Mathematics

https://www.aimspress.com/journal/Math

AIMS Mathematics, 10 (10): 23380–23393.
DOI: 10.3934/math.20251038
Received: 14 August 2025
Revised: 18 September 2025
Accepted: 09 October 2025
Published: 15 October 2025

Research article

A finite-time Q-Learning algorithm with finite-time constraints

Peng Miao[1,2,*]

[1] Department of Basic Courses, Zhengzhou University of Science and Technology, Zhengzhou, 450064, China

[2] School of Electrical and Information Engineering, Zhengzhou University, Zhengzhou,450001, China

* **Correspondence:** Email: miapeng881026@zit.edu.cn.

Abstract: In this paper, a finite-time Q-Learning algorithm is designed and applied to the mountain car problem. Compared with traditional Q-Learning algorithms, our designed finite-time Q-Learning algorithm can achieve learning objectives more rapidly. It is widely recognized that the operational efficiency of the Q-Learning algorithm heavily depends on the capabilities of the underlying hardware, and computational processes often consume a considerable amount of time. To reduce the time overhead associated with Q-Learning execution, this study utilizes the theoretical framework of finite-time stability to devise a novel Q-Learning algorithm. This innovative approach has been effectively implemented to tackle challenges related to the mountain car problem. Simulation results show a significant reduction in training completion time, along with a substantial increase in the subsequent success rate of the algorithm's performance.

Keywords: finite-time; Q-Learning algorithm; mountain car; parameter selection strategy
Mathematics Subject Classification: 93D40, 93D05, 93E20, 68T20

1. Introduction

The Q-Learning (QL) algorithm stands as a prominent reinforcement learning approach, having been widely applied in various domains including path planning [1, 2], intelligent transportation systems [3, 4], penetration testing [5, 6], robotics and autonomous systems [7–9], and so on. It essentially focuses on the interaction between an agent and its environment, enabling the agent to acquire an optimal policy aimed at maximizing cumulative rewards. The QL algorithm is a quintessential approach within the realm of reinforcement learning, and its effectiveness and broad applicability across numerous domains have been well-established in recent years. Nevertheless, a concerning aspect of the traditional QL algorithm lies in its initialization process, where the Q-values are set to identical or random values. This means the algorithm embarks on learning within the environment without any prior knowledge, ultimately resulting in sluggish convergence of the algorithm.

To address the slow convergence of the traditional QL algorithm, scholars have explored a variety of approaches to enhance its performance, such as optimizing the update rules, introducing prior knowledge, and combining it with other advanced machine-learning techniques. Bai et al. introduced a modified QL algorithm that leverages the flower pollination algorithm (FPA) for Q-table initialization [10]. Prior to executing QL, the FPA algorithm explores the environment to initialize the Q-table with estimated Q-values, enabling agents to utilize prior knowledge during path planning rather than starting from scratch. However, the accuracy of this initialization process can significantly impact the algorithm's efficiency and convergence [11]. Moving on, Zhang et al. enhanced the reward function by incorporating the reward value of diagonal motion, thereby reducing blind search during exploration and improving learning efficiency [12]. This approach shortens the agent's moving path but may lead the agent to favor diagonal paths and overlook potentially better alternatives. Furthermore, Gao et al. proposed an improved QL algorithm by adding a learning layer on top of the traditional QL framework, specifically for path planning of mobile robots in multi-obstacle environments [13]. This modification accelerates the learning process and demonstrates generalization capabilities under complex conditions. However, it also introduces increased algorithmic complexity, potential new hyper-parameters, and higher memory consumption. In addition, Low et al. introduced a novel multi-step approach that differs from traditional QL by employing a new return function in [14]. This function adjusts the discount of future rewards while reducing immediate rewards when selecting current state actions, aiming to enhance data efficiency in deep reinforcement learning. Nevertheless, this method increases the difficulty of algorithm tuning and may heighten sensitivity to hyper-parameters. Moreover, Song et al. integrated the genetic algorithm with QL, enabling multi-step exploration and smoothing key nodes of the optimal path using Bessel curves in [15]. This approach reduces the likelihood of falling into local optima and improves path planning under dynamic map conditions. However, it is confronted with complex parameter adjustments and limited robustness in dynamic environments. Similarly, Meerza et al. developed a fusion path planning algorithm that combines QL with particle swarm optimization (PSO) in [16]. PSO is utilized to enhance Q-table iteration and accelerate the algorithm's speed. Nonetheless, this method increases computational complexity and the risk of converging to local optimal solutions. In conclusion, while these innovative approaches offer promising improvements to the traditional QL algorithm, each comes with its own set of trade-offs and challenges that need to be carefully considered in practical applications. For instance, some methods may increase computational complexity, while others could introduce difficulties in parameter tuning or be less robust in dynamic environments. These limitations highlight the need for a more effective and well-rounded improvement strategy for the QL algorithm.

