Dynamic Rewards in Reinforcement Learning for Robotic Navigation

Ahmed Badi Alshammari
Department of Computer Sciences, Faculty of Computing and Information Technology, Northern Border University, Rafha, Saudi Arabia
ahmed.badi@nbu.edu.sa (corresponding author)

Received: 7 May 2025 | Revised: 23 May 2025 and 17 June 2025 | Accepted: 18 June 2025

 DOI: https://doi.org/10.48084/etasr.11986

ABSTRACT

The paper presents a new reinforcement learning method, called Adaptive Q-learning with Dynamic Reward (AQDR), for efficient route planning of mobile robots operating in a partially known and unknown environment. Traditional Q-learning techniques are often limited in their adaptability due to slow convergence in dynamic environments. To overcome these limitations, AQDR combines an adaptive reward mechanism that adjusts in real time based on the distance of the robot to the obstacle and the target position. This dynamic feedback allows for better informed decision making and reduces unnecessary exploration. The proposed algorithm was evaluated against the Q-learning method based on static rewards by performing simulated experiments in different environmental configurations. The results show that AQDR consistently exceeds the baseline in terms of convergence speed, path efficiency, and adaptability. These results highlight the potential of dynamic reward design to improve learning performance and robustness of reinforcement learning navigation systems.

Keywords-reinforcement learning; Q-learning; dynamic reward; mobile robot navigation; path planning; adaptive algorithms

I. INTRODUCTION

Path planning is a key challenge for the safe and efficient navigation of mobile robots, especially in unknown or dynamic environments that may include unexpected obstacles. There is a wide range of path-planning strategies, and the best approach is often determined by the specific characteristics of the robot and the operating environment [1]. Unlike fully known environments, where the previously mapped route selection is simplified, partially known environments are more complex due to incomplete spatial data and uncertainty surrounding hidden obstacles. The most challenging scenario occurs in an entirely unknown environment, where the robot must cross unknown territory without prior environmental knowledge. Each scenario requires specialized path-planning technology that adapts to the level of environmental knowledge and associated uncertainty. In a known environment, path-planning is relatively simple [2]. However, under partially or completely unknown conditions, the robot must first construct or improve its environmental map in real time before calculating a collision-free trajectory [3]. Mostly known environments are often the most applicable in real-world scenarios, enabling the integration of previous knowledge with real-time perception, thereby facilitating adaptive navigation in unknown areas.

The selection of appropriate path planning algorithms is crucial to determine the ability of mobile robots to navigate safely, efficiently, and reliably. These algorithms are generally divided into two main types: static and dynamic. Static planning uses the assumption that the environment is unchanged and a predetermined path can be calculated before the robot begins movement. On the other hand, dynamic route planning is designed to handle changing environments, including moving obstacles, and requires robots to adjust.

In general, static methods are easier to implement, but they are often insufficient for dynamic or unpredictable environments where real-time adaptability is essential to efficient navigation. To overcome these challenges, this study presents the Adaptive Q-learning with Dynamic Reward (AQDR) algorithm that strengthens traditional Q-learning by incorporating a contextually sensitive and adaptive reward framework. This framework allows robots to navigate unknown environments by learning from interactions and adjusting to environmental changes during execution. Experimental evaluations show that AQDR surpasses classical Q-learning in terms of convergence speed, path optimization, and adaptability. This work contributes threefold. First, the introduction of dynamic rewards uses partial environmental information to guide exploration, static rewards represent internal node characteristics, and dynamic rewards vary in proximity to targets, reducing inefficient exploration. Second, the AQDR learning process is moving through three stages, initial, mixed, and final exploration, to overcome the stagnation observed in traditional Q-learning. Third, experimental validation shows the effectiveness of AQDR in managing complex path-planning tasks in different unstructured environments.

A wide-ranging study of existing research revealed that mobile robotics navigation strategies are divided into two major categories: classical approaches and reactive techniques. Traditional approaches to robotic path planning include algorithms such as cell decomposition [4], road mapping planning [5], and Artificial Potential Field (APF) methods [6]. Despite their basic state, these techniques have several limitations, including computational overhead, tolerance to local minima, and reduced robustness in uncertain or dynamic environments. Their dependence on accurate sensors and static mapping data makes them less practical for real-time deployment in unstructured or evolving scenarios. In contrast, reactive strategies provide more adaptable solutions and excel in real-time navigation in a non-local environment. These methods are robust under uncertainty and are characterized by a simple and rapid response, often exceeding traditional ones in practical implementation.

