TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 07/24 pp1355–1367
DOI: 10.26599/TST.2023.9010087
Volume 29, Number 5, October 2024

An Effective Optimization Method for Integrated Scheduling of Multiple Automated Guided Vehicle Problems

Hongyan Sang*, Zhongkai Li, and M. Fatih Tasgetiren

Abstract: Automated Guided Vehicle (AGV) scheduling problem is an emerging research topic in the recent literature. This paper studies an integrated scheduling problem comprising task assignment and path planning for AGVs. To reduce the transportation cost of AGVs, this work also proposes an optimization method consisting of the total running distance, total delay time, and machine loss cost of AGVs. A mathematical model is formulated for the problem at hand, along with an improved Discrete Invasive Weed Optimization algorithm (DIWO). In the proposed DIWO algorithm, an insertion-based local search operator is developed to improve the local search ability of the algorithm. A staggered time departure heuristic is also proposed to reduce the number of AGV collisions in path planning. Comprehensive experiments are conducted, and 100 instances from actual factories have proven the effectiveness of the optimization method.

Key words: Automated Guided Vehicle (AGV); integrated scheduling; staggered time departure heuristics

1 Introduction

As an important part of industrial automation, transportation has been widely investigated over the years. In particular, Automated Guided Vehicles (AGV) are considered an indispensable part of the industrial transportation system[1]. With the increasing demand for flexible manufacturing systems for AGVs[2], research on AGV scheduling in production workshops has great value. The present paper mainly investigates the problem of task assignment and path planning of AGVs in matrix production workshops, while considering three factors, namely, capacity constraints, time constraints, and collision avoidance. This paper aims to improve the transportation efficiency of AGVs in the production workshop, reduce the transportation cost of AGVs, and ultimately improve the transportation safety of AGVs.

Considering the transportation cost of AGV, including total running distance, time penalty, and loss cost of AGV, Zou et al.[3–5] studied AGV task assignment for Foxconn's matrix production workshop. Task assignment is an NP-hard problem of constrained optimization[6]. Through heuristic[7–8] and meta-heuristic algorithms[9–13], a better scheduling scheme can quickly be found. The task assignment stage contains two important attributes: divisibility and fault tolerance, both of which ensure that tasks can be effectively assigned to multiple AGVs, while improving resource utilization efficiency. Typically, the task assignment stage should follow certain constraints below:

(1) Capacity constraint: The carrying capacity of each AGV has a fixed upper limit, and the redundant tasks need to be transported by other AGVs when the tasks transported by AGVs exceed the upper limit.

(2) Time constraint: The tasks to be delivered should be delivered within the specified time. Often, a penalty time is imposed if the delivery is carried out too early or too late.

• Hongyan Sang and Zhongkai Li are with School of Computer Science, Liaocheng University, Liaocheng 252000, China. E-mail: sanghongyan@lcu-cs.com; 493056738@qq.com.
• M. Fatih Tasgetiren is with Industrial Engineering Department, Baskent University, Ankara 06000, Türkiye. E-mail: ftasgetiren@gmail.com.
* To whom correspondence should be addressed.
Manuscript received: 2023-07-06; revised: 2023-07-30; accepted: 2023-08-10

© The author(s) 2024. The articles published in this open access journal are distributed under the terms of the Creative Commons Attribution 4.0 International License (http://creativecommons.org/licenses/by/4.0/).

Scholars used accurate algorithms in the early stage to solve the problem of task assignment[14−16]. Accurate algorithms can obtain accurate solutions for small cases in a short period; however, the solution time for large cases is too long. For large-scale instances, scholars use heuristic and meta-heuristic algorithms to solve such problems[17−27].

The current literature only considers the assignment of tasks, and does not consider the integration of path planning[28−32]. The reduction of AGV transportation costs and the improvement of transportation efficiency rely on a good scheduling scheme. However, the scheduling strategy cannot guarantee that there would be no collision in the actual production if it only considers the assignment of tasks. Therefore, considering path planning with a collision is of great significance.

Path planning serves to find the shortest path without collisions. The current path-planning methods are divided into single AGVs, multiple AGVs, and other situations[33]. De Ryck et al.[1] explained a graph search algorithm for a single AGV. Unlike a single AGV, deadlock and collision must be considered in the path planning problem of multiple AGVs. Thus far, scholars have used deadlock-free or collision-free strategies for path planning[34−38].

In multi-AGV scheduling, collision always occurs in the operation of AGVs. Thus, in the path planning of two AGVs, when a collision occurs in the generated path, it can be considered to regenerate the path of one AGV to avoid the collision[39]. However, when the number of AGVs is larger, the effect of regenerating the AGV path becomes worse and worse. Given that there is a high probability of AGV collision at this time, it is difficult to find a path without the collision. Furthermore, to avoid the collision, the transportation cost of the generated path will increase. AGVs can effectively avoid collision when the number of AGVs is not large[40].

There are many other ways to avoid collisions, including the use of the vector field histogram method[41], the dynamic windows approach[42], and Optimal Reciprocal Collision Avoidance (ORCA)[43]. The vector field histogram method solves the problem of obtaining the AGV path between two close obstacles by calculating the obstacle density. By establishing the solution space for the AGV feasible vector, the dynamic windows approach provides AGVs with the driving direction and distance without collision in unit time. In ORCA, multiple AGVs perceive each other's positions and calculate the collision boundary. Furthermore, the AGV does not touch the collision boundary to avoid a collision.

Considering the path planning problem with collision, three problems must be solved:

(1) How should the specific route be represented? Only when the route is represented can the collision be detected by comparison or judgment.

(2) How to detect collisions? The representation of the path can demonstrate the action track of AGV on the map. However, whether to pass through the intersection at the same time should also be considered apart from the coincidence of the action trajectories of different AGVs.

(3) How to reduce collision and solve collision problems? Collisions will always occur when multiple AGVs are running in the same matrix workshop. Reducing collisions can mitigate the computational complexity of AGV path planning. Related to this, solving the collision problem is the basis for the normal operation of multiple AGVs.

Although scholars have conducted numerous studies on the scheduling problem of AGVs, few have considered task assignment and path planning simultaneously. Considering that task assignment alone cannot avoid the collision in the actual operation of AGV. Similarly, considering path planning alone cannot meet the constraints existing in actual production, such as time constraints and capacity constraints. The following are the practical limitations of studying the task allocation and path planning problems of AGV separately.

(1) Studying the assignment problem of AGVs alone can maximize AGVs for transportation and increase the transportation efficiency of AGVs. However, in the production workshop, collisions are inevitable when the number of AGVs increases. Thus, AGVs cannot achieve the transportation effect studied in actual production if only the task allocation of AGVs is considered and not the accompanying collision issues.

(2) Ensuring the transportation efficiency of AGVs is difficult by simply solving the path planning problem through classical optimization algorithms. Current research on path planning for AGVs rarely considers that these are responsible for multiple tasks simultaneously. Thus, when there are many AGVs in the workshop, classical path planning algorithms are difficult to solve in a short time.

