Received 18 June 2024, accepted 17 July 2024, date of publication 23 July 2024, date of current version 31 July 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3432594

RESEARCH ARTICLE

An Effective Discrete Jaya Algorithm for Multi-AGVs Scheduling Problem With Dynamic Unloading Time

YINGYING CUI, BAOXIAN JIA, HONGYAN SANG, LEILEI MENG, BIAO ZHANG, AND WENQIANG ZOU
School of Computer Science, Liaocheng University, Liaocheng 252000, China

Corresponding authors: Baoxian Jia (jiabaoxian@lcu.edu.cn) and Wenqiang Zou (zouwenqiang@lcu-cs.com)

This work was supported in part by the National Natural Science Foundation of China under Grant 52205529, in part by the Natural Science Foundation of Shandong Province under Grant ZR2021QE195, in part by the Youth Innovation Team Program of Shandong Higher Education Institution under Grant 2023KJ206, in part by the Research Fund Project of Liaocheng University under Grant 318012110 and Grant 318052150, in part by the Cloud Native Database Architecture Innovation and High-Performance Core Technology Project under Grant ZTZB-23-990-024, and in part by Shandong Province Social Science Planning Digital Research Project ''Shandong Province Government Big Data Governance System Construction Effective Path Research'' under Grant 23CSDJ10.

ABSTRACT With the advance of Automated Guided Vehicles (AGVs) technology, the scheduling of multiple AGVs in a matrix manufacturing workshop has attracted considerable attention. However, little attention has been devoted to dynamic unloading time for multiple AGVs scheduling. This paper investigates a new multi-AGVs scheduling problem with dynamic unloading time ($\mathrm{MAGVS_{DUT}}$) in a matrix manufacturing workshop with the objective of minimizing the transportation cost, including travel cost, penalty cost, and vehicle cost. To solve $\mathrm{MAGVS_{DUT}}$, a mixed-integer linear programming model and a discrete Jaya (DJaya) algorithm are proposed. At first, a heuristic based on ant colony algorithm is designed to generate high-quality initial solution. And then, two DJaya operators are designed, one of which is a near optimal operator updating solutions towards better solutions found, while the other is the away worst operator updating solutions towards worst solutions. In addition, a sequence insertion operator is designed to help the population find better solutions within the global space. Finally, a battery of comparative experiments is conducted in conjunction with the actual situation of an electronic equipment manufacturing company. The computational results show that the proposed DJaya algorithm is superior to the existing algorithms in tackling the considered problem.

INDEX TERMS Matrix manufacturing workshop, multi-AGVs, dynamic unloading time, discrete Jaya algorithm, heuristic.

I. INTRODUCTION

With the rapid advancement of automatic guided vehicles (AGVs) technology, AGVs have gained extensive traction across diverse sectors including manufacturing, warehousing, logistics, healthcare, and other domains [1], [2], [3], [4], [5]. Especially in manufacturing, AGVs are often employed to carry out handling tasks to improve productivity [6]. Since employing a single AGV to complete all handling tasks is often time-consuming, the utilization of multiple AGVs working collaboratively has become inevitable [7]. To our knowledge, efficient scheduling of multiple AGVs is a feasible way for improving production efficiency and reducing production costs. When the AGVs unloaded the production materials at their designated site, if the characteristics of the problem are not considered, the unloading time can be ignored or set to a fixed time [8]. However, given the reality of the matrix manufacturing workshop, the unloading time can not be ignored or set to a fixed time, because it will lead to some production equipment

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

accidents. Currently, there are relatively few studies on the multi-AGVs scheduling problem with dynamic unloading time ($\mathrm{MAGVS_{DUT}}$) to optimize the cost and efficiency of matrix manufacturing workshops. Therefore, it is highly significant for manufacturing enterprises and researchers to study the $\mathrm{MAGVS_{DUT}}$ in a matrix manufacturing workshop.

