frontiers | Frontiers in Neurorobotics

TYPE Original Research
PUBLISHED 09 January 2023
DOI 10.3389/fnbot.2022.1053067

OPEN ACCESS

EDITED BY
Yimin Zhou,
Shenzhen Institutes of Advanced Technology (CAS), China

REVIEWED BY
Honglei Xu,
Curtin University, Australia
Jun Wang,
Western Sydney University, Australia
Jiaxi Wang,
Zhejiang Lab, China

*CORRESPONDENCE
Jianjun He
✉ jjhe@csu.edu.cn
Shuai Shen
✉ shenshuaiokay@163.com

RECEIVED 25 September 2022
ACCEPTED 14 December 2022
PUBLISHED 09 January 2023

CITATION
Hu E, He J and Shen S (2023) A dynamic integrated scheduling method based on hierarchical planning for heterogeneous AGV fleets in warehouses.
Front. Neurorobot. 16:1053067.
doi: 10.3389/fnbot.2022.1053067

A dynamic integrated scheduling method based on hierarchical planning for heterogeneous AGV fleets in warehouses

Enze Hu, Jianjun He* and Shuai Shen*

The School of Automation, Central South University, Changsha, China

In modern industrial warehouses, heterogeneous and flexible fleets of automated guided vehicles (AGVs) are widely used to improve transport efficiency. However, as their scale and limit of battery capacity increase, the complexity of dynamic scheduling also increases dramatically. The problem is to assign tasks and determine detailed paths to AGVs to keep the multi-AGV system running efficiently and sustainedly. In this context, a mixed-integer linear programming (MILP) model is formulated. A hierarchical planning method is used, which decomposes the integrated problem into two levels: the upper-level task-assignment problem and the lower-level path-planning problem. A hybrid discrete state transition algorithm (HDSTA) based on an elite solution set and the Tabu List method is proposed to solve the dynamic scheduling problem to minimize the sum of the costs of requests and the tardiness costs of conflicts for the overall system. The efficacy of our method is investigated by computational experiments using real-world data.

KEYWORDS

automated guided vehicles, dynamic integrated scheduling, task assignment, path planning, hierarchical planning, hybrid discrete state transition algorithm

1. Introduction

With the development in automation technology, AGVs as an important component of the modern warehouse logistics system is getting increased attention because of their accuracy, flexibility, and efficiency. More recently, heterogeneous AGV fleets are rapidly being adopted by industrial instances to perform different material handling tasks, where each vehicle has specific capabilities (e.g., pallet truck AGVs can tow loads, while backpack AGVs can lift loads). The minimization of travel costs is the most important objective of dynamic scheduling pursued in practice, which is affected by various decisions such as task assignment (i.e., assigning and sequencing tasks to AGVs), path planning (i.e., selecting optimal paths taken by each vehicle to reach the destination), and conflict management (i.e., avoiding conflicts between AGVs). These subproblems are interdependent; therefore, optimizing scheduling problems sequentially may yield a suboptimal performance of the overall AGV system (Maza and Castagna, 2005).

An example of a warehouse trying to implement an automated material handling system using a heterogeneous AGV fleet is Trucking Company (TC), which is a high-tech

listed company in Changsha, China. Currently, the vehicle management system used in TC relies on prepackaged software provided by AGV manufacturers. However, such software packages are not applicable to a heterogeneous AGV fleet and cannot handle dynamic problems such as the addition of new tasks and charging requests for AGVs. In addition, the optimal task assignment scheme may cause more traffic jams during path planning, and an evaluation index needs to be quantified and established for the delay time caused as a result of the waiting and detour strategy of the AGVs. As the configuration cost increases, a real-time and efficient integrated scheduling method becomes important in improving the economic performance of the warehouse. In this study, we focus on a dynamic integrated scheduling problem for heterogeneous AGVs with battery constraints.

Motivated by our collaboration with the TC, the main novelty of our problem setting in contrast to the existing literature is constituted by the combination of the following features. First, we specifically focused on solving the scheduling problems right on time, whereas the methods in most studies consume unreasonable computational effort, in particular, some exact methods (Schiffer and Walther, 2017; Ma et al., 2020; Singh et al., 2022). Second, the AGVs considered in this study are heterogeneous in terms of battery management, travel speed, and capabilities to perform transportation of different types of materials, which increases the complexity of the problem. Third, we simultaneously considered joint task assignments, path planning, and conflicts that reduce the problems of the AGV system. We aimed to make decisions on optimizing the overall AGV system performance rather than successively solving each subproblem. Our main contributions are summarized as follows:

First, we developed a mixed-integer linear programming (MILP) model for analyzing the scheduling of multi-AGVs, which combines both task assignment and path planning in automated warehouses. The model captures conflicts between a heterogeneous set of AGV fleets, allowing for scheduling according to the uncertain environment. The objective is to minimize the sum of the costs of requests and costs of conflicts. Constraints are also formulated to cope with features of capacity and battery management.

Second, the hierarchical planning method was used to decompose the complex and integrated scheduling problem. We propose a hybrid discrete state transition algorithm (HDSTA) considering the two-layer problems based on incorporating an elite solution set and the Tabu Search to find the optimal solution for the overall system instead of optimal solutions for each independent problem. Although our model is stylized for warehouses, the method can be applied to other applications such as flexible manufacturing systems and automated container terminals.

Third, we present the concept of a path expert database and its generation methods. The selection procedure based on a preset database is established for real-time path planning, which provides the foundation for dynamic scheduling.

Finally, numerical experiments are performed to validate the model according to the real-world data of warehouses in Changsha, China. Our approach is shown to yield approximate optimal solutions for AGV scheduling and path planning within a reasonable timeframe.

The remainder of this article is organized as follows. Relevant literature on the scheduling of multi-AGVs is discussed in the "Literature review" section. Problem description and the MILP model are formally established in the "Dynamic scheduling system and problem description" section. The "Hierarchical planning method" section presents the hierarchical planning method and introduces the proposed HDSTA and the selection procedure based on the path expert database. The "Computational experiments" section reports the experiments conducted to test the proposed method. Finally, conclusions and several future research directions are discussed in the "Conclusion" section.

2. Literature review

AGV scheduling can directly determine the efficiency and the cost of the overall transport system and therefore high attention is paid by researchers or manufacturing enterprises. Fazlollahtabar and Saidi-Mehrabad (2015) presented a literature review and divided AGV scheduling into three subproblems, task assignment, path planning, and collision avoidance. Many studies applied various methods, such as exact methods, heuristics, and meta-heuristics, to treat the subproblems separately or simultaneously.

As for exact methods, Desaulniers et al. (2003) designed an exact method including three algorithms (greedy search, column generation, and branch cutting), which enables solving the scheduling problem for four vehicles. Nishi et al. (2011) addressed a Lagrangian relaxation and cut scheme under the bilevel decomposition framework to optimize simultaneous task assignments and conflict-free routing problems. Fazlollahtabar and Hassanli (2018) presented a modified network simplex algorithm for blocking a scheduling problem in the manufacturing system. Nevertheless, because of the non-deterministic polynomial-time (NP)-hard nature of the scheduling problems, the exact method is only suitable for instances of small-scale problems.

For large-scale complex real-world problems, heuristics or metaheuristics are mainly adopted. Li et al. (2019) proposed an improved harmony search algorithm to improve the AGV scheduling rate, which can obtain the best harmony by considering the rate change. Zhang et al. (2019) proposed a genetic algorithm and a hybrid-load AGV scheduling model to reduce the total cost of the logistics system, which was successfully applied to a mixed-model automobile assembly

line. Abderrahim et al. (2020) used a variable neighborhood search algorithm to assign tasks in a manufacturing shop based on a vehicle manufacturing facility to minimize the maximum completion time. Zhang et al. (2022) proposed an improved iterated greedy algorithm to solve the AGV dispatching problem to minimize the total transportation cost of the matrix manufacturing workshop. In addition, many other meta-heuristics were also used in scheduling problems, such as the simulated annealing algorithm (Lu and Wang, 2019), the two-stage ant colony algorithm (Hamzeei et al., 2013), the evolutionary algorithm (Saidi-Mehrabad et al., 2015), and the particle swarm optimization algorithm (Gen et al., 2017). In the above literature, a common feature of the problems studied is that all the task information are stable and obtained in advance, and then, an analytical model was established and the problems are solved with a heuristic or meta-heuristic algorithm. Nevertheless, in a real-world instance, it is unrealistic to obtain all the task information in advance, while many uncertainties (e.g., urgent tasks and task rework) exist under dynamic and complex environments (Zhang et al., 2017). Therefore, the static scheduling method is insufficient for the complicated real-world industrial environment.

In recent years, with the development of IoT technology, many researchers focused on the dynamic scheduling problem. Li et al. (2020) proposed a multi-vehicle AGV scheduling mechanism for simulating multicustomer demands in an intelligent warehouse system. Mourtzis et al. introduced a cloud-based cyber-physical system with the help of IoT to achieve adaptive shop floor scheduling and condition-based maintenance. Umar et al. (2015) proposed an improved hybrid genetic algorithm method for dynamic scheduling that considers dispatching and conflict-free routing problems of AGVs under a flexible workshop environment. Lyu et al. (2019) presented an improved genetic algorithm combined with the Dijkstra algorithm considering time windows to solve the problems of optimal numbers, shortest transportation time, and conflict-free routing in the path planning process. Qiuyun et al. (2021) improved the particle swarm optimization algorithm to obtain the shortest transportation time for the AGV path planning problem of a one-line production line in manufacturing. Guo et al. (2020) studied the acceleration control method and the AGV priority determination method to improve the negotiation of AGVs that implement conflict-free path planning. Nevertheless, these researchers ignored the influence of not only the case of AGV heterogeneity but also battery management.