The emergence of metaheuristic algorithms brought a new dimension to robotic path planning [7]. These techniques iteratively generate several candidate paths and optimize for the most appropriate paths. Remarkable approaches include Genetic Algorithms (GA), Simulated Annealing (SA), Tabu searches, Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), Bacterial Foraging Optimization (BFO), and bee algorithms. These methods often yield better results than conventional ones, but they also present remarkable challenges. For example, GA [8] is computationally intensive and can be less efficient in high-dimensional spaces. SA [9] can have slow convergence due to the coolness factor. The Tabu search [10] may be ineffective in navigating complex topologies due to dependence on memory-based mechanisms. PSO [11] tends to preconverge and limits exploration. ACO [12] is biologically inspired but struggles with scalability in large environments. Similarly, the BFO [13] and Bee algorithms [14] can face problems in dynamic or real-time applications due to slow convergence and fixed communication models, respectively.

Other soft-computing methods have also been extensively studied. Fuzzy logic systems, known for their ability to manage uncertainty and enable adaptive decision-making [15], face challenges related to the complexity of the rule base, the dependence on expert knowledge, and the reduction of performance in dynamic contexts. Neural networks [16] provide strong pattern recognition capabilities, but often require large datasets, are susceptible to over- or under-fitting, and are difficult to interpret, especially in real-time navigation tasks. Among graph-based methods, the A* algorithm [17] is widely recognized for its almost optimal performance in static environments. However, because of the increase in computational demands and the sensitivity to heuristic accuracy, it is no longer effective in high-dimensional or dynamic scenarios. Reinforcement Learning (RL) has attracted a lot of attention in recent years, especially in the field of robotic navigation planning [18]. Q-learning is one of the most well-known RL algorithms, and agents associate actions with expected rewards through repeated interactions with the environment [19]. This learning process balances exploration and exploitation to improve policy over time, rewards safe paths for discovery, and penalizes collisions.

Although Q-learning shows great potential to navigate dynamic and unknown environments, its practical application is hampered by computational complexity, high parameter selection sensitivity, and slow convergence during the initial learning phase [20]. In summary, although there is a variety of path-planning algorithms, each with distinctive strengths, there are still important challenges in the navigation of unknown environments. These include maintaining safety under uncertainty, reducing redundant calculations, achieving rapid convergence, and ensuring real-time adaptability. To overcome these limitations, algorithms must have strong generalizability, adaptive dynamic learning, and computational efficiency. To address these challenges, this study proposes AQDR, an advanced reinforcement learning framework designed to improve robotic navigation performance in complex environments, including parts or entirely unknown ones.

II. DESIGN OF THE ENHANCED Q-LEARNING FRAMEWORK

Q-learning is a core RL method that operates based on four key elements: state, action, reward, and Q-value functions. These components are defined according to the specific application context and the environment the algorithm operates. The core components of Q-learning are defined as follows:

1. State (s): A representation of the current configuration of the environment as perceived by the agent. It encapsulates all the relevant information necessary to make informed decisions at a given time.
2. Action (a) is a set of possible operations that agents can perform in each state. As a result of executing an action, the current state changes to a new one, and the environment changes.
3. Rewards (r) involve a scalar feedback signal that the agent receives after acting in a particular state. It serves to evaluate the immediate utility of the action, encouraging positive rewards to perform desired behavior (e.g., achieving goals) and deterring negative rewards to achieve undesirable outcomes (e.g., collision with obstacles).
4. Q-function (Q): A value function that estimates the cumulative reward expected for an action in a particular state under an optimal policy. The Q-function guides agents to select actions that maximize long-term rewards through iterative updates based on experiential learning.

The RL process is inherently iterative, including repeated cycles of interaction. The agent monitors its current state, selects and performs actions, receives feedback in the form of rewards, and updates the Q-values accordingly. Over time, agents converge on an optimal policy that produces the highest cumulative return through exploration (new actions to find better solutions) and exploitation mechanisms (using existing knowledge to maximize rewards). To illustrate this process, consider the example shown in Figure 1, where an agent (a mobile robot) navigates through a 4×4 grid environment. The robot initiates its traversal at position (1, 2) to reach the target location at (4, 4). The robot can go in four directions: up, down, left, and right. The environment is structured as a grid

consisting of different state categories, including initial position, barriers that cannot be crossed, targets, and free cells that can be reached. The obstacles are visualized as black cells, and the numbering of state nodes reflects the sequence in which the robot explores the environment. During this simulation, robots must learn from their environmental interactions to develop a collision-free trajectory. The Q-learning model is the basis for the proposed improvements described in the following sections, which introduce dynamic reward adjustment and structured learning phases to improve convergence and adaptation in complex or unknown environments.

