Robotica (2026), **44**, pp. 52–70
doi:10.1017/S0263574725103093

CAMBRIDGE UNIVERSITY PRESS

RESEARCH ARTICLE

A novel path planning algorithm based on synergistic bidirectional optimization

Shixuan Qi[1], Haofei Lu[2], Ziyi Wang[2], Xin Gao[1], Siyao Chen[1], Yuanjian Liu[1] and Weigang Wang[1]

[1]College of Electronic and Optical Engineering & College of Flexible Electronics (Future Technology), Nanjing University of Posts and Telecommunications, Nanjing, Jiangsu, China
[2]Portland Institute, Nanjing University of Posts and Telecommunications, Nanjing, Jiangsu, China
Corresponding author: Weigang Wang; Email: wangwg2020@163.com

Received: 4 June 2025; **Revised:** 22 October 2025; **Accepted:** 4 December 2025; **First published online:** 2 January 2026

Keywords: path planning; mobile robot; Q-learning; LPA*; bidirectional optimization

Abstract

Path planning, as a critical component of mobile robotic systems, significantly impacts operational efficiency and energy consumption ratios. State-of-the-art algorithms often suffer from inadequate real-time adjustment capability, insufficient dynamic environment adaptation, and suboptimal computational efficiency. To resolve these limitations, we propose a bidirectionally optimized path planning algorithm named Bidirectional Q-learning LPA* (BQ-LPA*), which incorporates three key innovations. Specifically, to enhance the global search capability of the LPA* framework, we replace fixed heuristic functions with a Q-learning-driven adaptive heuristic mechanism, which improves path quality through dynamic heuristic weighting and update strategies. Additionally, to improve the convergence rate and sample efficiency of Q-learning in complex environments, we propose integrating the LPA* framework to provide prior knowledge guidance, which can effectively minimize redundant exploration attempts by informed pathfinding initialization. Moreover, the Q-learning method inherently faces dimensionality challenges in high-dimensional continuous spaces, which manifest as action space congestion, storage bottlenecks, and computational inefficiency. To mitigate these risks, we devise an LPA*-based space discretization strategy that can reduce action space dimensionality and preserve the path feasibility. Experimental results show that, compared with mainstream path planning algorithms, BQ-LPA* achieves higher accuracy and faster convergence in mobile robot path planning.

1. Introduction

In recent years, the rapid evolution of technologies and the increasing application demands have propelled the widespread deployment of mobile robots across diverse sectors such as transportation, emergency response, military operations, and precision agriculture [1–3]. Path planning, a pivotal technology in mobile robotics, empowers autonomous obstacle avoidance and efficient navigation toward target destinations [4, 5]. However, existing path planning solutions often rely on semi-autonomous frameworks that require manual waypoint configuration. This human-dependent approach introduces subjectivity into the planning process, which results in suboptimal resource utilization and increased operational costs in terms of time, labor, and materials. As intelligent technologies now permeate all industries, it is critically important to develop and implement intelligent path planning algorithms for mobile robots [6–8].

High-performance path planning algorithms are defined by their optimization speed, trajectory simplicity, and robust obstacle avoidance capabilities, all of which directly enhance robotic operational efficiency and safety [9–11]. According to existing literature [12–14], the most commonly used path planning algorithms include graph-search algorithms [A star(A*),D star(D*)], probabilistic sampling-based algorithms [Probabilistic Roadmaps (PRM), Rapidly-exploring Random Trees (RRT)], mathematical model-driven algorithms [Artificial Potential Field (APF), Dynamic Window Approach

© The Author(s), 2026. Published by Cambridge University Press.

https://doi.org/10.1017/S0263574725103093 Published online by Cambridge University Press

(DWA)], and bio-inspired algorithms [Ant Colony Optimization (ACO), Genetic Algorithm (GA)]. The A* algorithm, which incorporates greedy heuristics for prioritized node expansion, demonstrates strong search capabilities but suffers from efficiency degradation in complex environments due to its reliance on grid-based representations. RRT-based methods, though adaptable to unstructured environments, are computationally intensive and probabilistically inclined to produce suboptimal paths rather than globally optimal solutions. Genetic algorithms excel at solving multi-objective optimization problems but face challenges such as high computational complexity, parameter sensitivity, and premature convergence during iterative evolution.

In response to these limitations, researchers have conducted thorough investigations and proposed multiple enhanced path planning algorithms. In ref. [15], an improved A* algorithm was developed by incorporating adaptive weighting into its heuristic function and applying the Douglas-Pucker algorithm to eliminate redundant path nodes, which significantly reduces explored nodes. However, this method remains inapplicable to dynamic obstacles and demands substantial computational resources. The study [16] introduced the RDT-RRT algorithm, which integrates G1/G2 Hermite interpolation to minimize path oscillations and discontinuities and enhance trajectory smoothness. A dynamic collision detection mechanism further improved its obstacle avoidance capability in dynamic environments. Nevertheless, the algorithm requires complex parameter tuning and exhibits inadequate dynamic obstacle prediction. Reference [17] proposed a GA-based method that combines graph theory to optimize flight paths for mobile aerial base stations (MABS) in disaster scenarios, which ensures maximal user equipment coverage. While effective in complex environments, this approach suffers from high computational complexity and strong environmental dependency. The hybrid algorithm in ref. [18], which synergizes an enhanced RIME optimization with the DWA, demonstrates robust global search capability and operational efficiency. However, its performance in dynamic environments heavily relies on global path accuracy, coupled with prolonged computation times.

Considering the limitations of existing methods, such as low planning efficiency in dynamic environments, excessive reliance on precise environmental modeling, and deficiencies in optimizing high-dimensional state spaces, reinforcement learning-based path planning algorithms have emerged as a transformative solution. Reference [19] introduced an enhanced Q-learning algorithm incorporating radial basis function neural networks (RBFNNs), which demonstrates significant improvements in both convergence speed and safety performance. The method exhibits particularly outstanding performance in complex marine environments. However, it suffers from extended computation time and exhibits excessive dependency on parameter configuration. Reference [20] significantly improved the convergence rate and obstacle avoidance capability of Q-learning in complex environments by combining the flower pollination algorithm with tabular methods. Nevertheless, challenges such as high computational complexity, limitations in dynamic scenarios, and parameter sensitivity require further refinement. The multi-Q-table path planning (MQTPP) algorithm introduced in ref. [21] exhibits its advantages in planning efficiency and real-time responsiveness, especially in complex urban settings and dynamic obstacle handling. Nevertheless, it faces limitations in path optimality, computational overhead during initial learning phases, and scalability to larger environments. The study [22] proposed a prioritized weight-enhanced Q-learning variant that excels in convergence speed, path length optimization, and dynamic obstacle navigation. However, its applicability is constrained by Q-table scalability issues, overestimation biases, and computational burdens in early learning stages. In ref. [23], an improved Q-learning algorithm was developed by optimizing Q-table exploration strategies, resetting initial Q-values, and incorporating dynamic exploration factors. That enables real-time path adaptation in stochastic airport environments. Yet, challenges remain, including extended training durations, initial state dependency, and local optima entrapment.

In summary, merely optimizing existing path planning algorithms through incremental strategy integration fails to effectively address critical challenges in complex environments, such as high computational complexity, poor stability, and suboptimal planning efficiency. To overcome these limitations, we propose a bidirectional optimization algorithm named BQ-LPA* that synergistically combines the Q-learning method and the LPA* framework. In our study, bidirectional optimization denotes the

https://doi.org/10.1017/S0263574725103093 Published online by Cambridge University Press

synergistic collaboration between the Q-learning and LPA*, where their complementary strengths mitigate mutual limitations to achieve holistic performance enhancement. The principal contributions of this work are summarized as follows:

(1) To avoid unnecessary local updates and improve path quality, we propose a Q-learning-driven method that dynamically adapts the heuristic function weights in the LPA* framework. Our approach enables real-time adaptation of update strategies based on environmental characteristics. Furthermore, a composite heuristic function integrating objectives such as path length, obstacle avoidance success rate, and smoothness is designed, with a feedback mechanism enhancing the heuristic's estimation accuracy through iterative calibration.

(2) The prior knowledge provided by LPA*, specifically global path nodes as initial subgoals for Q-learning, guides the exploration process to reduce redundant search and accelerate convergence. Concurrently, LPA*'s incremental search data is integrated into Q-learning's experience replay buffer, which can significantly improve sample utilization efficiency.

