IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 15 July 2022, accepted 26 July 2022, date of publication 1 August 2022, date of current version 4 August 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3195173

RESEARCH ARTICLE

Dynamic Spare Point Application Based Coordination Strategy for Multi-AGV Systems in a WIP Warehouse Environment

RUI XU[1,2], HAOJUN FENG[1], JIA LIU[3], AND WENJING HONG[2,4], (Member, IEEE)

[1]School of Business, Hohai University, Nanjing 211100, China
[2]Guangdong Provincial Key Laboratory of Brain-Inspired Intelligent Computation, Southern University of Science and Technology, Shenzhen 518055, China
[3]China Tobacco Jiangsu Industrial Company Ltd., Nanjing 210011, China
[4]Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen 518055, China

Corresponding author: Rui Xu (rxu@hhu.edu.cn)

This work was supported in part by the Guangdong Provincial Key Laboratory under Grant 2020B121201001, in part by the National Natural Science Foundation of China under Grant 62106098, and in part by the Stable Support Plan Program of Shenzhen Natural Science Fund under Grant 20200925154942002.

ABSTRACT Multi Automated Guided Vehicle (multi-AGV) systems are widely used in Work-in-Process (WIP) warehouses to improve the efficiency of material transportation. However, collisions and deadlocks between AGVs are inevitable. Many algorithms have been proposed to solve these problems, but their performance is inefficient in the WIP warehouse due to the lack of consideration of its features. In this paper, to fill the gap between current research and real-world application requirement, we construct a collision and deadlock solving model for a multi-AGV system in a WIP warehouse (CDSMWW). Then we propose a coordination strategy based on dynamic spare point application (DSPA), aiming to improve the efficiency of solving AGV collisions and deadlocks in a WIP warehouse. In this method, the AGV records its predecessor, successor, and reserved points in its local controller and can only enter the set of sequential shared points if its application for spare points is successful. Otherwise, it has to stop to avoid causing an unresolvable deadlock. If a deadlock occurs while this AGV is in the sequential shared points, the AGV moves to the spare point to resolve it. Whenever an AGV leaves the set of consecutive shared points, it releases all its spare points. We apply the algorithm to both the central and local controllers to improve the calculation efficiency. Extensive simulation results demonstrate the feasibility of the DSPA scheme under realistic WIP warehouse environment and its higher efficiency compared to previous state-of-the-art methods.

INDEX TERMS AGVs, WIP warehouse, collision and deadlock resolution, dynamic spare point application.

I. INTRODUCTION

In recent years, Automated Guided Vehicles (AGVs) have been intensively used in warehouses to reduce operational costs and increase the throughput and efficiency of material transporting [1]. However, multiple AGVs inevitably have overlapping routes during handling.

Collisions usually occur when two AGVs move in the same or opposite direction or towards the intersection of two orthogonal segments. A deadlock means that multiple AGVs can not move further, as each AGV must occupy the position currently occupied by another vehicle in the same set [2]. Achieving collision prevention and deadlock avoidance in warehouses with multiple AGVs and ensuring high efficiency is an important and realistic challenge [3].

- **Multiple obstacles:**The racks in the warehouse are seen as obstacles that the AGVs cannot directly cross, which means that the AGVs have to go around further, traverse a longer transport path, and move in fewer directions.
- **Concentrated distribution of workstations:** In a WIP warehouse environment, AGVs need to transport materials to workstations, which are usually centrally

The associate editor coordinating the review of this manuscript and approving it for publication was Jesus Felez.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

distributed, resulting in more overlapping paths between AGVs.

- **Unequal distance between identification points:** AGVs running in the warehouse rely on laser scanning to identify the identification points marked on the floor to update their position. The distances between identification points in real warehouses are not all equal.
- **Delayed update of the AGV position:** The AGV can only update its position at an identification point, which means the current position recorded when moving on the lane is still the position of the previous identification point until it reaches the next identification point.

The above characteristics of the situation bring more complex requirements and more constraints on AGV collision and deadlock prevention. WIP warehouses are employed in the automotive industry, tobacco industry and semiconductor industry, etc. Improving the flexibility and operational efficiency of multi-AGV systems in the WIP warehouse while avoiding collisions and deadlocks is a subject worth exploring.

The major collision and deadlock resolution strategies can be divided into three categories: time-window-based, Petri net, and zone control [4]. Deadlock resolutions can be divided into deadlock detection and recovery strategies, deadlock prevention strategies, and deadlock avoidance strategies [5].

The time-window-based approach uses the time window to uniformly plan the paths of all AGVs to achieve collision and deadlock-free. Smolic *et al.* [6] used a time window to find the shortest path dynamically. Since the number of running vehicles and the corresponding tasks vary with time, the proposed method extends the time window to determine the shortest path feasible, resulting in collision-and-deadlock-free movement for all vehicles. However, this approach is usually used only in topologies with unidirectional channels. Kim *et al.* [7] proposed an algorithm based on the Dijkstra shortest path method, a method suitable only for small systems due to its time-consuming nature. Zhang *et al.* [8] proposed a collision-free routing method where the route is predefined by an improved Dijkstra algorithm, and four alternative methods to avoid each type of collision are proposed. Li *et al.* [9] proposed a conflict-free routing strategy to improve the system efficiency with the loop avoidance capability and designed a dynamic scheduling strategy to find the appropriate AGV to perform the task to reduce the total traveling distance and waiting time. Yang *et al.* [10] proposed a global vision-based hierarchical planning algorithm that adds road congestion to the evaluation metrics and combines A* and time window algorithms to search for idle paths and avoid collisions. In dynamic environments, the computation of time windows is affected by many factors, such as acceleration and deceleration times of AGVs and external obstacles, making it tough to calculate time windows accurately. Inaccurate time windows and external obstacles can lead to unpredictable collisions, making these techniques unsuitable for dynamic environments [11].

The Petri net, an effective tool for modeling and analyzing deadlock problems, is often used in AGV control systems. It is a directed bipartite graph containing nodes, transitions, and arcs [12]. Banaszak and Krogh [13] proposed a centralized deadlock avoidance algorithm based on the Petri net model of concurrent job flow and dynamic resource allocation. In this algorithm, the production sequence is divided into zones with or without shared resources, and the deadlock avoidance problem is solved by defining a restriction policy. Bocewicz [14] *et al.* proposed a new format that can solve the complex problem of integrating AGV fleet allocation, routing, and scheduling in multimodal networks. In [15], the authors decouple the upper-level planning from the lower-level logistics control in an AGV system using Petri nets. In [16], the authors present an AGV scheduling and conflict-free path planning using a temporal Petri net decomposition approach. However, all Petri net-based approaches may lead to state explosion because, at each period, the robot needs to check the entire state space to determine whether it is safe to return to its current state.

Zone control is an easy and effective way to prevent collisions and deadlocks and is widely used in AGV systems [17]. In this strategy, the workspace is divided into zones with nonoverlapping coverage, and each zone can only accommodate one operating AGV to avoid collisions and deadlocks. Ho *et al.* [18] introduced a dynamic area strategy based on two procedures to prevent collisions and sustain load balancing between vehicles in different areas. Zheng *et al.* [19] proposed a multi-AGV scheduling strategy based on the shortest waiting time to optimize the running time of AGVs and presented a decentralized control mechanism to solve the collision and deadlock problems of multi-AGV systems. Qi *et al.* [20] applied the zone control strategy through a constructed AGV simulation system, examining the efficiency of two deadlock resolution policies on eight layouts to help designers choose the most appropriate one under the adapted layout.

The characteristics of the zone control approach make it suitable for a realistic plant environment, and it is widely used in the literature for realistic factories like the WIP warehouse scenario in this paper. When zones are shared between AGVs, any method based on zone control has to take action to avoid collisions and deadlocks, which affects the efficiency. However, the dynamic environment of the plant demands higher real-time speed and flexibility in the strategy.

Therefore, scholars have also been working to improve the efficiency of this strategy. In [21], a dynamic zone strategy was introduced to realize collisions-free and maintain the systems' load balance. The simulation results show that the strategy outperform the fixed zone strategy in WIP inventory, throughout, and time. In [22], a chain of reservation based coordination method(COR) was proposed. In this method, the warehouse layout is divided into squares, in which there is a queue of initial reservations. A vehicle can start moving when and only when the first reservation in the queue of the next square belongs to that vehicle, making deadlocks and

collisions impossible. It is more flexible and efficient than the dynamic zone strategy. Zhao *et al.* [11] proposed a dynamic resource reservation(DRR) method based on shared points, which uses dynamic reservation of shared resource points to change the motion state of AGVs to avoid collisions and deadlocks. Also, its traveling time is 11%-28% less than the COR according to its simulation result. In [23], a hierarchical motion coordination algorithm based on spare zones(SZH) is presented. The central controller assigns spare zones to the shared zones of AGVs before they depart, and AGVs resolve deadlocks by moving to the spare zones when they encounter deadlocks. The experiment results show that the total traveling time of SZH is 43%-76% less than that of COR. This method sacrifices workspace and is more suitable for the workspace with enough spare zones.