The matrix manufacturing workshop represents a novel form of production facility that, in contrast to traditional workshop models, effectively caters to the demands of multi-variety, personalized, and small-scale manufacturing. The workshop consists primarily of three components: a depot, workstations, and AGVs. In the workshop, AGVs start from the depot, deliver production materials to the designated workstations, unload them considering the important factor of dynamic unloading time, and return to the depot after completing all tasks. The increase in the number of designated workstations will increase the complexity of solving the problem. As we all know, the AGV scheduling problem has been proved to be an NP-hard problem, so the $\mathrm{MAGVS_{DUT}}$ is also an NP-hard problem after adding the factor of dynamic unloading time. For an NP-hard problem, it is challenging to obtain the optimal solution using an exact algorithm within a specified time, while using heuristics and metaheuristics is considered to be one of the best approaches to finding the optimal or approximate optimal solutions [9]. As a metaheuristic, the discrete Jaya (DJaya) algorithm has demonstrated its powerful search capability in solving combinatorial optimization problems compared to other algorithms. Therefore, in this paper, an improved DJaya algorithm is proposed to solve the $\mathrm{MAGVS_{DUT}}$. The main achievements can be described as follows:

- Formulate the $\mathrm{MAGVS_{DUT}}$ and establish a mixed-integer linear programming model.
- Propose a heuristic based on ant colony algorithm to generate high quality initial solutions.
- Propose an efficient DJaya algorithm with advanced techniques, such as two DJaya operators in the updating mechanism for balancing algorithmic exploitation and exploration, and a sequence insertion operator for facilitating the population find better solutions.

The rest of paper is arranged as below. In Section II, we recall the literature intimately related to this issue. Section III formalizes problem. Section IV provides a brief introduction on the basic Jaya algorithm. This is followed by a presentation of the discretization of the DJaya algorithm in Section V. Section VI presents detailed experimental computational outcomes and algorithm comparisons, and at last, Section VII gives a comprehensive overview of the paper emphasizing the main discoveries and contributions.

II. RELATED LITERATURES

The problem considered is about AGV scheduling in the logistics of manufacturing systems, and many researchers have made significant contributions to it. Aiming at the scheduling problem of flexible assembly shop, Ge et al. [10] proposed an online scheduling method of multi-AGV system assembly shop based on shop static scheduling, and the AGV transportation system load was used as part of the objective function to establish the model. Aiming at the AGV flexible job-shop scheduling problem, Chen et al. [7] established a dual-resource integrated scheduling optimization model with the goal of minimizing the maximum completion time and proposed a hybrid discrete particle swarm optimization algorithm that can effectively avoid premature convergence. Meng et al. [11] addressed the distributed flexible job shop scheduling problem with minimizing maximum completion time. To improve the efficiency of the manufacturing system, Tian et al. [12] studied the joint scheduling problem of AGV and parallel machines in the automatic electrode foil production process and proposed a discrete grey Wolf optimization algorithm. To solve the joint production and transportation scheduling problem in flexible manufacturing systems, Fontes and Homayouni [13] utilized two sets of chain decisions connected to each other by the completion time constraints of machine operations and transportation tasks. Hu et al. [14] proposed a task allocation method based on adjacency combination and shortest path principle for conflict-free scheduling of large multi-load AGVs, and a heuristic search method based on variable neighborhood search was presented to optimize the multi-AGV task allocation problem. To improve the throughput performance of Automated Guided Vehicle (AGV) unmanned storage system, Tang et al. [15] established a two-stage mathematical model. A two-layer genetic algorithm was designed to optimize the task scheduling sequence of AGVs and picking stations. Niu et al. [16] aimed at optimizing AGVDP by fusing a multi-task chain model and the capacity prediction model based on support vector machine. In view of the flexible job-shop scheduling problem using segmented AGV, Liu et al. [17] presented a mathematical model of dual-resource scheduling optimization of machine tool and robot and established an AGV with the objective function of minimizing the maximum completion time. Zhang et al. [1] analyzed the workshop AGV scheduling task, modeled the workshop as a network of nodes, and applied an improved QMIX-Based AGV Scheduling Approach. As people's personalization needs increase, matrix manufacturing workshop is more and more favored by factories. The research on manufacturing systems has significant theoretical significance. Nevertheless, the existing theories cannot be readily applied to handle AGV scheduling problem in matrix manufacturing workshops.

