Received 11 October 2023
Revised 21 December 2023
8 January 2024
Accepted 11 January 2024

A dynamic reward-enhanced Q-learning approach for efficient path planning and obstacle avoidance in mobile robotics

Atef Gharbi
Department of Information Systems,
Faculty of Computing and Information Technology, Northern Border University,
Rafha, Saudi Arabia

Abstract

Purpose – The purpose of the paper is to propose and demonstrate a novel approach for addressing the challenges of path planning and obstacle avoidance in the context of mobile robots (MR). The specific objectives and purposes outlined in the paper include: introducing a new methodology that combines Q-learning with dynamic reward to improve the efficiency of path planning and obstacle avoidance. Enhancing the navigation of MR through unfamiliar environments by reducing blind exploration and accelerating the convergence to optimal solutions and demonstrating through simulation results that the proposed method, dynamic reward-enhanced Q-learning (DRQL), outperforms existing approaches in terms of achieving convergence to an optimal action strategy more efficiently, requiring less time and improving path exploration with fewer steps and higher average rewards.

Design/methodology/approach – The design adopted in this paper to achieve its purposes involves the following key components: (1) Combination of Q-learning and dynamic reward: the paper's design integrates Q-learning, a popular reinforcement learning technique, with dynamic reward mechanisms. This combination forms the foundation of the approach. Q-learning is used to learn and update the robot's action-value function, while dynamic rewards are introduced to guide the robot's actions effectively. (2) Data accumulation during navigation: when a MR navigates through an unfamiliar environment, it accumulates experience data. This data collection is a crucial part of the design, as it enables the robot to learn from its interactions with the environment. (3) Dynamic reward integration: dynamic reward mechanisms are integrated into the Q-learning process. These mechanisms provide feedback to the robot based on its actions, guiding it to make decisions that lead to better outcomes. Dynamic rewards help reduce blind exploration, which can be time-consuming and inefficient and promote faster convergence to optimal solutions. (4) Simulation-based evaluation: to assess the effectiveness of the proposed approach, the design includes a simulation-based evaluation. This evaluation uses simulated environments and scenarios to test the performance of the DRQL method. (5) Performance metrics: the design incorporates performance metrics to measure the success of the approach. These metrics likely include measures of convergence speed, exploration efficiency, the number of steps taken and the average rewards obtained during the robot's navigation.

Findings – The findings of the paper can be summarized as follows: (1) Efficient path planning and obstacle avoidance: the paper's proposed approach, DRQL, leads to more efficient path planning and obstacle avoidance for MR. This is achieved through the combination of Q-learning and dynamic reward mechanisms, which guide the robot's actions effectively. (2) Faster convergence to optimal solutions: DRQL accelerates the convergence of the MR to optimal action strategies. Dynamic rewards help reduce the need for blind exploration, which typically consumes time and this results in a quicker attainment of optimal solutions. (3) Reduced exploration time: the integration of dynamic reward mechanisms significantly reduces the time required for exploration during navigation. This reduction in exploration time contributes to more efficient and quicker path planning. (4) Improved path exploration: the results from the simulations indicate that the DRQL method leads to improved path exploration in unknown environments. The robot takes fewer steps to reach its destination,

Applied Computing and Informatics
Vol. 22 No. 3/4, 2026
pp. 262-275
Emerald Publishing Limited
e-ISSN: 2210-8327
p-ISSN: 2634-1964
DOI 10.1108/ACI-10-2023-0089

Conflict of interest: The author declares that there is no conflict of interest.

which is a crucial indicator of efficiency. (5) Higher average rewards: the paper's findings reveal that MR using DRQL receive higher average rewards during their navigation. This suggests that the proposed approach results in better decision-making and more successful navigation.

Originality/value – The paper's originality stems from its unique combination of Q-learning and dynamic rewards, its focus on efficiency and speed in MR navigation and its ability to enhance path exploration and average rewards. These original contributions have the potential to advance the field of mobile robotics by addressing critical challenges in path planning and obstacle avoidance.

Keywords Path planning, Mobile robot, Q-learning, Dynamic reward-enhanced Q-learning (DRQL)

Paper type Research paper

1. Introduction

Path planning is a fundamental challenge in mobile robot (MR) safe and efficient navigation in an unknown environment that may contain obstacles. Path planning encompasses a diverse array of approaches, with the optimal choice contingent upon the unique attributes of the environment and the robot in question. Contrary to the known environment [1], where the robot possesses knowledge of the terrain and path planning can be straightforward, in a partially known environment, only partial mapping is available and the robot grapples with uncertainty regarding concealed obstacles [2]. Completely unknown environment [3] is the most challenging scenario where the robot confronts uncharted terrain, devoid of any prior mapping. Each of these scenarios requires distinct path-planning techniques to address the specific challenges posed by varying degrees of environmental familiarity. Path planning in a known environment is relatively straightforward, as the robot knows where all the obstacles are. However, path planning in a partially known or unknown environment is more challenging, as the robot must first map the environment and then find a path that avoids obstacles. Partially known environments are often the most practical, as they allow the robot to benefit from the knowledge of previously mapped areas while still being able to navigate in new areas.

