Engineering and Technology Journal
Journal homepage: https://etj.uotechnology.edu.iq

Robot path planning using enhanced Q-learning algorithm based on single parameter

Noor H. Fallooh[a*], **Ahmed T. Sadiq**[b], **Eyad I. Abbas**[a], **Ivan A. Hashim**[a]

[a] Electrical Engineering Dept., University of Technology-Iraq, Alsina'a street, 10066 Baghdad, Iraq.
[b] Computer Science Dept., University of Technology-Iraq, Alsina'a street, 10066 Baghdad, Iraq.
*Corresponding author Email: eee.21.14@grad.uotechnology.edu.iq

HIGHLIGHTS

- The Q-learning algorithm's output was optimized to reduce time consumption
- Parameters were selected to enable the best possible path planning through a modified approach
- The modified Q-learning algorithm relies on the (1-α) parameter instead of the (γ) parameter
- Learning efficiency was enhanced by using priority trial replay to reach the target position
- The shortest distance between two points was represented by movement in an inclined direction

Keywords:
Reinforcement Learning, Q-Learning Algorithm, Robot Path Planning, Learning Rate (α), Discount Factor (γ).

ABSTRACT

One of the challenging aspects of robot navigation is path planning in a dynamic environment. The Q-learning algorithm is one of the reinforcement learning techniques that can be applied to the path planning of a mobile robot. The vital algorithm for any intelligent mobile robot is path planning. On the other hand, the traditional Q-learning method examines every conceivable state of the robot to choose the optimal path. As a result, this method is very computationally intensive, especially when there is a need to compute a large environment. This study proposes a modified version of the technique for planning robot paths. Using the learning rate (1-α) instead of the certification discount factor (γ), the algorithm became completely dependent on the reliance parameters, making it one of those that depend on a single parameter. This reliance can reduce the number of parameters and increase the algorithm's execution efficiency. A modified version of Q-learning was investigated with to determine the optimal path planning in several dynamic obstacle environments. Learning efficiency was enhanced by using priority trial replay in the improved Inclined Eight Connection Q-learning Algorithm (I8QA). A simulated environment was used for the suggested method, and it was shown that it can successfully plan optimal paths in dynamic obstacle environments. Overall, Q-learning, a strong and adaptable reinforcement learning method, is utilized for dealing with a wide range of problems. The improvement ratio of path length in the experiment environment is 40.812%, indicating that the I8QA algorithm is more compatible with dynamic environments.

1. Introduction

Robotics is a suitable foundation for teaching the basics of control engineering, for example, electronics, mechanics, programming, control, etc. [1-5]. Robots illustrate a closed control loop perfectly, as they are autonomous machines with sensors, controllers, and actuators. Several institutions employ robots to instruct fundamental control concepts [6]. A mobile robot is a machine that can navigate in its environment [7]. It is controlled by software that uses sensors and other technology to recognize its surroundings and move in a safe and controlled manner. Various sources, including batteries, fuel cells, solar panels, and power mobile robots. They usually have an array of sensors, such as cameras, lasers, and radar, to help them navigate their environment [8]. They also have a variety of actuators, such as tracks, legs, and wheels, to allow them to move [4]. The benefits of using mobile robots include increased productivity, improved safety, reduced costs, and improved quality [9]. Robot path planning is finding a safe and efficient path for a robot to move from one point to another in its environment [2]. The environment can be represented as a map representing the robot's surroundings. The map can be known or unknown to the robot [10]. There are many different path-planning algorithms, each with strengths and weaknesses [11]. Some common path-planning algorithms include A*, rapidly exploring random trees (RRT), and Grid-based path planning [3]. Depending on the specific application, a particular path-planning method is chosen. A* search is a suitable option, for instance, when the robot needs to find the quickest route in a known environment. RRT is suitable if the robot wants to swiftly discover a path through a huge or complex area [12,13]. Grid-based path planning is viable if the surroundings can be visualized as a grid [14]. Other elements, such as the

http://doi.org/10.30684/etj.2024.154230.1831
Received 08 October 2024; Received in revised form 17 December 2024; Accepted 19 December 2024; Available online 27 January 2025
2412-0758/University of Technology-Iraq, Baghdad, Iraq

precision of the map, the robot's speed, and the environment's complexity, can also impact how well path-planning algorithms operate for robots [15]. This is why many colleges employ robots to instruct on fundamental control concepts [1]. Robot path planning is a challenging but crucial issue in robotics [16]. The demand for precise and effective path-planning algorithms will increase as robots become more advanced and are used in more applications [17]. The process of determining a safe and effective path for a robot from its current location to the destination location is known as route planning. It is a component of robots used in healthcare, transportation, and industry [18]. The difficulties in planning robot paths include uncertainty: The surroundings in which the robot is traveling may be unpredictable. This might be caused by unforeseen impediments, moving items, or long-term environmental changes. Complexity: The environment in which the robot operates may be complex. This could result from numerous hurdles, constrained spaces, or challenging obstacles. Real-time limitations: the robot needs to have real-time path planning capabilities. This becomes particularly difficult in dynamic contexts where impediments may shift or change [19-21]. The contribution of this paper is to reduce the time consumption and adjust the Q-learning algorithm's output depending on the parameters selected for the best possible path planning by presenting a modified one. Our modified Q-learning is based on replacing relying (1-α) instead of the (γ) parameter. Learning efficiency was increased using priority trial replay to reach the target position. Additionally, the shortest distance between two places is represented by movement in the inclined direction with the smallest angle. Low et al. [10], demonstrated that when Q-values are initialized properly using the FPA, an expedited convergence of Q-learning can be obtained by testing the proposed enhanced Q-learning practically in a challenging setting with a special configuration of obstacles. A three-wheeled mobile robot is used in a real-world experiment to verify the efficiency of the suggested algorithm. By integrating prior knowledge obtained from the FPA into classical Q-learning, simulation results demonstrate that initialization of the Q-values serves as a solid foundation for exploration, thereby accelerating the learning process of the mobile robot. Guoa [11], has proposed an algorithm that speeds up convergence by adding a dynamic reward function to improve the initial Q schedule. This approach provides knowledge and experience through the case-based inference (CBR) algorithm and prevents entry into the encircled areas using an obstacle avoidance method. The experimental results show that the pathfinding performance of the modified Q-learning algorithm is much better than the original. Lee and Jeong [12], identified two fundamental warehouse settings with automated logistics where mobile robot paths are optimized using reinforcement learning techniques and algorithms to determine the rules required for path planning. The algorithms were tested in the same experimental environment and under the same conditions using a mobile robot's path optimization simulation. The experiment's findings assisted in understanding certain characteristics and changes of the reinforcement learning algorithm. The results of this research will assist in better understanding the fundamental ideas of reinforcement learning so that we can develop more realistic and complex path optimization algorithms and reinforcement learning skills with a single agent and a single path. The field environment in the warehouse, however, is much more complex and variable. Sang et al. [13], stated that there is a trend in uncrewed surface vehicles (USVs) to deploy several USVs in a fleet formation. The USV fleet must be navigated efficiently, and formation path planning algorithms are crucial for producing the best trajectories and offering workable collision avoidance tactics. The multiple sub-target artificial potential field (MTAPF), based on an enhanced APF, is a ground-breaking deterministic algorithm that ensures the formation trajectories' rationality, optimality, and path continuity. The global optimal path produced by an enhanced heuristic A* algorithm is what the MTAPF, which is a part of the local path planning algorithm. Maoudj and Hentout [14], obtained an Efficient Q-Learning (EQL) algorithm that can overcome these drawbacks and guarantee an optimal collision-free path in the shortest time. In the QL algorithm, the design of an efficient reward function and an efficient selection strategy for an optimal action that guarantees exploration and exploitation are critical components of successful learning. To this end, a new reward function is proposed to initialize the Q-table and give the robot background information about the environment. Next, a new efficient selection strategy is proposed to speed up the learning process by reducing search space and guaranteeing a quick convergence toward an optimized solution. Ma et al. [15], proposed a continuous local search Q-Learning (CLSQL) approach, which was introduced to solve these problems and ensure the standards of the intended learning path. The first step involves gradually dividing the global environment into separate local habitats. Then, using prior knowledge, the intermediate spots are searched for in each local area. The search is then between each intermediate place and the final destination. Finally, the suggested method ensures the best path while accelerating convergence and decreasing computing time compared to RL-based algorithms. Qi Jiang [16], searched and developed a path-planning method based on a Q-learning algorithm to address the path-planning problem for mobile robots. This method requires only the interaction between the current state and the environment to determine rewards and penalties for the robot’s actions and decide the next course of action. The Q-learning algorithm was enhanced to address the issues of poor efficiency and slow convergence in the original approach, permitting the robot to quickly complete its planning and choose the quickest and most efficient route. Using the grid method, the environment necessary to run the program and demonstrate the convergence process while gathering data was set up. Bonny and Kashkash [17], presented a novel, state-of-the-art strategy for mapping out the optimal path for a mobile robot in a two-dimensional space. The optimal path is found by combining the Q-Learning Algorithm with the Bees Algorithm (BA). The beginning population should be created using a creative method that can be used regardless of the number and location of environmental obstacles. The local search function of the BA is implemented using Q-Learning. The hybridization of Q-Learning with BA aims to find the optimal course with fewer iterations of the BA. This method combines the sanitation of Q-Learning to determine the BA and the quickest path to solve the problem without any restrictions. Fallooh et al. [18], stated that the robot was subjected to several simulated test scenarios to assess its adaptability to dynamic settings. The testing scenarios employed various target motion types, and obstacle counts with different dynamicity patterns. The test cases demonstrated how the robot could adapt to various environments. The algorithm is split into two phases: the operational stage and the learning stage. The robot constructs a learning matrix during the first phase, which uses rewards and environment data to help it learn how to get from its current location to a known goal. This learning matrix is then used during the operational stage. This research examined

the method to facilitate quick learning for the mobile robot and minimize the learning process's repetition by appropriately defining the values of alpha (α) and gamma (γ) to preserve variance and differentiate between them.

