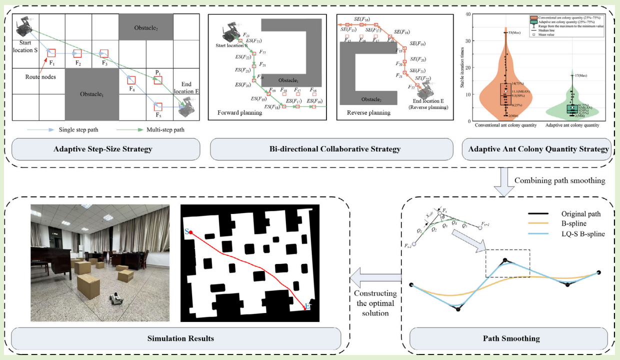
A New Robot Navigation Algorithm Based on Bi-Directional Collaborative Ant Colony Optimization

Xufei Chen, Yaowei Hu, Pingping Tang, *Member, IEEE*, Hui Zhang, *Member, IEEE*, Jiong Jin, *Member, IEEE*, and Shiwen Mao, *Fellow, IEEE*

***Abstract*—Swarm intelligence has been widely adopted and successfully applied in the field of autonomous robot navigation. Among various swarm intelligence algorithms, ant colony optimization (ACO) has shown significant potential in addressing complex navigation challenges. However, ACO faces challenges, for example, unclear initial search direction, slow convergence, limited ant flexibility, and the need to simplify robot motion control. To address these challenges, this article presents a novel bi-directional collaborative ACO (BC-ACO) algorithm with key innovations. First, the algorithm adopts a bi-directional ant colony with forward and reverse populations, achieving effective route planning through collaborative decision-making. Second, the algorithm employs an adaptive step-size strategy and a stage-based exploration ant quantity adjustment method. These innovations optimize the balance between exploration and exploitation, accelerate convergence, and address the inefficiencies of traditional ACO methods. Additionally, this article improves the heuristic function by integrating a node distance index within the bi-directional ant colony, guiding the transition of ants between nodes and further accelerating convergence. Simulation results show that BC-ACO reduces the computation time by 73.97% and improves the convergence stability by 63.64% compared to standard ACO. Additionally, BC-ACO successfully plans optimal paths, achieving a 20.85% reduction in path length. Further integration of a local quadratic segmented B-spline (LQ-S B-spline) curve results in paths with smooth transitions, reducing robot motion complexity. In summary, BC-ACO achieves fast global convergence, high stability, and short computation time.**



***Index Terms*—Bi-directional ant colony, collaborative decision-making, node distance index, segmented B-spline, stage-based exploration ant quantity adjustment.**

Received 14 February 2025; accepted 19 February 2025. Date of publication 4 March 2025; date of current version 16 April 2025. This work was supported in part by the National Natural Science Foundation of China under Grant 62071005 and Grant 62071252, in part by the National Key Research and Development Program of China under Grant 2020YFB2104004, in part by NSF under Grant CNS-2415208, in part by the Natural Science Foundation of Anhui Province under Grant 2308085Y02 and Grant 2208085MF155, and in part by the Natural Science Foundation of the Higher Education Institutions of Anhui Province under Grant KJ2021A0124. The associate editor coordinating the review of this article and approving it for publication was Dr. Chiman Kwan. *(Corresponding author: Pingping Tang.)*

Xufei Chen, Yaowei Hu, and Pingping Tang are with the School of Physics and Electronic Information, Anhui Normal University, Wuhu 241002, China (e-mail: chenxufei@ahnu.edu.cn; hyw0911@ahnu.edu.cn; tpping@ahnu.edu.cn).

Hui Zhang is with the School of Communication and Information Engineering, Nanjing University of Posts and Telecommunications, Nanjing 210003, China (e-mail: zhhjoice@126.com).

Jiong Jin is with the School of Engineering, Swinburne University of Technology, Melbourne, VIC 3122, Australia (e-mail: jiongjin@swin.edu.au).

Shiwen Mao is with the Department of Electrical and Computer Engineering, Auburn University, Auburn, AL 36849 USA (e-mail: smao@ieee.org).

Digital Object Identifier 10.1109/JSEN.2025.3545192

I. Introduction

WITH the rapid advancement of next-generation information technologies, including the Internet of Things and artificial intelligence, the manufacturing industry increasingly adopting intelligent systems. Autonomous navigation is a cutting-edge technology that enables robots to navigate and operate independently without human intervention [1]. Effective path planning is essential for achieving intelligent behavior in robots. Current path-planning methods are generally divided into three main categories: traditional algorithms, sampling-based algorithms, and intelligent algorithms.

Traditional algorithms, such as Dijkstra's algorithm [2], the A* algorithm [3], and the artificial potential field (APF) method [4], rely heavily on precise and comprehensive environmental data, limiting their adaptability to dynamic or complex environments. While Dijkstra's algorithm guarantees a globally optimal solution, its applicability is constrained by high computational complexity. Similarly, the A* algorithm relies on a carefully chosen heuristic function, which is often difficult to define in high-dimensional search spaces, leading to suboptimal performance when dealing with complex environments. Meanwhile, the APF method is prone to local minima, particularly in environments with dense obstacles or irregular layouts, which can cause the robot to become trapped and unable to reach its target, which reduces its effectiveness.

Sampling-based algorithms rely on randomness and often fail to guarantee path optimality. The rapidly exploring random tree (RRT) algorithm [5] produces paths that typically lack smoothness and necessitate postprocessing for optimization. It is also prone to narrow passage issues in environments with complex obstacles. The probabilistic roadmap method (PRM) [6], while capable of efficiently precomputing paths in static environments, suffers from relatively long graph construction times and limited adaptability to dynamic conditions. Moreover, these algorithms are highly sensitive to sampling strategies and search parameters, inappropriate choices can drastically diminish their efficiency.

Swarm intelligence, an evolving field in artificial intelligence, is characterized by self-organization, cooperation, and high adaptability to dynamic environments, which are achieved through individual decision-making and agent interaction [7]. Simple behaviors of individual agents and self-organizing interactions among them are observed in nature, exemplified by ant colonies and bird flocking. These phenomena have inspired the development of "artificial colonies of agents" capable of solving complex optimization problems [8]. Swarm intelligence is distinguished by the emergence of collective intelligence that surpasses the capabilities of individual agents. Drawing on these observations, researchers have focused on analyzing the behavioral patterns of groups to enhance swarm intelligence systems, leading to the development of a series of swarm intelligence algorithms, including ant colony optimization (ACO) [9], [10], [11], particle swarm optimization (PSO) [12], [13], [14], artificial fish swarm (AFS) [15], [16], bacterial foraging optimization (BFO) [17], [18], genetic algorithm (GA) [19], and artificial bee colony (ABC) [20], [21], [22]. These intelligent algorithms have been widely applied to optimization problems across various engineering fields, for example, scheduling, robotics, power systems, parameter optimization, system identification, image processing, and signal processing [23], [24], [25], [26].

In autonomous robot navigation, ACO is known for its applicability and robustness and has been widely applied to path planning. However, in practical applications, several issues remain, including slow convergence, limited flexibility of ants, and a tendency to get trapped in local optima. Numerous studies have sought to improve ACO's performance in path planning. Luo et al. [27] introduced strategies including nonuniform pheromone distribution, pseudorandom state transition rules, adaptive adjustment mechanisms, and dynamic penalty methods. Sun et al. [28] improved the heuristic function of ACO by incorporating factors including communication transmission distance, transmission direction, and the remaining energy of nodes. This improvement enabled the discovery of optimal data transmission paths in wireless sensor networks, significantly reducing average energy consumption and extending network lifespans. In [29], a novel pheromone update strategy called "ant weight" was introduced, along with a mutation operator. These studies primarily focused on refining ACO's initial parameters and pheromone update strategies, but they overlooked the drawbacks of excessive pheromone concentration, which can hinder ACO's exploration capabilities. To address this issue, Li et al. [30] proposed pheromone concentration with a terminal distance index and a fixed step size strategy to clarify ACO's exploration direction. Nonetheless, due to the limitation of updating the terminal distance index based solely on the results of each unidirectional search, the iterative update rate of the index is too slow, affecting the convergence speed of the results. Alobaedy et al. [31] investigated the effects of varying the number of ants on ACO's performance. Experiments demonstrated that optimizing the number of ants based on the specific problem model can significantly improve the algorithm's efficiency. Yet, this study did not examine how varying colony sizes at different stages impact the algorithm's outcomes.

Some explorations presented other algorithms to enhance planning efficiency while re-optimizing paths to simplify control [32], [33]. Huang et al. [34] combined the incremental flow method with a novel solution construction strategy into ACO, developing an ant colony evacuation planner to optimize evacuation path solutions. Sun et al. [35] employed the fruit fly optimization algorithm (FOA) to presearch the map, obtaining the initial pheromone distribution necessary for ACO exploration, thereby accelerating the planning process. Nevertheless, such measures have also led to an increase in computing resources, and there has been no attention paid to the situation of getting stuck in local optimal solutions due to U-shaped obstacles. Liu et al. [36] introduced a geometry-based local path optimization algorithm that effectively eliminates path redundancy. Yang et al. [37] developed and applied a segmented B-spline path smoothing algorithm to refine paths near turning points, achieving a collision-free trajectory that adheres to the robot's motion constraints. Despite this, while meeting the requirements for smooth paths, we should also focus on achieving a high degree of fitting with the initial path. In summary, there are still some unresolved issues, that is, the drawbacks of pheromone and fixed step-length mechanisms, slow convergence speed, the influence of stage-specific ant colony quantity on algorithm outcomes, U-shaped traps, and the need for simplified robot motion control.

