A Hybrid Path Planning Framework for Mobile Robot with Efficient Search and Shorter, Continuously Smooth Path

Ying Zhang, Xiaohui An

Abstract— **Path planning is a critical problem in robotics, aimed at generating efficient and smooth trajectories for robots. However, traditional algorithms like A* often suffer from low search efficiency, redundant nodes, and discontinuities in velocity and acceleration at path corners. To address these issues, this paper presents a path planning framework that combines the Optimized Bidirectional A* algorithm, the Critical Node Retention algorithm, and the Minimum Snap algorithm. The framework enhances the path planning process in three stages: First, the Bidirectional A* search is optimized by utilizing a weighted heuristic function to improve search efficiency and reduce unnecessary node expansions. Second, the Critical Node Retention algorithm refines the path by retaining only critical nodes, reducing redundancy and shortening the path length. Finally, the Minimum Snap algorithm smooths the path, ensuring continuous velocity and acceleration, thereby generating safer and smoother trajectories. Simulation experiments on grid maps of various sizes demonstrate that the proposed framework effectively reduces path length, improves search efficiency, and generates smooth, safe trajectories, providing an efficient solution to path planning.**

Index Terms—**Mobile Robot, Path Planning, A* Algorithm, Minimum Snap Algorithm**

I. INTRODUCTION

In recent years, due to technological innovation and development, mobile robots have been widely used in agriculture, healthcare, the military, etc.[1]. Path planning, crucial for mobile robots' autonomous navigation, is a key technology[2], aiming to find a lowest-cost and collision-free path between start and target nodes in a specific environment [3], where the lowest cost means the shortest path length, fastest planning time, least energy consumption, etc.[4]. Finding the lowest-cost safe path in a specific environment has become a hot research topic in mobile robot path planning.

Currently, numerous researchers have extensively studied the path planning problem for mobile robots and proposed various algorithms. Classical algorithms include the Genetic Algorithm (GA) [5], Ant Colony Optimization (ACO) [6], Particle Swarm Optimization (PSO) [7], Rapidly-exploring Random Tree (RRT) [8], and A* algorithm [9], among others. The GA is widely used in path planning due to its parallelism
and strong convergence. However, it suffers from low search accuracy and poor path convergence quality [10]. The ACO algorithm, known for its distributed computing capabilities and robustness, is commonly applied to mobile robot path planning. Despite this, it faces challenges regarding planning efficiency and slow convergence [11]. The PSO algorithm offers advantages such as fast search speed, simple structure, and ease of implementation [12]. However, its reliance on a single guidance mechanism and the design of acceleration and inertia weights leads to low efficiency and inability to guarantee convergence to the global optimum [13]. The RRT algorithm can quickly find feasible paths in complex environments [14], but its limited search range reduces its ability to navigate narrow passages, hindering the discovery of optimal paths [15]. In contrast, the A* algorithm, a classic heuristic search method, is favored in path planning due to its simplicity and efficient search strategy [16]. Nevertheless, it faces several challenges in complex environments, such as long search times, redundant nodes, and discontinuities in velocity and acceleration at path corners.

To address the issues in path search and improve both path quality and search efficiency of the A* algorithm, researchers have proposed a series of optimization methods. Ref.[17] improved the A* algorithm by combining Euclidean distance with point-to-line distance in a hybrid heuristic function, which reduced the number of search nodes and enhanced search efficiency. Ref.[18] incorporated jump point search with the traditional A* algorithm, further increasing search speed. Ref.[19] included angle as a cost function in the traditional A* algorithm to find paths with the least number of turns, thereby reducing path length and improving planning efficiency, particularly for large cargo land transportation. Ref.[20] considered the distance to obstacles in the heuristic function to improve the path's safety and somewhat optimize the overall path length. Ref.[21] proposed an ACO algorithm based on bidirectional A* to optimize ACO's pheromone updates, accelerating convergence and recommending the fastest routes for taxis in urban road networks. Ref.[22] added collision cost and vehicle heading angle cost to the A* heuristic function, which improved path smoothness while ensuring the vehicle's kinematic constraints, effectively reducing path length and improving driving efficiency. Ref.[23] employed bidirectional sector expansion and variable step-size search strategies to expand the search space and accelerate the search process, further improving the efficiency of path planning. Ref.[24] combined the A* algorithm with ACO and PSO to design a two-layer, multi-objective path planning model and algorithm, which generates shorter, collision-free paths while enhancing path safety. Ref.[25] introduced an obstacle ratio-optimized heuristic function and combined it with a dynamic window method for real-time obstacle avoidance, ensuring that the planned path is not only globally optimal but also capable of real-time obstacle avoidance, thus

Manuscript received April 09, 2025
Ying Zhang, Department of Software, Tiangong University, Tianjin, China
Xiaohui An, Department of Software, Tiangong University, Tianjin, China

guaranteeing path smoothness.

Although existing methods have achieved breakthroughs in certain aspects, such as search efficiency, path length, and path smoothness, limitations remain when considering multiple performance metrics comprehensively. Therefore, designing a comprehensive path planning optimization framework that can both improve search efficiency and generate smooth, safe paths remains a critical challenge. To address this, this paper proposes a novel path planning framework, BiA*-MS, which combines the Optimized Bidirectional A* Algorithm, Critical Node Retention Algorithm, and Minimum Snap Algorithm to overcome the issues of low efficiency, redundant path nodes, and discontinuities at path corners in traditional path planning methods.