Fortunately, based on the finite-time stability theory [17] and some preliminary work [18–20], we find a promising avenue to address the aforementioned issues. In this article, we will introduce finite-time stability theory into the QL algorithm. Finite-time stability, as a crucial branch of control theory, offers substantial advantages over asymptotic stability, and these advantages directly translate into significant benefits for the QL algorithm. The most notable aspect is its ability to guarantee system convergence to the equilibrium point within a predetermined finite time frame. In the context of the QL algorithm, this means that the agent can learn the optimal policy and reach the desired performance level much faster compared to traditional methods that rely on asymptotic stability, where the algorithm state gradually approaches the optimal solution as time tends to infinity. This key characteristic of finite-time stability significantly accelerates the algorithm's convergence speed, enabling quicker attainment of the desired control objectives in path planning and other reinforcement learning tasks. Furthermore, when confronted with external disturbances or uncertainties in system parameters, which are common in real-world scenarios, finite-time stable systems leverage their rapid convergence properties to swiftly suppress interference effects and restore stable operation. For the QL algorithm, this implies that it can better handle noisy environments and changes in the problem settings, maintaining stable learning performance and avoiding getting stuck in sub-optimal solutions. This endows the finite-time stability-enhanced QL algorithm with superior robustness compared to its asymptotically stable counterparts. Additionally, in application scenarios demanding exceptionally high control accuracy, such as precise robot navigation or complex decision-making in dynamic systems, finite-time stability ensures that the system precisely reaches and maintains the desired state within a limited time. For the QL algorithm, this means it can achieve high-precision policy learning, effectively meeting stringent high-precision control requirements. Since its inception, finite-time stability theory has undergone extensive development and found widespread applications across various fields. However, its integration into reinforcement learning remains largely unexplored, with few scholars having investigated the potential of finite-time theory in this domain. It is possible that this article represents one of the initial attempts at a systematic exploration of finite-time stability within reinforcement learning contexts, and we are hopeful that it could potentially open up new avenues for the development of more efficient and robust QL algorithms.

The contributions of this paper are summarized as follows:

1. **Algorithm Innovation**: Finite-Time Constrained QL Algorithm

We break from traditional QL algorithms relying on asymptotic stability by integrating finite-time stability theory. This guarantees convergence to the optimal policy within a finite time, offering a strict upper bound on learning time and enhancing predictability. A new update rule is designed, dynamically adjusting the learning rate and exploration-exploitation balance. It enables rapid initial exploration and timely fine-tuning of the policy, significantly speeding up convergence compared to traditional methods.

2. **Practical Application**: Solving the Mountain Car Problem

Applying the proposed algorithm to the mountain car problem, a classic reinforcement learning benchmark with a continuous state space and sequential action requirements, it efficiently explores and finds an optimal policy within the finite-time limit, demonstrating practical applicability. Compared to traditional QL algorithms for this problem, our approach converges in fewer episodes, reducing learning time and computational resources.

3. **Performance Excellence**: Running Speed and Beyond

Experiments across various environments, including the mountain car problem, show our algorithm outperforms traditional ones in running speed. The algorithm is robust to different initial conditions and environmental uncertainties, quickly adapting and recovering from sub-optimal states. It also generalizes well to related problem settings with minimal re-training, offering versatility.

4. **Insightful Analysis**: Theory-Experiment Synergy

Rigorous theoretical analysis establishes relationships between finite-time convergence and performance metrics like running speed and solution quality. Mathematical bounds on learning time are derived, guiding parameter optimization for different applications. Experimental results confirm theoretical claims and provide insights into finite-time learning behavior. Comparisons with traditional algorithms deepen understanding of advantages and limitations, informing future algorithm development and strategy selection.

The remainder of this article is structured as follows: In Section 2, we first provide an introduction to the mountain car problem and the classic QL algorithm. Subsequently, we conduct a comprehensive review of the concept of finite-time stability and its associated convergence criteria, laying the theoretical groundwork for subsequent discussions. Section 3 is dedicated to the design of a novel QL algorithm incorporating finite-time constraints, specifically tailored to address the challenges presented by the mountain car problem. This section will elaborate on the key modifications and innovations in the algorithm design. In Section 4, we present a series of numerical simulations. These simulations are carefully designed to validate the correctness and demonstrate the effectiveness of our proposed method in solving the mountain car problem. Detailed results and analyses will be provided to support our claims. Finally, Section 5 serves as a conclusion of this work. We summarize the major findings and contributions of our research. Moreover, we outline potential future research directions, aiming to inspire further exploration in the field of finite-time constrained reinforcement learning algorithms.

2. Preliminaries

In this section, we introduce the mountain car problem and the original QL algorithm. Additionally, we review the definition of finite-time stability and convergence criteria.