2. Mobile robot position control with RL

The mobile robot searches for the best route between the beginning and the goal point relating to predetermined efficiency parameters for route planning in a given working environment [17]. In general, based on whether the mobile robot has access to environmental data, routes are categorized into global and local [18]. Path planning steps: the mobile robot development process typically includes the following steps: environment modeling, optimization criteria, and pathfinding algorithm [22]. The algorithms A* and D* of the guided method are the most widely used algorithms for route search in general route planning [23]. Descriptive inference and heuristics have traditionally been used to optimize warehouse route design [24]. When the problem becomes complicated, most solutions demand significant computational time yet operate without additional learning [25]. Maintaining and scaling a path is challenging due to the complexity of heuristic algorithms in their creation and understanding. Artificial intelligence in route planning is becoming more widespread [26]. To identify a strategy for achieving a goal, reinforcement learning employs trial and error and learning from failure and reward [27].

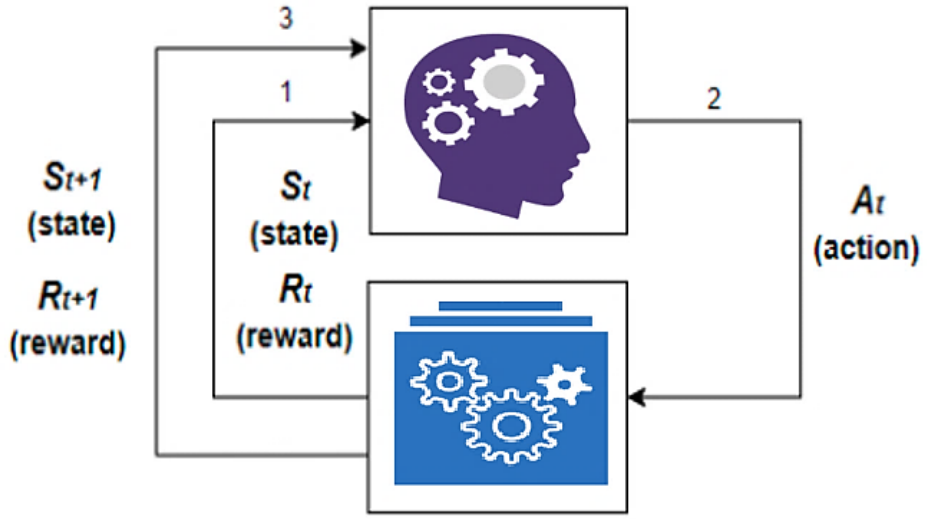


Figure 1: The main reinforcement learning components and their relationships

The environment is the outside world that an agent interacts with and is aware of, while the agents are the objects of learning and decision-making [3]. The environment's state influences the agent's decision-making [21]. A reward is the fourth element; it indicates whether an agent behaved desirably or undesirably as shown in Figure 1. A value function is the last element in reinforcement learning. What is advantageous in the long run is explained by the value function. The total award that an agent is qualified to receive is the value. The reward fluctuates depending on the current situation and is predicted by environmental models [29].

2.1 Q-Learning algorithm

The essential idea behind the Q-Learning algorithm is that, rather than using the later state-action pair corresponding to the present policy, it employs the future state-action pair generated by the policy that will be reviewed when updating the Q value of a state-action pair [30]. Specifically, the mobile robot uses multiple samples to generate paths in the path planning issue and randomly samples the surrounding environment [31]. During this period, there are iterations between the goal and manners policies until the best route is found through their interaction [32-34]. The formula for the Q-table of Q-learning is Equation 1:

$$Q'(s,a) = Q(s,a) + \alpha\,(r + \gamma\,(max_{a'}\,Q(s',a') - Q(s,a))) \quad (1)$$

where r: is the reward received after taking action an in state s, alpha(α): is the learning rate, gamma (γ): is the discount factor, Q(s, a): is the expected reward for taking action on an in-state s.s': is the next state after taking action a in state s, and max Q(s', a') is the maximum expected reward for taking any action a' in state s'.

The Q-table is updated according to the learning rate following all interactions with the surrounding environment [35]. The Q-table will update more quickly with an increasing learning rate, but there is a possibility that it may overfit the data. The Q-table will update slowly with a lower learning rate but may also converge more slowly. The discount factor determines the

weighting of future benefits. The Q-table will consider future incentives more heavily if the discount factor is larger and less heavily if the discount factor is lower [36]. Developing the optimum action plan for the next state is valued according to the highest possible projected benefit connected to any action a' in state s'. This allows the Q-table to learn which behaviors yield the greatest long-term rewards [23]. The Q-table changes each time the agent interacts with the surrounding environment while the agent determines a path of action that optimizes the expected reward. The policy is the allocation of events to actions that an agent will do to maximize the expected reward. The off-policy temporal-difference control method, often known as Q-Learning, is one of the most fundamental algorithms used in reinforcement learning [23]. When taking action, the Q-Learning algorithm stores the reward values in the Q table and chooses a random or maximum value based on the epsilon-greedy value [16].

$$Q(s,a) = r(s,a) + \gamma max_a(Q(s',a)) \quad (2)$$

Equation (2) is typically updated into Equation (1) by taking time into account.

$$Q'(s,a) \leftarrow Q(\,s,a) + \,\alpha\,(r + \gamma\,(\,max_{a'}\,Q(s',a') - Q(s,a)) \quad (3)$$

Algorithms for reinforcement learning depend heavily on the exploration and exploitation stage. Lower-priority actions should be selected before selecting and executing higher-priority actions to assess the optimal setting [37]. The Q-learning algorithm has become the cornerstone of the reinforcement learning algorithm due to its ease of use and remarkable learning performances in a single-agent environment. Due to Q-Learning's one-time update per action, complex issues become difficult to solve when many states and actions are unrealized [38]. Because the Q-table for reward values is predefined, a significant amount of storage RAM is further required. In a complex multi-agent system, this is a problem for the Q-Learning technique [39].

2.2 Modified q-learning (I8QA) for robot path planning

Combination and reduction are limiting parameters of the Q-learning algorithms that make up the Q-learning modification. The replanning method randomly selects samples using only previously experienced data, but the Q-learning algorithm modifies a policy in response to real-world experience. The replanning algorithm uses Search Control to gain a simulated experience, which is then used to modify the policy. Q-learning (1) is updated by the improvement Q-learning Equation (4):

$$Q(s,a) \leftarrow Q(s,a) + \propto (r + (1 - \propto) \max Q(s',a') - Q(s,a) \quad (4)$$

The Q-learning algorithm's learning phase is described in Algorithm 1, and Figure 2 shown the flow chart of modified Q-Learning for Robot Path Planning. Improving the Q-learning algorithm for generating the next state and a more accurate reward value involves storing past experiences in memory. To create the next state and a more accurate reward value, the Q-learning algorithm has been improved to keep previous experiences in memory. It is possible to assign a negative reward value for obstacles and a positive reward value for the goal location, creating a simple reward structure to guide a mobile robot's progress in a grid-style maze environment. The mobile robot's current and target positions define its direction. The path-planning method repeats numerous random path-finding attempts to determine the best path, which takes considerable time. The basic reward technique, previously discussed, involves an agent searching randomly for a path, whether or not it is located near or far from the destination. Once the journey path and target position have been determined, a portion of the area near the target position is selected to strengthen these weaknesses. Compared to the standard reward strategy (Reward adjustment area near the target point), the path placed inside the area selected in the previous phase is assigned a larger reward value. Then, whenever the movement path changes, we propose a method to choose a region, as mentioned previously, and to adjust the movement path to a spot in the selected area that has a high reward value. Update the reward modification based on this technique, as it relies on reducing the parameters to improve the process of calculating the procedure for the next policy by adjusting the learning rate to align with the discount factor of future rewards. The static and dynamic rewards are combined to optimize the reward function. In this study, we developed the method of searching space by developing the search within the reinforcement learning algorithm. The search was preferred using the inclined path method over searching the straight path due to the shorter path planning distance of the inclined path according to the law of distances of path length (PL) in Equation (5):

$$PL = \sum_{i=0}^{n} \sqrt{(y_{i+1} - y_i)^2 + (x_{i+1} + x_i)^2} \quad (5)$$

where i =0,1,2,...,n, when i =0, The beginning position of the mobile robot is s=(x_0, y_0). When i=n, the robot is at the target position T= (x_n, y_n), the current coordinates of the mobile robot are represented by (x_i, y_i); the coordinates of the next state are represented by (x_{i+1}, y_{i+1}).

