scientific reports

OPEN

Conflict-based strategy combined integrated optimal conflict avoidance algorithm

Kejin Jia, Cheng Peng, Yun Du✉, Hongwei Wen, Yufei Gao & Zhe Zhang

This paper presents a Conflict-Based Strategy - Combined Integrated Optimal Conflict Avoidance (CBS-CIOCA) algorithm for more efficient multi-robot path planning. The algorithm first refines the conflict types in the high-level model of the traditional CBS algorithm and introduces two conflict categories: avoidable conflicts and unavoidable conflicts. Next, the low-level model of the traditional CBS algorithm is improved, transforming the path search process into two distinct algorithms with different focuses. Finally, based on the different conflict categories, two optimized algorithms are proposed: a space-time A* algorithm enhanced by diagonal improvements and a dynamic adaptive space-time A* algorithm incorporating dynamic adaptive factors. Experimental results demonstrate that, compared to the traditional CBS algorithm, the CBS-CIOCA algorithm achieves maximum reductions of 97.37% and 94.99% in time and node expansion, respectively, in both traditional warehouse and fishbone-shaped warehouse environments, as well as 88.37% and 88.41% in a dynamic large-obstacle environment.

Keywords Multi-robot path planning, Conflict-Based Strategy - Combined Integrated Optimal Conflict Avoidance, Space-time A* algorithm, Diagonally improved, Dynamically adaptive

Multi-agent pathfinding (MAPF), also referred to as multi-robot path planning, is a critical and complex problem in modern intelligent systems. In this task, multiple robots must navigate from a specified starting point to a target location within a predefined, complex environment, while avoiding collisions with each other[1]. In recent years, Multi-Agent Path Finding (MAPF) has found extensive applications in various domains, such as autonomous driving[2], airport scheduling[3–5], drone swarm formation[6], and agricultural production management systems[7,1].

In multi-robot global path planning, the key performance indicators for evaluating path planning algorithms include the total path length after all robots reach their target, the computational complexity of the path planning process, and the success rate of the multi-robot system in unknown environments[8–10]. In recent years, several robot path planning methods have been proposed to enhance efficiency under various conditions. Manny Shankar[11] et al. introduced a hybrid approach for mobile robot path planning, combining Particle Swarm Optimization (PSO) and Artificial Potential Field (APF) methods. This approach overcomes the problem of local minima encountered when using the APF algorithm alone and the slow planning speed of the PSO algorithm when used independently. However, its performance is suboptimal in more complex map environments. Lei et al.[12] proposed a graph-based robot path planning method using the improved Seagull Optimization Algorithm (iSOA) and developed an enhanced Douglas-Peucker (mDP) algorithm. This method approximates irregular obstacles in rugged terrain as polygonal shapes based on environmental images. Experimental results demonstrate that this approach effectively addresses path planning challenges in complex map environments; however, the experimental setup did not consider the effects of dynamic obstacles or other unforeseen environmental factors on the algorithm's performance. Pham et al.[13] introduced the Multi-ST model, which enhances the adaptability, efficiency, and collaboration of multi-robot systems through spatiotemporal optimization and task allocation. This model is particularly suited for task allocation and path planning in dynamic environments during multi-robot collaborations. However, as the number of robots and task complexity increase, the model may face challenges due to high computational complexity when efficiently managing task scheduling in multi-robot collaborative operations. Xidias et al.[14] tackled uncertainty in task allocation and motion planning by using fuzzy time windows, enabling multi-robot systems to collaborate under more flexible time constraints. Despite this advantage, the method faces challenges when handling large-scale multi-robot systems due to its high computational complexity. In contrast, Sharon et al.[15] proposed the Conflict-Based Search (CBS) algorithm, a

School of Electrical Engineering, Hebei University of Science and Technology, Shijiazhuang 050800, China. ✉email: Yunny7503@163.com

two-layer method that has proven to be one of the best solutions for the MAPF problem in recent years. In this approach, conflict detection and constraint setting are managed at the high level, while the low level searches for paths that meet these constraints. Jin et al.[16] improved the CBS approach by replacing the traditional A* algorithm with the Dlite algorithm, decoupling conflict management from path planning and coordination. They introduced the CBS-d algorithm, which incorporates a strategy for waiting and looping, enhancing the success rate of pathfinding and obstacle avoidance in the traditional CBS approach while reducing the number of time steps required.The methods of application and limitations of some of the algorithms discussed in the aforementioned literature are more clearly explained in Table 1.

Although the methods mentioned above are effective for solving multi-agent path planning problems, they mainly rely on traditional single-path planning algorithms to manage different types of conflicts at a higher level of abstraction. This approach may cause issues in multi-robot path planning, such as a robot's path becoming trapped in a local optimum, which in turn increases the overall path length[17,18]. Furthermore, neglecting collision resolution can lead to deadlocks, and as both the number of robots and the complexity of the environment increase, the likelihood of collision resolution failure also rises. To address these issues, this paper proposes an improved collision-based strategy-the Comprehensive Optimal Collision Avoidance (CBS-CIOCA) algorithm, designed specifically for robot path planning in large and complex environments. Unlike the traditional CBS algorithm, which focuses only on two higher-level types of collisions, the CBS-CIOCA algorithm clarifies the classification of high-level collisions by adjusting collision types based on the robots' different operational states, categorizing them into two main collision attributes. Additionally, while the traditional CBS algorithm uses a single path planning model, the CBS-CIOCA algorithm improves the underlying path planning logic by introducing two specialized path planning algorithms. Each algorithm is optimized for different collision attributes, overcoming the limitations of a single path planner. This enhancement allows the CBS-CIOCA algorithm to redefine collision attributes at a high level and optimize the lower-level path planning logic, offering a more flexible and efficient collision resolution strategy. This significantly reduces the risk of deadlocks and greatly improves path planning efficiency in highly dynamic and complex warehouse environments. Compared to traditional methods, the proposed algorithm exhibits superior performance in large-scale, multi-robot systems, especially in highly dynamic and complex environments, ensuring both the efficiency and reliability of path planning.

The contributions of the CBS-CIOCA algorithm are summarized below:

The CBS-CIOCA algorithm enhances the accuracy and efficiency of conflict management by redefining conflict types according to the robot's various operational states and by refining the classification of high-level conflicts into two distinct conflict attributes.

The CBS-CIOCA algorithm improves on the traditional CBS by integrating two specialized path planners, each optimized for different conflict types. This enhancement boosts efficiency and conflict resolution, especially in complex environments, overcoming the limitations of a single path planner.

Experimental results demonstrate that the CBS-CIOCA algorithm outperforms other approaches in terms of runtime, number of expanded nodes, and system robustness.

Description and modeling of the MAPF problem

Description of the MAPF problem

(1) The input is a directed graph $G(V, E)$, as illustrated in Fig. 1. Here, V denotes the set of vertices, and E represents the set of edges connecting these vertices.Mapping $s : [1, \ldots, k] \longrightarrow V$indicates that each agent

The name of the algorithm	Approaches of the algorithm	Limitations of the algorithm
An integrated framework of decision making and motion planning with oscillation-free capability	The main contribution of this study is to propose an integrated decision and motion planning framework that can ensure oscillation-free and safe driving in dynamic environments.	The limitation of the proposed approach lies in its reliance on assumptions about vehicle dynamics and the handling of soft constraints
A hybrid path planning approach	A Hybrid Approach to Mobile Robot Path Planning Combining Particle Swarm Optimization (PSO) Techniques and Artificial Potential Field (APF) Methods	Performs poorly in more complex map environments
Improved seagull optimization algorithm (iSOA)	Use the modified Douglas-Peucker (mDP) algorithm to approximate irregular obstacles as polygonal obstacles based on environmental images in rugged terrain, and then apply the iSOA method for path planning.	Lack of consideration of unknown environmental information such as dynamic obstacles in experimental setups
A novel coverage path planning model (termed Multi-ST)	The model utilizes the spiral-spanning tree coverage algorithm with intelligent reasoning and knowledge-based methods to achieve optimal coverage, obstacle avoidance, and robot coordination.	As the number of robots and the complexity of tasks increase, the model may face problems such as high computational complexity and the inability to perform efficient task scheduling with multiple robots operating in concert
Balanced task allocation and motion planning of a multi-robot system under fuzzy time windows	Handling uncertainty in task assignment and motion planning through fuzzy time windows enables multi-robot systems to collaborate under more flexible time constraints	There are problems such as high computational complexity and difficulty in dealing with large-scale multi-robot systems
Conflict Based Search algorithm (CBS)	The Conflict-Based Search (CBS) algorithm manages conflict detection and constraint setting at a high level, while the low level searches for paths that meet these constraints.	Lack of path planning ability in the presence of dynamic obstacles and unknown environments
Conflict-based search with D* lite algorithm	The traditional A* algorithm in CBS is replaced with the Dlite algorithm and a wait-and-loop strategy is investigated.	Path planning under unknown map conditions is not considered and has high computational complexity, making it difficult to deal with large-scale multi-robot systems.

Table 1. Key Algorithms and Their Approaches and Limitations.

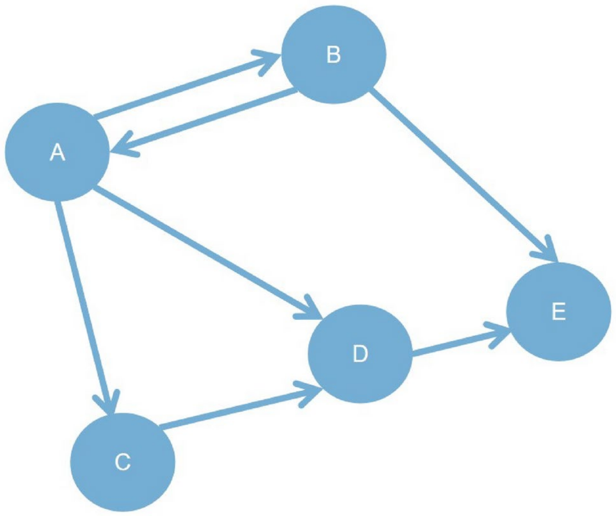


Fig. 1. Directed graphical models.

is associated with a starting node, while mapping $g:[1,...k]\longrightarrow V$ indicates that each agent is associated with a target node.

(2) $k$robots are represented as $a_1, a_2, ..., a_k$,with each robot having a starting point$s_i \in V$and a target point $g_i \in V$.Mapping $s:[1,...,k]\longrightarrow V$ specifies that each agent is associated with a starting node, while mapping $g:[1,...k]\longrightarrow V$ specifies that each agent is associated with a target node.

(3) let l_i^t represent the position of robot a_i at time step t,and let $P_i=\left\{s_i^0, s_i^1, ..., s_i^{T-1}\right\}$ denote the set of waypoints for robot a_i from s_i to g_i.

In the MAPF (Multi-Agent Pathfinding) problem, time is typically discretized into time steps. The path planning begins with a directed graph $G=(V,E)$ where m robots are distributed, represented by the set $A=\{a_1,a_2,...,a_k\}$. Each robot$a_i \in A$ must find a path $P_i=\{s_i,g_i\}$ from a given start point to a specified end point. $T=\{1,2,...,t\}$ denotes the discrete time sequence, and P_i represents the path of robot a_i. The function $T\rightarrow V$ maps the time sequence T to the vertex set V, with $l_{i(t)}$ indicating the position of robot a_i at any time $t \in T$. At each time step, every robot can perform one of five possible actions: move up, move down, move left, move right, or wait.Point conflict between two robots is defined as the scenario where both robots visit the same point at the same time step, denoted as $l_{i(t)}=l_{j(t)}$ Edge conflict occurs when both robots simultaneously traverse the same edge, represented by $l_{i(t)}=l_{j(t+1)}$ and $l_{i(t+1)}=l_{j(t)}$. The term $cost\left(l_i\right)$ represents the cost of the path l_i, measured in terms of time. Let t_s denote the start time and t_e denote the end time of the robot's path l_i. The optimization objective, known as the sum-of-costs (SOC), refers to the cumulative total of the time steps required for all robots to reach their target points, including any additional time steps spent beyond the target, expressed as:

$$SOC=\sum_{i=1}^{m} cost\left(l_i\right)$$

Additionally, the MAPF problem should encompass more than just minimizing time steps. The objective of this paper is to provide a feasible solution for each robot to find a path from its starting position to its destination, while minimizing the overall cost as much as possible.

As illustrated in Fig. 2, robots a_i and a_j encounter a vertex conflict at t=2 because they occupy the same location. Additionally, robots a_j and a_k face an edge conflict during the time intervals t=1, 2, and 3 as they traverse the same edge. Hence, the MAPF approach should not only focus on finding the optimal paths but also address potential conflicts with other robots during the path planning process.

Environmental modeling

In the classic MAPF model, various types of environmental maps are considered, such as traditional warehouse layout maps, fishbone-style warehouse layout maps, and complex maze layout maps. This paper utilizes the fishbone-style warehouse layout map for environmental modeling, following the same map modeling procedures as those used in subsequent experiments.

As shown in Fig. 3, in the research on mobile robot path planning, the grid method is an effective approach for modeling the workspace. This method involves dividing the working environment of the mobile robot into equally sized grids. White grids denote traversable areas, while black grids indicate obstacles. The robot navigates by moving along the centerlines of the traversable grids. This method is both straightforward and effective,

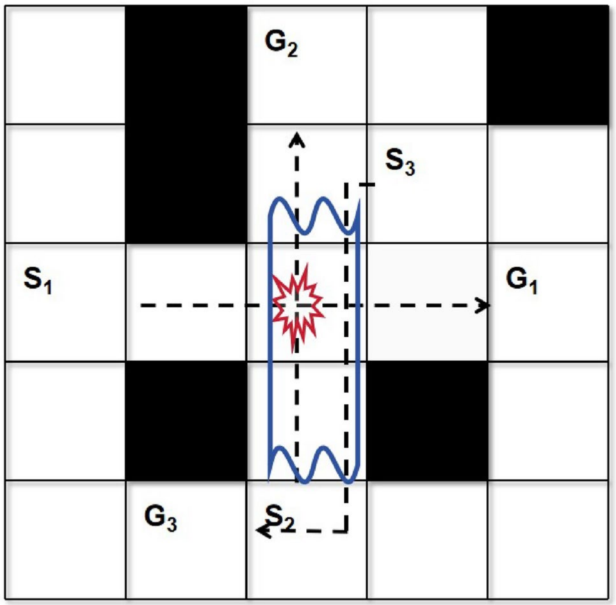


Fig. 2. Diagram of path conflicts in multi-Robot systems.