In this framework, we conducted extensive simulation experiments and tested it on grid maps of varying sizes. The experimental results show that the proposed framework not only significantly improves the search efficiency of path planning and reduces path length, but also demonstrates better performance in terms of path smoothness and safety compared to traditional methods. These advantages make the framework highly promising for path planning in complex environments, particularly for autonomous navigation and robotic tasks that require efficient search and smooth paths.

The main contributions of this paper are as follows:

1. A multi-stage path planning framework is proposed, which optimizes multiple performance metrics in the path planning process by combining the Optimized Bidirectional A* algorithm, Critical Node Retention algorithm, and Minimum Snap algorithm.

2. The effectiveness of the proposed framework is validated through simulation experiments on grid maps of varying sizes. The results demonstrate that the framework excels in improving search efficiency, reducing path length, and ensuring path smoothness.

3. A new approach to path planning is provided, offering valuable insights for future research in the field of robot path planning.

The structure of this paper is organized as follows: Section II defines the problem formulation; Section III presents a detailed description of the proposed path planning framework and its key technologies; Section IV showcases the experimental design and analysis of the results; Section V summarizes the research findings and outlines potential directions for future work.

II. Problem Formulation

This section presents the problem formulation for path planning on a 2D grid map. The formulation includes the map modeling, numerical definitions, and optimization objectives.

A. Map Modeling

The path planning environment is modeled as a 2D grid map, where the map is divided into discrete cells, each representing a unit square. The robot is considered as a particle of the same size as the unit square. However, to account for the robot's physical dimensions and mass in real-world applications, a protection radius is added around obstacles to ensure safe movement and prevent collisions. A 20×20 grid map as shown in Fig.1, the green node represents the starting node of the robot, while the red node represents the target node that the robot needs to reach. The black areas indicate the obstacle regions, and the gray areas represent the expansion zones of the obstacles. The blue nodes denote the nodes that have been expanded by the A* algorithm, and the pink nodes represent the final path nodes. Finally, the blue polyline represents the trajectory passing through the path nodes.

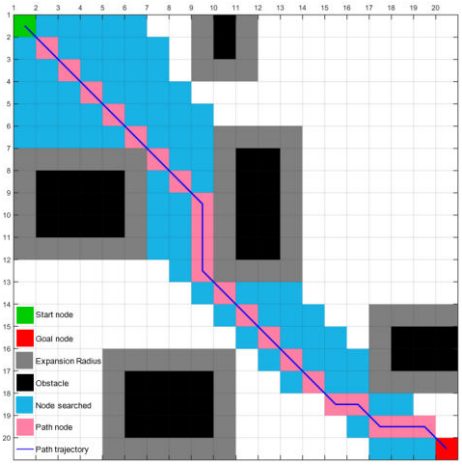


Fig.1 20×20 grid map

B. Mathematical Formulation of Path Planning

The path planning task is to find an optimal path from the starting node to the target node while avoiding obstacles and ensuring that the robot can move smoothly and safely. To formalize this path planning problem, it can be defined using the following mathematical elements:

Start Node: The robot's initial position, denoted by $S=(x_0,y_0)$, where (x_0,y_0) is the center coordinate of the grid cell where the start node is located.

Target Node: The robot's goal position, denoted by $T=(x_t,y_t)$, where (x_t,y_t) is the center coordinate of the grid cell where the target node is located.

Current Node: The robot's position during its movement process, denoted by $N=(x_n,y_n)$, where (x_n,y_n) is the center coordinate of the grid cell where the current node is located.

Movement Directions: The robot moves in an 8-neighborhood manner within the grid map, where it can move to adjacent cells either horizontally, vertically, or diagonally.

In A*-based path planning, the cost function is a critical element for evaluating the cost of a path and is essential for the search for the optimal path. The cost function is defined as:

$$f(N)=g(N)+h(N) \tag{1}$$

where $f(N)$ represents the estimated total cost of traveling from the start node S through the node N to the target node T, and it is a key metric used in path planning algorithms to evaluate node priority. The algorithm will prioritize the expansion of the node with the smallest $f(N)$ value in order to find the optimal path from the start to the goal node. Here, $g(N)$ represents the actual cost, and $h(N)$ represents the heuristic function.

Actual Cost $g(N)$: Represents the actual path cost from

the start node S to the current node N. Typically, the cost for moving from one node to an adjacent node in the horizontal or vertical direction is fixed and can be set to 1. For diagonal movements, the cost is usually set to $\sqrt{2}$.

Heuristic Function $h(N)$: Represents the estimated cost from the current node N to the target node T, aiming to predict the remaining cost to reach the goal. Common heuristic functions include:

Manhattan Distance:

$$h(N) = |x_t - x_n| + |y_t - y_n| \tag{2}$$

Euclidean Distance:

$$h(N) = \sqrt{(x_t - x_n)^2 + (y_t - y_n)^2} \tag{3}$$

Chebyshev Distance:

$$h(N) = \max(|x_t - x_n|, |y_t - y_n|) \tag{4}$$

Considering that the Euclidean distance closely approximates the true straight-line distance between two points, the Euclidean distance is chosen as the heuristic function.