The study of collision and deadlock prevention for multi-AGV systems has produced many different and appropriate strategies, with limitations in practical applications due to differences in real-world task types and facility layouts. Previous researches on AGV systems in the WIP warehouse have mainly focused on optimization problems of AGV task assignment [24], optimization problems of workstation arrangement [25], scheduling studies of workstation tasks and vehicle tasks [26], and minimization of completion time, material handling, and WIP costs [27]. Few studies have been conducted specifically on the characteristics of WIP warehouses to investigate efficient collision and deadlock prevention for multi-AGV systems in WIP warehouses. However, reality has placed a higher demand on this aspect.

To fill the gap between current research and real-world application requirements, we propose a collision and deadlock solving model for a multi-AGV system in a WIP warehouse (CDSMWW) by analyzing the layout of a realistic WIP warehouse and the control characteristics of laser-guided AGVs.

The type of control for motion coordination of multi-AGV systems can be centralized or decentralized [22]. In the decentralized approach, AGVs usually get information from others. In the centralized method, all information related to the state of AGVs and the transportation system is stored in the central controller and calculated in it. From the safety perspective, the centralized method lacks of scalability, and the failure in central controller means failure of whole system. On the other end, the decentralized method leads to faster responses but has drawback in solving deadlocks and finding optimal paths[28]. The hybrid method organically combines the above two methods and avoids the disadvantages of both. In this case the central controller could be used to plan global paths and goals. But coordination motions are left for specific AGV.

Through an in-depth analysis of the CDSMWW model features, we develop a strategy based on dynamic spare point application (DSPA) and implement it on both central and local controllers, which is a hybrid control method. By comparing it with the state-of-art algorithms, we validate the effectiveness of the approach in resolving collisions and deadlocks of multi-AGV systems in a WIP warehouse. In detail, the contributions and novelties of our work include:

- We construct a more realistic model, denoted CDSMWW, for the real-world traffic control problem of multi-AGV systems in WIP warehouses.
- We define a new decentralized method of AGV travel information to eliminate the impact of delayed update of AGV positioning information on system scheduling by recording the intended points of AGVs on a map and their corresponding predecessor and successor node positions in real time.
- We design a coordination strategy based on dynamic spare point application for CDSMWW to improve the efficiency of resolving collision and deadlock. The strategy uses zone control as the main policy. It effectively integrates two mechanisms—deadlock avoidance and deadlock detection and recovery—to regulate the subsequent state of AGVs by dynamically detecting spare points to strike a balance between deadlock avoidance and efficiency enhancement.
- We simulate a WIP warehouse layout based on a real plant layout in China and compare the overall performance of DSPA with the state-of-the-art methods to demonstrate the performance efficiency of the algorithm proposed in this paper for CDSMWW.

The rest of this paper is structured as follows. **Section 2** presents the workspace and the description of the problem. **Section 3** introduces the main design of our proposed scheme. Simulation results and some discussions are presented in **Section 4**. Finally, future work is discussed in **Section 5**, and conclusions are reviewed.

II. WORKSPACE DESCRIPTION AND PROBLEM ANALYSIS

A. WORKSPACE DESCRIPTION

The problem considered in this paper is related to coordinating a fleet of A vehicles delivering materials from the WIP warehouse to workstations. The WIP warehouse consists of individual storage spaces. The vehicles can be denoted as $R_i = \{r_i : 1 \leq i \leq A\}$. The environment is completely known and constructed. To simplify the model, we abstract the model as an undirected graph, as shown in Fig. 1. The workstations, material storage locations, parking places, and guide points are abstracted as points of the graph, and the adjacent points are connected by virtual edges, indicating a passable lane between them. We use a set of adjacent points to represent the guide path of an AGV. A point can hold an AGV, and each point is used as an identification point in reality. All points are categorized as parking points, material storage points, workstation points, or guide points, and each is an identification point set on the path to facilitate the positioning of the AGV. The AGVs are precisely positioned by laser scanning of identification points. After receiving the task, the AGV starts from the parking point, picks up the material at the warehouse point, delivers it to the workstation, and returns to the nearest parking point from the workstation.

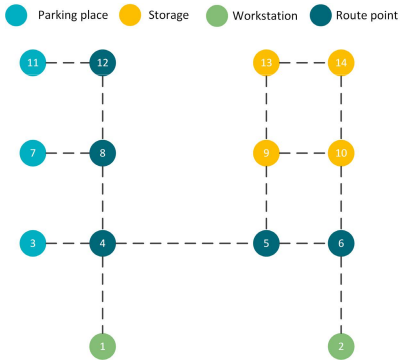

FIGURE 1. Workspace abstracted from the WIP warehouse scenario.

The process of AGV movement involves repeatedly passing from one identification point through an edge to another identification point, and the AGV cannot stop on the edge. The diagram must contain at least one parking place, one workstation, one storage place, and one guide point, so the set of all points in the diagram G is $V = \{v_x, 1 \leq x \leq n\}$. At the beginning, each AGV is in its parking place. The set of parking places is denoted as $V_p = \{v_x : v_x \in V\}$. This approach means that the number of parking places must equal or exceed the number of AGVs to optimize the utilization efficiency of AGV while considering economic reasons and avoiding more congestion. And since there must be a guide point, a storage place, and a workstation in the workspace, there must be three points that cannot be parking places. The set of material storage points is denoted $V_m = \{v_x : v_x \in V, v_x \notin V_p, V_w, V_g\}$, the set of workstations is $V_w = \{v_x : v_x \in V, v_x \notin V_p, V_m, V_g\}$, and the set of guide points is $V_g = \{v_x : v_x \in V, v_x \notin V_p, V_w, V_m\}$. This notation indicates that the parking points, guide points, material storage points, and workstations are independent of each other, and the number of workstations must be more than the number of AGVs. In this paper, the task-guided paths of AGVs are generated by the central controller that uses the Dijkstra algorithm.

Definition 1: The path of AGV r_i performing task t_j is denoted as $P_i^j = \{v_s^i, \ldots, v_m^j, \ldots, v_w^j \ldots, v_d^i\}$, where v_s^i is the origin parking point, v_m^j is the material pick-up point of the task, v_w^j is the workstation where AGV needs to deliver the materials of the task, and v_d^i is the parking point to which the AGV finally returns. The ellipsis in the middle of these symbols represents the guide point v_g^j on the path.

Definition 2: The residual transport route $\Pi_i = \{v_x, \ldots, v_d\}$ of AGV r_i performing task represents the residual sequence of points that remain to be visited by r_i before finishing the task.

B. PROBLEM ANALYSIS

When an AGV performs a task in a WIP warehouse, its initially generated shortest path does not consider the path information of other AGVs, which may result in path overlap. If no action is taken, using such paths will lead to collision or deadlock between vehicles. A collision occurs when two AGVs move to the same position simultaneously. A deadlock is a state in which the AGVs do not collide but neither can move.

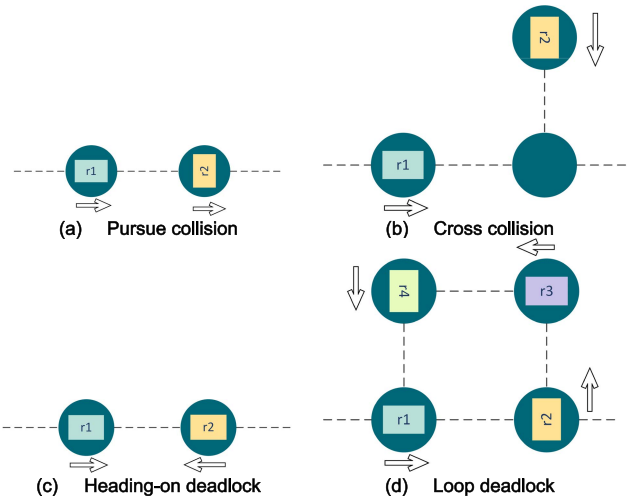

FIGURE 2. Different types of collision and deadlock.

There are two kinds of collisions and two kinds of deadlocks in the warehouse, as shown in Fig. 2.

- **Pursue collision:** An AGV wants to move to an identification point already occupied by another AGV.
- **Cross collision:**There are two AGVs in orthogonal directions that want to occupy an identification point simultaneously.
- **Heading-on deadlock:** Two AGVs are traveling in opposite directions. Their positions are adjacent to each other, and both want to move to the identification point currently occupied by the other.
- **Loop deadlock:**This is a situation where the directions of the related vehicles form a closed loop.

In addition, compared to the grid maps commonly used in the literature, realistic WIP warehouses have the following characteristics: multiple obstacles, concentrated distribution of workstations, unequal distances of identification points, and delayed update of AGV positions. Fig. 3 shows a simplified scenario of a WIP warehouse with a production line. The brown blocks are the material storages, and the AGV cannot pass across the racks; the green items are the assembly line stations arranged around the production line and concentrated at the bottom of the graph. The laser-guided AGV scans the internal environment of the factory by laser and sets the identification points, and the precise positioning of the AGV is performed by simultaneous location and mapping(SLAM) during the driving process [29]. This scheme is flexible, but the identification points generated by the mapping have the problems of distance differences and untimely updates.