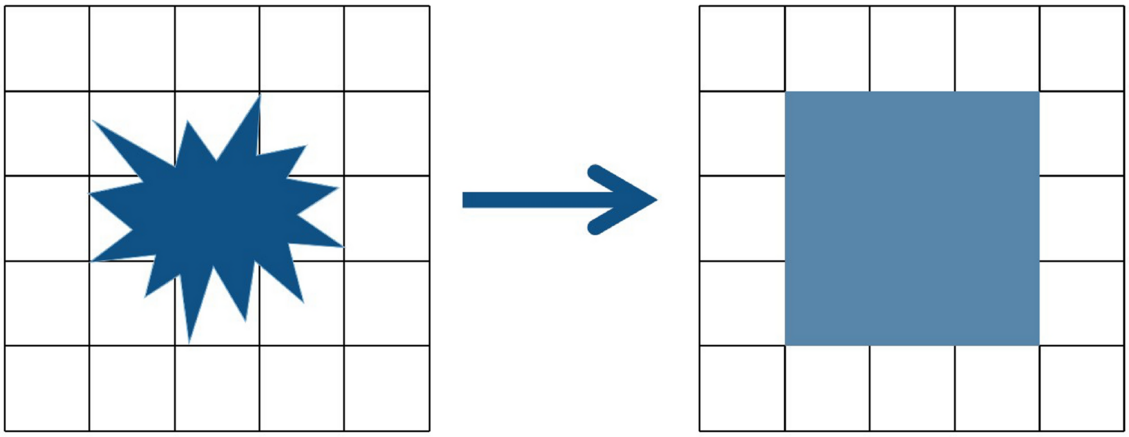


Fig. 3. Obstacle inflation in grid-based maps.

exhibiting strong adaptability to obstacles and significantly reducing environmental complexity. Hence, this paper employs this method for environmental map modeling.

To facilitate research and experimentation, the following assumptions are made:

1.The map and obstacle boundaries are established with consideration for the safety distance of mobile robots. Therefore, in simulations or experiments, the mobile robot can be treated as a point-like entity and will move within the grid map boundaries. 2.During the mobile robot's movement within the grid map, the surrounding environment is assumed to remain static. 3.The mobile robot is capable of moving to other unoccupied grid cells within the grid map, even in the presence of obstacles, by moving diagonally to the top-left, bottom-left, top-right, or bottom-right directions.

The paper employs a method that integrates the rectangular coordinate system with indexing to identify grids, where each grid is assigned a unique identifier and corresponding coordinates. As illustrated in Fig. 4, the mobile robot operates within a 37×18 grid map. The numbering of the grids begins at the top-left corner of the map, proceeding from top to bottom and from left to right. Grid coordinates increase from top to bottom and from left to right, respectively. The relationship between each grid's coordinates and its identifier is defined by formula:

$$\begin{cases} R_x = ceil\left(\frac{N}{rows}\right) \\ R_y = R_x * rows - N + 1 \end{cases} \tag{1}$$

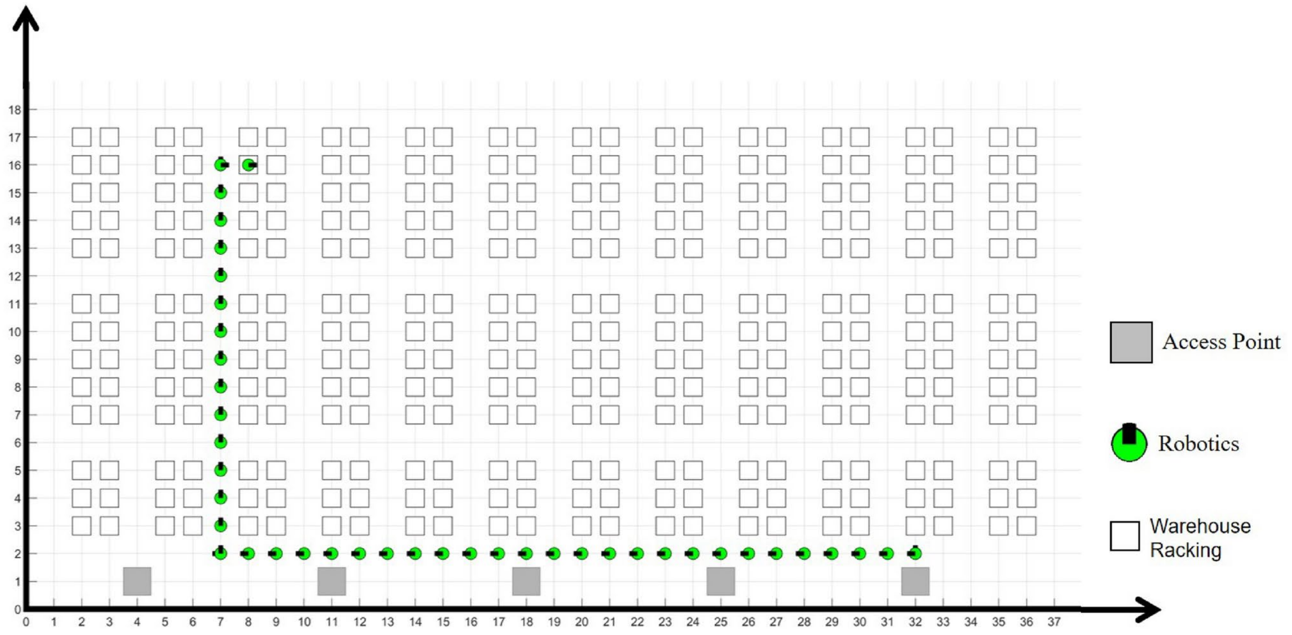


Fig. 4. Grid map of warehouse layout.

In the formula, *rows* denote the number of vertical grid cells in any column of the grid map. (R_x, R_y) indicates the coordinates (row and column) of any grid cell within the map. *N* represents the index of this grid cell in the map, *ceil* is an upward rounding operator.

The traditional warehouse layout, constructed using the grid map method, is illustrated in Fig. 4. This type of warehouse features straight main aisles as the core structure, complemented by parallel auxiliary pathways. In the figure, white hollow squares represent storage points, which are abstracted as obstacles and final destinations in the grid map method. Gray solid squares denote access points for storage and serve as the initial starting positions for the robots. When there are more than five robots, random tasks are generated on the map to simulate the emergence of tasks during robot operations. Red circles indicate the robots, and each intersection of the grid lines marks a potential path that the robots can traverse.

Conflict-based strategy - combined integrated optimal conflict avoidance algorithm

Conflict-based search algorithm

In multi-AGV path planning, the CBS algorithm solves the problem using a two-tier approach. At the lower tier, the algorithm employs a constrained single-agent path planning method to address the path planning issues of each individual AGV, typically utilizing the A* algorithm. The upper tier then reviews the paths planned by the lower tier to identify any conflicts. If conflicts are detected, constraints are imposed to adjust the lower-tier planning until all paths are free of conflicts, as shown in the algorithmic flow in Fig. 5.

As the scale of the map and the number of robots increase, the traditional A* algorithm often explores numerous unnecessary path nodes during the path planning process. This inefficiency can lead to increased solving times for the CBS algorithm and may even result in cases that are unsolvable. Furthermore, in traditional CBS algorithms, the conflict detection and resolution process is complicated by frequent conflicts and complex paths in the upper-level planning phase. The upper-level detection algorithm often merges various conflict types into a single constraint for the lower-level pathfinding algorithm. This conflation prevents the lower-level algorithm from effectively adjusting to different types of conflicts, leading to wasted computational resources and longer path lengths.

The paper first enhances the CBS algorithm by replacing its underlying pathfinding algorithm with the diagonal space-time A* algorithm and the dynamic adaptive space-time A* algorithm. The diagonal spatiotemporal A* algorithm offers faster search speeds and ensures optimal routing compared to the traditional space-time A* algorithm. In contrast, the dynamic adaptive space-time A* algorithm may utilize suboptimal routes as a trade-off, which significantly reduces the search time and the number of expanded nodes. Subsequently, during the conflict detection and resolution phase of the CBS algorithm, different types of conflicts are classified, and various underlying pathfinding algorithms are applied based on the specific characteristics of each conflict. This approach significantly enhances the overall system efficiency and success rate of the new algorithm.

Improvements for the underlying pathfinding algorithm

Space-time A algorithm*

The overall structure of the Spatial-temporal A* algorithm is similar to that of the A* search algorithm. Both algorithms perform pathfinding by repeatedly calculating the F values for nodes in the Open and Closed lists. They select the node with the lowest F value from the Open list as the starting point for the next iteration, and

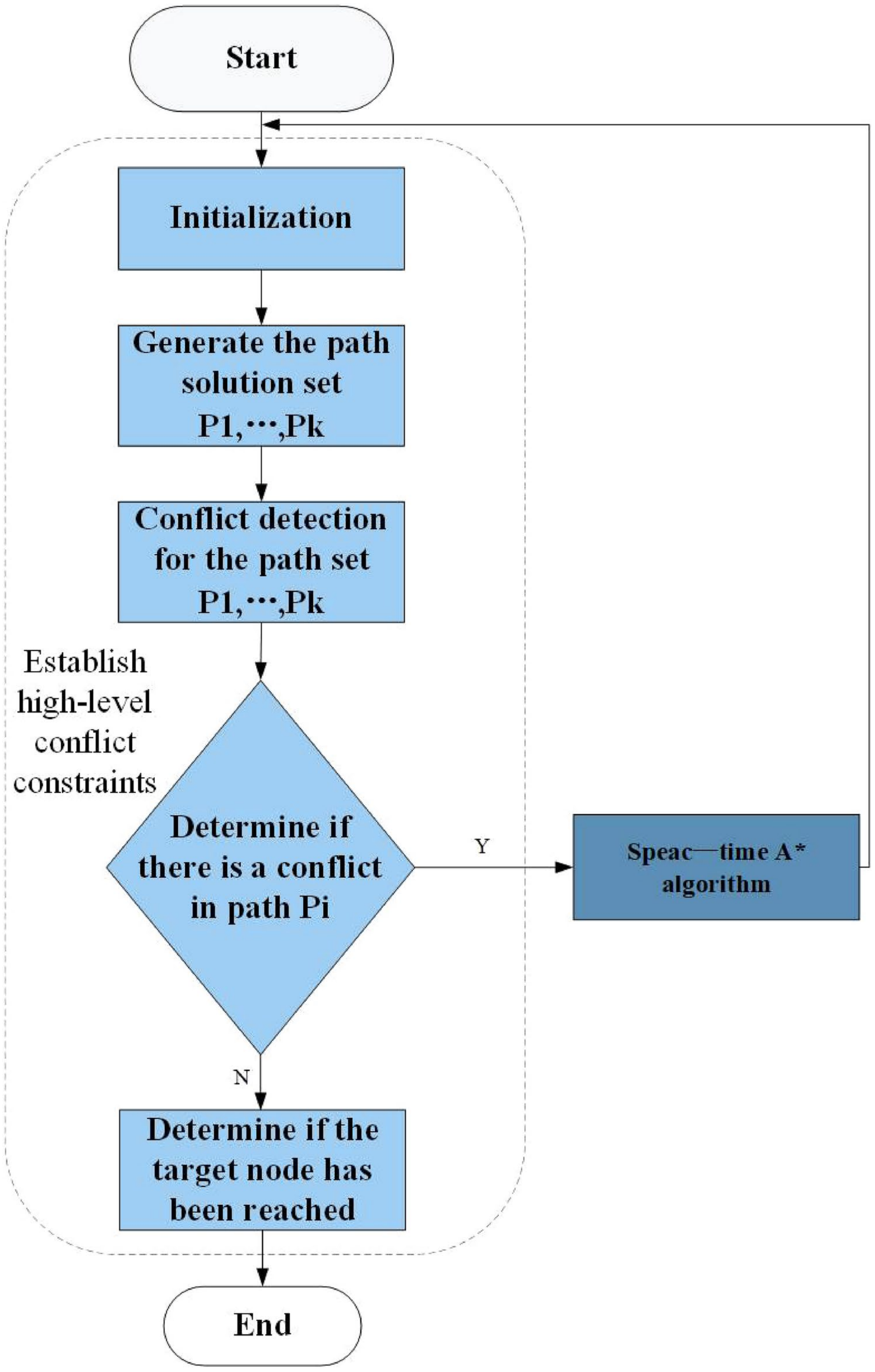

Start
Initialization
Generate the path solution set P1,⋯,Pk
Conflict detection for the path set P1,⋯,Pk
Establish high-level conflict constraints
Determine if there is a conflict in path Pi
Y
Speac—time A* algorithm
N
Determine if the target node has been reached
End

Fig. 5. CBS algorithmic framework.

continue updating until they reach the target node and identify all path nodes. The Open list stores the nodes that are expanded during the search, while the Closed list contains nodes with the minimal cost encountered during the search. The formulas for both the Spatio-temporal A* algorithm and the A* search algorithm are represented by:

$$f\left(n\right)=g\left(n\right)+h\left(n\right) \tag{2}$$

$f\left(n\right)$ represents the estimated cost to reach the goal state from the initial state through state n. $g\left(n\right)$ denotes the actual cost incurred to travel from the initial state to state n within the state space. $h\left(n\right)$ indicates the estimated cost of the optimal path from state n to the goal state.