C. Optimization Objectives

The goal of path planning is not only to find a feasible path but also to achieve the following optimization objectives:

Search Efficiency: Optimize the search strategy to reduce unnecessary node expansions during the path search, thereby improving the overall search efficiency.

Minimization of Path Total Cost and Node Count: Minimize the total cost $f(N)$ and the number of path nodes from the start node to the target node, thereby reducing path length, avoiding redundant paths, and ensuring that the robot reaches the target in the shortest time.

Path Smoothness: Ensure that the path is sufficiently smooth to reduce sharp turns along the path, thereby ensuring continuity in the robot's speed and acceleration at corners.

To achieve the aforementioned optimization objectives, this study proposes the BiA*-MS path planning framework. This framework aims to enhance search efficiency, minimize redundant nodes along the path, and ensure that the robot follows a smooth and continuous trajectory, thereby improving the stability of its movement.

III. BiA*-MS Path Planning Framework and Core Techniques

This section introduces the BiA*-MS path planning framework and its core techniques, which combine the Optimized Bidirectional A* Algorithm, the Critical Node Retention Algorithm, and the Minimum Snap Algorithm to address issues inherent in traditional path planning methods, such as excessive search time, redundant nodes, and discontinuities in velocity and acceleration at path corners. Specifically, the BiA*-MS framework is optimized through the following three components, as illustrated in Fig.2.

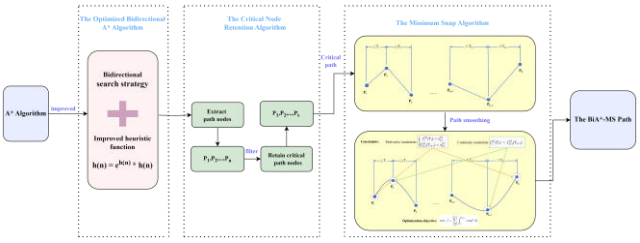


Fig.2 BiA*-MS Overall Framework

1. Optimized Bidirectional A* Algorithm: This algorithm employs a weighted heuristic function and performs bidirectional search from both the start and goal nodes, thereby reducing unnecessary node expansions and improving search efficiency.

2. Critical Node Retention Algorithm: This algorithm optimizes the generated path by extracting critical nodes, thereby reducing path redundancy and overall path length.

3. Minimum Snap Algorithm: During the path planning process, the Minimum Snap algorithm is applied to smooth the path, ensuring continuous velocity and acceleration profiles, which guarantees the robot's motion stability and safety.

A. Optimized Bidirectional A* Algorithm

The A* algorithm is widely used in path planning but faces challenges such as slow search speeds and inaccurate heuristics. To improve its efficiency, a bidirectional search strategy is introduced. The bidirectional A* algorithm expands from both the start and target nodes simultaneously, significantly reducing the search space and enhancing efficiency. The heuristic function for the forward search, based on Euclidean distance, is as in:

$$h_F(N_F) = \sqrt{(x_{N_B} - x_{N_F})^2 + (y_{N_B} - y_{N_F})^2} \tag{5}$$

and its cost function is provided in:

$$f_F(N_F) = g_F(N_F) + h_F(N_F) \tag{6}$$

Similarly, the heuristic function for the backward search is as in:

$$h_B(N_B) = \sqrt{(x_{N_F} - x_{N_B})^2 + (y_{N_F} - y_{N_B})^2} \tag{7}$$

and its cost function is presented in:

$$f_B(N_B) = g_B(N_B) + h_B(N_B) \tag{8}$$

where N_F and N_B represent the current nodes in the forward and backward searches, respectively, with x_{N_F} and y_{N_F} denoting the horizontal and vertical coordinates of N_F, and x_{N_B} and y_{N_B} representing the coordinates of N_B. $g_F(N_F)$ denotes the actual cost from the start node S to N_F, and $g_B(N_B)$ represents the cost from the target node T to N_B.

Moreover, in the traditional A* algorithm, the heuristic function $h(N)$ estimates the cost from the current node to the target, guiding the search by selecting the next node to expand. However, when the heuristic is inaccurate or the search space is large, the A* algorithm may expand too many nodes, reducing efficiency.

To address this issue, this paper adopts the optimization method proposed by ref.[26], which weights the heuristic functions of forward search and backward search through exponential functions $e^{h_F(N_F)}$ and $e^{h_B(N_B)}$, respectively. This modification enhances the nonlinearity and influence of the heuristic function. As a result, larger estimates are obtained

for nodes farther from the target, prompting the algorithm to discard these nodes more quickly and focus on those closer to the target. This adjustment accelerates the search process, particularly in complex problems with large search spaces.

The optimized cost functions for the forward and backward searches are shown in:

$$f_F(N_F) = g_F(N_F) + e^{h_F(N_F)} \cdot h_F(N_F) \quad (9)$$

$$f_B(N_B) = g_B(N_B) + e^{h_B(N_B)} \cdot h_B(N_B) \quad (10)$$

The Optimized Bidirectional A* Algorithm are provided in Algorithm 1.

B. *Critical Node Retention Algorithm*

The path planned by the Optimized Bidirectional A* Algorithm is a polyline path composed of many nodes, where not all nodes are crucial. The nodes that control the path direction are called critical nodes, which are usually turning points or decision points on the path and directly affect the effectiveness and safety of the path. The other nodes are called redundant nodes, which only serve a connecting role without significantly influencing the path's direction.