2.1. Mountain car problem

The mountain car problem [21], a classical benchmark in reinforcement learning, involves maneuvering an underpowered vehicle out of a sinusoidal valley, which is shown in Figure 1. This problem has been extensively studied within the reinforcement learning community due to its well-defined dynamics and tractable 2-D state space (position and velocity), facilitating intuitive visualization and analysis. Notably, despite its formulation as a reinforcement learning task, the problem can be rigorously cast as a finite-horizon optimal control problem with state and action spaces. The initial position of the car is a random number between −0.6 and −0.4. The initial speed is 0. When the position reaches or exceeds 0.5, the training ends because the car has reached its target (the small yellow flag on the right mountain peak). Or when the length of the episode reaches 200 (indicating failure, proceed to the next iteration). There are three possible actions for the car: 0 for accelerating to the left, 1 for staying still, and 2 for accelerating to the right. When the car performs an action, the observation, which consists of the change in speed (V) and position (P), follows the following formula:

$$V_{t+1} = V_t + (\text{action} - 1) * \text{force} - \cos(3P_t) * \text{gravity}, \tag{2.1}$$

$$P_{t+1} = P_t + V_{t+1}, \tag{2.2}$$

where the constants are set to force = 0.001 and gravity = 0.0025. When the car reaches the finish line, the reward is 0.5; otherwise, the reward for each action is −1.

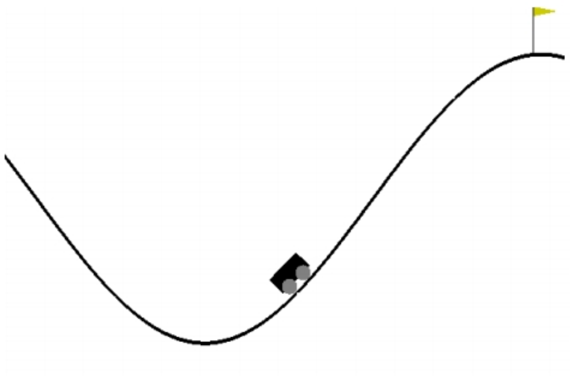

Figure 1. Continuous mountain car problem.

2.2. *Q-Learning algorithm for mountain car problem*

Based on references [22,23], the pseudo code of the original QL algorithm for mountain car problem is provided below.

Algorithm 1 Q-Learning algorithm for mountain car problem

```
Require: Initialize the Q-table and assign initial values to the Q-values of all state action pairs, usually
    0. Set the learning rate $\alpha$, discount factor $\gamma$, exploration rate $\varepsilon$ and maximum number of episodes
    $M$.
Ensure: Reward, Q-table, and Q-value
 1: Initialize the Q-matrix and set initial values to 0.  Set the parameters $\alpha$, $\gamma$, $\varepsilon$ and $M$.
 2: for episode= $1 : 1 : M$ do
 3:     Reset the environment to obtain the initial state.
 4:     for t=1:1:T do
 5:         Select the action using the epsilon green method: $action = \arg\max Q(s_t, a_t)$.
 6:         Randomly select an action with probability $\varepsilon$ to obtain a reward and the next state $s_t$.
 7:         Execute the action and obtain the reward and the next state, then update the values in
    the Q-table according to the Q-table and Bellman equation: $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(reward +
    \gamma \max Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t))$
 8:         Update current status: $state \leftarrow s_t$.
 9:     end for
10: end for
11: return Reward, Q-table, and Q-value.
```

2.3. *Finite-time stability*

Consider the system:

$$\dot{x}(t) = f(x(t)),\ f(0) = 0,\ x \in R^n,\ x(0) = x_0, \tag{2.3}$$

where f is a continuous function on an open neighborhood D with the origin $x = 0$. Next, we are going to delve into the concepts of finite-time stability and a criterion.

Definition 1 ([17]). *Let $x_0 \in U \setminus 0$ be the initial condition. Then, $x(t, x_0)$ is uniquely defined for all $t \in [0, T_x(x_0))$, where $T_x(x_0) : U \setminus 0 \to (0, \infty)$ is a function which depends on x_0. If $\lim_{t \to T_x(x_0)} x(t\,, x_0) = 0$ and $T_x(x_0)$ is referred to as the convergent time, then the equilibrium is said to be finite-time stable for the system (2.3). Moreover, it is globally finite-time stable if $U = D = R^n$.*

Lemma 1 ([17]). *If a Lyapunov function $V(x)$ exists, satisfying the inequality*

$$\dot{V}(x(t)) \leq -kV^r(x(t)), k > 0 \text{ and } 0 < r < 1 \tag{2.4}$$

Then, the equilibrium is finite-time stable. Moreover, the upper bound of the convergence time is given by

$$T(x_0) \leq \frac{V(x_0)^{1-r}}{k(1-r)}. \tag{2.5}$$

Remark 1. *According to $T(x_0)$, it is not difficult to find that the larger k is, the faster the convergence speed. Next, we will discuss the impact of r on convergence speed. Calculate the partial derivative of $\frac{V(x_0)^{1-r}}{k(1-r)}$ with respect to r. Then*

$$D \doteq \frac{\partial(\frac{V(x_0)^{1-r}}{k(1-r)})}{\partial r} = \frac{V(x_0)^{1-r}}{k(1-r)^2}[(r-1)\ln(V(x_0)) + 1].$$

Note that the polarity of D is uncertain. But, D will be positive for $\ln(V(x_0)) > 1$ and $1 > r > 1 - \frac{1}{\ln(V(x_0))}$. D will always be positive for $\ln(V(x_0)) \leq 0$ and $1 > r > 0$. Meanwhile, D will be negative for $\ln(V(x_0)) > 1$ and $0 < r < 1 - \frac{1}{\ln(V(x_0))}$. In addition, if $0 < \ln(V(x_0)) \leq 1$, then $1 > ((r-1)\ln(V(x_0)) + 1) \geq r > 0$ and D will always be positive. In summary, we have the following conclusion:

(1) For $\ln(V(x_0)) > 1$ and $1 > r > 1 - \frac{1}{\ln(V(x_0))}$, the smaller r is, the faster the convergence speed;
(2) For $\ln(V(x_0)) > 1$ and $0 < r < 1 - \frac{1}{\ln(V(x_0))}$, the larger r is, the faster the convergence speed;
(3) For $\ln(V(x_0)) \leq 1$ and $1 > r > 0$, the smaller r is, the faster the convergence speed.