Through the review of the above literature, there have been no studies on dynamic integrated scheduling in warehouses for a heterogeneous set of AGV fleets with battery constraints. Therefore, a novel scheduling approach for AGVs is in high demand. In this study, we propose an HDSTA under a hierarchical planning framework to solve the complex problem, which is a kind of intelligent optimization algorithm with good global search capability and convergence property, considering the solution as a state and the update of the solution as a state transition process. Thus, we evaluated the proposed method with an industrial case study finally.

3. Dynamic scheduling system and problem description

In this section, a dynamic scheduling system for AGVs is proposed, which is based on a control system using inertial navigation guidance and QR codes. The information service is implemented by network and wireless routers. The integrated scheduling problems of heterogeneous AGVs with battery constraints in the AGV system are described and formulated while the conflict problem is highlighted.

3.1. Dynamic schedule system

The overall architecture of the dynamic scheduling system is presented in Figure 1. The dynamic supervisory layer provides real-time information about AGVs and the current schedule. The AGV monitoring system is responsible for managing the AGVs in terms of recognition, positioning, motor control, and battery level. The schedule monitoring system is responsible for receiving new tasks while monitoring the implementation of the current schedule and requesting a new schedule as a result of a change of tasks. The rescheduled module initialization harmonizes additional parameters with the running schedule that includes active AGVs, new tasks, completed tasks, and in-process tasks.

The integrated scheduling layer is responsible for determining the task assignment/sequence and path planning, which is more complex because of the consideration of conflict avoidance. AGV movement on warehouse layout is a multigraph problem, in that there are various parallel paths between the presorting stations. The path expert database is established in the offline stage, which can be regarded as a dataset containing warehouse layout information and the candidate elite paths sets between each presorting station. Accessory equipment such as sensors are equipped which enables AGVs to detect moving objects by hardware and avoid collision by preprocessed combination strategies of traffic regulations (e.g., stop and wait for the higher priority AGV to pass first or move around the conflict location). Conflicts can also be reduced by combining and changing the task assignment/sequence if it cannot be solved separately by path selection. However, the delay time as a result of the waiting and detour strategy of AGVs needs to be quantified and reduced. The schedule contains task assignment/sequence and path planning, which are generated by the task scheme generator and the path planning generator. The generated schedule is downloaded for execution by the system.

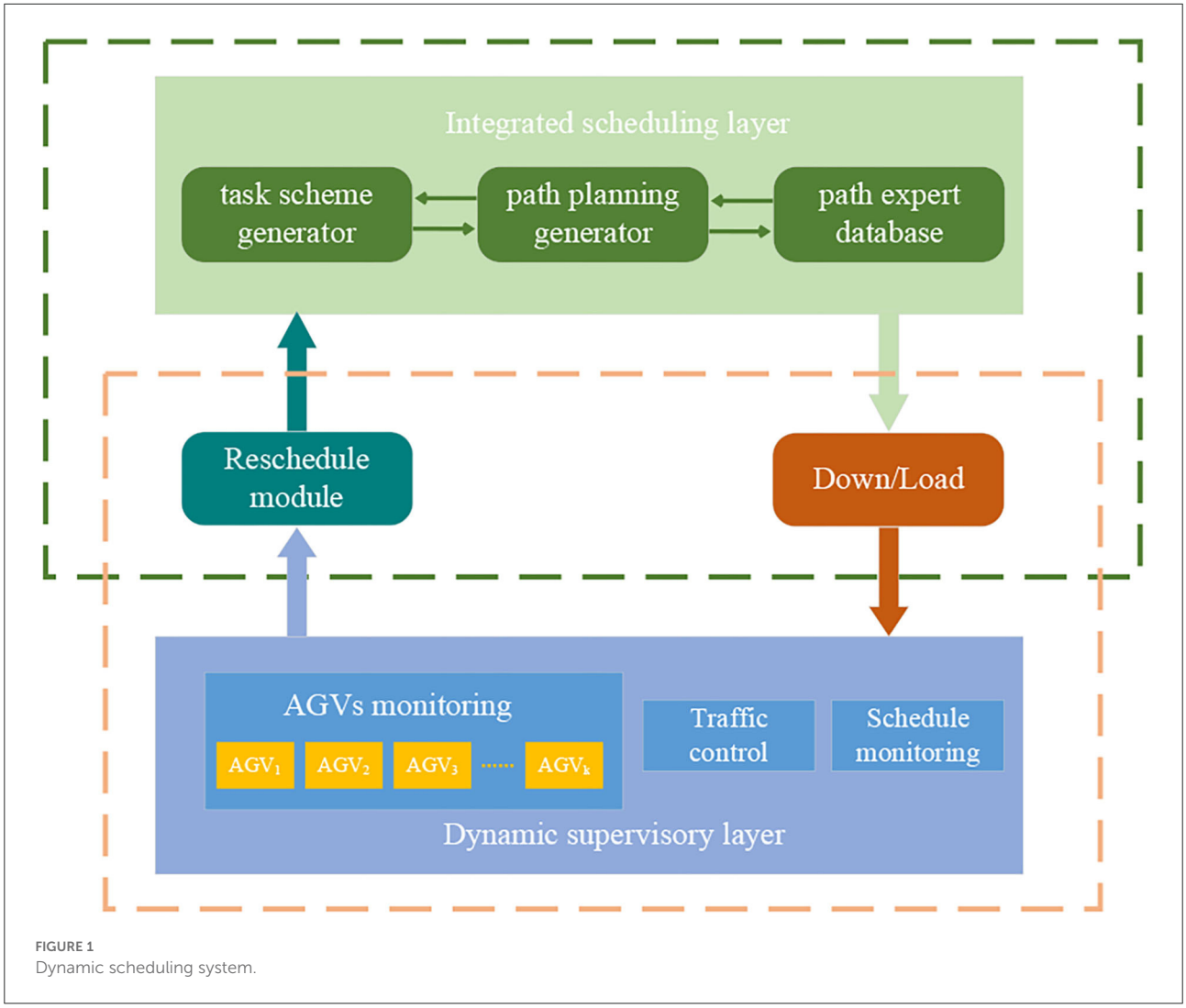

FIGURE 1
Dynamic scheduling system.

In a dynamic system, the assumptions considered are as follows: (1) loading and unloading times are fixed; (2) the AGVs move in four directions; and (3) the positioning deviation of the AGVs is negligible.

3.2. Problem statement

This study considers a real-world industry case of the TC where the goal is to have continuous material handling without human interference. Each transport process of the AGVs is composed of pickup travel, loading, delivery travel, and unloading. The layout of the warehouse is modeled as a multigraph, $G = (N, E)$, where $N = \{1, 2, \ldots, n\}$ is a set of all the nodes. Let $C \subset N$ denote the set of charging stations and $X \subset N$ denote the set of presorting stations, respectively. Moreover, $E = \{(i, j, p) : i, j \in N, i \neq j\}$ denotes the set of arcs between every node pair. The p^{th} path between the nodes i and j is represented by $(i, j, p) \in E$. Parallel paths are stored in the path expert database N_s which is established in the offline stage. If there is a collision between a pair of current paths, a path parallel to one of this pair of current paths can be used to replace this path for avoiding conflict.

In our problem setting, a set of transport tasks T are serviced by a set of heterogeneous AGVs K, and each task $r \in T$ contains a pickup node and a delivery node which are denoted by $u_r \in X$ and $d_r \in X$, respectively. Besides transport task requests, a set of charging requests is denoted by $B = \{1, 2, \ldots, |B|\}$, where $|B| = |C| \bullet |K|$ is the upper bound which is sufficiently large and C represents a set of charging stations. For each charging request $b \in B$, the pickup node and the delivery node are the same. The AGV makes a start instruction at the origin station s and each request contains only one delivery node. A termination instruction will be issued when the AGV reaches the terminal station e. Multiple request sets are defined by $R = T \cup B, R_s = R \cup \{s\}, R_e = R \cup \{e\}$, and $R_{se} = R \cup \{s\} \cup \{e\}$.

We considered battery constraints and the maximum and minimum battery levels for each AGV, where $k \in K$ are denoted by b_h^k and b_l^k, respectively. Before performing a new task, the battery level b_k of each AGV needs to be above the minimum threshold b_l^k. The AGV is not allowed to access the charging facilities while traveling with a load. AGVs are required to complete the current task with the first priority before going to the charging station. The discharging rate of AGV $k \in K$ is represented by d_k, while each AGV has a unit time travel cost of c_k.

As previously mentioned, AGVs are heterogeneous in terms of their capabilities to perform the transportation of different types of materials. Let C_r^T denote a set of capability requirements for each task, and each AGV has a specific capability C_k^K. The task $r \in T$ is only able to be performed by AGV $k \in K$ if $C_r^T \subseteq C_k^K$ holds to ensure that each task is performed by an AGV with corresponding handling capacity. For example, the tasks of lifting loads need to be performed by backpack AGVs while the pallet truck AGVs are only able to tow loads.