1) LONGER DISTANCES AND MORE OVERLAPS DUE TO WIP WAREHOUSE LAYOUT

In the WIP warehouse, because the storage places and the assembly line are distributed in different areas, and the

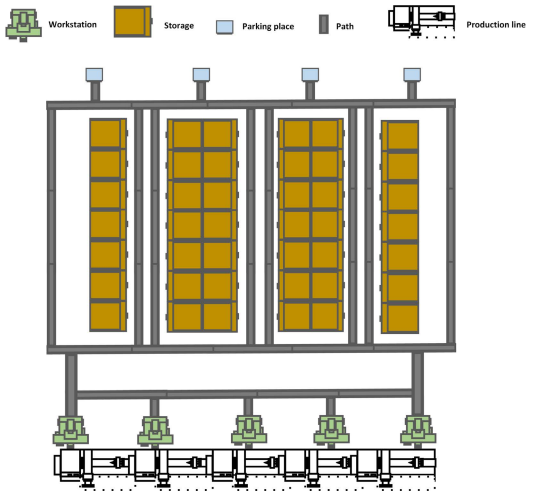

FIGURE 3. Structure of a WIP warehouse with a production line.

obstruction of the racks makes the AGV go indirectly, the path of the AGV performing the task is usually longer than that in a grid-type map. At the same time, since the task of AGVs is to transport materials to workstations, the centralized distribution of workstations on the assembly line increases the possibility of path overlap between AGVs. Both types of problems increase the probability of conflicts or deadlocks. If the algorithm only applies the idea of waiting techniques to avoid deadlock (as in [11]), the excessively long distances of shared points will make the waiting time very long. If we only adopt an approach like [23] to solve deadlocks, AGVs are likely to spend much time waiting for the successful application of spare points. Although both methods can solve deadlocks, they are not efficient.

2) SAFETY AND EFFICIENCY ISSUES CAUSED BY ASYNCHRONOUS UPDATING OF AGV POSITIONS

The identification point is set by laser scanning the internal environment of the factory in the WIP warehouse, so the identification point is not distributed at equal intervals as in a grid-type map. This means that the time the AGV arrives at the identification point and the time of the position update are asynchronous. The AGV typically records its current position and the next position. Since the AGV can only update its position through SLAM correction after reaching the identification point, the information obtained from other AGVs is not real-time, causing potential safety and efficiency problems.

a: SECURITY PROBLEM

As shown in Fig. 4-a, due to the different distances between identification points, the AGV position update is not synchronized. The AGV may move forward to the next identification point, believing no other AGV is at that next identification point, but there may be an AGV there that has not yet updated its position. Thus, two AGVs may collide and cause a security problem.

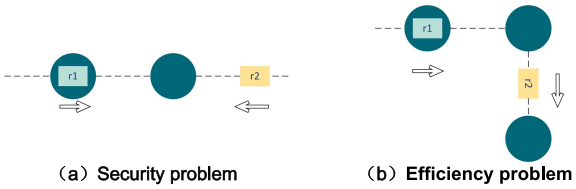

FIGURE 4. Security and efficiency problems.

b: EFFICIENCY PROBLEM

As shown in Fig. 4-b, due to the delay of AGV position updating, an AGV may wrongly believe that there are other AGVs at the next identification point and stay in the same place waiting. It will not advance to the next identification point as soon as it could, thus reducing the efficiency.

This problem is typically solved by a centralized approach, in which the AGV only records the information of the current point and the next point and marks a point on the map with which AGV belongs (occupied or reserved). Other AGVs scan the next point to determine whether it belongs to other AGVs. In this paper, since we are using a decentralized coordination method where AGVs obtain the location information of other AGVs instead of the map information, we must find another way to avoid this problem.

Our goal is to achieve efficient collision and deadlock solutions for multi-AGV systems in a WIP warehouse. Therefore, we will make the following assumptions in our algorithm.

Assumption 1: All Active AGVs in the System Cannot Replan Their Paths:

In most control algorithms, the guided paths of AGVs are assumed to be constant because the path replanning may lead to new collision and deadlock problems, which do not fundamentally solve the collision and deadlock problems [8] and increase the complexity of the algorithm.

Assumption 2: Vehicles Can Obtain Travel Information of other AGVs Through the Communication System:

When using the decentralized approach for motion coordination of AGVs, the AGVs need to exchange information with other AGVs through the communication system, such as the residual paths, to calculate the relevant parameters and make motion decisions.

Assumption 3: The Vehicle Cannot Stop Its Movement Between Points or Temporarily Change the Direction of Movement on the Lane Between Points:

Only when the vehicle arrives at the location of the identification point can it decide the movement state of the next movement. The arrival place is an identification point that is a parking point, workstation point, storage point, or guide point. On the way to its next arrival point, the vehicle follows the lane chosen when it started its movement without changing its travel speed or planned route.

Assumption 4: The AGV Speed is Constant During the Movement, Ignoring the Time Required for Acceleration and Deceleration of AGV. The Time for Rack Loading or Unloading and AGV Turning is Ignored:

To ensure the safety of material transportation, AGV generally runs at a slow and uniform speed. Since the speed is low, the acceleration and deceleration time can be ignored. On the other hand, in most of the coordinate algorithms, the loading and unloading time are negligible [11], [22], [23]. Since considering the loading and unloading time is actually equivalent to extending the distances from the storage places or workstations to the guide points, the loading and unloading don't have significant impact in the comparative results. Hence, to concentrate on the movement, we neglect the time for rack loading or unloading and AGV turning.

III. COORDINATION STRATEGY

In this section, we propose a traffic control strategy based on a dynamic spare point application to solve collisions and deadlocks. First, the calculation and update of shared points and spare points are introduced. Then, the motion coordination strategy among vehicles to prevent collisions and deadlocks is detailed, and deadlock detection and recovery methods are used to solve any deadlock problem encountered by the system. Then, we analyze the effectiveness of the proposed algorithm for collision and deadlock resolution. Finally, the implementation of the algorithm on the central and local controllers of the algorithm is presented.

A. SHARED POINT GENERATION

When an AGV performs a task, it will reserve, occupy, and release each identification point on the path. Each AGV can only occupy one identification point at a time. Obviously, collisions and deadlocks between AGVs always occur at the identification points where their residual paths overlap, which we call shared points. When AGVs have one shared point, they may collide. Heading-on and loop deadlocks occur only when vehicles have consecutive shared points.

Definition 3: For an active AGV r_i, CP^i consists of an ordered sequence of points shared with other AGVs, which means the points that may cause collisions between r_i and another AGV. The sequence can be denoted as $CP^i = \{v_x : v_x \in \Pi_i, v_x \in \Pi_j, j \neq i\}$, where j is the number of another AGV. Two or more adjacent points in CP^i are called sequential shared points. The sequential shared zone of r_i is a subset of CP^i, which can be denoted as SCP^i.

Definition 4: For AGV r_i, its sequential shared points can be denoted as $SCP^i = \{v_p : D(v_p, v_q) \neq 0, v_p \in CP^i, v_q \in CP^i\}$, where v_q and v_p are the shared *points* of r_i, and D is the adjacency matrix of undirected graph G.

Conflicts and deadlocks can occur when AGVs share certain points with other AGVs. To solve this problem dynamically, it is necessary to record each AGV's CP^i and SCP^i in real time. After an AGV reaches an identification point, the residual path of the AGV is updated. At the same time, each AGV updates CP^i and SCP^i according to Π_i.

Example 1: The generation of shared points is shown in Fig. 5. In this case, the residual transport routes of r_1, r_2, r_3 can be represented as $\Pi_1 = \{7, 15, 23\}$, $\Pi_2 = \{19, 11, 12, 13, 14, 15, 16\}$, $\Pi_3 = \{13, 12, 11, 10, 9\}$, respectively.

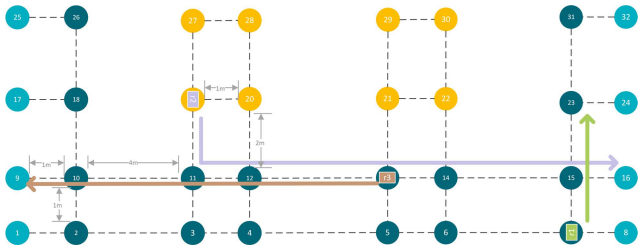

FIGURE 5. Example of generation of shared points set.

Hence, the shared points are denoted as $CP^1 = \{15\}$, $CP^2 = \{11, 12, 13, 15\}$, $CP^3 = \{13, 12, 11\}$, respectively. And the sequential shared points of r_1, r_2, and r_3 are represented as $SCP^1 = \emptyset$, $SCP^2 = \{11, 12, 13\}$, and $SCP^3 = \{13, 12, 11\}$, respectively.

B. SPARE POINT APPLICATION

In general, a collision can be prevented as long as two vehicles do not occupy the same identification point at the same moment. When a deadlock occurs, each deadlocked vehicle could consider temporarily leaving its current identification point or replanning the path to solve the deadlock. Under the premise of no path replanning, the main idea of deadlock resolution is to ensure enough free area for the deadlocked vehicles to temporarily leave to resolve the deadlock [23].

In this paper, we propose a collision and deadlock resolution mechanism based on a dynamic spare point application, which integrates deadlock avoidance and deadlock detection and recovery strategies. When an AGV is about to enter a sequential shared point set, and there is an identification point occupied by other AGVs in the shared point set, it has to apply for spare points to the central controller first and then decide whether it can move forward. If the AGV fails to apply for spare points, it can only stay at the current point to avoid deadlock. If the application is successful, it will be able to enter the shared point set. The central controller detects deadlocks in real time and moves the AGV with the spare point to the corresponding spare point to resolve the deadlocks, and the AGV needs to remove the extra spare points when it runs inside the sequential shared points. When an AGV leaves its sequential shared point, it will release its spare points and reapply for them the next time when needed. The dynamic application of spare points ensures that AGVs can handle tasks flexibly and efficiently. To describe the mechanism in detail, we first define some concepts as follows.