Accordingly, we propose a new robot navigation algorithm that divides the robot path-planning problem into two stages, that is, path generation and turning point smoothing. This article proposes a bi-directional collaborative ACO (BC-ACO) algorithm for autonomous robot navigation path generation. The proposed algorithm introduces key innovations, including an adaptive step-size strategy, a BC strategy, an adaptive ant colony quantity strategy, and a refactored heuristic function, with each innovation contributing to the design of an optimal, obstacle-free path. Unlike traditional methods, our approach leverages a dynamic and responsive framework that adapts to changing conditions, enhancing both efficiency and accuracy. Based on the initial path generated, we employ a local quadratic segmented B-spline (LQ-S B-spline) curve that precisely fits at the turning points, ensuring the smoothness of the optimal path trajectory. By integrating BC-ACO with LQ-S B-spline turning point smoothing, a comprehensive robot path-planning solution is achieved. The major contributions are summarized as follows.

1) Proposing a BC strategy that enables effective path planning through cooperation between forward and backward populations. Existing methods typically rely on single-direction exploration or have insufficient design for bi-directional structures. Our approach introduces the node distance index for the first time, using it to make BC decisions. This method effectively addresses the problem of local optima and significantly improves the efficiency of path planning.
2) Presenting an adaptive step-size strategy and dynamic adjustment of ant colony size to effectively balance exploration and exploitation, accelerate algorithm convergence, and optimize performance at different stages. The adaptive step-size strategy improves on previous works, while the adaptive ant colony quantity strategy is novel and newly introduced.
3) In constructing the bi-directional decision structure, this article defines the concept of the node distance index and demonstrates that this index can effectively replace Euclidean distance in the heuristic function. This not only improves computational efficiency but also enhances the directionality of path planning. By incorporating the node distance index into the bi-directional ant colony, the heuristic function is improved, guiding ants more effectively toward the target node. This innovation significantly accelerates the algorithm's convergence and enhances path-planning accuracy.
4) Proposing the LQ-S B-spline method optimizes local control points, resulting in smoother paths and significantly reduced computational overhead. By segmenting the path optimization, we decrease the complexity of overall path adjustments while enhancing path operability, particularly in complex autonomous navigation environments.

The remainder of this article is organized as follows. Section II defines the path-planning problem based on the grid map. Section III provides detailed explanations of the improvement strategies, planning steps of BC-ACO, and convergence proof. The smoothing of path turning points using the LQ-S B-spline curve is presented in Section IV. Section V provides the results of computer simulation experiments and analysis of spatiotemporal complexity. Finally, Section VI summarizes the content of this article.

II. Environment Description and Problem Overview

Various map representation methods have been developed for the autonomous navigation problem, for example, grid-based, topological-based, and visibility graph-based approaches. Among these, the grid-based method is widely adopted due to its simplicity in modeling and ease of implementation. Therefore, this study chose to use the grid map to simulate the operating environment of BC-ACO. Some

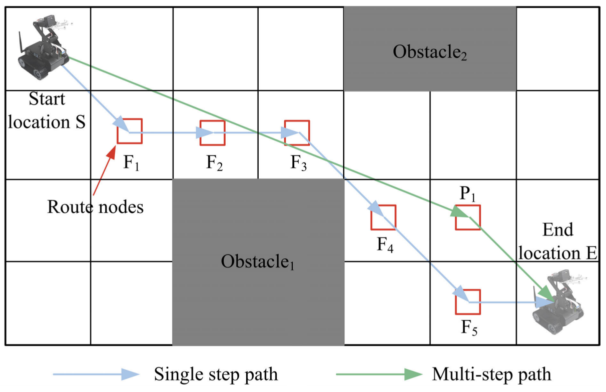


Fig. 1. Comparison of single-step and multistep forward path planning.

obstacles are placed on the map, and the positions of all the obstacles are known. The grid map is divided into free space grids and obstacle grids, as shown in Fig. 1, where black grids denote obstacle areas and white grids denote free passage areas.

The autonomous robot navigation problem on a grid map is defined as follows. Given a starting grid S and an end grid E, identify a set of grid data $\kappa : [0, t] \rightarrow G_{\text{free}}$, with $\kappa(0) = S$ and $\kappa(t) = E$, where G_{free} is the position of free passage areas and the path is represented by a sequence of grids in κ that connects the starting point S to the target E. Upon obtaining the initial path, the LQ-S B-spline curve is applied to smooth the path, thereby generating the optimal trajectory for the robot's movement. This optimal trajectory must ensure collision avoidance between the robot and any obstacles [38]. We aim to optimize the BC-ACO algorithm from four aspects, that is, path length, smoothness, convergence speed, and overall duration. Consequently, path planning is thus characterized as a multifaceted optimization problem with constraints [39].

III. Bi-Directional Collaborative ACO

A. Adaptive Step-Size Strategy

In the traditional ACO framework, the movement step size of ants is fixed at one, limiting their activity range to the seven adjacent grids surrounding the current node. This constraint significantly reduces the flexibility of the ants, increasing the overall complexity of the algorithm and negatively impacting the smoothness and efficiency of path planning. To address this issue, this article introduces a flexible node selection strategy to remove the limitation of step size, enabling ants to move with greater flexibility in terms of both distance and direction within a single step. Increasing the number of nodes searched in a single step reduces the total number of searches, thereby improving the algorithm's adaptability. The logic behind the feasible node selection strategy involves selecting nodes that can be directly reached from the current node without encountering obstacles as potential next-move nodes. Specifically, the pseudocode for the feasible node selection method is provided in Algorithm 1. A comparison between single-step and multistep advancements in path planning is presented in Fig. 1.

By analyzing the comparison of advancement methods, it is evident that after implementing the adaptive step-size

Algorithm 1 Feasible Node Selection

```
Input: current location node $n_a(x_a, y_a)$, position of
       obstacles $G_{obs} = \{G_{obs}(1), G_{obs}(2), \ldots, G_{obs}(n)\}$.
Output: transferable node list $allowed_a$.
$allowed_a \leftarrow \varnothing$.
for $b = $ S to E do
    Obtain coordinate $n_b(x_b, y_b)$.
    $d_{a,b} \leftarrow$ Distance between $n_a$ and $n_b$.
    if $b \in G_{obs}$&&Conflicts between line segment $l_{ab}$
    and $G_{obs}$ then
        $allowed_a \leftarrow \{x | x \in allowed_a\}$.
    else
        $allowed_a \leftarrow \left\{x | x \in allowed_a \bigcup x = b\right\}$.
return $allowed_a$.
```

strategy, ants in ACO are no longer constrained by a fixed step size when selecting forward nodes, effectively avoiding the selection of redundant nodes. This optimization strategy allows ants to identify an optimal path that is both shorter and smoother, significantly enhancing the algorithm's global optimization capability.

B. BC Strategy

To address the issues in ACO, for example, slow iterative convergence, poor quality of optimal solutions, and the U-shaped trap problem, this article proposes a BC strategy to enhance ACO. This strategy employs node distance index matrices $SE \in \mathbb{R}^{N_m \times N_m}$ and $ES \in \mathbb{R}^{N_m \times N_m}$ for the forward and reverse populations, respectively, where N_W represents the side length of the map. These matrices guide the selection of optimal nodes during the node selection process. The node distance index is promptly updated after each node transfer, following a preferential preservation strategy. Ants make node transfer decisions by evaluating the node distance index of each node. This index, along with the Euclidean distance from the current node to the target node, constitutes the heuristic function of the algorithm. This heuristic function enables ants to explore and optimize the path more efficiently during the search process. By continuously selecting nodes and updating the node distance index, the algorithm can more rapidly converge on the optimal solution in subsequent searches. Using forward exploration as an example, the pseudocode for the BC strategy is provided in Algorithm 2. Additionally, the use of bi-directional ant colonies can effectively address the issue of the algorithm falling into U-shaped traps during the early stages. In this article, we divide the ant colony population into forward and reverse populations, establishing a bi-directional search. The next node selected during node transfer is determined by the node distance index recorded by the opposing population. Specifically, the node with the smallest corresponding node distance index from the list of selectable transfer nodes is chosen. In the U-shaped trap scenario shown in Fig. 2, during the bi-directional search process, one of the ant colonies will manage to avoid the U-shaped trap, while the other will use the node distance index recorded by the opposing ant colony to navigate around the U-shaped trap.