The path from the pickup node to the delivery node r is denoted by P_r, and the path from the delivery node of task r to the pickup node of task r' is denoted by $P_{rr'}$. The AGV has its specific forward speed and velocity of rotation, respectively. For each path $p \in P_r$, the travel time of AGV k is denoted by T_{rp}^k, and for each path $p \in P_{rr'}$, the travel time of AGV k is denoted by $T_{rr'p}^k$. In addition, the conflict-free path is optional during path planning because collisions can be prevented by traffic regulations. A delay time returns when an AGV follows a waiting and detour strategy to avoid a collision. When the path $p \in P_r$ of AGV k conflicts with the path $q \in P_m$ of AGV g, the delay time of AGV k on the path $p \in P_r$ is defined by $\Phi_{rkp}^{mgq}(z_{rk}^u, z_{mg}^u)$, where z_{rk}^u represents the time for AGV k to arrive at the pickup node of request r and z_{mg}^u represents the time for AGV g to arrive at the pickup node of request m, respectively. When the path $p \in P_r$ of AGV k conflicts with the path $q \in P_{mm'}$ of AGV g, the delay time of AGV k on the path $p \in P_r$ is defined by $\Phi_{rkp}^{mm'gq}(z_{rk}^u, z_{mg}^d)$, where z_{mg}^d represents the time for AGV g to arrive at the delivery node of request m.

3.3. Mixed-integer linear programming model

In this section, we formulate a mathematical model based on the problem description, which is an improvement from the findings of Dang et al. (2021) and Singh et al. (2022). Decision variables are introduced as follows:

x_{rk}^p: binary variable equal to 1 if AGV $k \in K$ travels from the pickup node to the delivery node of request $r \in R$ using $p \in P_r$ or 0 otherwise

$y_{rr'k}^p$: binary variable equal to 1 if AGV $k \in K$ travels from the delivery node of request $r \in R_s$ to the pickup node of request $r' \in R_e$ using $p \in P_{rr'}$ or 0 otherwise

z_{rk}^u: time of AGV k at the pickup node u_r of request $r \in R_e$; z_{ek}^u is the termination time of AGV k

z_{rk}^d: time of AGV k at the delivery node d_r of request $r \in R_s$; z_{sk}^d is the start time of AGV k

λ_{rk}^u: percent amount of battery discharge of AGV k at the pickup node u_r of request $r \in R_e$

λ_{rk}^d: percent amount of battery discharge of AGV k at the delivery node d_r of request $r \in R_s$

The mathematical model of the described problem is presented as follows:

The objective function f is to minimize the sum of the costs of requests and the tardiness costs of conflicts as the cost of each AGV is directly proportional to its travel time.

$$f = \min \sum_{k \in K} c_k (z_{ek}^u - z_{sk}^d)$$

$$\sum_{k \in K} \sum_{p \in p_r} x^p{}_{rk} = 1 \quad \forall r \in T \tag{1}$$

$$\sum_{k \in K} \sum_{p \in p_r} x^p{}_{rk} \leq 1 \quad \forall r \in B \tag{2}$$

$$x^p{}_{rk} = 0 \quad \forall p \in P_r,\ \forall r \in T, \forall k \in K, C_r^T \not\subseteq C_k^K \tag{3}$$

$$y^p{}_{rrk} = 0 \quad \forall r \in R, \forall k \in K \tag{4}$$

$$\sum_{r \in R_s} \sum_{p \in p_{rr'}} y^p{}_{rr'k} = \sum_{p \in p_{r'}} x^p{}_{r'k} \quad \forall r' \in R, \forall k \in K \tag{5}$$

$$\sum_{p \in p_r} x^p{}_{rk} = \sum_{r' \in R_e} \sum_{p \in p_{rr'}} y^p{}_{rr'k} \quad \forall r \in R, \forall k \in K \tag{6}$$

$$\begin{aligned} z_{rk}^u + T_{rk}^p + & \sum_{g \in K} \sum_{m \in T} \sum_{q \in p_m} \Phi_{rkp}^{mgq}(z_{rk}^u, z_{mg}^u) x^q{}_{mg} \\ & + \sum_{g \in K} \sum_{m \in R_s} \sum_{m' \in R_e} \sum_{q \in p_{mm'}} \Phi_{rkp}^{mm'gq}(z_{rk}^u, z_{mg}^d) \\ & * y^q{}_{mm'g} - M(1 - x^p{}_{rk}) \leq z_{rk}^d \quad \forall p \in P_r, \\ & \forall r \in T, \forall k \in K \end{aligned} \tag{7}$$

$$\begin{aligned} z_{rk}^d + T_{rr'k}^p + & \sum_{g \in K} \sum_{m \in T} \sum_{q \in p_m} \Phi_{rr'kp}^{mgq}(z_{rk}^d, z_{mg}^u) x^q{}_{mg} \\ & + \sum_{g \in K} \sum_{m \in R_s} \sum_{m' \in R_e} \sum_{q \in p_{mm'}} \Phi_{rr'kp}^{mm'gq}(z_{rk}^d, z_{mg}^d) \\ & * y^q{}_{mm'g} - M(1 - y^p{}_{rr'k}) \leq z_{r'k}^u \\ & \forall p \in P_{rr'}, \forall r \in R_s, \forall r' \in R_e, \forall k \in K \end{aligned} \tag{8}$$

Constraint (1) ensure that each transport request is assigned only one time and can be followed by another request. Constraint (2) makes sure that each charging request is presented by at most one AGV. Each charging request r has only one path. Constraint (3) ensures that the capabilities of the requests and AGVs match. Constraint (4) ensures that self-visits

are avoided. Constraints (5) and (6) make sure that the number of entering paths of request, execution paths, and leaving paths of each request are consistent. The time constraints are given by (7) and (8), and Constraint (7) calculates the travel time from the pickup node of request r to the delivery node. Constraint (8) calculates the travel time between different requests. The travel time includes transportation time and delay time, where M is a large positive constant.

$$b_l^k \le \lambda_{rk}^d \le b_h^k \quad \forall r \in R_s, \forall k \in K \tag{9}$$

$$b_l^k \le \lambda_{rk}^u \le b_h^k \quad \forall r \in R_e, \forall k \in K \tag{10}$$

$$\lambda_{rk}^u + d_k(z_{rk}^d - z_{rk}^u) - M(1 - x^p{}_{rk}) \le \lambda_{rk}^d \quad \forall p \in P_r, \forall r \in T, \forall k \in K \tag{11}$$

$$\lambda_{rk}^d + d_k(z_{r'k}^u - z_{rk}^d) - M(1 - y^p{}_{rr'k}) \le \lambda_{r'k}^u \quad \forall p \in P_{rr'}, \forall r \in R_s, \forall r' \in R_e, \forall k \in K \tag{12}$$

The constraints related to power consumption are given by (9) to (12). Constraints (9) and (10) set the lower and upper bounds for an amount of battery discharge. Constraint (11) calculates the amount of battery discharge due to the travels between the source and destination of a request. Constraint (12) calculates the amount of battery discharge due to the travel between the destination and the source of two requests.

$$x^p{}_{rk} \in \{0, 1\} \quad \forall p \in P_r, \ \forall r \in R, \forall k \in K \tag{13}$$

$$y^p{}_{rr'k} \in \{0, 1\} \forall p \in P_{rr'} \quad \forall r \in R_s, \forall r' \in R_e, \forall k \in K \tag{14}$$

$$z_{rk}^u \ge 0 \quad \forall t \in R_e, \forall k \in K \tag{15}$$

$$z_{rk}^d \ge 0 \quad \forall t \in R_s, \forall k \in K \tag{16}$$

The valid domains of the binary variables are given by constraints (13)–(16), which guarantee valid domains for the other decision variables.

4. Hierarchical planning method

In this section, a hierarchical planning method is proposed to solve the joint task assignments, path planning, and conflict problem for just-in-time scheduling. This method is inspired by the work of Hooker and Ottosson (2003) and decomposes the integrated optimization problems into an aggregated upper-level master problem and a lower-level subproblem. The upper-level problem is to make decisions for AGV task assignment/sequence, which determines a candidate elite solution set where the collision constraints for AGVs are neglected. The lower-level subproblem is to solve the optimal path planning problem with collision constraints under the conditions of the tentative solution at the upper level. The conflict problem is considered in both the master problem and subproblem, the collision between AGVs can be reduced by changing the detailed paths for vehicles or the scheme of task assignment and sequence. In summary, the objective is to minimize the AGV transportation time, which is the sum of the total travel time and the delay time (waiting or detour time for avoiding collisions). The detailed steps are described as follows:

Step 1. The upper level: Task assignment and sequence to AGVs where the collision constraints are removed from the original problem, and the transportation time of each task for each AGV is defined as the minimal time from the starting node to the delivery node. The master problem is regarded as the task assignment/sequence problem with constraints such as heterogeneous AGVs and batteries. In this study, a tentative elite solution set φ_i sorted in the ascending order of the objective function value is generated by HDSTA where the solution in the elite solution set is denoted by p_n.

Step 2. The lower level: Select the specific paths to perform the assigned tasks for the AGVs under the condition that a tentative solution p_n is derived from a master problem. For each solution, $p_n \in \varphi_i$ selected in the ascending order, the subproblem, which is concerned with the path planning problem to select the optimal paths with collision constraints for AGVs, is solved by the select procedure, while a list of conflict results with memories is generated, called Tabu List Λ_i. If the result of the selected procedure is conflict-free paths (termination criterion 1), the algorithm is completed; otherwise, recording conflict results to Λ_i. In the iteration, the solution with the minimum objective function values is recorded as the tentative optimal solution p_{best} and its delay time is defined by t_p.

Step 3. Algorithm termination criterion 2: The maximum allowable delay time is defined by ε. If t_p derived in Step 2 is less than ε, the algorithm is completed.

Step 4. Regenerate the tentative elite solution set φ_i considering the information is recorded in the tabu list Λ_i by HDSTA. If there is no improvement in the objective function value after five iterations, the algorithm is completed (termination criterion 3); otherwise, updating Λ_i and returning to Step 2.