(3) We address the high-dimensional action selection problem in Q-learning by employing LPA*-based discretization of continuous spaces. The method maps continuous spaces to finite path nodes generated by LPA*, which prevents exponential growth of the action space and reduces its dimensionality. Simultaneously, LPA* dynamically updates nodes in response to environmental changes, which ensures that the discretized action space consistently reflects real-time environmental states and enhances dynamic adaptability.

2. Related work

2.1. Q-learning method

Q-learning [24], a model-free method, is a cornerstone methodology in reinforcement learning. Within Markov decision processes, it demonstrates robust learning capabilities by leveraging action experience to select optimal actions in autonomous systems. Developed on the temporal difference (TD) framework, Q-learning operates as a model-agnostic iterative learning process. Its model-free advantage lies in directly estimating action-value functions without environmental dynamics modeling. The Q-value quantifies the expected cumulative reward when transitioning from the current state to the next one after executing an action. The Q-function formulation integrates state transition probabilities and successor states to evaluate the current state-action pair. Agents receive immediate rewards after action execution, while Q-learning's policy maximizes discounted future rewards. The optimal Q-value is defined as the cumulative rewards of actions executed under the optimal policy, mathematically represented as

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha[r_t + \gamma \max_{\alpha} Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)], \tag{1}$$

where s_t denotes the current state, a_t is the action taken in state s_t, and r_t represents the immediate reward received after executing action a_t. Similarly, s_{t+1} refers to the resulting next state, and a_{t+1} is the action chosen in state s_{t+1}. The learning rate $\alpha(0 < \alpha < 1)$ controls the update magnitude of Q-values, while the discount factor $\gamma(0 < \gamma < 1)$ weights the importance of future rewards relative to immediate ones.

The learning process for each agent starts from a random initial state and selects subsequent actions using either an ε-greedy policy or a Boltzmann distribution strategy. When adopting the ε-greedy strategy with ε initialized to 0.95, the agent prioritizes exploitation by choosing the Q-table-optimal action with 95% probability, while allocating 5% probability to random exploration. To enable agents to explore all possible actions and update each action's $Q(s, a)$, a stochastic action selection policy is implemented. After executing a chosen action, the agent observes the resultant new state. The $Q(s, a)$ corresponding to the preceding state-action pair is subsequently updated according to the maximum Q-value of the new state and the immediate reward received [25]. Based on the updated state, the agent continues selecting actions iteratively until reaching a terminal state.

https://doi.org/10.1017/S0263574725103093 Published online by Cambridge University Press

2.2. LPA* framework

The LPA* (Lifelong Planning A*) framework [26] is an incremental heuristic search algorithm that efficiently updates shortest paths in dynamic environments. Notably, it extends the heuristic search framework of the A* algorithm with incremental update mechanisms, so that environmental changes trigger selective updates to affected regions, which bypass computationally intensive global recomputation. This is enabled by maintaining two key node cost metrics(g and rhs) to track state transitions. $g(u)$ denotes the actual cost, defined as the true path cost from the start node to the node u. Initially, $g(u)$ is set to infinity and dynamically updated to the minimum known cost during the search process. $rhs(u)$ represents a temporary cost, derived from the g-values of predecessor nodes and edge costs, with the cost function defined as

$$rhs(u) = \min_{v \in \text{Predecessors(u)}} (g(v) + c(v, u)), \tag{2}$$

where Predecessors(u) denotes the set of predecessor nodes for node u, and $c(v, u)$ represents the cost of the edge formed by nodes u and v. If u is the start node, $rhs(\text{start}) = 0$.

In dynamic environments, edge costs change due to events such as obstacle appearance or disappearance. When an edge cost changes, all nodes dependent on that edge are marked. For each affected node u, its $rhs(u)$ is recalculated by function (2). If $rhs(u) \neq g(u)$, node u is inserted into priority queue U, after which the g and rhs values of relevant nodes are recomputed until all nodes reach local consistency. The detailed workflow of the LPA* framework is shown in Figure 1.

3. BQ-LPA* algorithm

Bidirectional Q-learning LPA* (BQ-LPA*) is a path planning algorithm that bidirectionally optimizes solutions by integrating an enhanced Q-learning method with the LPA* framework. Its core strength lies in synergistically combining these two algorithms to mutually compensate for their individual limitations.

3.1. Dynamic heuristic strategy adjustment

The fixed heuristic function in the LPA* framework exhibits significant limitations in complex or dynamic environments. Firstly, it fails to adapt to intricate obstacle distributions, which potentially leads to suboptimal detours or local optima entrapment. Secondly, in scenarios with frequent dynamic obstacle changes, the static heuristic function cannot adjust in real time, which results in path conflicts. Local updates may trigger chain reactions, which require multiple iterations for convergence. Additionally, the inflexibility of this function hinders multi-objective optimization during path planning. Incremental update strategies may further trap paths in local optima, which necessitates additional computational corrections.

We replace the fixed heuristic function in the LPA* framework with Q-learning, designing the heuristic function as a weighted sum represented as

$$h(n) = \omega_1 \cdot h_{\text{distance}}(n) + \omega_2 \cdot h_{\text{obstacle}}(n) + \omega_3 \cdot h_{\text{smooth}}(n). \tag{3}$$

Here, ω_1, ω_2, and ω_3 are weights dynamically adjusted by Q-learning, while $h_{\text{diatance}}(n)$, $h_{\text{obstacle}}(n)$, and $h_{\text{smooth}}(n)$ represent distinct heuristic action designs. $h_{\text{diatance}}(n)$ increases the weight on Euclidean distance to prioritize path length reduction. $h_{\text{obstacle}}(n)$ elevates the weight on obstacle distance to emphasize collision avoidance, and $h_{\text{smooth}}(n)$ augments the weight on path smoothness to reduce turning frequency.

We set the initial value of the distance weight ω_1 as

$$\omega_1 = 0.7 \cdot \left(1 - \frac{d_{\text{start_goal}}}{d_{\max}}\right) + 0.3, \tag{4}$$

https://doi.org/10.1017/S0263574725103093 Published online by Cambridge University Press

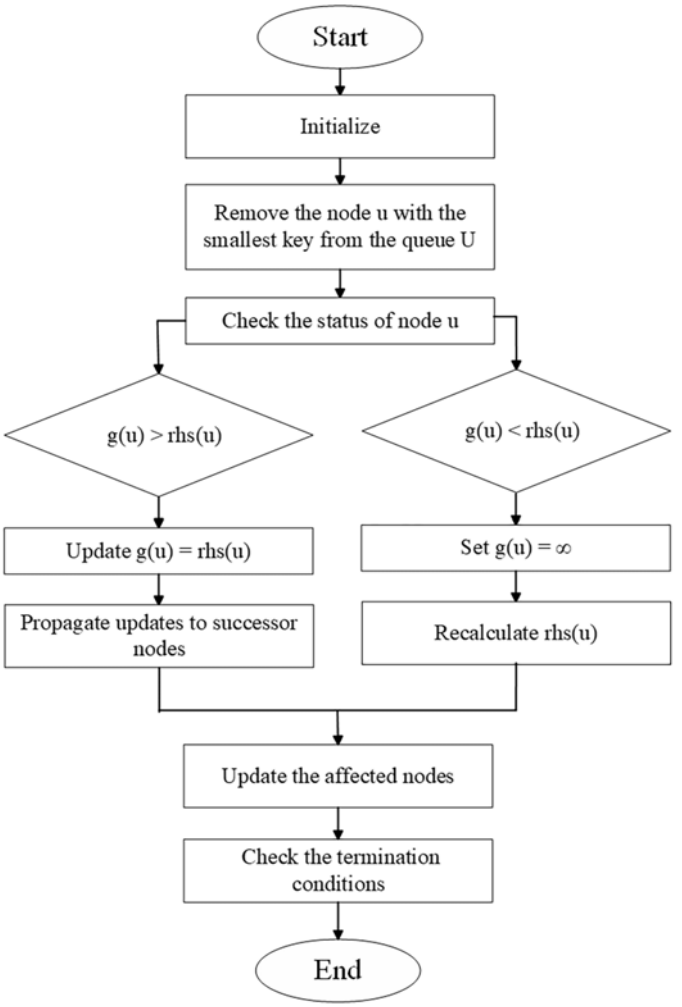


Figure 1. *LPA* framework flow.*

where $d_{\text{start_goal}}$ represents the Euclidean distance from the start point to the goal, while $d_{\max}$ denotes the diagonal distance of the map. The initial value of the obstacle weight ω_2 is set as

$$\omega_2 = \min(0.5, 0.1 + 0.4 \cdot p_{\text{obs}}), \tag{5}$$

where p_{obs} represents the obstacle density within the 3×3 grid neighborhood centered at the start point. Therefore, the computational formula for the initial value of ω_3 is

$$\omega_3 = 1 - \omega_1 - \omega_2 \tag{6}$$

The weights ω_1, ω_2, and ω_3, dynamically adjusted via Q-learning, achieve adaptability to time-varying rewards. In open areas, the distance weight ω_1 is prioritized to ensure the shortest path, while in narrow passages, the obstacle weight ω_2 is increased to guarantee collision avoidance. This contextual adaptability prevents suboptimal detours caused by static, overly conservative obstacle avoidance strategies, effectively mitigating path oscillation issues.