The selection of a path planning algorithm significantly impacts a robot's navigation in terms of safety, efficiency and robustness. Path planning in robotics is broadly categorized into static and dynamic approaches. In static planning, the robot charts a fixed route around stationary obstacles in the environment [4]. Conversely, dynamic planning adapts to moving obstacles, requiring the robot to continuously adjust its path to navigate safely through the evolving surroundings [5]. While static planning is generally simpler, it may not suffice when obstacles are in motion, rendering dynamic planning essential for safe navigation.

This study introduces the dynamic reward-enhanced Q-learning (DRQL) algorithm, merging Q-learning techniques with dynamic reward mechanisms. This innovative approach allows robots to navigate unknown environments, adapting to environmental changes and task variations during movement. Empirical results demonstrate that this algorithm outperforms traditional Q-learning, showcasing improved convergence speed, optimization and adaptability. Our contributions can be outlined in three main aspects. First, we introduce dynamic rewards, leveraging information limitations in unknown environments. Static rewards relate to state node characteristics, while dynamic rewards vary based on target point distance, preventing blind searches and excessive exploration, enhancing learning efficiency. Second, the DRQL algorithm encompasses three steps: (1) exploration, (2) exploration and exploitation and (3) exploitation, addressing limitations of classical Q-learning in path planning. Third, experiments validate the effectiveness of DRQL in tackling complex path-planning challenges encountered by MR in diverse environments.

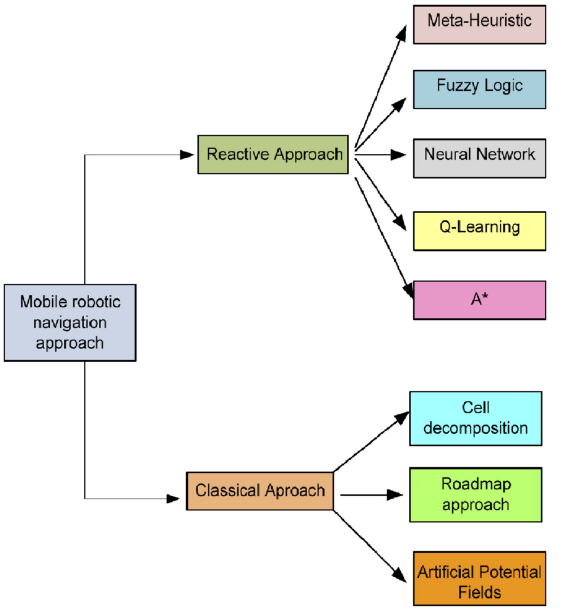
The subsequent sections of this paper are arranged as follows: the second section contains an exploration of related works in the field. Section 3 describes the formulation of the modified Q-learning algorithm, describing its four core components. Section 4 details the simulation methods used. Finally, the final section summarizes the key findings and indicates potential avenues for future research.

2. Related works

Following an extensive assessment of the literature, the navigation methodologies in robotics are categorized into two principal paradigms: classical approaches and reactive strategies (see Figure 1).

Historically, robotics has heavily focused on classical approaches, such as cell decomposition [6], roadmap approach [7] and artificial potential field [8]. However, these methods suffer limitations in computational complexity, susceptibility to local minima, uncertainty handling, reliance on precise data and the need for accurate real-time sensing. As a result, doubts persist regarding their practicality in real-time applications. Efforts to enhance these approaches through strategies like artificial potential fields and hybrids have not consistently surpassed reactive methods, especially in real-time scenarios.

Reactive strategies excel in navigating unfamiliar environments, leveraging their simplicity, adaptability to uncertainty, efficient behavior and real-time performance, often outperforming classical methodologies. Meta-heuristic methodologies revolutionize path planning by iteratively generating candidate solutions and selecting the best-fit trajectory for execution, encompassing various methods like genetic algorithm, simulated annealing, Tabu search, particle swarm optimization, ant algorithm, bacterial foraging optimization and bee algorithms. Yet, despite their advantages over classical methods, these approaches are not devoid of limitations. Genetic algorithms [9], while effective, can face challenges in complex environments due to their reliance on population-based optimization, potentially struggling with computational intensity in scenarios with extensive search spaces. Simulated annealing [10] might face limitations in swiftly adapting to rapidly changing environments due to its gradual cooling process, potentially leading to suboptimal paths or slower convergence in dynamically evolving scenarios. Tabu search [11] might struggle in navigating complex and high-dimensional



