machines

MDPI

Article

MT-SIPP: An Efficient Collision-Free Multi-Chain Robot Path Planning Algorithm

Jinchao Miao [1], Ping Li [1,*], Chuangye Chen [1], Jiya Tian [1] and Liwei Yang [1,2,*]

[1] Faculty of Information Engineering, Xinjiang Institute of Technology, Aksu 843100, China; jinchao_miao@xjit.edu.cn (J.M.); chencyhe@163.com (C.C.); tianjiya@sohu.com (J.T.)
[2] College of Mechanical and Electrical Engineerin, Shihezi University, Shihezi 832003, China
* Correspondence: ping_li@xjit.edu.cn (P.L.); 18916336783ylw@gmail.com (L.Y.)

Abstract: Compared to traditional multi-robot path planning problems, multi-chain robot path planning (MCRPP) is more challenging because it must account for collisions between robot units and between the bodies of a chain and the leading unit during towing. To address MCRPP more efficiently, we propose a novel algorithm called Multi-Train Safe Interval Path Planning (MT-SIPP). Based on safe interval path planning principles, we categorize conflicts in the multi-train planning process into three types: travel conflicts, waiting conflicts, and station conflicts. To handle travel conflicts, we use an improved k-robust method to ensure trains avoid collisions with other trains during movement. To resolve waiting conflicts, we apply a time correction method to ensure the safety of positions occupied by trains during waiting periods. To address station conflicts, we introduce node constraints to prevent other trains from occupying the station positions of trains that have reached their target stations and are stopped. Experimental results on three benchmark maps show that the MT-SIPP algorithm achieves about a 30% improvement in solution success rate and nearly a 50% increase in the maximum number of solvable instances compared to existing methods. These results confirm the effectiveness of MT-SIPP in addressing the challenges of MCRPP.

Keywords: multi-chain robot; path planning; travelling conflicts; waiting conflicts; station conflicts

check for updates

Citation: Miao, J.; Li, P.; Chen, C.; Tian, J.; Yang, L. MT-SIPP: An Efficient Collision-Free Multi-Chain Robot Path Planning Algorithm. *Machines* **2024**, *12*, 482. https://doi.org/10.3390/machines12070482

Academic Editor: Antonios Gasteratos

Received: 25 June 2024
Revised: 9 July 2024
Accepted: 16 July 2024
Published: 17 July 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Intelligent robots are increasingly being applied in smart environments and are capable of autonomously or semi-autonomously performing diverse tasks through instruction reception and execution. Particularly in scenarios necessitating that multiple robots cooperatively reach different target locations for task execution, such as in logistics [1,2], agriculture [3], and factory handling [4,5], the technology of Multi-Agent Path Finding (MAPF) [6] becomes pivotal. This technology aims to devise optimal and collision-free paths from starting points to target points for each robot while ensuring safety during the movement process.

In pursuit of this objective, researchers have extensively explored the field of MAPF [7–9], introducing a variety of planning algorithms [10–12] tailored to diverse path planning needs. Among these, Sharon et al. [13] proposed the Conflict-Based Search (CBS) algorithm, which has made significant strides in achieving optimal and comprehensive solutions for MAPF. CBS utilizes a conflict constraint tree to expand nodes, detecting conflicts and imposing constraints based on optimal single-agent path planning. It subsequently replans conflicting robots within existing constraints until conflict-free paths are established. In order to further enhance the efficiency of the CBS algorithm, Boyarski et al. [14] improved strategies for conflict priority and meta-agents, which effectively reduced the number of conflicting nodes. Additionally, Ariel et al. [15] optimized the node expansion process within the CBS algorithm by introducing the Add Heuristics to Conflict-Based Search (CBSH) algorithm. By integrating heuristics as reference metrics for node selection during high-level node

expansions, CBSH effectively reduces the number of expanded nodes, thereby accelerating the solving process. Building on the CBSH algorithm, Li et al. [16] addressed the issue of long processing times for handling rectangle conflicts. By introducing different methods to infer and add rectangle conflict constraints, Li effectively reduced invalid or redundant node expansions, thereby further improving the solving efficiency of the CBSH algorithm.

Taking into account potential delays encountered by robots during actual movement, Dor et al. [17] introduced the k-Robust Conflict-Based Search (kR-CBS) algorithm. This algorithm allocates a safety margin of time steps for each path node during planning, ensuring robots can safely execute tasks even when delays in time steps occur. This proactive approach effectively mitigates potential collision risks. To tackle the challenge of exploring large path node spaces in coupled algorithms for multi-agent planning, Phillips et al. [18] introduced the Safe Interval Path Planning (SIPP) algorithm. This method prioritizes sequential planning of multiple robots based on their respective priorities. As robots complete their planning, they dynamically update safe intervals on the map grid, guiding subsequent robots in selecting path nodes. This decoupled and priority-driven approach ensures conflict-free paths once all robots have finalized their plans, significantly enhancing overall solving efficiency. However, Yakovlev et al. [19] identified a limitation in the SIPP algorithm during node expansion, where each grid node permits passage for only one robot at a time, potentially resulting in scenarios where solutions are not achievable. Therefore, he introduced a Weighted Safe Interval Path Planning algorithm that maintains optimal and suboptimal nodes during path node expansion. If the optimal node path cannot be expanded further, the algorithm leverages suboptimal nodes to improve solution success rate and adaptability.

The MAPF algorithms mentioned above demonstrate adaptability for traditional robots that occupy a single grid node. However, they are not applicable to path planning problems for chain robots [20], such as trains and serpent robots, which occupy multiple grid nodes. Chain robots, compared to traditional robot transport methods [21,22], offer greater flexibility by integrating multiple transport units to carry large quantities of goods. Additionally, their transport mechanism is highly efficient and energy-saving, requiring only the head unit as a power source to pull an entire train-like assembly of carriages, aligning with principles of green sustainable development. Currently, chain robots show broad application prospects across various fields, including efficient baggage handling at airports, flexible product transport in factory production lines, and convenient flower transport in greenhouse environments. For the path planning problem of chain robots, Lin et al. [23] proposed an innovative multi-train scheduling algorithm based on simulated annealing, capable of delivering high-quality solutions within a limited timeframe. Svancara et al. [24] introduced a novel constraint-based scheduling model for the MAPF problem, emphasizing optional activities. He et al. [25] developed a dynamic train trajectory planning method that integrates model predictive control and multi-swarm particle swarm optimization, effectively reducing energy consumption. Zhang et al. [26] presented a multi-objective integer nonlinear programming model aimed at minimizing passenger travel time and optimizing average train load utilization. Arsalani et al. [27] addressed the train formation and transportation path optimization problem with a mixed-integer linear programming model, providing satisfactory solutions within a reasonable time frame. Dor et al. [28] defined the Multi-Train Path Finding (MTPF) problem and introduced the Multi-Train Conflict-Based Search (MT-CBS) algorithm. Building on the MT-CBS algorithm, Chen et al. [29] proposed enhanced versions including MT-ICBS, MT-CBSH, and MT-CBSH-RM, offering more comprehensive solutions for the path planning of chain robots. However, CBS-based algorithms inherently rely on node search and splitting mechanisms, which present limitations for large-scale MTPF problems. In contrast, the Safe Interval Path Planning (SIPP) algorithm utilizes safe intervals as the foundational planning element, allowing for the simultaneous detection and adjustment of multiple consecutive time steps, thereby ensuring higher search efficiency. While the MT-CBS [28] algorithm is a recognized approach to solving

MTPF problems, there is a notable gap in research on leveraging the principles of the SIPP algorithm to address these issues.

Our work aims to bridge a significant gap by substantially reducing the pathfinding complexity for chain robots, thus enhancing the success rate in solving large-scale multi-train pathfinding problems. By leveraging the principles of Safe Interval Search Algorithms, we developed a novel planning strategy called the Multi-Train Safe Interval Path Planning (MT-SIPP) algorithm, designed to efficiently address the path planning challenges that come with multiple chain robots. We introduce a new path planning framework for multi-chain robots based on the concept of safe intervals. Within this framework, conflicts are classified into train movement conflicts, waiting conflicts, and station conflicts. To resolve these conflicts, we employ robust methods for train movement conflicts, adjust the occupancy time steps for waiting conflicts, and manage station conflicts through grid checks and the addition of node constraints. To validate our approach, we conducted extensive simulations on benchmark maps of varying scales. Additionally, we visualized the planning solutions for specific instances to demonstrate the effectiveness of the proposed method.

The structure of this paper is organized as follows: Section 2 provides an introduction to the MAPF and MTPF problems. Section 3 elaborates on the concept of safe intervals. Section 4 presents the k-robust planning method. Section 5 offers a detailed description of our proposed method for resolving multi-type conflicts. Section 6 outlines the algorithmic workflow. Section 7 discusses the simulations conducted on three types of benchmark maps. Section 8 concludes this study.

2. Problem Description

2.1. Multi-Agent Path Finding

MAPF is a technique for planning paths for multiple robots in an undirected graph $G(V, E)$, aiming to find a path from the start point to the end point for each robot within the obstacle-free region. Here, V represents the grid node and E represents the edges connecting the grid. During the planning process, the robot i typically occupies only one grid V_i at time step T_i, and the complete path P_i is defined as $\{(V_i, T_i)|T_i = 0, 1, 2, \cdots, cost_i\}$, where V_i denotes the grid position of robot i at T_i, and $cost_i$ represents the total time cost required for robot i to reach the destination. To ensure the effectiveness of planned paths, the returned path nodes must adhere to the following constraints:

$$\forall\ V_i \notin L_{obstacles} \tag{1}$$

$$P_i \cap P_j = \varnothing, \text{ where } i \neq j \text{ and } i, j = 1, 2, \cdots, n \tag{2}$$

$$\left(V_i = V_{j-1}\right) \cap \left(V_j = V_{i-1}\right) = \varnothing, \text{ where } V_i \in P_i, V_j \in P_j \text{ and } i \neq j \tag{3}$$

where, $L_{obstacles}$ represents the list storing all obstacle grids; V_i and V_{i-1} are the grids the robot passes through in consecutive time steps, and similarly for v_j and v_{j-1}; n is the number of robots. Equation (1) guarantees that all robot movements are strictly constrained within the non-obstacle region, ensuring the selected path nodes' navigability and safety. Equation (2) effectively prohibits two robots from occupying the same grid node simultaneously, thereby preventing potential vertex conflicts. Equation (3) avoids situations where two robots traverse the same edge of a grid node simultaneously at the same time step (i.e., $V_i = V_{j-1}$ and $V_j = V_{i-1}$), thereby eliminating the possibility of edge conflicts. Furthermore, based on potential conflict scenarios among multiple robots, these can be categorized into vertex conflicts and edge conflicts. Vertex conflicts occur when two robots occupy the same grid node simultaneously, as depicted in Figure 1a. On the other hand, edge conflicts arise when two robots attempt to traverse the same edge of a grid node simultaneously, as shown in Figure 1b.