Algorithm 2 Bi-Directional Collaborative Strategy

```
Input: current node $n_i$, transferable node list $allowed_i$,
       $ES$, forward path length $L_{\text{now}}$.
Output: $n_{\text{next}}$, update $SE$.
for all $j \in allowed_i$ do
    if $ES(j) = \min\{ES(s) | s \in allowed_i\}$ then
        $d_{i,j} \leftarrow$ Distance between $n_i$ and $n_j$.
        $\varepsilon_{i,j} = \frac{\beta_1 + \beta_2 \times k + \beta_3 \times M_f(k)}{ES(j) + d_{i,j}}$.
    else
        $\varepsilon_{i,j} = \frac{1}{ES(j) + d_{i,j}}$.
    Calculate probability $P_{ij}$.
Probability selection $n_{\text{next}}$.
$n_i \leftarrow n_{\text{next}}$.
$d_{i,\text{next}} \leftarrow$ Distance between $n_i$ and $n_{\text{next}}$.
$L_{\text{now}} = p_{\text{now}} + d_{i,\text{next}}$.
if $L_{\text{now}} < SE(\text{next})$ then
    $SE(\text{next}) = L_{\text{now}}$.
return $SE$.
```

C. Adaptive Ant Colony Quantity Strategy

In the early exploration stage of ACO, a certain degree of randomness affects the algorithm's efficiency and performance. This article proposes an adaptive strategy where the number of exploring ants varies based on the number of iterations. By dynamically adjusting the stage-based quantity of exploration ants at different iteration stages, this approach enables extensive exploration in the early stages and focused convergence in the later stages. The function for adaptively adjusting the number of ants is provided as

$$M_f(k) = M_r(k) = \left\lfloor \frac{\chi_{\text{total}}}{2 \cdot K_{\max}} \cdot \left(1 + \gamma \cdot \cos\left(\frac{\pi k}{K_{\max}}\right)\right) \right\rfloor \tag{1a}$$

$$\text{s.t. } \chi_{\text{total}} = \int_1^{K_{\max}} M \, dk \tag{1b}$$

where $M_f(k)$ and $M_r(k)$ represent the ant colony number functions for the forward and backward populations when the current number of iterations is k. χ_{total} denotes the total number of regular searches performed by the ant colony algorithm, γ is the adjustment coefficient controlling the dynamic fluctuation range of ant numbers, $K_{\max}$ is the maximum iteration limit, and M is the fixed number of ant colonies in each iteration of the regular exploration. To verify the effectiveness of this strategy, a comparative analysis will be conducted between the algorithm employing the adaptive ant colony quantity strategy and the one using the conventional ant colony quantity strategy. The comparison will focus on the optimal, worst, and average stable iteration counts. The comparison results are presented in Fig. 3. It can be observed that experiments without the adaptive ant colony quantity strategy yield an average stable iteration count of 11.1. In contrast, the application of this strategy reduces the average stable iteration count

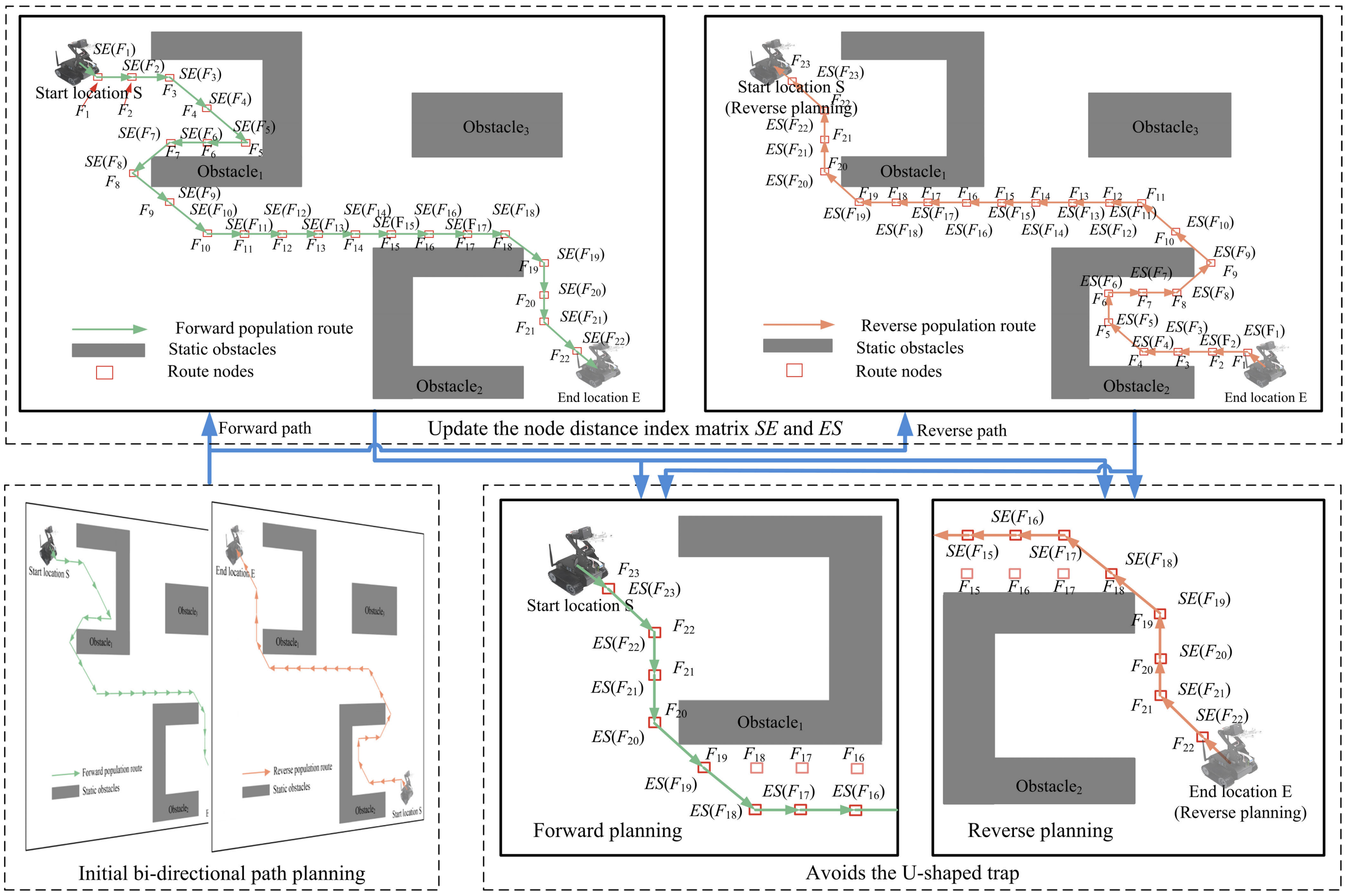


Fig. 2. Illustration of the process of avoiding U-shaped obstacles. The process is divided into three steps: initializing the bi-directional path planning, updating the node distance index matrices SE and ES, and avoiding U-shaped traps. The lines with light green arrows and brown arrows represent the forward and the reverse population paths, respectively. Red squares indicate path nodes, while gray squares represent obstacles. To clarify the process of avoiding U-shaped obstacles more explicitly, grid lines have been omitted from this diagram, although they are present in the actual diagram.

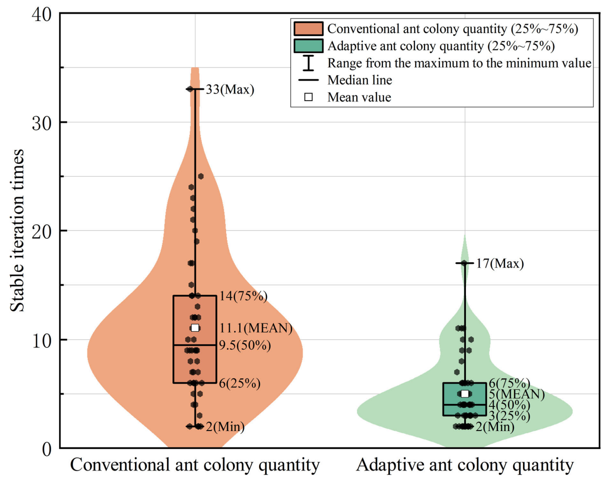


Fig. 3. Comparison chart of convergence results. When the exploration results reach stable iterations, this figure presents a comparison of the convergence results between the proposed algorithm using a conventional ant colony quantity and the one with an adaptive ant colony quantity. The brown and light green box plots represent the convergence results for the conventional and adaptive ant colony quantities, respectively.

to 5, resulting in an approximate performance improvement of 54.95%.

In summary, dynamically adjusting the population size enables ACO to thoroughly explore the map during the early stages, thereby discovering more potential solutions. This enhancement significantly improves the stability and global optimization capability of the algorithm, ensuring solution quality and preventing entrapment in local optima.