Algorithm (1)

Input: Source (start state) location, destination (goal state) location, and solution space.
Output: optimal path for the robot from start to goal.
1 initialize Q(s,a) ← 0 (s states set, a actions set);
2 **for** each episodic(iteration) **do**
3 set $s_t \leftarrow a$ random state from state set S;
4 **while** ($s_{t \neq goal}$) **do**
5 **for** each $\boldsymbol{s_t^i} \in \boldsymbol{S}$ where $i \in$ [up, down, left, right, up-right, up-left, down-right, down-left] **do**
6 Determine location $loc_{s_t^{i+1}}$ of the agent by doing an action a_t^{i+1}
7 Calculate distance $dis_t^{i+1} \in dis_t$ from $loc_{a_t^{i+1}}$ to goal location [dis_t is allow distance without collision of all 8-neighboring connections]
8 Choose $loc_{s_t^{i+1}}$ corresponds to smallest dis_t^{i+1} from dis_t
9 If $loc_{s_t^{i+1}} \neq loc_{obstcal}$ do
10 Choose a_t^{i+1} corresponds to $loc_{s_t^{i+1}}$, which makes the agent move closer to the goal location
Perform action and receive a penalty or reward
11 Update Q ($s_{t+1,} a_{t+1}$)
12 Else
13 Chose a_t^i corresponds to $loc_{s_t^i}$, which makes the agent move closer to the goal location
14 **end**
15 Perform action and receive a penalty or reward
16 Update Q ($s_{t,} a_t$)
17 end
18 end

Furthermore, in a cartesian coordinate system, which represents an environment grid with the x- and y-axes denoting the horizontal and vertical directions, the shortest distance between two locations is represented by moving in the inclined direction with the smallest angle. The notation for the coordinates is (x, y) [27]. The first and second dimensions of the grid map represent the x-horizontal and y-vertical. This is achieved through the reward value variable in the Q-table, where the reward value is high for the inclined path that offers a shorter distance between the starting point and the target, assuming a lower number of collisions to increase the search speed. On the other hand, the execution time of the algorithm is reduced. The reward value is lower for the open path free of obstacles, but it is a straight path, which means it has a longer distance between the starting point and the target position. Using eight connections, this work develops the reward function (r) as expressed in equation (3) to (R) in Equation (6), representing the Q-values of the inclined path instead of the four connections used in classic Q-learning.

$$R = \begin{cases} 1 & Single\ horizontal.or\ Vertical. \\ 1.4 & Single\ Diagonal \\ 3 & s_{t+1}\ is\ the\ start\ node \\ 10 & s_{t+1}\ target\ node \\ -5 & is\ the\ eight\ grids\ around\ s_{t+1}\ forbidden\ node \end{cases} \quad (6)$$

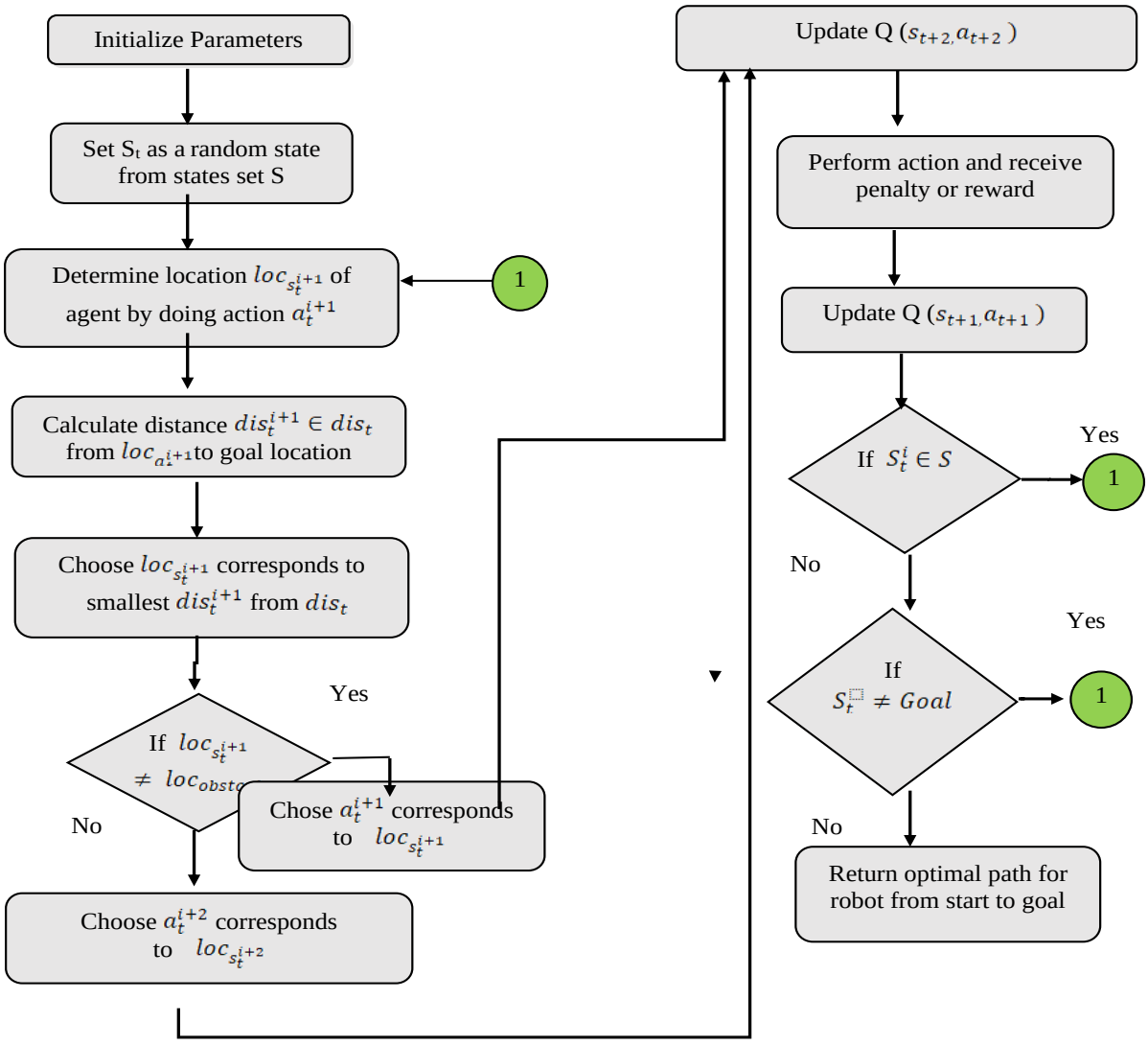


Figure 2: The flow chart of Modified Q-Learning for Robot Path Planning

3. Results and discussion

3.1 Maze simulation environment

A Study by Jiang et al. [28], was utilized and the previously mentioned maze setup as a point of reference to create a virtual experiment environment. A maze environment includes content obstacle, open path state, start state, and goal state Figure 3 shows the simple maze environment with start and goal position. Several obstacles were included in a 16 × 16 grid layout. The movement of a mobile robot between the starting point and the goal point was analyzed. In this experiment, the defined task for a mobile robot was to find the optimal path to the target point while avoiding obstacles at the starting site and constantly starting from the same place, never experimenting to find the best route from beginning to end. Move_from_abstacles toward each terminal state based on the action value traveling to the right, left, up, or down. Thorough simulation research was done to show the effectiveness of the new strategy and train the robot. To put this simulation into practice, MATLAB R2012b was employed. The efficacy of the suggested Q-learning strategy was evaluated using the effects of these various states. Several environments were used for different conditions. The training and testing phases are the two sections of the simulation. Different scenarios for various conditions were implemented using a range of target and obstacle behaviors during the training phase. The robot is learning as it moves through the various states of each scenario. The Q-table, which was previously described, becomes a repository for the knowledge collected through the various situations. The testing phase begins with exposing the robot to a more challenging environment and follows the learning phase. The number of obstacles in the testing situations is greater; some are static, and some are dynamic. Some testing scenarios for particular common problems, such as the local minima challenge, were used to test the technique. We also ran additional tests using random settings, such as the value of the test parameter utilized in the Q-learning and improvement of various state simulation environments.