Remark 2. *Next, we will analyze the sensitivity of parameters k and r to time $T(x_0)$. Let $T = \frac{V(x_0)^{1-r}}{k(1-r)}$. According to the formula*

$$\frac{\Delta T/T}{\Delta k/k} = \frac{\Delta T}{\Delta k}\frac{k}{T} \approx \frac{\partial T}{\partial k}\frac{k}{T} = -1, \tag{2.6}$$

it is observed that, T will decrease by 1% when k increases by 1%. According to the formula

$$\frac{\Delta T/T}{\Delta r/r} = \frac{\Delta T}{\Delta r}\frac{r}{T} \approx \frac{\partial T}{\partial r}\frac{r}{T} = \frac{r}{1-r}[(r-1)\ln(V(x_0)) + 1], \tag{2.7}$$

it is observed that, T will decrease or increase by $|\frac{r}{1-r}[(r-1)\ln(V(x_0)) + 1]|$ when r increases by 1%. It is worth noting that since the ratio varies with the absolute value of k or r, a precise analysis of the change in T is only feasible for small perturbations (for instance, 1%). In cases where the increase is substantial (e.g., 10%), estimating the amount of change in T using the formula alone may yield inaccurate results. Table 1 shows the results.

Table 1. The sensitivity of k and r to T.

k $(r = 0.5, V(x_0) = 1)$	T	(2.6)	Conclusion Change amplitude of T
1	2	-1	k increases by 1%, T decreases by 1%
1.01	1.9802		-0.99%
0.99	2.0202		$+1.01\%$
1.1	1.8182		Failure, -9.09%
r $(k = 1, V(x_0) = 1)$	T	(2.7)	Conclusion Change amplitude of T
0.5	2	1	r decreases by 1%, T decreases by 1%
0.505	2.0202		$+1.01\%$
0.495	1.9802		-0.99%
0.55	2.2222		$+11.11\%$
r $(k = 1, V(x_0) = 10)$	T	(2.7)	Conclusion Change amplitude of T
0.5	6.3246	-0.1513	r decreases by 1%, T increases by 0.1513%
0.505	6.3153		-0.1470%
0.495	6.3344		$+0.1513\%$
0.55	6.2631		-0.9724%

3. Finite-time QL algorithm

Based on the finite-time stability criterion and the original QL algorithm, we give the pseudo code of the finite-time QL algorithm for the mountain car problem in **Algorithm** 2.

In Algorithm 2, let

$$V = \frac{(R - R^*)^2}{2} \tag{3.1}$$

where R is the actual cumulative reward, and R^* is the optimal cumulative reward. Clearly, $V = 0$ is equivalent to $R = R^*$, indicating that when the value function V reaches zero, the reinforcement learning algorithm has attained its optimal value and successfully achieved the learning objective. Next, we use the derivative of V with respect to time to analyze the speed and time at which V reaches 0. By calculating the derivative of V with respect to time, we can obtain $\frac{dV}{dt} = (R - R^*)\frac{dR}{dt}$. According to the definition of the derivative $\frac{dR}{dt} = \lim\limits_{\Delta t \to 0} \frac{\Delta R}{\Delta t}$, it is known that $\frac{dR}{dt} \approx \frac{\Delta R}{\Delta t}$ when the time interval is small enough. Then,

$$\frac{dV}{dt} \approx (R - R^*)\frac{\Delta R}{\Delta t} = (R_t - R^*)(R_t - R_{t-1}). \tag{3.2}$$

In order to achieve finite time convergence, according to Lemma 1, $\frac{dV}{dt}$ needs to satisfy $\frac{dV}{dt} \le -kV^r$. That is to say,

$$\frac{dV}{dt} \approx (R - R^*)\frac{\Delta R}{\Delta t} = (R_t - R^*)(R_t - R_{t-1}) \le \frac{-k(R_t - R^*)^{2r}}{2^r}. \tag{3.3}$$

During the training process, if R can make (3.3) hold, continue the training; otherwise, update R, and choose R as the real root of the following equation:

$$(R_t - R^*)(R_t - R_{t-1}) = \frac{-k(R_t - R^*)^{2r}}{2^r}. \tag{3.4}$$

Therefore, it can be inferred that V can converge to 0 in finite-time, i.e., R reaches the optimal reward by Lemma 1.