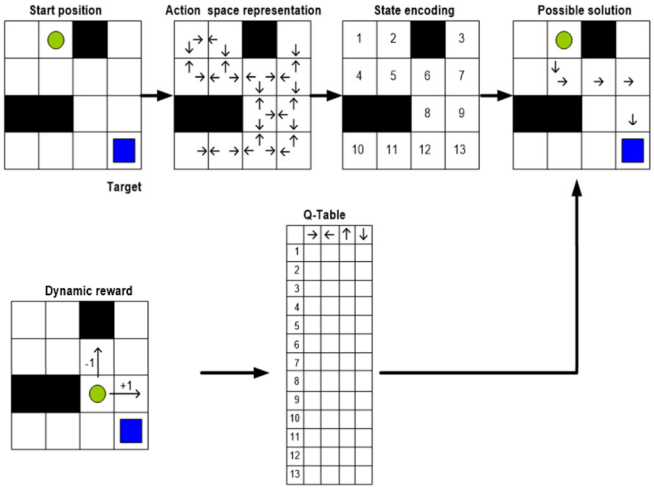

Fig. 1. Illustration of Q-learning-based path planning with dynamic reward in a grid environment.

A. Dynamic Environment Encoding

In the proposed framework, the state space is not static, but dynamically generated based on the current available area of the robot. Each state represents the immediate spatial configuration of the robot within this adaptive region. As the robot moves through the environment, the state is continuously updated, accurately displaying its current position and direction in real time. This dynamic modeling of the state space ensures accurate and responsive path planning, accommodates real-time changes in the operating environment of the robot, and allows effective obstacle prevention and trajectory optimization in a dynamic and partially known environment. The robot's action space consists of eight different motion commands, each corresponding to a specific movement direction in the environment. These include the four main cardinal directions of the left, right, forward, and backward actions from a_1 to a_4, allowing robots to choose from a wide range of paths, adapt to obstacles, and dynamic environmental constraints more flexibly:

$$A = \{a_1, a_2, a_3, a_4\}$$

B. Reward Mechanism

A customized reward function was developed to evaluate the performance of the agent based on its current state and selected actions in the environment. The RL framework uses the following components:

- t indicates the iteration step of the learning process.
- s_t represents the current state of the agent in the t iteration and encodes the position of the agent and its environment.
- $r(s_t, a_t)$ defines the scale reward received by the agent when executing the action at state s_t.

To improve learning efficiency and responsiveness, a dynamic reward function is introduced that identifies both the proximity of targets and the avoidance of obstacles. This approach distinguishes between two major behavioral scenarios:

- Scenario 1: If the agent moves towards the goal without obstacles, it receives a small positive reward to encourage the behavior directed toward the goal.
- Scenario 2: If the agent moves away from the goal and avoids obstacles, it earns a small negative reward that discourages ineffective but safe movements. The dynamic reward is modelled by the following equation:

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

where

- d_t is the straight-line distance from the target position defined as s_g. It is the distance between the agent and the target position. As d_t decreases, the agent receives a higher reward.
- d_{obs} is the spatial separation between the agent and the nearest detected obstacle. Smaller d_{obs} are punished because of increased collision risk.
- s_g: The state of the target or the goal that the agent seeks to achieve.
- C_1, C_2 : Static constants that scale the reward and penalty components, ensuring that the reward for the progress of the goal exceeds the penalty for the proximity of obstacles (i.e., $C_1 > C_2$). C_1 and C_2 were empirically set to 10 and 5, respectively, to prioritize goal proximity over obstacle avoidance.

This formula ensures that learning agents are encouraged to reduce distances from the target while maintaining a safe distance from obstacles, thereby achieving efficient and accident-free navigation in complex environments.