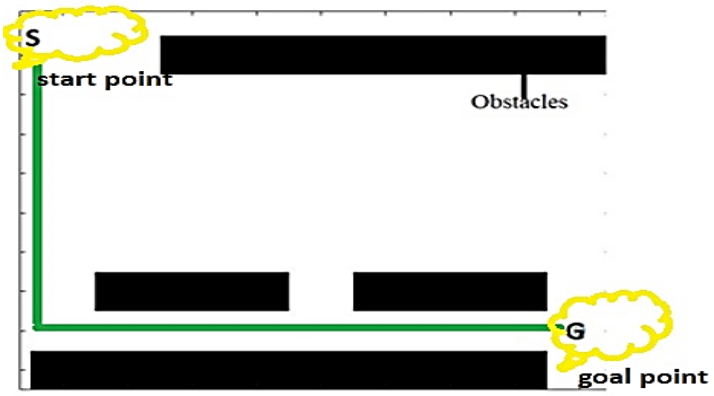


Figure 3: The simple maze environment with start and goal position

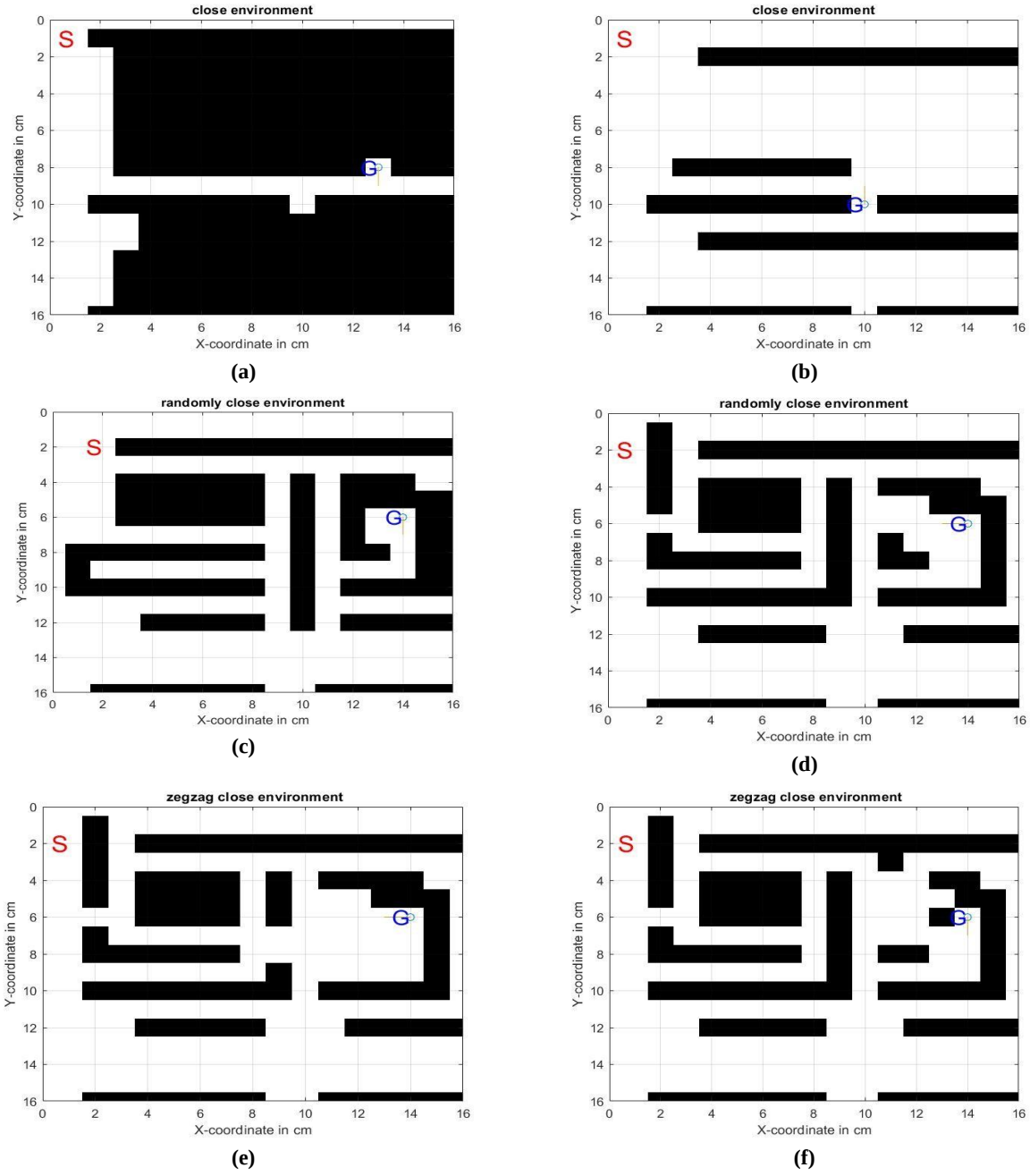


Figure 4: The different scenarios of test environments a) close, b) open, c) randomly close (and d) randomly open. e) zigzag close f) zigzag open environments

The proposed methods are tested with 2-D various difficulties suggested maps for both x-y dimensions and contain various shapes of static obstacles. Figure 4 demonstrates the six suggested environments with the variant shapes of obstacles and corresponding free up and down spaces in addition to the free Cartesian space. The Figure 4a is specification of suggests that the first environment comprises a variety of 102 obstacles and a closed environment with only one optimum path from the start to the goal point. The Figure 4b specification of suggests the second environment is composed of a variety of obstacles, 33 numbers, and different lengths of the optimum path of the mobile robot from the start to the goal point. The suggested open environment as shown in Figure 4c is specification of the third suggested environment composed of different lengths of the optimum path of the mobile robot from start to goal point. In comparison, the suggested random environment has a 77 variety of obstacles as shown in Figure 4d.

Figure 4e is specification of the fourth suggested environment is composed of various obstacles. It has only one length of the optimum path of the mobile robot, but the joints are limited. Figure 4f is specification of the fifth suggested zigzag-closed environment is composed of different lengths of the optimum path of the mobile robot from the start to the goal point. The suggested zigzag-open environment has a variety of obstacles. It has more than one path for the mobile robot from the start to the goal point, so the robot must learn the optimum path with smooth movement.

The results obtained through the experiment in many environments shown in Figure 4 with different dynamic obstacles and different Q-learning scenarios are shown in Table 1 for the original Q-learning and Table 2 for the modified Q-learning based on (α) and (γ) with improved discount rates that vary depending on the learning rate.

Table 1: The Results of Original Q-Learning in Different Obstacle Environments

Env.	Learning Rate (α)	Discount factor (γ)	Time of execution algorithm in sec.	Time of execution classic algorithm in sec.	No. of obstacle	Path
	0.1	0.7	1.882	2.282	33	Smoothing
Open		0.8	1.756	2.156	33	Smoothing
environment		0.9	1.775	2.257	33	Less smoothing
	0.2	0.7	1.551	1.975	33	Winding path
		0.8	1.528	1.941	33	Smoothing
		0.9	1.524	1.992	33	Less smoothing
	0.3	0.7	1.505	2.190	33	Smoothing
		0.8	1.466	2.138	33	Less smoothing
		0.9	1.407	2.061	33	Winding path
Close	0.2	0.7	2.257	2.793	102	Without Smoothing
environment		0.8	2.314	2.819	102	Smoothing
		0.9	2.252	2.750	102	Winding Smoothing
	0.3	0.7	2.461	2.961	102	Smoothing
		0.8	2.398	2.896	102	Less smoothing
		0.9	2.328	2.828	102	Winding
	0.4	0.6	2.563	3.031	102	Smoothing
		0.7	2.583	3.019	102	Less smoothing
		0.8	2.471	2.917	102	Less
		0.9	2.348	1.998	102	Winding
Randomly	0.2	0.7	3.495	4.291	77	Less smoothing
environment		0.8	3.490	4.203	77	Smoothing
		0.9	3.412	4.112	77	Winding
	0.3	0.7	3.628	4.226	77	Smoothing
		0.8	3.618	4.218	77	Less smoothing
		0.9	3.537	4.137	77	Less winding
	0.4	0.6	4.892	5.292	77	Smoothing
		0.7	4.872	5.216	77	Less smoothing
		0.8	4.607	5.103	77	Winding
		0.9	4.639	5.632	77	Less winding
					77	

Table 1 notes the results for the six environments. The shortest path for the mobile robot is when the learning coefficient values are small and close to 0.2. At the same time, we must maintain the best distinction between the discount and learning coefficients to obtain the shortest path free of twists and characterized by smoothness. From the starting point to the goal, the mobile robot must learn the optimum path with smooth movement and suitable speed. Figure 5 and Figure 6 show the execution time of each algorithm depending on the changed parameters.