The main scheme of hierarchical planning is illustrated in Algorithm 1. The accurate information about all AGVs and tasks are known, and the path expert database must be computed offline in advance.

An HDSTA with a path-select procedure and tabu list is proposed to find the optimal solution. The algorithm starts with a dynamic serve framework by generating a reschedule at the appropriate time interval methodology, based on the concept of dynamic scheduling, when there is a requirement for an additional task or AGV charging (lines 1–5). Then, the initialization of the tentative elite solution set φ_i using HDSTA (line 6) was carried out. For each solution $p_n \in \varphi_i$, detailed paths are generated using the path select procedure

10.3389/fnbot.2022.1053067

Require: set of AGVs K, set of tasks T, path expert database N_s
1: **if** there is a change in transport or charging requires **begin**
2: **Extract** finished tasks from the running schedule
3: **Combine** the remaining tasks and new tasks
4: **Update** K, T
5: **end**
6: **Initialize** tentative elite solution set φ_i by HDSTA
7: **while** ($y_1 = 1$ **or** $y_2 = 1$**or** $y_3 = 1$)
8: **for each** $p_n \in \varphi_i$
9: $[p, T, t_p, S_p] \leftarrow$ SP (p_n)
10: **if** $t_p = 0$ then
11: $D \leftarrow p_n$; $P_{best} \leftarrow p$; $T_{best} \leftarrow T$; $y_1 \leftarrow 1$
12: **break for;**
13: **end if**
14: **if** $t_p < t_{best}$ then
15: $t_{best} \leftarrow t_p$; $P_{best} \leftarrow p$; $T_{list} \leftarrow [p_n, S_p]$; $D \leftarrow p_n$; $T_{best} \leftarrow T$
16: **end if**
17: **end for**
18: **if** $t_{best} < \varepsilon$ then
19: $y_2 \leftarrow 1$
20: **end if**
21: **if** $y_1 = 0$ **or** $y_2 = 0$ then
22: **update** the elite solution set φ_i by HDSTA
23: **update** l
24: **if** $l > 5$
25: $y_3 \leftarrow 1$
26: **end if**
27: **end if**
28: **end while**
29: $S \leftarrow (D, P_{best})$
return S

Algorithm 1. Main scheme of hierarchical planning.

(SP). The transport time, delay time, and conflict points are calculated, and a tabu list is generated (lines 8–17). If a solution exists in the elite solution set that conflict-free paths can be generated in path planning is marked as y_1(lines 10–13). The optimal solution in the elite solution set whose delay time is less than ε is marked as y_2 (lines 18–20). If the objective function values showed no improvement after multiple iterations are marked as y_3 (lines 24–26), the iteration of the elite solution set is updated by HDSTA by incorporating the Tabu List constraints until one of the termination conditions are met and generating an integrated scheduling solution

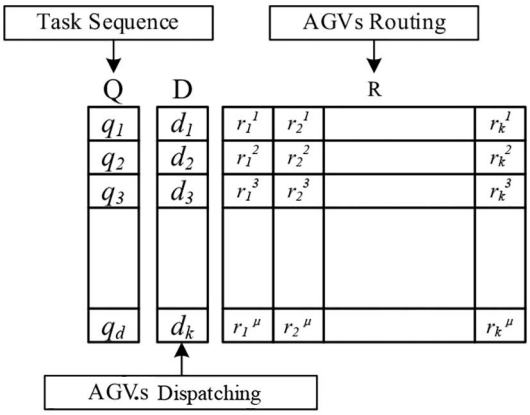

FIGURE 2
State space representation.

Require: path expert database N_s, dispatching D
1: $T_c \leftarrow 0$
2: **repeat**
3: **for** each $d_k \in D$
4: **for** each $t \in T_k$
5: $r_u^k \leftarrow Path(o_u^t, 1)$; $r_{u'}^k \leftarrow Path(o_d^t, 1)$
6: $r_t \leftarrow r_u^k$; $r_t \leftarrow r_{u'}^k$; $R_k \leftarrow r_t$
7: **end for**
8: $R \leftarrow R_k$
9: **end for**
10: $[T_c, C_i] \leftarrow$ Con (R)
11: **if** $T_c \neq 0$
12: **for** each C_i
13: $R' \leftarrow$ Replace (R_k)
14: $T_c^{'} \leftarrow$ Con (R')
15: **if** $T_c^{'} < T_c$
16: $T_c \leftarrow T_c^{'}$
17: $R \leftarrow R'$
18: **end if**
19: **end for**
20: **end if**
21: **return** T_c, R

Algorithm 2. Select procedure for AGV routing.

containing the sequential assignment solution and the detail path solution.

4.1. HDSTA

The state transition algorithm (STA) (Yang et al., 2013) is a kind of intelligent optimization algorithm originally proposed

Require: graph G, set of depots
$X = \{x_1, x_2, x_3 \ldots x_s\}$
Initialize matrix H_s
repeat
CLOSE list $\leftarrow$ 0; final $\leftarrow$ 0; Mark $= \varnothing$
OPEN list $\leftarrow$ start
  while
   while OPEN list $\neq$ 0
    if the number of the latest node in OPEN list $= 1$
     Current node $\leftarrow$ the latest node in OPEN list
    else Current node $\leftarrow$ the first node
      Mark $\leftarrow$ OPEN list; Mark $\leftarrow$ CLOSE list
    end if
    if current node $=$ end point
     break final $\leftarrow$ CLOSE list
    end if
     for each neighbor (Current node)
     if neighbor $\notin$ Obstacle
      new cost $= f$(neighbor)
       if new cost $<$ cost allel neighbor $\notin$ CLOSE list
        OPEN list $\leftarrow$ neighbor
       else CLOSE list $\leftarrow$ neighbor
      end if
     CLOSE list $\leftarrow$ current node
    end if
   end for
  end
  if Mark $\neq \varnothing$
   OPEN list $\leftarrow$ Mark (i); CLOSE list $\leftarrow$ Mark (j)
  else break
  end if
 end
final $=$ sort (final)
$N_S \leftarrow$ final
until the specified termination criterion is met
return N_S

Algorithm 3. Improved *A** algorithm for establishing the path expert database.

by Zhou et al. (2012) with good global search capability and convergence property. In our proposed HDSTA, the integrated problem is decomposed into individual elements and the individual S is defined by three "state spaces" related to its tasks sequence, AGV dispatch, and the corresponding routes, as depicted in Figure 2.

The task sequence state space Q registers for each task. The space Q consists of some types of tasks, with the subspace $q_d \in Q$ representing one type of task collection. The first type is charging requirement, while the others depend on the number of heterogeneous AGV types. It is worth mentioning that, to ensure performing urgent tasks first, the sequence of the task in each subspace must observe the rule of task priority. The dispatching state space D contains the task assignment for all AGVs, and the subspace $d_k \in D$ represents the dispatching of AGV $k \in K$. Finally, the AGV routing R corresponds to all paths, while the subspace $r_k^{u_k} \in R$ represents the path of the uth task of the kth AGV, respectively.

The integrated scheduling state space of AGVs is decoded for three subproblems, namely, task sequence, dispatching, and routing. Regularly representing an individual solution with appropriate random numbers is a very effective method to solve combinatorial optimization problems. However, to solve scheduling optimally in an integrated manner, intrinsic connections and constraints between the subproblems must be established.

An illustration of the integrated method is given in Figure 3. A sequence space consists of three types of requirements: a charging request (denoted in red), a piggyback transportation requirement (denoted in yellow), and a pallet transportation requirement (denoted in green). By randomly assigning tasks in each subspace to the matching AGVs, the corresponding dispatching space is generated. Each time the generated dispatching space performs the routing procedure, that is, to select the optimal path with the least collision in the path expert database and generate the conflict result feedback C. The optimization objective result S is the sum of total travel time *Cost* based on the current solution of dispatching space and conflict result C. The feasible solution of scheduling consists of a dispatching scheme and a detailed routing scheme. The role of sequence space is to define various requirements with priority and increase the search range of the algorithm.

In the task sequence state space Q, a candidate solution set is generated by the three special operators, a swap operator, shift operator, and symmetry operator (Yang et al., 2013), which are very effective to solve discrete optimization problems. Moreover, a candidate solution set is created by the times of the transformation called the search enforcement (SE) and the translation operator is performed only if a better new trail is found.

In the AGV dispatching space, the same four operators are applied to produce a candidate solution set, which is referred to as self-learning. However, the search space of the basic state transformation algorithm is normalized or specialized and cannot directly solve the problem with multiple subspaces. In the dispatching space, the communication strategy between the subspace d_k is necessary to exchange information for increasing the search intensity. Thus, we employed two move operators,

illustrated by Figure 4. (1)—the single insertion operator (SI): displaces the last task element of one subspace of dispatching to a random position in another subspace of dispatching and (2) the position-based crossover operator (PBC): exchange task elements of the same position randomly in two different subspaces of dispatching.