C. Q-Table Representation and Learning Strategy

In the context of Q-learning frameworks, Q-tables are two-dimensional data structures that encode the expected utility of specific actions in corresponding states. Formally, the table is organized into rows m and n, where m represents the total number of discrete environment states, and n represents the possible actions available to each state. Each entry $Q(s_t, a_t)$ estimates the expected cumulative reward associated with the action at state s_t, considering the subsequent optimal policy.

The Q value is improved by iterative refinement using the Bellman optimality equation, a recursive formulation that updates the value estimates based on the expected value of the observed reward and future state. This update process is mathematically expressed as:

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

where $\alpha \in [0, 1]$ is the learning rate of the new information, $\gamma \in [0, 1]$ is the discount factor that determines the weight of future rewards, $r(s_t, a_t)$ is the immediate reward obtained after the execution of the action in state s_t, and $\max_a Q(s_{t+1}, a)$ is the highest predicted Q-value for the next state that reflects the best choice of the agent.

To overcome the shortcomings of conventional Q-learning in dynamic and partially observable environments, this study introduces AQDR, which is specially adapted for path planning in mobile robotics. AQDR incorporates an adaptive exploration and exploration strategy along with a context-sensitive reward mechanism that adapts to the robot's relative position regarding both the target and the surrounding obstacles.

Adaptive Q-Learning with Dynamic Rewards for Mobile Robot Navigation
Inputs:
  S_g: Target goal state
  O_j: Environmental information
Output:
  $Q_{m\times n}$: Learned Q-values
Procedure:
1. Initialize $Q(s_t, a_t) = 0$ for all state-action pairs.
2. For each episode:
  a. Randomly initialize the current state s_t.
  b. Set iteration counter $i = 0$.
  c. Generate two random values $\xi, x \in [0, 1]$.
3. While $s_t \neq S_g$ and $i < N$:
  a. Evaluate the safety of the current state s_t.
  b. If near obstacle → choose obstacle-avoidance action.
  c. Else:
    i. If $x < \xi$ → choose a random action (exploration).
    ii. Else → select optimal action using $Q(s_t, a_t)$.
  d. Execute action a_t, observe reward r, transition to new state s_{t+1}.
  e. Update Q-value using the Bellman update rule.
  f. Set $s_t = s_{t+1}$, increment i.
4. End loop and return $Q_{m\times n}$.

TABLE I. ALGORITHM PARAMETERS

Parameter	Description
ξ	A threshold value in [0,1] that controls the trade-off between exploration and exploitation.
x	A randomly generated number in [0,1] used to determine the action selection strategy. If $x < \xi$, exploration is triggered; otherwise, exploitation is preferred.

AQDR incrementally refines the Q-table by taking into account dynamic rewards, obstacle proximity, and adaptive exploration, ultimately enabling the robot to identify efficient and safe paths. As the learning process progresses, the Q-values converge, allowing the robot to autonomously generate near-optimal trajectories in real-time.

III. SIMULATION SETUP AND COMPARATIVE EVALUATION

To evaluate the performance of AQDR, a comparative analysis was performed against a traditional Q-learning model utilizing a fixed, static reward scheme. This section presents the method adopted to define the static reward structure and provides a comprehensive overview of the parameters applied across different experimental configurations.

A. Static Reward Formulation

The static reward function is defined using a set of fixed conditions that evaluate the agent's progress toward the goal and its interaction with surrounding obstacles. The reward assignment is governed by the conditions outlined below and encapsulated in:

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

- Goal Achievement Condition: If $S_t = S_g$, i.e., the agent's current state S_t coincides with the goal state S_g, a positive reward of magnitude C_1 is assigned. This condition signifies successful goal achievement in the absence of obstacle proximity.
- Collision Condition: If the agent's distance to the nearest obstacle $d_{obs} = 0$, indicating a collision, a negative reward of magnitude $-C_1$ is applied to penalize unsafe navigation behavior.
- Progress Condition: If $d_t < d_{t-1}$ and $d_{obs} \neq 0$, where d_t and d_{t-1} are the current and previous distances to the goal, respectively, a moderate reward C_2 is granted. This reward encourages movement toward the goal while maintaining a safe distance from obstacles.
- Deviation Condition: If $d_t > d_{t-1}$ and $d_{obs} \neq 0$, the agent is penalized with a small negative reward $-C_2$, discouraging regressive actions away from the target despite maintaining obstacle clearance.

This static reward structure provides a rule-based framework that promotes efficient pathfinding and collision avoidance based on discrete state evaluations.