The present paper investigates the integrated scheduling of the task assignment problem and the path planning problem of collision avoidance. Taking a matrix workshop of Foxconn as the research background, the route is initialized using the Manhattan distances method. In the task assignment stage, we use the mathematical model established by Zou et al.[3] to optimize the scheduling scheme.

In the path planning stage, we consider path generation, collision detection, and collision avoidance. In path generation, the path is represented by a set of coordinate points. In collision detection, we can determine whether there is a collision by comparing potentially similar coordinate points in the paths of different AGVs simultaneously. In collision avoidance, we propose an AGV running direction priority strategy to effectively reduce the occurrence of AGV collision. At the same time, we establish a stacked time departure heuristic to reduce the time of vehicle collisions.

The main contributions of this paper are as follows. First, this paper proposes a joint scheduling method for AGV task allocation and path planning, which is unlike the classical AGV scheduling problem. Second, in the case of multiple AGVs driving in the workshop, common path-planning algorithms are no longer applicable. To solve this problem, the present paper proposes a path generation rule that establishes priority for driving directions to effectively reduce collisions. Finally, this paper reduces the number of AGVs present in the workshop simultaneously by delaying departure, effectively reducing collisions and eliminating the difficulty of path planning.

The remainder of the article is organized as follows. In Section 2, the specific problem of the matrix workshop is described, and a mathematical model based on this problem is established in Section 3. The referenced Discrete Invasive Weed Optimization (DIWO) algorithm, as well as a heuristic, are proposed in Section 4. Section 5 presents and analyzes the computational results. The conclusions are provided in Section 6.

2 Integrated Scheduling Problem

The layout of the Foxconn matrix production workshop consists of multiple independent workstations and AGVs. Workstations are distributed throughout the workshop in a matrix manner. An AGV is used to distribute the materials required for production to the workstation for processing, after which the AGV returns to the depot and waits for the next instruction. Figure 1 depicts the matrix workshop.

The working mode of an AGV in the matrix workshop is described as follows. The workshop has a total of 10 rows, each consisting of 10 workstations. Each workstation consists of multiple Computer Numerical Control (CNC) machines and a material buffer. CNC machines are used for processing, while the material buffer is used to store the materials required for processing. Each workstation is set up next to the aisle to facilitate the transportation of materials. When the workstation is short of materials, it sends a call to request the distribution of materials. This kind of workstation is called a "task". When an AGV receives a call, it carries materials from the depot and drives on the aisle. The AGV returns to the depot once the requested material is sent to the designated workstation.

2.1 Capacity and time constraints

In relation to the multi-AGV integrated scheduling problem, the AGV distribution problem should be considered first. Here, certain constraints must be satisfied to be able to assign multiple tasks to multiple AGVs. This paper mainly considers the capacity constraint of AGVs and the latest delivery constraint of tasks.

There is a fixed upper limit on the transport capacity of an AGV, and tasks are randomly sorted and assigned to the AGVs. Here, the system tries to assign the first task to the first AGV, and assesses whether the materials carried by the AGV exceed the capacity. If

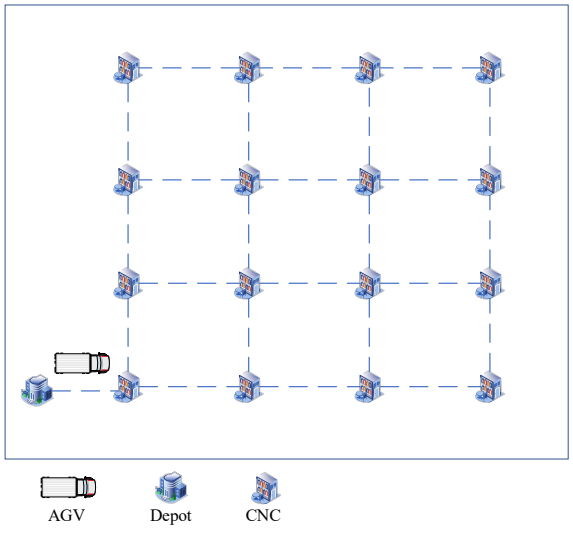

Fig. 1 Matrix workshop.

the capacity is not exceeded, the second task will be assigned to the AGV, and the capacity will continuously be checked. When the i-th task is assigned to the j-th AGV and the capacity is exceeded, the i-th task will be assigned to the (j+1)-th AGV. In this way, multiple tasks can be assigned to multiple AGVs to meet capacity constraints.

In actual production, the call time and the latest delivery time are different for each task. In a multi-AGV scheduling problem, each AGV must be responsible for the distribution of multiple tasks. Certain constraints must be met when assigning tasks in order to ensure that each task can be delivered on time. When the i-th task is assigned to the j-th AGV, and if the AGV cannot meet the requirement that the i-th task be delivered on time, the i-th task will be assigned to the (j+1)-th AGV. Thus, using the above method, the constraint of the latest delivery time of the task can be met.

In the scheduling process, it is obvious that considering the above two constraints separately cannot achieve the purpose of optimization. Therefore, this paper considers two constraints simultaneously. In the process of task assignment, if any constraint cannot be met, the task will be assigned to a new AGV.

2.2 Collision constraint

In the matrix production workshop, the aisle between two adjacent workstations only runs AGVs with the same driving direction at the same time. When two AGVs in different directions appear on this aisle simultaneously, a collision is likely to occur.

To consider the collision constraint, we must first be able to judge the collision. We express the path in the form of a coordinate point set by taking the matrix workshop as the coordinate system and the workstation as the coordinate point. Here, (2, 3, 0, 21, 1) is a coordinate point in the path, where 2 and 3 represent X and Y coordinates, respectively; 0 indicates that the running direction of AGV is upward, down, left, and right as represented by 1, 2, and 3, respectively; 21 represents the moment when AGV runs to the specified coordinates; and 1 indicates that this coordinate point belongs to the driving path of the first AGV. The path of each AGV can be represented through the collection of coordinate points. Using this method, we can easily detect the collision by comparing the paths of different AGVs to determine whether two AGVs are traveling in different directions on the same aisle at a certain time.

3 Mathematical Model

This part introduces the establishment of a multi-AGV integrated scheduling model in the matrix workshop.

3.1 AGV task assignment model

We have established the mathematical model with time and capacity constraints, as described above. The parameters and decision variables are as follows.

(1) K: It represents the set of AGVs.

(2) T: It represents the set of tasks. Notably, $t \in T$; t^{call} and t^{latest} represent the call time and the latest delivery time of the task, respectively; and t^{num} represents the number of materials called.

(3) L: It represents the location of the task. Each location $l_i \in L$ contains x_i and y_i coordinates that can be recorded as $l_i = (x_i, y_i)$.

(4) S: It represents the amounts of materials required for the buffer.