D. Refactoring Heuristic Functions

In ACO, pheromones play a crucial role. By simulating the pheromone mechanism observed in natural ant foraging behavior, ACO can search for optimal solutions on a global scale and solve many complex optimization problems. However, excessively high pheromone concentrations can negatively impact the algorithm, preventing it from correctly identifying better solutions due to the accumulation of excessive pheromone levels on the current path. To address this issue, this study proposes the use of bi-directional ant colony alternate exploration, adopting the node distance index from the opposing exploration to substitute pheromones in constructing a new heuristic function. Taking forward exploration as an example, node transfer selection is determined by a roulette wheel selection method. Building upon the work in [9] and [40], the formula has been improved, and the node probability transfer formula and the new heuristic function are detailed as

$$P_{ij} = \begin{cases} \dfrac{\varepsilon_{i,j}^{\alpha}}{\sum_{q\in \text{allowed}_i} \varepsilon_{i,q}^{\alpha}}, & \text{if } j \in \text{allowed}_i \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

$$\varepsilon_{i,j} = \begin{cases} \dfrac{\beta_1 + \beta_2 \times k + \beta_3 \times M_f}{ES(j) + d_{i,j}}, & \text{if } ES(j) = \min ES \\ & (\text{allowed}_i) \\ \dfrac{1}{ES(j) + d_{i,j}}, & \text{otherwise} \end{cases} \tag{3a}$$

$$\text{s.t. } ES(\text{allowed}_i) = \{ES(s) | s \in \text{allowed}_i\} \tag{3b}$$

where $\varepsilon_{i,j}$ is the heuristic function from node i to node j and α is the heuristic function influence factor. β_1, β_2, and β_3 are the probability transfer influence factors, $ES(j)$ is the node distance index of node j in ES, k is the current iteration count, and allowed$_i$ is the set of feasible nodes for node i. Additionally, $\min ES(\text{allowed}_i)$ represents the minimum value in set $ES(\text{allowed}_i)$. The node distance index offers ants clearer guidance, faster iteration speeds, and the capability to avoid local optima during movement. To address the uncertainty in node transfer, an optimized probability extension is introduced. As demonstrated in (3a), when calculating the heuristic function $\varepsilon_{i,j}$, the probability extension is applied to the node corresponding to the minimum value in $ES(\text{allowed}_i)$.

By optimizing the node with the minimum cost, the algorithm progressively increases the probability of selecting this node as the number of iterations increases. This approach guides the algorithm to converge more rapidly toward the optimal solution, significantly improving the efficiency of the optimization process and accelerating convergence speed.

E. Process of BC-ACO

Based on the above analysis, the core elements of BC-ACO can be summarized as follows: adopting an adaptive step-size strategy to liberate the ant colony from the limitations of a fixed step size; utilizing a BC strategy to share node information and guide movement; employing an adaptive ant colony quantity strategy to increase the number of ants in the early iterations and decrease them in the later iterations; and reconstructing the heuristic function to be dominated by the node distance index. The standardized process definition of BC-ACO is as follows.

1) Initialize Parameters: Establish a grid map based on the obstacle node set $G_{\text{obs}} = \{G_{\text{obs}}(1), G_{\text{obs}}(2), \ldots, G_{\text{obs}}(n)\}$ and obtain the set of feasible nodes for each node through Algorithm 1 to generate the adaptive step-size adjacency matrix. Initialize the algorithm's defined parameters, including the heuristic function influence factor α, probability extension influence factors β_1, β_2, and β_3, the iteration limit $K_{\max}$, and the node distance indices SE and ES. Divide the ant colony into forward and reverse populations.

2) Preliminary Planning: Employ the traditional ACO for the initial stage of path planning to update SE and ES, providing a data calculation foundation for BC-ACO.

3) Adjust the Number of Ants: Dynamically adjust the number of ants in the current exploration according to (1a) as part of the adaptive ant colony quantity strategy during BC-ACO exploration.

4) Node Selection Method: Ants move between nodes until reaching the endpoint. Assume the current node is $n_i(x_i, y_i)$. Taking forward exploration as an example, the steps to determine the next transfer node n_{next} are as follows. For the set of feasible nodes allowed$_i$ of the current node n_i, obtain it through Algorithm 1. Perform the following operations for each feasible node, that is, query ES to obtain the node distance index set $ES(\text{allowed}_i)$ of all feasible nodes. When calculating heuristic function $\varepsilon_{i,j}$, apply probability extension to the node corresponding to the minimum value in $ES(\text{allowed}_i)$ to give it a higher selection probability, without changing other nodes, as shown in (3a).

5) Transfer and Update: Calculate the transfer probability for each node and select the next transfer node based on the roulette wheel method. The probability selection process for the roulette wheel is given below. Assuming the feasible node set for node n_i as

$$\text{allowed}_i = \{j_1, j_2, \ldots, j_m\} \tag{4}$$

where m is the number of elements in the set, and the selection probability for each node P_{ij} is calculated. At this point, the cumulative probability formula as

$$S_{i,j_p} = \sum_{t=1}^{p} P_{i j_t}. \tag{5}$$

By comparing a random number r with the cumulative probability, one node from the set allowed$_i$ is selected as the transfer node. The specific selection rule is shown as

$$j = \begin{cases} j_1, & \text{if } r \le S_{i,j_1} \\ j_2, & \text{if } S_{i,j_1} < r \le S_{i,j_2} \\ j_3, & \text{if } S_{i,j_2} < r \le S_{i,j_3} \\ \vdots & \\ j_m, & \text{if } S_{i,j_{m-1}} < r \le S_{i,j_m} \end{cases} \tag{6a}$$

$$\text{s.t. } r = \text{rand}(0, 1). \tag{6b}$$

After transferring to node n_{next}, update the node distance index $SE(\text{next})$ of node n_{next}. The update formula as

$$SE(\text{next}) = \begin{cases} L_{\text{now}}, & \text{if } L_{\text{now}} < SE(\text{next}) \\ SE(\text{next}), & \text{otherwise} \end{cases} \tag{7}$$

where L_{now} is the path length traveled. Record the nodes traversed and the path length L_{now} traveled.

6) Termination Condition: When all ants have completed their exploration and the maximum iteration limit $K_{\max}$ is reached, BC-ACO terminates. Output the shortest path L_{best} and the nodes contained in the path, obtaining the global optimal solution.

F. Convergence Proof

To prove the convergence of the algorithm, it is sufficient to demonstrate that the path length after each iteration tends to decrease or remain unchanged. The proof steps are as follows.

1) First, analyze the probability selection mechanism of the algorithm and prove how the algorithm consistently selects the optimal node during node transfer. According to (2), the probability of selecting the optimal node increases with the number of iterations as

$$\lim_{k\to\infty} p_{\min} = \frac{P_{i,\min}}{\sum P_{i,\text{allowed}_i}} = 1 \tag{8}$$

where min is the node corresponding to the minimum value in set $ES(\text{allowed}_i)$. In node transfer selection, the random probability selection during early exploration ensures solution diversity, guarantees the generation of global solutions, and prevents the algorithm from getting trapped in local optima. It is observed that the node with the minimum path node distance index gradually exhibits a higher probability of being selected. As the number of iterations, k, reaches a certain threshold, this probability will approach 1. At this stage, during node transfers in constructing the path solution, each selected node contributes to minimizing the total path length, thereby ensuring the convergence of the algorithm's result.

2) When the probability of selecting the optimal node approaches 1, it becomes crucial to analyze how the algorithm converges to the optimal solution. When transferring to a node, the estimated shortest path is expressed as

$$\text{path}^* = \text{path}_i^* + ES(i) \tag{9}$$

where path^* is the shortest path from the forward search, path_i^* is the current path length from the start to node i, and $ES(i)$ is the known shortest distance from node i to the endpoint. Continue node transfers according to the preferential transfer method. Assuming the next transfer node is j, then the estimated shortest path path^* will be less than or equal to the path length from the previous reverse planning, expressed as

$$\text{path}^* = \text{path}_j^* + ES(j) \leqslant \text{path}_i^* + ES(i). \tag{10}$$

If it is shorter than the previous reverse planning path length, the path nodes are recorded, and the forward node distance index matrix is updated. Consequently, in the next reverse search planning, the algorithm will follow the guidance of the forward node distance index. At this point, the estimated shortest path length of the reverse planning will also be less than or equal to the previous forward planning path length, illustrated as

$$\text{path}' \leqslant \text{path}^* \tag{11}$$

where path' is the shortest path from the reverse search. Equality occurs when the current planning path is identical to the previous forward planning path. Similarly, it can be deduced that with the increase in the number of iterations and the alternation of forward and reverse searches, there will be $\text{path}^* \leqslant \text{path}'$ in forward planning and $\text{path}' \leqslant \text{path}^*$ in reverse planning. Until the forward and reverse planning paths remain consistent and the forward and reverse node distance indices no longer update, that is, at this point, $\text{path}' = \text{path}^*$, the algorithm converges to the final result.