Definition 5: For a point v_x, its free points can be defined as a set of points linked to v_x but not belonging to the working vehicles' residential path. The free point set of v_x can be denoted as $FP(n_x)$.

Since deadlocks can only be triggered inside sequential shared points, we only assign spare points to the sequential shared points of the vehicle.

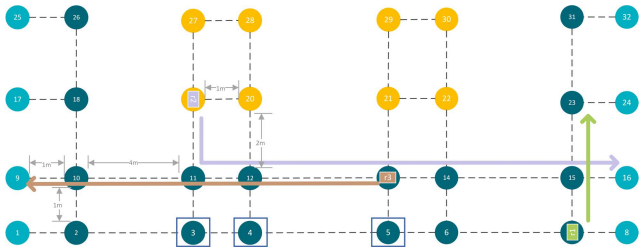

FIGURE 6. Example of spare point application. Points surrounded by squares are spare points.

Definition 6: For a shared point v_x, its spare point is one of the nearest free points, which ensures the resolution of a deadlocked vehicle by a moving action. The spare point for v_x is denoted as $SP(v_x)$. Hence, the spare points of r_i is defined as $SP^i = \{(v_x, SP(v_x)), v_x \in SCP^i\}$.

The central controller first obtains the set of shared points, sequential shared points, and residual path of the vehicle and then assigns the spare points.

Example 2: The application process for spare points is shown in Fig. 6. Since we have already calculated the shared point set and sequential shared point set of AGV in Example 1, we will not repeat that process here. Taking the AGV r_2 as an example, $SCP^2 = \{11, 12, 13\}$. For point 11, its only free point is point 3, so its spare point is point 3. The free points of point 12 are points 4 and 20, but point 4 is chosen as the spare point because it is closer. Similarly, point 5 is chosen as the spare point of point 13. Then, each sequential shared point has its spare point, which indicates that the application of spare points is successful and $SP^2 = \{11, 3; 12, 4; 13, 5\}$.

C. DYNAMIC SPARE POINT APPLICATION BASED STRATEGY

Before introducing the collision and deadlock resolution strategy to solve the security and efficiency problems caused by the asynchronous and delayed position updates of AGVs, we first define the motion state. We store the traveling information of AGVs using a decentralized mechanism.

Definition 7: The transport state of the vehicle r_i in the system can be idle ($SA^i = 0$), moving ($SA^i = 1$), and waiting ($SA^i = 2$). The idle state implies that a vehicle has no mission to execute. The moving state refers to a vehicle on its way to the next reserved point. The waiting state is a state where a vehicle has to stop at its current point. In addition, we set a Boolean variable F^i to indicate whether the vehicle moves along its shared points while having sufficient spare points: $F^i = 1$ means it is traveling in the shared points while $F^i = 0$ means it is not.

Definition 8: For an active AGV r_i, its traveling information is denoted as $I^i = \{v_c^i, v_n^i, v_r^i\}$, where v_c^i represents the point where r_i is currently located or the point the AGV left last time; v_n^i represents the next point to be visited by r_i; and v_r^i represents the point that the AGV has reserved.

It is a resource request process that AGV running in the workspace. The AGV applies to the system to reserve the next identification point, and if the application is successful, the AGV will reserve this point and move to the next identification point. In a realistic warehouse, when the AGV moves to the next reserved point, it cannot reach the next point instantly, and there will be a movement process in the lane. In this process, this paper uses v_c^i to denote the last point the AGV left, v_n^i to denote the next point to be visited, and v_r^i is equal to v_n^i in this situation. If it fails to reserve the next point, it must stay at the current point, and at this time, v_c^i denotes the point where the AGV is currently located, and v_r^i is equal to v_c^i. The AGV determines whether it can move to the next point based on whether any other AGV has reserved the point rather than on the current position of other AGVs. So far, the security and efficiency problems caused by the asynchronous and delayed update of the actual position of the AGV are solved under the decentralized mechanism.

Definition 9: If $v_n^i = v_c^j$ and $v_n^j = v_c^i$, then a heading on deadlock occurs between r_i and r_j. A loop deadlock occurs if there exists a set of vehicles $R = \{r_i, r_j, r_p, \ldots, r_q\}$ such that $v_n^i = v_c^j, v_n^j = v_c^p, \ldots, v_n^q = v_c^i$.

The method proposed in this paper resolves collisions and deadlocks based on the reservation positions of other AGVs and the dynamic application of spare points. The AGVs determine the motion state of the AGVs at each identification point based on the following conditions.

Condition 1: $v_n^i \notin SCP^i$ and $v_n^i \notin \left\{v_r^j, j \neq i\right\}$

This condition indicates that the next point of the AGV does not belong to a sequential set of shared points, and the next point is not reserved by another AGV.

Condition 2: $v_n^i \in SCP^i$ but $\forall v_x \in SCP^i, v_x \notin \left\{v_r^j, j \neq i, F^j = 0\right\}$ and $v_n^i \notin \left\{v_r^j, j \neq i\right\}$

This condition indicates that the next point of the AGV belongs to a sequential shared point set, but for all points within the shared point set, none of them are reserved by other AGVs without spare points, and the next point is not reserved by any other AGVs.

Condition 3: $v_n^i \in SCP^i$ and $\exists v_x \in SCP^i, v_x \in \left\{v_r^j, j \neq i, F^j = 0\right\}$ but $SP^i \neq \emptyset$ and $v_n^i \notin \left\{v_r^j, j \neq i\right\}$

This condition indicates that the next point of the AGV belongs to a sequential set of shared points. There are also points within the set of shared points occupied by other AGVs without spare points, but the set of spare points of this AGV is not empty (spare points are successfully allocated), and the next point is not reserved by all other AGVs.

After reaching an identification point other than the destination, the AGV first updates v_c^i and v_n^i. When the vehicle meets one of the above conditions, it can successfully request the next point and reserve it, which means $SA^i = 1$ and $v_r^i = v_n^i$. When the AGV evaluates **Condition 3** and $F^i = 0$, it must first apply to the central controller for spare points. If the application fails, then $SP^i = \emptyset$; if the application is successful, then $F^i = 1$, which means that the AGV already

has spare points. The AGV then evaluates **Condition 3** and does not need to reapply for the spare point but removes the extra spare points whenever it reaches a point. In addition, $F^i = 1$ indicates that the AGV has the ability to resolve deadlocks, which is why some of the judgment conditions of **Condition 2** and **Condition 3** do not take the AGV with $F^i = 1$ into account. If the AGV satisfies one of the first two conditions later, then there is no possibility of deadlock when the AGV moves to the next point, so $F^i = 0$ and $SP^i = \emptyset$.

When the AGV fails to meet the above conditions, then $SA^i = 2$ to avoid collisions and deadlocks. The central controller will detect a deadlock for the halted AGV in real time. If there is no deadlock, the vehicle stops and waits, and $v_r^i = v_c^i$. If a deadlock occurs and the corresponding spare point is not occupied by another AGV, the vehicle with $F^i = 1$ will move to the corresponding spare point, and it will set its next point v_n^i as the corresponding spare point and add it to the residual path, while the other vehicles involved in the deadlock can move on. By incorporating the two ideas of AGV deadlock avoidance and AGV deadlock detection and recovery, we solve the problem of longer AGV paths and more overlaps, making the AGVs' performance more efficient.

Theorem 1: The proposed method based on dynamic spare point application ensures that the motion of multiple AGVs is collision-free and deadlock-free in the system.

Proof: As mentioned above, the necessity for an AGV to reserve and move to the next point is that the next point is not reserved by another AGV, and if the vehicle does not meet the condition, it will stop temporarily. What's more, all AGVs are sequential to determine whether they can reserve the next point, rather than parallel. Therefore, two AGVs will not reserve and occupy the same point simultaneously, so collisions between AGVs are not possible. Meanwhile, the necessary condition for deadlock is that there are sequential sets of shared points between AGVs. However, we apply for spare points for all consecutive shared points of AGVs before entering the set of shared points so that the deadlock can be solved by moving the AGV to the spare point. If the AGV fails in its application for spare points, it is prohibited from entering the sequential shared point set; this prohibition avoids deadlock. Therefore, deadlock between vehicles is not possible. □

To understand more intuitively how the DSPA algorithm prevents collisions and resolves deadlocks, we describe the process in detail in Example 3 and display it in Fig. 7.

Example 3: There are three vehicles in the system, which can be represented as $R = \{r_1, r_2, r_3\}$. Vehicle r_1 is already on its way, while r_2, r_3 are preparing to carry out new tasks. Vehicle r_1 is located at point 7 right now, and the parking points of r_2, r_3 are points 17 and 32, respectively. The path of each vehicle and its residual path can be represented as follows. $\Pi_1 = P_1 = \{7, 6, 5, 13, 21\}$, $\Pi_2 = P_2 = \{17, 18, 10, 11, 12, 13, 14, 22\}$, $\Pi_3 = P_3 = \{32, 31, 23, 15, 14, 13, 12, 11, 19\}$. The set of shared points and sequential shared points are $CP^1 = \{13\}$, $SCP^1 = \emptyset$, $SCP^1 = \emptyset$, $SCP^2 = CP^2 = \{11, 12, 13, 14\}$, $SCP^3 = CP^3 = \{14, 13, 12, 11\}$. Each vehicle moves at a speed of 1m/s.