Recently, researchers have initiated investigations into the AGV scheduling problem in matrix manufacturing workshops owing to the advantages associated with such workshops, including their large scale, variety and personalized customization. Zou et al. [8] first proposed discrete artificial bee colony algorithm to handle scheduling problem of multi-AGV in matrix manufacturing workshop. Subsequently, Zhang et al. [18] proposed an improved Iterative Greedy (IIG) algorithm to solve the multi-AGVs scheduling problem in the manufacturing workshop. In the algorithm, the AGV path merging strategy and the workshop division strategy were designed. Li et al. [19] proposed an efficient discrete invasive weed optimization algorithm to study automated guided vehicle scheduling problem with time and capacity constraints. Li et al. [20] proposed a new discrete invasive weed optimization algorithm to solve dynamic multiple automated guided vehicles scheduling problem. Zou et al. [21] proposed an iterative greedy algorithm to solve multi-compartment automated guided vehicle scheduling problem. Li et al. [22] proposed an enhanced genetic algorithm for the AGV scheduling problem with unloading preparation time. Subsequently, Zou et al. [23] proposed an iterative greedy algorithm to solve the multi-AGV scheduling problem with offloading safety detection. Aiming at the problem of multi-AGV charging and maintenance, Zou et al. [24] proposed an adaptive iterative greedy algorithm. Zou et al. [23] proposed a mixed integer linear programming model and a population-based iterative greedy (PIG) algorithm to solve the multi-AGV scheduling problem with offload safety detection in a matrix manufacturing shop. Wang et al. [25] used a population-based variable neighborhood search (PVNS) algorithm to solve the multi-AGV scheduling problem with sudden faults. The above studies are all about the scheduling problem of AGV in matrix production workshop, but they do not consider the actual problem of dynamic change of AGV unloading time. Therefore, it is crucial to propose an appropriate method to solve the $\text{MAGVS}_{\text{DUT}}$ problem.

Jaya algorithm is a swarm based intelligent optimization algorithm, which has shown excellent performance in solving combinatorial optimization problems. It was first proposed by Rao in 2016 for continuous optimization problems [26]. Mumtaz et al. [27] proposed a hybrid spider monkey optimization (HSMO) algorithm to solve the PCB assembly line problem with multi-level planning and scheduling. Rauf et al. [28] proposed Raccoon family optimization (RFO) algorithm to solve the problem of integrated planning and scheduling of multiple manufacturing projects under resource constraints. Khalid et al. [29] combined particle swarm optimization with NEH algorithm to solve the product scheduling problem in cell manufacturing system. Wang et al. [30] studied the balance problem of mixed-flow human-machine collaborative disassembly line, using an improved artificial fish swarming algorithm (IAFSA) to optimize the number of workstations, balance the idle time, and minimize the disassembly cost. However, compared with the existing algorithms, Jaya algorithm has the characteristics of few parameters, optimization and error avoidance, and has been used for various optimization issues in the past, such as: The urban traffic signal control problem [31] and the flexible flow shop scheduling problem [32]. Caldeira and Gnanavelbabu [33] presented an improved Jaya algorithm to handle flexible job shop scheduling problem, which overcame problem of adjusting a mass of parameters and improved the quality and diversity of the solution. Fan et al. [34] proposed a hybrid Jaya algorithm combined with tabu search to solve the FJSP. In the local search phase, three methods are proposed to deal with multiple critical paths. Gao et al. [35] proposed a discretized Jaya algorithm for solving the flexible job-shop scheduling problem with new job insertion, with the objective of minimizing the maximum machine effort. Thus, although the Jaya algorithm has many advantages and has been applied to various practical problems, its performance in terms of dependence on the initial solutions, lack of diversity maintenance, and limited adaptability to specific problems still requires further improvement.