(5) Decision variables:

x_{ijk}: $x_{ijk} = 1$ if arc (i, j) is traveled by AGV k, and 0 otherwise. In addition, the parameter settings are shown in Table 1, and the constant settings are shown in Table 2.

From the variables given above, the problem can be defined as follows:

$$d_{ij} = \left|x_i - x_j\right| + \left|y_i - y_j\right| \tag{1}$$

$$t_{ij} = d_{ij}/v \tag{2}$$

Table 1 Parameters for AGV task assignment model.

Parameter	Specific description
n	Total number of tasks
m	Total number of AGVs used
d_{ij} (m)	Distance between tasks i and j
t_{ij} (s)	Travel time between tasks i and j
c_{a}	Loss cost per AGV
c_{d}	Distance cost per AGV
c_{t}	Time penalty cost per AGV
W_j^{r}	Time when the AGV arrives at task j

Table 2 Constants for AGV task assignment model.

Constant	Specific description
v (m/s)	Speed of the AGV
Q	Capacity of AGV
W_0	Departure time of AGV
W_{u}	Unloading time of AGV
W_{p} (s)	Processing time required per unit of material
g	Weight per unit of material

$$W_j^{\mathrm{r}} = W_i^{\mathrm{r}} + t_{ij} + W_{\mathrm{u}},\ i \in T,\ j \in T,\ i \neq j \tag{3}$$

$$t_j^{\mathrm{num}} = \left[S_j + \left\lceil \left(W_j^{\mathrm{r}} - t_j^{\mathrm{call}}\right) \right\rceil / W_{\mathrm{p}}\right] \times g,\ j \in T \tag{4}$$

Equation (1) is used to calculate the distance between task i and task j, while Eq. (2) is used to obtain the travel time of AGV between tasks i and j. Furthermore, Eq. (3) determines when the AGV arrives at task j, while Eq. (4) determines the number of materials to be delivered by AGV for task j.

3.2 AGV path planning model

The modeling of path planning takes the matrix workshop as the background, and the path is set by Manhattan distances. The specific parameters and decision variables are as follows.

(1) P: It represents the path point of AGVs. Notably, $p \in P$; p^x and p^y represent the x and y coordinates of the path point, respectively; p^k represents which AGV the path point belongs to; p^{t} represents the time when the AGV reaches the path point; and p^{w} represents the direction of travel when the AGV passes the path point.

(2) Z^{t}: It represents the time required for the next AGV to avoid the first AGV when the collision is predicted.

(3) H^k: It represents the number of path points of AGV k.

(4) c_h^k: It is a decision variable. $c_h^k = 1$ if AGV needs to avoid the vehicle's collision, and 0 otherwise.

The problem can be defined with the variables given above:

$$p^k = k,\ p^k \in P \tag{5}$$

$$0 \leqslant p^x \leqslant 10,\ p^x \in P \tag{6}$$

$$0 \leqslant p^y \leqslant 10,\ p^x \in P \tag{7}$$

$$p_h^{\mathrm{t}} \leqslant p_{h+1}^{\mathrm{t}},\ p_h^{\mathrm{t}} \in P,\ p_{h+1}^{\mathrm{t}} \in P \tag{8}$$

$$0 \leqslant p^{\mathrm{w}} \leqslant 3,\ p^{\mathrm{w}} \in P \tag{9}$$

Constraint of Eq. (5) ensures that the route points belong to the same AGV. Constraints of Formulas (6) and (7) ensure that the path point is within the specified area and will not exceed the limit. In addition, constraint of Formula (8) ensures that the sequence of AGV driving on the path point will not be confused, while constraint of Formula (9) ensures that the AGV will only run in four directions: 0 for upward, 1 for down, 2 for left, and 3 for right.

3.3 Model objectives and constraints

The main objective of this paper is to reduce the transportation cost of multi-AGV scheduling and reduce the impact caused by potential AGV collisions. The following equations can achieve those objectives:

$$C_{\mathrm{d}} = c_{\mathrm{d}} \sum_{k=1}^{m} \sum_{i=0}^{n} \sum_{j=0}^{n} x_{ijk} \times d_{ij} \tag{10}$$

$$C_{\mathrm{a}} = c_{\mathrm{a}} \sum_{k=1}^{m} \sum_{i=0}^{n} x_{0ik} \tag{11}$$

$$C_{\mathrm{t}} = c_{\mathrm{t}} \left[\left[\sum_{k=1}^{m} \sum_{i=0}^{n} \sum_{j=0}^{n} x_{ijk} \left(t_j^{\mathrm{latest}} - W_j^{\mathrm{r}}\right) \right] + \sum_{k=1}^{m} \sum_{h=1}^{H^k} c_h^k Z^{\mathrm{t}} \right] \tag{12}$$

$$\min F(i,\ j,\ k,\ h) = C_{\mathrm{a}} + C_{\mathrm{d}} + C_{\mathrm{t}} \tag{13}$$

Equations (10)–(13) represent the traveling cost of all AGVs, the cost of all AGV losses, the punishment of all AGVs for violating the time (including the punishment of early arrival), and the objective function, respectively.

The additional constraints are as follows:

$$\sum_{k=1}^{m} \sum_{i=0}^{n} x_{ijk} = 1,\ j \in T,\ i \neq j \tag{14}$$

$$\sum_{k=1}^{m} \sum_{j=0}^{n} x_{ijk} = 1,\ i \in T,\ i \neq j \tag{15}$$

$$\sum_{i=1}^{n} \sum_{j=0}^{n} x_{ijk} \times t_j^{\mathrm{num}} < Q,\ k \in K \tag{16}$$

$$\sum_{k=1}^{m} \sum_{j=0}^{n} x_{ijk} \times t^{\mathrm{c}} < W_j^{\mathrm{r}} < \sum_{k=1}^{m} \sum_{j=0}^{n} x_{ijk} \times t^{\mathrm{d}} \tag{17}$$

Constraints of Eqs. (14) and (15) ensure that a task is transported by only one AGV, constraint of Formula (16) ensures that each AGV is not overloaded, and constraint of Formula (17) ensures that the arrival time of each AGV is between the call time and the most recent delivery time.

4 Proposed Optimization Method

This section introduces the optimization method proposed for AGV integrated scheduling. The DIWO algorithm used will be introduced first. Then, a heuristic is explained to quickly judge the feasible solution. After that, a running direction priority rule is proposed, along with a stacked time departure

heuristic.

4.1 DIWO algorithm

The DIWO algorithm shows good performance in our previous investigation into multi-AGV task assignments[44]. In the present paper, we once again use the DIWO algorithm to solve the AGV integrated scheduling problem. The detailed procedure of the DIWO algorithm is shown in Algorithm 1.