At the beginning, all three vehicles can satisfy **Condition 1**, so the vehicles move towards the next point. At time 5 s, r_3 reaches point 15, and its next point, point 14, belongs to SCP^3. At this time, the traveling information of r_1 and r_2 is that $I^1 = \{5, 13, 13\}$, $I^2 = \{10, 11, 11\}$, $\Pi_1 = \{13, 21\}$, $\Pi_2 = \{11, 12, 13, 14, 22\}$. Since both point 11 and point 13 are reserved, and they both belong to SCP^3, r_3 does not satisfy Conditions 1 and 2; therefore, it needs to test **Condition 3**. It can be seen that points 3, 4, 5, and 6 are not reserved and do not belong to any of the residual paths of the AGV, indicating that they can be assigned to points 11, 12, 13, and 14 as spare points, so $SP^3 = \{11, 3; 12, 4; 13, 5; 14, 6\}$. The application for spare points is successful and **Condition 3** is satisfied, so $I^3 = \{15, 14, 14\}$, $\Pi_3 = \{14, 13, 12, 11, 19\}$, and $F^3 = 1$. At time 7 s, r_2 reaches point 11, and although the point 14 belongs to SCP^2 and is reserved by r_3, r_2 still satisfies **Condition 2** since $F^3 = 1$, so $I^2 = \{11, 12, 12\}$. At the 8 s mark, r_1 reaches point 21 and completes its task, so $SA^1 = 0$. Other AGVs travel forward normally.

At time 9 s, r_3 arrives at point 14, while the traveling information of r_2 is $I^2 = \{12, 13, 13\}$ and r_2 is traveling between points 12 and 13. At this time, r_3 does not satisfy any of the motion conditions, nor does it satisfy the deadlock conditions, so it stops at point 14, and $SA^3 = 2$. At time 12 s, r_2 reaches point 13 while r_3 is still at point 14, thus $v_n^2 = v_c^3$ and $v_n^3 = v_c^2$ and deadlock occurs. Therefore, r_3 has to move to the spare point and add the spare point to the residual path. Then, r_2 moves forward and $I^2 = \{13, 14, 14\}$, $I^3 = \{14, 6, 6\}$, $\Pi_3 = \{6, 14, 13, 12, 11, 19\}$. At time 13 s, since $I^2 = \{14, 22, 22\}$, r_3 satisfies **Condition 1** after updating its traveling information, and it will return to point 14. Then, set $I^3 = \{6, 14, 14\}$, $F^3 = 0$, $SP^3 = \emptyset$, and the deadlock is resolved successfully. At 15 s, r_2 reaches its destination, completes its task and $I^3 = \{13, 12, 12\}$ at the same time. Thereafter r_3 will advance without incident and arrive at point 19 at time 22 s to complete the task.

To demonstrate vividly the efficiency of the motion coordination strategy proposed in this paper, we compared our method's efficiency with those of the current state-of-the-art strategies DRR and SZH in this scenario. The details of the DRR and SZH strategies can be found in [11], [23]. The results are shown in Table 1.

As can be seen from Table 1, the Sum of completion time, the timespan and average waiting time of DSPA are the smallest among the three methods, and the average waiting time is reduced by up to 62% compared to the other two strategies. This shows the superiority and efficiency of this paper's proposed method. The total traveling distance is slightly shorter for DRR than for SZH and DSPA because the AGV has to move to the spare point and back in the SZH and DSPA methods.

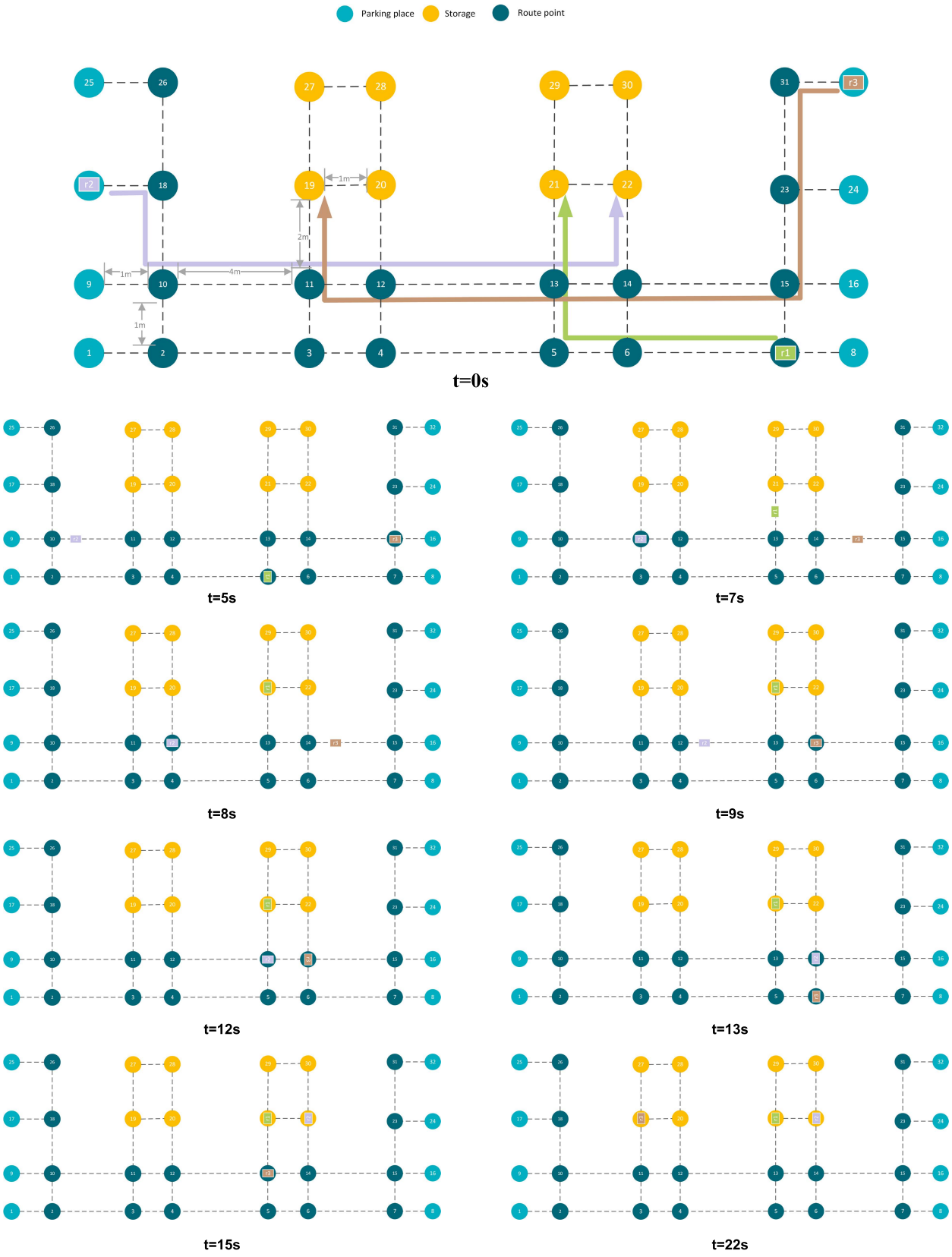

FIGURE 7. Example of coordination based on the DSPA.

TABLE 1. Comparative results of example 3.

Strategy	DRR			SZH			DSPA		
	r_1	r_2	r_3	r_1	r_2	r_3	r_1	r_2	r_3
Completion time	8	15	25	8	17	24	8	15	22
Waiting time	0	0	8	0	2	5	0	0	3
Sum of completion time		48			49			45	
Timespan		25			24			22	
Average waiting time		2.67			2.33			1	
Total traveling distance		40			42			42	

Note. Specific explanations of indicators can be seen in Section Ⅳ-B.

D. ALGORITHM IMPLEMENTATION

The above-mentioned AGV collision and deadlock resolution method can be applied to the traffic control system. In this section, we elaborate on the control algorithm for both central and local controllers.

The central controller is mainly responsible for assigning tasks to idle AGVs, calculating the shortest path and other related information, and then sending the initialized information to AGVs. When an AGV sends an application for spare points, the central controller is responsible for allocating spare points. Also, the central controller detects and solves AGV deadlock problems in real time. The local controller is responsible for updating the travel information when the AGV is performing the task, evaluating the motion conditions, and controlling the movements of the AGV.

Algorithm 1 Task Dispatching of the Central Controller

```
1 while system is in operating state do
2    Read input data and create task list T
3    for each task in the task list T do
4         Dispatch task t_j to idle AGV r_i
5         Find the nearest parking zone for r_i
6         Find the shortest route path P_i^j
7         Create the residual route Π_i
8         Calculate the shared points set CP^i and the
   sequential shared points set SCP^i
9         Send P_i, Π_i, CP^i, SCP^i to r_i
10   end
11 end
```

The task assignment method of the central controller is described in detail in**Algorithm 1**. In this algorithm, T is an ordered list of tasks consisting of picking points and delivery workstations for the tasks. Before assigning the task to an AGV, the closest parking point to the workstation of the task is assigned to the AGV (**line 5**). Then relevant information such as the shortest path is calculated and transmitted to the local controller of the AGV (**lines 6-9**). The central controller will repeat this process until all tasks in the task list are done.