https://doi.org/10.1017/S0263574725103093 Published online by Cambridge University Press

To ensure the convergence of Q-learning, all state-action pairs must be visited infinitely often. In our policy, we employ the ε-greedy strategy to balance exploration and exploitation. Additionally, the learning rate η_t is denoted as

$$\eta_t = \frac{1}{1+t}. \tag{7}$$

It decays according to function (4) and satisfies

$$\sum_{t=1}^{\infty} \eta_t = \infty, \sum_{t=1}^{\infty} \eta_t^2 < \infty. \tag{8}$$

After each path planning iteration, the error between the actual cost and the estimated cost is computed as

$$\Delta C = C_{\text{actual}} - C_{\text{heuristic}}. \tag{9}$$

We need to adjust the weights based on this function. If $\Delta C > 0$, it indicates that the heuristic function underestimates the actual cost, which requires an increase in obstacle avoidance or smoothness weights. Conversely, if $\Delta C < 0$, the heuristic overestimates the cost, which indicates the necessity to increase the distance weighting factor.

Using the L1 norm as the loss function, we obtain

$$L = |\Delta C| . \tag{10}$$

Compute the partial derivative with respect to weight ω_i

$$\frac{\partial L}{\partial \omega_i} = \text{sign}(\Delta C) \cdot \frac{\partial(\Delta C)}{\partial \omega_i}. \tag{11}$$

By the chain rule, we obtain

$$\frac{\partial(\Delta C)}{\partial \omega_i} = \frac{\partial}{\partial \omega_i}\left(C_{\text{actual}} - \sum_j \omega_j h_j\right) = -h_i(n), \tag{12}$$

$$\frac{\partial L}{\partial \omega_i} = -\text{sign}(\Delta C) \cdot h_i(n). \tag{13}$$

The weight update rule is defined as

$$\omega_i = \omega_i - \beta \frac{\partial L}{\partial \omega_i}. \tag{14}$$

Substitute to get

$$\omega_i = \omega_i + \beta \cdot \text{sign}(\Delta C) \cdot h_i(n), \tag{15}$$

$$\omega_i = \omega_i + \beta \cdot \text{sign}(\Delta C) \cdot \frac{\partial h}{\partial \omega_i}. \tag{16}$$

Here, β is the learning rate for weight adjustment. This can be further simplified to

$$\omega_i = \omega_i + \beta \cdot \Delta C \cdot h_i(n). \tag{17}$$

Although any suboptimal path selection caused by noise will result in a large error after execution, this error is immediately fed back through the weight update rule, generating a strong correction signal, thereby ensuring that noise-induced errors are transient and are quickly corrected through the iterative process.

This iterative calibration process continuously minimizes the heuristic estimation error, ensuring that the search priorities of the LPA* algorithm dynamically adapt to multi-objective requirements such as

https://doi.org/10.1017/S0263574725103093 Published online by Cambridge University Press

path length, smoothness, and safety. This mechanism shifts the algorithm's objective from a strict graph-search for the shortest path, which is prone to causing oscillations, toward kinematically optimal paths that are more suitable for physical robots.

Q-learning learns optimal weight combinations across environments through trial-and-error. Specifically, in open-area scenarios, prioritizing the distance weight ω_1 shortens path lengths. In narrow-channel scenarios, increasing the obstacle weight ω_2 prevents collisions. While the LPA* framework establishes a global path framework, Q-learning locally optimizes paths via dynamic weight adjustments. Concurrently, closed-loop error feedback computes ΔC from actual path execution outcomes, then iteratively adjusts weights through error backpropagation to reduce heuristic estimation errors progressively.

Additionally, Q-learning improves real-time performance by dynamically adjusting LPA*'s update policies to reduce redundant computations. The state vector is defined as

$$s = \left[f_{\text{change}}, p_{\text{collision}}, N_{\text{local}}, \eta\right], \tag{18}$$

where f_{change} denotes the environment change frequency, calculated as the ratio of environmental change occurrences to the time window duration. $p_{\text{collision}}$ represents the path conflict probability, calculated as the ratio of recent collision instances to path update attempts. N_{local} indicates the historical local update count, while η is the ratio of the current path cost to the optimal cost.

Regarding the reward mechanism, we set the computational efficiency reward as

$$r_{\text{efficiency}} = \rho \cdot \left(1 - \frac{t_{\text{new}}}{t_{\text{old}}}\right), \tag{19}$$

where t_{old} and t_{new} denote the computation time before and after strategy adjustment, and ρ is the reward coefficient. If an action reduces computation time, the reward is proportional to the time saved. If an action increases path cost, the penalty is set as

$$r_{\text{penalty}} = -\kappa \cdot \left(\frac{c_{\text{new}} - c_{\text{old}}}{c_{\text{old}}}\right), \tag{20}$$

c_{old} and c_{new} represent the costs before and after strategy adjustment, respectively. κ is the reward coefficient, and the total reward function is

$$r = r_{\text{efficiency}} + r_{\text{penalty}}. \tag{21}$$

The update policy mapping for LPA* divides the action execution logic into three strategies. The first strategy (a_1), which updates only the first-order neighbors of the affected nodes, incurs the minimal computational cost. The second (a_2) strategy updates second-order neighbors to balance update scope. The third strategy (a_3) resets the OPEN list to activate global A* search. Q-learning perceives environmental dynamics and algorithm states through state s, while regulating the update scope of LPA* by selecting action a. In low-dynamic environments with small f_{change} values, action a_1 is prioritized to reduce computational overhead. In high-dynamic environments with large f_{change} values, action a_3 is selected for global replanning to prevent local updates from entering infinite loops. When path quality deteriorates($\eta > 1.2$), action a_3 is enforced to escape local optima.

3.2. Incremental experience reuse

Q-learning, a trial-and-error-based reinforcement learning method, faces core challenges in efficiently exploring and identifying optimal policies within complex or high-dimensional environments. In unknown environments, Q-learning requires extensive random exploration, which results in slow convergence and low sample efficiency. LPA*, as an incremental path planning framework, generates and maintains global paths in dynamic environments. By leveraging its output path nodes and cost information as prior knowledge, LPA* significantly accelerates Q-learning's learning process.

During path planning with the LPA* framework, the generated global path comprises a sequence of nodes denoted as $N = \{n_1, n_2, n_3, \ldots, n_k\}$, where each node n contains positional coordinates and path

https://doi.org/10.1017/S0263574725103093 Published online by Cambridge University Press

cost information. LPA* maintains the minimum path cost $C_{\min}$ from the start to the goal while calculating the heuristic estimate $h(n_i)$ representing the remaining cost from each node n_i to the target. We integrate LPA*'s path nodes with real-time environmental data to define the state vector as

$$s = \left[x_{\text{robot}}, x_{\text{goal}}, \{h(n_i)\}_{i=1}^{k}, \text{od}\right], \tag{22}$$

where x_{robot} represents the robot's current position. x_{goal} denotes the target position, and od quantifies obstacle distribution. We replace Q-learning's original action space with the path node set N provided by the LPA* framework and define $A = N$ as the new candidate action set. During action selection, Q-learning chooses action $a \in A$ under state s, corresponding to moving to a specific path node n_i. When successfully reaching node n_i, the reward is proportional to the reduction ratio of path cost, expressed as

$$r_{\text{reach}} = \lambda \cdot \left(1 - \frac{C(n_i)}{C_{\min}}\right), \tag{23}$$

where λ represents the reward coefficient for node arrival, and $C(n_i)$ denotes the actual navigation cost from the current position to node n_i. The path cost $C_{\min}$ from LPA* serves as the reward benchmark for Q-learning. When $C(n_i) < C_{\min}$ (indicating an optimized path), the agent receives higher rewards. Conversely, when $C(n_i) > C_{\min}$ (signifying suboptimal path quality), the system reduces rewards or triggers penalties.