During initialization, one high-quality individual is generated by the Nearest-Neighbor-based Heuristic (NNH)[3], and the rest are generated randomly. In the procedure, the number of new individuals is obtained based on the objective value of the individual population ($\text{seed} = \frac{f_{\text{cur}} - f_{\min}}{f_{\max} - f_{\min}} \times (S_{\max} - S_{\min}) + S_{\min}$, $f_{\min}$ and $f_{\max}$ represent the minimum and maximum objective function value of population, f_{cur} represents the objective function value of current individual, $S_{\max}$ and $S_{\min}$ represent the maximum and minimum number of seeds, respectively). Then, every individual generates a corresponding number of new individuals through insertion and exchange operators. The specific details are porvided in Algorithm 2.

Local search is based on the insertion neighborhood, in which a task is selected randomly and inserted in all positions. If the solution obtained is better than the current solution, then the current solution is replaced.

4.2 Heuristic for the rapid judgment of the feasible solution

Due to time and capacity constraints, infeasible solutions are inevitable. The rapid detection of infeasible solutions can effectively improve computational efficiency. A heuristic for quickly judging feasible solutions is introduced below.

In constraint of Formula (16), each AGV cannot be overloaded. Then, when the task changes, the reduction of the task will not lead to the overloading of the AGV. In this case, the increase in the number of tasks has become the main reason for AGV overload.

In constraint of Eq. (4), the material demand of each task also includes the part consumed by the machine during transportation. This heuristic can judge whether AGV is overloaded in most cases, after which the system assesses whether or not it is an infeasible solution. The implementation of this heuristic is given in Heuristic 1.

Algorithm 1 DIWO

Initialization: population with PS_0 weeds
Procedure:
while (termination criterion)
  Calculate the number of new individuals;
  Generate new individuals;
  Evaluate all new individuals;
  Initiate a local search for the best new individual;
  Poor individuals are competitively excluded;
  Initiate a local search for the current best individual;
end while
Report the best solution found.

Algorithm 2 Local search

Procedure:
Set l_i the selected task and position p=0;
for each position p in P
  Set l_i to p in P;
  if the new solution meets the constraints
    Calculate the objective value of the solution;
    if the new solution is better
      Replace the old with the new solution;
    end if
  end if
end for
Report the solution.

4.3 Path generation

Current research on path planning focuses on generating the shortest path to reach the target point on the map and avoiding obstacles in the process of traveling. There are multiple paths to ensure the shortest driving distance of AGV, especially in the

Heuristic 1 Rapid judgement

Initialization: $i, j, Q' = 0$
Procedure:
for task i in AGV j
  $t_i^{\text{m}} = S_i \times g$;
  $Q'_j = Q'_j + t_i^{\text{m}}$;
  if $Q'_j < Q$
    AGV j is not overloaded;
    Continue;
  else
    AGV j is overloaded;
    break
  end if
end for
Report the status.

absence of obstacles. There are two shortest paths, as shown in Fig. 2.

In the matrix workshop, the time calculation cost would be too high if each AGV generates the path through algorithms. In this paper, the path is generated by formulating the rules of AGV driving direction priority, effectively reducing the time and calculation cost. As shown in Fig. 2, the blue and yellow paths are generated in a top-down priority and left-right priority manner, respectively. Under the top-down priority rule, the AGV drives up to the same line as the task point and then drives laterally. Given that AGVs traveling in the same direction will not collide at the same speed, this rule can also effectively reduce the number of collisions in multi-AGV scheduling.

4.4 Stacked time departure heuristic

The collision of multi-AGV scheduling in matrix workshops occurs between AGVs. Thus, an effective way to avoid collision is to reduce the number of AGVs running in the workshop simultaneously. In this paper, taking the loss cost of AGV as a part of the objective function can effectively reduce the number of scheduled AGVs. We propose a stacked time departure heuristic to further reduce the number of AGVs running in the workshop at the same time. This heuristic method is inspired by Ref. [44]. The implementation of this heuristic is given in Heuristic 2.

By calculating the early arrival time of all tasks belonging to AGV k, the minimum early arrival time can be selected as the delayed departure time. As shown in Eq. (4), the delay of departure will lead to an increase in AGV carrying capacity. Thus, it is necessary to judge whether the AGV is overloaded after a delayed departure. In the case of overload, it is necessary to calculate the amount of overload and how long it would take to start early to eliminate the overload.

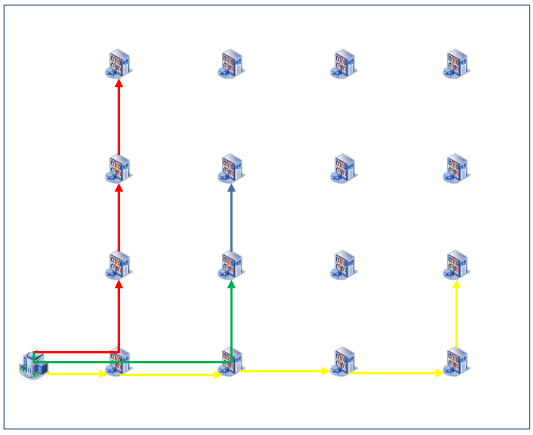

Fig. 2 Path generation.

Heuristic 2 Stacked time departure

Initialization: $i, j, k, t' = 0, t''=\infty$

Procedure:

1: Let H^k be the number of tasks in AGV k;
2: **for** task i in AGV k
3: $t' = t_i^{\mathrm{d}} - W_i^{\mathrm{r}}$;
4: **if** $t' < t''$
5: $t'' = t'$;
6: **else**
7: Continue;
8: **end if**
9: **end for**
10: $t_i^{\mathrm{m}} = \left[S_i + \left\lceil \left(W_i^{\mathrm{r}} - t_i^{\mathrm{call}} + t'\right) \right\rceil / W_{\mathrm{p}} \right] \times g$;
11: **if** $\left(\sum_{i=0}^{H^k} \sum_{j=0}^{H^k} x_{ijk} \times t_i^{\mathrm{num}}\right)$
12: Continue;
13: **else**
14: $t' = \left[\left(Q - \sum_{i=0}^{H^k} \sum_{j=0}^{H^k} x_{ijk} \times t_i^{\mathrm{num}}\right) / g \times W_{\mathrm{p}}\right] / H^k$;
15: $t'' = t'' - t'$;
16: Report t''.

4.5 Joint scheduling execution process

In the joint scheduling problem, this paper first solves the assignment problem of AGV. In particular, this paper addresses the joint scheduling problem of AGV in two stages. First, we use the DIWO algorithm to find a better task allocation solution for an AGV. A delayed departure heuristic method is used based on the obtained task allocation scheme to determine the departure time for each AGV. Then, the number of AGVs running simultaneously in the workshop is reduced by setting different departure times for AGVs. Finally, specific driving routes are formulated for each AGV using the proposed path generation method and directional priority rules. The process described above is shown in Fig. 3.

5 Case Analysis

We propose an optimization method and some optimization strategies. In this section, we will verify whether or not the optimization method under the model is effective.