In the path planning process, some redundant nodes are inevitable, however, optimization strategies can be employed to minimize their number, resulting in a more concise and efficient path. In this paper, the Critical Node Retention Algorithm is used to filter out redundant nodes from the path. The algorithm is described as follows:

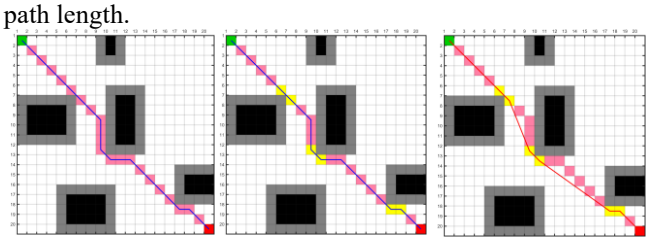
Step1: Connect the start node to the next path node sequentially, treating each connected node as the current node. Check for obstacles between the start and current nodes; if obstacles are detected, proceed to Step 2; otherwise, continue to Step 1.

Step2: If obstacles are detected between two nodes, mark the previous node as a critical node. Then, check if the current node is the target. If it is, proceed to Step 4; otherwise, proceed to Step 3.

Step3: Mark the current node as a critical node, treat it as the new start node, and repeat Step 1 with this new start node.

Step4: End the process and output the critical path nodes.

The intuitive process of the Critical Node Retention Algorithm is illustrated in Fig.3. In Fig.3(a), the path obtained using the Optimized Bidirectional A* Algorithm is shown, but the path contains numerous redundant nodes. To simplify the path and make it clearer, the Critical Node Retention Algorithm can be applied to filter out redundant nodes. After filtering, the yellow nodes in Fig.3(b) are identified as critical nodes. By connecting these critical nodes, a critical path is formed, as shown by the red path in Fig.3(c). Compared to Fig.3(a), the critical path obtained using the Critical Node Retention Algorithm not only more clearly represents the critical nodes of the path but also has a shorter path length.



(a) (b) (c)

Fig.3 The Process of the Critical Node Retention Algorithm

Algorithm 1: The Optimized Bidirectional A* Algorithm

Initialize the forward open list O_f with start node S
Initialize the backward open list O_b with target node T
Initialize forward closed list C_f and backward closed list C_b as empty
Initialize forward parent list P_f and backward parent list P_b as empty
while O_f *is not empty and* O_b *is not empty* **do**
  Extract node N_F with minimum f_F from O_f and add it to C_f
  if $N_F \in C_b$ **then**
    return the Path by combining P_f and the reverse of P_b
  end
  for *each adjacent node* N'_F *of* N_F **do**
    if $N'_F \notin C_f$ *or* $g_F(N_F) + cost(N_F, N'_F) < g_F(N'_F)$ **then**
      $g_F(N'_F) \leftarrow g_F(N_F) + \text{cost}(N_F, N'_F)$
      $f_F(N'_F) \leftarrow g_F(N'_F) + h_F(N'_F)$
      if $N'_F \notin O_f$ **then**
        Add N'_F to O_f with the value of $f_F(N'_F)$;
      end
      else
        Update the f_F value of N'_F in O_f to $f_F(N'_F)$;
      end
      $P_f[N'_F] \leftarrow N_F$;
    end
  end
  Extract node N_B with minimum f_B from O_b and add it to C_b
  if $N_B \in C_f$ **then**
    return the Path by combining P_f and the reverse of P_b;
  end
  for *each adjacent node* N'_B *of* N_B **do**
    if $N'_B \notin C_b$ *or* $g_B(N_B) + cost(N_B, N'_B) < g_B(N'_B)$ **then**
      $g_B(N'_B) \leftarrow g_B(N_B) + \text{cost}(N_B, N'_B)$
      $f_B(N'_B) \leftarrow g_B(N'_B) + h_B(N'_B)$
      if $N'_B \notin O_b$ **then**
        Add N'_B to O_b with the value of $f_B(N'_b)$;
      end
      else
        Update the f_B value of N'_B in O_b to $f_B(N'_B)$;
      end
      $P_b[N'_B] \leftarrow N_B$;
    end
  end
end
return "No path found";

C. *Minimum Snap Algorithm*

In robot path planning, the paths generated by the Optimized Bidirectional A* algorithm and the Critical Node Retention algorithm are typically piecewise linear, as shown in Fig.2. These paths often exhibit discontinuities in velocity, acceleration, and higher-order derivatives at the corners, which violate the robot's kinematic constraints, potentially leading to instability in robot motion and excessive energy consumption. To address this issue, the BiA*-MS path planning framework incorporates the Minimum Snap algorithm, which further refines the key path into a smooth, higher-order continuous trajectory. This optimization eliminates discontinuities in higher-order derivatives, such as velocity and acceleration, at the turns, ensuring better alignment between the path and the robot's actual motion, thereby achieving more efficient and higher-quality path planning.