Algorithm 2 Control Policy of the Local Controller

```
1  while system is in operating state do
2     Read the path P_i, Π_i, CP^i, SCP^i from central
   controller
3     repeat
4           SA^i = 2
5        if r_i in point then
6          Update I^i, Π_i, CP^i, SCP^i, SP^i
7            if v_n^i ∉ {v_r^j, j ≠ i} then
8               if v_n^i ∉ SCP^i then
9                  SA^i = 1, F^i = 0, SP^i = ∅, update I^i
10              else if v_n^i ∈ SCP^i and ∀v_x ∈ SCP^i,
   v_x ∉ {v_r^j, j ≠ i, ,F^j = 0} then
11                 SA^i = 1, F^i = 0, SP^i = ∅, update I^i
12              else
13                 if F^i = 1
14                    removal SP^i (v_c^i)
15           else
16                    Apply for spare points SP^i
17                 end
18                 if v_n^i ∈ SCP^i and ∃v_x ∈
   SCP^i, v_x ∈ {v_r^j, j ≠ i, F^j = 0} and SP^i ≠ ∅ then
19                    SA^i = 1, F^i = 1, update I^i
20                 end
21              end
22           else
23              SA^i = 1
24           end
25        end
26     until v_c^i = v_d^i
27     Remove t_i from T
28  end
```

The method applied to the local controller control is described in **Algorithm 2**. When the AGV receives the task and related information from the central controller, it starts the task execution state and runs **lines 4 - 23** repeatedly until finished with the task. The default motion state of the AGV is stop (**line 4**). When the AGV is in the position of the identification point, it updates the information and uses the strategy to determine the next state (**lines 5 - 20**). If the AGV is on an edge, it moves forward (**line 21**). When making a state decision, if the AGV satisfies **Condition 1** and **Condition 2**, the AGV can be reserved and move to the next point (**lines 8 - 11**). If neither of them is satisfied, the AGV will apply for spare points first (**lines 13 -17**, see **Algorithm 4** for details) and then evaluate **Condition 3** (**lines 18 - 20**). When the AGV reaches the final target point, namely back at the parking point, the task is completed and removed from the task list (**line 27**).

Algorithm 3 describes how the central controller resolves deadlocks of AGVs. The central controller reads the relevant traveling information of all AGVs and performs deadlock

Algorithm 3 Deadlock Resolution of the Central Controller

```
1   while system is in operating state do
2     Read AGV information from R
3     for each AGV r_i in R do
4       if SA^i = 2
5         if v_n^i = v_c^j and v_n^j = v_c^i then
6           if F^i = 1 then
7             r_i moves to SP^i(v_c^i)
8             r_j moves to v_n^j
9           else
10            r_j moves to SP^j(v_c^j)
11            r_i moves to v_n^i
12          end
13        else if v_n^j = v_c^i then
14          if loop deadlock, then
15            Select an AGV m for which F^m = 1
16            r_m moves to SP^m(v_c^m)
17            Other AGVs move forward
18          end
19        end
20      end
21    end
22  end
```

detection for AGVs that are judged to be stopped by the local controller strategy. If there is a heading-on deadlock, an AGV with a spare point is selected to move to the corresponding spare point. Then another AGV is allowed to move to the next point, and the updated traveling information is sent to the local controller of the involved AGVs (**lines 5 - 12**). If some AGVs form a cyclic deadlock, the central controller will select an AGV with a spare point to move to the corresponding spare point (allowing the other AGVs to move to their next points), update its traveling information, and send the update to the local controller of the AGV (**lines 13 - 17**).

Algorithm 4 Allocation of the Spare Points of r_i

```
1  Read SCP^i from r_i
2  for each v_x^i ∈ SCP^i do
3        if FP(v_x^i) ≠ ∅ then
4           Select one of the nearest FP(v_x^i) and add it
   to SP^i(v_x^i)
5        else
6           return SP^i = ∅
7        end
8  end
```

Algorithm 4 offers the pseudocode for the central controller to allocate spare points. The central controller searches for free points one by one for consecutive shared points, and if there exist free points, it selects the nearest one as the spare point for the corresponding point and adds it to the spare point set of the AGV (**lines 3 - 4**). If there is a point where no free point exists, namely, the free point set is empty, the empty spare point set is returned, indicating that the allocation failed (**lines 5 - 6**).

The computational complexity analysis of the proposed motion coordination algorithm involves the complexity of path planning and operational control. In this paper, the Dijkstra algorithm is used to generate the paths of vehicles with a complexity of $O(n^2)$, where n is the number of vertices of the graph. The major part of motion coordination includes the calculation of shared points and spare point allocation. To calculate the shared points of vehicles, vehicles need to compare their residual paths with those of other vehicles. Therefore, the computation time of the proposed coordination algorithm is linearly related to the number of AGVs, which means the complexity is O(A), where A is the number of AGVs.

IV. SIMULATION

We built a simulation program to implement the motion coordination in the AGV system so that the operating AGVs in the system can avoid collisions and deadlocks. We used MATLAB 2020a to code this program on a computer with an AMD-4600U CPU and 16GB RAM.

The collision and deadlock solutions based on SZH and DRR are the state-of-the-art traffic control algorithms available and can be successfully applied on a grid map. However, no simulation has been done in a WIP warehouse. In this paper, extensive simulations are conducted to verify the effectiveness of the DSPA approach in a WIP warehouse. The effectiveness of the DSPA method is demonstrated by comparing it with DRR and SZH algorithms applied in a WIP warehouse. Thus, we first overview the performance of the DSPA algorithm for different numbers of AGVs. Then the impact of the number of tasks and the workspace scale on the efficiency of collision and deadlock prevention is illustrated.

A. SIMULATION SETTING

The experiment layout with bidirectional lanes is shown in Fig. 8. Based on realistic manufacturing plants, we set a total of 892 vertices in the diagram, including 78 parking points, 30 workstation points, and 12 rows of storage racks with a total of 576 storage points. The rest of the points are guide points. In order to decentralize the distribution of AGVs and improve the operation efficiency of the warehouse, parking points are generally placed on the left and right sides of the warehouse as well as on the upper side, and the number of parking points is at least the number of AGVs. The bottom side of the figure is the assembly line, and workstations are located on the assembly line. What is in the center is the WIP warehouse. Since the assembly line is a flow line, there are only two entrances and exits between the inventory and the assembly line.

When performing a task, the AGV starts from the initial parking point, picks up material from a storage point, delivers the material to the designated workstation point, and finally returns to the nearest parking point. The speed of AGVs is fixed to 1m/s.

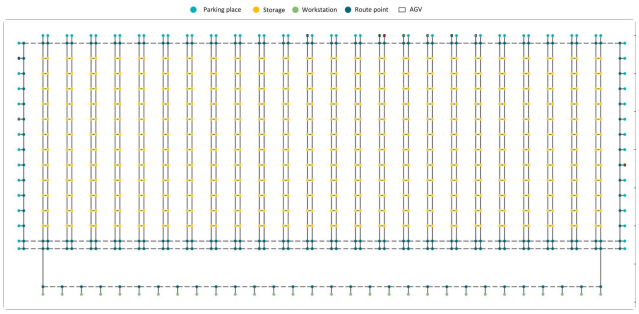

FIGURE 8. Layout for the simulation.

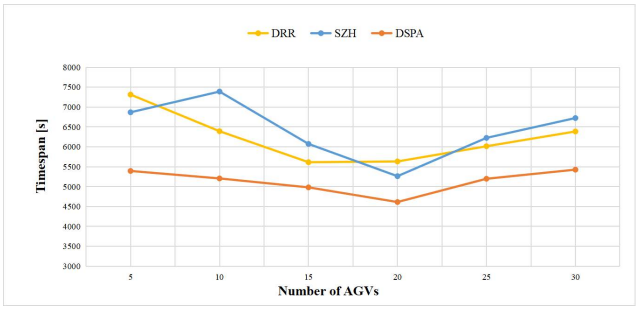

FIGURE 9. Timespan for different numbers of vehicles.

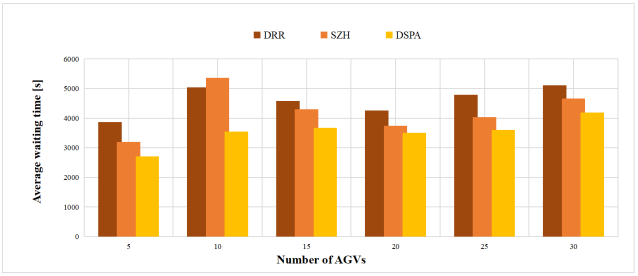

FIGURE 10. Average waiting time for different numbers of vehicles.

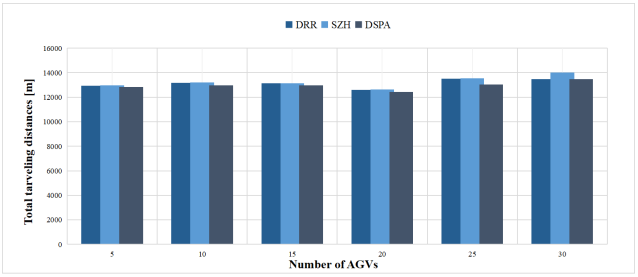

FIGURE 11. Total traveling distances for different numbers of vehicles.