First, we introduce the instances collected by the actual factory. Then, we compare the DIWO algorithm

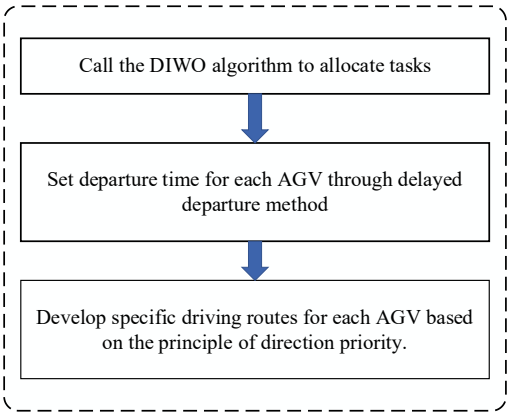

Fig. 3 Joint scheduling execution process.

with and without the optimization method. Finally, we compare other algorithms used for similar research problems to further verify the effectiveness of the proposed method.

5.1 Instances and experimental results

According to Zou et al.[3], the matrix workshop can be divided into countless continuous production cycles, and the tasks collected in each production cycle can be scheduled for the next production cycle. The number of tasks collected in each cycle may vary and can be divided into 10, 20, 30, 40, and 50 tasks. Therefore, the experiments are carried out from different task numbers. The first and second experiments compare the DIWO algorithm with and without the optimization strategies, respectively. An instance is selected for the introduction, and its specific parameters are shown in Table 3. Each instance is collected in a Foxconn matrix production workshop within 6 minutes.

In the second part, the comparative experiments of different algorithms are shown. The comparison algorithms are those with better current effects, such as Discrete Artificial Bee Colony (DABC)[3], Artificial Bee Colony (ABC)[45], Harmony Search (HS)[38], and Iterative Greedy (IG)[46]. The specific parameter settings of the algorithm are shown in Table 4, where PS denotes population size; l denotes a predetermined number of trials; r denotes a predetermined number; τ denotes the number of replications; HMS denotes harmony memory size; HMCR denotes harmony memory considering rate; d denotes the number of deletion tasks; OperIter denotes the number of iterations in the local search phase.

A total of 100 instances are used as the objects of the experiment. Each instance is replicated 10 times and compared with the average value to avoid the contingency of the experiment and to make the results more realistic. The execution time for each instance is 5 seconds. The experiment is conducted on an Intel (R) Core (TM) i7-8550U CPU @ 1.80 GHz with 8.00 GB RAM in a Windows 10 Operation System. The experimental results are compared by the Relevant Percentage Decrease (RPD) as follows:

$$\mathrm{RPD} = \left(\frac{f_{\mathrm{cur}} - f_{\mathrm{min}}}{f_{\mathrm{min}}}\right) \times 100 \tag{18}$$

The final results are analyzed by multi-factor analysis of variance.

5.2 Results and discussion

In this section, we present the optimization results of the previously mentioned instances in the form of graphs and tables. First, the effectiveness of the optimization strategies is verified and analyzed. Then, we compare the proposed method with other heuristic optimization algorithms and analyze the performance of different algorithms. Based on the above results, we present a summary of the performances of the multi-

Table 3 Task properties.

Task	Station number	Coordinate X	Coordinate Y	Call time (s)	Material required	Latest delivery time (s)
1	69	7	9	14	28	614
2	10	1	10	70	29	700
3	35	4	5	97	28	697
4	45	5	5	115	28	715
5	59	6	9	136	30	796
6	47	5	7	190	28	790
7	19	2	9	230	30	890
8	61	7	1	263	30	923
9	54	6	4	310	28	910
10	77	8	7	348	30	1008

Table 4 Parameter settings.

Algorithm	Parameter
DIWO	$PS_0 = 50$, $PS_{max} = 70$, $S_{max} = 15$, $S_{min} = 1$
DABC	PS = 150, $l = 800$, $r = 80$, and $\tau = 20$
ABC	PS = 50 and $l = 1200$
IG	$d = 5$ and OperIter = 60
HS	HMS = 4, $HMCR_{min} = 0.2$, and $HMCR_{max} = 0.8$

AGV integrated scheduling optimization methods.

5.2.1 Verification of optimization strategies

The experimental results of the optimization strategies mentioned in Section 4 are shown in Table 5. The results show the DIWO algorithm with the optimization method and the $DIWO_{bas}$ algorithm without the optimization method. The first column represents the number of tasks, each with 20 instances. The average values of RPD and number of collisions (COL) of instances with the same number of tasks are listed subsequently.

To verify the effectiveness of the strategies, this paper conducts experimental comparisons by combining the strategies separately. DIWO1 includes all optimization methods, DIWO2 does not include the stacked time departure heuristic, DIWO3 excludes local search, and DIWO4 excludes the stacked time departure heuristic and local search.

Table 5 also shows the specific data for each DIWO, while Figs. 4 and 5 present the comparative trends of RPD and collision frequency, respectively. As shown in Figs. 4 and 5 and Table 5, DIWO3 with the stacked time departure heuristic is closer to DIWO1, while DIWO2 without the stacked time departure heuristic is worse than DIWO3, thus proving that this heuristic plays a major role in the optimization. Similarly, DIWO2 and DIWO4 validate the effectiveness of local search.

Overall, in the AGV scheduling process, the stacked time departure heuristic can reduce the punishment of AGVs by controlling the departure time of different AGVs, while simultaneously reducing the number of AGVs in the workshop. Doing so drastically reduces the possibility of collisions.

Table 5 RPD of different strategies of DIWO algorithm.

Number of tasks	DIWO1	DIWO2	DIWO3	DIWO4
10	**0.64**	11.21	0.69	11.19
20	**6.50**	21.54	11.76	25.10
30	**2.99**	24.43	14.72	29.19
40	**2.60**	24.61	20.02	33.56
50	**2.51**	28.24	24.74	36.72
Average	**3.05**	22.02	14.38	27.16

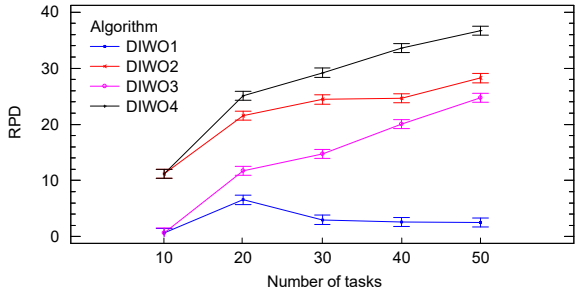

Fig. 4 RPD comparison of the optimization methods.

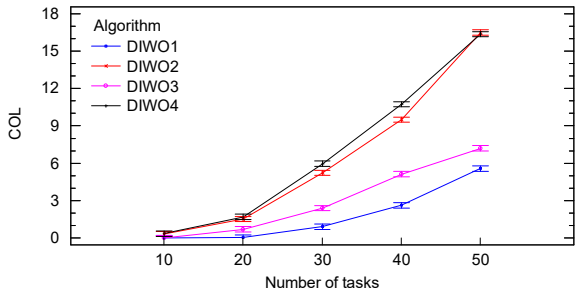

Fig. 5 Collision comparison of the optimization methods.