The traditional A* algorithm, when applied to search in a two-dimensional spatial map, only requires recording the positions of expanded nodes. However, when extending the search to include a time dimension, it becomes necessary to also record the time at each position in addition to the spatial coordinates. In contrast, the Spatial-temporal A* algorithm integrates time as a state within the graph, where time progresses linearly. This approach transforms the graph into a spatiotemporal tree, where each level of the tree corresponds to an increment in time. In environments with a single Automated Guided Vehicle (AGV), conflicts do not arise as the environment is static, and path planning is performed solely within the two-dimensional spatial map. Conversely, when multiple AGVs are present, each operating AGV becomes a dynamic obstacle for others. Consequently, the spatiotemporal A* algorithm controls vehicles in both spatial and temporal dimensions, expanding the two-dimensional map into a three-dimensional spatiotemporal map to accommodate this complexity.

Using the small-scale map depicted in Fig. 6 as an example, we assume that the cost between each pair of adjacent points is uniform. When point A represents the starting location of the Automated Guided Vehicle (AGV) and point E represents the destination, the search tree generated by the Space-time A* algorithm is illustrated in Fig. 7.

In the tree diagram, each branch point is labeled with letters that indicate node positions, while the associated numbers represent the time points at which these nodes are reached. The search begins at time 0, with each transition to an adjacent node taking one unit of time. As illustrated in Fig. 6, node A only connects to node B, thus the next level of the tree comprises node B and its corresponding time point of 1. Node B connects to nodes F and C; therefore, the subsequent level includes nodes F and C with a time point of 2. Using the four-neighborhood approach, node C connects to nodes H, G, and D, and node F connects to node H. Consequently, the next level of the tree includes nodes H, G, and D, each with a time point of 3. This process continues iteratively until the target node E is reached.

Improved diagonal space-time A algorithm*

The estimated cost function $h\left(n\right)$ represents the cost from the current node n to the goal node. The choice of heuristic function for $h\left(n\right)$ can significantly impact the operational efficiency of different pathfinding algorithms. To ensure that the search algorithm finds the shortest path while minimizing the number of expanded nodes, selecting an appropriate heuristic function is crucial.

Figure 8 illustrates the commonly used heuristic functions for estimating $h\left(n\right)$, which include the Manhattan distance, Euclidean distance, and diagonal distance.

The Manhattan distance algorithm calculates the distance between the current node and the target node as the sum of the absolute differences in their x and y coordinates. This distance is represented by the heuristic function:

$$h_{Manhattan}=\left|x\left(s\right)-x\left(g\right)\right|+\left|y\left(s\right)-y\left(g\right)\right| \tag{3}$$

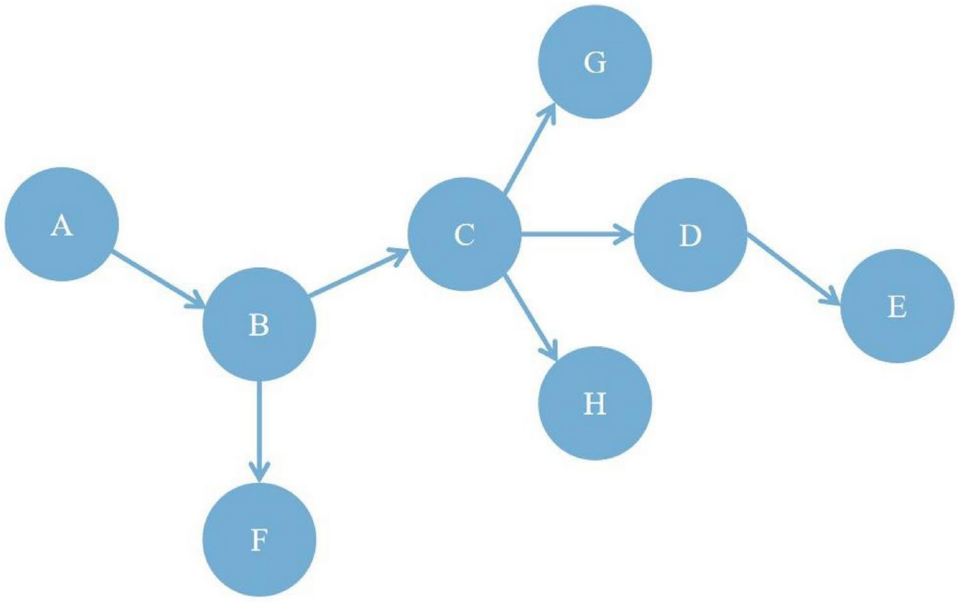


Fig. 6. Example of a simple map.

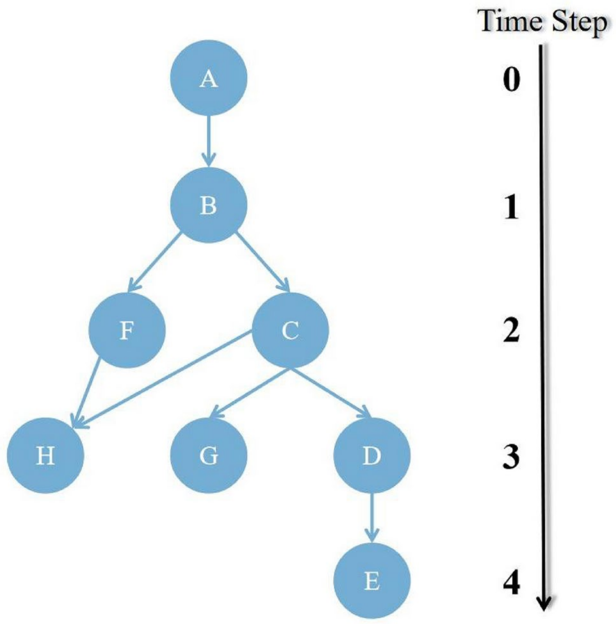


Fig. 7. Diagram of the time tree.

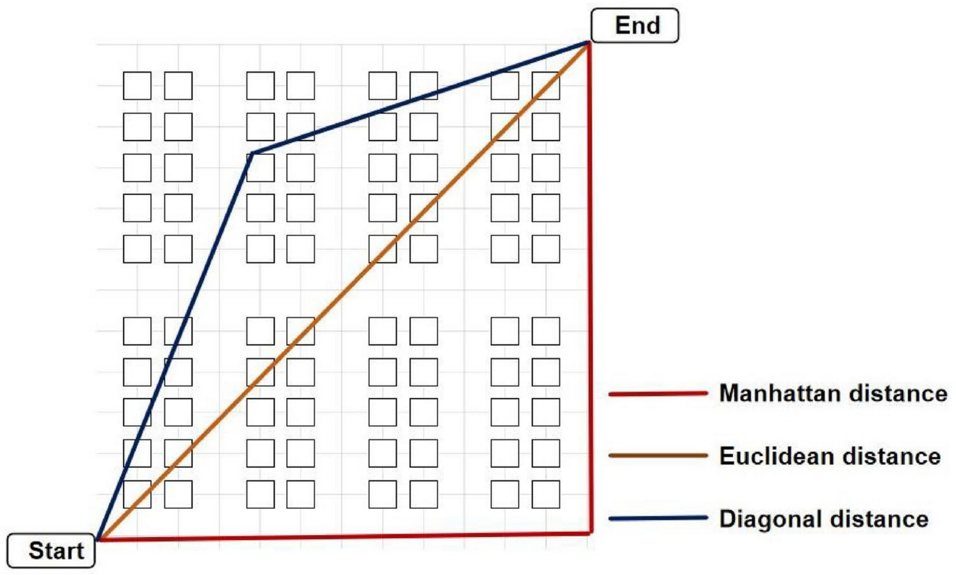


Fig. 8. Diagrams of three distance measurement methods in a 2D space.

The Euclidean distance measures the true distance between the current node and the target node, calculated as the square root of the sum of the squared differences between their horizontal and vertical coordinates. The heuristic function is given by:

$$h_{Euclid} = \sqrt{\left[x\left(s\right) - x\left(g\right)\right]^2 + \left[y\left(s\right) - y\left(g\right)\right]^2} \tag{4}$$

Diagonal distance is a specific case of Euclidean distance, commonly applied in path planning problems within grid or coordinate spaces. It represents the shortest distance between two points, measured as the maximum of the absolute differences in their horizontal and vertical coordinates. The heuristic function is given by:

$$h_{diagonal} = \left|x\left(s\right) - x\left(g\right)\right| + \left|y\left(s\right) - y\left(g\right)\right| + \left(\sqrt{2} - 2\right) * min\left(\left|x\left(s\right) - x\left(g\right)\right|, \left|y\left(s\right) - y\left(g\right)\right|\right) \tag{5}$$

In the CBS pathfinding algorithm, grid maps are used for global path planning, assuming that robots can move in four directions. In this context, the Manhattan distance may not ensure the optimal path, while the Euclidean distance tends to expand many unnecessary nodes. The algorithm performs optimally when the distance between the starting and target nodes closely matches the cost of moving between nodes. To address this, an optimization factor is introduced into the diagonal distance, which provides a more accurate representation of distance compared to Manhattan and Euclidean distances, particularly for complex warehouse maps.(k is a variable, $k = 1 - \frac{min(dx,dy)}{max(dx,dy)}$). The formula for the improved diagonal distance is as follows:

$$h_{improved\ diagonal} = |x(s) - x(g)| + |y(s) - y(g)| + k * \left(\sqrt{2} - 2\right) max\left(|x(s) - x(g)|, |y(s) - y(g)|\right) \tag{6}$$

We propose an enhanced version of the spatiotemporal A* algorithm, where the traditional Manhattan distance heuristic is replaced with an improved diagonal distance formula. This algorithm is referred to as the Improved Diagonal Spatiotemporal A* (IDSA*) algorithm. This enhanced spatiotemporal A* algorithm serves as the foundational pathfinding component within the Conflict-Based Search (CBS) framework, which we refer to as the Improved Diagonal CBS (IDCBS) algorithm.

Dynamic weighted space-time A algorithm*

In practical applications, the CBS algorithm's upper layer identifies conflicts that are suitable for rapid path re-planning. Consequently, we propose a bottom-level path search algorithm designed for efficient path planning. During the initial phase of path search, the algorithm must rapidly navigate to various positions. However, as the search nears completion, it becomes increasingly important to ensure both precise and swift movement to the target point. To address this, we introduce a weight $\delta\ (\delta \geq 1)$ into the heuristic function. This weight is progressively reduced as the search nears the target, thereby diminishing the heuristic function's influence and enhancing the relative importance of the actual path cost. The modified objective function is:

$$h_{improved} = \delta \left[d(p)_{improved} + \frac{cross * d(p)_{improved}}{d(s)} \right] \tag{7}$$

$h_{improved}$ denotes the objective function used by the Spatial-temporal A* algorithm for node expansion; $d(p)_{improved}$ refers to the distance from the current node to the goal node during the search process, measured using an enhanced diagonal distance; the term "cross" signifies the crossover influence factor, which guarantees the generation of a unique and deterministic path when multiple equally valid paths are present; $d(s)$ represents the diagonal distance between the start point and the goal point.

The specific steps to enhance the objective function are as follows:

Employ the improved diagonal distance, as described earlier, to quantify the distance between the current point and the target point. This distance is denoted by:

$$diag(node) = dx + dy + \left(\sqrt{2} - 2\right) * max(dx, dy) * k \tag{8}$$

where k is a variable value, specifically defined as $k = 1 - \frac{min(dx,dy)}{max(dx,dy)}$

Incorporate an additional term into the objective function. Specifically, let $\overrightarrow{a}$ represent the vector from the initial point to the target point, and $\overrightarrow{b}$ represent the vector from the current point to the target point. The dot product of these two vectors is used as an additional term for the objective function h, with the dot product expressed as:

$$Cross\ Product = \vec{a} * \vec{b} \tag{9}$$

To modify the heuristic function, a weight factor $\delta\ (\delta \geq 1)$ is introduced. In this context, $d(s)$ refers to the Manhattan distance from the start point to the goal, while $d(p)_{improved}$ represents the improved diagonal distance from the current point to the goal. As the algorithm approaches the goal, the value of $d(s)$ remains unchanged, whereas $d(p)_{improved}$ gradually decreases. This means that as the algorithm nears the target, the weight factor δ is reduced, thereby decreasing the influence of the objective function and increasing the relative significance of the actual path cost. The weight factor δ is defined as follows:

$$\delta = 1 + \frac{d(p)_{improved}}{d(s)} \tag{10}$$

In the heuristic of the IDSA* algorithm, we incorporate a dynamic weight factor, leading to the Dynamic Weighted Spatiotemporal A* (DWSTA*) algorithm. When this algorithm is applied as the underlying path search strategy within the CBS framework, it is referred to as the Dynamic Weighted CBS (DWCBS) algorithm.

Improvements in the conflict resolution mode of the CBS algorithm

In the CBS algorithm, the upper layer detects two types of conflicts-edge conflicts and vertex conflicts-during the execution of best-first search and conflict resolution, as illustrated in Fig. 9. The lower layer treats these conflicts as constraints, imposes them on the conflicting robots, and re-plans their paths to select the one with the least cost. However, in real-world warehouse environments, conflict types are typically more complex. Firstly, robots are classified based on their operational status as either load-bearing or non-load-bearing. Secondly, the combination of different operational statuses and various types of conflicts further complicates the conflict scenarios. Using a single path planning algorithm to address all conflict types results in inefficiency for the entire path planning process.

To address the aforementioned issues, this paper introduces a new classification scheme for robots in path planning, distinguishing warehouse robots into load-carrying and empty-carrying types. In addition, unlike the traditional CBS algorithm, which identifies only two types of upper-level conflicts, this approach refines the classification of vertex conflicts into four distinct categories, as illustrated in Fig. 10. In the Fig. S1 denotes the starting points of specific segments of robot a_i path, while G1 represents the endpoints of these segments. Similarly, S2 denotes the starting points of specific segments of robot a_j path, and G2 represents their respective endpoints.

1)Intersection vertex conflict

$$\begin{cases} (a_i, a_j, V, t) \\ V_{(a_i,t+1)} \neq V_{(a_j,t-1)} \\ V_{(a_i,t-1)} \neq V_{(a_j,t+1)} \end{cases} \tag{11}$$

Specifically, if Robot a_i and Robot a_j both occupy vertex V at time t, there are two possible scenarios: either the position of Robot a_i at time $t+1$ differs from the position of Robot a_j at time $t-1$, or the position of Robot a_i at time $t-1$ differs from the position of Robot a_j at time $t+1$.