Source(s): Authors' own work

Figure 1.
Classification of navigation methods for mobile robots

spaces due to its reliance on memory structures, potentially limiting its efficiency in certain intricate environments. Particle swarm optimization [12] can suffer from premature convergence and getting stuck in local optima, hindering its ability to thoroughly explore complex search spaces efficiently. Ant algorithms [13] might struggle with scalability and large search spaces due to their reliance on pheromone trails, potentially leading to suboptimal solutions or increased computational requirements. While bacterial foraging optimization (BFO) [14] presents strengths in exploration and optimization, its sensitivity to parameters, slower convergence and potential challenges in dynamic environments might limit its suitability for real-time and highly dynamic robotics. Bee algorithms [15] might face challenges in handling dynamic environments efficiently due to their reliance on fixed communication patterns among agents, potentially limiting their adaptability to real-time changes. The firefly algorithm [16], although an effective optimization technique in certain contexts, presents drawbacks in path planning and obstacle avoidance for mobile robotics. The algorithm's performance can be affected by parameter tuning, and finding the right balance between exploration and exploitation can be challenging.

Fuzzy logic, while adept at managing uncertainties and enabling adaptive decision-making for MR in intricate terrains [17], faces limitations in computational intensity due to complex rule bases, challenges in representing dynamic uncertainties, dependency on expert knowledge and struggles in highly dynamic settings, requiring potential integration with other methods to address these constraints.

Neural networks, known for their ability to learn complex patterns [18], face challenges in MR path planning and obstacle avoidance due to reliance on extensive training data, potential interpretability issues, susceptibility to overfitting or underfitting in diverse environments and computational complexity, calling for hybrid or complementary approaches to mitigate these limitations.

The A* algorithm, known for its efficiency in finding near-optimal paths [19], faces challenges in scaling to complex environments with high-dimensional spaces or intricate obstacles, potentially struggling in dynamic settings or when heuristic estimates are inaccurate.

Reinforcement learning, applicable in various environments, particularly in path planning, features Q-learning as a prevalent algorithm, creating state-action pairs with associated Q-values denoting anticipated rewards for actions in specific states [20]. The learning process involves the agent navigating through trial and error, initially exploring random actions and assessing their resulting rewards. Over time, it identifies rewarding actions within specific states while balancing the trade-off between trying new actions (exploration) and choosing known rewarding ones (exploitation) [21]. In path planning, the agent is incentivized for finding collision-free paths but penalized for actions resulting in collisions [22]. Despite its potency in path planning, Q-learning's computational demands and parameter selection are crucial considerations. Reinforcement learning stands out for path planning due to its adaptability in diverse environments, showcasing versatility and adeptness in acquiring navigation skills, particularly in challenging and dynamic settings [23]. Reinforcement learning, while beneficial for path planning, poses challenges like computational complexity, demanding substantial resources, requiring careful hyperparameter selection and involving time-consuming learning iterations to develop effective navigation strategies in varied environments [24].

In summary, diverse path planning algorithms offer unique advantages for robot navigation in unknown environments yet face persistent challenges: ensuring safety in unknown environments using general rules, redundant calculations in consecutive searches due to a lack of prior knowledge, slow convergence rates and limited dynamic path planning. Addressing these challenges requires an algorithm enabling efficient navigation in unknown

environments with strong generalization, rapid convergence and reduced computational load, such as our introduced DRQL algorithm.

3. Designing the modified Q-learning algorithm with four key components

The Q-learning algorithm has four important elements: state, action, reward and Q-table. The definition of these elements is specific to the application context of the Q-learning algorithm. Figure 2 presents reinforcement learning (RL), which is a machine learning paradigm focused on learning optimal actions through interactions within an environment to achieve a certain goal. Its framework comprises the following components:

(1) State (s): The current situation or configuration of the environment that the agent perceives. It represents all the relevant information necessary for decision-making.

(2) Action (a): The choices available to the agent in each state. Actions lead to transitions from one state to another.

(3) Reward (r): The immediate feedback the agent receives from the environment after taking an action in a certain state. It quantifies the desirability of the agent's action. For example, if a robot navigates to a hazardous location near obstacles, he will receive a negative reward. On the other hand, reaching the destination gives a positive reward.

(4) Q-function (Q): Estimates the value of taking a particular action in each state. It helps in decision-making by evaluating action values.

The RL system operates through an iterative process, where the agent interacts with the environment, observes states, takes actions, receives rewards and updates its policy and value functions based on these experiences. The objective is for the agent to learn an optimal policy that maximizes cumulative rewards over time. This learning occurs through exploration (trying new actions) and exploitation (using learned knowledge to select the best actions).