Additionally, a global goal proximity reward is introduced, which grants extra rewards when approaching the final target. The reward is defined as

$$r_{\text{goal}} = \varphi \cdot \left(1 - \frac{\|x_{\text{robot}} - x_{\text{goal}}\|}{\|x_{\text{start}} - x_{\text{goal}}\|}\right), \tag{24}$$

where φ represents the global goal proximity reward coefficient. Importantly, a collision penalty $r_{\text{collision}}$ ($r_{\text{collision}} < 0$) is imposed if the agent collides during movement. Therefore, the total reward function can be represented as

$$r = r_{\text{reach}} + r_{\text{goal}} + r_{\text{collision}}. \tag{25}$$

To enhance efficiency and ensure robust path planning during agent navigation, dynamic node maintenance is implemented through two mechanisms. Node pruning refers to the process where nodes are deprioritized or removed from the graph structure due to prolonged inactivity or collision risks. Dynamic node addition is the process triggered by the LPA* framework's detection of a new path. Following this, the relevant nodes are integrated into set, and the Q-learning action space is updated.

The node sequence generated by LPA* provides structured exploration guidance for Q-learning, while the heuristic cost $h(n_i)$ assists Q-learning in evaluating node priorities to favor those closer to the goal. Q-learning's action space is compressed by a continuous high-dimensional domain to a discrete set of path nodes, which significantly reduces exploration complexity and avoids inefficient search patterns. When environmental changes occur, LPA* updates the path node queue and prompts Q-learning simultaneously adjust its action space, which ensures that prior knowledge remains consistently valid.

After environmental changes trigger LPA* replanning, Q-learning immediately acquires updated global knowledge, avoiding inefficient random exploration. The updated path node sequence N generated by LPA* is used as the initial subgoal set for Q-learning. This structured guidance minimizes redundant search and accelerates the convergence of Q-learning.

3.3. Discretization of high-dimensional action spaces

Q-learning is confronted with the curse of dimensionality in high-dimensional continuous action spaces, where the action space size grows exponentially with dimensionality, rendering memory and computational resources insufficient to support storage and processing demands. Furthermore, random exploration in continuous spaces requires excessive samples, which leads to low sample efficiency, slow convergence rates, and difficulties in discovering globally optimal actions while becoming trapped in

https://doi.org/10.1017/S0263574725103093 Published online by Cambridge University Press

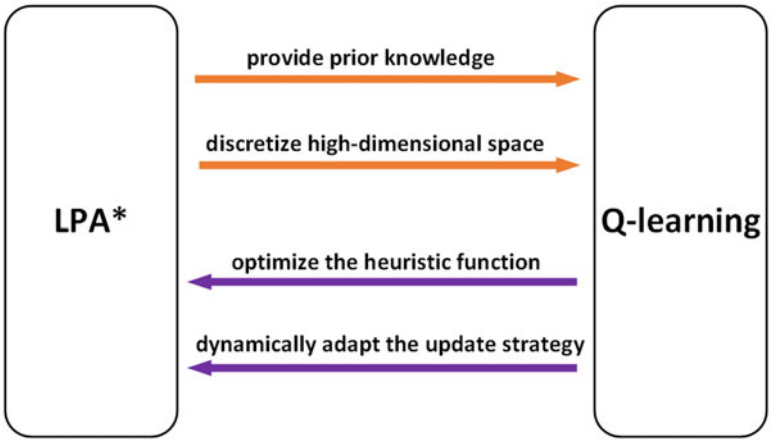


Figure 2. *Schematic diagram of bidirectional optimization.*

local optima. LPA* naturally generates a discrete candidate action set by operating within the continuous space to produce path nodes. This integration of path nodes as Q-learning's discrete action space significantly reduces computational complexity while preserving the feasibility and optimality of path planning solutions.

When the LPA* algorithm operates in continuous space δ for path generation, with path nodes N forming a low-dimensional discrete subspace, the equivalence $N \subset \delta$ holds under subspace dimensionality constraints. By compressing Q-learning's action space from δ to N, the computational complexity is reduced from exponential $O(N^d)$ to linear $O(K)$ with $K = |N|$. This discrete action space $A = N$ enables Q-learning to cover all state-action pairs in polynomial time, satisfying the convergence requirement

$$\lim_{t\to\infty} Q_t(s,a) = Q^*(s,a). \tag{26}$$

The design logic of the reward function can be inferred from the discussion in section 3.2. The reward component r_{reach} incentivizes the selection of path-shortening nodes to optimize path costs, while the penalty term $r_{\text{collision}}$ suppresses hazardous actions to ensure safety. Concurrently, the goal-oriented reward r_{goal} accelerates convergence toward the target. To enhance decision efficiency, node prioritization is implemented through a priority weight dynamically adjusted during exploration, defined as

$$w(n_i) = \frac{1}{C(n_i) + \mu}, \tag{27}$$

where μ serves as a small constant to prevent division-by-zero errors. Q-learning samples actions proportionally to their weights, and thus prioritizes high-quality nodes during selection. Path nodes are dynamically generated by LPA* to satisfy obstacle avoidance and kinematic constraints, which prevents Q-learning from selecting infeasible actions. When the environmental changes occur, LPA* updates the node queue, which maintains real-time validity of the action space while ensuring persistent compliance with safety and motion feasibility requirements.

3.4. Algorithm design and complexity analysis

Although many details of bidirectional optimization have been described in previous sections, to better explain the process of bidirectional optimization, we have prepared Figure 2.

As shown in Figure 2, LPA* optimizes Q-learning mainly in two aspects: providing prior knowledge and discretizing high-dimensional space. In unknown environments, Q-learning requires extensive random exploration, resulting in slow convergence and low sample efficiency. The incremental nature of

https://doi.org/10.1017/S0263574725103093 Published online by Cambridge University Press

LPA* ensures its ability to generate and maintain global paths in dynamic environments. Its output (path nodes and cost information) serves as prior knowledge, significantly accelerating Q-learning's learning process. Simultaneously, Q-learning struggles to find optimal paths in high-dimensional action spaces. Mapping continuous space to LPA*-generated finite path nodes avoids the expansion of high-dimensional action spaces and reduces action space dimensionality.

Q-learning optimizes LPA* primarily in two aspects: optimizing the heuristic function and dynamically adapting the update strategy. The fixed heuristic function of LPA* cannot adapt to complex obstacle distributions, potentially causing circuitous paths or local optima entrapment. Simultaneously, in scenarios with frequent dynamic obstacle changes, fixed heuristics cannot adjust in real-time, leading to path conflicts and chain reactions from local updates that require multiple iterations to converge. Q-learning redesigned the heuristic function as a weighted sum, ensuring optimization of estimation accuracy through feedback mechanisms. Furthermore, Q-learning dynamically adjusts LPA*'s update strategy based on environmental characteristics in real-time, avoiding unnecessary local updates and enhancing path quality.

The operational sequence of the BQ-LPA* algorithm is illustrated in Figure 3. Following parameter initialization, it first generates a global path based on LPA* and computes the remaining cost estimate from each waypoint to the goal to accelerate Q-learning convergence. Subsequently, it executes moves and updates states according to our designed heuristic function. When environmental dynamics change, the algorithm incrementally updates path N using LPA* and expands Q-learning's action space. Concurrently, our dynamic update strategy activates to eliminate redundant computations and enhance real-time performance. These procedures iterate iteratively until the destination is reached.

We now analyze the time complexity and space complexity of BQ-LPA*.

Time complexity: In the classic LPA*, the time complexity is primarily determined by the number of updated nodes U and the total number of nodes V, typically represented as $O(U \log V)$, where $logV$ arises from priority queue operations. In classic Q-learning, the time complexity is $O(N \cdot A)$, where N is the number of states and A is the number of actions. In BQ-LPA*, since Q-learning optimizes the heuristic function and update strategy, U is further reduced, denoted as U' ($U' \ll U$). Simultaneously, LPA* reduces the effective state-action pairs for Q-learning by narrowing the action space (via discretization of high-dimensional spaces) and providing global path guidance, resulting in reduced N' and A' ($N' \ll N$, $A' \ll A$). Thus, the overall time complexity of BQ-LPA* is $O(U' \log V + N' \cdot A')$.