B. Learning and Decision Parameters

The simulation employs specific hyperparameters to control the learning process, including the learning rate α, the discount factor γ, the maximum number of learning episodes N, and the greedy exploration parameter ξ. These parameters are detailed in Table II. The values were tuned using grid search on a small subset of environments.

TABLE II. LEARNING PARAMETERS

Variable	Value	Description
Learning rate α	0.02	Controls the influence of new information on Q-values
Discount factor γ	0.9	Weighs the importance of future rewards
Iterations N	1000	Maximum allowed learning iterations per episode
Greedy factor ξ	0.1	Probability of selecting a random action (exploration)

C. Experimental Evaluation and Comparative Results

To ensure the statistical reliability and reproducibility of the simulation results, each experimental configuration was executed over 20 independent trials. The results were then averaged to derive meaningful performance metrics, including the number of movement steps and the average cumulative reward. This method minimizes the influence of stochastic variation inherent in RL processes.

1) Movement Efficiency: Step Count Comparison

Figure 2 illustrates the learning performance of two RL algorithms, AQDR and SRQL (Standard Reward Q-learning), in terms of the number of steps required to reach the goal across 50 episodes. ADQR (green curve) shows a consistently decreasing trend, indicating that the agent using the dynamic reward mechanism learns more efficiently, reducing the number of steps needed to reach the goal more rapidly and stabilizing at a lower step count. This suggests better convergence and path optimization. In contrast, the SRQL (red curve) shows a slower decrease with higher overall step counts and more fluctuation, reflecting slower learning and suboptimal path decisions due to reliance on static rewards.

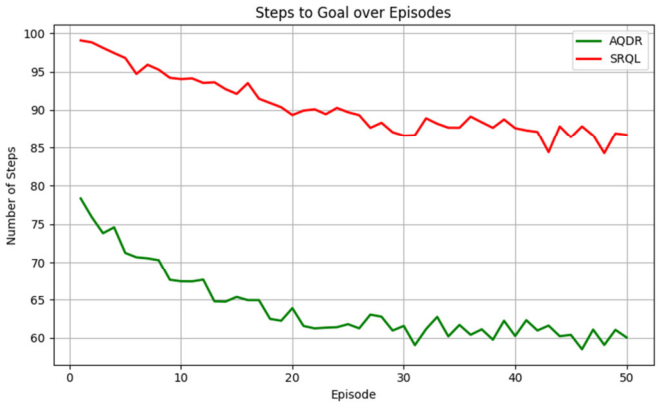

Fig. 2. Comparison of learning efficiency: AQDR vs. SRQL based on steps to goal.

The dynamic reward strategy implemented in AQDR significantly enhances learning efficiency and path planning performance compared to the static reward method used in SRQL. This shows that incorporating adaptive reward mechanisms enables the agent to learn faster and execute more efficient trajectories in fewer steps. In rare scenarios where obstacles fully encircle the goal, AQDR may fail to converge. Future work will integrate backtracking mechanisms.

2) Reward Learning Dynamics

Figure 3 shows that both algorithms exhibit an upward trend, indicating progressive learning and performance improvement over time. AQDR, represented by the blue line, consistently achieves higher average rewards compared to SRQL throughout most episodes, especially in the later stages. This performance suggests that AQDR benefits from faster convergence and superior adaptability, largely attributed to its dynamic reward mechanism that offers more informative feedback. In contrast, SRQL, shown by the orange line, also improves over time, but at a slower pace and with a lower reward ceiling, reflecting less efficient learning dynamics.

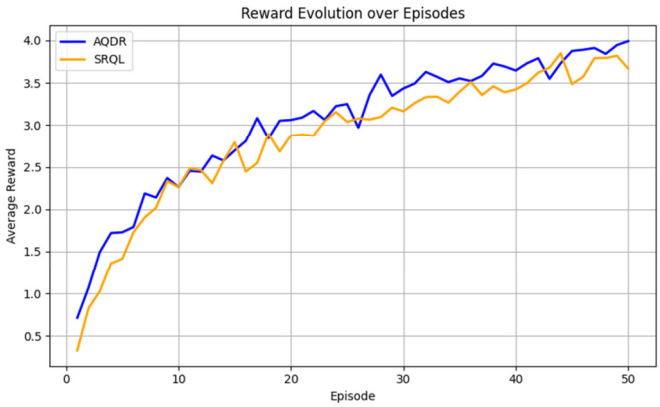

Fig. 3. Comparison of average reward accumulation: AQDR vs. SRQL over training episodes.