The discrete critical path nodes obtained through the Critical Node Retention algorithm (including the start node S and the goal node T) are defined as control points $P = \{P_1, P_2, \ldots, P_k\}$. The core idea of the Minimum Snap algorithm is to construct a piecewise polynomial trajectory $Q(t)$ using these control points, which is composed of $k-1$ segments. Each sub-trajectory $Q_i(t)$ represents the path between two adjacent control points P_i and P_{i+1} , and is described by a polynomial of degree n :

$$Q_i(t) = a_{i0} + a_{i1}t + a_{i2}t^2 + \ldots + a_{in}t^n = \sum_{j=0}^{n} a_{ij}t^j, \quad i \in [1, k-1], j \in [0, n], t \in [t_i, t_{i+1}] \tag{11}$$

In this expression, a_{ij} represents the polynomial coefficients to be optimized, and t_i and t_{i+1} are the start and end times of the sub-trajectory $Q_i(t)$, respectively. The jerk and snap refer to the third and fourth derivatives of the trajectory, respectively, and are mathematically expressed as:

$$jerk = \frac{d^3 Q_i(t)}{dt^3} \tag{12}$$

$$snap = \frac{d^4 Q_i(t)}{dt^4} \tag{13}$$

The objective function of the Minimum Snap algorithm can be defined as:

$$min\ J = \sum_{i=1}^{k-1} \int_{t_i}^{t_{i+1}} snap^2 dt = \sum_{i=1}^{k-1} \int_{t_i}^{t_{i+1}} \left(\frac{d^4 Q_i(t)}{dt^4}\right)^2 dt = a^T H a \tag{14}$$

where the vector a contains the coefficients of all the piecewise polynomial segments, and H is the positive definite quadratic objective function matrix. By minimizing the squared integral of the $snap$, the abrupt changes in the higher-order derivatives of the trajectory can be reduced, thereby improving its smoothness.

In addition, to ensure continuity of the trajectory in terms of position, velocity, acceleration, and $jerk$, the following constraints must be satisfied while minimizing the objective function: First, at the time instances t_1 and t_k, the trajectory must pass through the control points P_1 (the start node) and P_k (the goal node), respectively, with zero velocity, acceleration, and jerk. This can be expressed as:

$$\begin{aligned} &Q_1(t_1) = P_1, \frac{dQ_1(t_1)}{dt} = 0, \frac{d^2 Q_1(t_1)}{dt^2} = 0, \frac{d^3 Q_1(t_1)}{dt^3} = 0, \\ &Q_{k-1}(t_k) = P_k, \frac{dQ_{k-1}(t_k)}{dt} = 0, \frac{d^2 Q_{k-1}(t_k)}{dt^2} = 0, \frac{d^3 Q_{k-1}(t_k)}{dt^3} = 0. \end{aligned} \tag{15}$$

Secondly, the trajectory $Q(t)$ must pass through all the control points $P = \{P_1, P_2, \ldots, P_k\}$ and maintain positional continuity at the connection points t_{i+1} between any adjacent sub-trajectories. This can be expressed as:

$$Q_i(t_i) = P_i, \quad Q_i(t_{i+1}) = P_{i+1}, \quad \forall i = 1, 2, \ldots, k-1 \tag{16}$$

Additionally, at the connection points between any adjacent sub-trajectories, the trajectory must maintain continuity in velocity, acceleration, and jerk. This can be expressed as:

$$\frac{d^j Q_i(t_{i+1})}{dt^j} = \frac{d^j Q_{i+1}(t_{i+1})}{dt^j}, \quad j = 1, 2, 3 \quad \forall i = 1, 2, \ldots, k-2 \tag{17}$$

By combining the objective function with the constraints, the Minimum Snap algorithm can be formulated as a quadratic programming problem:

$$\begin{cases} min\ a^T H a \\ s.t.\ \ Aa = b \end{cases} \tag{18}$$

The matrix A contains the linear constraint conditions for position, velocity, acceleration, and jerk, while the vector b includes the corresponding constraint values. By solving the quadratic programming problem, the optimized polynomial coefficients can be obtained, thereby constructing a piecewise polynomial trajectory $Q(t)$ that satisfies smoothness and higher-order continuity.

IV. Simulation and Analysis of the BiA*-MS Path Planning Framework

To validate the effectiveness of the BiA*-MS path planning framework, simulation experiments were conducted using MATLAB 2021 on a Windows 10 operating system with an i5-9300 processor. The experiments were designed to compare and analyze three modules: the Optimized Bidirectional A* algorithm, the Critical Node Retention algorithm, and the Minimum Snap algorithm. The evaluation was performed in four grid maps of different scales (40×40, 60×60, 80×80, and 100×100) to assess the performance of each module in terms of path planning efficiency, path length, and smoothness.

A. Simulation and Analysis of the Optimized Bidirectional A Algorithm*

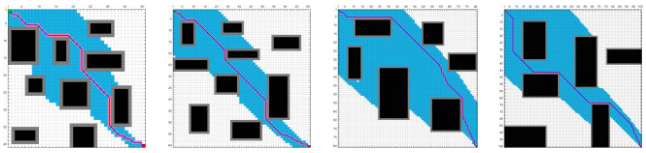
In the BiA*-MS path planning framework, the Optimized Bidirectional A* algorithm serves as the initial module of the entire framework, playing a key role in improving search efficiency and optimizing the path, as it is responsible for generating the initial path. To evaluate the performance of the Optimized Bidirectional A* algorithm, experiments were conducted comparing it with the traditional A* and bidirectional A* algorithms, as shown in Fig.4.

By comparing Fig.4(a) - (l), it is evident that the Optimized Bidirectional A* algorithm significantly reduces the search area and minimizes the number of path turns, resulting in a more concise path compared to both the traditional A* algorithm and the bidirectional A* algorithm. This highlights its superiority in both search efficiency and path optimization.