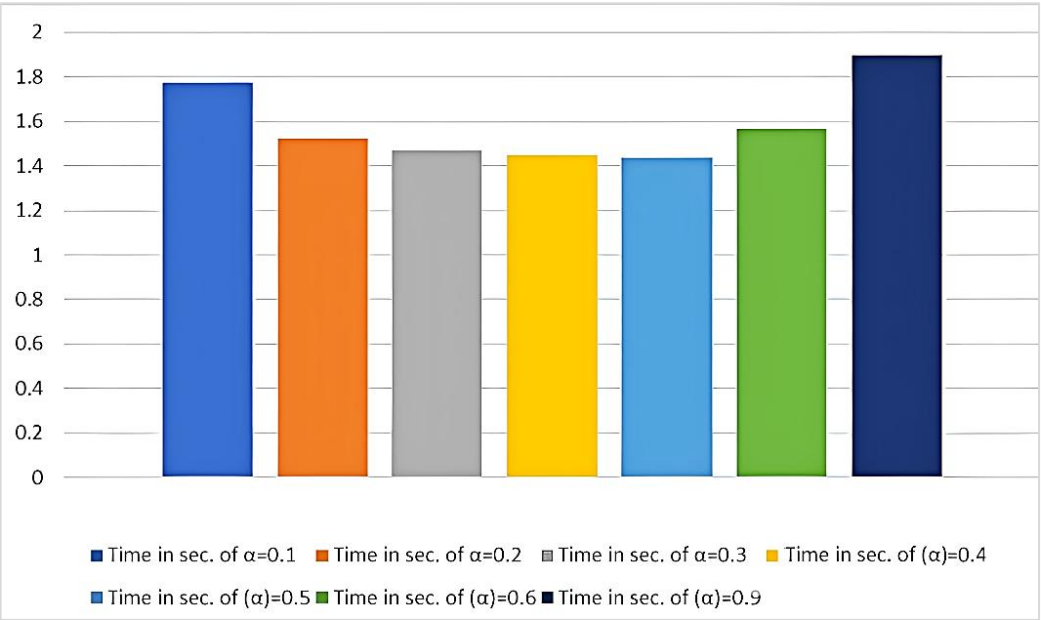


Figure 5: Time algorithm execution of open environment with γ= 0.9

Table 2: Results of Modified Q-Learning in Different Dynamic Obstacle Environments

Env	Learning Rate (α)	Time of execution algorithm in sec.	Time of execution classic algorithm in sec.	No. of obstacles	Path
Open environment	0.1	1.882	2.328	33	Smoothing
		1.856	2.256	33	Smoothing
		1.775	2.274	33	Less smoothing
	0.2	1.551	1.951	33	Winding path
		1.524	1.930	33	Smoothing
		1.528	1.925	33	Less smoothing
	0.3	1.505	2.016	33	Smoothing
		1.466	1.966	33	Less smoothing
		1.407	1.912	33	Winding path
Close environment	0.2	2.257	2.775	102	Less smoothing
		2.314	2.851	102	Smoothing
		2.252	2.725	102	Winding path
	0.3	2.461	2.916	102	Smoothing
		2.398	2.898	102	Less smoothing
		2.328	2.828	102	Winding path
	0.4	2.563	3.102	102	Smoothing
		2.471	2.971	102	Less smoothing
		2.348	2.858	102	Winding path
Randomly environment	0.2	3.495	3.959	77	Less smoothing
		3.490	3.990	77	Smoothing
		3.412	3.812	77	Winding path
	0.3	3.628	4.262	77	Less smoothing
		3.618	4.102	77	Smoothing
		3.537	3.975	77	Winding path
	0.4	4.892	5.394	77	Smoothing
		4.872	5.349	77	Less smoothing
		4.607	5.102	77	Winding path

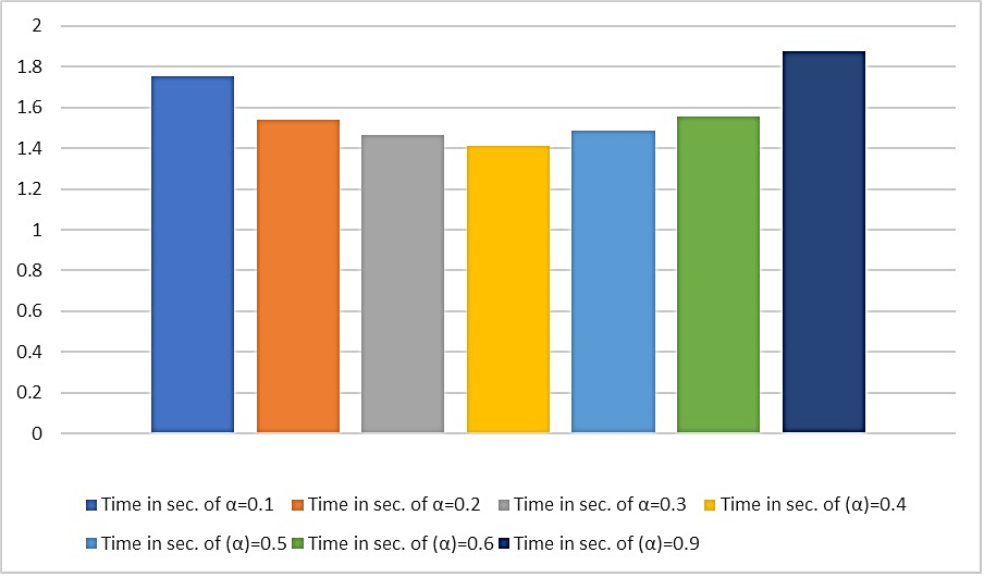
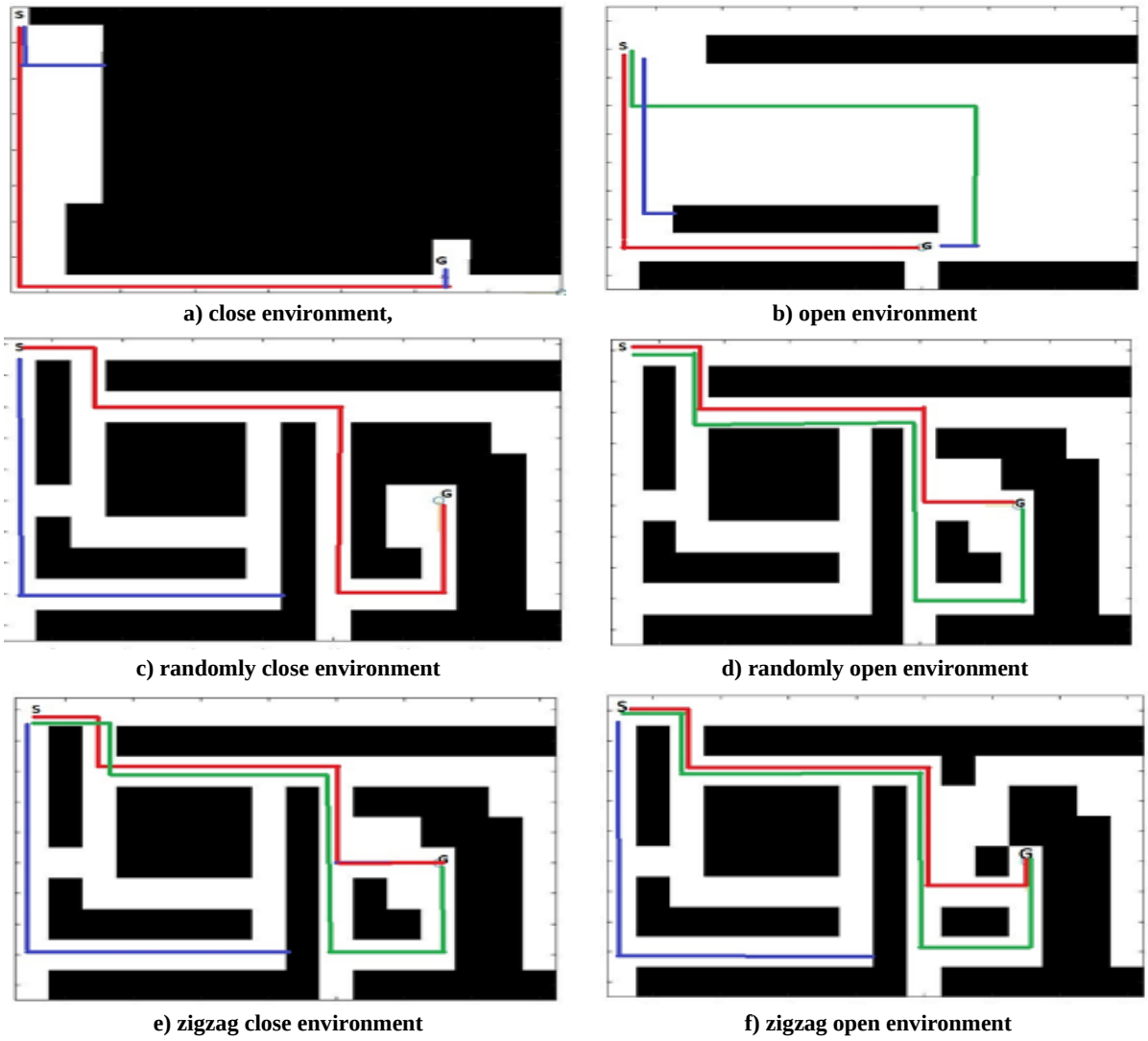