Space complexity: In the classic LPA*, the space complexity is also primarily determined by the number of updated nodes U and the total number of nodes V, typically expressed as $O(V + U)$, approximately $O(V)$. In classic Q-learning, the space complexity is $O(N \cdot A)$, where N is the number of states and A is the number of actions. In BQ-LPA*, since LPA* provides prior knowledge and refines the action space for Q-learning, the number of effective state-action pairs is reduced to N' and A' ($N' \ll N, A' \ll A$). Therefore, the overall space complexity of BQ-LPA* is $O(V) + O(N' \cdot A')$.

Through action space dimensionality reduction, incremental experience reuse, and improvements to heuristics and update strategies, BQ-LPA* significantly enhances its practical feasibility and scalability in complex dynamic environments while reducing complexity.

4. Simulation experiment setup and analysis

4.1. Simulation experiment setup

To validate the advantages of the BQ-LPA* algorithm in path planning, simulations are conducted under the following configurations. The hardware platform utilizes a Windows 11 operating system with an Intel Core i7-13650H processor operating at a base frequency of 2.60 GHz and 16 GB of RAM. MATLAB 2023b serves as the simulation environment. The grid map used in the simulation consists of square cells with a unit length of 1, where white areas designate traversable regions and black areas represent obstacles. In Q-learning, α is the learning coefficient, and reflects the degree to

https://doi.org/10.1017/S0263574725103093 Published online by Cambridge University Press

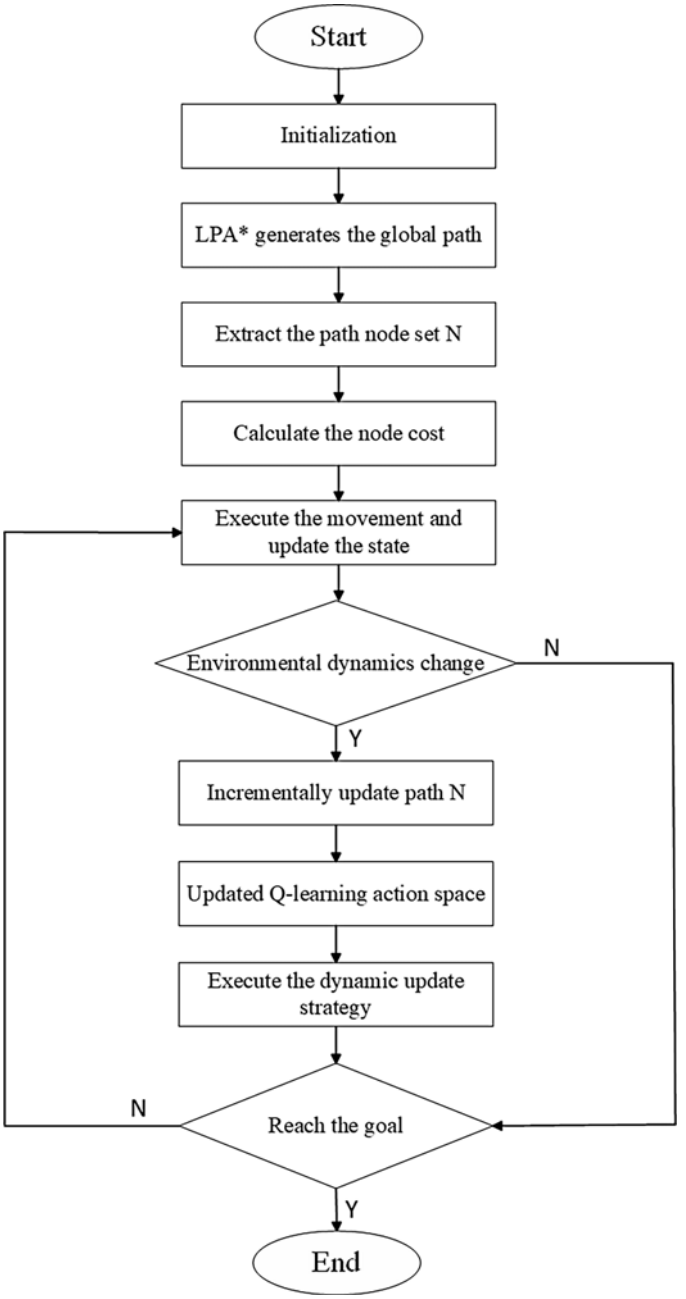


Figure 3. *BQ-LPA* algorithm flow.*

which the previous Q-value is retained. α is set to 0.2 in our study. The variable γ is the discount factor, which represents the degree of importance an agent attaches to future rewards when it receives the current reward, and it is set to 0.8 [27]. The replanning threshold for the LPA* algorithm is set at 0.3, meaning replanning is triggered when environmental changes exceed 30%. The heuristic scaling factor is configured to 1.2, which effectively achieves a balance between path quality and computational efficiency.

https://doi.org/10.1017/S0263574725103093 Published online by Cambridge University Press

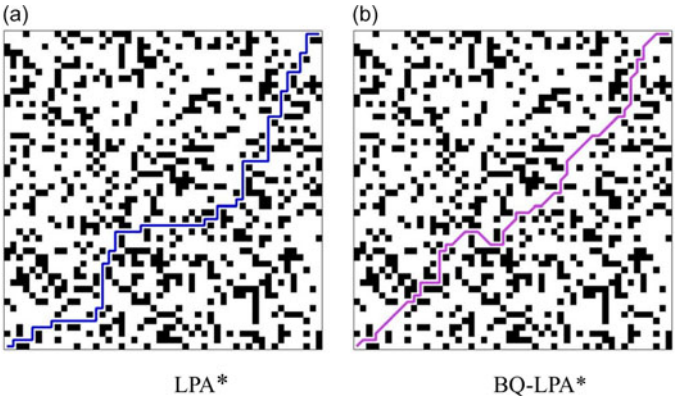


Figure 4. *Comparative results of dynamic heuristic strategy adjustments under 50 × 50.*

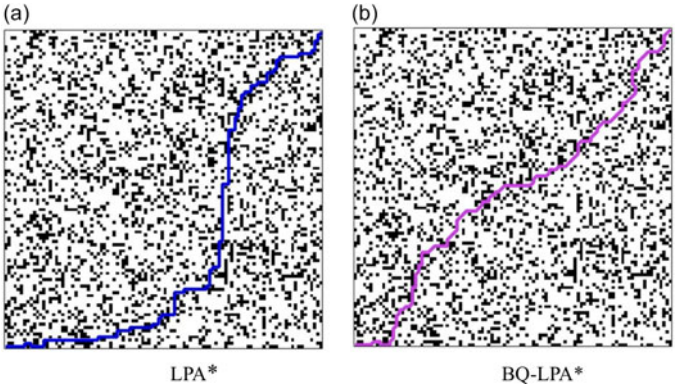


Figure 5. *Comparative results of dynamic heuristic strategy adjustments under 100 × 100.*

4.2. Comparative analysis of dynamic heuristic strategy adjustments

As described in Section 3.1, BQ-LPA* employs Q-learning to optimize the fixed heuristic function and update strategy of LPA*, enhancing estimation accuracy through a feedback mechanism. To verify its reliability, simulation comparisons are conducted between the classical LPA* algorithm and the proposed BQ-LPA* algorithm using 50×50 and 100 × 100 grid maps, with obstacle density set to 0.25. Both algorithms are configured with a starting point at (1, 1) and goals at (50, 50) and (100, 100) for the respective map sizes. The number of iterations for BQ-LPA* is 1000. The path planning results are shown in Figures 4 and 5, and the relevant data are presented in Table I.

As shown in Figures 4 and 5, classic LPA* generated paths with unnecessary detours and zigzag patterns across both grid map scales due to obstacle circumvention strategies. These redundant maneuvers inevitably increase energy consumption and travel duration during robot navigation. In contrast, BQ-LPA* produced significantly streamlined trajectories that directly target subgoals, which proves more efficient in practical scenarios by minimizing redundant path segments and reducing planning costs.