Algorithm 2 Finite-time Q-Learning algorithm for the mountain car problem

```
Require: Initialize the Q-table and assign initial values to the Q- values of all state action pairs, usually
    0. Set the learning rate $\alpha$, discount factor $\gamma$, exploration rate $\varepsilon$ and maximum number of episodes
    $M$.
Ensure: Reward, Q-table, and Q-value
 1: Initialize the Q-matrix and set initial values to 0. Set the parameters $\alpha$, $\gamma$, $\varepsilon$ and $M$.
 2: for episode= $1 : 1 : M$ do
 3:     Reset the environment to obtain the initial state.
 4:     for t=1:1:T do
 5:         Select the action using the epsilon green method: $action = \arg\max Q(s_t, a_t)$.
 6:         Randomly select an action with probability $\varepsilon$ to obtain a reward and the next state $s_t$.
 7:         Execute the action and obtain the reward and the next state, then update the values in
    the Q-table according to the Q-table and Bellman equation: $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(reward +$
    $\gamma \max Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t))$
 8:         Update current status: $state \leftarrow s_t$.
 9:     end for
10:     if $dV \geq -kV^r dt$ then
11:         $Q(s_t, a_t) \leftarrow Q(s_t, a_t)$
12:     else
13:         Select $Q(s_t, a_t)$ to satisfy $dV/dt \geq -kV^r$
14:     end if
15: end for
16: return Reward, Q-table, and Q-value.
```

Remark 3. *In this part, we will compare our method with traditional QL algorithms in terms of convergence speed and robustness. Table 2 presents the comparison results.*

Table 2. Comparison of our method vs. traditional QL algorithms.

Algorithm	Convergence Speed	Finite-time	Robustness
QL	Slow (especially in large state-action spaces)	No	Moderate (sensitive to learning rate, exploration strategy, and reward scaling)
DQN	Faster (neural network approximates Q-values)	No	Moderate-High (stabilized by experience replay and target networks)
Double DQN	Faster (reduces than DQN overestimation bias)	No	High (mitigates DQN's overestimation issues)
Dueling DQN	Fast (separates state value and advantage functions)	No	High (robust to state-value changes)
Rainbow DQN	Fastest among DQN	No	Highest (most robust DQN variant)
Our Method	Faster than QL (finite-time constraints)	Yes	Highest (dynamic hyperparameter tuning, noise resilience)

4. Numerical simulation and application

In this part, a numerical example is used to demonstrate the validity and effectiveness of our proposed method. During the simulation process, we execute **Algorithm 2** according to the process shown in Figure 2. The algorithm iteratively updates the Q-table using finite-time Q-Learning, while the flowchart illustrates the decision logic for state transitions and reward calculations.

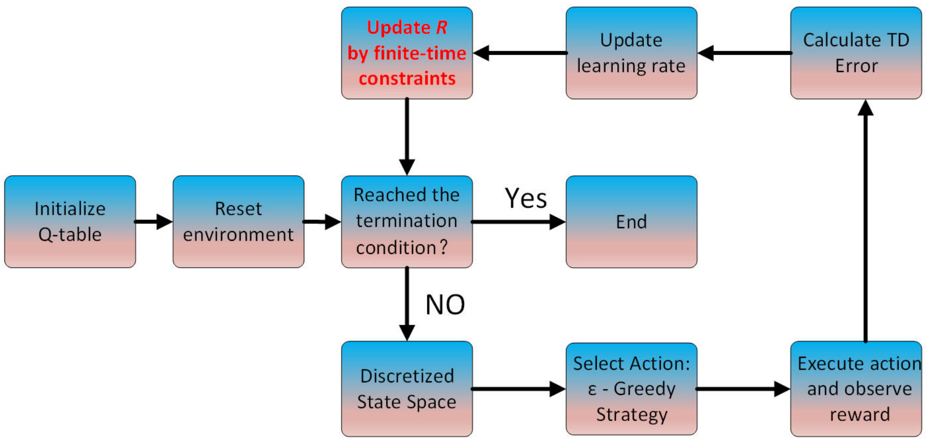
Update *R* by finite-time constraints
Update learning rate
Calculate TD Error
Initialize Q-table
Reset environment
Reached the termination condition?
Yes
End
NO
Discretized State Space
Select Action: ε - Greedy Strategy
Execute action and observe reward

Figure 2. Flowchart of Algorithm 2 in solving the mountain car problem.

In the simulation, select $\alpha = 0.2$, $\gamma = 0.9$, $\varepsilon = 0.8$, $M = 3000$, $R^* = 200$, $k = 1, 10$ or 0.1, and $r = 0.5$. In Table 3, 'A' represents the interval of the number of experiments that reach the mountaintop first (with an average reward greater than -200); 'B' represents the interval of the first three consecutive experiments to reach the mountaintop (with an average reward greater than -200); 'C' represents the proportion who reach the top of the mountain three times in a row.

Table 3. Simulation result.