4.2. Select procedure

The method of AGV routing based on the path expert database is proposed for the first time. In the industrial context, collisions can be avoided by hardware, and conflict-free path planning is not our purpose as it takes a lot of computing

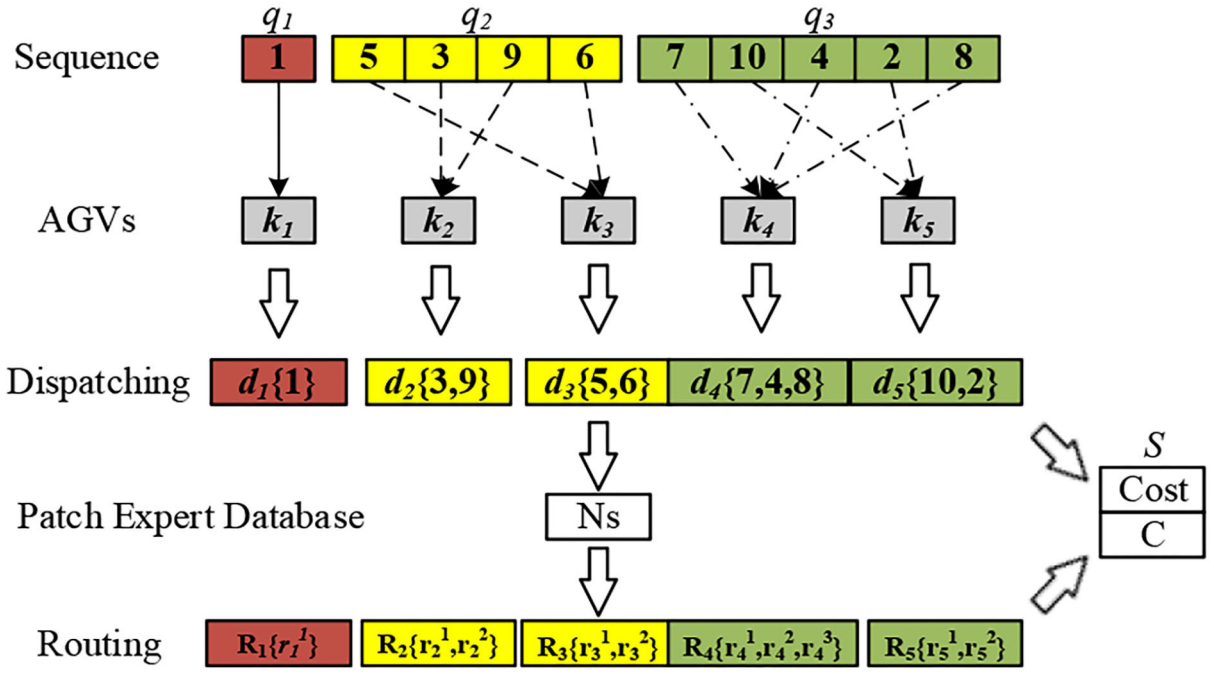

FIGURE 3
A description of the integrated scheduling problem.

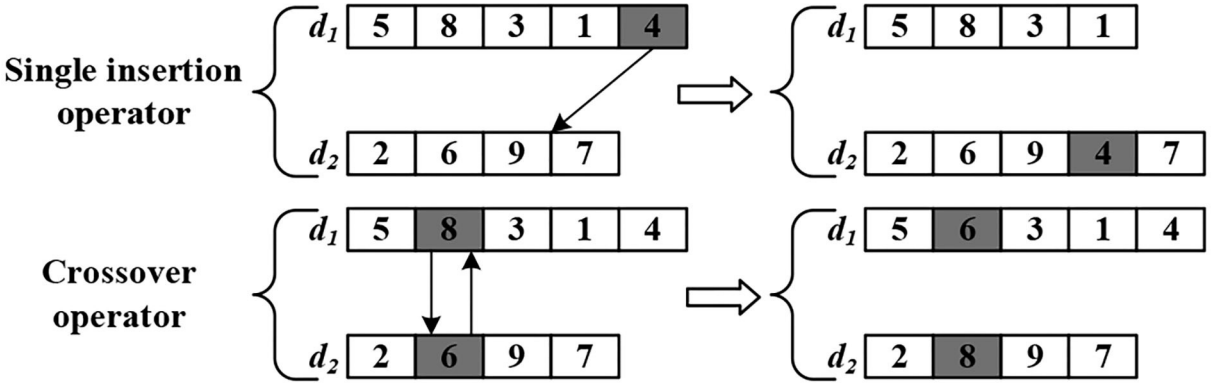

FIGURE 4
A description of move operators.

time and easily falls into locking in large-scale problems. The proposed method is to select a detailed path with the least conflicts from the path expert database according to the current dispatching and recording of the information of the conflict.

In this section, we give a path expert database N_s and task assignment for all AGVs, $D = d_1, d_2, d_3 \ldots d_k$ in this section. Let T_c denote the punishment time of the conflict process. As mentioned earlier, each task t consists of a pickup request o_u^t and a delivery request o_d^t. To represent a path-planning solution, we assign to the route r_t of each task two paths from the path expert database. The r_u^k and $r_{u'}^k$ are said to be traveling paths for pickup and delivery, respectively, by AGV k, and the nth path of the route r_{ij} from depot i to depot j in the path expert database is defined as $Path\left(i, j, n\right) \in N_s$. The selection procedure for AGV routing to optimize path planning and obtain punishment time is shown in Algorithm 2.

Set T_c to zero at the beginning (line 1). Then, a loop is executed to assign paths to each subspace d_k (lines 3–9). Each task $t \in T_k$ is assigned to the first path in the path expert database by the loop (lines 4–7). The time coordinates and punishment time of conflicts are calculated based on the time window, referred to as "CON" (line 10). When the conflict time is not zero, replace the best path with another path in the path expert database, referred to as "Replace," and update the routing data if the new path is better than all the queried paths (lines 11–20). Finally, the procedure returns the best solution (line 21).

In the dispatching space, the computational procedure does not consider the conflict situation. Thus, we establish a tabu list to record the conflict situation and then feed it back to dispatching and eliminate the unfeasible solutions in the next state to reduce conflicts whenever possible.

For example, in the current dispatching space, subspace $d_1\{2, 1, 3, 5\}$ and subspace $d_2\{8, 7, 6, 4\}$ have a conflict in the routing procedure and the conflict situation is for AGV 1 in performing task 2 and AGV 2 in performing task 8. The conflict results $d_1\{2, x, x, x\}$ and $d_2\{8, x, x, x\}$ are recorded for infeasible solution domains, where x is an arbitrary task. The result represents the infeasible solution that AGV 1 first performs task 2 while AGV 2 first performs task 8. In the following stage, the infeasible solutions are removed.

4.3. Path expert database

To the best of our knowledge, the establishment of the path expert database in the offline state for path planning is proposed for the first time. The optimal path between depots is several; besides, there are many good paths as well.

The concept of a path expert database is a collection that contains all optimal paths and good paths with sequences between depots for path replacement in case of a conflict.

The path expert database can be established by manually experience or algorithm programs depending on the different specifications of the warehouse. In this section, we propose an improved A* algorithm to generate a path expert database as shown in Algorithm 3.

The initial matrix of depots is defined by H_s (line 1). The algorithm loops over each route $r \in H_s$. A loop program calculates the paths between each depot (lines 2–33). Let the CLOSE list and the OPEN list denote a collection of nodes that have already been estimated and the collection of nodes that waiting for estimating. The path result is recorded in "final" and the points of the same valuation are recorded in "mark" (lines 3–4). The algorithm executes a loop that finds the optimal path between the two depots based on the A* algorithm and records the other points of the same valuation in each iteration (lines 6–25). If the collection "mark" is not blank, remount the information of points recorded successively to find all good paths between two depots (lines 26–29)—a record of all the path results (lines 31–32) and the procedure returns path expert database N_S finally (line 34).

5. Computational experiments

To evaluate the performance of the proposed method, computational experiments are performed in a dynamic scenario and under different scenarios with varying fleet sizes and numbers of tasks. We implemented the proposed dynamic scheduling method on a computer with an Intel (R) Core (Tm) CPU i7-9700 4.8 GHz and 8 GB RAM with a 64-bit Windows 10 operation system, while the scheduling rule is implemented in Python v3.6. The study adopts the warehouse production data located in Changsha, China. The layout of this warehouse is illustrated in Figure 5, which consists of 12 buffer area depots, 12 shop depots, 15 automatic vertical warehouse depots, and 5 charging stations. From the feedback from the practitioners, the average number of requests waiting for assigning is about 30 in a horizon, a horizon with more than 60 requests is regarded as a busy period.

The position of the depots (or stations) in the layout is fixed. Therefore, we made use of a distance matrix to compute the travel time of AGVs. We generated a path expert database through the program while offline and also note again that the collision-free trajectories are not considered in our experiments, since those collisions between the AGVs can be avoided by hardware.