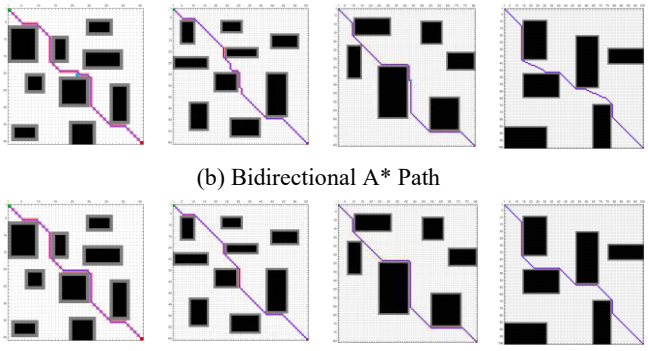
B. Simulation and Analysis of the Critical Node Retention Algorithm

The Critical Node Retention Algorithm is the second module of the BiA*-MS path planning framework. Its primary function is to filter out redundant nodes in the path, retaining only the critical path nodes. This reduces the total number of nodes, resulting in a more concise and efficient critical path, thereby addressing the issue of path length in the Optimized Bidirectional A* Algorithm.

As shown in Fig.5, compared to the paths generated by the traditional A*, Bidirectional A*, and Optimized Bidirectional A* algorithms, the critical paths filtered by the Critical Node Retention Algorithm demonstrate superior performance in terms of both total node count and path length. The algorithm significantly reduces the number of nodes and shortens the



(a) Traditional A* Path



(b) Bidirectional A* Path

(c) Optimized Bidirectional A* Path

Fig.4 Three Algorithms on Four Grid Maps of Varying Scales

path length, resulting in a more concise and efficient path. To visually highlight the advantages of the Critical Node Retention Algorithm, a bar chart is used to compare the performance of each algorithm in terms of path node count and path length, with the results shown in Fig.6.

Fig.6 illustrates that the critical paths filtered by the Critical Node Retention Algorithm have fewer nodes and shorter path lengths compared to those generated by the A*, Bidirectional A*, and Optimized Bidirectional A* algorithms. Therefore, introducing the Critical Node Retention Algorithm into the BiA*-MS path planning framework can further optimize path length, addressing the limitations of the Optimized Bidirectional A* in this regard, and thereby achieving better path planning results.

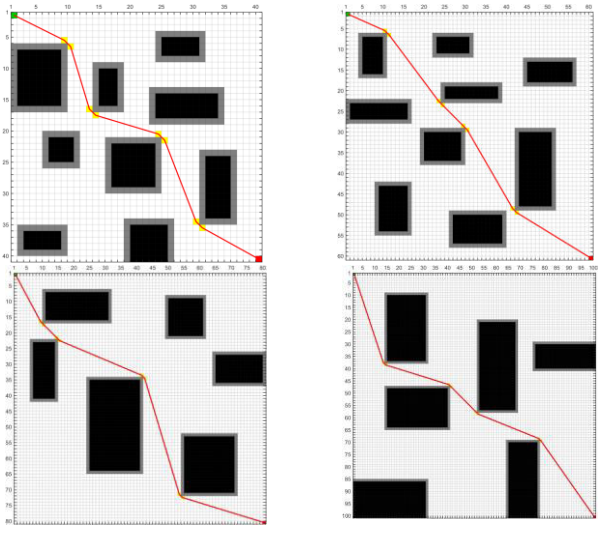
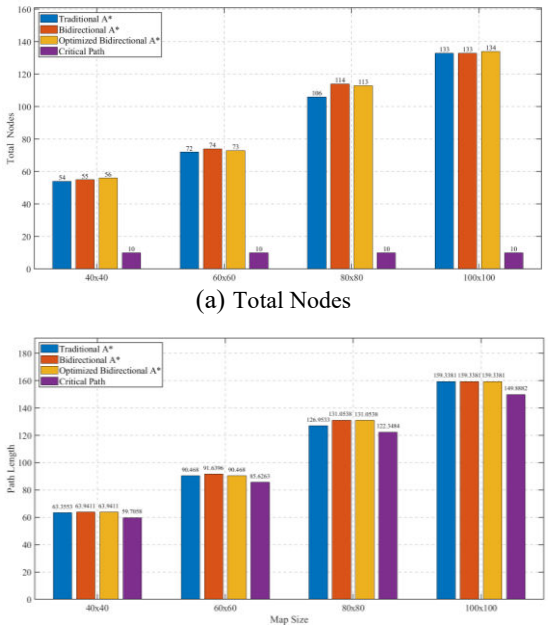


Fig.5 Critical Path on Four Grid Maps of Varying Scales

C. Simulation and Analysis of the Minimum Snap Algorithm

The Minimum Snap algorithm is the final module in the BiA*-MS path planning framework. Its primary function is to smooth the key path, generating the final path that satisfies kinematic constraints. Since the optimized path generated by the Minimum Snap algorithm is the final output of the BiA*-MS path planning framework, its performance directly determines the overall performance of the entire framework.