The data listed in Table I reveal that the BQ-LPA* algorithm reduces planned path lengths by 18.5% to 23.2% compared to the classic LPA* algorithm. This improvement stems from its Q-learning-integrated reward mechanism, which prioritizes straight-line paths and minimizes redundant turns. Queue operations refer to the cumulative actions of inserting, extracting, and deleting nodes in the priority queue during algorithm execution, which directly impact computational efficiency and serve as a critical performance indicator. Higher operation counts increase computational resource consumption and degrade replanning agility in dynamic environments.

https://doi.org/10.1017/S0263574725103093 Published online by Cambridge University Press

Table I. *Simulation data of dynamic heuristic strategy adjustments.*

Algorithm	Length	Time/s	Queue ops
Map (50 × 50)			
LPA*	97	1.314	3625
BQ-LPA*	76	0.01	763
Map (100 × 100)			
LPA*	189	3.505	46,832
BQ-LPA*	154	0.03	2026

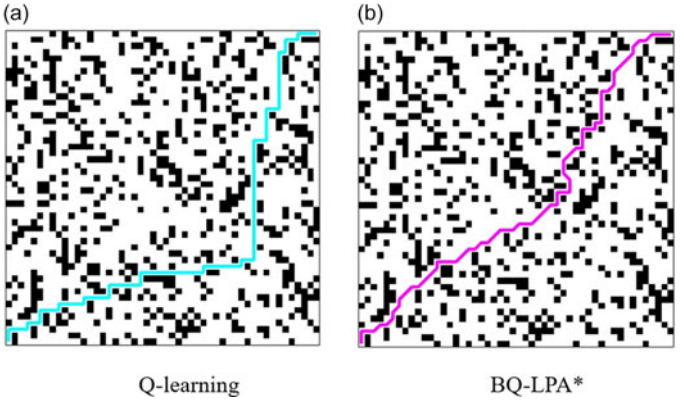


Figure 6. *Comparative results of dynamic adaptability under 50×50.*

While the classic LPA* suffers computational explosions due to frequent queue operations, BQ-LPA* leverages Q-learning's incremental learning to drastically reduce inefficient node expansions. Consequently, queue operation counts are reduced by 78.9 and 95.7% for the 50 × 50 and 100 × 100 maps. This substantial decrease confirms that the optimization of the LPA* update strategy by Q-learning is successful, avoiding update chain reactions, thereby reducing the average planning time by more than 90% compared to the classical LPA*.

4.3. Comparative analysis of dynamic adaptability

As detailed in sections 3.2 and 3.3, the BQ-LPA* algorithm leverages prior knowledge provided by LPA* to accelerate Q-learning convergence while mitigating inefficient exploration. By discretizing continuous spaces, it also resolves Q-learning's high-dimensional action selection challenges. To validate these advantages, simulations of Q-learning and BQ-LPA* are conducted on a 50 × 50 grid map. The starting point of the planning map for Q-learning and BQ-LPA* is set to (1,1), the endpoint to (50,50), with obstacle density set to 0.25.

The path planning result of Q-learning and BQ-LPA* is shown in Figure 6, which is generated within an identical grid-based simulation environment to ensure fairness. The BQ-LPA* algorithm generates notably shorter paths with fewer detours and angular deviations, attributable to its integration of LPA*-derived prior knowledge that accelerates convergence and minimizes unproductive exploration.

Additionally, to test the path planning capability of BQ-LPA* in dynamic environments, we configure three dynamic environments for simulation testing. We still implement dynamic configurations based on the 50 × 50 grid map with obstacle density set to 0.25. Regarding collision behaviors of dynamic obstacles, we establish specific protocols. A dynamic obstacle automatically reverses direction upon reaching

https://doi.org/10.1017/S0263574725103093 Published online by Cambridge University Press

Table II. *Simulation data in various dynamic environments.*

Algorithm	Length	Time/s	Collision count	Success rate	CPU usage
Low-dynamic environment					
Q-learning	110.02	2.37	0.270	84%	55.07%
LPA*	80.52	1.63	0.076	94%	38.30%
BQ-LPA*	70.08	1.21	0.052	100%	40.57%
Medium-dynamic environment					
Q-learning	110.60	2.58	0.310	82%	56.63%
LPA*	89.28	1.76	0.097	90%	39.02%
BQ-LPA*	74.38	1.25	0.059	96%	40.85%
High-dynamic environment					
Q-learning	132.88	3.01	0.450	78%	59.18%
LPA*	92.08	1.83	0.106	92%	39.67%
BQ-LPA*	78.18	1.34	0.076	98%	42.66%

its path endpoint. When encountering static obstacles during movement, it immediately reverses to simulate wall-bouncing behavior. Dynamic obstacles neither perform mutual avoidance nor collide with each other. They are permitted to pass through one another to maintain motion continuity.

Low-dynamic environment: The objective is to test the algorithm's fundamental obstacle avoidance and path fine-tuning capabilities for small quantities of predictable dynamic obstacles. Among 625 obstacle cells, we configure 10 dynamic obstacles moving at a constant low speed of 0.5 grid cells per second. This quantity sufficiently introduces dynamism without causing excessive map congestion. Each dynamic obstacle moves along a predetermined linear path and then reverses direction in a continuous loop.

Medium-dynamic environment: The objective is to test the algorithm's replanning and local obstacle avoidance capabilities in moderately congested environments with stochastic obstacle behaviors. We configure 25 dynamic obstacles whose velocities are not constant but randomly selected within the range of 1.0 to 2.0 grid cells per second. This results in localized congestion requiring more frequent replanning by the algorithm. Each dynamic obstacle still moves along a predetermined linear path and then reverses direction in a continuous loop.

High-dynamic environment: The objective is to evaluate the algorithm's real-time decision-making and high robustness in dense, complex, and unpredictable dynamic environments. We configure 50 dynamic obstacles with velocities randomly selected between 0.5 and 3.0 grid cells per second. This creates severe map congestion simulating high-traffic scenarios, requiring efficient real-time local obstacle avoidance. At simulation start, each obstacle randomly selects a traversable grid cell as its target point and moves toward it at constant speed. Upon reaching the target, it briefly pauses before selecting a new random target point and repeating the process.

For Q-learning, LPA*, and BQ-LPA*, simulation experiments consisting of 50 runs each are conducted in the aforementioned three dynamic environments. The average results of multiple metrics over 50 rounds are presented in Table II, while Figures 7 and 8, respectively, illustrate the comparative performance of the three algorithms regarding mean replanning count (MRC) and mean suboptimality percentage (MSP) across different dynamic environments.

Replanning count refers to the number of times an algorithm recalculates its path during the entire journey from start to goal due to the emergence or positional changes of dynamic obstacles. This metric reflects the algorithm's real-time performance and computational overhead. A higher replanning count indicates more frequent responses to environmental changes, but simultaneously implies greater computational burden. Suboptimality percentage measures the deviation between the path generated by the

https://doi.org/10.1017/S0263574725103093 Published online by Cambridge University Press

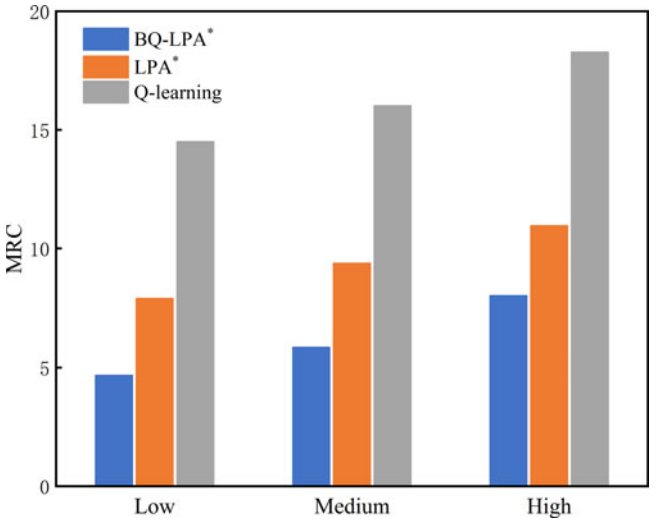


Figure 7. *Comparison of mean replanning count in different dynamic environments.*

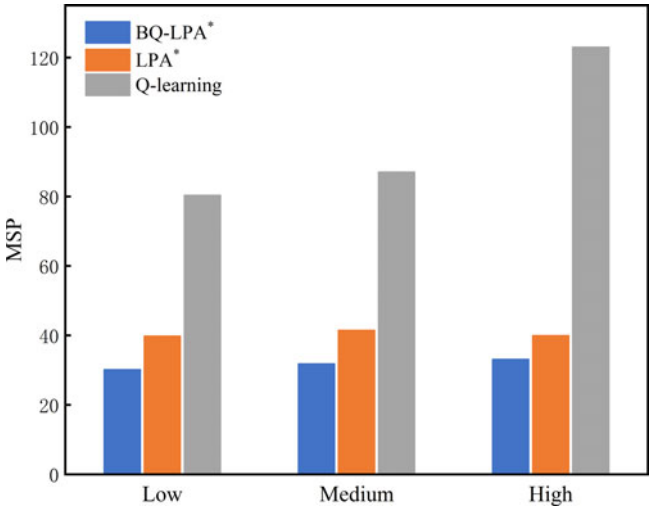


Figure 8. *Comparison of mean suboptimality percentage in different dynamic environments.*

algorithm and the theoretically optimal path (typically the shortest path in static environments). This metric primarily evaluates path efficiency in dynamic environments. A lower suboptimality percentage signifies that the algorithm effectively navigates around dynamic obstacles while maintaining minimal detours, thus preserving near-optimal path length.