B. PERFORMING METRICS

In the simulation, we evaluate the effectiveness of the traffic control algorithm proposed in this paper with three major metrics: timespan, average waiting time, and total traveling distance.

- Timespan T : $T = T_e - T_s$, where T_s is the starting time of the first mission and T_e is the time that all the vehicles stop at the parking zones.
- Average waiting time T_w: $T_w = \frac{1}{n}\sum_i^m t_w^i$, where m is the number of all the tasks, t_w^i is the time the AGV is stopped to avoid deadlocks and collisions for performing the i^{th} task, and n is the number of AGVs.
- Total traveling distance D : $D = \sum_i^m d_i$ where m is the number of all the tasks, and d_i is the moving distance of the vehicle to perform the i^{th} task.

C. RESULTS

1) PERFORMANCE OVERVIEW

To validate the effectiveness of DSPA, we compare the timespan, average waiting time, and total traveling distance of DSPA with the SZH and DRR methods by simulation. The number of tasks is set to 60. The number of vehicles is set to vary from 5 to 30, and other parameters are left the same as the default values in **Section IV-A.**

The relationship between the number of vehicles and the timespan is shown in Fig. 9. The timespan of DSPA is reduced by 11%-26% and 12%-29% compared to SZH and DRR, respectively. There are several reasons for this result. Compared with DRR, DSPA reduces the long waiting time for vehicles because vehicles can move into the shared point under certain conditions due to the spare point application. Compared with SZH, as the application for the spare points is dynamic, vehicles do not have to wait until the application is successful before departure. The dynamically applied spare points can make full use of the latest remaining paths of AGVs, improving the efficiency of spare point application and reducing the waiting time of vehicles due to failure to apply for spare points. As displayed in Fig. 10, the average waiting time for DSPA is 17%-29% and 6%-33% less than that for DRR and SZH, respectively. In terms of traveling distance, it can be seen that the total traveling distance of the three methods is similar, and the gap between them is only 0.7%-3%, with an average difference of 1.8%, as shown in Fig. 11. The traveling distance discrepancy is caused by the fact that the order of the task list is determined, while the location of the AGV receiving the task is uncertain, and some AGVs may receive tasks that are far away, which leads to the discrepancy seen in this metric. The DSPA method is still the most efficient, even when considering the impact of distance differences.

With an increasing number of AGVs, the timespan of DSPA decreases and reaches the minimum value when the number of vehicles equals 20. When the number of vehicles is above 20, the increase in vehicles does not significantly improve efficiency but does increase the timespan. This is because as the number of vehicles increases, although multiple AGVs can perform tasks simultaneously, the set of shared points also extends, and there are not enough spare points assigned to AGVs in this situation. Hence, the chance of AGVs using dynamic spare points decreases. The SZH method, which involves spare points, has a similar result. In the DRR method, as the number of AGVs increases

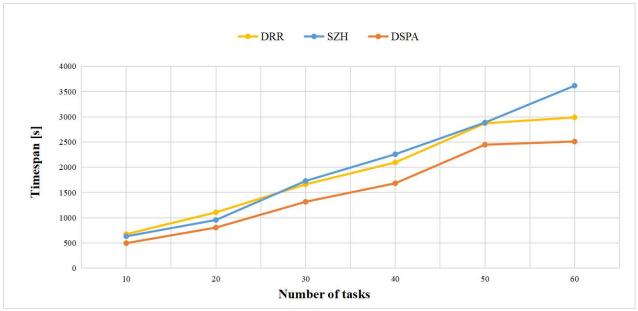

FIGURE 12. Timespan for different numbers of tasks.

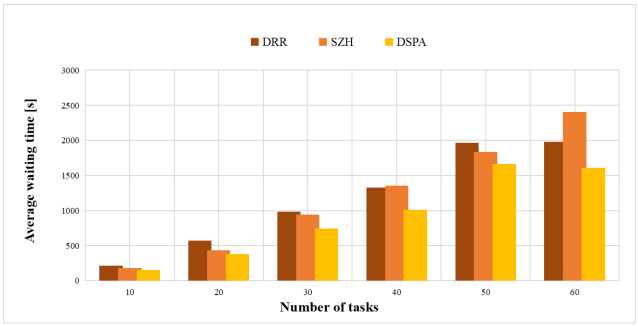

FIGURE 13. Average waiting time for different numbers of tasks.

making congestion grow, the timespan reaches a minimum with 15 AGVs and then gradually rises.

It can be seen that when the number of tasks and the scale of the workspace is fixed, more AGVs would not be better. Too many AGVs will raise costs and reduce efficiency. In all, the DSPA method proposed in this paper is more efficient than DRR and SZH for various numbers of AGVs.

2) IMPACT OF THE NUMBER OF TASKS

To have an insight into the performance, the impact of the number of tasks on the performance of those three methods is investigated. We set the number of vehicles as 15. From Fig. 12. we can see that the timespan with different numbers of tasks based on DSPA is 14%-26% and 15%-30% less than that of DRR and SZH, respectively. The average waiting time is 15%-33% and 9%-33% less than that of DRR and SZH, respectively, as shown in Fig. 13. The traveling distances of the three schemes differ much less, only 0.4%-5%, with an average of 2%, and the traveling distance of DSPA is always the least, as shown in Fig. 14.

As expected, we can also see that the timespans and the total traveling distances increase for all methods as the number of tasks increases. However, the timespan and average waiting time of SZH grows more than the others because most of the waiting time of the SZH method is generated before departure. The reason for this result is that the more tasks there are, the longer the waiting time of the SZH method due to the delay in starting the tasks for the failure to allocate spare points. Hence, for different numbers of tasks, the DSPA shows better performance with less timespan, waiting time, and traveling distance.

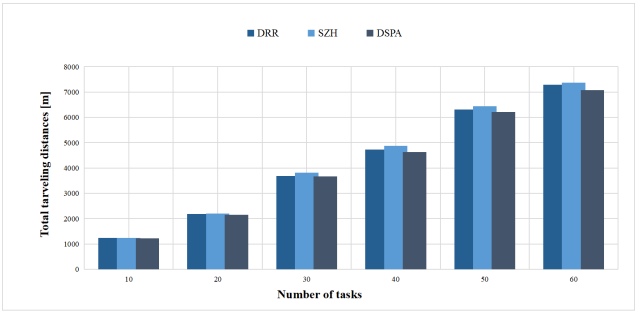

FIGURE 14. Total traveling distances for different numbers of vehicles.

TABLE 2. Relationship between the workspace scale and the number of identification points.

Scale of workspace	W10 R8	W15 R8	W20 R10	W25 R10	W30 R12
Number of identification points	260	366	544	616	892

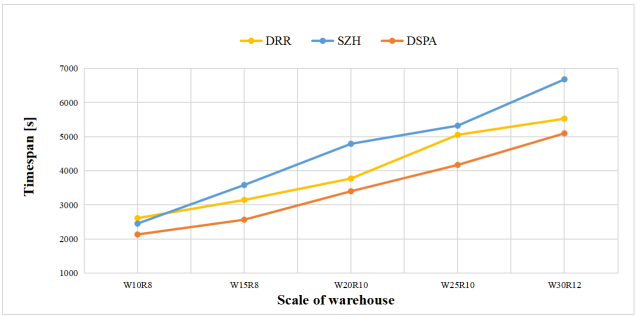

FIGURE 15. Timespan for different scales of workspace.

3) IMPACT OF THE WORKSPACE SCALE

Next, we evaluate the performance of DSPA under different scales of workspace by varying the number of workstations and the number of rows of the WIP warehouse, which means the number of identification points in the warehouse varies between 260 and 892. The number of vertices of the graph corresponding to different workspace sizes is shown in Table 2. The range of workspace is expressed as the combination of workstation numbers. The scale specification of the WIP warehouse tells the number of workstations, then rows of racks: e.g., W10R8 indicates ten workstations and eight rows of racks.

As shown in Figs. 15-16, as the warehouse size increases, both the timespan and the average waiting time increase. The reason is that the larger the warehouse, the longer the path could be for the AGV to perform tasks, and the longer the time to complete the task. However, the larger the warehouse and the longer the path, the more shared points there are and the

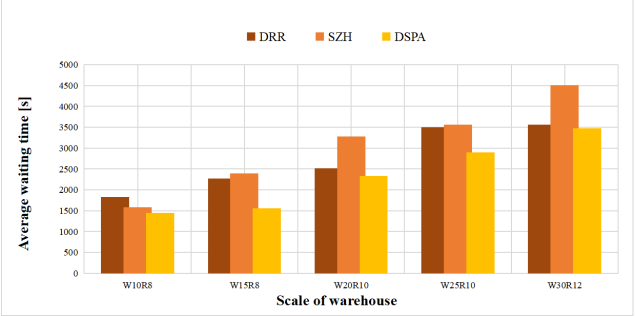

FIGURE 16. Average waiting time for different scales of workspace.

higher the possibility of failed spare point requests, causing a longer waiting time. Overall, in a horizontal comparison, our method is still the most efficient.

V. CONCLUSION

In this paper, a dynamic spare point application-based collision and deadlock prevention method is proposed for WIP warehouses. The technique can ensure collision-free and deadlock-free AGV travel while achieving high time efficiency. The DSPA approach makes full use of the advantages of SZH and DRR according to the characteristics of the WIP warehouse. It efficiently avoids AGV collision and deadlock by reserving the identification point and applying spare points dynamically.