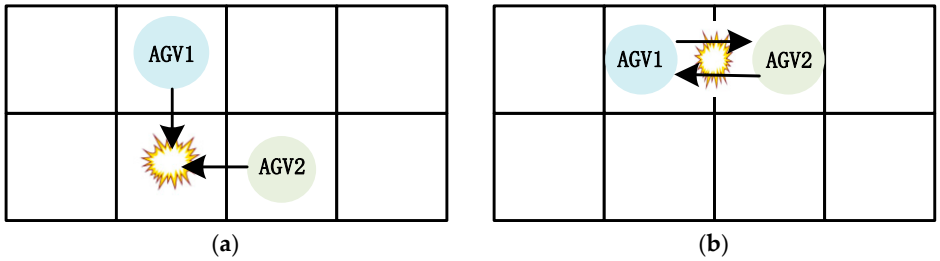


Figure 1. Diagram of conflicts. (**a**) point conflict; (**b**) edge conflict.

2.2. Multi-Train Path Finding

MTPF differs significantly from MAPF in its approach to handling chain-like robots. For robots with a body length of k, as they depart from the starting point, the number of grids they occupy gradually expands from 1 to $k+1$. Specifically, if a robot's head occupies grid V_i at time step T_i, its body also occupies nodes such as $(V_{i-1},\ T_{i-1})$, $(V_{i-2},\ T_{i-2})$, $(V_{i-3},\ T_{i-3})$, $\cdots$, $(V_{i-k},\ T_{i-k})$, where V_{i-k} are the grids previously traversed by the robot's head, and these nodes constitute the "car_body". The path P_i of the multi-train robot i is represented as $\{((V_i,\ T_i),\ car_body) \mid T_i = 0,\ 1,\ 2,\ \cdots,\ cost_i\}$. The core objective of MTPF algorithms is to identify a set of path sets, ensuring all robot paths are collision-free and minimizing the total cost sum of all paths. This necessitates that the algorithm consider not only the optimal path for each individual robot during planning but also to coordinate path selections among multiple robots to achieve global optimality.

MTPF Objectives:

$$\min SOC = \min\{\textstyle\sum_{i=0}^{n} cost_i\} \tag{4}$$

MTPF Constraints:

$$\forall\ V_i \in R \tag{5}$$

$$P_i \cap P_j = \varnothing,\ \text{ where } i \neq j \text{ and } \ i, j = 1, 2, \cdots, n \tag{6}$$

where, n denotes the number of chain robots, R denotes the list consisting of non-obstacle grids and SOC refers to the aggregate cost in time for all chain robots to reach their target points individually. Equation (5) ensures unrestricted movement of all chain robots within non-obstacle regions. Equation (6) specifically addresses the unique characteristics of chain robots, ensuring that no part of the robot, including the head and multiple body segments, simultaneously occupies the same grid cell as any part of another chain robot during movement. This precaution is necessary to prevent point conflicts, as depicted in Figure 2.

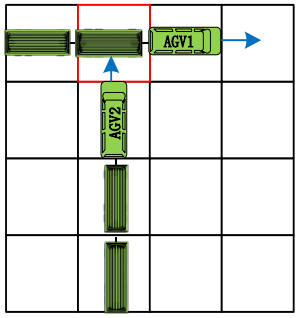


Figure 2. Point conflict of multi-train robot.

Notably, since chain robots occupy multiple contiguous grids during movement, the MTPF problem avoids edge conflict scenarios that could arise in MAPF, such as when two chain robots might swap positions, as depicted in Figure 2.

3. SIPP Algorithm

The Safe Interval Path Planning (SIPP) algorithm innovatively extends the classical A* algorithm [30] by incorporating the concept of safe intervals. In this algorithm, each grid cell is pre-defined with a global safe interval list. For example, obstacle grid cells have an empty safe interval list, while unvisited grid cells initially possess an initial safe interval list $[(0, \infty)]$, indicating safe accessibility at any time. During MAPF, each robot evaluates whether to add expanded grid cells to the OPEN list based on their respective safe interval lists. This mechanism enables precise node selection, significantly reducing unnecessary node expansions. After successful path planning for multiple robots, they backtrack from their respective target nodes to generate individual paths. Subsequently, the SIPP algorithm updates the global safe interval lists of involved grid cells in these paths to ensure subsequent planning robots can avoid conflicts and select nodes based on updated safe interval information.

Consider one non-obstacle grid with a safe interval list $[(0, \infty)]$ in Figure 3 as an example. Following the completion of multi-robot planning, assume Robot 1 passes through grid cell A at time step 3, and Robot 2 passes through the same grid cell at time step 5. Based on this information, the safe interval list for this grid cell will be updated to $[(0, 2), (4), (6, \infty)]$, effectively representing its availability across different time periods. This mechanism illustrates how the SIPP algorithm offers a clear and efficient conflict avoidance strategy for subsequent robot planning.

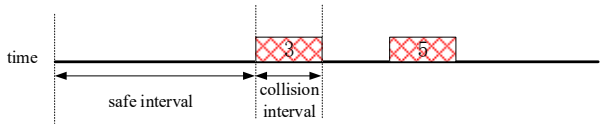


Figure 3. Schematic of grid security interval update.

4. *K*-Robust Planning Methods

As referenced in [17], to mitigate potential delays encountered by robots during path execution, an effective strategy involves preemptively accounting for delay factors during the planning stage and assigning suitable safety buffers for each robot. This measure significantly enhances the robustness of multi-robot path planning, enabling it to withstand delays to a certain extent. In practical implementation, since most multi-robot path planning algorithms expand nodes based on integer time steps, *K*-robust planning algorithms typically simplify computations by setting k as an integer value. For instance, in Figure 4, if Robot 1 is allocated a safety buffer of 1 time step during the planning phase (i.e., $k = 1$)), and Robot 1 passes through a grid at time step 3 while Robot 2 passes through the same grid at time step 5, the safe interval for grid A would accordingly update to $[(0, 2), (7, \infty)]$. This updating strategy ensures that robots can traverse grids safely and sequentially according to their planned paths, even in the presence of potential delays.

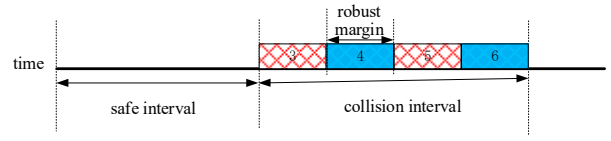


Figure 4. Grid safety interval update under k robust planning ($k = 1$).

Upon comparing Figures 3 and 4, we can clearly discern the practical application effects of the *K*-robust planning algorithm. This algorithm effectively extends the time duration that a robot occupies a grid from a single time step to $K + 1$ time steps. Therefore, if Robot 1 or Robot 2 encounters a delay of up to k time steps upon leaving grid A, they may potentially remain on grid A during these k time steps. To mitigate this scenario, the *K*-robust planning algorithm assigns a safety buffer of k time steps for these robots. This approach treats the subsequent k time steps as collision intervals during planning,

effectively preventing other robots from passing through grid A within $k+1$ time steps. Such a strategy ensures that Robot 1 and Robot 2 can safely execute their path planning even in the presence of delays, thus minimizing potential collisions and conflicts.

5. MT-SIPP Algorithm

Based on the SIPP algorithm [18], this paper extends its application to multi-train path planning. The MT-SIPP algorithm categorizes potential conflicts into three types when planning paths for multiple trains: train movement conflicts, waiting conflicts, and station stopping conflicts. To effectively address these conflicts, the algorithm employs the following strategies:

(1) For movement conflicts, the MT-SIPP algorithm employs an improved K-robust method. This approach ensures that trains maintain sufficient safety distances during their movement, effectively preventing collisions with other trains.
(2) When facing waiting conflicts, the algorithm utilizes time adjustment techniques. By appropriately adjusting the waiting times of trains, this method ensures that the track positions occupied by waiting trains remain unobstructed by others, thereby ensuring the safety of waiting trains.
(3) Addressing station stopping conflicts, the MT-SIPP algorithm implements strategies such as adding node constraints. These strategies ensure that when a train reaches and stops at its destination station, other trains do not occupy or block its stopping position, thereby mitigating station stopping conflicts.

5.1. Train Movement Conflicts and Resolution Methods

5.1.1. Definition of Conflict

Train movement conflicts refer to collisions or overlaps between the bodies of two trains during their traction and movement. This occurs when parts of two or more trains occupy the same grid simultaneously, denoted as $p_T_I \cap p_T_J \neq \varnothing$. In train path planning, the process typically revolves around planning based on the train head and indirectly inferring the body's movement path through the head's trajectory. However, this simplified approach may lead to potential conflicts. To completely avoid such conflicts, it is essential to consider the paths of entire trains, especially when updating safety intervals.

As depicted in Figure 5a, if no additional time step constraints are imposed on the orange grid traversed by train a during its head path planning, there is a risk that train b could enter the same grid immediately after train a's head has departed. This scenario may lead to a collision between train a's body and train b. Similarly, in Figure 5b, without additional time step constraints on the orange grid along train a's head path, when train a's head reaches the blue grid, train b's head might occupy the orange grid. Even if train b's head has already vacated the orange grid before train a arrives, the presence of train b's body in the area could still result in a collision between the two trains. Therefore, to prevent train movement conflicts, it is essential to comprehensively consider the paths of entire trains during the planning process and apply appropriate time step constraints to grids where conflicts may occur.

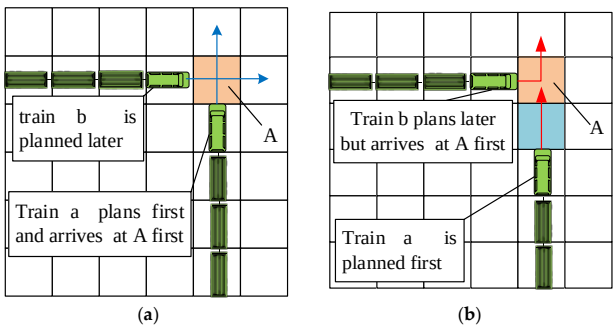


Figure 5. Schematic diagram of train collision. (**a**) Status 1; (**b**) Status 2.

5.1.2. Conflict-Free Strategy

In addressing the train movement conflicts outlined in Section 5.1.1, this paper proposes the adoption of an enhanced K-robust method [17] to effectively mitigate these conflicts. As depicted in Figure 6, during the progression of a train, once its head traverses a grid (designated as grid x), the system automatically designates all time steps from $t-k$ to $t+k$ (specifically, $t-k, t-k+1, \cdots, t-1, t, t+1, \cdots, t+k-1, t+k$) as collision intervals for that grid. This precaution prohibits other trains from passing through the grid during this defined time window, thereby ensuring robust prevention of train movement conflicts.