In Table 6, it is apparent that the DIWO algorithm with the optimization method performs better in all instances. In terms of RPD value, DIWO obtains a better value than $DIWO_{bas}$. According to the experimental results, under the condition of meeting the capacity and time constraints simultaneously, an AGV can be responsible for about 9 tasks. Therefore, when the instance contains 10 tasks, the probability of collision is relatively small because there are only two AGVs in the workshop for transportation. Thus, the collision can be avoided when the optimization method is implemented. When the task increases gradually, the optimization method can reduce the collision by 50%. RPD and COL are shown in Figs. 6 and 7, respectively.

Table 6 RPD and COL of the proposed DIWO and the base DIWO.

Number of tasks	DIWO		$DIWO_{bas}$	
	RPD	COL	RPD	COL
10	**0.63**	**0**	11.05	0.35
20	**5.81**	**0.67**	20.67	1.49
30	**2.99**	**2.08**	24.85	5.35
40	**2.62**	**4.77**	25.44	9.61
50	**2.40**	**7.74**	28.49	16.09
Average	**2.89**	**3.64**	22.10	6.58

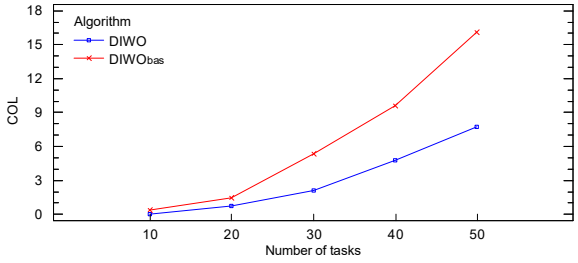

Fig. 6 Collision comparison of the optimization methods.

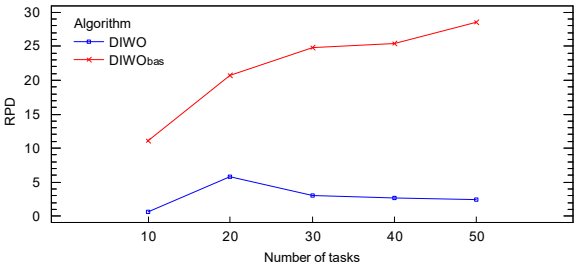

Fig. 7 Validation results of the optimization methods.

5.2.2 Verification of the comparison algorithms

The purpose of this case study is to verify the effectiveness of the proposed DIWO algorithm. Thus, to fully prove its effectiveness, the other four algorithms that perform well in related fields are compared, as shown in Figs. 8 and 9.

As can be seen, all algorithms generate paths according to the rule of top-down priority rule, which ensures that all AGVs will not collide until they reach the first task for which they are responsible. This section will no longer verify the effectiveness of the top-down priority. The experimental results are shown in Table 7. From Table 7, it is obvious that the DIWO algorithm obtains the minimum RPD in instances of all sizes. Similar to previous cases, the DIWO algorithm also has excellent performance in reducing the number of collisions. In particular, the average collision times of the DIWO algorithm is one-third that of the worst ABC algorithm. Furthermore, its average number of collisions is nearly 40% less than the second-best DABC algorithm.

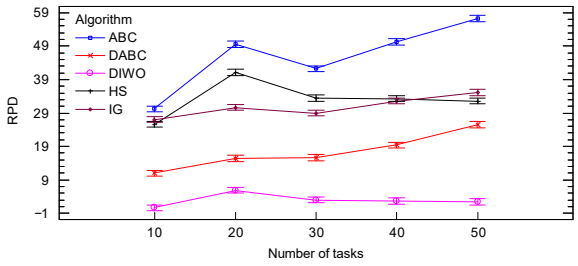

Fig. 8 Validation results of the algorithms.

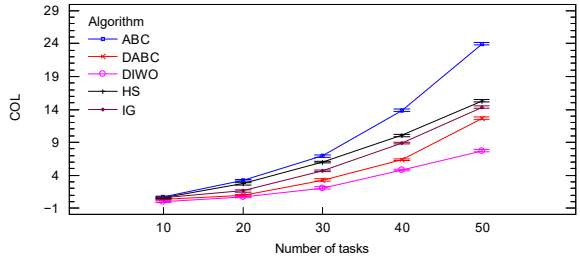

Fig. 9 Collision comparison of the optimization algorithms.

Table 8 shows the results of the Wilcoxon paired signed test of the proposed DIWO algorithm compared with other algorithms. We calculate 100 instances for 10 times and got 1000 samples. R+ represents the number of times DIWO algorithm is superior to other algorithms in 1000 samples. R− represents the number of times DIWO algorithm is inferior to other algorithms in 1000 samples. The sum of Bonding Value, R+ and R− is equal to 1000. The experimental results indicate a significant difference between the proposed and the comparative algorithms. Furthermore, the optimization results clearly indicate that the optimization method proposed in this paper can effectively reduce the transportation cost of multi-AGV scheduling, reduce the number of collisions, and increase the safety of AGV operation.

6 Conclusion

This paper investigates an optimization method of multi-AGV integrated scheduling in a matrix

Table 7 RPD amd COL of different optimization algorithms.

Instance	DABC		ABC		DIWO		IG		HS	
	RPD	COL	RPD	COL	RPD	COL	RPD	COL	RPD	COL
T10I1	10.98	0.34	30.20	0.71	**0.63**	**0**	27.05	0.54	25.60	0.62
T10I2	15.39	0.95	49.63	3.21	**5.86**	**0.74**	30.65	1.62	41.15	2.71
T10I3	15.66	3.28	42.25	6.92	**2.99**	**2.08**	28.99	4.7	33.46	6.01
T10I4	19.40	6.35	50.24	13.87	**2.62**	**4.77**	32.57	8.89	33.29	10.05
T10I5	25.58	12.65	57.22	23.96	**2.40**	**7.74**	35.15	14.35	32.59	15.31
Average	17.40	4.71	45.91	9.73	**2.90**	**3.06**	30.88	6.02	33.22	6.94

Table 8 Results achieved by the Wilcoxon paired signed test (α = 0.05).

Comparison	R+	R-	*p*-value
DIWO versus DABC	952	48	0
DIWO versus ABC	999	1	0
DIWO versus HS	998	2	0
DIWO versus IG	997	3	0

production workshop. The goals are to reduce the transportation cost and AGV collision as well as to increase the security of multi-AGV scheduling. In this method, we used an optimized DIWO algorithm to calculate the objective function. To improve the local search ability of the swarm intelligence algorithm, we also used an insertion-based local search. In accordance with the established mathematical model, a heuristic to quickly judge the feasible solution is proposed. In the process of path generation, we established a direction priority rule that can reduce collision to a certain extent. A stacked time departure heuristic is also proposed to avoid collisions by reducing the number of AGVs in the matrix workshop at the same time. Finally, we proved the effectiveness of this optimization method by conducting comprehensive experiments.