2)Pursuit vertex conflict

$$\begin{cases} (a_i, a_j, V, t) \\ V^{E}_{(a_i,t)} = V^{L}_{(a_j,t)} \end{cases} \tag{12}$$

At time t,both the empty robot a_i and the load-carrying robot a_j occupied the same vertex. In this context, E denotes the empty robot and L denotes the load-carrying robot.Specifically, the speed of the unladen robot a_i is typically 1.5 times faster than that of the laden robot a_j. Therefore, when both robots occupy the same corridor at a given time, the unladen robot a_i will occupy the same vertex V as the laden robot a_j at time t. In this case, the high-level path planning algorithm will prioritize planning the path of the unladen robot a_i to resolve the potential conflict.

3)Map vertex intersection conflict

$$\begin{cases} (a_i, a_j, V, t) \\ V_{(a_i,t+1)} = V_{(a_j,t-1)} \\ V_{(a_i,t-1)} = V_{(a_{j,t+1})} \end{cases} \tag{13}$$

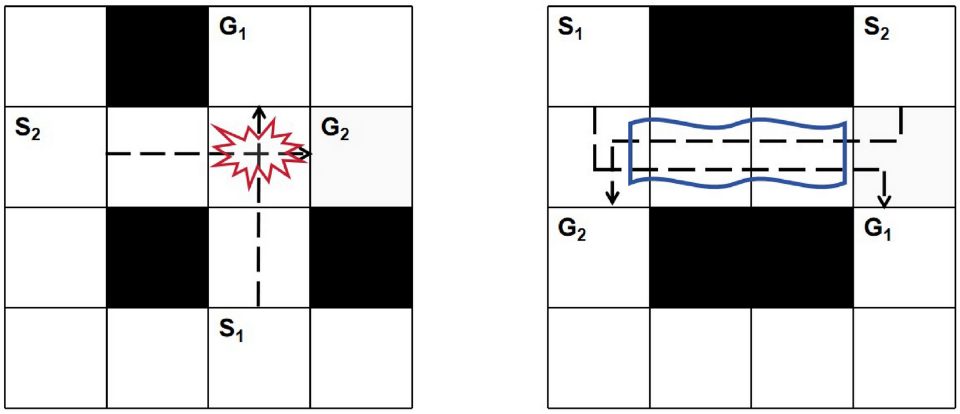


Fig. 9. Types of conflicts.

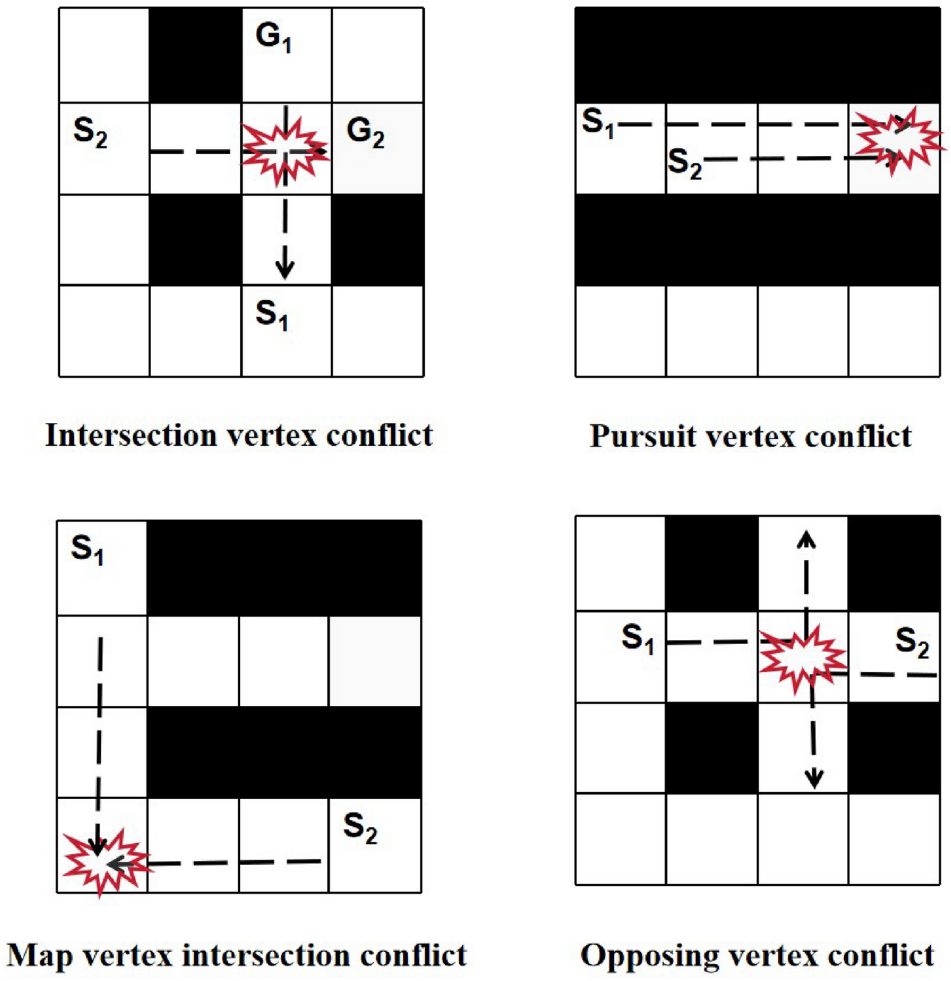


Fig. 10. Four distinct subtypes of vertex conflicts in the CBS-CIOCA algorithm.

At time *t*, both robots a_i and a_j are located at vertex *V*. Additionally, the position of robot a_i at time $(t+1)$ coincides with the position of robot a_j at time $(t-1)$, and conversely, the position of robot a_i at time $(t-1)$ matches the position of robot a_j at time $(t+1)$. Since this situation represents a cross conflict at a map vertex, there are precisely two viable paths around this conflict. Consequently, this type of map vertex cross conflict is defined as a special case of edge conflict.

4)Opposing vertex conflict

$$\begin{cases} (a_i, a_j, V, t) \\ V_{(a_i,t+1)} = V_{(a_j,t-1)} \\ V_{(a_i,t-1)} = V_{(a_j,t+1)} \end{cases} \tag{14}$$

At time *t*, both robots a_i and a_j are located at vertex *V*. Additionally, the position of robot a_i at time $(t+1)$ coincides with the position of robot a_j at time $(t-1)$, and conversely, the position of robot a_i at time $(t-1)$ matches the position of robot a_j at time $(t+1)$.

The types of edge conflicts are classified into two categories: edge conflict and pursuit edge conflict, as illustrated in Fig. 11.

1)Edge conflict

$$(a_i, a_j, X, t) \tag{15}$$

At time t, Robot a_i and Robot a_j both shared edge X.

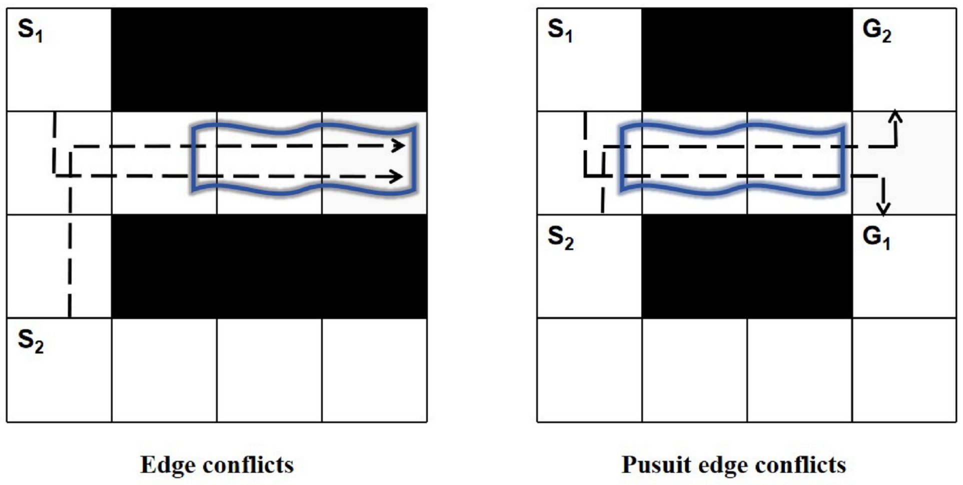


Fig. 11. Edge conflicts and pursuit edge conflicts.

2)Pursuit edge conflict

$$\begin{cases} (a_i, a_j, X, t) \\ X^E_{(a_i,t)} = X^L_{\left(a_j,t\right)} \end{cases} \tag{16}$$

The unloaded robot a_i and the loaded robot a_j jointly occupy the edge X at moment t. E denotes that the robot is in the empty state, and L denotes that the robot is in the loaded state. Specifically, the speed of the unladen robot a_i is typically 1.5 times faster than that of the laden robot a_j. Therefore, when both robots occupy the same corridor at a given time, the unladen robot a_i will occupy the same edge X as the laden robot a_j during the time interval t. In this case, the high-level path planning algorithm will prioritize planning the path of the unladen robot a_i to resolve the potential conflict.

In summary, based on the nature of the robot tasks, we classify conflicts into two primary categories, as shown in Fig. 12: Avoidable Replanning Path Conflicts (ARP) and Unavoidable Replanning Path Conflicts (URP). ARP refers to conflicts that can be resolved without modifying the original path, simply by introducing delays before the conflict occurs. Conversely, URP refers to conflicts that cannot be avoided by delay strategies and require replanning.
Based on different conflict attributes, we select appropriate path search algorithms. To address ARP conflicts in the MAPF problem, we employ an enhanced spatiotemporal A* algorithm that incorporates diagonal optimization. This enhancement facilitates more effective path replanning, producing paths closer to the optimal solution. For handling UPR conflicts, we utilize a dynamic adaptive spatiotemporal A* algorithm that introduces an adaptive weight factor during node expansion. This approach allows for the acceptance of suboptimal paths when resolving edge conflicts, significantly reducing both path search time and the number of node expansions. The algorithm flowchart is shown in Fig. 13. After modifying the CBS (Conflict-Based Search) algorithm, the proposed method is referred to as the “Conflict-Based Strategy with Comprehensive Optimal Conflict Avoidance (CBS-CIOCA) Algorithm.” Algorithm 1 presents the pseudocode of the CBS-CIOCA algorithm.

Algorithm 1.CBS-CIOCA algorithm	
1	input: Plan();
2	Initialize();
3	paths = cell (robot’s paths);
4	for i = 1:numel(robots);
5	paths{i} = space-time AStar(map, robots(i).start, robots(i).goal);
6	while true
7	If no conflict
8	Break;
9	end
10	conflictType = detectConflictType;
11	switch conflictType
12	Case vertex_conflict type
13	If the conflict direction attributes are the same
14	conflictType = ’vertex_cross’;

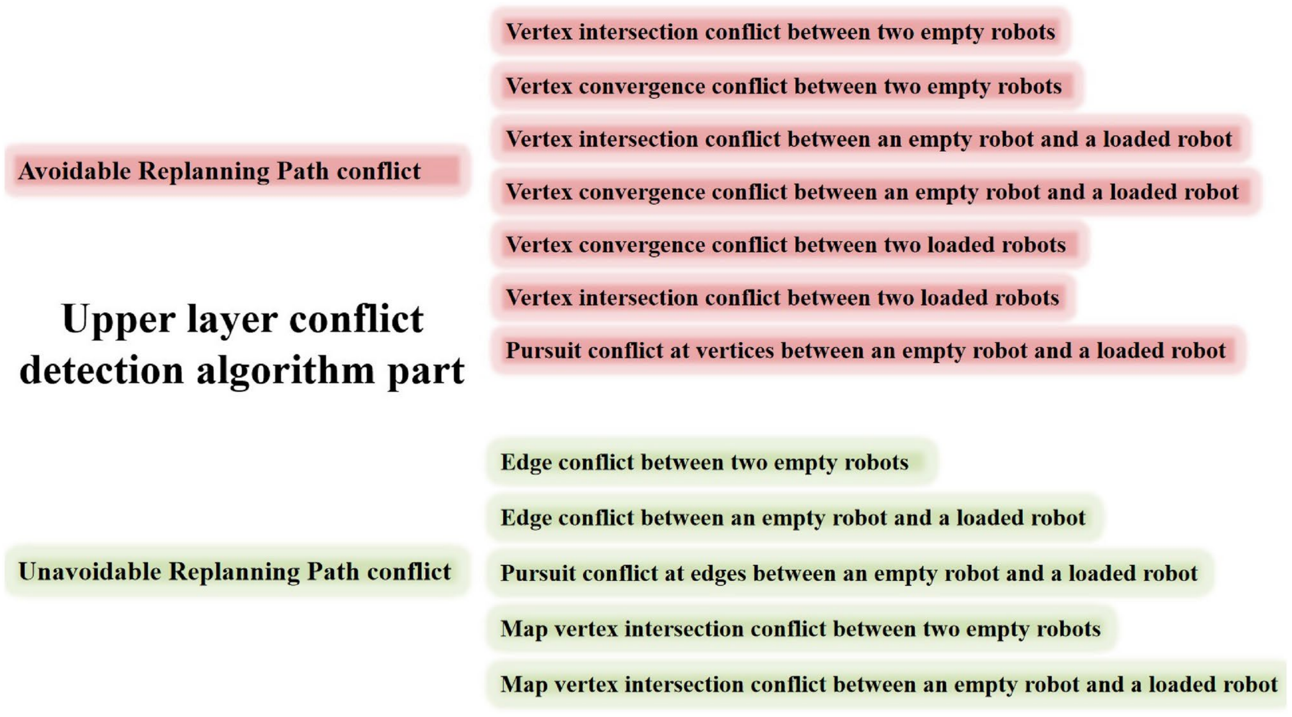

Upper layer conflict detection algorithm part
Avoidable Replanning Path conflict
Vertex intersection conflict between two empty robots
Vertex convergence conflict between two empty robots
Vertex intersection conflict between an empty robot and a loaded robot
Vertex convergence conflict between an empty robot and a loaded robot
Vertex convergence conflict between two loaded robots
Vertex intersection conflict between two loaded robots
Pursuit conflict at vertices between an empty robot and a loaded robot
Unavoidable Replanning Path conflict
Edge conflict between two empty robots
Edge conflict between an empty robot and a loaded robot
Pursuit conflict at edges between an empty robot and a loaded robot
Map vertex intersection conflict between two empty robots
Map vertex intersection conflict between an empty robot and a loaded robot

Fig. 12. Multiple conflict attribute types for ARP and URP in the CBS-CIOCA algorithm.