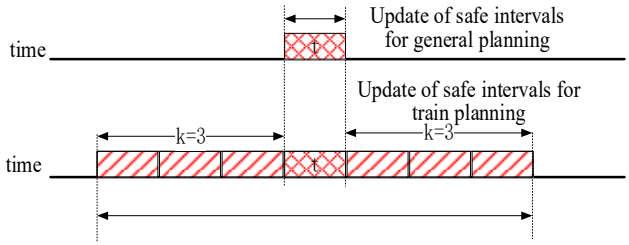


Figure 6. Schematic diagram of the improved robust method for resolving train travel conflicts.

This strategy ensures that during the movement of the current train, its body will be completely cleared from the current occupied grid after k time steps. Alternatively, if another train has passed through and vacated the grid before k time steps, the current train's body can also be fully cleared from the grid occupied by its head within k time steps. This approach effectively prevents collisions between two trains during their movement.

By setting collision interval constraints $(t-k,\ t+k)$ for grids traversed by the current planned train, we ensure that after completing the current train's planning phase, no other train passes within k grids behind or ahead of it. This measure effectively mitigates the risk of the train's body colliding with other trains or being collided with during its movement.

The following is a segment of pseudocode for identifying successor nodes at time step t that satisfy the robust safety criteria for trains within $\pm k$ time steps Algorithm 1:

Algorithm 1: get_k_robust_successors(s)

```
neighbors = get_neighbors(s)
for m in neighbors do
    for interval in m_safe_interval_list do
        if interval_start <= t − k && t + k <= interval_end then
            node = (m, t), successors append node
        end if
    end for
end for
return successors
```

Obtaining nodes that satisfy the $\pm$k robust conditions is essential, yet insufficient to ensure absolute conflict-free operation between trains. To safeguard system integrity, it is imperative to methodically update the safety intervals of grids encompassed by the planned paths following successful planning completion. This practice prevents inadvertent occupation of already allocated safety intervals during node expansion in subsequent planning phases, thus effectively preempting potential conflicts.

5.2. Train Waiting Conflicts and Resolution Methods

5.2.1. Definition of Conflict

Train waiting conflicts occur when a train, during its journey, is forced to halt and wait due to obstruction by another train. Improper handling of such situations can lead to a phenomenon where multiple cars of the halted train occupy the same grid simultaneously. As depicted in Figure 7, when applying an improved k-robust method to update safety

intervals (with k = 3), the train's locomotive occupies grids D, C, B, A at consecutive time steps t = 1, 2, 3, 4, respectively. Under normal conditions, at time step t = 4, while the locomotive occupies grid A, the train's three cars are positioned on grids B, C, D. According to the k-robust method, to prevent conflicts, the system restricts other trains from passing through grids D, C, B, A during time steps 1 to 4, 1 to 5, 1 to 6, and 1 to 7, respectively. However, complications arise when a train waits at grid A for 6 time steps (until t = 10) before resuming its journey. Despite the enhanced k-robust method prohibiting other trains from passing through grids D, C, B, A during time steps 1 to 4, 1 to 5, 1 to 6, and 1 to 13, in such cases, the train's cars remain stationary on their original grids from time steps 7 to 9, leading to a lapse in collision avoidance intervals. This indicates that if another train attempts to pass through grids D, C, B during time steps 5 to 10, 6 to 11, 7 to 12, collisions may potentially occur.

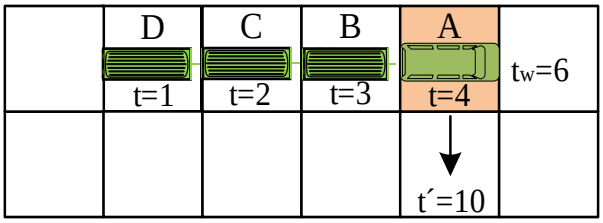


Figure 7. Schematic diagram of train waiting.

Hence, to prevent such waiting conflicts, it is imperative to incorporate additional temporal constraints into the train waiting process. This measure ensures that, even during train halts, other trains are prevented from accessing grids where collisions could occur, thereby comprehensively enhancing system safety.

5.2.2. Conflict-Free Strategy

To address train waiting conflicts, this paper proposes a time-step adjustment method designed to effectively prevent collisions between trains during waiting periods. Upon completion of route planning and safety interval updates at the train head, we analyze the positional relationship of adjacent path nodes to identify instances where the train is waiting at specific locations. We calculate the waiting duration (t_w) based on the cost differential between these adjacent nodes.

Specifically, if nodes node_a and node_b are consecutive path nodes where node_a_position equals node_b_position, this indicates that the train is waiting at node_b's position. The waiting time t_w can be determined by calculating the difference between node_b_g (the cost of node_b) and node_a_g (the cost of node_a). When updating the safety interval of the train head, if the head waits for t_w time steps at a certain time step, the corresponding time that the train body occupies adjacent grids will also extend by t_w time steps. Since the grids occupied by the train body are those the train head has already passed through, adjusting the time the train body occupies these grids essentially adjusts the time the train head passes through them. To determine the grids occupied by the train body during waiting, we trace back from the current position of the train head while excluding the waiting path nodes. Ultimately, the time steps at which the train head passes through the grids occupied by the car i will be adjusted from the original t to $t + t_w$. As illustrated in Figure 8, suppose a train of length k (where $k \leq 10$) passes through grids D, C, B, A at time steps 11, 12, 13, 14, respectively, and waits at grid A for 5 time steps until time step 19, still occupying grid A. To prevent conflicts, the time steps during which the train head passes through grids D, C, B need adjustment to 11 to 16, 12 to 17, 13 to 18. Consequently, other trains will be prohibited from passing through the corresponding grids during time steps $(11 - k, 16 + k)$, $(12 - k, 17 + k)$, $(13 - k, 18 + k)$.

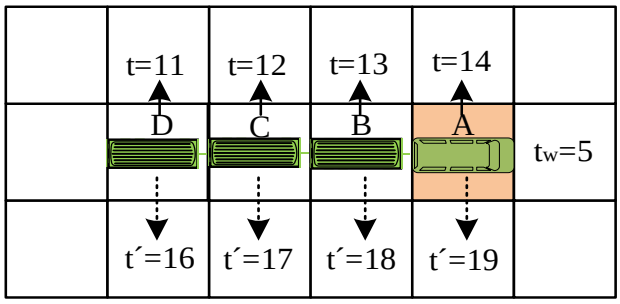


Figure 8. Time step adjustment of the car body occupying the grid when the train front is waiting.

Through this approach, we can introduce supplementary temporal constraints on the grids occupied by the train body during its waiting periods. This effectively prevents other trains from traversing these grids while the train is in a state of wait, thereby preempting potential collisions.

Before updating the safety interval list following successful single-machine planning, the pseudocode for identifying waiting nodes in the path and implementing time-step adjustment to prevent waiting conflicts is outlined as follows Algorithm 2:

Algorithm 2: detection_and_correction_of_waiting_nodes(*goal*)

```
path = make_path(goal)
for node in path do
    update_safe_interval(node)
    if node_position=next_node_position then
        t_w=next_node_g-node_g
        car_body=get_car_body(node)
        for M in car_body do
            for t0 in (M_t, M_t + t_w) do
                node0 = (M_position,t0)
                update_safe_interval(node0)
            end for
        end for
    end if
end for
```

To effectively prevent waiting conflicts after successful single-machine planning, we identify waiting nodes in the path before updating the safety interval list and implement time-step adjustments. This method is selected for two primary reasons: first, to avoid unnecessary checks for waiting conflicts and unnecessary time-step adjustments for every expanded node during the planning process, thereby reducing computational complexity and improving solution efficiency; second, because in single-machine planning, although many nodes are expanded, not all nodes ultimately contribute to the final path. Only those nodes selected to form the final path are considered genuinely significant.

5.3. Train Station Stop Conflicts and Resolution Methods

5.3.1. Definition of Conflict

Train station stop conflicts refer to situations during train operations where, for various reasons, one train attempts to stop at a station but encounters positional conflicts with another train. These conflicts primarily encompass three scenarios: firstly, the train's body occupies the destination point intended for another train after it has stopped (i.e., e_j $\in$ p_T_I and e_j $\neq$ e_i); secondly, the planned stopping position of the train is already occupied by another train; thirdly, after stopping, the train's body completely obstructs another train's path to its destination point.

During train head path planning, failure to adequately consider the actual grid occupation by the train body can lead to station stop conflicts, even when the target points of

two trains differ. Figure 9a illustrates the first conflict scenario where the body of train 1 occupies the target point of another train after stopping, directly preventing the other train from stopping as planned and causing planning failure. Figure 9b shows the second conflict scenario where the planned stopping position of train 2 is already occupied by another train, similarly leading to planning failure for train 2. Figure 9c depicts the third conflict scenario where, in narrow settings, one train completely obstructs another train's path to its target point after stopping, resulting in planning failure for the obstructed train. To mitigate these station stop conflicts, it is essential to fully consider the actual grid occupation by the train body during the path planning phase. This consideration should also encompass the relative positions and operational timings with respect to other trains, ensuring safe and efficient station stops and operations.

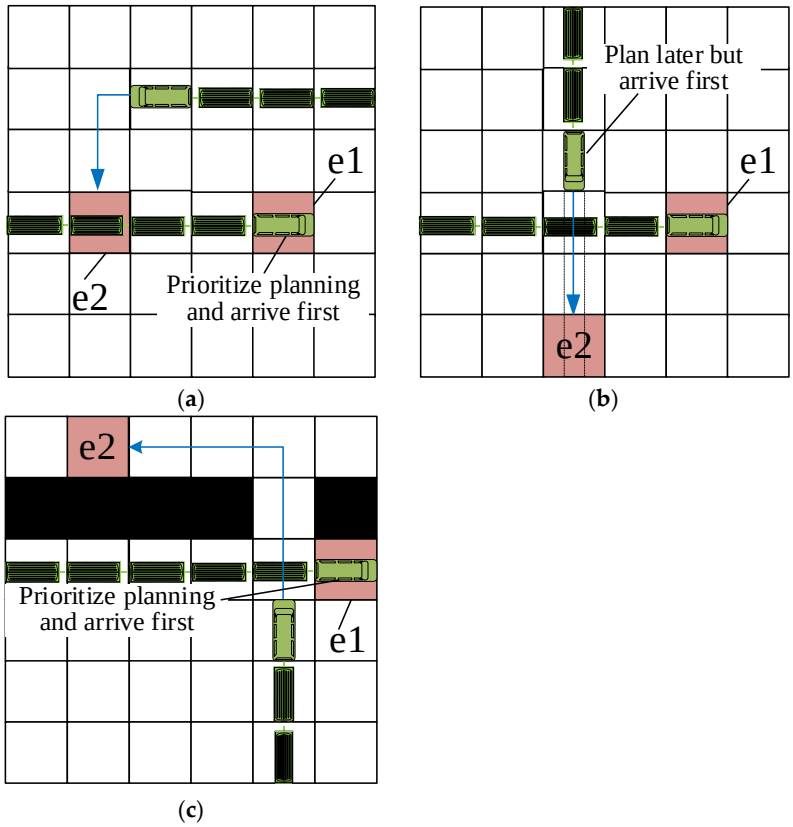


Figure 9. Three situations in which train stop conflicts occur. (**a**) Status 1; (**b**) Status 2; (**c**) Status 3.