IV. Path Smoothing

The purpose of this section is to smooth the turning points of the route. When using B-spline curves to smooth the global path, the spacing between control points can be large, reducing the ability to control the curve effectively and potentially leading to conflicts between the optimized path and obstacles. Such conflicts significantly increase the risk of collisions. Therefore, we propose using the LQ-S B-spline curve path smoother to achieve a higher degree of curve fitting. The control points in a B-spline curve are crucial for defining the direction and boundary range of the spline curve. A B-spline curve $R(u)$ is defined by $n+1$ control points, denoted as $Q_i = \{Q_0, Q_1, Q_2, \dots, Q_n\}$, and a knot vector u

$$R(u) = \begin{bmatrix} Q_0 & Q_1 & \cdots & Q_n \end{bmatrix} \begin{bmatrix} C_{0,s}(u) \\ C_{1,s}(u) \\ \vdots \\ C_{n,s}(u) \end{bmatrix} = \sum_{i=0}^{n} Q_i C_{i,s}(u) \tag{12}$$

$$C_{i,s}(u) = \begin{cases} \begin{cases} 1, & u_i \le u < u_{i+1} \\ 0, & \text{otherwise} \end{cases}, & s = 1 \\ \dfrac{u-u_i}{u_{i+s-1}-u_i} C_{i,s-1}(u) + \dfrac{u_{i+s}-u}{u_{i+s}-u_{i+1}} C_{i+1,s-1}(u), & s \ge 2 \end{cases} \tag{13}$$

where $C_{i,s}(u)$ represents the basis function defined by the Carl de Boor recursion formula [41], s is the order of B-spline, and u_i is a set of continuously varying values of a nondecreasing sequence known as the knot vector, with the first and last values typically set to 0 and 1, respectively. The LQ-S B-spline curve adopts a local smoothing method, specifically targeting the turning points of the original path for smoothing. As depicted in Fig. 4, $F_{x-1} \to F_x \to F_{x+1}$ represents a section of the turning points in the original path, and we specifically smooth the turning point F_x. To facilitate this smoothing, five control points are set, that is, adding Q_1, Q_2 at $F_{x-1}F_x$, Q_4, Q_5 at F_xF_{x+1}, and Q_3 at Q_2Q_4. The positions of these control points are calculated as

$$d_{Q_2,F_x} = d_{F_x,Q_4} = X_{\text{safe}} \tag{14a}$$

$$\frac{d_{Q_1,Q_2}}{d_{Q_2,Q_3}} = \frac{d_{Q_3,Q_4}}{d_{Q_4,Q_5}} = 1 \tag{14b}$$

$$d_{Q_2,Q_3} = d_{Q_3,Q_4} = d_{Q_2,F_x} \cdot \sin\theta \tag{14c}$$

where X_{safe} represents the safety distance and θ denotes the angle between vectors $\overrightarrow{F_xQ_3}$ and $\overrightarrow{F_xQ_4}$. The B-spline curve is constrained by the five control points, that is, Q_1, Q_2, Q_3, Q_4, and Q_5. Similarly, this smoothing operation is applied at each turning point along the global path. Based on (14), the control points are determined, and an LQ-S B-spline is used to smooth the inflection point path, achieving the desired smooth route. As illustrated in Fig. 5, this provides a visual comparison between the LQ-S B-spline curve and the traditional B-spline.

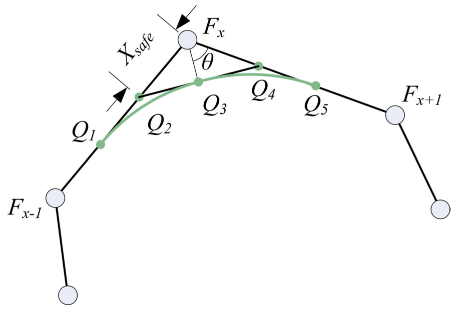


Fig. 4. Illustration of local smoothing using a quadratic segmented B-spline to smooth the path at turning points.

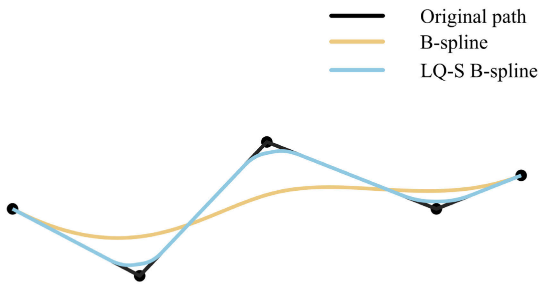


Fig. 5. Using traditional B-spline and LQ-S B-spline to smooth the same path segment. The black line represents the original path, while the light brown and light blue lines represent the paths fit by the traditional B-spline and the LQ-S B-spline, respectively.

It can be observed that the path smoothed by the LQ-S B-spline curve exhibits a higher degree of fit to the original path compared to the traditional B-spline, more effectively meeting the movement requirements of robots.

V. Simulation Experiments and Analysis

In this section, we verify the correctness of our conclusions through simulation experiments. We conducted simulations of the path-planning algorithm on grid maps and compared the results with those of other path-planning algorithms. The simulations were conducted on a desktop system equipped with a 12th Gen Intel[1] Core[2] i5-12400F processor, NVIDIA GeForce RTX 4060 GPU, and 16.0 GB of RAM under Windows 10.

A. Parameter Settings and Simulation Results

In this study, extensive experimental analysis was conducted to identify the optimal parameter values, ensuring the algorithm's effectiveness and performance. Parameters such as the number of ants and the significance of heuristic information were determined through rigorous experimentation and cross-validation. A wide range of experiments was performed with various parameter combinations to pinpoint the best configuration. The preset iteration limit, denoted as k, influences both the solution quality and computation time. Our findings indicate that when the number of iterations exceeds 20, additional computations result in minimal improvements in solution quality, while significantly increasing computation time. To facilitate a more straightforward comparative analysis, we set the iteration limit to 50, as other path-planning algorithms often fail to converge to an optimal result within 20 iterations. The number of ants is adaptively adjusted based on the initial size of the ant colony, ensuring thorough search coverage during each iteration while minimizing computational resource waste. Heuristic information key parameters, including β_1, β_2, and β_3, plays a critical role in affecting the transition probabilities. The selection principle for this parameter combines the transition probability with the current ant population and iteration count, guiding the algorithm toward gradual convergence under the heuristic function's influence. The key parameter settings related to BC-ACO are outlined in Table I. To demonstrate the feasibility and effectiveness of BC-ACO in complex environments, two complex maps were selected for simulation experiments, that is, 20 × 20 and 30 × 30 square grid areas, both featuring a high proportion of obstacles. In this simulation, comparisons will be made with ACO, IACO [27], and TDI-MSACO [30]. These comparisons will test and validate four critical aspects, that is, optimal path length, stable iteration count, number of nodes, and the runtime required to reach stable iterations. In addition, the comprehensive improvement percentage was calculated by weighting the composite indicators, where the weights of each indicator are equal. The statistical experimental data are presented in Table II, and the simulation results are shown in Fig. 6.

TABLE I
Key Parameter Settings

Parameter	Value	Description
$K_{\max}$	50	Number of maximum iterations
M	50	Number of ants
α	3	Weight of heuristic function
β_1	3.4	Probability transition influence factor
β_1	0.078	
β_3	0.02	
γ	0.5	Adjustment coefficient
X_{safe}	0.5	Safe distance

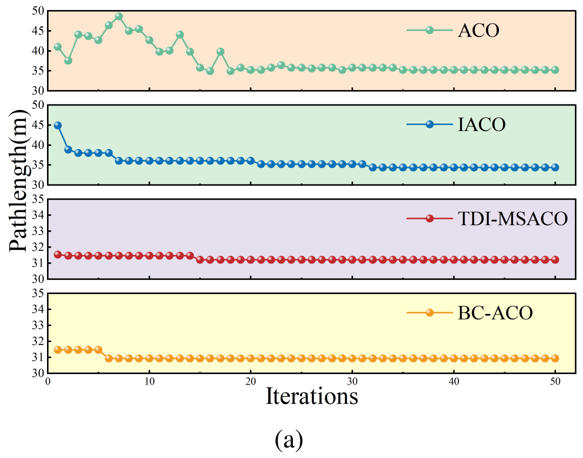
B. Comparative Analysis

Figs. 7 and 8 illustrate the optimal paths generated by each algorithm in the same complex environment. It is evident that the paths generated by BC-ACO are shorter and require fewer iterations. The simulation results also reveal that ACO and IACO are susceptible to falling into U-shaped traps in path planning, with this issue becoming more pronounced in complex environments. Additionally, while TDI-MSACO also encounters U-shaped traps during early exploration, its integration of the terminal distance index enables it to overcome these challenges in subsequent explorations. However, a drawback is that the method does not achieve rapid convergence, and with step lengths set to 2 or 3, TDI-MSACO may not reach a globally optimal solution in certain scenarios. Table II and Fig. 6 summarize the comparative data between the proposed algorithm and the other three algorithms, covering optimal path length, stable iteration count, number of nodes, and the runtime required to reach stable iterations.

[1]Registered trademark.
[2]Trademark.