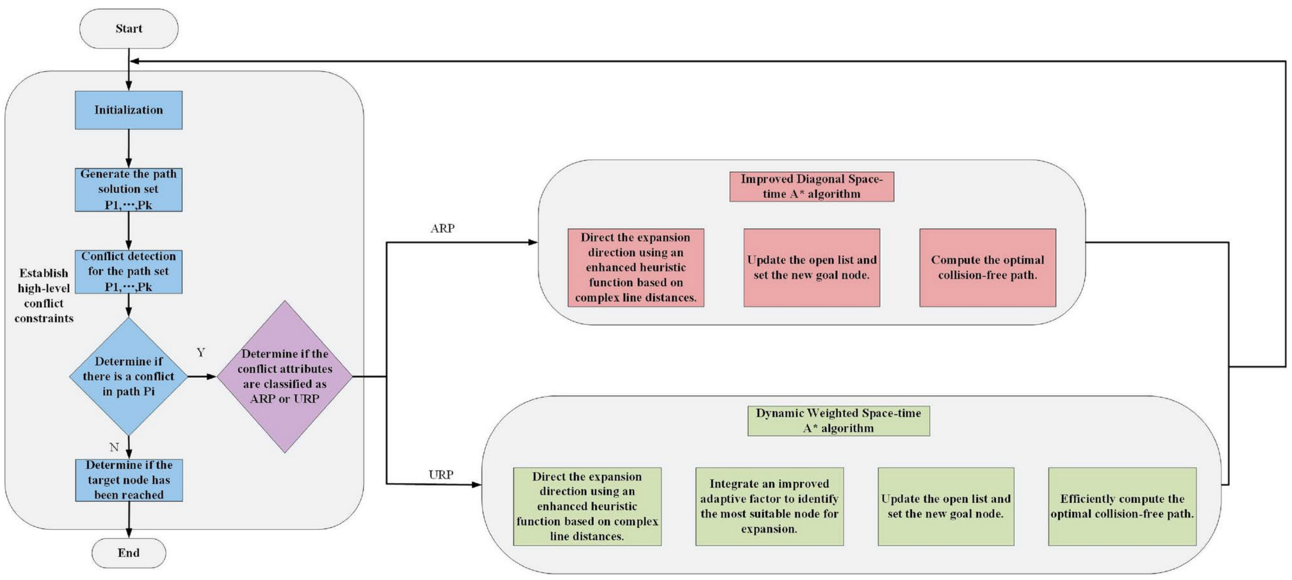

Start
Initialization
Generate the path solution set P1,⋯,Pk
Conflict detection for the path set P1,⋯,Pk
Establish high-level conflict constraints
Determine if there is a conflict in path Pi
Y
N
Determine if the conflict attributes are classified as ARP or URP
Determine if the target node has been reached
End
ARP
URP
Improved Diagonal Space-time A* algorithm
Direct the expansion direction using an enhanced heuristic function based on complex line distances.
Update the open list and set the new goal node.
Compute the optimal collision-free path.
Dynamic Weighted Space-time A* algorithm
Direct the expansion direction using an enhanced heuristic function based on complex line distances.
Integrate an improved adaptive factor to identify the most suitable node for expansion.
Update the open list and set the new goal node.
Efficiently compute the optimal collision-free path.

Fig. 13. The algorithm flowchart of CBS-CIOCA is illustrated below.

Algorithm 1.CBS-CIOCA algorithm	
15	elseif
16	conflictType = ’vertex_opposing’;
17	else
18	conflictType = ’map vertex_cross’;
19	end if
20	Case edge_conflict type
21	conflictType = ’edge_conflict’;
22	if conflictType = ’vertex_cross’or ‘vertex_opposing’
23	conflictAttributes = ’ARP’;
24	paths = replanPaths_IDSTA*(map, paths, ’ARP’);
25	elseif
26	conflictAttributes = ’URP’;
27	paths = replanPaths_DWSTA*(map, paths, ’ARP’);
28	disp(paths);
29	end if
30	end switch
31	end while

Simulation results

This study aims to assess the performance of the CBS-CIOCA algorithm in multi-AGV path planning in complex maps and large warehouse environments through ablation experiments. First, we evaluate the computational time and the number of nodes expanded during the path search of the CBS and CBS-CIOCA algorithms in single-path planning scenarios using 20x20 and 30x30 complex obstacle maps. Subsequently, multi-robot path planning experiments were conducted using the 35x18 fishbone-shaped map and the 37x18 traditional warehouse map. These experiments were compared against the traditional CBS algorithm, the IDCBS algorithm, and the DWCBS algorithm as reference methods to evaluate the comprehensive performance of the CBS-CIOCA algorithm in terms of planning time and the number of expanded nodes.Finally, to test the robustness of the CBS-CIOCA algorithm, large-scale maps of 30x60 and 30x30, containing 10% to 30% randomly distributed obstacles and randomly chosen starting points, were used to evaluate the algorithm’s stability under unknown conditions.

Considering practical production constraints, the time for solving each experiment was limited to 5 minutes. If the system failed to generate a valid path solution within this time frame, the experiment was considered inconclusive. The experiments were conducted on a 2.10 GHz AMD Ryzen 5 3550H processor, and programming was performed using MATLAB 2021b.

Complex obstacle map environment experiment

We assessed the path planning capabilities of two algorithms for single robots using two complex obstacle maps, specifically sized 20×20 and 30×30. Additionally, we evaluated these algorithms’ performance in multi-robot path planning using a complex obstacle map of size 60×30. The results of the single robot path planning tests are presented in Figs. 14 and 15, and detailed in Table 2.

In Figs. 14 and 15, the blue paths represent the CBS algorithm, while the red paths represent the CBS-CIOCA algorithm. In the 20×20 complex obstacle map environment with a single robot, both algorithms produce paths of equal length from start to finish. However, the CBS-CIOCA algorithm demonstrates a 94.83% improvement in execution time over the CBS algorithm and reduces the number of expanded nodes needed for path planning by 90.01%, as detailed in Table 2. In the 30×30 complex obstacle map environment, while the CBS-CIOCA algorithm shows a slight increase in path cost compared to the CBS algorithm, it achieves a 95.69% reduction in pathfinding time and an 86.52% reduction in the number of expanded nodes.

Figures 16 and 17 illustrate the path solution sets produced by two different algorithms with 15 robots. As shown in Table 3, the improved algorithm reduced the running time to 197.71 seconds compared to 2576.4 seconds for the previous version, representing a 92.33% decrease in planning time. According to the criteria established at the start of the experiment, the path planning was considered unsuccessful if the planning time exceeded certain limits. The results demonstrate that the enhanced algorithm significantly improves time efficiency in path planning for both single and multiple robots, thus greatly reducing the likelihood of path planning failures.

Results of multi-Robot testing in two warehouse layouts

To evaluate the performance of the CBS-CIOCA algorithm versus the original algorithm in multi-robot warehouse environments, we conducted experiments using two warehouse layouts: a traditional layout of 37×18 and a fishbone-type layout of 35×18.

For objective and reliable results, each algorithm was tested 10 times with different numbers of robots, and the average results were recorded. For experiments involving only five robots, the starting point was fixed at the gray square, designated as the storage and retrieval point. When the number of robots exceeded five, simulations

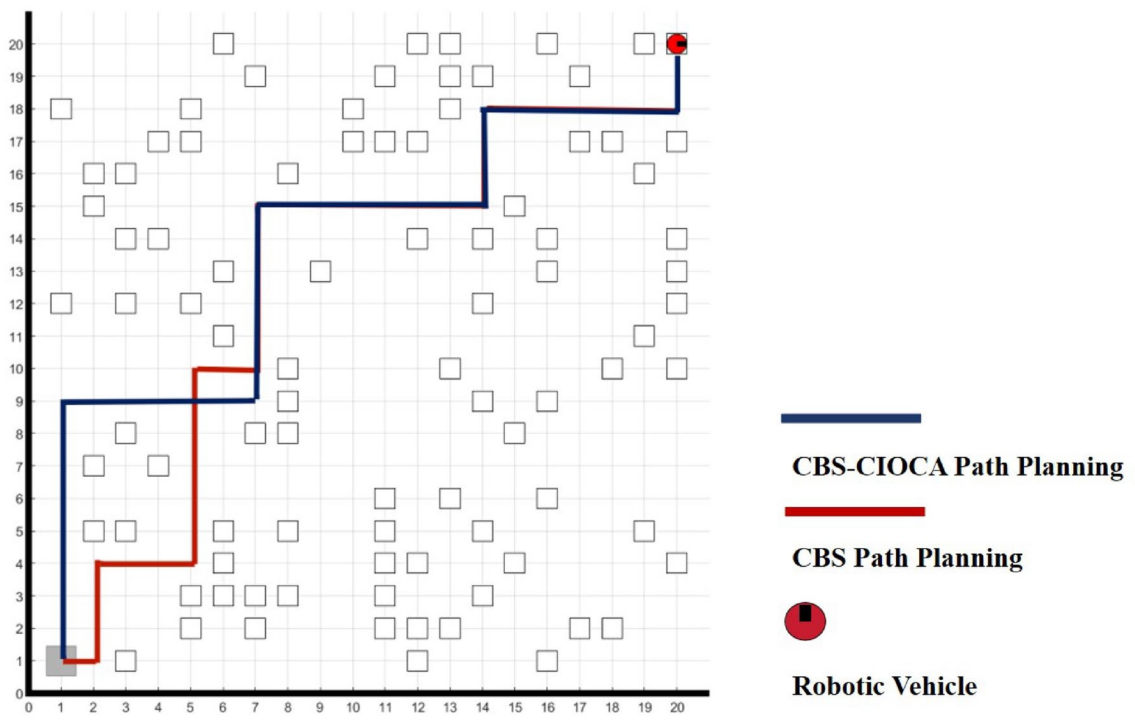


Fig. 14. Paths generated by two algorithms on a 20 × 20 complex obstacle map.

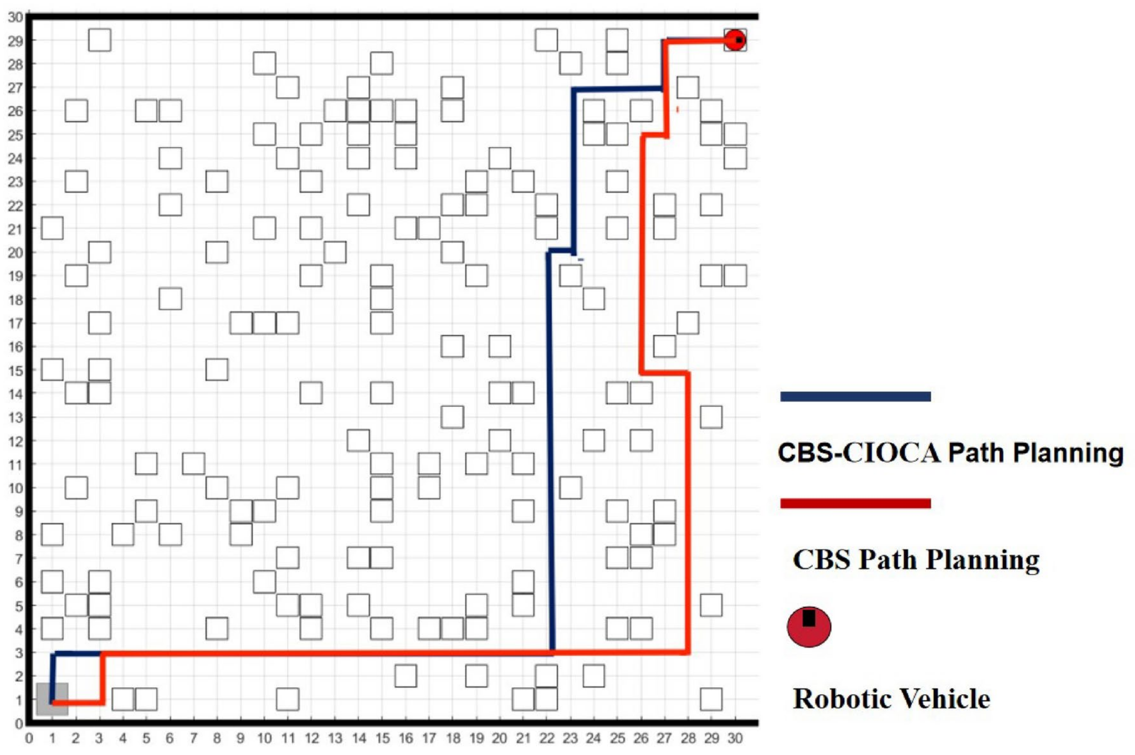


Fig. 15. Paths generated by two algorithms on a 30 × 30 complex obstacle map.

involved randomly assigning delivery tasks to other robots in the warehouse, while the starting positions of robots numbered 1-15 remained fixed as detailed in Table 4.

Figures 18, 19, 20 and 21 show the path maps of 15 robots using the traditional CBS algorithm and the CBS-CIOCA algorithm on traditional warehouse layouts and fishbone-shaped warehouse layouts.

Map size	Path Search Algorithm	Time (seconds)	Node Expansion	Cost Steps
20 × 20 complex obstacle map	CBS	3.5617	3864	38
20 × 20 complex obstacle map	CBS-CIOA	0.1841	386	38
30 × 30 map of complex obstacles	CBS	10.86	7181	57
30 × 30 map of complex obstacles	CBS-CIOA	0.4678	968	61

Table 2. Comparative analysis of CBS and CBS-CIOCA algorithms in two different complex obstacle map scenarios.