5.3.2. Conflict-Free Strategy for Status 1 and Status 2

To effectively mitigate station stop conflicts described in scenarios 1 and 2, this study employs a strategy that combines grid position checks for parking and the addition of node constraints. When planning the path for each train to reach its destination point, the system traces back several preceding path nodes before the target point to determine the specific grid range occupied by the train body upon arrival (referred to as car_position). Subsequently, the system checks whether car_position intersects with the destination points of other trains (stored in the GOALS list) or grids occupied by already stopped trains (stored in the CARS list). If car_position $\cap$ GOALS $\neq \varnothing$ or car_position $\cap$ CARS $\neq \varnothing$, it signifies that stopping at the current extended path node would lead to conflicts with other trains. In such cases, the algorithm discards the current node and continues exploring other potential path nodes until finding a car_position that neither overlaps with any destination points in the GOALS list nor with grids occupied by stopped trains in the CARS list (i.e., car_position $\cap$ GOALS $= \varnothing$ and car_position $\cap$ CARS $= \varnothing$). This ensures that the current train can safely stop at that node. The algorithm retains this path node for subsequent backtracking and generation of a comprehensive conflict-free path.

It is noteworthy that nodes in the GOALS list are extracted during the algorithm initialization phase by reading map task information. In contrast, the CARS list requires dynamic updating after completion of each train's path planning phase. This involves backtracking from the target points to capture the grid positions occupied by the train body upon the final station stop, ensuring real-time reflection of track occupancy.

The pseudocode for this process is as follows Algorithm 3:

Algorithm 3: resolve_station_stopping_conflict-1(s)

```
1: if s = goal do
2:     car_position = get_car_position(s)
3:     if car_position ∩ {GOALS ∪ CARS} ≠ ∅ then
4:         s = get_best_node(OPEN)
5:         conflict = True
6:     else
7:         conflict = False
8:     end if
9: end if
```

5.3.3. Conflict-Free Strategy for Status 3

To address the third type of station stop conflict as depicted in Figure 9, which cannot be preemptively detected during the initial train planning phase, we adopt a post-processing strategy. In cases where train 2 encounters a scenario where its path to the destination is entirely blocked by other trains during planning, we prioritize the planning of train 2. This prioritization ensures that train 2 is planned first, allowing train 1, which is planned subsequently, to make decisions based on train 2's path, either waiting for train 2 to reach its destination point before occupying its own destination point or opting for an alternative path to avoid obstructing train 1's arrival.

To achieve this objective, this study employs a method for continuous multi-node endpoint conflict detection. In a priority-based multi-train path planning algorithm, whenever a train's leading node extends to reach a target point, we examine the g value of the current extension node (which represents the time cost for the train to reach its target point) and assess whether other trains are scheduled to pass through the same target point subsequently. If the g-value of another train passing through the current train's target point (denoted as g_i) is less than or equal to the current train's g value (i.e., $g \leq g_i$), it indicates a conflict. In such cases, we discard that node and continue expanding until we find a node with a g value less than g_i, indicating that no further trains are scheduled to pass through the current train's target point, thereby achieving successful planning.

In multi-train path planning, in addition to single-node conflict detection for each train's head, it is crucial to perform endpoint conflict detection for grids occupied by the connected train bodies along adjacent connections. This ensures that the current planned train can stop safely without obstructing previously planned trains that have already passed through its station grids, thus preventing potential collisions. However, in specific scenarios where prioritizing the planning of one train, such as train 2, inadvertently blocks another train, like train 1, from reaching its destination, it indicates a mutual constraint between the trains. To resolve this issue, when replanning train 1, we proactively add the stop position of train 2 from the previous planning round (referred to as car_position_2) to the constraint list. This ensures that when prioritizing the planning of train 1, the algorithm selects a stopping location that avoids occupying grids in car_position_2, thereby enabling train 2 to reach its destination unimpeded. Nevertheless, it is important to note that in highly congested and narrow scenarios where each train has only one viable path to its destination, mutual constraints between trains may render the algorithm unable to find a solution. In such cases, additional strategies or measures may be required to resolve these path conflicts effectively.

The pseudocode for this section is as follows Algorithm 4:

Algorithm 4: resolve_station_stopping_conflict-2(s)

```
if s = goal do
    car_position = get_car_position(s)
    result←conflict detection(car_position)
    if result = False then
        s←get_best_node(OPEN)
        conflict = True
    else
        conflict = False
    end if
    if car_position ∩ constaints ≠ ∅ do
        s←get best node(OPEN)
        conflict = True
    else
        conflict = False
    end if
end if
```

Finally, as depicted in Figure 10, following the completion of train planning, the system ensures operational safety and prevents future conflicts by excluding grids occupied by the train body after it reaches its final destination and extending its safety interval. This measure effectively prohibits other trains from passing through grids where a train has already halted, thereby maintaining safety intervals between trains and substantially mitigating potential conflicts.

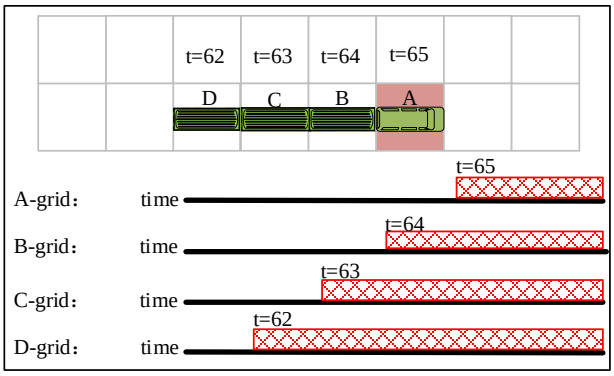


Figure 10. Update of the safety interval of the car body occupying the grid after the train stops.

6. Flowchart of MT-SIPP Algorithm

In conclusion, the framework of the MT-SIPP algorithm, illustrated in Figure 11, comprises four meticulously designed core components aimed at achieving efficient and secure train path planning. These components include: Station Conflict Detection and Handling, responsible for real-time detection and effective resolution of potential station conflicts during planning; Waiting Node Identification and Time Step Adjustment, ensuring the recognition of waiting scenarios and adjusting time steps accordingly to prevent conflicts; Effective Safety Interval Update, which updates and excludes safety intervals of grids occupied by completed train plans to preempt subsequent conflicts; and Priority Adjustment, which flexibly reorders train planning based on mutual constraints and priorities, ensuring smooth problem resolution.

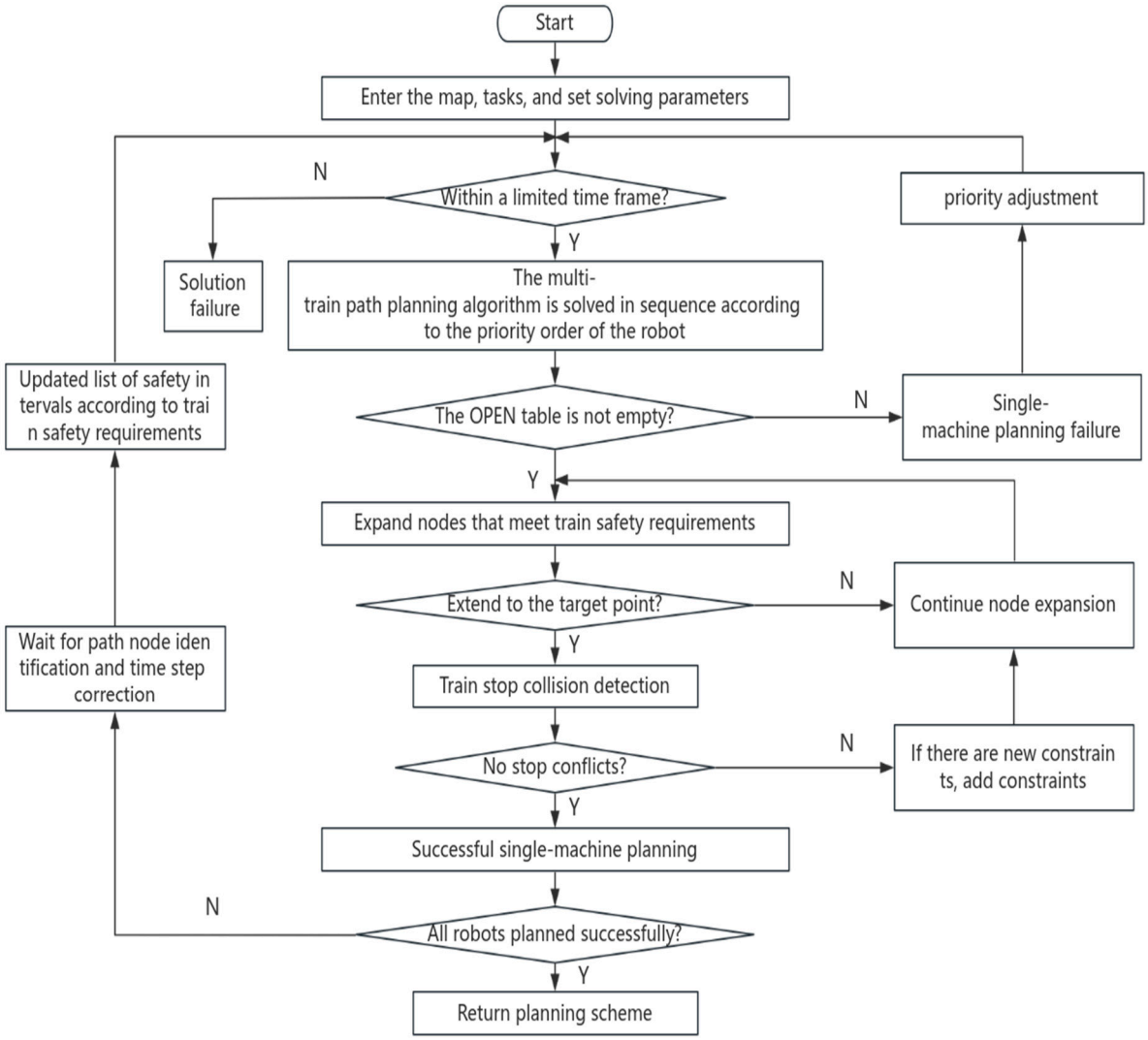


Figure 11. MT-SIPP algorithm block diagram.