IV. CONCLUSION

This study presented an enhanced RL strategy, AQDR, that aims to address critical limitations associated with conventional Q-learning in the context of mobile robot navigation, particularly in unknown or dynamically changing environments. The AQDR framework integrates the traditional Q-learning algorithm with a context-aware adaptive reward mechanism, allowing more efficient path planning by reducing reliance on exhaustive environmental exploration. By dynamically modulating the reward signal based on the robot's proximity to the goal and nearby obstacles, the AQDR algorithm accelerates convergence toward an optimal policy and minimizes unnecessary exploration. Extensive simulation-based evaluations across multiple test scenarios demonstrate that AQDR consistently outperformed the static reward-based Q-learning variant. The results highlight faster convergence, shorter path lengths, and higher efficiency in obstacle avoidance, thus validating the robustness and generalization capabilities of the proposed approach.

AQDR consistently achieved higher average rewards and lower step counts, indicating enhanced learning performance and decision-making efficiency. These findings confirm the effectiveness of using dynamic feedback mechanisms in RL to support intelligent autonomous navigation. The modular structure of the AQDR framework allows for its extension to larger multi-agent systems and more complex environments with minimal modifications. Furthermore, as the algorithm relies on local observations and adaptive rewards, it can operate efficiently in decentralized settings where full environmental knowledge is not available, making it suitable for deployment in real-world scenarios such as robotic swarms, disaster response, and autonomous vehicle coordination. These aspects underscore the practicality and generalizability of the proposed method beyond simulation.

Future work aims to enhance AQDR by developing adaptive parameter tuning strategies, particularly for the learning rate α and discount factor γ, allowing these parameters to evolve dynamically in response to real-time agent performance. This advancement will further strengthen the algorithm's adaptability and responsiveness in complex and partially observable environments.

ACKNOWLEDGMENT

The authors extend their appreciation to the Deanship of Scientific Research at Northern Border University, Arar, KSA, for funding this research work through project number NBU-FFR-2025-1181-01.

REFERENCES

[1] L. Liu, X. Wang, X. Yang, H. Liu, J. Li, and P. Wang, "Path planning techniques for mobile robots: Review and prospect," *Expert Systems with Applications*, vol. 227, Oct. 2023, Art. no. 120254, https://doi.org/10.1016/j.eswa.2023.120254.

[2] U. Orozco-Rosas, K. Picos, J. J. Pantrigo, A. S. Montemayor, and A. Cuesta-Infante, "Mobile Robot Path Planning Using a QAPF Learning Algorithm for Known and Unknown Environments," *IEEE Access*, vol. 10, pp. 84648–84663, 2022, https://doi.org/10.1109/ACCESS.2022.3197628.

[3] Y. Wang, X. Li, J. Zhang, S. Li, Z. Xu, and X. Zhou, "Review of wheeled mobile robot collision avoidance under unknown environment," *Science Progress*, vol. 104, no. 3, Jul. 2021, Art. no. 00368504211037771, https://doi.org/10.1177/00368504211037771.

[4] O. A. A. Salama, M. E. H. Eltaib, H. A. Mohamed, and O. Salah, "RCD: Radial Cell Decomposition Algorithm for Mobile Robot Path Planning," *IEEE Access*, vol. 9, pp. 149982–149992, 2021, https://doi.org/10.1109/ACCESS.2021.3125105.

[5] A. N. A. Rafai, N. Adzhar, and N. I. Jaini, "A Review on Path Planning and Obstacle Avoidance Algorithms for Autonomous Mobile Robots," *Journal of Robotics*, vol. 2022, no. 1, 2022, Art. no. 2538220, https://doi.org/10.1155/2022/2538220.

[6] J. Luo, Z. X. Wang, and K. L. Pan, "Reliable Path Planning Algorithm Based on Improved Artificial Potential Field Method," *IEEE Access*, vol. 10, pp. 108276–108284, 2022, https://doi.org/10.1109/ACCESS.2022.3212741.

[7] Y. Xu, Q. Li, X. Xu, J. Yang, and Y. Chen, "Research Progress of Nature-Inspired Metaheuristic Algorithms in Mobile Robot Path Planning," *Electronics*, vol. 12, no. 15, Jan. 2023, Art. no. 3263, https://doi.org/10.3390/electronics12153263.