(a) Total Nodes

(b) Path Length

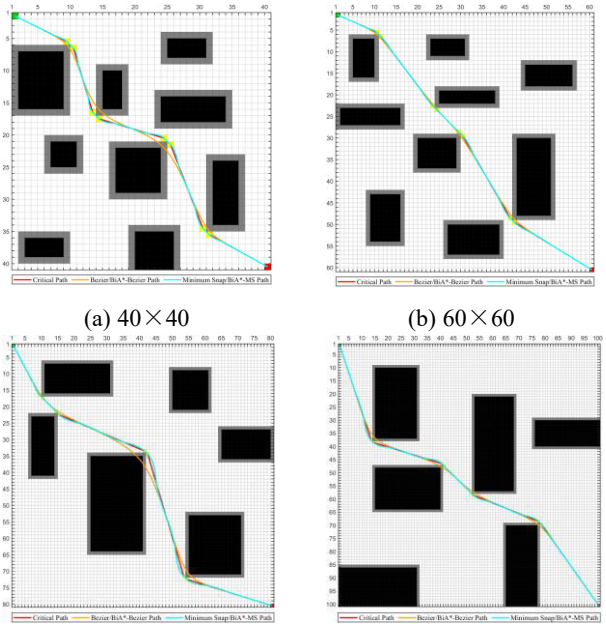
Fig.6 Comparison of Total Nodes and Path Length Across Algorithms on Four Grid Maps of Varying Scales

Therefore, the experiments in this section not only validate the performance of the Minimum Snap algorithm but also serve as a verification of the overall performance of the BiA*-MS path planning framework.

To validate the performance of the Minimum Snap algorithm, a set of comparative experiments is designed using the Bézier path smoothing method. The Bézier algorithm generates smooth curves through linear interpolation and polynomial equations. The basic principle is to first connect the control points to form a control polygon, then use the Bézier curve formula to approximate this polygon, thereby generating the final Bézier curve. For k critical path nodes (control points) $P_1, P_2, ..., P_k$, the Bézier curve is defined as:

$$B(t) = \sum_{i=1}^{k} \binom{k-1}{i-1} (1-t)^{k-i} t^{i-1} P_i, \quad t \in [0,1] \tag{19}$$

In the equation, t is the time parameter, $\binom{k-1}{i-1}$ is the binomial coefficient, and P_i is the i-th critical path node. The Bézier curve algorithm only depends on the positions of the control points to determine the path shape, without explicitly specifying which points the curve must pass through. This imposes certain limitations on its ability to avoid obstacles at the critical nodes. Therefore, to improve the obstacle avoidance capability of the smooth path, the critical path nodes, the quarter points between every two critical nodes, and their midpoints are selected as control points for both the Minimum Snap and Bézier algorithms in the comparative experiments. Since the generated smooth path is both the optimized result of the Minimum Snap algorithm and the final output of the BiA*-MS framework, the path smoothed by the Minimum Snap algorithm is defined as the BiA*-MS path. Similarly, the path smoothed by the Bézier algorithm is defined as the BiA*-Bézier path. The smooth paths generated by both algorithms based on the critical path in grid maps of different scales are shown in Fig.7.



(a) 40×40 (b) 60×60

(c) 80×80 (d) 100×100

Fig.7 Path Optimization Results of Bezier and Minimum Snap Algorithms Using Critical Nodes and Midpoints as Control Points on Four Grid Maps of Varying Scales

By comparing the two smoothed paths, it can be observed that the Bézier algorithm is highly dependent on the number and distribution of control points, resulting in relatively limited path optimization, especially in terms of obstacle avoidance performance. In contrast, the Minimum Snap algorithm, with its continuity constraints, demonstrates superior performance in both smoothness and obstacle avoidance, generating more continuous and safer paths that better meet practical requirements.

In conclusion, the Minimum Snap algorithm outperforms the Bézier algorithm in path optimization, showcasing greater flexibility and adaptability. This further highlights the advantages of the BiA*-MS framework: by combining the optimized Bidirectional A* algorithm, the Critical Node Retention algorithm, and the Minimum Snap algorithm, it not only enables faster generation of shorter and more concise paths but also achieves efficient path smoothing optimization, meeting the practical application needs in complex environments.

V. Conclusion

This paper proposes a new path planning framework, BiA*-MS, which combines the optimized Bidirectional A* algorithm, the Critical Node Retention algorithm, and the Minimum Snap algorithm. The framework aims to address several issues in traditional path planning algorithms, including long search times, redundant path nodes, excessive turns, and discontinuities in velocity and acceleration at path corners. First, BiA*-MS improves search efficiency by optimizing the Bidirectional A* algorithm, effectively reducing path search time. Next, the Critical Node Retention algorithm filters path nodes to generate a critical path, reducing both path length and the number of turns, further simplifying the path. Finally, the Minimum Snap algorithm smooths the path, ensuring continuity in position, velocity, and acceleration at the turns, thus improving the feasibility and safety of the path. Simulation experimental results demonstrate that the BiA*-MS framework offers significant advantages in terms of path search efficiency, path optimization quality, and obstacle avoidance performance. It generates smoother, more continuous, and safer paths, meeting the path planning requirements in complex environments. This framework not only enhances the efficiency and reliability of path planning but also provides an effective solution for real-world applications.