7. Experimental Analysis

This paper compares five multi-train path planning algorithms, MT-CBS, MT-ICBS, MT-CBSH, MT-CBSH-RM [29], and MT-SIPP, on three benchmark map environments [6] (blank-empty-48-48, random-random-32-32-20, and room-room-32-32-4) for a performance evaluation. MT-CBS, MT-ICBS, MT-CBSH, and MT-CBSH-RM are based on the basic CBS (Conflict-Based Search) algorithm, the modified CBS algorithm, the CBS algorithm with heuristics, and the CBS algorithm combining the heuristics and MDD (Manhattan Distance on a Grid) for rectangular conflict reasoning is implemented in the CBS algorithm. In the experiments, the number of trains n was systematically increased, with 25 instances tested for each train count. In tables, algorithms with notably shorter average runtime under identical conditions are highlighted in bold. "-" marks denote instances where the compared algorithms failed to produce solutions under the corresponding conditions, enhancing the visual comparison of algorithm performance across different scenarios.

Our experimental setup consisted of a Windows 10 operating system, a 2.3 GHz processor, and 8 GB of memory. The algorithms MT-CBS, MT-ICBS, MT-CBSH, and MT-CBSH-RM were implemented in C++, while the MT-SIPP algorithm was implemented in Python. Each algorithm was restricted to solving individual instances within a 2 min time frame. Instances failing to provide a valid solution within this limit were categorized as unsuccessful.

7.1. Testing in a Blank Map Environment

The blank map environment is distinguished by a prominent feature: the entire map space consists of obstacle-free open areas. This environment provides optimal conditions for the unrestricted movement and efficient interaction of multiple robots. As depicted in Figure 12, this study selected this typical blank map environment, sized 48 × 48, for evaluating algorithm performance. Figure 13 visually illustrates the comparative results of MT-CBS, MT-ICBS, MT-CBSH, MT-CBSH-RM, and MT-SIPP algorithms in terms of their success rates in solving problems within the blank map environment. This comparison offers a clear insight into the performance disparities among the algorithms under these specific environmental conditions.

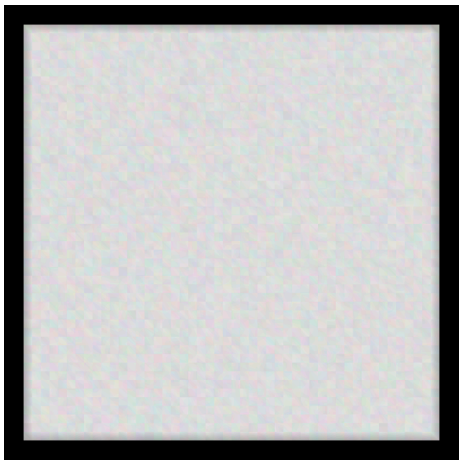


Figure 12. Empty map Empty-48-48.

Based on the success rate comparison depicted in Figure 13, it is evident that among the five multi-train path planning algorithms MT-CBS, MT-ICBS, MT-CBSH, MT-CBSH-RM, and the proposed MT-SIPP algorithm consistently exhibit the highest success rates in the empty map environment. Of particular note is that although the MT-CBSH-RM algorithm, a prominent CBS-type algorithm, shows comparable success rates to the MT-SIPP algorithm when the value of k (train body length) is small, the superiority of the MT-SIPP algorithm becomes more pronounced as k increases. Detailed statistical analysis reveals that at $k = 1$, the MT-SIPP algorithm exhibited average success rate improvements of 44.7%, 44%, 42.6%, and 5.6% compared to the other four algorithms. At $k = 2$, these enhancements were 40%, 38%, 39%, and 17%. At $k = 3$, the improvements were consistently 42%, 42%, 42%, and 33%. Notably, at k = 4, the algorithm achieved substantial improvements of 49.1%, 48.4%, 48.4%, and 43.6%, while at $k = 5$, the figures were 48%, 47.6%, 46.9%, and 44%. These findings underscore a significant performance advantage of the MT-SIPP algorithm over CBS-like multi-agent pathfinding algorithms in achieving higher success rates, with an average improvement nearing 40% in blank map environments. Furthermore, regarding the scalability in handling multiple train instances, particularly at larger k values (e.g., $k = 4.5$), MT-SIPP demonstrated superior capability, effectively managing nearly twice the maximum train instances compared to alternative algorithms.

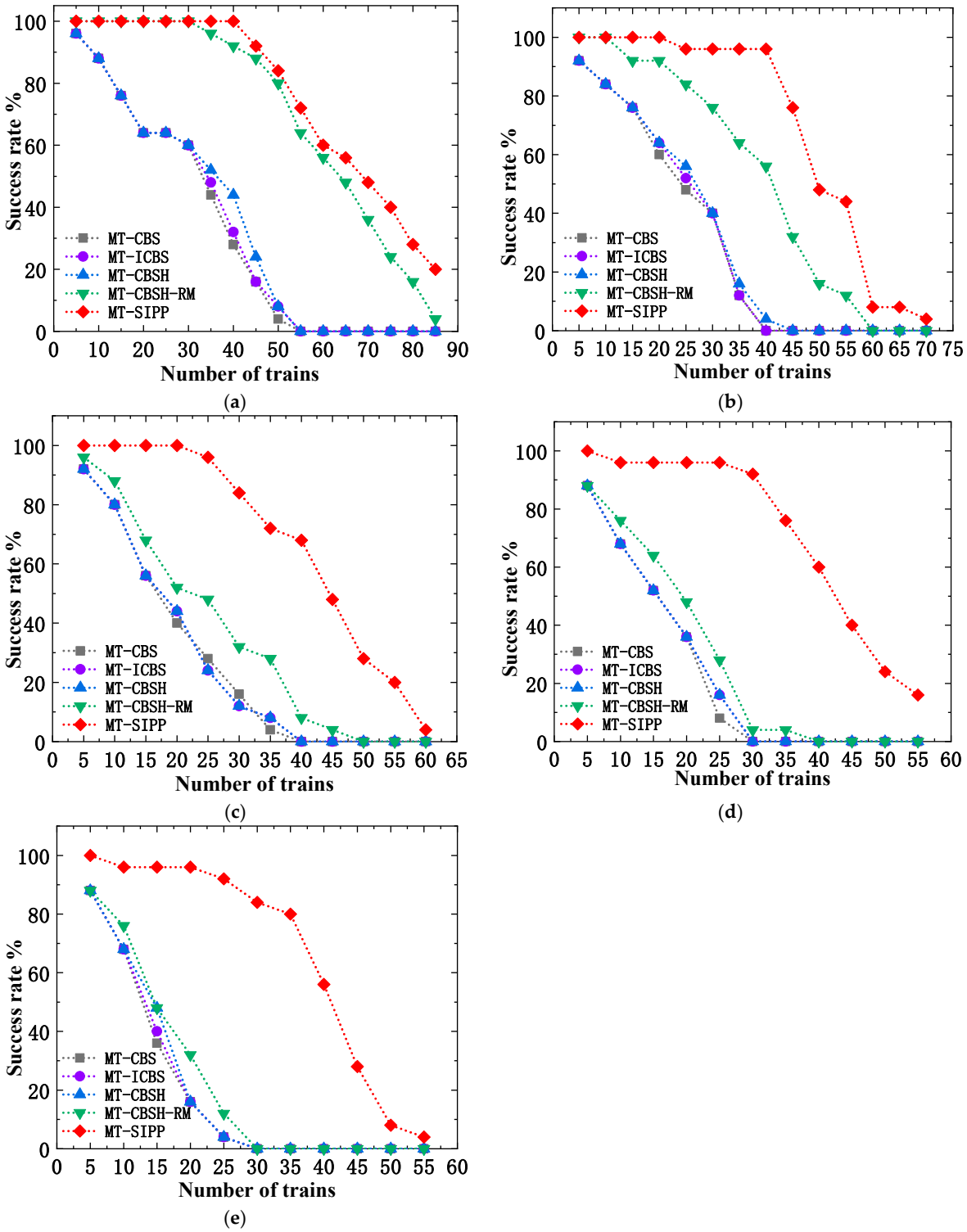


Figure 13. Comparison of success rates of several algorithms in blank map environment. (**a**) $k = 1$; (**b**) $k = 2$; (**c**) $k = 3$; (**d**) $k = 4$; (**e**) $k = 5$.

In our statistical analysis of algorithm runtime (as shown in Table 1), we compared the MT-CBSH-RM algorithm, known for its superior efficiency in CBS-like multi-train planning algorithms, with the MT-SIPP algorithm. In a blank map environment, MT-CBSH-RM demonstrates better algorithmic runtime efficiency than MT-SIPP when k (train length) is small or when solving a small number of trains. However, as the number of trains or k increases, CBS-like multi-train planning algorithms experience a rapid expansion of their

solution space, resulting in a gradual decline in efficiency. This efficiency gap becomes more pronounced with increasing problem complexity.

Table 1. Runtime statistics (s) of two algorithms in blank map environments.

n	k = 1		k = 2		k = 3		k = 4		k = 5	
	MT-SIPP	**MT-CBSH-RM**	**MT-SIPP**	**MT-CBSH-RM**	**MT-SIPP**	**MT-CBSH-RM**	**MT-SIPP**	**MT-CBSH-RM**	**MT-SIPP**	**MT-CBSH-RM**
5	0.416	**0.003**	0.432	**0.016**	**0.483**	5.153	**0.443**	14.934	**0.663**	15.201
10	0.563	**0.007**	0.615	**0.022**	**0.729**	16.249	**6.111**	29.459	**6.363**	30.023
15	0.725	**0.019**	**1.103**	9.75	**1.058**	39.504	**6.541**	44.933	**6.664**	64.845
20	1.051	**0.035**	**1.573**	9.946	**1.474**	57.664	**7.456**	62.731	**8.188**	82.982
25	1.548	**0.053**	**6.874**	20.533	**6.918**	62.574	**9.1**	86.532	—	—
30	1.851	**0.109**	**7.495**	29.742	**22.032**	81.689	—	—	—	—
35	7.377	**5.155**	**7.996**	44.751	**36.948**	87.988	—	—	—	—
40	8.529	**7.244**	**9.569**	64.221	**42.265**	110.499	—	—	—	—
45	**11.073**	17.743	**34.138**	85.798	**66.665**	115.241	—	—	—	—
50	**21.69**	26.874	**66.568**	101.118	—	—	—	—	—	—
55	**35.186**	46.069	**72.455**	105.727	—	—	—	—	—	—
60	**48.575**	46.335	—	—	—	—	—	—	—	—
65	**54.116**	62.783	—	—	—	—	—	—	—	—
70	**68.208**	78.754	—	—	—	—	—	—	—	—
75	**79.471**	92.465	—	—	—	—	—	—	—	—
80	**93.359**	101.921	—	—	—	—	—	—	—	—

The aforementioned results indicate that in a blank map environment, CBS-like multi-train planning algorithms, particularly the MT-CBSH-RM algorithm, excel in both success rate and runtime efficiency when k values are small and the number of trains is limited. This is largely attributed to the expansive layout of the map, which allows ample maneuvering space for trains. However, as the number of trains increases or k values grow larger, the available maneuvering space for trains gradually diminishes, leading to a significant decrease in the solving efficiency of CBS-like algorithms. In contrast, under these circumstances, the MT-SIPP algorithm consistently maintains higher solving efficiency, demonstrating its superior performance in handling complex multi-train pathfinding problems.