Running example: In Figure 3, the map's size is specified as 4×4, and the robot's objective is to navigate from its starting position (1,1) to the goal (4,4). The simulation allows the robot movement in four directions – forward, backward, left and right – while avoiding collisions with obstacles or the environment's edges. For recording purposes,

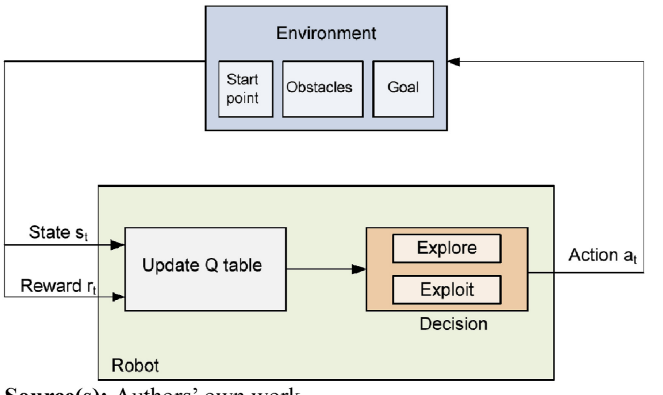
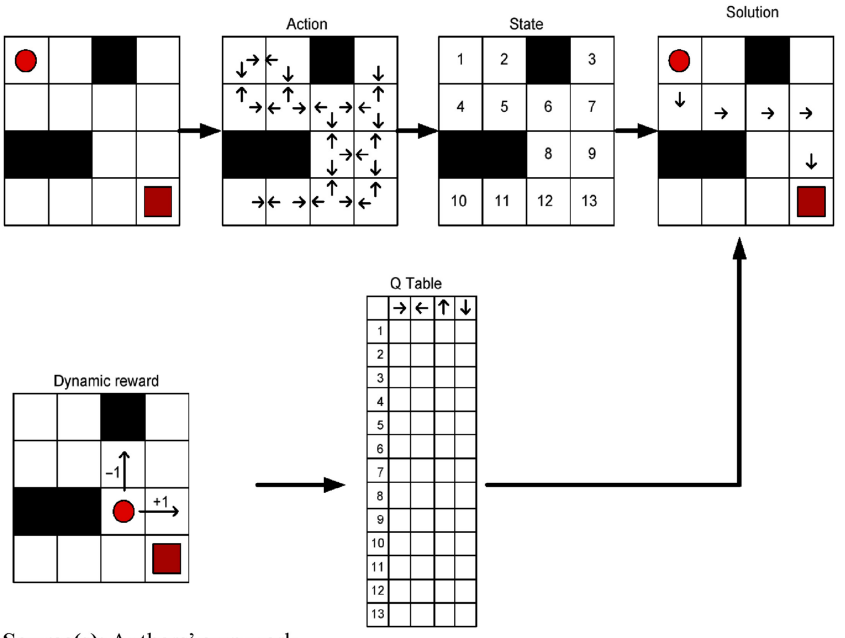


Figure 2.
Framework of reinforcement learning system

Source(s): Authors' own work



Source(s): Authors' own work

Figure 3.
The robot's path planning based on Q-learning algorithm

state nodes are numbered based on the robot's exploration sequence. The environmental state nodes are categorized into four distinct types: the starting point, forbidden points, the target point and free points. Figure 3 illustrates forbidden points where the obstacle space is depicted by the black area.

3.1 State space

In this system, the state space is not fixed but dynamically determined by the path meshing range of the robot. Essentially, the state encapsulates the precise configuration or location of the robot within this meshing range. Consequently, as the robot traverses its environment, the state undergoes continuous updates to accurately mirror its evolving position and configuration, ensuring a real-time representation that enables efficient path planning and obstacle avoidance within the given dynamic context.

3.2 Action space

The robot's action space comprises eight distinct movements, each representing a specific direction for navigation within its environment. These actions encompass standard movements, including left, right, forward and downward motions, denoted as Actions 1 to 4, respectively. Additionally, the robot can execute diagonal movements to cover a broader range, with Actions 5 and 6 corresponding to forward-left and forward-right movements and Actions 7 and 8 representing downward-left and downward-right operations, enabling the robot to efficiently navigate through its surroundings with flexibility and adaptability.

$$A = \{a1; a2; \ldots ; a8\}.$$

3.3 Reward mechanism

A reward function tailored to the agent's specific real-world application is crafted through an analysis of the agent's state following its action selection:

(1) t: This is the number of times the algorithm has been executed.

(2) $r(s_t, a_t)$: This is the reward that the agent receives at the current iteration. The reward function can be designed to encourage the agent to take certain actions or to discourage it from taking other actions.

(3) s_t: This is the agent's current state. The state of the agent is a representation of its environment and its position in the environment.

An enhanced reward function, considering obstacle proximity, has been proposed to offer a more dynamic evaluation of the agent's performance across various situations. Specifically, the enhanced reward function considers two main scenarios:

(1) Scenario 1: The agent's actions lead it closer to the target position without encountering obstacles. In this case, the agent is rewarded with a small positive value.

