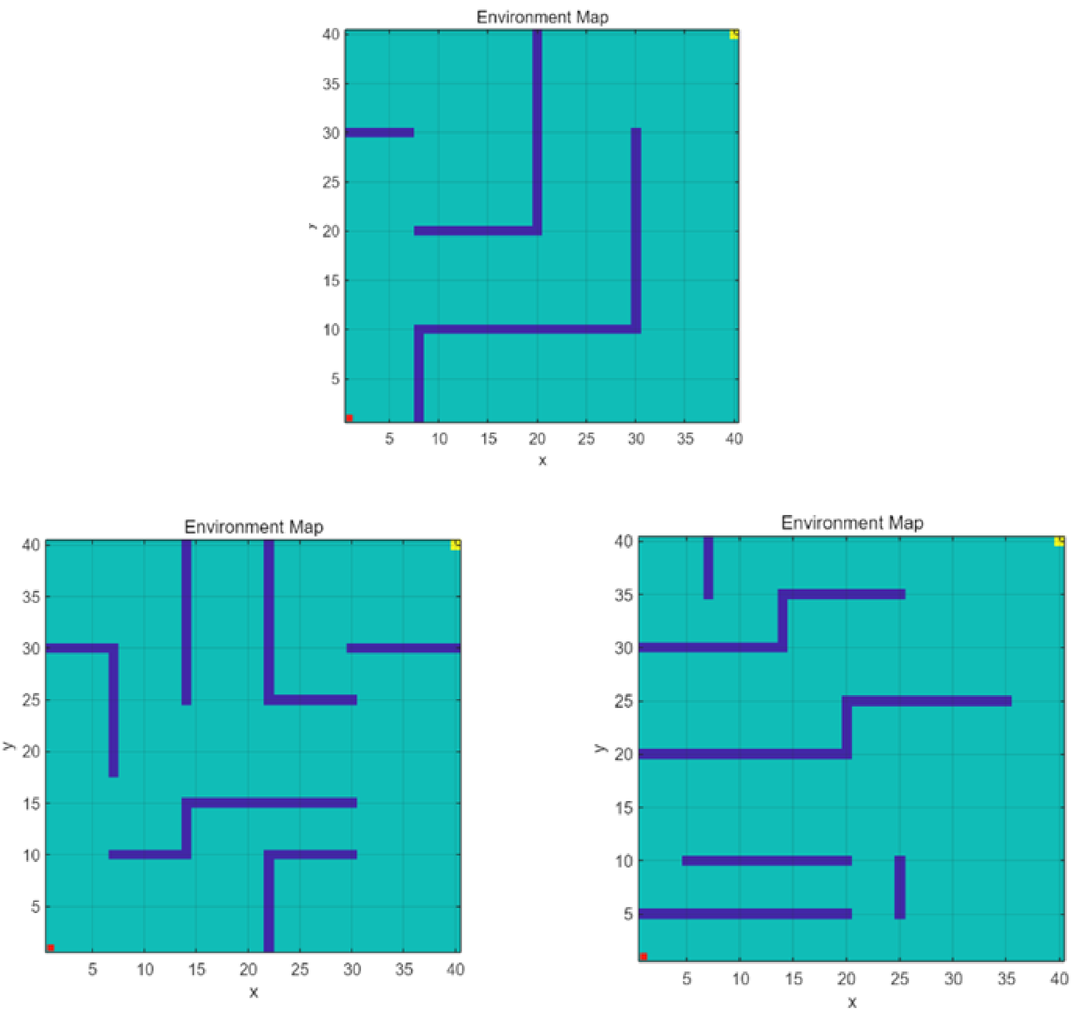

**Figure 2. (a)(b)(c).** Mazes with branches = 5,6,7.
(Picture Credit to: Original)

### *3.2. Parameter setting*

To investigate the influence of parameter variations and maze complexity on learning rates, the author conducted two distinct sets of experiments in mazes featuring different branching factors: 5, 6, and 7.

The experiments aimed to shed light on how changes in these parameters affected the speed of learning in the following manner:

In this initial group of experiments, the author maintained a constant γ value of 0.1 and observed how different α values (specifically, 0.1, 0.3, 0.5, 0.7, and 0.9) impacted the final learning times of agents. The goal was to ascertain how the choice of α influenced the rate at which the agents converged to a solution in the maze.