7.2. Random Map Environment Testing

In a random map environment, the distribution of obstacles is stochastic, leading to dispersed and variably-sized passable areas for trains. This variability undoubtedly presents considerable challenges for multi-train planning on such maps. As depicted in Figure 14, this study utilized a representative random map measuring 32 × 32 for testing purposes. To thoroughly evaluate the performance of diverse algorithms in this setting, comprehensive testing was conducted, and the findings are consolidated in Figure 15 and Table 2.

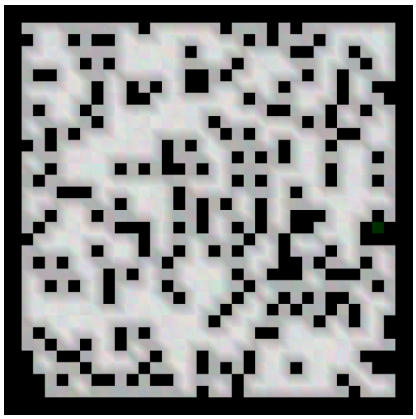


Figure 14. Random map random-32-32-20.

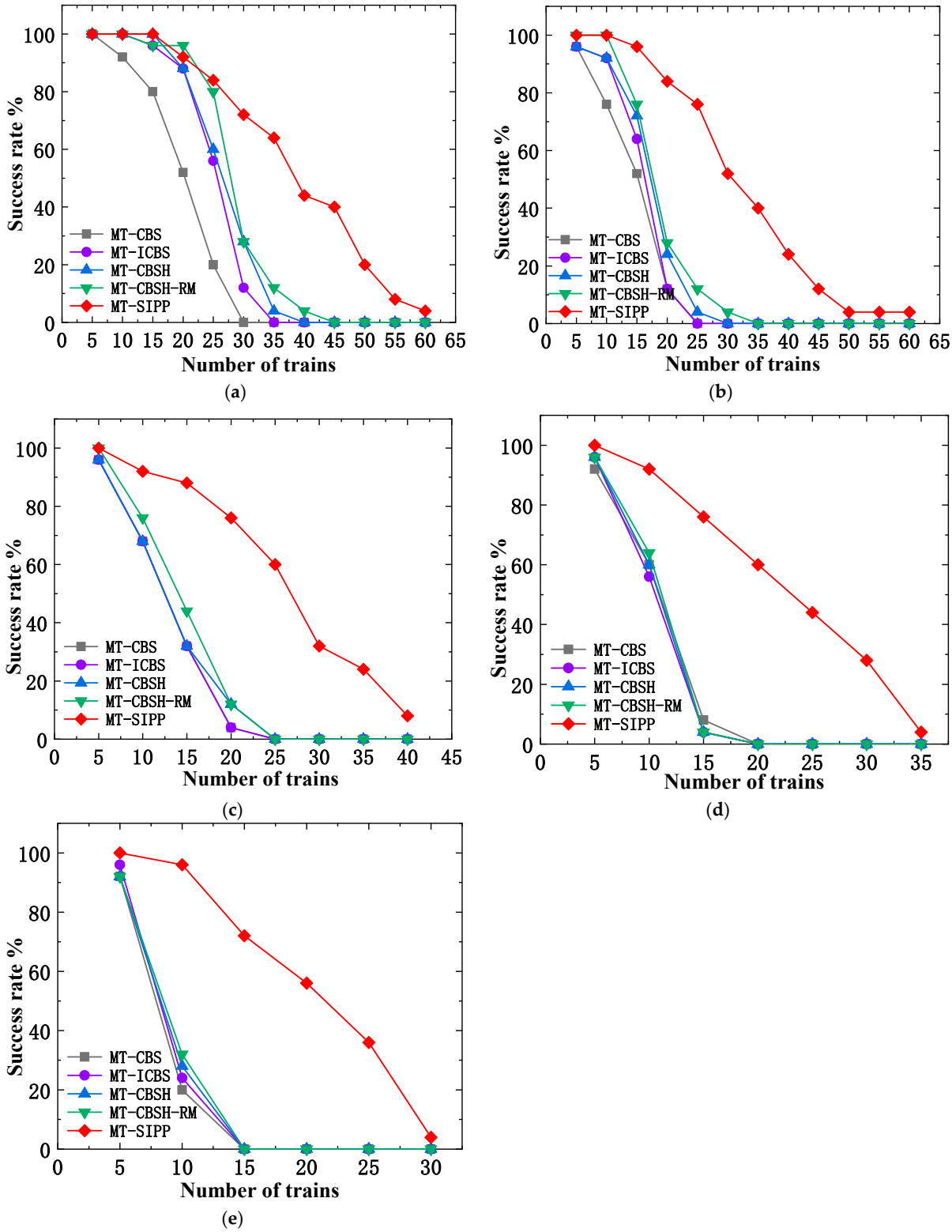


Figure 15. Comparison of solving success rates of several algorithms in a random map environment. (**a**) $k = 1$; (**b**) $k = 2$; (**c**) $k = 3$; (**d**) $k = 4$; (**e**) $k = 5$.

Table 2. Runtime statistics (s) of two algorithms in random map environments.

n	k = 1		k = 2		k = 3		k = 4		k = 5	
	MT-SIPP	**MT-CBSH-RM**	**MT-SIPP**	**MT-CBSH-RM**	**MT-SIPP**	**MT-CBSH-RM**	**MT-SIPP**	**MT-CBSH-RM**	**MT-SIPP**	**MT-CBSH-RM**
5	0.525	**0.007**	0.586	**0.011**	0.597	**0.043**	**0.592**	4.914	**0.594**	10.639
10	0.589	**0.051**	**0.919**	2.651	**10.545**	31.168	**10.406**	56.327	**6.234**	82.654
15	**0.712**	4.976	**6.104**	35.48	**15.924**	77.715	**31.447**	115.636	—	—
20	**10.062**	10.319	**20.281**	91.697	**31.347**	108.167	—	—	—	—
25	**20.361**	35.737	**30.889**	107.895	—	—	—	—	—	—
30	**35.097**	88.66	—	—	—	—	—	—	—	—
35	**44.867**	109.902	—	—	—	—	—	—	—	—
40	**68.949**	118.032	—	—	—	—	—	—	—	—

Based on the comparative results of success rates depicted in Figure 15, it is evident that in random map environments, the high coverage of obstacles leads to an increasing number of conflicts among trains as the number of trains to be solved increases. This diminishes the available maneuvering space and gradually lowers the success rates of all algorithms. Furthermore, as the value of k (train car length) increases, the number of empty grids occupied by trains and their occupation duration also increase, thereby reducing the maximum feasible number of trains that can be effectively managed by several multi-train planning algorithms. Nevertheless, it is worth noting that across various k values, the MT-SIPP algorithm consistently achieves higher success rates than several other algorithms. This superiority is particularly evident at $k = 5$. In contrast, CBS-like multi-train path planning algorithms show minimal differences in success rates, with their performance nearly converging as k increases. This trend primarily stems from the constrained passable areas in random maps. At lower k values, the enhancement strategies integrated into the MT-CBS algorithm contribute to improved success rates. However, as k increases, further reduction in passable space and rapid growth in conflict search state space diminish the effectiveness of these strategies in enhancing success rates. Overall, the MT-SIPP algorithm exhibits an average increase in success rates of approximately 30% compared to CBS-like multi-train planning algorithms. Particularly noteworthy is its ability to handle more than twice the maximum number of solvable trains compared to CBS-like algorithms when $k > 5$. This robust performance underscores the superiority of MT-SIPP in addressing multi-train path planning challenges in random map environments.

According to the data analysis from Table 2, the prevalence of numerous random obstacles in random map environments significantly complicates the task of multi-train planning. In scenarios with smaller k values and fewer trains to solve, the MT-CBSH-RM algorithm indeed exhibits shorter runtimes compared to the MT-SIPP algorithm. However, once the number of trains to be solved exceeds 10, regardless of the k value, the runtime of the MT-CBSH-RM algorithm experiences exponential growth, making it difficult to provide solutions within a reasonable time frame.

In contrast, while the MT-SIPP algorithm does experience increased runtime as the number of trains to solve increases, the magnitude of this increase is significantly smaller compared to the MT-CBSH-RM algorithm. Consequently, when solving for more than 10 trains (i.e., $n > 10$), the MT-SIPP algorithm consistently demonstrates shorter runtimes than MT-CBSH-RM, sometimes averaging as little as one-tenth of the latter's runtime. This finding strongly validates the superior performance of the MT-SIPP algorithm in addressing large-scale multi-train planning problems.

7.3. Indoor Map Environment Testing

The indoor map environment replicates the layout of partially enclosed rooms typically found in real-world scenarios, interconnected by narrow passages allowing only one train to pass at a time. In this unique map environment, characterized by spatial constraints and restricted passages, congestion between trains can readily occur, thereby greatly augmenting the complexity and challenges associated with multi-train planning. The

indoor room map we tested is illustrated in Figure 16, with comprehensive test results detailed in Figure 17 and Table 3.

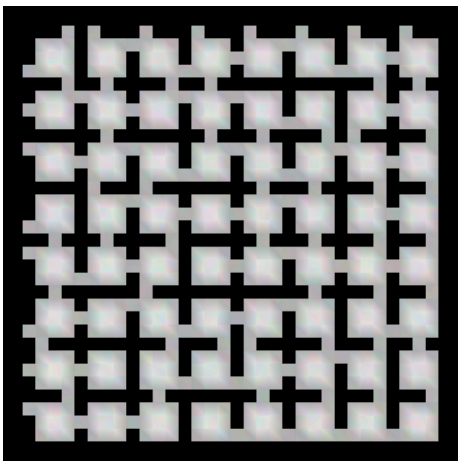


Figure 16. Room map Room-32-32-4.

In Figure 17, we have contrasted the success rates of several algorithms in solving room-based maps. The findings reveal that MT-CBS exhibits the lowest success rate, whereas the MT-SIPP algorithm displays the highest success rate. It is worth noting that the performance of the multi-train path planning algorithms that incorporate improvements over MT-CBS shows varying degrees of success and consistency. In specific terms, at $k = 1$, both the MT-CBSH and MT-CBSH-RM algorithms achieve identical success rates, slightly edging out MT-ICBS. However, as k increases, the scenario evolves. At $k = 2$ and $k = 4$, MT-ICBS shows slightly superior performance compared to MT-CBSH and MT-CBSH-RM. Conversely, at $k = 3$, MT-CBSH-RM outperforms MT-ICBS and MT-CBSH. By $k = 5$, MT-CBSH exhibits marginally better performance than MT-CBSH-RM and MT-ICBS. This inconsistency in results may be attributed to the unique characteristics of room-based maps and the high complexity inherent in multi-train path planning. In certain cases, the addition of more improvement strategies might inadvertently reduce success rates due to increased algorithmic complexity. Overall, in room-based maps, the MT-SIPP algorithm exhibits a notable average increase of approximately 27% in success rates compared to CBS-like multi-train path planning algorithms. This improvement is quite significant.