(2) Scenario 2: The agent's actions lead it away from the target position without collisions. In this case, the agent is punished with a small negative value.

The dynamic reward can be calculated as shown in equation (1).

$$r(s_t, a_t) = \begin{cases} C_1, S_t = S_g \\ -C_1, d_{obs} = 0 \\ C_2 * \frac{d_{t+1}}{D}, d_t < d_{t+1}, d_{obs} \neq 0 \\ -C_2 * \frac{d_{t+1}}{D}, d_t > d_{t+1}, d_{obs} \neq 0 \end{cases} \quad (1)$$

where.

(1) d_t: This is the distance between the agent and the goal location. The reward value increases as the agent approaches the target location.

(2) d_{obs}: This is the distance between the agent and the nearest obstacle. The closer the agent is to an obstacle, the lower the reward value.

(3) s_g: This is the target state that the agent aims to reach. The target state can be a specific location or it can be a certain condition that the agent must meet.

(4) C_1 and C_2: These are constant values that symbolize the rewards obtained by the agent during its interactions with the environment. These values can be adjusted to make the reward function sensitive to the agent's actions ($C_1 > C_2$).

Equation (1) can be interpreted as follows:

(1) The agent is rewarded for getting closer to the target location;

(2) The agent is penalized for getting closer to an obstacle and

(3) The reward for getting closer to the target location is greater than the penalty for getting closer to an obstacle.

3.4 Q-table

The Q-table's rows represent state nodes within the environment, and the columns represent possible actions in each of these states. The Q-table's dimension is m*n, with m representing the number of states and n representing the number of actions. To obtain the Q-table, the Bellman equation was used to emulate the agent's learning trajectory within the Q-learning algorithm, as described in equation (2):

$$Q(s_{t+1}, a_{t+1}) = (1 - \alpha) * Q(s_t, a_t) + \alpha * [r(s_t, a_t) + \gamma * \max_a(Q(s_{t+1}, a))] \quad (2)$$

where:

(1) $Q(s_t, a_t)$ is the expected value of taking action a_t in-state s_t;

(2) α is the learning rate;

(3) γ is the discount factor;

(4) $r(s_t, a_t)$ is the reward received for taking action a_t in-state $s_{t\ and}$

(5) $Max\ _a(Q(s_{t+1}, a))$ is the maximum expected value of taking any action in state s_{t+1}

The DRQL approach for MR path planning that we propose in Algorithm 1, is designed to facilitate MR path planning in dynamic environments. It takes as input the goal point (S_g) and environmental information (O_j) and produces learning values (Qm*n). In each episode, the algorithm initializes Qm*n for all state-action pairs, randomly selects an initial state (s_t) and enters a loop with a maximum iteration count (N). During each iteration, it assesses whether s_t is safe or not. If s_t is unsafe, it selects an action based on obstacle avoidance knowledge; if safe, it employs a dynamic exploration strategy, where it may choose a random action with probability ξ or select the best action using Qm*n with probability (1-ξ). The chosen action is executed, resulting in a reward, and the algorithm updates Qm*n accordingly. This process repeats until either the goal state (s_g) is reached or the maximum iteration count (N) is exhausted and the final Qm*n values are returned to guide MR path planning in dynamic environments.

The DRQL algorithm iteratively updates the $Q(s_t; a_t)$ values by applying the Bellman equation, considering the rewards collected at each state node. As the Q-table converges, the robot acquires the ability to plan the shortest path through its accumulated knowledge.

The parameters used in the described algorithm 1 for path planning and obstacle avoidance are presented in Table 1.

Variable	Meaning
ξ	This variable represents a random number generated between [0, 1]. It is utilized to determine whether the robot should make a random action choice or opt for an informed decision based on the *Q*-values
x	x, a random number in the range [0, 1], is compared to ξ. When $x < \xi$, the algorithm opts for a random action, encouraging exploration. When $x \geq \xi$, it chooses the best action using *Q*-values, favoring a more informed decision-making process

Source(s): Author's own work

Table 1. Parameters and their meaning

Algorithm 1. DRQL for MR path planning

Input:

Goal point, S_g

Environmental information, Oj

Output:

Learning values, Qm*n

Begin

Initialize Qm*n(s_t, a_t) to 0 for all state-action pairs (s_t, a_t).

For each episode

Set s_t to a random state from the state set S.

i=0

Generate a random number, ξ between [0, 1]

Generate a random number, x between [0, 1]

While s_t is not equal to s_g and (i<N)

Evaluate the robot's current state s_t

If it is near an obstacle then

Choose an action a_t to avoid obstacle

end if

If s_t is at a safe level then

If $x < \xi$ then

Choose a random a_t in s_t.

else

Select the best a_t in s_t using Qm*n.

End if

End if

Execute action a_t and receive reward r.

Determine the new state s_{t+1}.

Based on Equation (2), update Qm*n(s_t, a_t)

i = i+1

Set s_t to s_{t+1}.