5.1. Dynamic scheduling

A FlexSim-based digital simulation system is established to dynamically analyze the operation of AGV systems

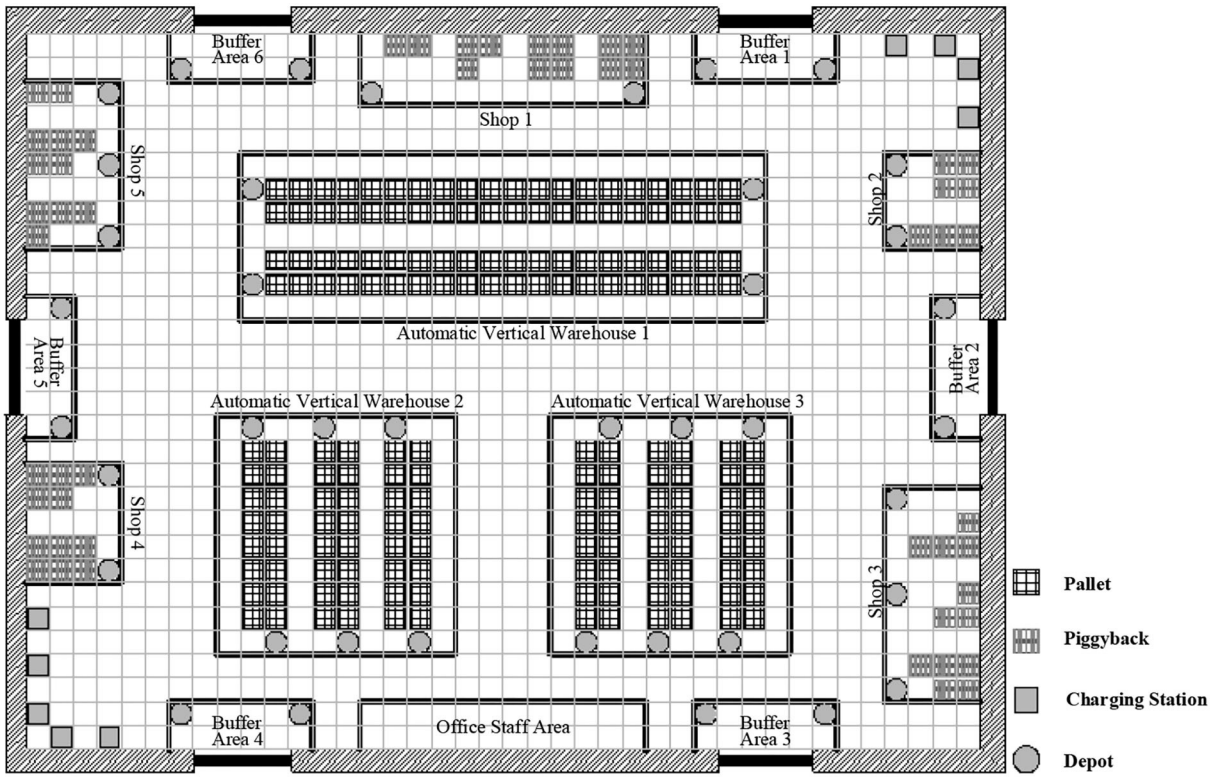

FIGURE 5
Warehouse layout.

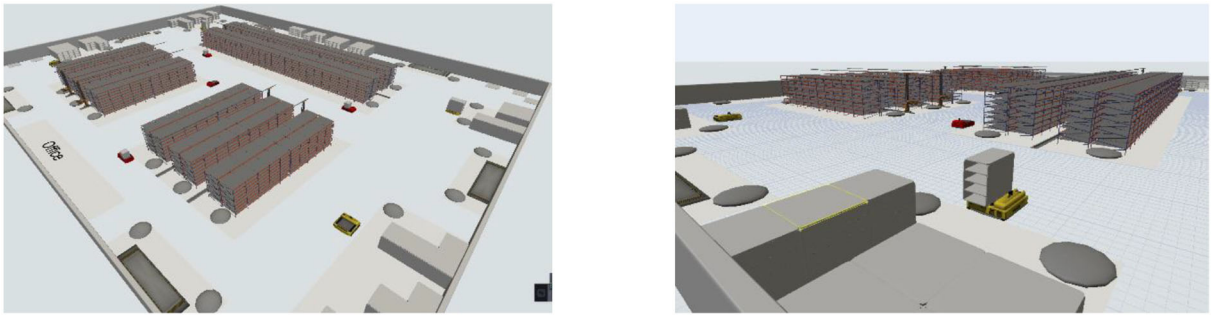

FIGURE 6
FlexSim-based digital simulation system.

under the industrial warehouse instances, as shown in Figure 6.

The description of the dynamic scheduling problem is shown in Table 1. The integrated scheduler algorithm processes a total of 60 tasks arriving at three different random intervals of time. Initially, 25 tasks are scheduled. While executing the initial schedule, 15 new tasks are added to the system at time $t = 14$ min. This results in a dynamic rescheduling of the system. While executing the current schedule, 20 more new tasks were added to the system at time $t = 22$ min.

As stated in the methodology, this is based on the concept of scheduling and rescheduling under an appropriate time intervals methodology of dynamic scheduling. Figure 7 shows the Gantt chart of the initial schedule. The dashed line at time $t = 14$ min represents the interruption and rescheduling, the points when new tasks are added to the system. The uncompleted tasks currently at the execution stage at the interruption and the

rescheduling points are task 1, task 8, task 7, and task 4. The tasks in execution will continue with the preemption in the next planning time horizon until the operation is completed. Figure 8 shows the Gantt chart for the generated new schedule, in which the new tasks are added after the interruption of the previous schedule. The operations at tasks 1, 8, 7, and 4 marked by the parallel slanted lines are the remaining operation from the previous schedule. On this schedule, all tasks in the system are either completed or the last task is under execution before the interrupt point in time $t = 22$ min. Figure 9 shows the Gantt chart for the generated new schedule. The tasks completed at the current interrupt and the rescheduling point are tasks 28, 26, 31, 33, 29, 36, 27, 37, and 34. Dynamic path planning adjusts the path without interrupting the current task execution process.

TABLE 1 The description of the dynamic scheduling problem.

Schedule	Start time (min)	New tasks
Initial	0	Task 1, Task 2, . . . , Task 25
Interrupt 1	14	Task 26, Task 27, . . . , Task 40
Interrupt 2	22	Task 41, Task 42, . . . , Task 60

5.2. Analysis of the scheduling results

The efficacy of our method is verified by computational experiments using real-world data with varying fleet sizes and numbers of tasks. The number of AGVs to be dispatched is 5,

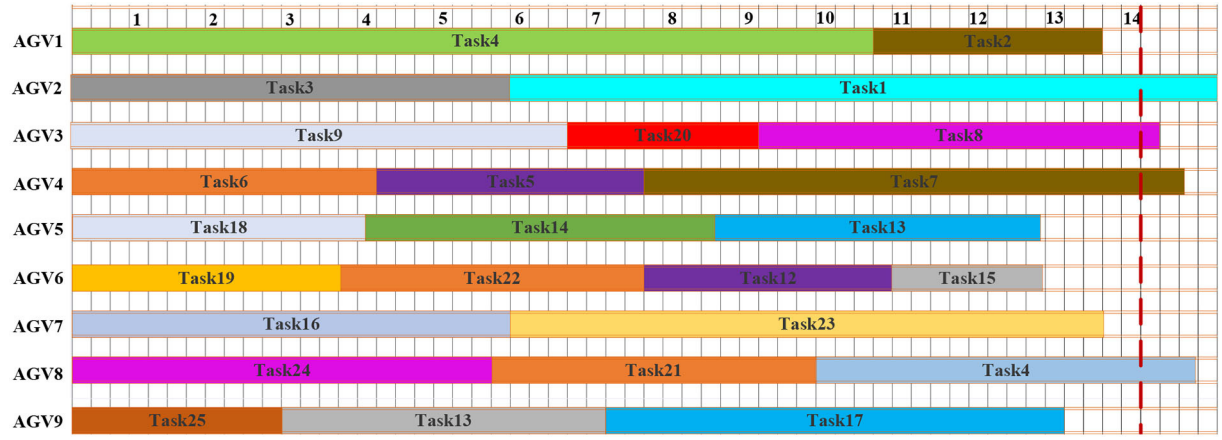

FIGURE 7
Gantt chart for dynamic scheduling 1.

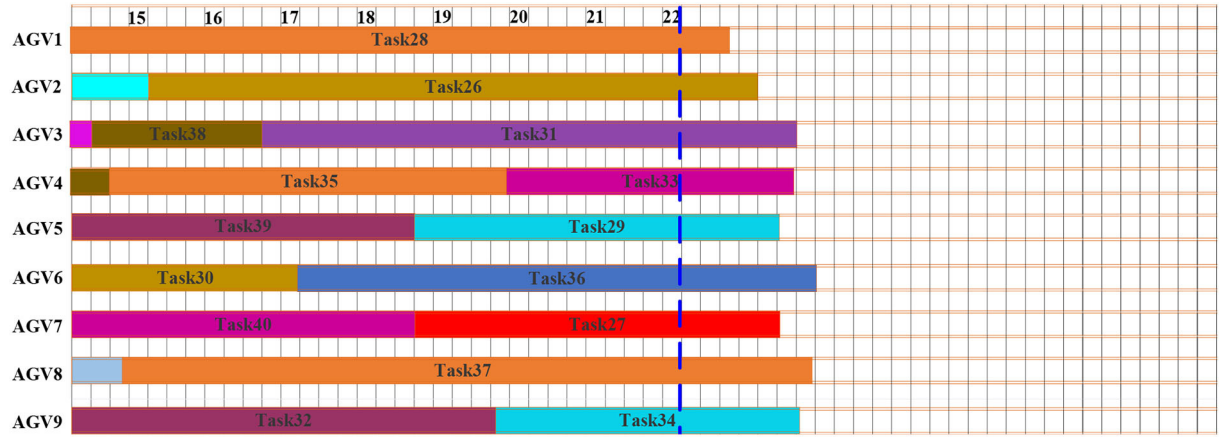

FIGURE 8
Gantt chart for dynamic scheduling 2.

10, and 15, respectively. The number of tasks to be allocated in the case is 50, 60, 70, 80, 90, and 100, respectively. Each case randomly generates five groups of tasks and runs them 10 times, for a total of 50 runs of the program. The average value is taken as the result. At present, the advanced AGV systems in industrial warehouses adopt the scheduling method of sequential optimization, of which the method proposed by Lian et al. (2020) is the most representative. Therefore, this method is selected for comparative verification of the analyses of real warehouse cases. Problems not considered in this method, such as the heterogeneity of the AGVs and battery constraint, are improved before the comparative verification in this study. In the case study, the comparison results of the task completion time and the delay time of the two methods are shown in Figures 10–12.