The research of this paper has some limitations. First, this paper studies the multi-AGV scheduling problem based on a matrix workshop. In path planning, an aisle is restricted to AGVs in the same direction simultaneously. While this regulation can solve the collision problem to a great extent, it can also affect the transportation efficiency of AGV to a certain extent. In our future work, we will consider multi-AGV scheduling in the workshop with obstacles along with deadlock constraints.

Acknowledgment

This work was supported by the National Natural Science Foundation of China (Nos. 62273221 and 52205529) and the Discipline with Strong Characteristics of Liaocheng University Intelligent Science and Technology (No. 319462208).

References

[1] 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, 2020.

[2] S. Riazi, K. Bengtsson, and B. Lennartson, Energy optimization of large-scale AGV systems, *IEEE Trans. Automat. Sci. Eng.*, vol. 18, no. 2, pp. 638–649, 2021.

[3] W. Q. Zou, Q. K. Pan, T. Meng, L. Gao, and Y. L. Wang, An effective discrete artificial bee colony algorithm for multi-AGVs dispatching problem in a matrix manufacturing workshop, *Expert Syst. Appl.*, vol. 161, p. 113675, 2020.

[4] W. Q. Zou, Q. K. Pan, and L. Wang, An effective multi-objective evolutionary algorithm for solving the AGV scheduling problem with pickup and delivery, *Knowl. Based Syst.*, vol. 218, p. 106881, 2021.

[5] W. Q. Zou, Q. K. Pan, and M. F. Tasgetiren, An effective iterated greedy algorithm for solving a multi-compartment AGV scheduling problem in a matrix manufacturing workshop, *Appl. Soft Comput.*, vol. 99, p. 106945, 2020.

[6] H. Hu, X. Chen, T. Wang, and Y. Zhang, A three-stage decomposition method for the joint vehicle dispatching and storage allocation problem in automated container terminals, *Comput. Ind. Eng.*, vol. 129, pp. 90–101, 2019.

[7] J. Luo and Y. Wu, Scheduling of container-handling equipment during the loading process at an automated container terminal, *Comput. Ind. Eng.*, vol. 149, p. 106848, 2020.

[8] G. Fragapane, R. de Koster, F. Sgarbossa, and J. O. Strandhagen, Planning and control of autonomous mobile robots for intralogistics: Literature review and research agenda, *Eur. J. Oper. Res.*, vol. 294, no. 2, pp. 405–426, 2021.

[9] Q. K. Pan, L. Gao, and L. Wang, An effective cooperative co-evolutionary algorithm for distributed flowshop group scheduling problems, *IEEE Trans. Cybern.*, vol. 52, no. 7, pp. 5999–6012, 2022.

[10] X. L. Jing, Q. K. Pan, L. Gao, and L. Wang, An effective iterated greedy algorithm for a robust distributed permutation flowshop problem with carryover sequence-dependent setup time, *IEEE Trans. Syst. Man. Cybern. Syst.*, vol. 52, no. 9, pp. 5783–5794, 2022.

[11] F. Wang, Y. Li, and J. Chen, Bi-level programming model for post-disaster emergency supplies scheduling with time windows and its algorithm, *Int. J. Autom. Control*, vol. 16, no. 1, pp. 45–63, 2022.

[12] E. Jiang, L. Wang, and J. Wang, Decomposition-based multi-objective optimization for energy-aware distributed hybrid flow shop scheduling with multiprocessor tasks, *Tsinghua Science and Technology*, vol. 26, no. 5, pp. 646–663, 2021.

[13] L. Meng, K. Gao, Y. Ren, B. Zhang, H. Sang, and C. Zhang, Novel MILP and CP models for distributed hybrid flowshop scheduling problem with sequence-dependent setup times, *Swarm Evol. Comput.*, vol. 71, p. 101058, 2022.

[14] A. Mohammad, O. Saleh, and R. A. Abdeen, Occurrences algorithm for string searching based on brute-force algorithm, *J. Comput. Sci.*, vol. 2, no. 1, pp. 82–85, 2006.

[15] O. Karasakal, L. Kandiller, and N. E. Özdemirel, A branch and bound algorithm for sector allocation of a naval task group, *Nav. Res. Logist.*, vol. 58, no. 7, pp. 655–669, 2011.

[16] W. C. Yeh, An efficient branch-and-bound algorithm for

the two-machine bicriteria flowshop scheduling problem, *J. Manuf. Syst.*, vol. 20, no. 2, pp. 113–123, 2002.

[17] H. Zhang, J. Xie, J. Ge, J. Shi, and Z. Zhang, Hybrid particle swarm optimization algorithm based on entropy theory for solving DAR scheduling problem, *Tsinghua Science and Technology*, vol. 24, no. 3, pp. 282–290, 2019.

[18] X. Wang, L. Wang, S. Wang, J. Pan, H. Ren, and J. Zheng, Recommending-and-grabbing: A crowdsourcing-based order allocation pattern for on-demand food delivery, *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 1, pp. 838–853, 2023.

[19] F. Ming, W. Gong, L. Wang and L. Gao, Balancing convergence and diversity in objective and decision spaces for multimodal multi-objective optimization, *IEEE Trans. Emerg. Top. Comput. Intell.*, vol. 7, no. 2, pp. 474–486, 2023.

[20] X. Wang, Y. Dai, L. Wang, and Z. Jia, Transient analysis and scheduling of bernoulli serial lines with multi-type products and finite buffers, *IEEE Trans. Automat. Sci. Eng.*, vol. 20, no. 4, pp. 2367–2382, 2023.

[21] X. Wu, X. Xiao, and Q. Cui, Multi-objective flexible flow shop batch scheduling problem with renewable energy, *Int. J. Autom. Control*, vol. 14, nos. 5&6, pp. 519–553, 2020.

[22] F. Zhao, S. Di, and L. Wang, A hyperheuristic with Q-learning for the multiobjective energy-efficient distributed blocking flow shop scheduling problem, *IEEE Trans. Cybern.*, vol. 53, no. 5, pp. 3337–3350, 2023.

[23] M. Tan, Z. Zhang, Y. Ren, I. Richard, and Y. Zhang, Multi-agent system for electric vehicle charging scheduling in parking lots, *Complex Syst. Model. Simul.*, vol. 3, no. 2, pp. 129–142, 2023.

[24] Z. Li, H. Sang, Q. Pan, K. Gao, Y. Han, and J. Li, Dynamic AGV scheduling model with special cases in matrix production workshop, *IEEE Trans. Ind. Inf.*, vol. 19, no. 6, pp. 7762–7770, 2023.