TABLE II
COMPARISON SIMULATION RESULTS

Map scale	Algorithm	Optimal path length(m)	Variance	Standard deviation	Iterations	Nodes	Run time(s)	Improvement
20×20	ACO	35.2132	0.5917	0.7692	35	30	7.61	——
	IACO	34.3848	1.3271	1.1520	32	30	2.94	18.07%
	TDI-MSACO	31.2163	0.1145	0.3384	12	18	3.15	43.93%
	BC-ACO	**30.9300**	0.0009	0.0301	**6**	**9**	**1.69**	**60.71%**
30×30	ACO	59.6985	8.4733	2.9109	33	48	17.78	——
	IACO	50.6274	11.1965	3.3461	30	45	12.37	15.22%
	TDI-MSACO	47.6054	0.8871	0.9418	18	23	7.08	44.50%
	BC-ACO	**47.2496**	0.6758	0.8221	**12**	**14**	**4.61**	**57.35%**



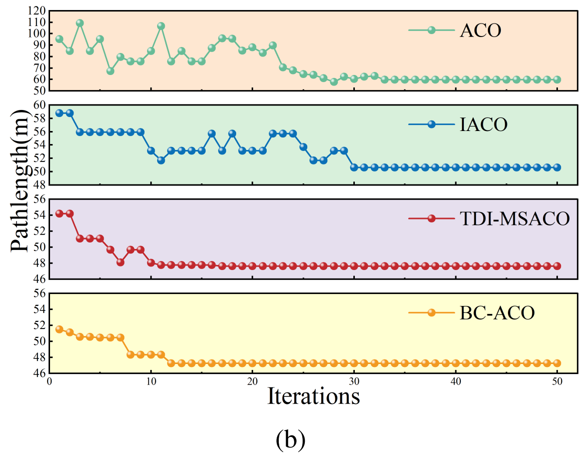


Fig. 6. Comparison of the optimal path length and stable iteration counts for different algorithms in simulations. (a) and (b) Simulation results for ACO, IACO, TDI-MSACO, and BC-ACO on 20 × 20 and 30 × 30 scale maps, respectively.

According to the simulation results, In a 20 × 20 environment experiment, BC-ACO achieved reductions in path length of 12.16%, 10.05%, and 0.92% compared to ACO, IACO, and TDI-MSACO, respectively. In terms of iteration count, number of nodes, and runtime, BC-ACO demonstrated particularly significant enhancements. Specifically, when compared to ACO, the iteration count was reduced by 82.86% and runtime decreased by 77.79%. When compared with other improved algorithms (IACO and TDI-MSACO), BC-ACO also exhibited notable optimization in operational efficiency and computational resource usage. Based on the variance and standard deviation calculations of 50 sets of experimental data, the BC-ACO algorithm exhibits strong robustness and effectiveness across maps of varying scales. The proposed algorithm consistently achieves the shortest path length and the lowest computation time across all scenario maps. Moreover, the advantages of our algorithm become more pronounced in complex maps.

In summary, as the complexity of the map model increases, BC-ACO not only exhibits high stability but also demonstrates significant advantages in terms of solution quality and computation time when compared to other algorithms.

C. Real Scene Path Planning

This section presents simulation experiments on path planning in real-world environments. Fig. 9 illustrates the generation of feasible paths on a real-world map. Initially, obstacles in the real environment are accurately marked, and a 2-D grid map is created. Next, the optimal path is planned and smoothed using the algorithm discussed in this study. Note that grid-based algorithms are limited by map resolution settings. High-resolution grid mapping can lead to an excessive number of cells, greatly reducing planning efficiency. Therefore, irregular obstacle regions should be adequately filled to reduce overall resolution, as shown in Fig. 10. Fig. 11 presents the results of indoor experiments, including the operational map and the optimal route trajectory. The laboratory environment was modeled as a 2-D grid map, and after the route was generated, motion commands were transmitted to the robot, allowing it to move along the predefined optimal path. Additional real-world environment trajectories are shown in Fig. 12. As shown in the trajectories, the proposed algorithm effectively finds the optimal solution for each map, and smoothing is applied at path turning points to ensure continuity. In summary, the experimental results confirm the practical effectiveness of this algorithm in real-world applications.

D. Analysis of Time and Space Complexities

In this section, we analyze the time and space complexity of the algorithm to visualize its performance and convergence speed. As shown in Fig. 13, the planning time of BC-ACO is 1.69 s in the 20 × 20 environment and 4.61 s in the 30 × 30 environment. It is evident that the computation time of BC-ACO is less than that of TDI-MSACO, IACO, and ACO. The main computations involved in BC-ACO include the following components.

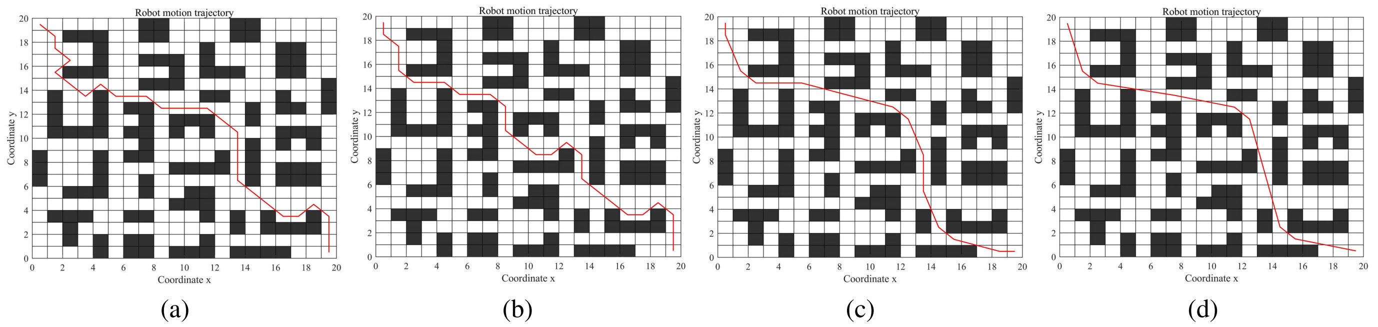


Fig. 7. Optimal path generated under 20 × 20 scale. Simulation results of (a) ACO optimal path, (b) IACO optimal path, (c) TDI-MSACO optimal path, and (d) BC-ACO optimal path.

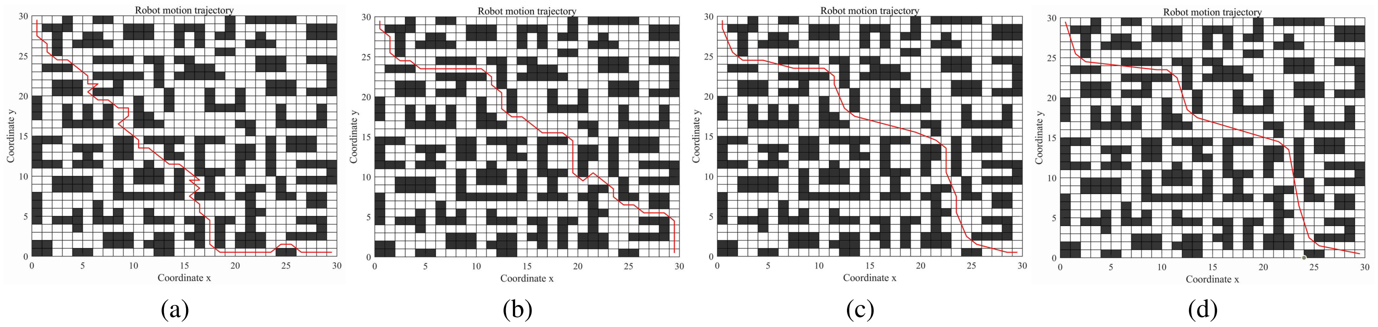


Fig. 8. Optimal path generated under 30 × 30 scale. Simulation results of (a) ACO optimal path, (b) IACO optimal path, (c) TDI-MSACO optimal path, and (d) BC-ACO optimal path.

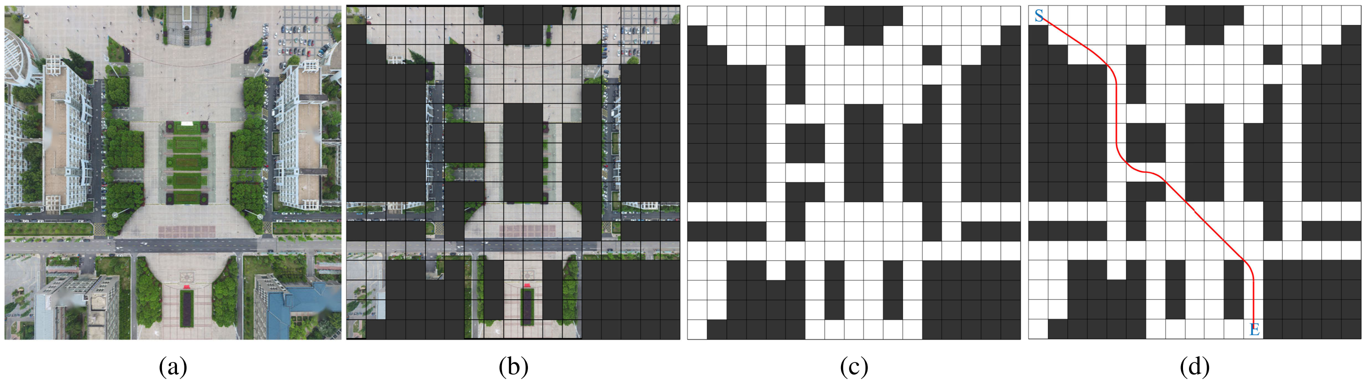


Fig. 9. Real-world scene path-generation process diagram. (a) Original map image. (b) Identify obstacles and feasible areas. (c) Generate a grid map. (d) Generate optimal path.