Based on the data from Table 3, in the context of room-based map environments, it is observed that when $k = 1$ and the number of trains n is less than 10, the MT-CBSH-RM algorithm indeed shows shorter runtime compared to the MT-SIPP algorithm. However, across all other test conditions, the MT-SIPP algorithm consistently demonstrates shorter runtime.

It is noteworthy that many of the connecting passages between rooms in the room-based maps are single-channel, significantly restricting the throughput of trains. This structural limitation predisposes CBS-like multi-train planning algorithms to conflict generation when applied in such environments. To find conflict-free solutions, algorithms must continually search and replan numerous potential conflict nodes, which inevitably consumes substantial computational resources. Particularly in densely populated train scenarios, achieving a conflict-free solution becomes increasingly challenging. Therefore, the MT-SIPP algorithm demonstrates superior efficiency and adaptability in tackling these intricate indoor multi-train planning problems.

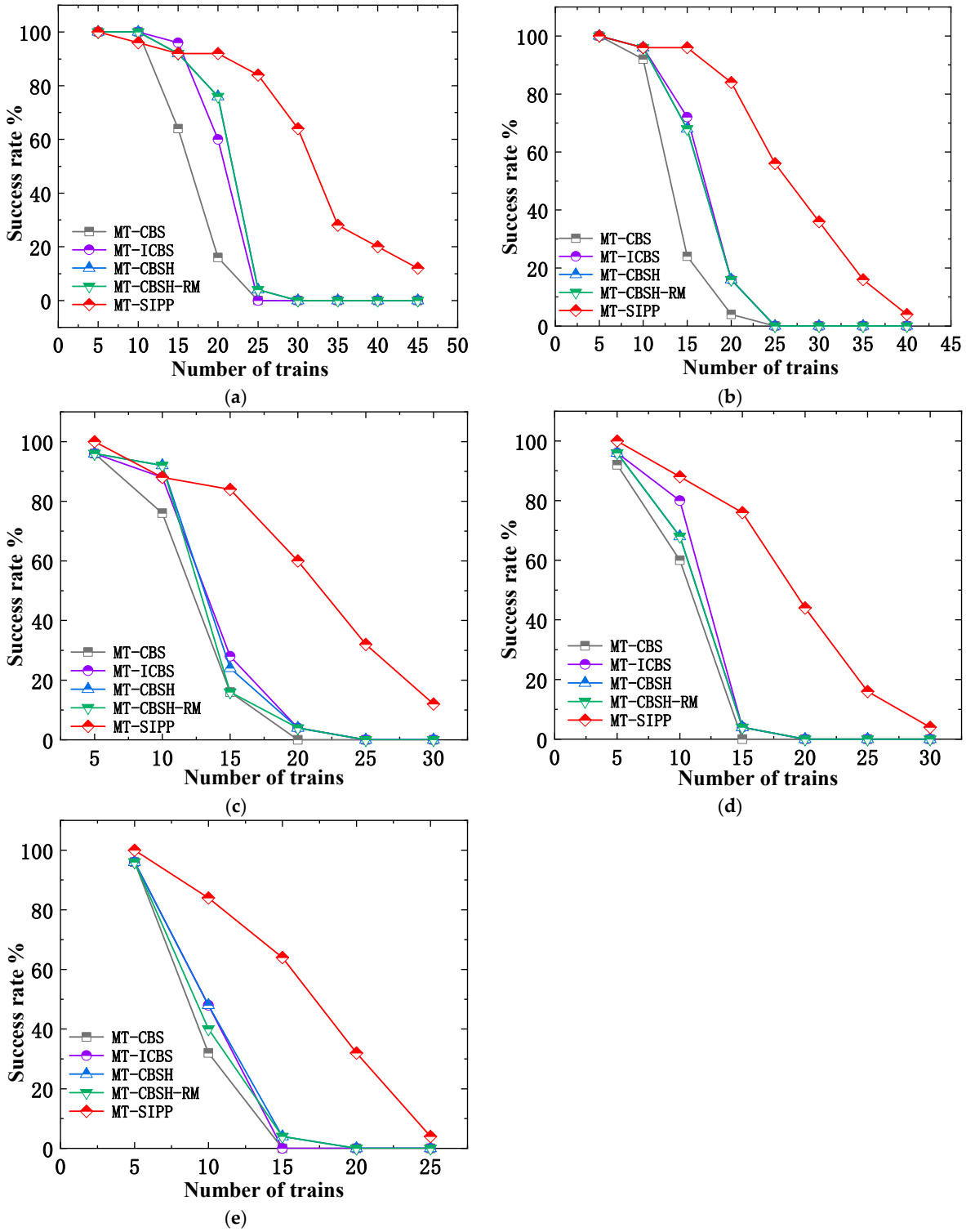


Figure 17. Comparison of the success rates of several algorithms in the room map environment. (**a**) $k = 1$; (**b**) $k = 2$; (**c**) $k = 3$; (**d**) $k = 4$; (**e**) $k = 5$.

Table 3. Runtime statistics (s) of two algorithms in indoor map environment.

n	k = 1		k = 2		k = 3		k = 4		k = 5	
	MT-SIPP	**MT-CBSH-RM**	**MT-SIPP**	**MT-CBSH-RM**	**MT-SIPP**	**MT-CBSH-RM**	**MT-SIPP**	**MT-CBSH-RM**	**MT-SIPP**	**MT-CBSH-RM**
5	0.141	**0.003**	**0.182**	1.978	**0.206**	4.825	**0.201**	4.835	**0.197**	5.028
10	5.043	**0.048**	**4.282**	5.012	**15.235**	11.454	**15.659**	46.146	**20.641**	75.108
15	**9.918**	11.623	**5.499**	45.446	**21.503**	103.262	**31.383**	115.334	**45.167**	117.688
20	**10.313**	45.438	**20.964**	102.241	**50.649**	115.955	—	—	—	—
25	**20.793**	118.389								

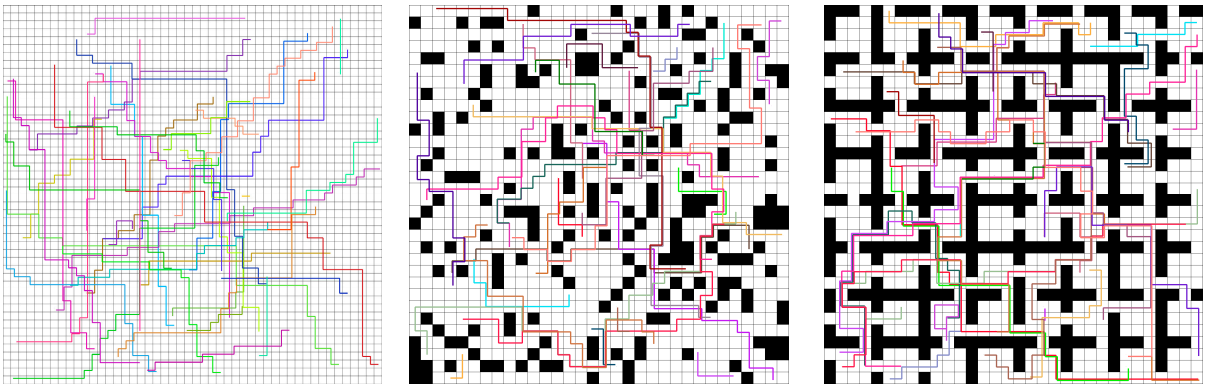
7.4. Analysis and Discussion of Experimental Results

The experimental results across various map environments indicate that while strategies such as integrating heuristic function *h*-value guidance in search and employing MDD for rectangle conflict resolution can enhance the solving capabilities of the MT-CBS algorithm to some extent, its effectiveness is hindered by the CBS algorithm's reliance on unit discretization nodes for path expansion and conflict detection mechanisms. This limitation results in an exponential increase in solution space as the number of trains n and train lengths k grow. Despite the incorporation of additional improvement strategies, the overall enhancement in algorithmic efficiency remains rather limited. In contrast, the MT-SIPP algorithm proposed in this study primarily expands path nodes and avoids conflicts based on global safe intervals, demonstrating a linear relationship between search space and both n and k. Consequently, as n and k increase beyond certain thresholds, the superior efficiency of the MT-SIPP algorithm in solution space becomes increasingly apparent.

All test results unequivocally demonstrate that our algorithm surpasses existing multi-train path planning algorithms (MT-CBS, MT-ICBS, MT-CBSH, MT-CBSH-RM) in several key metrics: success rate, maximum number of solvable trains, and algorithm runtime. Particularly notable is the MT-SIPP algorithm's ability to achieve shorter runtimes and higher solving efficiency when confronted with larger numbers of trains n and longer train lengths k. Furthermore, our algorithm guarantees that the total cost of generated path solutions deviates from the optimal path's total cost by no more than 10%. In scenarios involving blank maps or random and room-based maps with fewer trains, this deviation is further minimized to within 5%. This thoroughly validates the feasibility and efficacy of our algorithm. Despite making compromises in total path cost, its significant advancements in success rate, maximum solvable capacity, and solving efficiency underscore its broad applicability in the realm of multi-train path planning.

7.5. Visualization of Path Solutions

Figure 18 visually demonstrates the path planning solutions generated by the MT-SIPP algorithm across diverse map environments. The figure highlights that in open, obstacle-free blank maps, despite the absence of direct obstacles, the paths of a significantly increased number of robots become intricately intertwined, leading to higher path overlaps and potential conflicts. This underscores the necessity for multi-chain robot planning algorithms to exhibit strong conflict coordination capabilities. Conversely, in maps with random layouts or distinct room partitions, robot paths concentrate heavily on narrow passages that must be traversed. These areas serve as conflict hotspots, thereby imposing higher demands on the algorithm's conflict management strategies.



(**a**) (**b**) (**c**)

Figure 18. Path planning solutions by our MT-SIPP algorithm. (**a**) empty-48-48 ($n = 40, k = 4$); (**b**) random-32-32-20 ($n = 30,\ k = 3$); (**c**) room $-32-32-4$ ($n = 30, k = 2$).