End while

End for

Return Qm*n.

End

4. Simulation

We compare our proposed dynamic reward solution DRQL against a conventional approach that relies on a static reward mechanism. To facilitate this comparison, we begin by elucidating the methodology employed for computing the static reward. The static reward can be calculated as shown in equation (3):

$$r(s_t, a_t) = \begin{cases} C_1, S_t = S_g \\ -C_1, d_{obs} = 0 \\ C_2, d_t < d_{t-1}, d_{obs} \neq 0 \\ -C_2, d_t > d_{t-1}, d_{obs} \neq 0 \end{cases} \quad (3)$$

(1) C_1, St = Sg: This condition indicates that if the current state (S_t) is equal to the goal state (S_g), a static reward C_1 is assigned. This reward is given when the robot reaches its intended destination without any proximity to obstacles.

(2) -C_1, $d_{obs} = 0$: If the distance between the robot and the nearest obstacle (d_{obs}) is zero, meaning there is no separation between the robot and the obstacle, a static reward C_1 is given. This situation represents a collision with an obstacle, so a reward is assigned.

(3) C_2, $d_t < d_{t-1}$, $d_{obs} \neq 0$: This condition states that if the current distance to the target location (d_t) is less than the previous distance (d_{t-1}) and there is some non-zero distance (d_{obs}) between the robot and the nearest obstacle, a static reward C_2 is assigned. This reward encourages the robot to get closer to the target while avoiding obstacles.

(4) -C_2, $d_t > d_{t-1}$, $d_{obs} \neq 0$: Conversely, if the current distance to the target location (d_t) is greater than the previous distance (d_{t-1}) and there is some nonzero distance (d_{obs}) between the robot and the nearest obstacle, another static reward C_2 is given. This reward motivates the robot to move further away from obstacles while still progressing towards the target.

In summary, these static rewards provide a way to guide the robot's behavior based on its current state, proximity to the goal and proximity to obstacles, helping it make decisions that lead to successful path planning and obstacle avoidance.

Table 2 shows the parameters incorporated into Equation (2), i.e. α and γ, and the variables used in Algorithm 1, such as N and ξ. The specific values are set to control various aspects of the learning and decision-making process of the robot, such as the balance between exploration and exploitation.

Variable	Value
Learning rate α	0.02
Discount factor γ	0.9
Number of repetitions N	1000
Greedy factor ξ	0.1

Source(s): Author's own work

Table 2. Parameters and their values

In all experiments, the configuration of MR was always represented by q(x; y), with x and y representing coordinates. Table 3 describes three configurations ($Config_1$, $Config_2$ and $Config_3$) of the test environment used for the experiment. All these configurations share a uniform starting point and endpoint (goal). Each configuration of the test environment consists of n obstacles called $O_j(x_j, y_j)$, each obstacle positioned at a coordinate point (x_j, y_j). These carefully designed test environments have been designed to thoroughly evaluate and analyze the effectiveness, performance and accuracy of the DRQL learning algorithm.

To maintain the reliability of the experimental results, we iteratively reproduced the same experiment 20 times. Afterward, we calculate average values that include the number of moving steps and the average reward. Table 4 provides data on the number of moving steps required for the MR to reach the endpoint under these three configurations ($Config_1$, $Config_2$ and $Config_3$).

In this context, the "Moving Step Count for SRQL" represents the number of moving steps the robot takes when using a static reward mechanism. The SRQL, which refers to the static reward Q-learning algorithm, is very similar to the algorithm previously introduced DRQL, which differs only in the methodology of the calculation of the reward, which is calculated statically. Similarly, the "Moving Step Count for DRQL" indicates the number of moving steps when employing a dynamic reward mechanism.

The key observation here is that the dynamic reward approach results in significantly fewer steps (lower moving step count) compared to the static reward approach across all three configurations ($Config_1$, $Config_2$ and $Config_3$). This reduction in steps indicates that the dynamic reward strategy is more efficient in guiding the robot to reach the goal point. The percentages provided (e.g. 9.23, 20.78 and 12.05%) represent the extent to which dynamic reward reduces the number of steps compared to static reward for each respective configuration.

Table 5 presents data on the average reward achieved by a MR under different configurations ($Config_1$, $Config_2$ and $Config_3$).

The key observation here is that, for each configuration, the dynamic reward approach results in a lower average reward compared to the static reward approach. This difference is quantified as a percentage reduction in the average reward. For instance, in the first configuration ($Config_1$), the average reward for the dynamic reward approach is 12% lower than that for the static reward approach.