BQ-LPA* achieves planning success rates exceeding 96% across diverse dynamic environments, even attaining 100% in low-dynamic environments. This underscores its exceptional mission completion reliability under varying environmental conditions. In terms of path efficiency metrics, BQ-LPA* demonstrates the shortest mean path length and lowest mean collision count, validating its superior capability in generating highly efficient collision-free trajectories. The algorithm's lower replanning count and suboptimality percentage further substantiate the effectiveness and optimality of its path planning capabilities. Notably, as environmental complexity intensifies, both planning time and CPU usage exhibit only marginal increases while consistently maintaining low levels. This confirms BQ-LPA*'s

https://doi.org/10.1017/S0263574725103093 Published online by Cambridge University Press

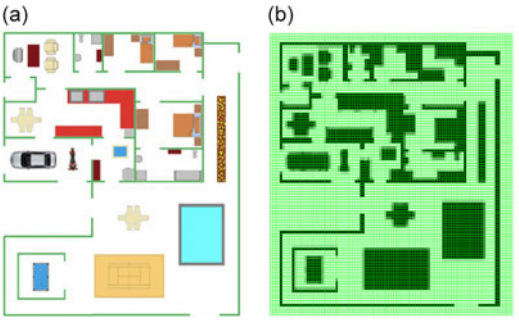


Figure 9. *An authentic environmental map.*

robust adaptability to high-frequency environmental perturbations, sustaining computationally efficient replanning with minimal resource consumption.

LPA* also maintains relatively high success rates across diverse environments. However, compared to BQ-LPA*, it exhibits higher values in metrics including mean step count, collision count, replanning count, and suboptimality percentage, along with more significant time consumption. This confirms that while LPA*'s incremental planning mechanism remains effective in dynamic environments, its path efficiency and computational efficiency are inferior to BQ-LPA*.

Q-learning struggles even in mildly dynamic environments, achieving only an 84% success rate while exhibiting high collision count, elevated replanning count, and excessive CPU usage. This indicates its failure to converge toward near-optimal solutions for dynamic path planning tasks, suggesting potential issues such as prohibitive exploration costs or slow convergence rates. As environmental complexity escalates, its learning process cannot keep pace with rapid environmental changes, demonstrating an inability to learn effective strategies. Such performance makes Q-learning unsuitable for real-time dynamic path planning applications.

The integration of prior knowledge from LPA* significantly enhances BQ-LPA*'s convergence rate and path quality. During environmental changes, LPA* dynamically updates the node set to maintain the real-time validity of the action space, while the discretized action space ensures comprehensive coverage of all state-action pairs within bounded time frames. This guarantees the algorithm's convergence criteria. By deeply integrating adaptive LPA* with Q-learning, the hybrid framework achieves an optimal balance between planning efficiency and solution quality, which provides an effective decision-making solution for autonomous systems operating in complex dynamic environments.

4.4. Comparative analysis of multiple path planning algorithms

To validate the planning capability of the BQ-LPA* algorithm in a real-world environment, we select an authentic environmental map, which is made publicly available by Hidalgo-Paniagua et al. [28]. The map is illustrated in Figure 9, with Figure 9(a) showing the original real-world scene and Figure 9(b) displaying the rasterized map after grid processing.

We removed Q-learning, which performed poorly in Section 4.3, and incorporated two mainstream path planning algorithms, RRT and ACO. In the aforementioned real-world environment, we conducted 50 simulation experiments and selected the best-performing results for each algorithm. The specific path planning outcomes are illustrated in Figure 10, with relevant data recorded in Table III.

As shown in Figure 10, the RRT algorithm connects multiple randomly sampled nodes, which results in a path with pronounced randomness that appears tortuous and non-smooth. Notably, when avoiding obstacles, it generates numerous sharp turns that substantially reduce path planning efficiency. The ACO-produced path is smoother than RRT's, but still exhibits noticeable curvature and appears less direct than LPA* or BQ-LPA*. It also detours around the central obstacle with a longer route resembling RRT's approach, yet less pronounced, while adhering more closely to obstacle contours in certain

https://doi.org/10.1017/S0263574725103093 Published online by Cambridge University Press

Table III. *Simulation data of multiple path planning algorithms in the authentic environment.*

Algorithm	Length	Time/s	Danger grids
LPA*	99.53	2.2	16
RRT	112.30	10.25	10
ACO	105.69	2.56	28
BQ-LPA*	91.27	0.22	7

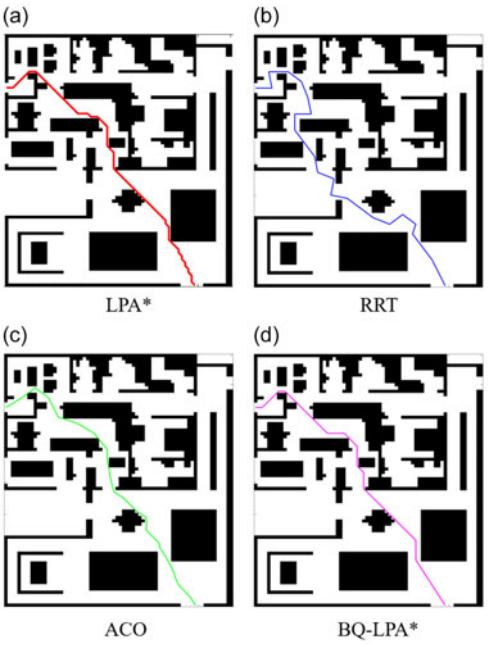


Figure 10. *Comparative results of multiple path planning algorithms.*

regions. The path generated by LPA* appears relatively direct and navigates the environment efficiently, successfully traversing the cluttered central area. This path features several abrupt, orthogonal turns, which is a characteristic feature of grid-based search algorithms moving along grid lines. BQ-LPA* produces the most direct, smooth, and visually efficient path. It traverses the central zone with minimal detours, executing gentler turns than LPA*, demonstrating exceptional effectiveness in identifying optimal collision-free trajectories. According to Table III, BQ-LPA* achieves the shortest path length and fastest planning speed among all algorithms, showcasing superior planning efficiency.

In our study, A grid is classified as dangerous if the distance between an obstacle and the planned path falls below a predefined threshold (set at 0.25). Fewer danger grids indicate fewer unsafe segments along the path, resulting in enhanced overall safety. Specifically, BQ-LPA* reduces danger grids by over 50% compared to the relatively efficient LPA*, and achieves a 75% reduction versus ACO, which exhibits moderate obstacle avoidance. Although RRT's danger grid count approaches BQ-LPA*'s, this safety improvement entails disproportionately longer paths, representing an impractical trade-off. These results demonstrate that through obstacle weighting and collision penalties, BQ-LPA* not only effectively mitigates collision risks but also outperforms mainstream algorithms in both planning efficiency and computational complexity.

https://doi.org/10.1017/S0263574725103093 Published online by Cambridge University Press

5. Conclusion

This paper proposes a novel mobile robot path planning algorithm named BQ-LPA*, which shows significant advantages over existing mainstream algorithms in three key innovations: 1) We replace the fixed heuristic function of the LPA* algorithm with Q-learning, which enhances the algorithm's global search capability and path quality by dynamically adjusting heuristic weights and update strategies. 2) The proposed method, which leverages the prior knowledge that the LPA* algorithm provides, reduces inefficient exploration so that it significantly improves convergence speed and sample efficiency in complex environments. 3) The continuous space is discretized through the LPA* algorithm, which reduces the dimensionality of the action space. This process resolves issues of action space congestion, storage limitations, and computational inefficiencies, while improving path feasibility.