8. Conclusions

To achieve cost savings and improve transport efficiency, snake-like robots, akin to train-like transport vehicles, are widely employed across multiple scenarios for cargo handling operations. However, in CBS-like multi-train path planning algorithms, the detection and resolution of conflicts for each segment of a snake-like robot occupying grid nodes often entail intricate processes of conflict node splitting and path searching. This significantly increases the algorithm's runtime. At times, such complexity may prevent the algorithm from delivering feasible solutions within the allocated time, thereby failing to meet the efficient path planning requirements of multi-robot systems.

In addressing this challenge, we propose the Multi-Train Safety Interval Path Planning (MT-SIPP) algorithm in this study. This algorithm harnesses the benefits of safety intervals in solving path planning issues for snake-like robots. By directly adjusting safety intervals and incorporating additional node constraints, it efficiently mitigates conflicts among snake-like robots, thereby markedly improving path planning efficiency. The simulation test results convincingly validate the superiority of the MT-SIPP algorithm. Compared to MT-CBS, MT-ICBS, MT-CBSH, and MT-CBSH-RM algorithms, MT-SIPP demonstrates significant advancements in both solving capability and efficiency. Specifically, our algorithm shows an average 28% increase in success rate and can handle up to twice the maximum number of trains solvable by other algorithms. Particularly in scenarios with a large number of trains and longer train lengths, MT-SIPP consistently provides solutions rapidly and efficiently, thereby demonstrating its exceptional performance in the field of multi-train path planning.

In the future, we aim to significantly enhance the performance of our MT-SIPP algorithm by optimizing conflict resolution strategies for greater efficiency and robustness. Moreover, we will address more complex aspects of chain robots, such as kinematic constraints (such as precise steering times and acceleration/deceleration durations), larger scales, and extended vehicle lengths, thereby advancing solutions for industrial automation and intelligent logistics.

Author Contributions: Conceptualization, J.M. and J.T.; Formal Analysis, P.L. and L.Y.; Methodology, J.M. and P.L.; Resources, P.L. and C.C.; Writing—Raw Drafts, J.M.; Writing—Review and Editing, P.L. and L.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by "Scientific Research Project on Basic Research Operating Expenses of Universities in the Autonomous Region (XJEDU2024P090)" and "Xinjiang Uygur Autonomous Region Natural Science Foundation Program (2022D01C461)".

Data Availability Statement: The original contributions presented in the study are included in the article, further inquiries can be directed to the corresponding authors.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Li, P.; Yang, L. Conflict-free and energy-efficient path planning for multi-robots based on priority free ant colony optimization. *Math. Biosci. Eng.* **2023**, *20*, 3528–3565. [CrossRef] [PubMed]
2. Ma, H.; Hönig, W.; Kumar, T.S.; Ayanian, N.; Koenig, S. Lifelong path planning with kinematic constraints for multi-agent pickup and delivery. In *Proceedings of the AAAI Conference on Artificial Intelligence*; AAAI Press: Washington, DC, USA, 2019; Volume 33, pp. 7651–7658. [CrossRef]
3. Furchì, A.; Lippi, M.; Carpio, R.F.; Gasparri, A. Route optimization in precision agriculture settings: A multi-steiner TSP formulation. *IEEE Trans. Autom. Sci. Eng.* **2022**, *20*, 2551–2568. [CrossRef]
4. Han, S.D.; Yu, J. Effective heuristics for multi-robot path planning in warehouse environments. In Proceedings of the 2019 International Symposium on Multi-Robot and Multi-Agent Systems (MRS), New Brunswick, NJ, USA, 22–23 August 2019; pp. 10–12. [CrossRef]
5. Chen, X.; Li, Y.; Liu, L. A coordinated path planning algorithm for multi-robot in intelligent warehouse. In Proceedings of the 2019 IEEE International Conference on Robotics and Biomimetics (ROBIO), Dali, China, 6–8 December 2019; pp. 2945–2950. [CrossRef]
6. Stern, R.; Sturtevant, N.; Felner, A.; Koenig, S.; Ma, H.; Walker, T.; Li, J.; Atzmon, D.; Cohen, L.; Kumar, T.K. Multi-agent pathfinding: Definitions, variants, and benchmarks. In *Proceedings of the International Symposium on Combinatorial Search*; AAAI Press: Washington, DC, USA, 2019; Volume 10, pp. 151–158. [CrossRef]
7. Ma, H.; Kumar, T.S.; Koenig, S. Multi-agent path finding with delay probabilities. In *Proceedings of the AAAI Conference on Artificial Intelligence*; AAAI Press; Washington, DC, USA, 2017; Volume 31. [CrossRef]
8. Hönig, W.; Kiesel, S.; Tinka, A.; Durham, J.W.; Ayanian, N. Persistent and robust execution of MAPF schedules in warehouses. *IEEE Robot. Autom. Lett.* **2019**, *4*, 1125–1131. [CrossRef]
9. Li, B.; Page, B.R.; Moridian, B.; Mahmoudian, N. Collaborative Mission Planning for Long-Term Operation Considering Energy Limitations. *IEEE Robot. Autom. Lett.* **2020**, *5*, 4751–4758. [CrossRef]
10. Cohen, L.; Uras, T.; Kumar, T.K.; Koenig, S. Optimal and bounded-suboptimal multi-agent motion planning. In *Proceedings of the International Symposium on Combinatorial Search*; AAAI Press: Washington, DC, USA, 2019; Volume 10, pp. 44–51. [CrossRef]
11. Li, J.; Chen, Z.; Harabor, D.; Stuckey, P.J.; Koenig, S. MAPF-LNS2: Fast repairing for multi-agent path finding via large neighborhood search. In *Proceedings of the AAAI Conference on Artificial Intelligence*; AAAI Press: Washington, DC, USA, 2022; Volume 36, pp. 10256–10265. [CrossRef]
12. Li, B.; Ma, H. Double-deck multi-agent pickup and delivery: Multi-robot rearrangement in large-scale warehouses. *IEEE Robot. Autom. Lett.* **2023**, *8*, 3701–3708. [CrossRef]
13. Sharon, G.; Stern, R.; Felner, A.; Sturtevant, N.R. Conflict-based search for optimal multi-agent pathfinding. *Artif. Intell.* **2015**, *219*, 40–66. [CrossRef]
14. Boyarski, E.; Felner, A.; Stern, R.; Sharon, G.; Betzalel, O.; Tolpin, D.; Shimony, E. ICBS: The improved conflict-based search algorithm for multi-agent path finding. In *Proceedings of the International Symposium on Combinatorial Search*; AAAI Press: Washington, DC, USA, 2015; Volume 6, pp. 223–225. [CrossRef]
15. Felner, A.; Li, J.; Boyarski, E.; Ma, H.; Cohen, L.; Kumar, T.S.; Koenig, S. Adding Heuristics to Conflict-Based Search for Multi-Agent Path Finding. In Proceedings of the Adding Heuristics to Conflict-Based Search for Multi-Agent Path Finding: 28th International Conference on Automated Planning and Scheduling, Delft, The Netherlands, 24–29 June 2018; Volume 28, pp. 83–87. [CrossRef]
16. Li, J.; Harabor, D.; Stuckey, P.J.; Ma, H.; Koenig, S. Symmetry-breaking constraints for grid-based multi-agent path finding. In *Proceedings of the AAAI Conference on Artificial Intelligence*; AAAI Press: Washington, DC, USA, 2019; Volume 33, pp. 6087–6095. [CrossRef]
17. Atzmon, D.; Stern, R.; Felner, A.; Wagner, G.; Barták, R.; Zhou, N.F. *Robust multi-agent path finding. Proceedings of the International Symposium on Combinatorial Search*; AAAI Press: Washington, DC, USA, 2018; Volume 9, pp. 2–9. [CrossRef]
18. Phillips, M.; Likhachev, M. Sipp: Safe interval path planning for dynamic environments. In Proceedings of the 2011 IEEE International Conference on Robotics and Automation, Shanghai, China, 9–13 May 2011; pp. 5628–5635. [CrossRef]
19. Yakovlev, K.; Andreychuk, A.; Stern, R. Revisiting bounded-suboptimal safe interval path planning. In Proceedings of the International Conference on Automated Planning and Scheduling, Nancy, France, 20–26 October 2020; Volume 30, pp. 300–304. [CrossRef]
20. Takemori, T.; Tanaka, M.; Matsuno, F. Gait design for a snake robot by connecting curve segments and experimental demonstration. *IEEE Trans. Robot.* **2018**, *34*, 1384–1391. [CrossRef]
21. Laurent, F.; Schneider, M.; Scheller, C.; Watson, J.; Li, J.; Chen, Z.; Zheng, Y.; Chan, S.H.; Makhnev, K.; Svidchenko, O. Flatland competition 2020: MAPF and MARL for efficient train coordination on a grid world. In Proceedings of the NeurIPS, Competition and Demonstration Track, Virtual, 6–12 December 2020; Volume 133, pp. 275–301.
22. Lusby, R.M.; Larsen, J.; Ehrgott, M.; Ryan, D. Railway track allocation: Models and methods. *OR Spectr.* **2011**, *33*, 843–883. [CrossRef]
23. Lin, B.; Zhao, Y.; Lin, R.; Liu, C. Integrating traffic routing optimization and train formation plan using simulated annealing algorithm. *Appl. Math. Model.* **2021**, *93*, 811–830. [CrossRef]

24. Svancara, J.; Barták, R. Tackling Train Routing via Multi-agent Pathfinding and Constraint-based Scheduling. *ICAART* **2022**, 306–313.
25. He, Z.; Li, Y.; Li, H.; Xu, N. Multi-objective optimization for online train trajectory planning with moving window method. *IAENG Int. J. Comput. Sci.* **2023**, *50*, 1074–1082.
26. Zhan, Y.; Ye, M.; Zhang, R.; He, S.; Ni, S. Multi-objective optimization for through train service integrating train operation plan and type selection. *Transp. Lett.* **2023**, 1–20. [CrossRef]
27. Arsalani, P.; Reisi-Nafchi, M.; Dardashti, V.; Moslehi, G. Two new mixed-integer programming models for the integrated train formation and shipment path optimization problem. *Networks* **2023**, *81*, 359–377. [CrossRef]
28. Atzmon, D.; Diei, A.; Rave, D. Multi-train path finding. In *Proceedings of the International Symposium on Combinatorial Search*; AAAI Press: Washington, DC, USA, 2019; Volume 10, pp. 125–129. [CrossRef]
29. Chen, Z.; Li, J.; Harabor, D.; Stuckey, P.J.; Koenig, S. Multi-Train Path Finding Revisited. In *Proceedings of the International Symposium on Combinatorial Search*; AAAI Press: Washington, DC, USA, 2022; Volume 15, pp. 38–46. [CrossRef]
30. Lan, X.; Lv, X.; Liu, W.; He, Y.; Zhang, X. Research on robot global path planning based on improved A-star ant colony algorithm. In Proceedings of the 2021 IEEE 5th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), Chongqing, China, 12–14 March 2021; Volume 5, pp. 613–617. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.