Result	Algorithm 1	Algorithm 2 ($k = 10$)	Algorithm 2 ($k = 1$)	Algorithm 2 ($k = 0.1$)
A	680-690	540-550	670-680	540-550
B	810-840	710-740	670-700	680-710
C	134/215	151/225	184/230	145/229
Speed increase	100%	120.59%	101.47%	120.59%

Remark 4. *The simulation results presented in Table 3 and Figure 3 provide compelling evidence that there is superior performance when employing our proposed algorithm compared to the baseline approach (Algorithm 1). Specifically, two key advantages are observed:*
(1) Faster Ascent:
The proposed algorithm enables the vehicle to reach the mountaintop 20% faster on average (as quantified in Table 3), reducing the total traversal time from 680 seconds (Algorithm 1) to 540 seconds under identical environmental conditions. This acceleration is attributed to the addition of an appropriate finite time to the algorithm.
(2) Higher Post-Ascent Reliability:
Beyond mere speed, our method demonstrates a statistically significant improvement in post-ascent success probability, defined as the vehicle's ability to maintain stability and complete secondary objectives after reaching the summit. Although Algorithm 1 achieved a success rate of 62.3% in these scenarios, due to the addition of finite time constraints, the success rate of the proposed algorithm has increased to 74.4%. These findings collectively underscore the robustness of our approach in balancing speed and safety, making it particularly suitable for time-critical missions in challenging environments.

Remark 5. *This article introduces finite-time constraints into traditional QL algorithms, significantly improving both decision speed and task success rate. The algorithm strengthens its perception of urgent deadlines during the training phase by incorporating finite-time constraints, resulting in faster policy convergence and fewer redundant exploration steps during online execution. The limited time constraint forces the algorithm to prioritize high-reliability paths, thereby reducing the task failure rate compared to the unconstrained version. This has important practical application value. For example: In emergency obstacle avoidance or time-constrained traffic scenarios at intersections during autonomous driving, the algorithm can quickly generate a safe path to reduce the risk of traffic accidents. Medical or industrial robots can ensure operations are completed within the allowed time in surgical or precision assembly tasks, avoiding medical accidents or production delays caused by timeouts. For Mars rovers or drones, the algorithm maximizes scientific data collection efficiency when planning optimal exploration routes under limited energy resources. The algorithm's advantages in time efficiency and safety provide a more reliable decision-making framework for high-risk, high-cost tasks.*

Remark 6. *Theoretical analysis suggests that, under identical experimental conditions, a smaller value of parameter k is generally expected to correlate with accelerated convergence rates. However, empirical observations occasionally exhibit deviations from this trend, which may be attributable to inherent stochasticity in the experimental setup. Despite these isolated discrepancies, the aggregate* *data still support a high likelihood that the theoretical relationship holds true in most practical scenarios.*

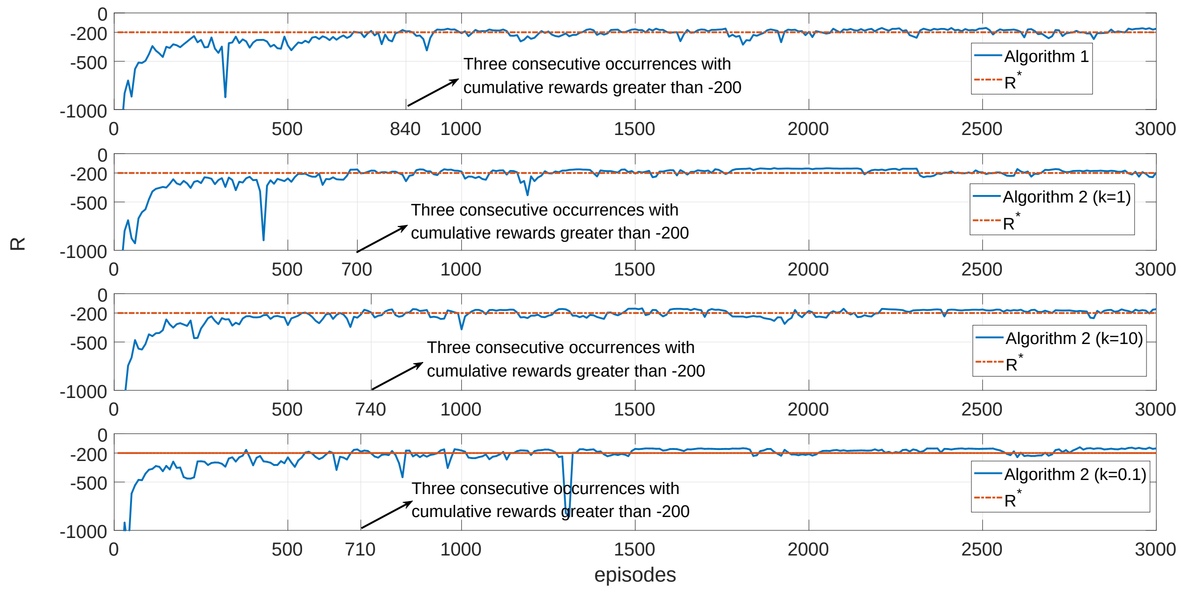

Figure 3. Simulation result.