References

[1] PATLE B, PANDEY A, PARHI D, et al. A review: On path planning strategies for navigation of mobile robot [J]. Defence Technology, 2019, 15(4): 582-606.
[2] MIAO C, CHEN G, YAN C, et al. Path planning optimization of indoor mobile robot based on adaptive ant colony algorithm [J]. Computers & Industrial Engineering, 2021, 156: 107230.
[3] KARUR K, SHARMA N, DHARMATTI C, et al. A survey of path planning algorithms for mobile robots [J]. Vehicles, 2021, 3(3): 448-68.
[4] CHANG L, SHAN L, JIANG C, et al. Reinforcement based mobile robot path planning with improved dynamic window approach in unknown environment [J]. Autonomous robots, 2021, 45: 51-76.
[5] AB WAHAB M N, NAZIR A, KHALIL A, et al. Improved genetic algorithm for mobile robot path planning in static environments [J]. Expert Systems with Applications, 2024, 249: 123762.
[6] HENG H, GHAZALI M H M, RAHIMAN W. Exploring the application of ant colony optimization in path planning for Unmanned Surface Vehicles [J]. Ocean Engineering, 2024, 311: 118738.
[7] LIN S, LIU A, WANG J, et al. An improved fault-tolerant cultural-PSO with probability for multi-AGV path planning [J]. Expert Systems with Applications, 2024, 237: 121510.
[8] TU H, DENG Y, LI Q, et al. Improved RRT global path planning algorithm based on Bridge Test [J]. Robotics and Autonomous Systems, 2024, 171: 104570.
[9] WANG R, LU Z, JIN Y, et al. Application of A* algorithm in intelligent vehicle path planning [J]. Mathematical Models in Engineering, 2022, 8(3): 82-90.
[10] HAO K, ZHAO J, WANG B, et al. The application of an adaptive genetic algorithm based on collision detection in path planning of mobile robots [J]. Computational Intelligence and Neuroscience, 2021, 2021(1): 5536574.
[11] LIU C, WU L, XIAO W, et al. An improved heuristic mechanism ant colony optimization algorithm for solving path planning [J]. Knowledge-based systems, 2023, 271: 110540.
[12] LI X, WU D, HE J, et al. An improved method of particle swarm optimization for path planning of mobile robot [J]. Journal of Control Science and Engineering, 2020, 2020(1): 3857894.
[13] PAIKRAY H, DAS P, PANDA S. Optimal multi-robot path planning using particle swarm optimization algorithm improved by sine and cosine algorithms [J]. Arabian Journal for Science and Engineering, 2021, 46(4): 3357-81.
[14] MENG L, QING S, JUN Z Q. UAV path re-planning based on improved bidirectional RRT algorithm in dynamic environment; proceedings of the 2017 3rd International Conference on Control, Automation and Robotics (ICCAR), F, 2017 [C]. IEEE.
[15] ZHANG H, WANG Y, ZHENG J, et al. Path planning of industrial robot based on improved RRT algorithm in complex environments [J]. IEEE Access, 2018, 6: 53296-306.
[16] ZHANG J, WU J, SHEN X, et al. Autonomous land vehicle path planning algorithm based on improved heuristic function of A-Star [J]. International Journal of Advanced Robotic Systems, 2021, 18(5): 17298814211042730.
[17] LIU H, ZHANG Y. ASL-DWA: An improved A-star algorithm for indoor cleaning robots [J]. Ieee Access, 2022, 10: 99498-515.
[18] ZHANG D, CHEN C, ZHANG G. AGV path planning based on improved A-star algorithm; proceedings of the 2024 IEEE 7th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), F, 2024 [C]. IEEE.
[19] KANG N K, SON H J, LEE S-H. Modified A-star algorithm for modular plant land transportation [J]. Journal of Mechanical Science and Technology, 2018, 32: 5563-71.
[20] YU J, HOU J, CHEN G. Improved safety-first A-star algorithm for autonomous vehicles; proceedings of the 2020 5th International Conference on Advanced Robotics and Mechatronics (ICARM), F, 2020 [C]. IEEE.
[21] XIA D, SHEN B, ZHENG Y, et al. A bidirectional-a-star-based ant colony optimization algorithm for big-data-driven taxi route

recommendation [J]. Multimedia Tools and Applications, 2024, 83(6): 16313-35.
[22] WANG P, LIU Y, YAO W, et al. Improved A-star algorithm based on multivariate fusion heuristic function for autonomous driving path planning [J]. Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering, 2023, 237(7): 1527-42.
[23] ZHANG Z, JIANG J, WU J, et al. Efficient and optimal penetration path planning for stealth unmanned aerial vehicle using minimal radar cross-section tactics and modified A-Star algorithm [J]. ISA transactions, 2023, 134: 42-57.
[24] SUI F, TANG X, DONG Z, et al. ACO+ PSO+ A*: A bi-layer hybrid algorithm for multi-task path planning of an AUV [J]. Computers & Industrial Engineering, 2023, 175: 108905.
[25] PANG Y-X, YUAN D-C. Mobile Robot Path Planning Based on Fusion of Improved A* and DWA Algorithms [J]. Computer and Modernization, 2022, (01): 103.
[26] LI C, HUANG X, DING J, et al. Global path planning based on a bidirectional alternating search A* algorithm for mobile robots [J]. Computers & Industrial Engineering, 2022, 168: 108123.

Ying Zhang is a master's student majoring in software engineering at Tiangong University of Technology, mainly researching path planning and trajectory tracking control for mobile robots.

Xiaohui An is a master's student at the School of Software, Tiangong University of Technology, majoring in Electronic Information Software Engineering and researching trajectory tracking and control of unmanned aerial vehicles.