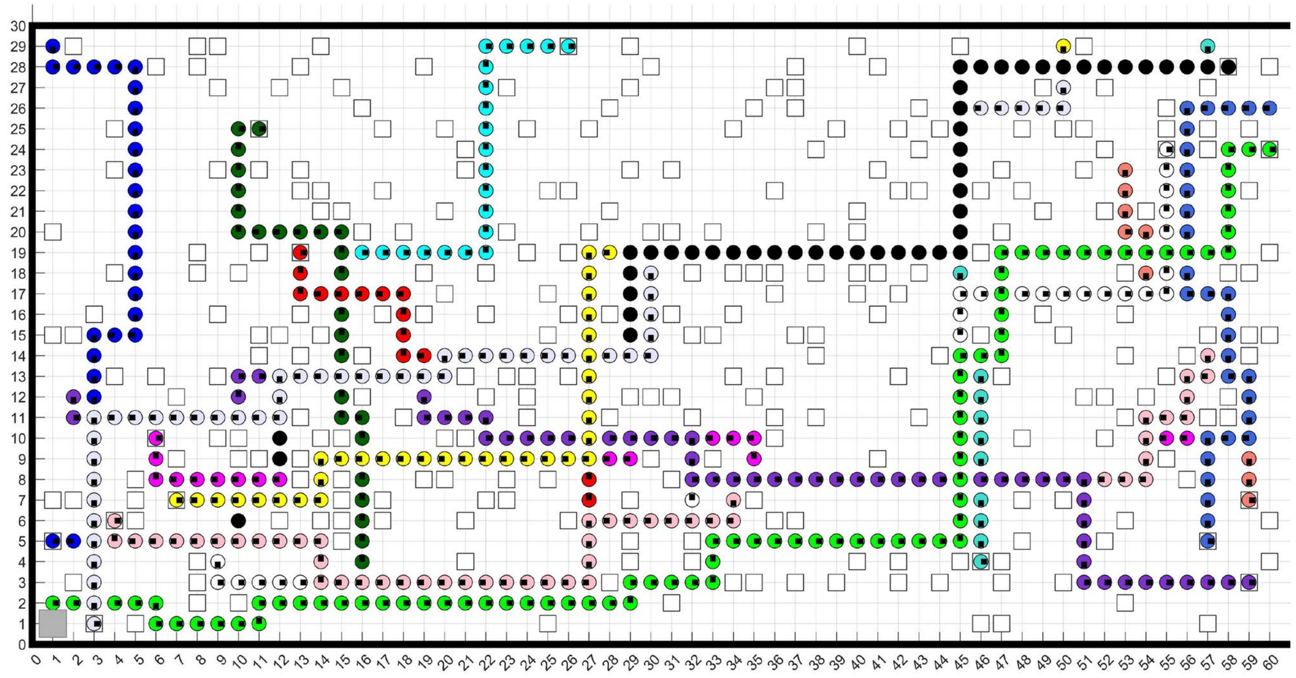


Fig. 16. Path generated by the CBS algorithm in a 60 × 30 complex obstacle map.

According to the data from Fig. 22 for the traditional warehouse layout, it can be observed that the CBS-CIOCA algorithm significantly outperforms the traditional CBS algorithm in terms of average runtime and node expansion volume. When the number of robots is small, the average runtime and node expansion volume of the DWCBS algorithm are similar to those of the CBS-CIOCA algorithm. However, as the number of robots increases, the average runtime and node expansion volume of the DWCBS algorithm are less than those of the CBS-CIOCA algorithm. This is because as the number of robots increases, the number of conflicts also increases, leading to more frequent calls to the diagonally improved Space-time A* algorithm in CBS-CIOCA. Consequently, the CBS-CIOCA algorithm tends to select more optimal paths, which slightly reduces its performance when the number of robots is large.

As shown in Fig. 23, when the number of robots is greater than or equal to 9, the CBS-CIOCA algorithm expands more nodes than the DWCBS algorithm. This occurs because, in the process of planning random start and end points for multiple robots, the underlying pathfinding algorithm, which utilizes IDSA*, is invoked more frequently as the number of robots increases, leading to a higher number of nodes being expanded by the CBS-CIOCA algorithm compared to the DWCBS algorithm. However, compared to other algorithms, the CBS-CIOCA algorithm shows superior performance in terms of both average runtime and node expansion in fishbone-type warehouse layouts. Specifically, the CBS-CIOCA algorithm is able to minimize the number of expanded nodes while ensuring optimal route planning, thus achieving the highest efficiency in finding the optimal path for each robot.

Based on the results presented in Figs. 22 and 23, it can be observed that the CBS-CIOCA algorithm outperforms the CBS algorithm in both execution time and the number of expanded nodes within the traditional warehouse layout compared to the fishbone-type warehouse layout. The fishbone-type layout, although having a smaller overall spatial extent, contains more obstacles, making pathfinding more complex. While the initial positions of the robots in these two warehouse layouts are not standardized and vary, the experimental data shown in Figs. 22 and 23 clearly indicate that the CBS-CIOCA algorithm consistently performs better than the CBS algorithm in both types of layouts.

Based on the traditional warehouse layout map data presented in Fig. 24, increasing the number of robots results in notable reductions in path planning time and node expansion for the CBS-CIOCA and DWCBS

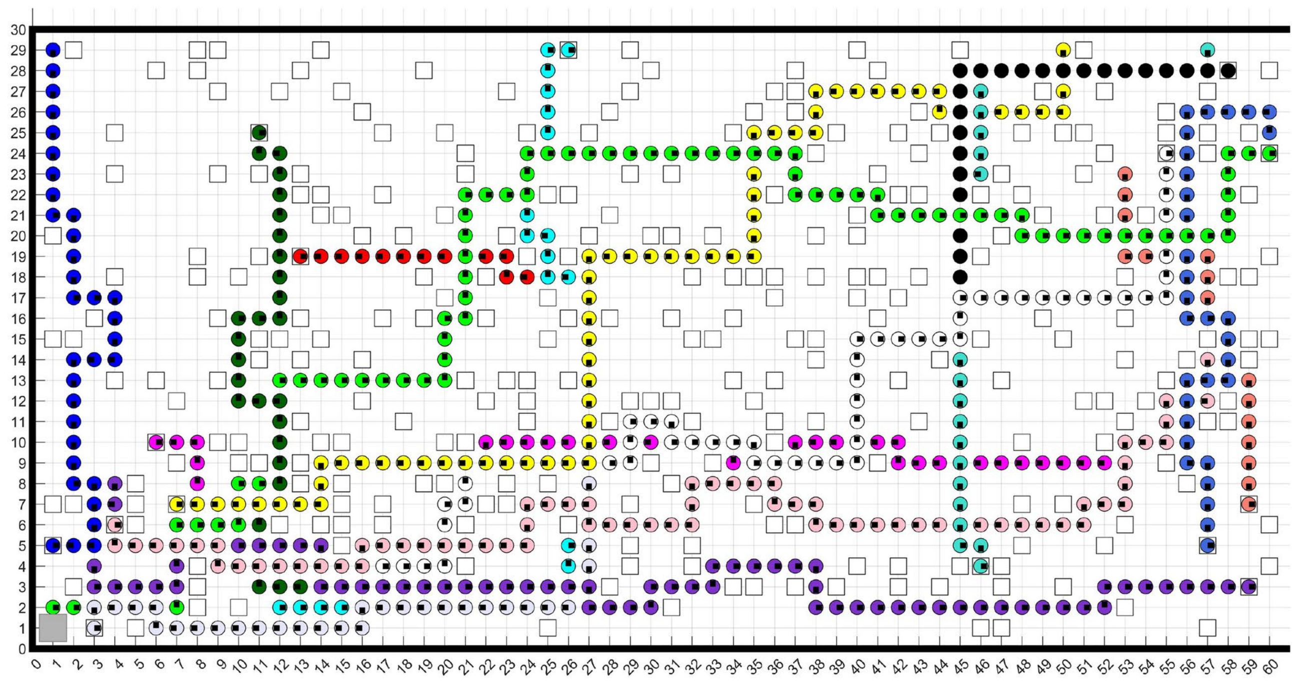


Fig. 17. Path generated by the CBS-CIOCA algorithm in a 60 × 30 complex maze map.

Map size	Path Search Algorithm	Time(seconds)
60 × 30 complex obstacle map	CBS	2576.40
	CBS-CIOCA	197.71

Table 3. Planning time comparison between CBS and CBS-CIOCA algorithms in a 60 × 30 complex maze map.

Fishbone Warehouse Layout			Traditional Warehouse Layout		
AGV Number	AGV Starting Position	AGV Ending Position	AGV Number	AGV Starting Position	AGV Ending Position
1	(11,11)	(11,3)	1	(8,2)	(18,3)
2	(8,9)	(7,6)	2	(4,13)	(18,14)
3	(15,9)	(10,4)	3	(16,10)	(35,15)
4	(3,4)	(5,3)	4	(14,2)	(30,8)
5	(10,1)	(16,3)	5	(28,17)	(17,14)
6	(10,11)	(7,12)	6	(31,10)	(12,4)
7	(15,8)	(24,11)	7	(7,4)	(33,16)
8	(9,9)	(21,16)	8	(11,12)	(29,3)
9	(6,6)	(30,6)	9	(25,12)	(5,13)
10	(8,10)	(13,10)	10	(30,6)	(6,11)
11	(4,1)	(16,8)	11	(4,1)	(36,4)
12	(11,1)	(24,17)	12	(11,1)	(33,11)
13	(18,1)	(14,9)	13	(18,1)	(26,3)
14	(25,1)	(6,5)	14	(25,1)	(12,3)
15	(32,1)	(34,18)	15	(31,1)	(8,16)

Table 4. Coordinates of the AGV starting position.

algorithms compared to the original algorithm. Furthermore, the IDCBS algorithm also shows significant enhancements in these metrics. Specifically, the IDCBS algorithm achieves up to a 78.83% reduction in path planning time and up to a 76.92% decrease in node expansion. The DWCBS algorithm demonstrates up to a 97.85% reduction in path planning time and up to a 95.39% decrease in node expansion. Meanwhile, the CBS-

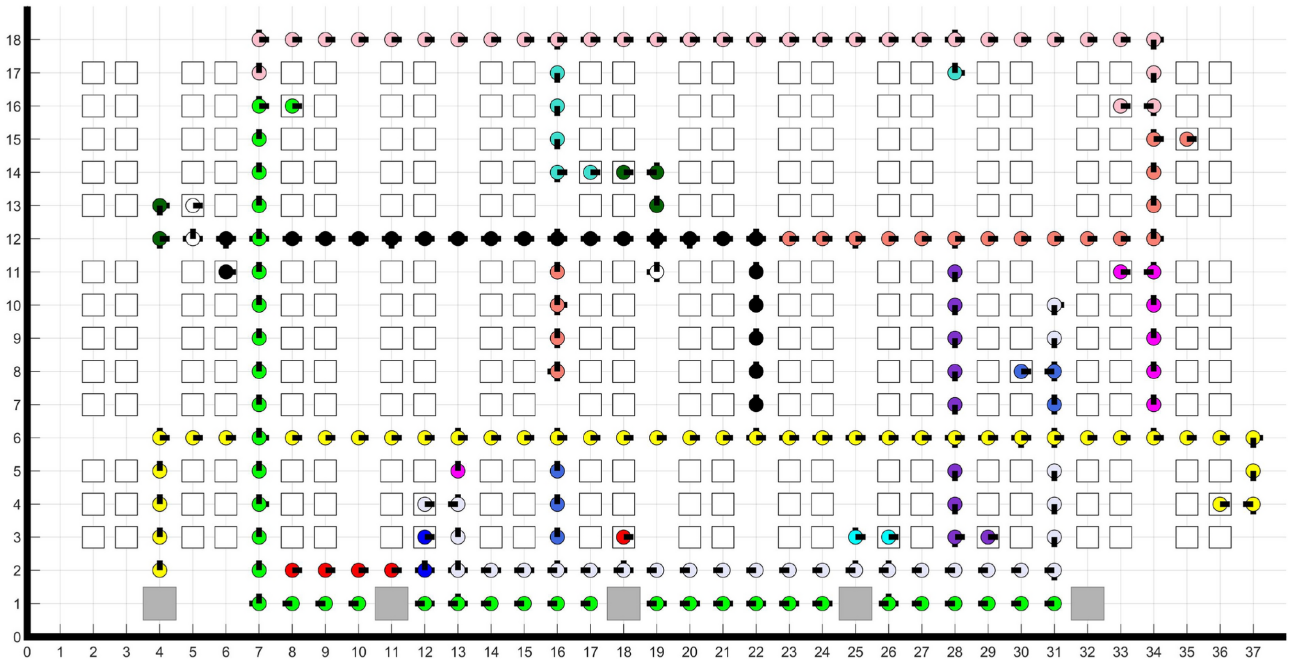


Fig. 18. Paths of 15 robots in Traditional Warehouse layout map based on CBS algorithm.