Figure 6: Time algorithm execution of open environment with smoothing path at γ= 0.8



a) close environment, **b) open environment**

c) randomly close environment **d) randomly open environment**

e) zigzag close environment **f) zigzag open environment**

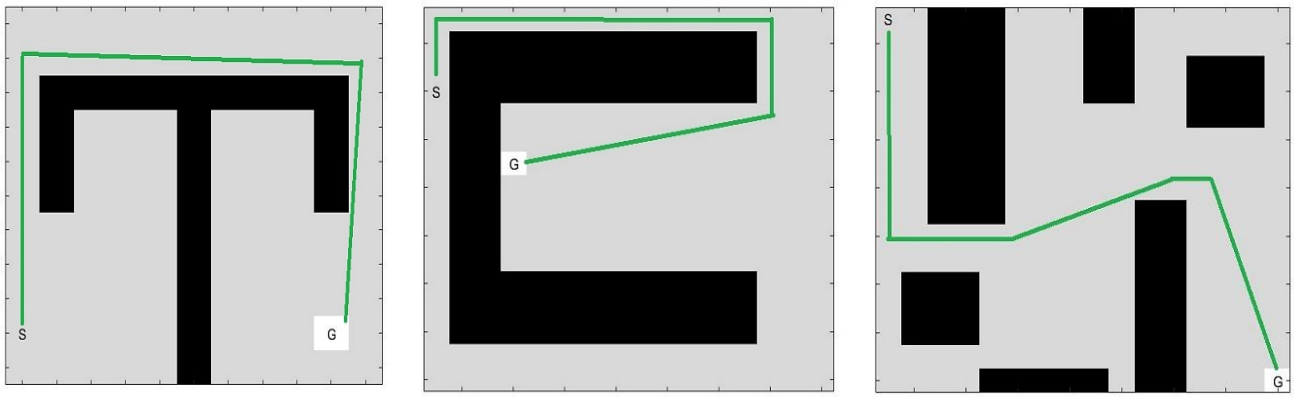
Figure 7: The best path planning for various scenarios in the different test environments, a) close, b) open, c) randomly close (and d) randomly open. e) zigzag close f) zigzag open environments, the best path planning is red, the longer path planning is green, and the path planning closed is blue

The best path planning for various scenarios in the different test environments is shown in Figure 7. The red color points to the optimum path between the proposed algorithm's starting point and goal point; the longer path planning is green, and the path planning closed is blue in the different test environments. Additionally, each environmental test includes a different path scenario, though it may not be optimum. Furthermore, the algorithm's performance was improved by searching in different environments for the inclined path to reach the goal instead of the straight path. This improvement is because the inclined path is shorter than the straight path, as detailed in the next section.

The relationship between α and γ must be considered in terms of the discount coefficient values and the learning rate to achieve the best results in correlating time with the values of current and future rewards. We also notice from the results that the best differentiation and variance of the results occur when the learning rate α take values are close to zero and do not exceed 0.2. As the values increase to α, we notice an increase in the learning time of the algorithm, but this is not desirable because of its effect on the smoothness of the path. As for the discount factor, having a value of γ Approximately 1 is preferable, but not less than 0.8. The lower the discount rate, the longer the algorithm's learning time. The best results were achieved with a learning rate 0.2 when the algorithm was studied in various environments with dynamic obstacles. In the closed environment, the execution time varied depending on the number of obstacles. The Smoothed path at the best value for a learning rate of 0.2 and a discount factor of 0.8 had an execution time of 2.314 sec. If we need to increase the path smoothing, the discount factor should be set to 0.9 and the learning rate to 0.1. This setup yields good results, however. The algorithm execution time is (2.281) sec. This means that the smoothing path of the mobile robot between the start and target positions requires optimization, relying on the learning rate and discount factor relative to the execution time of the algorithm. In addition, the algorithm's performance was improved by searching different environments for the inclined path to reach the goal instead of the straight path. This is because the length of the inclined path is shorter than the straight path.

4. Comparison of the results with recent works

To prove the efficiency of the proposed algorithm in finding the shortest path, a comparison was made with other algorithms across different test environments, such as Bonny and Kashkash [17]. To ensure a fair comparison with recent works, the I8QA is applied to the same maps used in those studies, considering the same map's dimensions (width and height), obstacle positions, starting and goal points, and mobile robot configuration. Then, the length of the final path for all methods was comparedThe comparison results of the environment (map (a), map (b), and map (c) are shown in Figure 8. The algorithms used for comparison are the results of QBA and I8QL. The final mean path length of (QBA) is 24.1733, 26.253, and 36.2587. Meanwhile, the I8QA obtained the optimal path with the mean path length of the environment, as shown in Figure 9, which is 20.2451, 21.8671, and 29.2851. Table 3 shows the result of the comparison of the different environments.



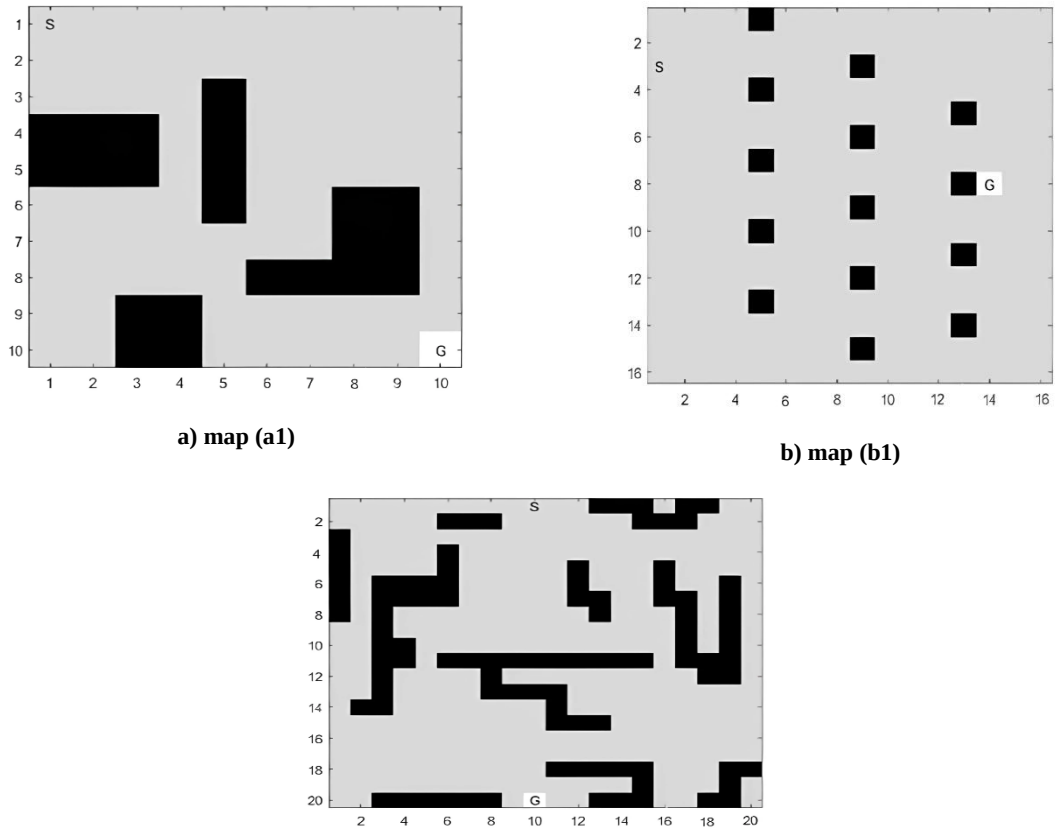
a) map (a) **b) map (b)** **c) map (c)**