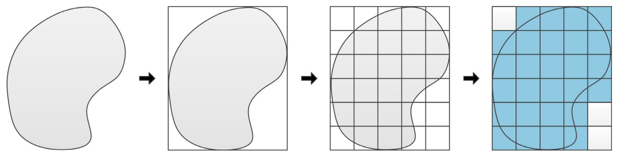


Fig. 10. Illustration of the irregular obstacle region grid-filling process.

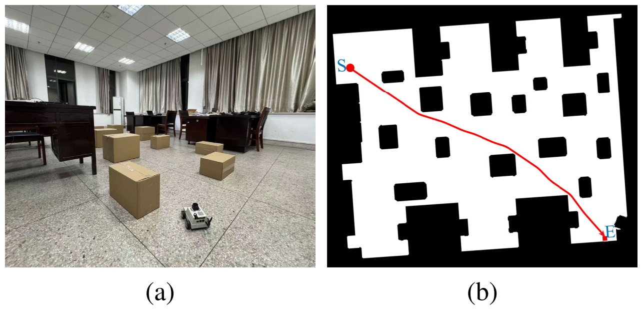


Fig. 11. Diagram of path generation in an indoor environment scene. (a) Real indoor environment. (b) Simulated generated path.

1) Initial Data Preprocessing: This step establishes the initial bi-directional node distance index matrix. According to Algorithm 2, the node distance index matrix can be updated by comparing the distances traveled by the nodes after transferring. For future comparison, the number of ant colonies here is uniformly set to M. The computational complexity for this update is $O(MN_a)$, as only one iteration of exploration by the ant colony is required to complete it, where N_a represents the average number of transferring nodes in each exploration. The space complexity necessary to record the node distance exponential matrix is $O(2N_m)$.

2) Constructing a Feasible Solution: When a single ant constructs a solution, it first calculates the probability of

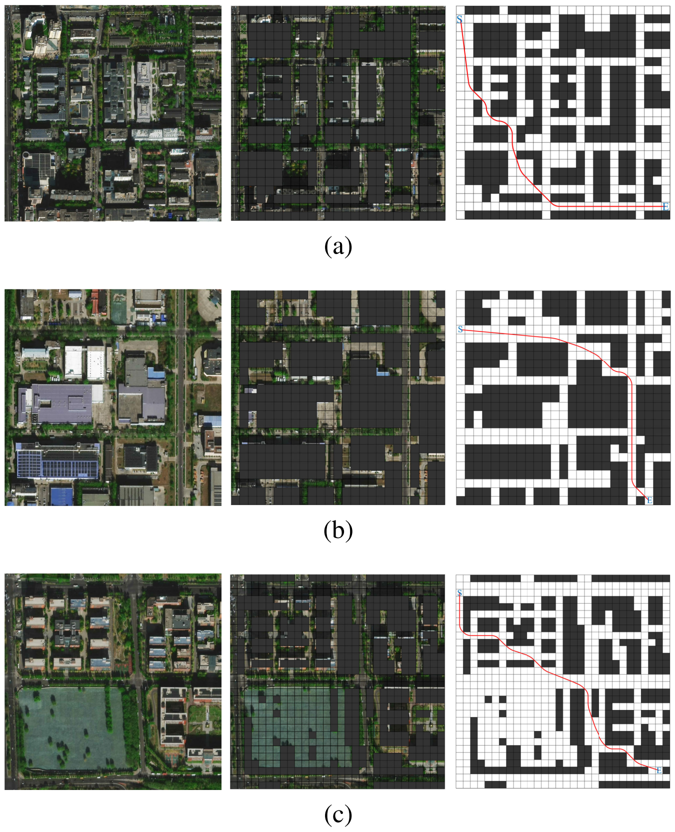


Fig. 12. Path planning in real environments. (a)–(c) Optimal path-planning process across four distinct real-world scenarios, including the original map display, obstacle grid generation, and optimal path generation.

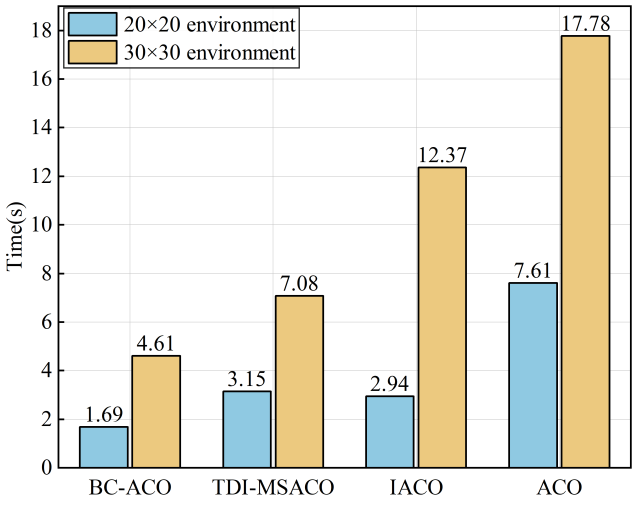


Fig. 13. Comparison chart of algorithm simulation time.

transferring to each node before the transfer occurs, utilizing a roulette wheel for selection. Additionally, after the node transfer, it performs a pairing operation. As a result, the time complexity of constructing a solution can be estimated as $O(N_t N_a)$, where N_t is the average number of optional nodes for each transfer.

3) Call of the Adjacency Matrix: During the exploration process, it is essential to determine which nodes to select by continuously referencing the adjacency matrix. To streamline data access, an auxiliary set is employed to record the current node's selectable and nonselectable nodes, with the size of this set denoted as N_m^2. Each time data is accessed, only the values in this set are updated. As a result, the space complexity of this part is $O(N_m^2)$.

4) Recording Feasible Solutions During Exploration: A set of approximately N_a is required to record the current feasible solutions in each exploration process, resulting in a space complexity of $O(N_a)$.

5) Recording the Optimal Solution: When documenting the optimal path in each iteration, in situ computation and an array with a resolution of N_a are employed to record the optimal path length and the nodes in the path, respectively. The time complexity for this operation is $O(N_a)$. The data is stored optimally, retaining only the current optimal result, leading to space complexities of $O(1)$ for the path length and $O(N_a)$ for the nodes.

As analyzed in Section V-D, the total time complexity of BC-ACO is $O(K_{\max} M N_a + K_{\max} M N_t N_a + K_{\max} M N_a) \approx O(K_{\max} M N_t N_a)$, and the overall complexity calculation depends on the structure of the algorithm. Since space complexity measures the size of the temporary extra storage space occupied by an algorithm during operation, it is not limited by the number of iterations $K_{\max}$. Thus, the overall space complexity is $O(N_m^2 + 2N_m + 2N_a + 1) = O((N_m + 1)^2 + 2N_a)$. Based on the previous simulation experiments, the parameters $K_{\max}$, M, and N_m are fixed. Here, we focus only on the variable parameters, that is, N_t and N_a. During the search process, the average number of nodes in the BC-ACO planning path, N_a, is significantly smaller than that of the compared algorithms. When the obstacles in the simulation environment become more complex, the values of N_t in each algorithm remain similar. However, the time loss for the compared algorithms increases significantly. Moreover, the temporal and spatial complexity of BC-ACO depends on N_t and N_a, while the temporal and spatial complexity of other methods depends not only on N_t and N_a but also on additional influencing factors. Therefore, the algorithm presented in this article exhibits the lowest time and space complexity.

VI. Conclusion

To address the limitations of ACO in global path planning, this article proposes BC-ACO for robot navigation. The adaptive step-size strategy precisely avoids unnecessary node selection, guiding ants toward shorter paths. The BC strategy effectively circumvents U-shaped traps, prevents entrapment in local optima, and promotes global progressive convergence, significantly enhancing both exploration and convergence rates. Additionally, integrating an adaptive ant colony quantity strategy with an improved heuristic function greatly enhances algorithm stability, reduces computational time, and improves overall performance. Furthermore, employing an LQ-S B-spline curve to smooth path turning points reduces robot braking and simplifies control. Extensive simulation results demonstrate that BC-ACO produces shorter and smoother optimal paths compared to ACO, IACO, and TDI-MSACO, while also outperforming them in iteration speed and runtime. These improvements in all aspects highlight BC-ACO's superior efficiency and effectiveness. This algorithm can be integrated into robotic control systems in industries requiring rapid adaptation to changing terrains or obstacles for autonomous navigation, helping design optimal routing solutions. Future research directions are as follows: 1) extending the application of the proposed algorithm to three-dimensional environments and 2) investigating cooperative path planning and trajectory optimization among multiple robots.

References

[1] J. Yang, Q. Ni, G. Luo, Q. Cheng, L. Oukhellou, and S. Han, "A trustworthy Internet of Vehicles: The DAO to safe, secure, and collaborative autonomous driving," *IEEE Trans. Intell. Vehicles*, vol. 8, no. 12, pp. 4678–4681, Dec. 2023.

[2] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Math.*, vol. 1, no. 1, pp. 269–271, Dec. 1959.

[3] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. Syst. Sci. Cybern.*, vol. SCS-4, no. 2, pp. 100–107, Jul. 1968.

[4] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Proc. IEEE Int. Conf. Robot. Autom.*, St. Louis, MO, USA, Jun. 1985, pp. 500–505.