Experimental results show that, compared to other mainstream algorithms, BQ-LPA* achieves an 18.51% to 36.36% reduction in path length, reduces dangerous turns by up to 75%, and shortens planning time by over 90%. Across diverse dynamic environments, BQ-LPA* exhibits enhanced dynamic adaptability, safety, and planning efficiency with minimal computational overhead. By combining the incremental characteristics of the LPA* algorithm with the dynamic strategy of Q-learning, the algorithm solves the path update problem in dynamic environments. Its dynamic heuristic adjustment capability optimizes global search efficiency, while incremental experience reuse and adaptive conflict resolution mechanisms significantly reduce computational resource consumption. Overall, BQ-LPA* surpasses traditional algorithms in dynamic adaptability, complex scenario compatibility, and computational efficiency. These improvements substantially enhance both planning efficiency and safety for mobile robots during path planning, which offers significant value for optimizing robotic inspection tasks.

Author contributions. Shixuan Qi conceived and designed the study. Haofei Lu and Ziyi Wang conducted data gathering. Xin Gao and Siyao Chen performed statistical analyses. Shixuan Qi, Weigang Wang, and Xin Gao wrote the article.

Financial support. National Natural Science Foundation of China under Grant (No. 62371245, No. 62371248), National Key R&D Program of China (2023YFB2904000), Jiangsu Key Development Planning Project (BE2023004-2), Natural Science Foundation of Jiangsu Province (Higher Education Institutions) (20KJA520001), The 14th Five Year Plan project of Equipment Development Department (315107402), China State Railway Scientific Research and Development Program (L2024G006), Jiangsu Hongxin Information Technology Co., Ltd Project.

Competing interests. The authors declare no conflicts of interest exist.

Ethical standards. Not applicable.

References

[1] J. Zheng, T. Chen, J. He, Z. Wang and B. Gao, "Review on security range perception methods and path-planning techniques for substation mobile robots," *Energies* **17**(16), 4106 (2024).
[2] X. Wang, P. Duan, L. Meng and K. Yang, "An improved iterated greedy algorithm for solving rescue robot path planning problem with limited survival time," *Comput. Mater. Contin.* **80**(1), (2024).
[3] Y. Dai, S. Li, X. Chen, X. Nie, X. Rui and Q. Zhang, "Three-dimensional truss path planning of cellular robots based on improved sparrow algorithm," *Robotica* **42**(2), 347–366 (2024).
[4] B. B. K. Ayawli, J. K. Dawson, E. Badu, I. E. B. Ayawli and D. Lamusah, "Comparative analysis of popular mobile robot roadmap path-planning methods," *Robotica*, 1–24 (2025).
[5] L. Zhu, G. Zhou, Y. Zhou, Q. Luo, H. Huang and X. Wei, "Self-adaptive differential evolution-based coati optimization algorithm for multi-robot path planning," *Robotica* **43**(3), 959–996 (2025).
[6] S. Liu, S. Liu and H. Xiao, "Improved gray wolf optimization algorithm integrating a* algorithm for path planning of mobile charging robots," *Robotica* **42**(2), 536–559 (2024).
[7] Q. Zhang, X. Ning, Y. Li, L. Pan, R. Gao and L. Zhang, "Path planning of patrol robot based on modified grey wolf optimizer," *Robotica* **41**(7), 1947–1975 (2023).
[8] Y. Luo, S. Lin, Y. Wang and K. LiangPath planning for mobile robots in complex environments based on enhanced sparrow search algorithm and dynamic window approach," *Robotica*, 1–24 (2025).
[9] Z. Qadir, F. Ullah, H. S. Munawar and F. Al-Turjman, "Addressing disasters in smart cities through uavs path planning and 5g communications: A systematic review," *Comput. Commun.* **168**, 114–135 (2021).

https://doi.org/10.1017/S0263574725103093 Published online by Cambridge University Press

[10] S. Aradi, "Survey of deep reinforcement learning for motion planning of autonomous vehicles," *IEEE Trans. Intell. Transp.* **23**(2), 740–759 (2020).
[11] F. Wu, W. Tang, Y. Zhou, S.-W. Lin, Z. Ding and Y. Liu, "Robust motion planning for mobile robots under attacks against obstacle localization," *Robotica* **42**(8), 2781–2800 (2024).
[12] H.-y. Zhang, W.-m. Lin and A.-x. Chen, "Path planning for the mobile robot: A review," *Symmetry* **10**(10), 450 (2018).
[13] P. Raja and S. Pugazhenthi, "Optimal path planning of mobile robots: A review," *Int. J. Phys. Sci.* **7**(9), 1314–1320 (2012).
[14] J. R. Sánchez-Ibáñez, C. J. P.-D. Pulgar and A. García-Cerezo, "Path planning for autonomous mobile robots: A review," *Sensors* **21**(23), 7898 (2021).
[15] T. Liao, F. Chen, Y. Wu, H. Zeng, S. Ouyang and J. Guan, "Research on path planning with the integration of adaptive a-star algorithm and improved dynamic window approach," *Electronics* **13**(2), 455 (2024).
[16] J. Yu, C. Chen, A. Arab, J. Yi, X. Pei and X. Guo, "Rdt-rrt: Real-time double-tree rapidly-exploring random tree path planning for autonomous vehicles," *Expert Syst. Appl.* **240**, 122510 (2024).
[17] M. S. Adam, R. Nordin, N. F. Abdullah, A. Abu-Samah, O. A. Amodu and M. H. Alsharif, "Optimizing disaster response through efficient path planning of mobile aerial base station with genetic algorithm," *Drones* **8**(6), 272 (2024).
[18] L. Hu and C. Cai, "Arime-dwa: Robot path planning utilizing advanced rime optimization and dynamic windows approach," *IEEE Access* (2024).
[19] Y. Wang, C. Lu, P. Wu and X. Zhang, "Path planning for unmanned surface vehicle based on improved q-learning algorithm," *Ocean Eng.* **292**, 116510 (2024).
[20] L. Bo, T. Zhang, H. Zhang, J. Yang, Z. Zhang, C. Zhang and M. Liu, "Improved q-learning algorithm based on flower pollination algorithm and tabulation method for unmanned aerial vehicle path planning," *IEEE Access* (2024).
[21] M. R. Jones, S. Djahel and K. Welsh, "An efficient and rapidly adaptable lightweight multi-destination urban path planning approach for uavs using q-learning," *IEEE Trans. Intell. Veh.* (2024).
[22] C. Wang, X. Yang and H. Li, "Improved q-learning applied to dynamic obstacle avoidance and path planning," *IEEE Access* **10**, 92879–92888 (2022).
[23] Z. Xiang, H. Sun and J. Zhang, "Application of improved q-learning algorithm in dynamic path planning for aircraft at airports," *IEEE Access* **11**, 107892–107905 (2023).
[24] C. J. Watkins and P. Dayan, "Q-learning," *Mach. Learn.* **8**, 279–292 (1992).
[25] Y. Li, H. Wang, J. Fan and Y. Geng, "A novel q-learning algorithm based on improved whale optimization algorithm for path planning," *PLoS One* **17**(12), e0279438 (2022).
[26] S. Koenig, M. Likhachev and D. Furcy, "Lifelong planning a*," *Artif. Intell.* **155**(1-2), 93–146 (2004).
[27] L. Khriji, F. Touati, K. Benhmed and A. Al-Yahmedi, "Mobile robot navigation based on q-learning technique," *Int. J. Adv. Robot. Syst.* **8**(1), 4 (2011).
[28] A. Hidalgo-Paniagua, M. A. Vega-Rodríguez, J. Ferruz and N. Pavón, "Solving the multi-objective path planning problem in mobile robotics with a firefly-based approach," *Soft Comput.* **21**(4), 949–964 (2017).

Cite this article: S. Qi, H. Lu, Z. Wang, X. Gao, S. Chen, Y. Liu and W. Wang (2026). "A novel path planning algorithm based on synergistic bidirectional optimization", Robotica **44**, 52–70. https://doi.org/10.1017/S0263574725103093

https://doi.org/10.1017/S0263574725103093 Published online by Cambridge University Press