Figure 8: Optimum path of different environments a) map (a), b) map (b), c) map (c) [17]

Table 3: The result of the Mean Path length and Standard deviation of different environment

Map Algorithms of other reference	Mean Path Lengh (QBA)	Standard deviation	The mean Path length of I8QA	Standard deviation I8QA
Map (a) Bonny and Kashkash [17]	24.1733	0.1259	20.2451	0.1011
Map (b) Bonny and Kashkash [17]	26.253	0.1102	21.8671	0.1
Map (c) Zhang [21]	36.2587	0.15735	29.2851	0.1

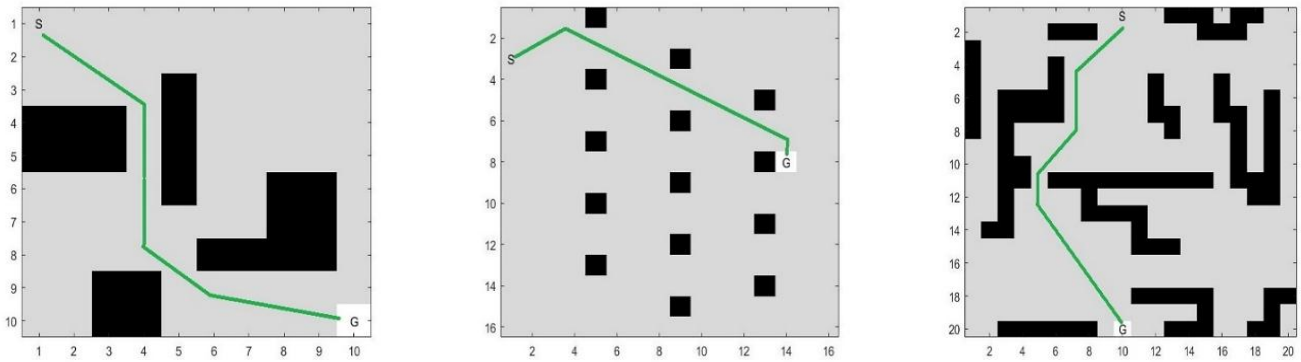
The algorithms used for comparison are the CLSQL [15], QBA [17], and EQL [11]. Table 4 shows the results of the optimum path of I8QA, and Figure 9 (a–c) shows the different environments (map a1, map b1, and map c1). Figure 10 (a–c) shows the final path length of the different environments map a1, map b1, and mab c1.



a) map (a1) **b) map (b1)**

c) map (c1)

Figure 9: The different environments:a) map (a1),b) map(b1),c) and map (c1)



(a) map (a1) **(b) map (b1)** **(c) mab (c1)**

Figure 10: Optimum path of different environments: a) map (a1) b) map (b1) and c) map (c1)

The path length of Map (a1) is 14.1102 [17], and the shortest path length of I8QA is 11.053, so the number of iterations is reduced from reference [17] to 43 iterations. The path length of Map (b1) is 21.04 [15], and the shortest path length of I8QA is 12.453; with a time of execution, the algorithm is reduced from 0.33 msec to 0.284 msec. The path length of Map (b1) is 20.20 [11], and the shortest path length of I8QA is 12.453; the path length of Map (c1) is 21.771 [17], and the shortest path length of I8QA is 17.153, so, the number of iterations is reduced than reference [17] to 50 iterations. The improvement ratio is 21.666%,38.351%, 40.812%, and 19.047%. Whereas the I8QA obtained the optimal path with the length and enhancement time of execution, the algorithm improved the search strategy to reach the goal by planning the optimal path within different environments containing different obstacles and spaces. Table 4 shows the results of the comparison of the environments.

Table 4: The Optimum Path length of I8QA and other algorithms of different environment

Map	Map (a1) [17]	Map (b1) [15]	Map (c1) [17]	Map (b1) [11]
Map size	10*10	16*16	20*20	16*16
Path length	14.1102	21.04	21.7716	20.20
The path length of the proposed algorithm	11.053	12.453	17.6246	12.453
Improvement ratio	21.666%	38.351%	19.047%	40.812%

5. Conclusion

This paper presented a Q-learning algorithm with a single parameter; the modified Q-learning algorithm was applied to optimizing paths in a labyrinth setting and empirically validated methods to reduce passive exploration. The main purpose of the proposed modification is to reduce the number of parameters; thus, we do not need many experiments to choose the best values for these parameters. We constructed multiple environments to test path navigation within the Q-learning maze environment and enhance the Q-learning algorithm for RL. The search times for the path and final path in the experiment findings were compared. The modified Q-learning algorithm outperformed the Q-learning algorithm in terms of path search time, producing a final path that was both smoother and faster than that of the Q-learning algorithms. This is why the experimental outcomes of the Q-learning and Q-improvement algorithms demonstrated acceptable outcomes for both the end path length and the path search time. A dynamic reward system focused on learning rather than solely on action was used to improve worker performance and efficiency. The Q-learning algorithm has the property of finding paths randomly. To reduce the randomness in finding an optimal path, the movement toward the target position was selected, as the reward value for the unobstructed path was higher than that for the path with obstacles in the direction of the target position. According to additional tests, the path search time was faster than the Q-learning technique, and the final path length was optimized. A smooth path results from learning that converges to the Q-learning optimum value function. This may take a long time if the learning rate exceeds 0.3 or occur too quickly if the learning rate is smaller and equal to 0.2. The smoothness of the mobile robot's movement while navigating various paths within the environment is essential to avoid the problem of battery drain and collision with obstacles. In future work, the smoothing rate of path planning will be increased and improved by integrating it with a metaheuristic algorithm and reducing the collision rate to make the algorithm more compatible with dynamic environments.

Author contributions

Conceptualization, **N. Fallooh, A. Sadiq, E. Abbas,** and **I. Hashim.**; data curation, **N. Fallooh**.; formal analysis, **N. Fallooh**.; investigation, **N. Fallooh**, **A. Sadiq**, **E. Abbas**, and **I. Hashim.**; methodology, **N. Fallooh**, **A. Sadiq.**; project administration, **N. Fallooh**, **E. Abbas**, and **I. Hashim.**; resources **N. Fallooh.**; software, **N. Fallooh**, **A. Sadiq**.; supervision, **A. Sadiq**, **E. Abbas**, and **I. Hashim.**; validation, **N. Fallooh**, **A. Sadiq**, **E. Abbas**, and **I. Hashim**.; visualization, **N. Fallooh**, **A. Sadiq**, **E. Abbas**, and **I. Hashim**.; writing—original draft preparation, **N. Fallooh**, **A. Sadiq**, **E. Abbas**, and **I. Hashim**.; writing—review and editing, **N. Fallooh**, **A. Sadiq**, **E. Abbas**, and **I. Hashim**. All authors have read and agreed to the published version of the manuscript.

Funding

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

Data availability statement

The data that support the findings of this study are available on request from the corresponding author.

Conflicts of interest

The authors declare that there is no conflict of interest.

References

[1] P. Wang, Ch. Chan and A. de-L. Fortelle, A Reinforcement learning based approach for automated lane change maneuvers, 2018 IEEE I Intell. Veh Symposium (IV), China, 2018. https://doi.org/10.48550/arXiv.1804.07871

[2] M. Naeem, S.T.H. Rizvi and A. Coronato, A Gentle introduction to reinforcement learning and its application in different fields, IEEE Access, 8 (2020) 209320-209344. https://doi.org/10.1109/ACCESS.2020.3038605

[3] G.U.O. Tong, N. Jiang, L.I. Biyue, Z.H.U. Xi, Y. Wang, UAV navigation in high dynamic environments: A deep reinforcement learning approach, Chin. J. Aeronaut., 34 (2020) 479-489. http://dx.doi.org/10.1016/j.cja.2020.05.011

[4] C. Wang, J. Wang, Y. Shen, Autonomous navigation of UAVs in large-scale complex environments: A deep reinforcement learning approach, IEEE Trans Veh. Technol., 68 (2019) 2124-36. https://doi.org/10.1109/TVT.2018.2890773

[5] L. Chang, L. Shan, Ch. Jiang, and Y. Dai, Reinforcement based mobile robot path planning with improved dynamic window approach in unknown environment, Auton. Rob., 45 (2021) 51-76. https://doi.org/10.1007/s10514-020-09947-4

[6] J. Kim, M. Hong, K. Lee, D.W. Kim, Y.-L. Park, and S. Oh, Learning to Walk a Tripod Mobile Robot Using Nonlinear Soft Vibration Actuators with Entropy Adaptive Reinforcement Learning, IEEE Rob. Autom. Lett., 5 (2020) 2317-2324. https://doi.org/10.1109/LRA.2020.2970945