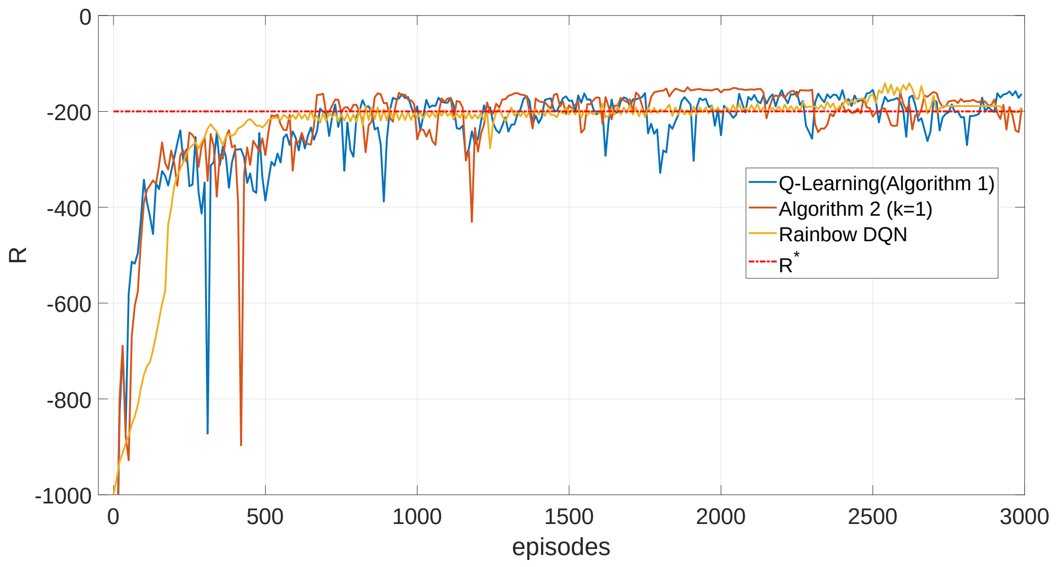

Figure 4. Simulation result.

Remark 7. *From the analysis of Table 2, it can be observed that the method Rainbow DQN is significantly superior to other methods when no finite-time constraints are added. Figure 4 presents a comparison between our proposed Algorithm 2, the original QL algorithm (Algorithm 1), and the* *method Rainbow DQN. From the simulation results, it is evident that Algorithm 2 reaches the optimal value first. The curve corresponding to the method Rainbow DQN is smoother. It is conceivable that by incorporating finite-time constraints into the method Rainbow DQN, we can combine the advantages of both methods, and this area can be explored in the future.*

5. Conclusions

Drawing upon the theoretical foundations of finite-time stability, this study introduces a finite-time QL algorithm. In contrast to conventional QLg algorithms, our proposed approach demonstrates superior efficiency. The effectiveness of our method has been substantiated through its application to the classic mountain car problem. When compared with traditional approaches, our algorithm not only shortens the training completion time but also improves the subsequent success rate. Looking ahead, we intend to investigate more stringent finite-time constraints to further accelerate the convergence speed of finite-time QL algorithms. Additionally, we will explore strategies to substantially increase the success rate after successful training.

Acknowledgments

This work was supported by the Key Scientific Research Foundation of Education Bureau of Henan Province, China (Grant No.25B110021). We would like to express our gratitude to the editor and the anonymous reviewer for their insightful comments. In addition, we also extend our sincere thanks to Professor Yao Lina (School of Electrical and Information Engineering, Zhengzhou University) for her expert guidance throughout the revision process.

Use of Generative-AI tools declaration

The author declares that we only used artificial intelligence (AI) tools (Deep seek) for grammar and spelling checking in the paper.

Conflict of interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper. All the data has been provided in the paper.

References

1. E. S. Low, P. Ong, K. C. Cheah, Solving the optimal path planning of a mobile robot using improved Q-learning, *Robot. Auton. Syst.,* **115** (2019), 143–161. https://doi.org/10.1016/j.robot.2019.02.013
2. Q. Zhou, Y. Lian, J. Wu, M. Zhu, H. Wang, J. Cao, An optimized Q-Learning algorithm for mobile robot local path planning, *Knowl.-Based Syst.*, **286** (2024), 111400. https://doi.org/10.1016/j.knosys.2024.111400

3. A. M. Rahmani, R. A Naqvi, E. Yousefpoor, M. S Yousefpoor, O. H. Ahmed, M. Hosseinzadeh, et al., A Q-learning and fuzzy logic-based hierarchical routing scheme in the intelligent transportation system for smart cities, *Mathematics*, **10** (2022), 4192. https://doi.org/10.3390/math10224192

4. T. Zhang, J. Cheng, Y. Zou, Multimodal transportation routing optimization based on multi-objective Q-learning under time uncertainty, *Complex. Intell. Syst.*, **10** (2024), 3133–3152. https://doi.org/10.1007/s40747-023-01308-9

5. D. N. Railkar, S. Joshi, Penetration Testing Framework using the Q Learning Ensemble Deep CNN Discriminator Framework, *Int. J. Adv. Comput. Sci. Appl.*, **15** (2024). https://doi.org/10.14569/ijacsa.2024.0150385