In the second group of experiments, the author set α to a fixed value of 0.5 and examined how various γ values (0.1, 0.3, 0.5, 0.7, and 0.9) affected the convergence values of the final learning times for the agents. This set of experiments aimed to reveal how altering the γ parameter affected the convergence rate and whether it differed from the previous set of experiments.

To ensure the reliability and generality of the experimental findings, each experiment was repeated three times. After collecting data from these repetitions, the author calculated the average values to establish the final dataset for analysis. This meticulous approach helped reduce the influence of potential outliers or random variations, thus ensuring the validity of the results.

The resulting data from these experiments were then used to create visual representations, such as graphs or plots, to illustrate the relationships between parameter changes, maze complexity, and learning rates. This systematic approach allowed the author to draw meaningful conclusions about how the selected parameters impacted the learning process in the context of maze navigation.

### *3.3. Experimental results*

The experimental findings are displayed in Figure 3 when learning rate is varied and discount factor is invariant.

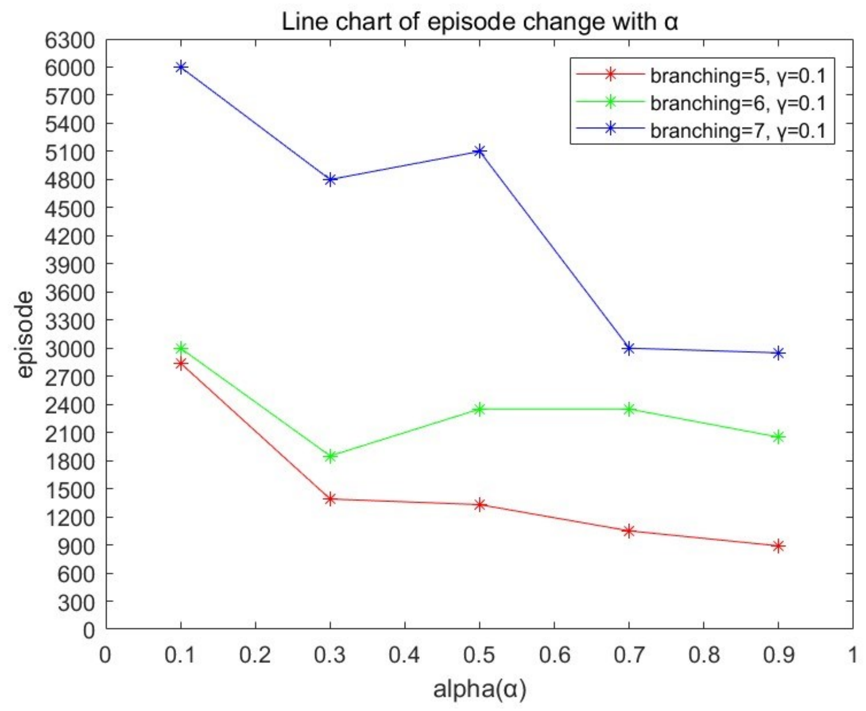

**Figure 3.** Line chart of episode change with α. (Picture Credit to: Original)

According to Figure 3, overall, as the α of agents increases, the number of times they learn decreases, indicating an improvement in learning efficiency. Meanwhile, as the complexity of the maze increases, the learning efficiency of the agent gradually decreases. But it is worth noting that, at α = 0.5, the learning efficiency decreased, which is not a simple monotonic function relationship. This is because when α gradually increases between 0 and 1, the agent may miss the optimal solution due to excessively high learning rates in some cases, thus increasing the time and frequency required for the entire learning process.

Figure 4 displays the experimental findings with the discount factor as a variable and the learning rate as an invariant.