Simulation results show that the timespan of DSPA is 11% to 28% less than DRR and SZH with different numbers of AGVs while also having a lower traveling distance. The timespan and waiting time using the DSPA technique are also less than those of DRR and SZH, with different numbers of tasks and different scales of workspace. All this confirms the effectiveness of the DSPA algorithm.

Future work can focus on flexibly defining the shared points between AGVs and pinpointing the areas where deadlocks may happen to improve the efficiency of AGV operation. Also, the path planning and optimization to improve the efficiency can also be a topic studied in the future. In addition, this algorithm is not only applicable to WIP warehouses but also to other warehouse layouts with similar characteristics.

REFERENCES

[1] S. Wan, X. Li, Y. Xue, W. Lin, and X. Xu, "Efficient computation offloading for Internet of Vehicles in edge computing-assisted 5G networks," *J. Supercomput.*, vol. 76, pp. 2518–2547, Apr. 2020, doi: 10.1007/S11227-019-03011-4.

[2] I. F. A. Vis, "Survey of research in the design and control of automated guided vehicle systems," *Eur. J. Oper. Res.*, vol. 170, no. 3, pp. 677–709, May 2006.

[3] R. Bhattacharya and S. Bandyopadhyay, "An improved strategy to solve shop deadlock problem based on the study on existing benchmark recovery strategies," *Int. J. Adv. Manuf. Technol.*, vol. 47, nos. 1–4, pp. 351–364, Mar. 2010.

[4] H. Chen, N. Wu, and M. Zhou, "A novel method for deadlock prevention of AMS by using resource-oriented Petri nets," *Inf. Sci.*, vol. 363, pp. 178–189, Oct. 2016.

[5] E. G. Coffman, M. J. Elphick, and A. Shoshani, "System deadlocks," *ACM Comput. Surv.*, vol. 3, no. 2, pp. 67–78, Jun. 1971.

[6] N. Smolic-Rocak, S. Bogdan, Z. Kovacic, and T. Petrovic, "Time windows based dynamic routing in multi-AGV systems," *IEEE Trans. Autom. Sci. Eng.*, vol. 7, no. 1, pp. 151–155, Jan. 2010, doi: 10.1109/TASE.2009.2016350.

[7] C. W. Kim and J. M. A. Tanchoco, "Conflict-free shortest-time bidirectional AGV routeing," *Int. J. Prod. Res.*, vol. 29, no. 2, pp. 2377–2391, 1991.

[8] Z. Zhang, Q. Guo, J. Chen, and P. Yuan, "Collision-free route planning for multiple AGVs in an automated warehouse based on collision classification," *IEEE Access*, vol. 6, pp. 26022–26035, 2018.

[9] X. Li, C. Zhang, W. Yang, and M. Qi, "Multi-AGVs conflict-free routing and dynamic dispatching strategies for automated warehouses," in *Proc. Int. Conf. Mobile Wireless Technol.*, 2018, pp. 277–286.

[10] Q. Yang, Y. Lian, and W. Xie, "Hierarchical planning for multiple AGVs in warehouse based on global vision," *Simul. Model. Pract. Theory*, vol. 104, Nov. 2020, Art. no. 102124, doi: 10.1016/j.simpat.2020.102124.

[11] Y. Zhao, X. Liu, G. Wang, S. Wu, and S. Han, "Dynamic resource reservation based collision and deadlock prevention for multi-AGVs," *IEEE Access*, vol. 8, pp. 82120–82130, 2020, doi: 10.1109/ACCESS.2020.2991190.

[12] M. De Ryck, M. Versteyhe, and F. Debrouwere, "Automated guided vehicle systems, state-of-the-art control algorithms and techniques," *J. Manuf. Syst.*, vol. 54, pp. 152–173, Jan. 2020, doi: 10.1016/j.jmsy.2019.12.002.

[13] Z. A. Banaszak and B. H. Krogh, "Deadlock avoidance in flexible manufacturing systems with concurrently competing process flows," *IEEE Trans. Robot. Autom.*, vol. 6, no. 6, pp. 724–734, Dec. 1990.

[14] G. Bocewicz, I. Nielsen, and Z. Banaszak, "Automated guided vehicles fleet match-up scheduling with production flow constraints," *Eng. Appl. Artif. Intell.*, vol. 30, pp. 49–62, Apr. 2014.

[15] N. Wu and W. Zeng, "Deadlock avoidance in an automated guidance vehicle system using a coloured Petri net model," *Int. J. Prod. Res.*, vol. 40, no. 1, pp. 223–238, Jan. 2002.

[16] J. López, E. Zalama, and J. Gómez-García-Bermejo, "A simulation and control framework for AGV based transport systems," *Simul. Model. Pract. Theory*, vol. 116, Apr. 2022, Art. no. 102430, doi: 10.1016/j.simpat.2021.102430.

[17] Q. Li, J. T. Udding, and A. Pogromsky, "Zone-control-based traffic control of automated guided vehicles," in *Coordination Control of Distributed Systems*. Cham, Switzerland: Springer, 2015, pp. 53–60.

[18] Y.-C. Ho, "A dynamic-zone strategy for vehicle-collision prevention and load balancing in an AGV system with a single-loop guide path," *Comput. Ind.*, vol. 42, nos. 2–3, pp. 159–176, Jun. 2000.

[19] K. Zheng, D. Tang, W. Gu, and M. Dai, "Distributed control of multi-AGV system based on regional control model," *Prod. Eng.*, vol. 7, no. 4, pp. 433–441, Jul. 2013, doi: 10.1007/S11740-013-0456-4.

[20] M. Qi, X. Li, X. Yan, and C. Zhang, "On the evaluation of AGVS-based warehouse operation performance," *Simul. Model. Pract. Theory*, vol. 87, pp. 379–394, Sep. 2018, doi: 10.1016/j.simpat.2018.07.015.

[21] Y.-C. Ho and T.-W. Liao, "Zone design and control for vehicle collision prevention and load balancing in a zone control AGV system," *Comput. Ind. Eng.*, vol. 56, no. 1, pp. 417–432, Feb. 2009, doi: 10.1016/j.cie.2008.07.007.

[22] W. Małopolski, "A sustainable and conflict-free operation of AGVs in a square topology," *Comput. Ind. Eng.*, vol. 126, pp. 472–481, Dec. 2018, doi: 10.1016/j.cie.2018.10.002.

[23] Y. Zhao, X. Liu, S. Wu, and G. Wang, "Spare zone based hierarchical motion coordination for multi-AGV systems," *Simul. Model. Pract. Theory*, vol. 109, May 2021, Art. no. 102294, doi: 10.1016/j.simpat.2021.102294.

[24] C. W. Kim, J. M. A. Tanchoco, and P.-H. Koo, "AGV dispatching based on workload balancing," *Int. J. Prod. Res.*, vol. 37, no. 17, pp. 4053–4066, Nov. 1999, doi: 10.1080/002075499189925.

[25] C. Chen and L. K. Tiong, "Using queuing theory and simulated annealing to design the facility layout in an AGV-based modular manufacturing system," *Int. J. Prod. Res.*, vol. 57, no. 17, pp. 5538–5555, Sep. 2019, doi: 10.1080/00207543.2018.1533654.

[26] C. Xie and T. T. Allen, "Simulation and experimental design methods for job shop scheduling with material handling: A survey," *Int. J. Adv. Manuf. Technol.*, vol. 80, nos. 1–4, pp. 233–243, 2015, doi: 10.1007/S00170-015-6981-X.

[27] M. F. Anwar and R. Nagi, "Integrated conflict free routing of AGVs and workcenter scheduling in a just-in-time production environment," in *Proc. 6th Annu. Ind. Eng. Res. Conf.*, Miami, FL, USA, 1997, pp. 216–221.

[28] S. Alexovič, M. Lacko, J. Bačík, and D. Perduková, "Introduction into autonomous mobile robot research and multi cooperation," in *Artificial Intelligence in Intelligent Systems*, vol. 229, R. Silhavy, Ed. Cham, Switzerland: Springer, 2021, pp. 326–336, doi: 10.1007/978-3-030-77445-5_30.
[29] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: Part I," *IEEE Robot. Autom. Mag.*, vol. 13, no. 2, pp. 99–110, Jun. 2006, doi: 10.1109/MRA.2006.1638022.

RUI XU received the Ph.D. degree in management science and engineering from the University of Science and Technology of China, in 2011. He is currently an Associate Professor with the School of Business, Hohai University, China. His current research interests include scheduling, simulation, and optimization using meta-heuristic algorithms.

HAOJUN FENG is currently pursuing the B.M.S. degree in information management and information systems with Hohai University, China. His research interests include operational research and optimization.

JIA LIU received the B.S. degree in computer science from the University of Lethbridge, Canada, and the M.S. degree in business administration from the Nanjing University of Science and Technology, Nanjing, China. He is currently the Director of the Business Operation Division of Logistics Center, China Tobacco Jiangsu Industrial Company Ltd., Nanjing. His research interests include supply chain and logistics management, simulation modeling, and logistics digital transformation.

WENJING HONG (Member, IEEE) received the B.Eng. and Ph.D. degrees in computer science and technology from the University of Science and Technology of China, China, in 2012 and 2018, respectively. Since 2021, she has been a Research Assistant Professor with the Department of Computer Science and Engineering, Southern University of Science and Technology, China. Her research interests include evolutionary computation and multiobjective optimization.