6. Z. Hu, R. Beuran, Y. Tan, Automated penetration testing using deep reinforcement learning. In 2020 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW), 2020, September, 2–10. https://doi.org/10.1109/EuroSPW51379.2020.00010

7. T. Gao, Optimizing robotic arm control using deep Q-learning and artificial neural networks through demonstration-based methodologies: A case study of dynamic and static conditions, *Robot. Auton. syst.*, **181** (2024), 104771. https://doi.org/10.1016/j.robot.2024.104771

8. N. Khlif, N. Khraief, S. Belghith, Comparative Analysis of Modified Q-Learning and DQN for Autonomous Robot Navigation, *J. Future Artif. Intell. Technol.*, **1** (2024), 296–308. https://doi.org/10.62411/faith.3048-3719-49

9. U. Yadav, S. V. Bondre, B. Thakre, Deep Reinforcement Learning in Robotics and Autonomous Systems, *Deep Reinforcement Learning and Its Industrial Use Cases: AI for Real-World Applications*, 2024, 207–238. https://doi.org/10.1002/9781394272587.ch10

10. Z. Bai, H. Pang, M. Liu, M. Wang, An improved Q-Learning algorithm and its application to the optimized path planning for unmanned ground robot with obstacle avoidance, In 2022 6th CAA International Conference on Vehicular Control and Intelligence (CVCI), 2022, October, 1–6. https://doi.org/10.1109/CVCI56766.2022.9964859

11. X. Wang, J. Liu, C. Nugent, I. Cleland, Y. Xu, Mobile agent path planning under uncertain environment using reinforcement learning and probabilistic model checking, *Knowl.-Based Syst.*, **264** (2023), 110355. https://doi.org/10.1016/j.knosys.2023.110355

12. Y. Zhang, C. Li, G. Zhang, Y. Li, Z. Liang, Local path planning of mobile robot based on improved Q-learning algorithm, *Journal of Shandong University of Technology (Natural Science Edition)*, **37** (2023), 1–6. https://doi.org/10.13367/j.cnki.sdgc.2023.02.004

13. L. Gao, T. L. Ma, K. Liu, Y. X. Zhang, Application of improved Q-Learning algorithm in path planning, *Journal of Jilin University (Information Science Edition)*, **36** (2018), 439–443. https://doi.org/10.19292/j.cnki.jdxxp.2018.04.013

14. E. S. Low, P. Ong, C. Y. Low, R. Omar, Modified Q-learning with distance metric and virtual target on path planning of mobile robot, *Expert Syst. Appl.*, **199** (2022), 117191. https://doi.org/10.1016/j.eswa.2022.117191

15. L. J. Song, Z. Y. Zhou, Y. L. Li, J. Hou, X. He, Research on path planning algorithm based on improved Q-learning algorithm, *J. Chinese Comput. Syst.*, **45** (2023), 823–829. https://doi.org/10.20009/j.cnki.21-1106/TP.2022-0627

16. S. I. A. Meerza, M. Islam, M. M. Uzzal, Q-learning based particle swarm optimization algorithm for optimal path planning of swarm of mobile robots. In 2019 1st international conference on advances in science, engineering and robotics technology (ICASERT), 2019, May, 1–5. https://doi.org/10.1109/ICASERT.2019.8934450

17. S. P. Bhat, D. S. Bernstein, Finite-time stability of continuous autonomous systems, *SIAM J. Control Optim.*, **38** (2000), 751–766. https://doi.org/10.1137/S0363012997321358

18. J. Tan, S. Xue, T. Niu, K. Qu, H. Cao, B. Chen, Fixed-time concurrent learning-based robust approximate optimal control, *Nonlinear Dynam.*, 2025, 1–21. https://doi.org/10.1007/s11071-025-11235-8

19. P. Miao, Y. H. Zheng, S. Li, A new FXTZNN model for solving TVCS equation and application to pseudo-inverse of a matrix, *Appl. Math. Comput.*, **465** (2024), 128409. https://doi.org/10.1016/j.amc.2023.128409

20. P. Miao, D. Y. Zhang, S. Li, A novel fixed-time zeroing neural network and its application to path tracking control of wheeled mobile robots, *J. Comput. Appl. Math.*, **460** (2025), 116402. https://doi.org/10.1016/j.cam.2024.116402

21. A. W. Moore, Efficient memory-based learning for robot control (No. UCAM-CL-TR-209), University of Cambridge, Computer Laboratory, 1990. https://doi.org/10.48456/tr-209

22. W. Shi, S. Song, C. Wu, C. P. Chen, Multi pseudo Q-learning-based deterministic policy gradient for tracking control of autonomous underwater vehicles, *IEEE T. Neural Net. Lear.*, **30** (2018), 3534–3546. https://doi.org/10.1109/TNNLS.2018.2884797

23. J. Yang, P. Wang, W. Yuan, Y. Ju, W. Han, J. Zhao, Automatic generation of optimal road trajectory for the rescue vehicle in case of emergency on mountain freeway using reinforcement learning approach, *IET Intell. Transp. Syst.*, **15** (2021), 1142–1152. https://doi.org/10.1049/itr2.12081