However, it's important to note that despite the lower average reward values, the dynamic reward strategy is more efficient in terms of path planning and obstacle avoidance, as

Table 3.
Configuration of the test environment with information about obstacles

Test environment	Obstacle O(xi, yi)
$Config_1$	(3,2); (4,8); (10, 3)
$Config_2$	(2,5); (3,3); (6,2); (7,9); (8,1); (10,9); (12,4)
$Config_3$	(1,3); (4,2); (5,8); (9,10); (12,6)

Source(s): Author's own work

Table 4.
Moving step count

Configuration	$Config_1$	$Config_2$	$Config_3$
Moving step count for SRQL	71	96	83
Moving step count for DRQL	52	65	67

Source(s): Author's own work

indicated by the lower number of moving steps (moving step count) required to reach the endpoint, which was discussed in a previous explanation. This suggests that the dynamic reward approach provides a better trade-off between reward and path efficiency.

Figure 4 presents results comparing the performance of two algorithms, DRQL and SRQL, over a series of iterations. The values represent the accumulated rewards or Q-values for both algorithms at specific iteration points.

At the beginning of the iterations (Iteration 0), both DRQL and SRQL start with an initial reward value of 0, which is expected as they have not yet learned the optimal policy.

As the iterations progress, both algorithms begin to learn and improve their Q-values; however, notable differences emerge. DRQL demonstrates a more rapid learning rate compared to SRQL. By iteration 40, DRQL achieves a Q-value of 1.3, while SRQL reaches 0.9, indicating that DRQL learns faster and accumulates higher rewards.

The divergence between the two algorithms continues throughout the iterations. DRQL consistently outperforms SRQL in terms of accumulated rewards, suggesting that the incorporation of dynamic reward mechanisms enhances the learning efficiency and performance of the algorithm compared to the static reward approach. These results emphasize the effectiveness of dynamic rewards in guiding the learning process of an agent in a reinforcement learning environment.

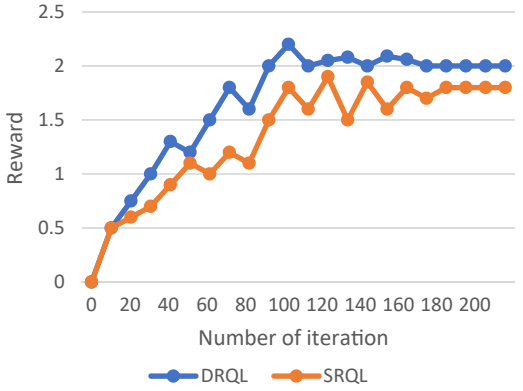
5. Conclusion

We introduce innovative solutions to the persistent challenges of long convergence rates and extensive planning cycles encountered by MR in unknown and complex environments. Using dynamic rewards in the Q-learning framework called the DRQL, this approach significantly improves navigation performance. DRQL strategically combines the inherent learning

Configuration	$Config_1$	$Config_2$	$Config_3$
Average reward for SRQL	3.5	2.9	3.7
Average reward for DRQL	2.9	2.4	3.1
Reduction	12%	28%	8%

Source(s): Author's own work

Table 5. The average reward for the same configuration



Source(s): Authors' own work

Figure 4. Comparative analysis of reward evolution patterns in SRQL and DRQL algorithms

capability of Q-learning with an adaptive and dynamic reward mechanism. As a result, this methodology reduces the need for a comprehensive exploration and accelerates convergence rates. Extensive simulations confirm DRQL's superiority over the existing methods. It manifests accelerated convergence toward an optimal action strategy that requires shorter time and exploration steps. In addition, DRQL has consistently obtained higher average rewards, which means that it is more efficient in road planning in an unknown environment.

In our ongoing research, we are exploring avenues to refine the DRQL approach by optimizing parameters (α, γ) in Equation (2), focusing on adaptive techniques to dynamically adjust these values based on real-time feedback.

References

[1]. Wang B, Liu Z, Li Q, Prorok A. Mobile robot path planning in dynamic environments through globally guided reinforcement learning. IEEE Robotics Automation Lett. 2020; 5(4): 6932-9. doi: 10.1109/lra.2020.3026638.

[2]. Zhou Y, van Kampen EJ, Chu Q. Hybrid hierarchical reinforcement learning for online guidance and navigation with partial observability. Neurocomputing. 2019; 331: 443-57. doi: 10.1016/j.neucom.2018.11.072.

[3]. Chang L, Shan L, Jiang C, Dai Y. Reinforcement based mobile robot path planning with improved dynamic window approach in unknown environment. Autonomous Robots. 2021; 45(1): 51-76. doi: 10.1007/s10514-020-09947-4.

[4]. Ajeil FH, Ibraheem IK, Azar AT, Humaidi AJ. Grid-based mobile robot path planning using aging-based ant colony optimization algorithm in static and dynamic environments. Sensors. 2020; 20(7): 1880. doi: 10.3390/s20071880.

[5]. Chang L, Shan L, Jiang C, Dai Y. Reinforcement based mobile robot path planning with improved dynamic window approach in unknown environment. Autonomous Robots. 2021; 45(1): 51-76. doi: 10.1007/s10514-020-09947-4.