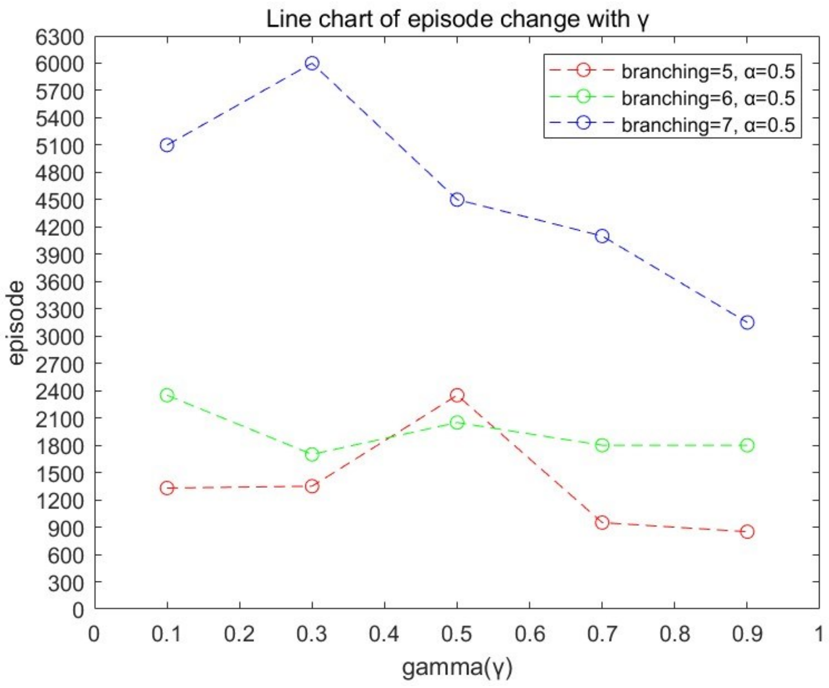

**Figure 4.** Line chart of episode change with γ. (Picture Credit to: Original)

According to Figure 5, overall, with the increase of γ, the learning frequency of the agent shows a trend of first increasing and then decreasing, indicating that the learning efficiency first decreases and then increases. Meanwhile, as the complexity of the maze increases, the learning efficiency of the agent also gradually decreases. Due to the increase in complexity of the maze, the values of γ at which the learning efficiency is at its lowest point are not the same. This is because as γ gradually increases between 0 and 1, the agent will consider long-term benefits more, leading to multiple collisions with obstacles for long-term benefits. So with the γ increasing, the size of the episode increases first and then decreases as it grows.

Judging from the results of the two sets of experiments, the relationship between the agent's learning rate and α and γ is not a simple monotonic function. But in general, the closer α and γ are to 1, the faster the learning rate of Q-learning. At the same time, it can be seen that the complexity of the environment plays a decisive role in the learning rate of Q-learning. The simpler the environment, the faster the learning rate of Q-learning. Therefore, we should simplify the problem as much as possible when modelling. At the same time, in specific problems, the parameters α and γ cannot be directly set to 1. As γ gradually increases between 0 and 1, the agent will consider long-term benefits more, leading to multiple collisions with obstacles to long-term benefits. So, with the γ increasing, the size of the episode increases first and then decreases as it grows. When α gradually increases between 0 and 1, the agent may miss the optimal solution due to excessively high learning rates in some cases, thus increasing the time and frequency required for the entire learning process. Therefore, when facing specific problems, it is still necessary to determine the best parameters through specific experiments.

## 4. Conclusion

This article focuses on the Q-learning method and attempts to identify the best parameters by using MATLAB to analyse the effects of the Q-learning parameters on robot route planning issues in various complicated situations. According to research, environmental complexity significantly affects how quickly robots can handle path planning difficulties, and the best solutions to problem parameters in various settings call for in-depth examination of issues. This article has certain reference significance for the navigation problems of robots in the real world. When modelling, minimizing the complexity of the environment is of great significance to solving practical problems. When dealing with specific problems in the future, more precise experiments can be conducted to find the optimal solution to the parameters of the specific problem, thereby greatly improving the work efficiency of the robot.

## References

[1] Khatib O 1985 IEEE Int. Conf. on Robotics and Automation vol **25e28** p 500e5
[2] Choset H and Burdick J 2000 *The International Journal of Robotics Research* vol **19(2)** pp 96-125
[3] Holland, J. H 1992 Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence
[4] Arulkumaran K, Deisenroth M P, Brundage M arXiv (Preprint arXiv 1708.05866, 2017)
[5] Sun C, Liu W and Dong L Reinforcement 2020 IEEE Transactions on Neural Networks and Learning Systems, vol **32(5)** pp 2054-2065
[6] Everett M, Chen Y F and How J P 2018 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS) pp 3052-3059
[7] Konar A, Chakraborty I G, Singh S J 2013 IEEE Transactions on Systems, Man, and Cybernetics: Systems vol **43(5)** pp 1141-1153.
[8] Even-Dar E, Mansour Y and Bartlett P 2003 *Journal of machine learning Research* vol 5(1).
[9] Zhou X 2022 *Journal of Physics: Conference Series* p 2386(1): 012037
[10] Gasparetto A, Boscariol P, Lanzutti A 2015 *Motion and Operation Planning of Robotic Systems: Background and Practical Approaches* pp 3-27