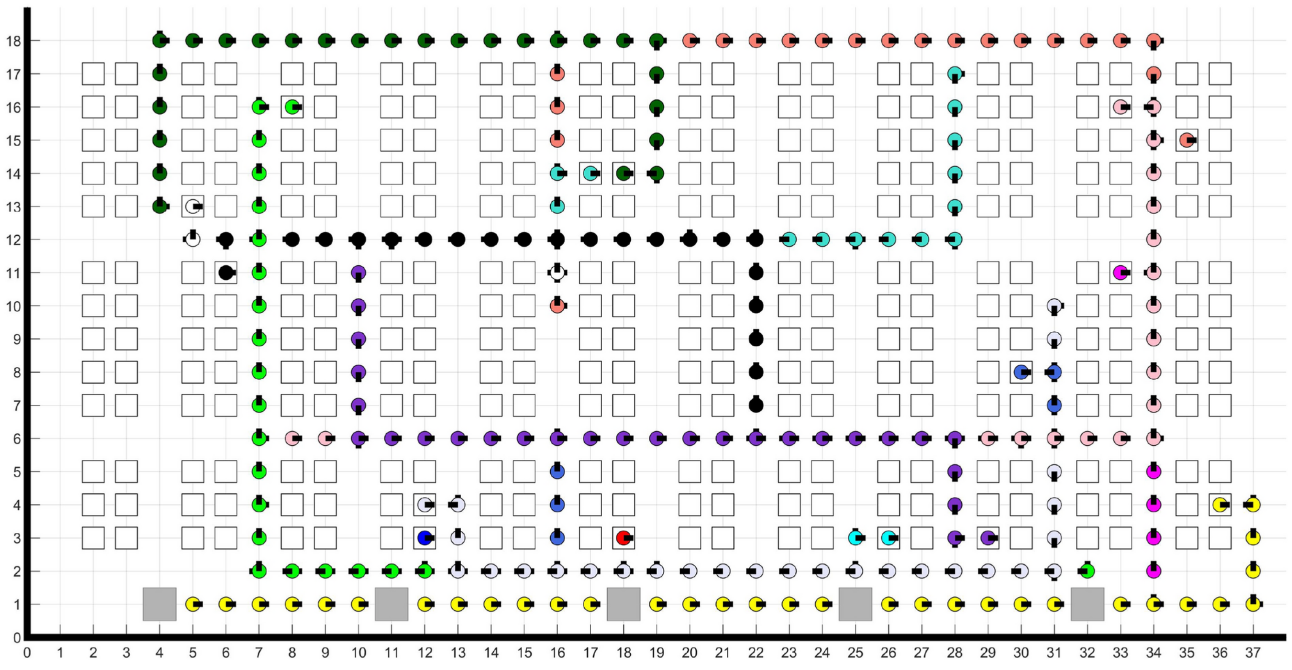


Fig. 19. Paths of 15 robots in Traditional Warehouse layout map based on CBS-CIOCA algorithm.

CIOCA algorithm realizes up to a 97.37% reduction in path planning time and up to a 94.95% decrease in node expansion.

According to the data presented in Fig. 25, within the fishbone-shaped warehouse layout, as the number of robots increases, the CBS-CIOCA algorithm demonstrates superior performance compared to the other two improved algorithms. This superiority is evident in both the time improvement rate and the reduction in node expansions. Specifically, the IDCBS algorithm achieves a maximum time improvement of 86.85% and a maximum node expansion reduction of 80.23%. The DWCBS algorithm achieves a maximum time improvement of 88.22% and a maximum node expansion reduction of 85.53%. In contrast, the CBS-CIOCA algorithm achieves the highest maximum time improvement of 88.37% and the greatest reduction in node expansions at 88.41%.

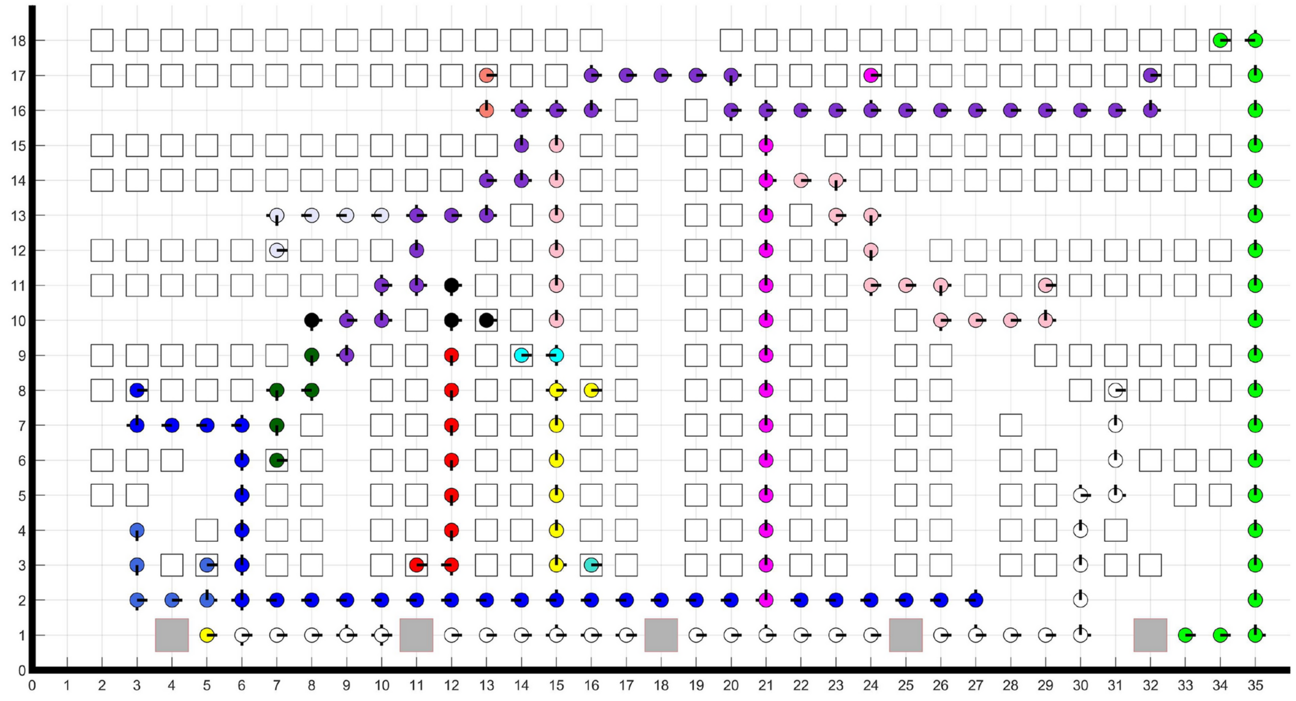


Fig. 20. Paths of 15 robots in Fishbone Warehouse layout map based on CBS algorithm.

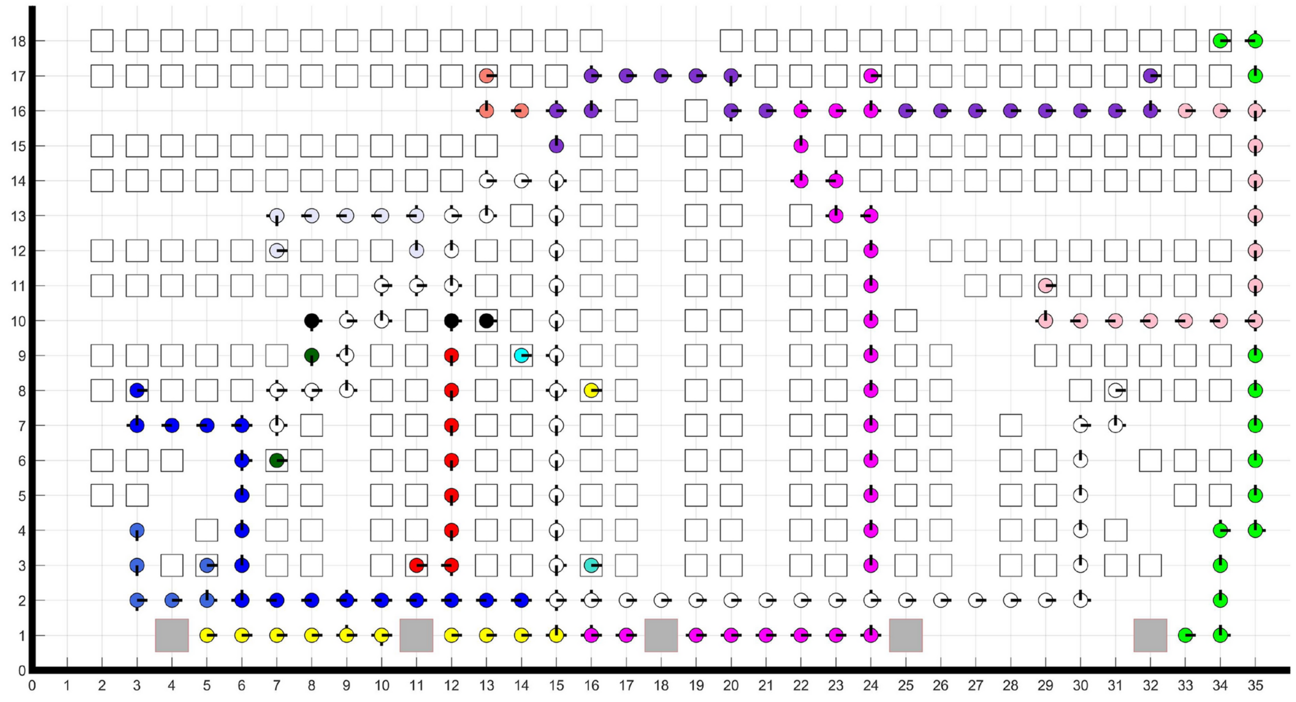
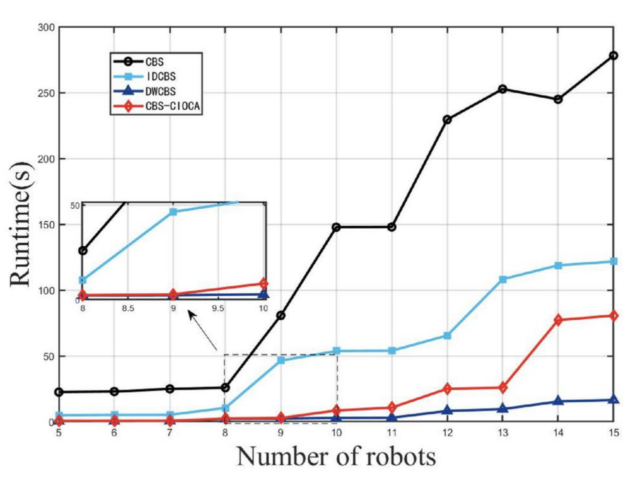
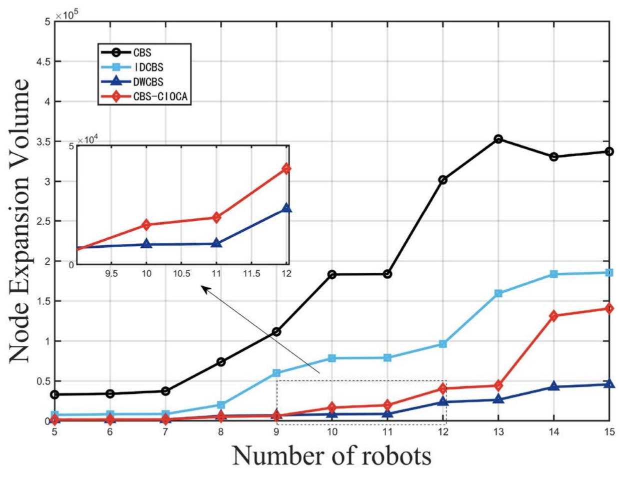


Fig. 21. Paths of 15 robots in Fishbone Warehouse layout map based on CBS-CIOCA algorithm.

Figure 26 presents the average success rate of 100 trials, with 5 robots starting from randomly assigned positions, on 30x30 and 30x60 unknown obstacle maps. The CBS algorithm, IDCBS algorithm, DWCBS algorithm, and CBS-CIOCA algorithm were tested. It is evident that the DWCBS and CBS-CIOCA algorithms significantly outperform the other two in terms of pathfinding success rate. In contrast, the IDCBS algorithm falls behind the other two improved algorithms because it prioritizes maintaining path optimality, which limits its performance. The CBS algorithm shows the lowest average success rate. These results demonstrate that the CBS-CIOCA algorithm has stronger robustness compared to the original CBS algorithm.

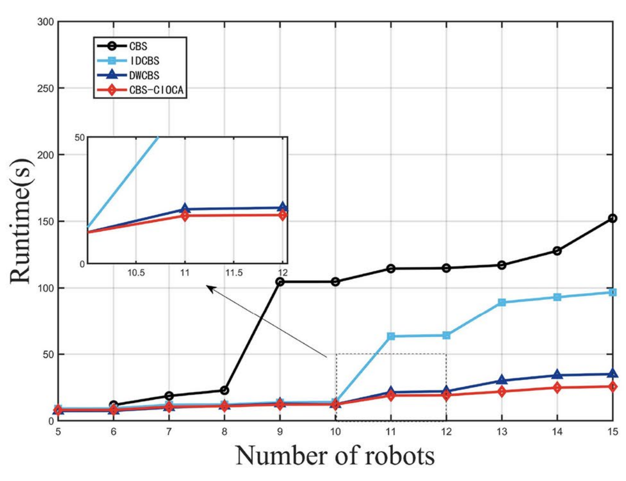


(a) Comparison of average runtime

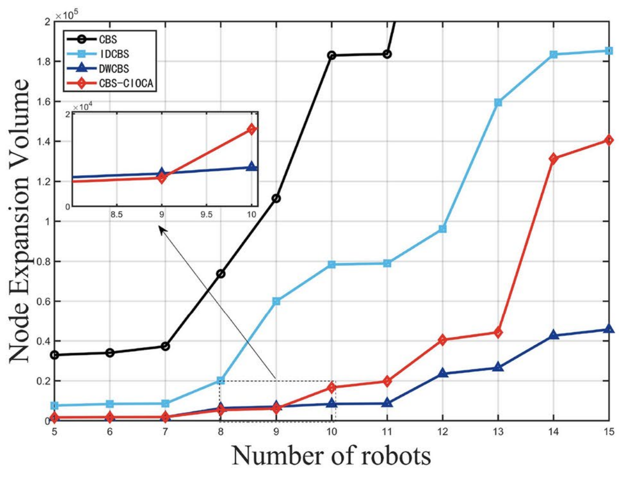


(b) Comparison of node expansion counts

Fig. 22. Comparative analysis of the performance of Traditional Warehouse layout algorithms.