[25] X. He, Q. K. Pan, L. Gao, L. Wang, and P. N. Suganthan, A greedy cooperative co-evolutionary algorithm with problem-specific knowledge for multiobjective flowshop group scheduling problems, *IEEE Transactions on Evolutionary Computation*, vol. 27, no. 3, pp. 430–444, 2023.

[26] Z. Shu, A. Song, G. Wu, and W. Pedrycz, Variable reduction strategy integrated variable neighborhood search and NSGA-II hybrid algorithm for emergency material scheduling, *Complex Syst. Model. Simul.*, vol. 3, no. 2, pp. 83–101, 2023.

[27] X. Zhang, H. Sang, Z. Li, B. Zhang, and L. Meng, An efficient discrete artificial bee colony algorithm with dynamic calculation method for solving the AGV scheduling problem of delivery and pickup, *Complex Intell. Syst*., doi: 10.1007/s40747-023-01153-w.

[28] Y. Fu, Y. Hou, Z. Wang, X. Wu, K. Gao, and L. Wang, Distributed scheduling problems in intelligent manufacturing systems, *Tsinghua Science and Technology*, vol. 26, no. 5, pp. 625–645, 2021.

[29] M. X. Tian, H. Y. Sang, W. Q. Zou, Y. T. Wang, M. P. Miao, and L. L. Meng, Joint scheduling of AGVs and parallel machines in an automated electrode foil production factory, *Expert Systems with Applications*, http://doi.org/10.1016/jeswa.2023.122197, 2024.

[30] S. Nesmachnow, An overview of metaheuristics: Accurate and efficient methods for optimisation, *Int. J. Metaheuristics*, vol. 3, no. 4, pp. 320–347, 2014.

[31] R. Choe, J. Kim, and K. R. Ryu, Online preference learning for adaptive dispatching of AGVs in an automated container terminal, *Appl. Soft Comput.*, vol. 38, pp. 647–660, 2016.

[32] J. Luo and Y. Wu, Modelling of dual-cycle strategy for container storage and vehicle scheduling problems at automated container terminals, *Transp. Res. E Logist. Transp. Rev.*, vol. 79, pp. 49–64, 2015.

[33] J. Luo, Y. Wu, and A. B. Mendes, Modelling of integrated vehicle scheduling and container storage problems in unloading process at an automated container terminal, *Comput. Ind. Eng.*, vol. 94, pp. 32–44, 2016.

[34] N. Wu and M. Zhou, Shortest routing of bidirectional automated guided vehicles avoiding deadlock and blocking, *IEEE/ASME Trans. Mechatron.*, vol. 12, no. 1, pp. 63–72, 2007.

[35] M. Saidi-Mehrabad, S. Dehnavi-Arani, F. Evazabadian, and V. Mahmoodian, An ant colony algorithm (ACA) for solving the new integrated model of job shop scheduling and conflict-free routing of AGVs, *Comput. Ind. Eng.*, vol. 86, pp. 2–13, 2015.

[36] Y. Yang, M. Zhong, Y. Dessouky, and O. Postolache, An integrated scheduling method for AGV routing in automated container terminals, *Comput. Ind. Eng.*, vol. 126, pp. 482–493, 2018.

[37] J. Euchi and A. Sadok, Hybrid genetic-sweep algorithm to solve the vehicle routing problem with drones, *Phys. Commun.*, vol. 44, p. 101236, 2021.

[38] G. Li, B. Zeng, W. Liao, X. Li, and L. Gao, A new AGV scheduling algorithm based on harmony search for material transfer in a real-world manufacturing system, *Adv. Mech. Eng.*, vol. 10, no. 3, pp. 1–13, 2018.

[39] S. C. Srivastava, A. K. Choudhary, S. Kumar, and M. K. Tiwari, Development of an intelligent agent-based AGV controller for a flexible manufacturing system, *Int. J. Adv. Manuf. Technol.*, vol. 36, no. 7, pp. 780–797, 2008.

[40] D. K. Liu and A. K. Kulatunga, Simultaneous planning and scheduling for multi-autonomous vehicles, in *Evolutionary Scheduling*, K. P. Dahal, K. C. Tan, and P. I. Cowling, eds. Berlin, Germany: Springer, 2007, pp. 437–464.

[41] A. Derhab, M. Belaoued, I. Mohiuddin, F. Kurniawan, and M. K. Khan, Histogram-based intrusion detection and filtering framework for secure and safe in-vehicle networks, *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 3, pp. 2366–2379, 2022.

[42] Y. Kang, D. A. De Lima, and A. C. Victorino, An approach of human driving behavior correction based on dynamic window approach, in *Proc. 2014 IEEE Intelligent Vehicles Symposium*, Dearborn, MI, USA, 2014, pp. 304−309.

[43] J. van den Berg, S. J. Guy, M. Lin, and D. Manocha, Reciprocal *n*-body collision avoidance, in *Proc. 14th International Symposium ISRR on Robotics Research*, Berlin, Germany, 2011, pp. 3–19.

[44] Z. K. Li, H. Y. Sang, X. J. Zhang, W. Q. Zou, B. Zhang, and L. L. Meng, An effective discrete invasive weed optimization algorithm for multi-AGVs dispatching problem with specific cases in matrix manufacturing workshop, *Comput. Ind. Eng.*, vol. 174, p. 108755, 2022.

Hongyan Sang received the MEng degree from Liaocheng University, China in 2010, and the PhD degree in industrial engineering from Huazhong University of Science Technology, China in 2013. Since 2003, she has been working at School of Computer Science, Liaocheng University, where she became a professor in 2021. Her current research interests include intelligent optimization and scheduling. She has authored more than 60 refereed papers.

Zhongkai Li received the BEng degree from North China University of Technology, China in 2016. He is currently a master student in software engineering at Liaocheng University, China. His current research interests include AGV's task assignment and scheduling.

[45] W. Y. Szeto, Y. Wu, and S. C. Ho, An artificial bee colony algorithm for the capacitated vehicle routing problem, *Eur. J. Oper. Res.*, vol. 215, no. 1, pp. 126–135, 2011.

[46] J. P. Huang, Q. K. Pan, L. Gao, and L. Wang, An effective iterated greedy algorithm for PCBs grouping problem to minimize setup times, *Appl. Soft Comput.*, vol. 112, p. 107830, 2021.

M. Fatih Tasgetiren received the BEng and MEng degrees in industrial engineering from Istanbul Technical University (ITU), Türkiye, and the PhD degree in production and operations management from Istanbul University, Türkiye. He is a professor at Industrial Engineering Department, Baskent University, Türkiye. His research focuses on modeling, analysis, and optimization of complex systems through the use of computational intelligence methods. He works on the design and development of modern meta-heuristic algorithms to solve discrete/combinatorial/binary, as well as real-parameter unconstrained/constrained optimization problems. His main research interest has been in sequencing and scheduling problems. His current Google academic citations are 10 340, with an h-index of 45.