[5] S. M. LaValle and J. J. Kuffner, "Rapidly-exploring random trees: Progress and prospects," *Algorithmic Comput. Robot., New Directions*, vol. 5, pp. 293–308, Nov. 2001.

[6] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. Robot. Autom.*, vol. 12, no. 4, pp. 566–580, Aug. 1996.

[7] D. Yu, J. Li, Z. Wang, and X. Li, "An overview of swarm coordinated control," *IEEE Trans. Artif. Intell.*, vol. 5, no. 5, pp. 1918–1938, May 2024.

[8] A. Slowik and H. Kwasnicka, "Nature inspired methods and their industry applications—Swarm intelligence algorithms," *IEEE Trans. Ind. Informat.*, vol. 14, no. 3, pp. 1004–1015, Mar. 2018.

[9] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: Optimization by a colony of cooperating agents," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 26, no. 1, pp. 29–41, Feb. 1996.

[10] J.-W. Lee and J.-J. Lee, "Ant-colony-based scheduling algorithm for energy-efficient coverage of WSN," *IEEE Sensors J.*, vol. 12, no. 10, pp. 3036–3046, Oct. 2012.

[11] S. Tao, Y. Xia, L. Ye, C. Yan, and R. Gao, "DB-ACO: A deadline-budget constrained ant colony optimization for workflow scheduling in clouds," *IEEE Trans. Autom. Sci. Eng.*, vol. 21, no. 2, pp. 1564–1579, Apr. 2024.

[12] X. Li and X. Yao, "Cooperatively coevolving particle swarms for large scale optimization," *IEEE Trans. Evol. Comput.*, vol. 16, no. 2, pp. 210–224, Apr. 2012.

[13] T.-Y. Wu and C.-H. Lin, "Low-SAR path discovery by particle swarm optimization algorithm in wireless body area networks," *IEEE Sensors J.*, vol. 15, no. 2, pp. 928–936, Feb. 2015.

[14] A. S. Al-Araji, A. K. Ahmed, and M. K. Hamzah, "Development of a path planning algorithms and controller design for mobile robot," in *Proc. 3rd Scientific Conf. Electr. Eng. (SCEE)*, Baghdad, Iraq, Dec. 2018, pp. 72–77.

[15] Z. Xiao and X. Wang, "Evolutionary niche artificial fish swarm algorithm for dynamic subgroup size adjustment in robot swarms," *IEEE Trans. Cognit. Develop. Syst.*, vol. 16, no. 4, pp. 1274–1290, Aug. 2024.

[16] X. Zhou, X. Yu, Y. Zhang, Y. Luo, and X. Peng, "Trajectory planning and tracking strategy applied to an unmanned ground vehicle in the presence of obstacles," *IEEE Trans. Autom. Sci. Eng.*, vol. 18, no. 4, pp. 1575–1589, Oct. 2021.

[17] J. Yi, D. Huang, S. Fu, H. He, and T. Li, "Multi-objective bacterial foraging optimization algorithm based on parallel cell entropy for aluminum electrolysis production process," *IEEE Trans. Ind. Electron.*, vol. 63, no. 4, pp. 2488–2500, Apr. 2016.

[18] M. A. K. Azad, A. M. A. C. Rocha, and E. M. G. P. Fernandes, "Improved binary artificial fish swarm algorithm for the 0–1 multidimensional knapsack problems," *Swarm Evol. Comput.*, vol. 14, pp. 66–75, Feb. 2014.

[19] Z. Zhu, X. Li, H. Chen, X. Zhou, and W. Deng, "An effective and robust genetic algorithm with hybrid multi-strategy and mechanism for airport gate allocation," *Inf. Sci.*, vol. 654, Jan. 2024, Art. no. 119892.

[20] W.-F. Gao, S.-Y. Liu, and L.-L. Huang, "A novel artificial bee colony algorithm based on modified search equation and orthogonal learning," *IEEE Trans. Cybern.*, vol. 43, no. 3, pp. 1011–1024, Jun. 2013.

[21] B. Ye et al., "A novel magnetic spiral capsule endoscope localization method based on an improved artificial bee colony algorithm," *IEEE Sensors J.*, vol. 24, no. 2, pp. 1740–1750, Jan. 2024.

[22] S. M. Hussein and A. S. Al-Araji, "Enhancement of a path-finding algorithm for the hovercraft system based on intelligent hybrid stochastic methods," *Int. J. Intell. Eng. Syst.*, vol. 17, no. 2, pp. 346–364, Feb. 2024.

[23] J. Tang, G. Liu, and Q. Pan, "A review on representative swarm intelligence algorithms for solving optimization problems: Applications and trends," *IEEE/CAA J. Autom. Sinica*, vol. 8, no. 10, pp. 1627–1643, Oct. 2021.

[24] W. Deng, J. Wang, A. Guo, and H. Zhao, "Quantum differential evolutionary algorithm with quantum-adaptive mutation strategy and population state evaluation framework for high-dimensional problems," *Inf. Sci.*, vol. 676, Aug. 2024, Art. no. 120787.

[25] J. Zheng, P. Liang, H. Zhao, and W. Deng, "A broad sparse fine-grained image classification model based on dictionary selection strategy," *IEEE Trans. Rel.*, vol. 73, no. 1, pp. 576–588, Mar. 2024.

[26] H. Zhao, Y. Wu, and W. Deng, "An interpretable dynamic inference system based on fuzzy broad learning," *IEEE Trans. Instrum. Meas.*, vol. 72, pp. 1–12, 2023.

[27] Q. Luo, H. B. Wang, Y. Zheng, and J. C. He, "Research on path planning of mobile robot based on improved ant colony algorithm," *Neural Comput. Appl.*, vol. 32, no. 6, pp. 1555–1566, 2020.

[28] Y. Sun, W. Dong, and Y. Chen, "An improved routing algorithm based on ant colony optimization in wireless sensor networks," *IEEE Commun. Lett.*, vol. 21, no. 6, pp. 1317–1320, Jun. 2017.

[29] B. Yu, Z.-Z. Yang, and B. Yao, "An improved ant colony optimization for vehicle routing problem," *Eur. J. Oper. Res.*, vol. 196, no. 1, pp. 171–176, Jul. 2009.

[30] D. Li, L. Wang, J. Cai, K. Ma, and T. Tan, "Research on terminal distance index-based multi-step ant colony optimization for mobile robot path planning," *IEEE Trans. Autom. Sci. Eng.*, vol. 20, no. 4, pp. 2321–2337, Oct. 2023.

[31] M. M. Alobaedy, A. A. Khalaf, and I. D. Muraina, "Analysis of the number of ants in ant colony system algorithm," in *Proc. 5th Int. Conf. Inf. Commun. Technol. (ICoIC7)*, Melaka, Malaysia, May 2017, pp. 1–5.

[32] O. Wahhab and A. Al-Araji, "Path planning and control strategy design for mobile robot based on hybrid swarm optimization algorithm," *Int. J. Intell. Eng. Syst.*, vol. 14, no. 3, pp. 565–579, Jun. 2021.

[33] H. T. Najm, N. S. Ahmad, and A. S. Al-Araji, "Enhanced path planning algorithm via hybrid WOA-PSO for differential wheeled mobile robots," *Syst. Sci. Control Eng.*, vol. 12, no. 1, pp. 1–18, Apr. 2024.

[34] Z.-M. Huang, W.-N. Chen, Q. Li, X.-N. Luo, H.-Q. Yuan, and J. Zhang, "Ant colony evacuation planner: An ant colony system with incremental flow assignment for multipath crowd evacuation," *IEEE Trans. Cybern.*, vol. 51, no. 11, pp. 5559–5572, Nov. 2021.

[35] J. Sun, Y. Yu, and L. Xin, "Research on path planning of AGV based on improved ant colony optimization algorithm," in *Proc. 33rd Chin. Control Decis. Conf. (CCDC)*, Kunming, China, May 2021, pp. 7567–7572.

[36] J. Liu, J. Yang, H. Liu, X. Tian, and M. Gao, "An improved ant colony algorithm for robot path planning," *Soft Comput.*, vol. 21, no. 19, pp. 5829–5839, Oct. 2017.

[37] H. Yang, J. Qi, Y. Miao, H. Sun, and J. Li, "A new robot navigation algorithm based on a double-layer ant algorithm and trajectory optimization," *IEEE Trans. Ind. Electron.*, vol. 66, no. 11, pp. 8557–8566, Nov. 2019.

[38] C. Zhang, Y. Li, and L. Zhou, "Optimal path and timetable planning method for multi-robot optimal trajectory," *IEEE Robot. Autom. Lett.*, vol. 7, no. 3, pp. 8130–8137, Jul. 2022.

[39] Z. Jian, S. Zhang, S. Chen, Z. Nan, and N. Zheng, "A global–local coupling two-stage path planning method for mobile robots," *IEEE Robot. Autom. Lett.*, vol. 6, no. 3, pp. 5349–5356, Jul. 2021.

[40] M. Dorigo and L. M. Gambardella, "Ant colony system: A cooperative learning approach to the traveling salesman problem," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 53–66, Apr. 1997.

[41] C. de Boor, "A practical guide to splines," *Appl. Math. Sci.*, vol. 27, no. 149, pp. 87–115, Jan. 1978.