The results show that the integrated scheduling method proposed in this study has better performance and better solutions are found in all cases. In particular, the average task completion time is 13.62% less and the average delay time is 76.69% less than the sequential optimization of the scheduling method. The average delay time difference between the two methods is only 219 s when the number of tasks is 50 using 5 AGVs, but it increases to 3,591 s when the number of tasks increases to 100 using 15 AGVs. With the increase in task scale, the probability of conflicts between AGVs also increases dramatically. The sequential optimization scheduling method cannot avoid the impact of conflicts from the task allocation process, while the proposed integrated scheduling can avoid most conflicts by changing the task assignment and specific execution path.

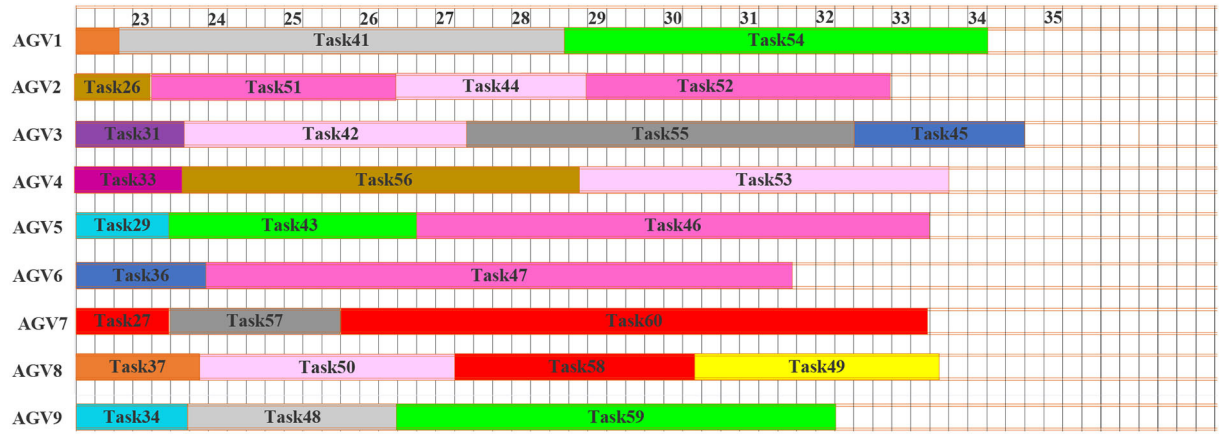

FIGURE 9
Gantt chart for dynamic scheduling 3.

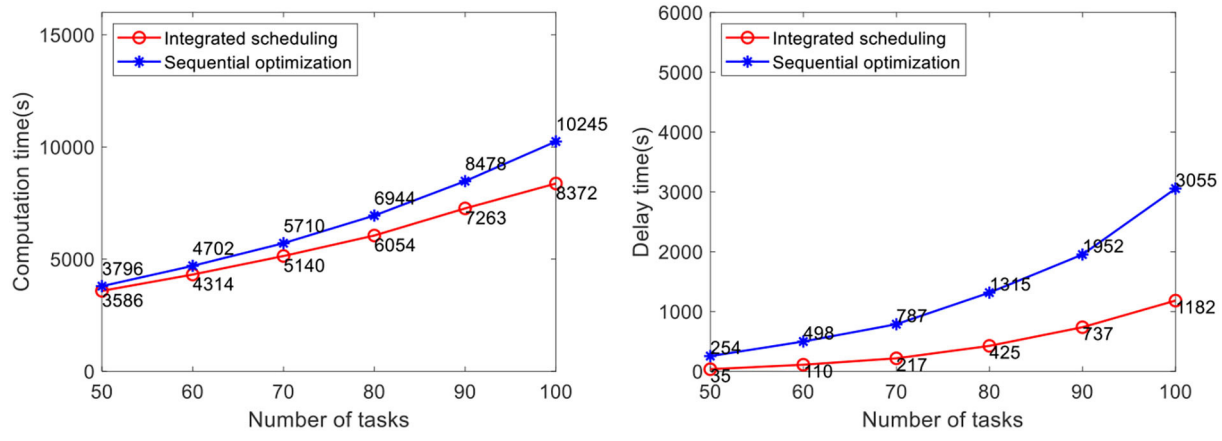

FIGURE 10
Comparative analysis of task completion time and delay time with 5 AGVs.

5.3. Comparison of algorithms

To verify the performance of HDSTA, a larger-scale case template needs to be established. We generate 10 instances for each scenario with 50–150 tasks and AGVs to be scheduled, ranging from 5 to 25; each instance runs ten times to compute the mean. In each instance, HDSTA with the adaptive large neighborhood search algorithm (HALNS) (Dang et al., 2021) was compared with the preplanning algorithm (PPA) (Maza and Castagna, 2005). HALNS is a hybrid algorithm of the adaptive large neighborhood search algorithm and the linear programming algorithm, which is proposed to solve the heterogeneous AGV scheduling problem with charge capacity constraints. However, this method does not consider the problem of conflict and deadlock. For comparison, we developed the conflict detection method to compute the delay time of its optimal solution. PPA is a strategy to generate conflict-free paths.

Table 2 compares the task completion time and scheduling computation time for varying scenarios with different numbers of AGVs and different numbers of tasks, where the task completion time is directly proportional to the operation cost, which can visually reflect the collaborative operation efficiency of AGVs, and the computation time is an important index of dynamic scheduling, which can reflect the computation efficiency of AGV systems. Table 2 compares the performance and computation time of PPA, HALNS, and HDSTA for 150 sets of tasks in 15 case types. For example, when the task volume is 50 and the number of AGVs is 5, 9, and 11, respectively, the task completion times of the scenarios calculated by optimal scheduling with the HDSTA algorithm are 3,738, 3,811, and 3,845 s and the computation times are 2.23, 2.33, and 2.37 s, respectively. The results of 150 sets

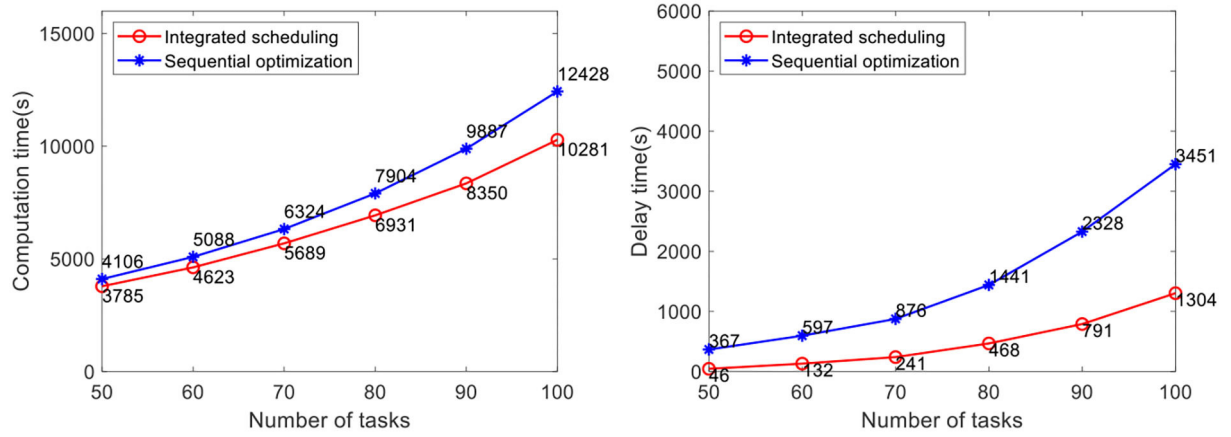

FIGURE 11
Comparative analysis of task completion time and delay time with 10 AGVs.

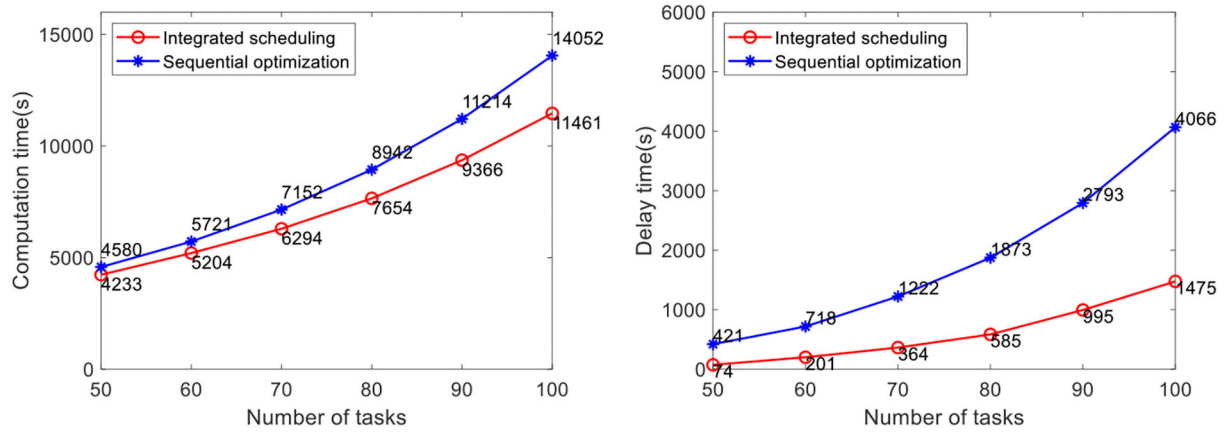

FIGURE 12
Comparative analysis of task completion time and delay time with 15 AGVs.

TABLE 2 Results of the varying scenarios.