[6]. Salama OA, Eltaib ME, Mohamed HA, Salah O. RCD: radial cell decomposition algorithm for mobile robot path planning. IEEE Access. 2021; 9: 149982-92. doi: 10.1109/access.2021.3125105.

[7]. Ravankar AA, Ravankar A, Emaru T, Kobayashi Y. HPPRM: hybrid potential based probabilistic roadmap algorithm for improved dynamic path planning of mobile robots. IEEE Access. 2020; 8: 221743-66. doi: 10.1109/access.2020.3043333.

[8]. Orozco-Rosas U, Montiel O, Sepúlveda R. Mobile robot path planning using membrane evolutionary artificial potential field. Appl Soft Comput. 2019; 77: 236-51. doi: 10.1016/j.asoc.2019.01.036.

[9]. Choueiry S, Owayjan M, Diab H, Achkar R. Mobile robot path planning using genetic algorithm in a static environment. In: 2019 Fourth International Conference on Advances in Computational Tools for Engineering Applications (ACTEA). IEEE; 2019. p. 1-6.

[10]. Shi K, Wu Z, Jiang B, Karimi HR. Dynamic path planning of mobile robot based on improved simulated annealing algorithm. J Franklin Inst. 2023; 360(6): 4378-98. doi: 10.1016/j.jfranklin.2023.01.033.

[11]. Khaksar W, Hong TS, Sahari KSM, Khaksar M, Torresen J. Sampling-based online motion planning for mobile robots: utilization of Tabu search and adaptive neuro-fuzzy inference system. Neural Comput Appl. 2019; 31(S2): 1275-89. doi: 10.1007/s00521-017-3069-6.

[12]. Zhang L, Zhang Y, Li Y. Mobile robot path planning based on improved localized particle swarm optimization. IEEE Sensors J. 2020; 21(5): 6962-72. doi: 10.1109/jsen.2020.3039275.

[13]. Miao C, Chen G, Yan C, Wu Y. Path planning optimization of indoor mobile robot based on adaptive ant colony algorithm. Comput Ind Eng. 2021; 156: 107230. doi: 10.1016/j.cie.2021.107230.

[14]. Muni MK, Parhi DR, Kumar PB. Improved motion planning of humanoid robots using bacterial foraging optimization. Robotica. 2021; 39(1): 123-36. doi: 10.1017/s0263574720000235.

[15]. Szczepanski R, Tarczewski T. Global path planning for mobile robot based on Artificial Bee Colony and Dijkstra's algorithms. In: 2021 IEEE 19th International Power Electronics and Motion Control Conference (PEMC). IEEE; 2021. p. 724-30.

[16]. Li F, Fan X, Hou Z. A firefly algorithm with self-adaptive population size for global path planning of mobile robot. IEEE Access. 2020; 8: 168951-64. doi: 10.1109/access.2020.3023999.

[17]. Cherroun L, Boumehraz M, Kouzou A. Mobile robot path planning based on optimized fuzzy logic controllers. New Dev Adv Robot Control. 2019: 255-83. doi: 10.1007/978-981-13-2212-9_12.

[18]. Yu J, Su Y, Liao Y. The path planning of mobile robot by neural networks and hierarchical reinforcement learning. Front Neurorobotics. 2020; 14: 63. doi: 10.3389/fnbot.2020.00063.

[19]. Wang X, Liu Z, Liu J. Mobile robot path planning based on an improved A* algorithm. In: International Conference on Computer Graphics, Artificial Intelligence, and Data Processing (ICCAID 2022), SPIE, 12604; 2023. p. 1093-8.

[20]. Low ES, Ong P, Cheah KC. Solving the optimal path planning of a mobile robot using improved Q-learning. Robotics Autonomous Syst. 2019; 115: 143-61. doi: 10.1016/j.robot.2019.02.013.

[21]. Lluvia I, Lazkano E, Ansuategi A. Active mapping and robot exploration: a survey. Sensors. 2021; 21(7): 2445. doi: 10.3390/s21072445.

[22]. Pang L, Cao Z, Yu J, Zhang W, Chen X. A collision-free person-following approach based on path planning. In: 2020 IEEE International Conference on Mechatronics and Automation (ICMA). IEEE; 2020. p. 327-31.

[23]. Dong Y, Zou X. Mobile robot path planning based on improved DDPG reinforcement learning algorithm. In: 2020 IEEE 11th International Conference on software engineering and service science (ICSESS). IEEE; 2020. p. 52-6.

[24]. Zhu K, Zhang T. Deep reinforcement learning based mobile robot navigation: a review. Tsinghua Sci Technology. 2021; 26(5): 674-91. doi: 10.26599/tst.2021.9010012.

Corresponding author
Atef Gharbi can be contacted at: atef.gharbi@nbu.edu.sa