(a) Comparison of average runtime



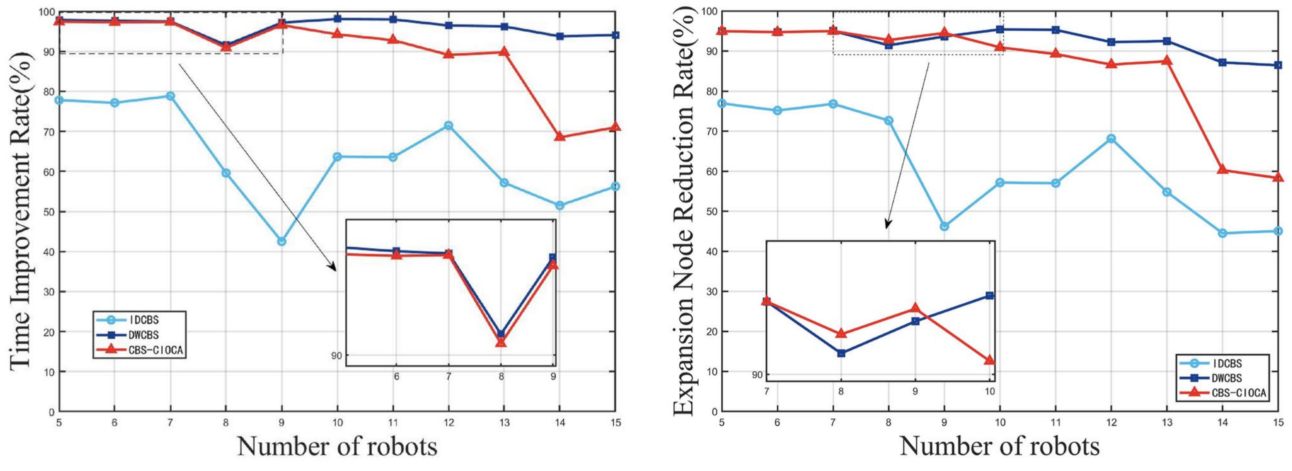
(b) Comparison of node expansion counts

Fig. 23. Comparative analysis of the performance of Fishbone Warehouse layout algorithms.

Conclusion

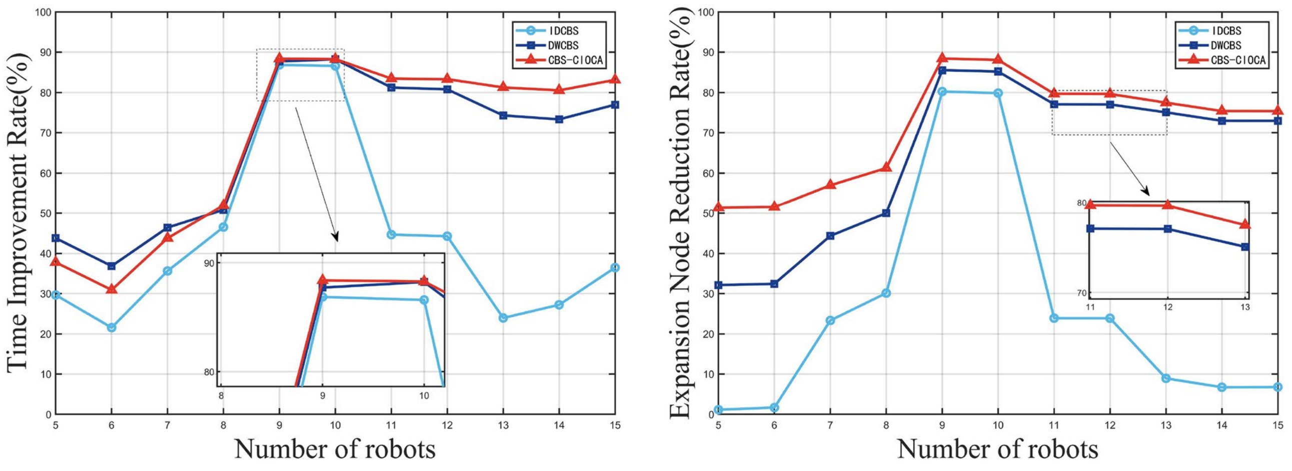
To enhance the time efficiency and success rate of multi-robot path planning, we propose an advanced algorithm: the Conflict-based Comprehensive Optimization Conflict Avoidance Algorithm (CBS-CIOCA). The algorithm first improves the space-time A* algorithm based on diagonals for the pathfinding phase of the CBS algorithm, enabling the determination of optimal paths with fewer node expansions and reduced computation time. Building on this, we integrate a dynamic adaptive factor, which leads to the development of a dynamic adaptive space-time A* algorithm. While this algorithm may slightly sacrifice optimality, it substantially improves path search efficiency and reduces the number of node expansions. In the conflict resolution phase, various underlying pathfinding algorithms are applied according to the specific types of conflicts encountered. This approach not only improves the success rate of path planning but also significantly enhances overall algorithmic efficiency. Experimental results show that, across a range of map types and structures, the CBS-CIOCA algorithm significantly outperforms the original CBS-STA algorithm in both path planning time and node expansion metrics, demonstrating superior performance. By incorporating the conflict classification method outlined in this study and combining it with the two underlying search algorithms, future multi-robot path planning can benefit from further optimization.

This study focuses on multi-robot path planning and has achieved significant advancements in this field. Future research will concentrate on path planning in dynamic environments with multiple moving obstacles. It will also address challenges arising from the algorithm's dependency on conflict classification systems, which could lead to suboptimal paths in cases of inaccurate classification. Additionally, future work will aim to overcome these issues while exploring the integration of machine learning or real-time deployment within robotic systems.



(a) Comparison of time improvement rates (b) Comparison of node reduction rates

Fig. 24. Improvement of three algorithms in Traditional Warehouse layout environments.



(a) Comparison of time improvement rates (b) Comparison of node reduction rates

Fig. 25. Improvement of three algorithms in Fishbone Warehouse layout environments.

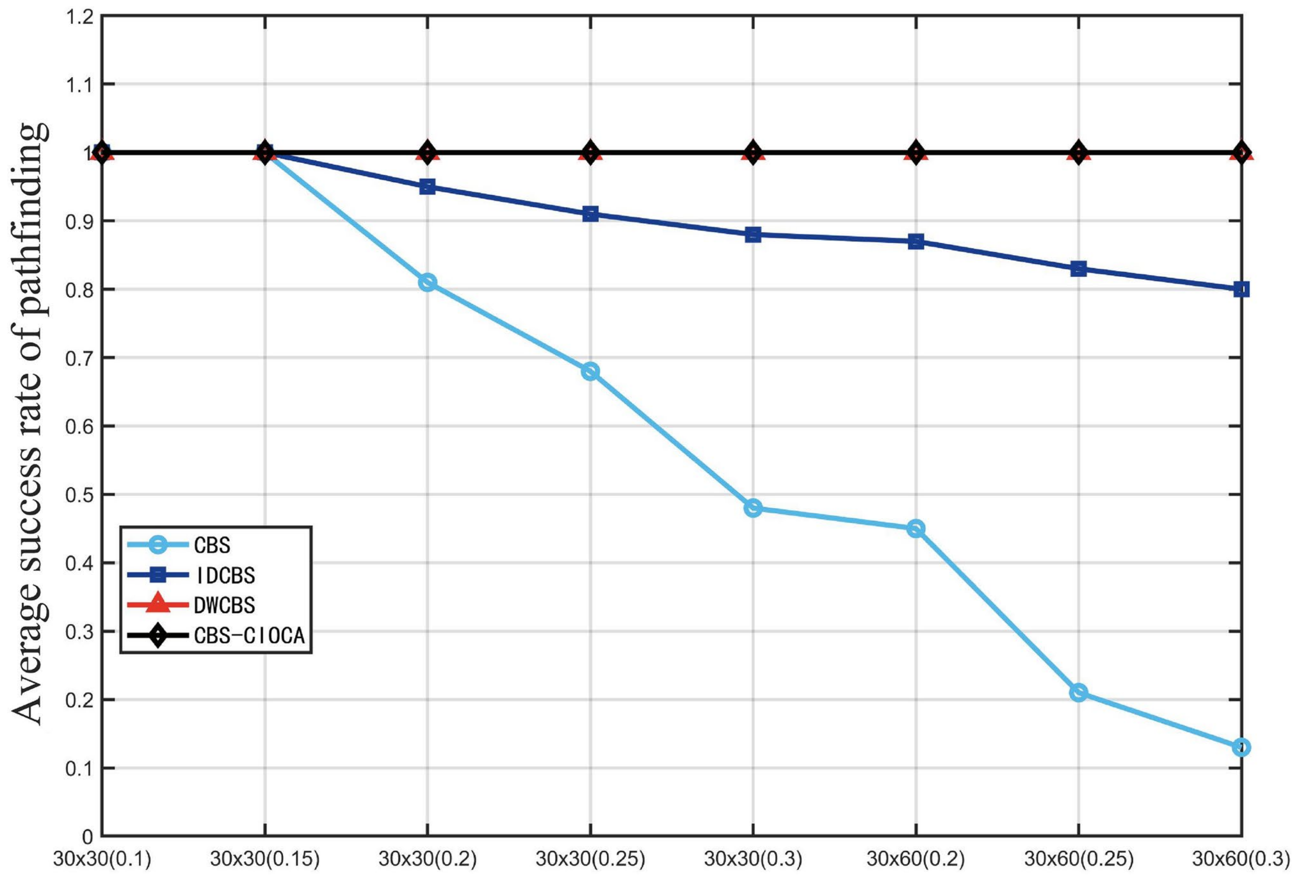


Fig. 26. Comparison of the Performance of Four Algorithms in Unknown Environments.

Data Availability

The datasets used and analyzed during the current study are available from the corresponding author on reasonable request.

Received: 31 October 2024; Accepted: 6 February 2025
Published online: 15 February 2025

References

1. Yang, L. et al. Multi-area collision-free path planning and efficient task scheduling optimization for autonomous agricultural robots. *Sci. Rep.* **14**, 18347. https://doi.org/10.1038/s41598-024-69265-y (2024).
2. Li, Z., Hu, J., Leng, B., Xiong, L. & Fu, Z. An integrated of decision making and motion planning framework for enhanced oscillation-free capability. *IEEE Trans. Intell. Transp. Syst.* **25**, 5718–5732. https://doi.org/10.1109/TITS.2023.3332655 (2024).
3. Shirini, K. et al. A comprehensive survey on multiple-runway aircraft landing optimization problem. *Int. J. Aeronaut. Space Sci.* **25**, 1574–1602. https://doi.org/10.1007/s42405-024-00747-z (2024).
4. Shirini, K. et al. Modified imperialist competitive algorithm for aircraft landing scheduling problem. *J. Supercomput.* **80**, 13782–13812. https://doi.org/10.1007/s11227-024-05999-w (2024).
5. Shirini, K. et al. Multi-objective aircraft landing problem: A multi-population solution based on non-dominated sorting genetic algorithm-ii. *J. Supercomput.* **80**, 25283–25314. https://doi.org/10.1007/s11227-024-06385-2 (2024).
6. Nonut, A. et al. A small fixed-wing uav system identification using metaheuristics. *Cogent Eng.* **9**, 2114196. https://doi.org/10.1080/23311916.2022.2114196 (2022).
7. Shirini, K., Taherihajivand, A. & Samadi Gharehveran, S. A review of algorithms for solving the project scheduling problem with resource-constrained considering agricultural problems. *Agric. Mech.* **8**, 1–14. https://doi.org/10.22034/jam.2023.55751.1227 (2023).
8. Tejani, G., Savsani, V., Patel, V. & Bureerat, S. Topology, shape, and size optimization of truss structures using modified teaching-learning based optimization. *Adv. Comput. Des.*[SPACE]https://doi.org/10.12989/acd.2017.2.4.313 (2017).
9. Oskouei, A. G., Abdolmaleki, N., Bouyer, A., Arasteh, B. & Shirini, K. Efficient superpixel-based brain mri segmentation using multi-scale morphological gradient reconstruction and quantum clustering. *Biomed. Signal Process. Control* **100**, 107063. https://doi.org/10.1016/j.bspc.2024.107063 (2025).
10. Tejani, G., Savsani, V. & Patel, V. Modified sub-population based heat transfer search algorithm for structural optimization. *Cogent Eng.* **8**, 1–23. https://doi.org/10.4018/IJAMC.2017070101 (2017).
11. Shankar, M. & Sushnigdha, G. A hybrid path planning approach combining artificial potential field and particle swarm optimization for mobile robot. *IFAC-PapersOnLine* **55**, 242–247. https://doi.org/10.1016/j.ifacol.2023.03.041 (2022).
12. Lei, T., Sellers, T., Luo, C., Carruth W, D. & Bi, Z. Graph-based robot optimal path planning with bio-inspired algorithms. *Biomim. Intell. Robot.* **3**, 100119. https://doi.org/10.1016/j.birob.2023.100119 (2023).

13. Pham, H. V. et al. Proposed multi-st model for collaborating multiple robots in dynamic environments. *Machines* **12**, 197. https://doi.org/10.3390/machines12110797 (2024).
14. Xidias, E. & Zacharia, P. Balanced task allocation and motion planning of a multi-robot system under fuzzy time windows. *Eng. Comput.* **41**, 1301–1326. https://doi.org/10.1108/EC-09-2023-0612 (2024).
15. Sharon, G., Stern, R., Felner, A. & Sturtevant, N. R. Conflict-based search for optimal multi-agent pathfinding. *Artif. Intell.* **219**, 40–66. https://doi.org/10.1016/j.artint.2014.11.006 (2015).
16. Jin, J. et al. Conflict-based search with d* lite algorithm for robot path planning in unknown dynamic environments. *Comput. Electr. Eng.* **105**, 108473. https://doi.org/10.1016/j.compeleceng.2022.108473 (2023).
17. Zhou, Z., Abdi, E., Chea, C. P. & Bai, Y. Global path planning for autonomous construction vehicles in building construction: A comparative study with a focus on vehicle kinematic characteristics. *J. Build. Eng.* **93**, 109837. https://doi.org/10.1016/j.jobe.2024.109837 (2024).
18. Lin, S., Liu, A., Wang, J. & Kong, X. An improved fault-tolerant cultural-pso with probability for multi-agv path planning. *Expert Syst. Appl.* **237**, 121510. https://doi.org/10.1016/j.eswa.2023.121510 (2024).

Declarations

Ethics declarations

Competing interests The authors declare no competing interests

Additional information

Correspondence and requests for materials should be addressed to Y.D.