NO.	Number of tasks	Number of AGVs	PPA		HALNS		HDSTA	
			Computation time (s)	Task completion time (s)	Computation time (s)	Task completion time (s)	Computation time (s)	Task completion time (s)
1	50	5	7.61	3,747	4.24	3,744	2.23	37,38
2	50	9	7.73	3,830	4.36	3,862	2.33	3,811
3	50	11	8.13	3,853	4.68	3,876	2.37	3,845
4	60	3	11.41	4,522	6.88	4,529	2.84	4,522
5	60	8	12.14	4,615	7.20	4,628	2.83	4,608
6	60	13	12.81	4,693	8.36	4,854	2.90	4,685
7	80	4	19.91	6,570	16.07	6,569	3.55	6,558
8	80	6	22.89	6,683	18.15	6,711	3.91	6,654
9	100	7	38.92	8,421	31.94	8,408	5.64	8,311
10	100	15	43.85	8,636	30.11	8,756	6.22	8,541
11	120	15	50.21	12,850	41.65	13,365	8.60	12,305
12	120	18	54.33	12,902	56.84	14,025	8.84	12,654
13	120	19	55.12	13,357	58.52	14,359	8.92	13,147
14	150	22	98.21	18,724	84.74	19,015	13.45	17,521
15	150	23	116.89	19,031	91.61	19,584	14.16	16,984

of tasks for 15 case types are analyzed and compared with PPA and HALNS. The average task completion time of the HDSTA solution proposed in this study is lower by 3.44 and 7.27% and the computation time is less by 84.15 and 81.92%.

6. Conclusion

This article studied the problem of scheduling a heterogeneous fleet of AGVs. A MILP model was formulated to minimize the sum of the costs of requests and the tardiness costs of conflicts. The hierarchical planning method is used to decompose the complex and integrated scheduling problem. We propose that HDSTA combine select procedures. The major novelty of this study is the ability to solve the dynamic integrated scheduling problem for heterogeneous AGV fleets with battery constraints. We performed numerical experiments to validate our model according to the real-world conditions of the automated warehouses in Changsha, China.

In the future, we may extend our research to improve our approach to multiple pickups and deliveries along the same route (multi-load AGVs) and the inclusion of path planning in the scheduling process.

Data availability statement

The original contributions presented in the study are included in the article/supplementary material, further inquiries can be directed to the corresponding authors.

Author contributions

EH and SS: conceptualization and writing—original draft preparation. EH: data curation. EH and JH: methodology, validation, and formal analysis. JH: writing—review and editing and funding acquisition. All authors contributed to the article and approved the submitted version.

Funding

This study was supported by National Natural Science Foundation of China under Grant No. 61873282.

Conflict of interest

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Publisher's note

References

Abderrahim, M., Bekrar, A., Trentesaux, D., Aissani, N., and Bouamrane, K. (2020). Bi-local search based variable neighborhood search for job-shop scheduling problem with transport constraints. *Optimiz. Lett.* 16, 255–280. doi: 10.1007/s11590-020-01674-0

Dang, Q. V., Singh, N., Adan, I., Martagan, T., and van de Sande, D. (2021). Scheduling heterogeneous multi-load AGVs with battery constraints. *Comput. Operat. Res.* 136, 105517. doi: 10.1016/j.cor.2021.105517

Desaulniers, G., Langevin, A., Riopel, D., and Villeneuve, B. (2003). Dispatching and conflict-free routing of automated guided vehicles: an exact approach. *Int. J. Flexible Manufact. Syst.* 15, 309–331. doi: 10.1023/B:FLEX.0000036032.41757.3d

Fazlollahtabar, H., and Hassanli, S. (2018). Hybrid cost and time path planning for multiple autonomous guided vehicles. *Appl. Intellig.* 48, 482–498. doi: 10.1007/s10489-017-0997-x

Fazlollahtabar, H., and Saidi-Mehrabad, M. (2015). Methodologies to optimize automated guided vehicle scheduling and routing problems: a review study. *J. Intell. Robotic Syst.* 77, 525–545. doi: 10.1007/s10846-013-0003-8

Gen, M., Zhang, W., Lin, L., and Yun, Y. (2017). Recent advances in hybrid evolutionary algorithms for multiobjective manufacturing scheduling. *Comput. Indus. Eng.* 112, 616–633. doi: 10.1016/j.cie.2016.12.045

Guo, K., Zhu, J., and Shen, L. (2020). An improved acceleration method based on multi-agent system for AGVs conflict-free path planning in automated terminals. *IEEE Access* 9, 3326–3338. doi: 10.1109/ACCESS.2020.3047916

Hamzeei, M., Farahani, R. Z., and Rashidi-Bejgan, H. (2013). An exact and a simulated annealing algorithm for simultaneously determining flow path and the location of P/D stations in bidirectional path. *J. Manufact. Syst.* 32, 648–654. doi: 10.1016/j.jmsy.2013.07.002

Hooker, J. N., and Ottosson, G. (2003). Logic-based Benders decomposition. *Mathe. Programm.* 96, 33–60. doi: 10.1007/s10107-003-0375-9

Li, G., Li, X., Gao, L., and Zeng, B. (2019). Tasks assigning and sequencing of multiple AGVs based on an improved harmony search algorithm. *J. Ambient Intell. Humaniz. Comput.* 10, 4533–4546. doi: 10.1007/s12652-018-1137-0

Li, Z., Barenji, A. V., Jiang, J., Zhong, R. Y., and Xu, G. (2020). A mechanism for scheduling multi robot intelligent warehouse system face with dynamic demand. *J. Intell. Manuf.* 31, 469–480. doi: 10.1007/s10845-018-1459-y

Lian, Y., Yang, Q., Xie, W., and Zhang, L (2020). Cyber-physical system-based heuristic planning and scheduling method for multiple automatic guided vehicles in logistics systems. *IEEE Trans. Indus. Inform.* 17, 7882–7893. doi: 10.1109/TII.2020.3034280

Lu, H., and Wang, S. (2019). A study on multi-ASC scheduling method of automated container terminals based on graph theory. *Comput. Indus. Eng.* 129, 404–416. doi: 10.1016/j.cie.2019.01.050

Lyu, X., Song, Y., He, C., Lei, Q., and Guo, W. (2019). Approach to integrated scheduling problems considering optimal number of automated guided vehicles and conflict-free routing in flexible manufacturing systems. *IEEE Access* 7, 74909–74924. doi: 10.1109/ACCESS.2019.2919109

Ma, X., Bian, Y., and Gao, F. (2020). An improved shuffled frog leaping algorithm for multiload AGV dispatching in automated container terminals. *Math. Probl. Eng.* 2020, 1260196. doi: 10.1155/2020/1260196

Maza, S., and Castagna, P. (2005). A performance-based structural policy for conflict-free routing of bi-directional automated guided vehicles. *Comput. Industry* 56, 719–733. doi: 10.1016/j.compind.2005.03.003

Nishi, T., Hiranaka, Y., and Grossmann, I. E. (2011). A bilevel decomposition algorithm for simultaneous production scheduling and conflict-free routing for automated guided vehicles. *Comput. Operat. Res.* 38, 876–888. doi: 10.1016/j.cor.2010.08.012

Qiuyun, T., Hongyan, S., Hengwei, G., and Ping, W. (2021). Improved particle swarm optimization algorithm for AGV path planning. *IEEE Access* 9, 33522–33531. doi: 10.1109/ACCESS.2021.3061288

Saidi-Mehrabad, M., Dehnavi-Arani, S., Evazabadian, F., and Mahmoodian, V. (2015). An Ant Colony Algorithm (ACA) for solving the new integrated model of job shop scheduling and conflict-free routing of AGVs. *Comput. Indus. Eng.* 86, 2–13. doi: 10.1016/j.cie.2015.01.003

Schiffer, M., and Walther, G. (2017). The electric location routing problem with time windows and partial recharging. *Eur. J. Oper. Res.* 260, 995–1013. doi: 10.1016/j.ejor.2017.01.011

Singh, N., Dang, Q. V., Akcay, A., Adan, I., and Martagan, T. (2022). A matheuristic for AGV scheduling with battery constraints. *Eur. J. Oper. Res.* 298, 855–873. doi: 10.1016/j.ejor.2021.08.008

Umar, U. A., Ariffin, M. K. A., Ismail, N., and Tang, S. H. (2015). Hybrid multiobjective genetic algorithms for integrated dynamic scheduling and routing of jobs and automated-guided vehicle (AGV) in flexible manufacturing systems (FMS) environment. *Int. J. Adv. Manuf. Technol.* 81, 2123–2141. doi: 10.1007/s00170-015-7329-2

Yang, C. H., Tang, X. L., Zhou, X. J., and Gui, W. H. (2013). A discrete state transition algorithm for traveling salesman problem. *Control Theory Applic.* 30, 1040–1046. doi: 10.7641/CTA.2013.12167

Zhang, L., Hu, Y., and Guan, Y. (2019). Research on hybrid-load AGV dispatching problem for mixed-model automobile assembly line. *Proc. CIRP* 81, 1059–1064. doi: 10.1016/j.procir.2019.03.251

Zhang, X., Sang, H., Li, J., Han, Y., and Duan, P. (2022). An effective multi-AGVs dispatching method applied to matrix manufacturing workshop. *Comput. Indus. Eng.* 163, 107791. doi: 10.1016/j.cie.2021.107791

Zhang, Y., Zhu, Z., and Lv, J. (2017). CPS-based smart control model for shopfloor material handling. *IEEE Trans. Indus. Inform.* 14, 1764–1775. doi: 10.1109/TII.2017.2759319

Zhou, X., Yang, C., and Gui, W. (2012). State transition algorithm. *J. Ind. Manag. Optim.* (2012) 8, 1039–1056. doi: 10.3934/jimo.2012.8.1039