[7] S.S. Mousavi, M. Schukat, and E. Howley, Traffic light control using deep policy gradient and value-function-based reinforcement learning, IET Inst. Eng. Technol., 11 (2017) 417-423. https://doi.org/10.1049/iet-its.2017.0153

[8] Bakr S. Shihab, Hadeel N. Abdullah and Layth A. Hassnawi, Improved Artificial Bee Colony Algorithm-based Path Planning of Unmanned Aerial Vehicle Using Late Acceptance Hill Climbing, Int. J. Intell. Eng. Syst., 15 (2022) 432-442. https://doi.org/10.22266/ijies2022.1231.39

[9] A.T. Sadiq, and A. H. Hasan, Robot Path Planning Based on PSO and D* Algorithms in Dynamic Environment, Int. Conf. Current Research in Computer Science and Information Technology (ICCIT), Slimani – Iraq, 2017. http://dx.doi.org/10.1109/CRCSIT.2017.7965550

[10] E.S. Low, P. Ong, and K. Ch. Cheah, Solving the optimal path planning of a mobile robot using improved Q-learning, Rob. Auton. Syst., 115 (2019) 143-161. https://doi.org/10.1016/j.robot.2019.02.013

[11] X. Luo, Y. Gao, and Sh. Huang, Modification of Q-learning to Adapt to the Randomness of Environment, Int. Conf. Control, Automation and Information Sciences (ICCAIS), Chengdu, China , 2019. https://doi.org/10.1109/ICCAIS46528.2019.9074718

[12] H.S. Lee, and J. Jeong, Mobile Robot Path Optimization Technique Based on Reinforcement Learning Algorithm in Warehouse Environment, Appl. Sci., 11 (2021) 1209. https://doi.org/10.3390/app11031209

[13] H. Sang, Y. You, X. Sun, Y. Zhou, and F. Liu, The hybrid path planning algorithm based on improved A* and artificial potential field for unmanned surface vehicle formations, Ocean Eng., 223 (2021) 108709. http://dx.doi.org/10.1016/j.oceaneng.2021.108709

[14] A. Maoudj, A. Hentout, Optimal path planning approach based on Q-learning algorithm for mobile robots, Appl. Soft Comput. P.A., 97 (2020) 106796. https://doi.org/10.1016/j.asoc.2020.106796

[15] T. Ma, and J. Lyu, J. Yang, R. Xi, Y. Li , J. An, and Ch. Li, CLSQL: Improved Q-Learning Algorithm Based on Continuous Local Search Policy for Mobile Robot Path Planning, Sensors, 22 (2022) 5910. https://doi.org/10.3390/s22155910

[16] J. Qin, Path Planning Method of Mobile Robot Based on Q-learning, Journal of Physics: Conference Series, International Symposium on Artificial Intelligence and Intelligent, 2181, 2022, 012030. https://doi.org/10.1088/1742-6596/2181/1/012030

[17] T. Bonny and M. Kashkash, Highly optimized Q-learning-based bees' approach for mobile robot path planning in static and dynamic environments, J. Field Rob., 39 (2022) 317-334. http://dx.doi.org/10.1002/rob.22052

[18] N. H. Fallooh, A. T. Sadiq, E. I. Abbas and I. A. Hashim, Modifiedment the Performance of Q-learning Algorithm Based on Parameters Setting for Optimal Path Planning, Fifth Int. Sci. Conf. Alkafeel University, BIO Web of Conf., 97 (2024) 00010. https://doi.org/10.1051/bioconf/20249700010

[19] M.A.K. Jaradat, M. Al-Rousan, L. Quadan, Reinforcement based mobile robot navigation in dynamic environment, Rob. Comput. Integr. Manuf., 27 (2011) 135-149. https://doi.org/10.1016/j.rcim.2010.06.019

[20] H. A. R. Akkar, F. R. Mahdi, Adaptive Path Tracking Mobile Robot Controller Based on Neural Networks and Novel Grass Root Optimization Algorithm, Int. J. Intell. Syst. Appl., 9 (2017) 1-9. https://doi.org/10.5815/ijisa.2017.05.01

[21] X. Zhang, Y. Zhao1, N. Deng, and K. Guo, Dynamic path planning algorithm for a mobile robot based on visible space and an improved genetic algorithm, Int. J. Adv. Rob. Syst., 13 (2016) 1-17. http://dx.doi.org/10.5772/63484

[22] N. H. Fallooh, A. T. Sadiq, E. I. Abbas and I. A. Hashim, Dynamic Path Planning using a modification Q-Learning Algorithm for a Mobile Robot, Fifth Int. Sci. Conf. Alkafeel University (ISCKU 2024), 97 (2024) 00011. https://doi.org/10.1051/bioconf/20249700011

[23] Richard S. S. and Andrew G. B., Introduction to Reinforcement Learning, 2nd ed., MIT Press: London, UK, 2018.

[24] H. A. Atiyah and M. Y. Hassan, Outdoor Localization of 4 Wheels for Mobile Robot Using CNN with 3D Data, Int. J. Adv. Sci. Eng. Inf. Technol., 12 (2022) 1403-1409. http://dx.doi.org/10.18517/ijaseit.12.4.16181

[25] N. Kohl, and P. Stone, Policy gradient reinforcement learning for fast quadrupedal locomotion, Int. Conf. Robotics and Automation, IEEE, 2004. https://doi.org/10.1109/ROBOT.2004.1307456

[26] M. Kirtas, K. Tsampazis, N. Passalis and Deepbots, A Webots-Based Deep Reinforcement Learning Framework for Robotics, Proc. 16th IFIP WG 12.5 Int. Conf. AIAI 2020, Marmaras, Greece, (2020) 64-75. http://dx.doi.org/10.1007/978-3-030-49186-4_6

[27] F.A. Raheem, A.T. Sadiq, N. A. F. Abbas, Optimal Trajectory Planning of 2-DOF Robot Arm Using the Integration of PSO Based on D* Algorithm and Cubic Polynomial Equation , the first for Conference engineering researches, 2017.

[28] L. Jiang, R. Wei and D. Wang, Multi-UAV Roundup Inspired by Hierarchical Cognition Consistency Learning Based on an Interaction Mechanism, Drones, 7 (2023) 462. https://doi.org/10.3390/drones7070462

[29] Hidayat , A. Buono , K. Priandana and S. Wahjuni , Modified Q-Learning Algorithm for Mobile Robot Real-Time Path Planning using Reduced States, Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi), 7 (2023) 628-636. https://doi.org/10.29207/resti.v7i3.4949

[30] B. Wang, Z. Liu, Q. Li, A. Prorok, Mobile robot path planning in dynamic environments through globally guided reinforcement learning, EEE Rob. Autom. Lett., 5 (2020) 6932-6939. https://doi.org/10.48550/arXiv.2005.05420

[31] Z. Wang, The impact of Q-learning parameters on robot path planning problems in different complex environment, Proc. 4th Int. Conf. Signal Processing and Machine Learning, 135-141, 2024. https://doi.org/10.54254/2755-2721/54/20241447

[32] N. H. Fallooh, A. T. Sadiq, E. I. Abbas and I. A. hashim, Modifiedment the Performance of Q-learning Algorithm Based on Parameters Setting for Optimal Path Planning, 5th Int. Sci. Conf. Alkafeel University (ISCKU 2024), 00010, 2024. https://doi.org/10.1051/bioconf/20249700010

[33] Y. Cao and X. Fang , Optimized-Weighted-Speedy Q-Learning Algorithm for Multi-UGV in Static Environment Path Planning under Anti-Collision Cooperation Mechanism, Mathematics, 11 (2023) 2476, https://doi.org/10.3390/math11112476

[34] Z. Li, L. Shi, L. Yang, Z. Shang, An adaptive learning rate Q-Learning algorithm based on lalman filter inspired by pigeon pecking-color learning, Int. J. Bio-Inspir. Com., 1160 (2020) 693-706. https://link.springer.com/chapter/10.1007/978-981-15-3415-7_59

[35] A. Sonny, S. R. Yeduri and L. R. Cenkeramaddi, Q-learning-based unmanned aerial vehicle path planning with dynamic obstacle avoidance, Appl. Soft Comput., 147 (2023) 110773. https://doi.org/10.1016/j.asoc.2023.110773

[36] K.B. de Carvalho, I.R.L. de Oliveira, D.K.D. Villa, A.G. Caldeira, M. Sarcinelli Filho, A.S. Brandão, Q-learning based path planning method for UAVs using priority shifting, 2022 Int. Conf. Unmanned Aircraft Systems (ICUAS), 2022, 421-426.

[37] C. Wang, X. Yang, H. Li, Improved Q-learning applied to dynamic obstacle avoidance and path planning, IEEE Access, 10 (2022) 92879-92888. http://dx.doi.org/10.1109/ACCESS.2022.3203072.