[8] M. N. Ab Wahab *et al.*, "Improved genetic algorithm for mobile robot path planning in static environments," *Expert Systems with Applications*, vol. 249, Sep. 2024, Art. no. 123762, https://doi.org/10.1016/j.eswa.2024.123762.

[9] K. Shi, Z. Wu, B. Jiang, and H. R. Karimi, "Dynamic path planning of mobile robot based on improved simulated annealing algorithm," *Journal of the Franklin Institute*, vol. 360, no. 6, pp. 4378–4398, Apr. 2023, https://doi.org/10.1016/j.jfranklin.2023.01.033.

[10] L. Qin and Z. Fan, "Robot Path Planning Based on Tabu Particle Swarm Optimization Integrating Cauchy Mutation," in *2024 14th International Conference on Information Science and Technology (ICIST)*, Chengdu, China, Dec. 2024, pp. 244–253, https://doi.org/10.1109/ICIST63249.2024.10805364.

[11] B. Tao and J. H. Kim, "Mobile robot path planning based on bi-population particle swarm optimization with random perturbation strategy," *Journal of King Saud University - Computer and Information Sciences*, vol. 36, no. 2, Feb. 2024, Art. no. 101974, https://doi.org/10.1016/j.jksuci.2024.101974.

[12] H. Huang, G. Tan, and L. Jiang, "Robot Path Planning Using Improved Ant Colony Algorithm in the Environment of Internet of Things," *Journal of Robotics*, vol. 2022, no. 1, 2022, Art. no. 1739884, https://doi.org/10.1155/2022/1739884.

[13] Z. Wang, J. Peng, S. Ding, and School of Electrical Engineering, Zhengzhou University, No. 100 of Science Avenue, Zhengzhou 450001, China, "A Bio-inspired trajectory planning method for robotic manipulators based on improved bacteria foraging optimization algorithm and tau theory," *Mathematical Biosciences and Engineering*, vol. 19, no. 1, pp. 643–662, 2021, https://doi.org/10.3934/mbe.2022029.

[14] Y. Cui, W. Hu, and A. Rahmani, "A reinforcement learning based artificial bee colony algorithm with application in robot path planning," *Expert Systems with Applications*, vol. 203, Oct. 2022, Art. no. 117389, https://doi.org/10.1016/j.eswa.2022.117389.

[15] A. Hentout, A. Maoudj, and M. Aouache, "A review of the literature on fuzzy-logic approaches for collision-free path planning of manipulator robots," *Artificial Intelligence Review*, vol. 56, no. 4, pp. 3369–3444, Apr. 2023, https://doi.org/10.1007/s10462-022-10257-7.

[16] J. Wang, J. Liu, W. Chen, W. Chi, and M. Q. H. Meng, "Robot Path Planning via Neural-Network-Driven Prediction," *IEEE Transactions on Artificial Intelligence*, vol. 3, no. 3, pp. 451–460, Jun. 2022, https://doi.org/10.1109/TAI.2021.3119890.

[17] C. Han and B. Li, "Mobile Robot Path Planning Based on Improved A* Algorithm," in *2023 IEEE 11th Joint International Information Technology and Artificial Intelligence Conference (ITAIC)*, Chongqing, China, Dec. 2023, pp. 672–676, https://doi.org/10.1109/ITAIC58329.2023.10408799.

[18] L. Dong, Z. He, C. Song, and C. Sun, "A review of mobile robot motion planning methods: from classical motion planning workflows to reinforcement learning-based architectures," *Journal of Systems Engineering and Electronics*, vol. 34, no. 2, pp. 439–459, Apr. 2023, https://doi.org/10.23919/JSEE.2023.000051.

[19] Q. Zhou, Y. Lian, J. Wu, M. Zhu, H. Wang, and J. Cao, "An optimized Q-Learning algorithm for mobile robot local path planning," *Knowledge-Based Systems*, vol. 286, Feb. 2024, Art. no. 111400, https://doi.org/10.1016/j.knosys.2024.111400.

[20] A. N. Kadhim and M. S. Salim, "Enhancing Navigation Efficiency in Robotics with PRM-DDPG," *Engineering, Technology & Applied Science Research*, vol. 15, no. 3, pp. 24077–24086, Jun. 2025, https://doi.org/10.48084/etasr.11213.