In summary, AGV scheduling problem has become one of the popular topics studied by scholars. However, there is no research on $\text{MAGVS}_{\text{DUT}}$ in the current literatures, and even though the available optimization algorithms have presented solutions to the AGV scheduling problem, they cannot be straightly applied to $\text{MAGVS}_{\text{DUT}}$. Therefore, it is crucial to address the $\text{MAGVS}_{\text{DUT}}$ with discrete Jaya algorithm.

III. PROBLEM DESCRIPTION AND FORMULATION

A. PROBLEM DESCRIPTION

A typical matrix manufacturing workshop was mentioned by Zou et al. [8], as shown in Fig.1, where workstations and call-workstations are arranged neatly. When the workstations are short of production materials, their status change to call-workstations. For descriptive purposes, we call call-workstations tasks. AGVs are tasked with delivering production materials to these tasks. AGVs start from the depot, deliver materials to the designated tasks, and return to the depot after completing all tasks. Since the amount of material unloaded in designated tasks is uncertain, the time taken by AGVs to unload the material is also dynamic and uncertain. The unloading time is an important factor for the AGV scheduling system and cannot be ignored, otherwise it may directly affect the normal production of the workshop. Therefore, this study is that how to find an optimal method to dispatch AGVs to complete all delivery tasks while considering the dynamic unloading time. Without loss of generality, several assumptions are given as follows:

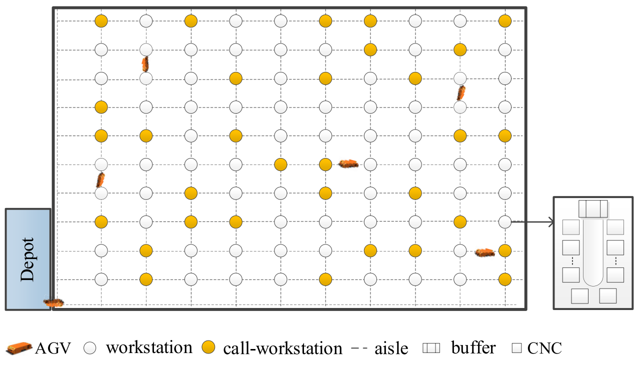

FIGURE 1. The layout diagram of the matrix manufacturing workshop.

1) All equipment in the depot is functioning normally and there will be no faults such as downtime or collisions.
2) Each task can only be executed by a single AGV, and each aisle allows only one AGV to pass at a time.
3) The speed of the AGVs stays unchanged during the entire transportation process.
4) Each task must meet time constraints and capacity constraints.
5) All AGVs leave the depot and eventually come back to the depot.
6) The cost of AGVs is very expensive, so reduce the use of AGVs as much as possible.

B. PROBLEM FORMULATION

In this section, a mathematical model is set up as follows.

Parameters and constants:

i, j	unique identifier for the task.
p_i	location of task i.
x_i	abscissa of task i.
y_i	ordinate of task i.
n	number of tasks.
n'	maximum number of tasks the AGV can perform.
k	present AGV (or AGV route).
k'	anticipated number of AGVs.
k''	number of AGVs allowed for scheduling.
v	speed of AGV.
Q	capacity of AGV.
Q_t	unloading volume of AGV per unit time.
q_i	materials needed for task i.
d_{ij}	the distance traveled between tasks i and j.
t_{ij}	the time taken to travel between tasks i and j.
T_i^c	call time (i.e., time when task i gives a signal).
T_i^l	delivery time (i.e., latest time for the AGV to reach task i).
T_0	time when the AGV departs from the depot.
t_u	unloading time of every task.
t_m	consumption time of each piece of production material.
S	overall stock of the material buffer.
S_i^c	inventory of material buffer at the time of request.
g	weight per piece of production material.
c_t	unit costs of travel distance on an AGV route.
c_a	cost of per AGV.
c_e	penalty cost for earliness.

Decision Variables:

x_{ijk}	if there is a feasible AGV route between task i and j, it is 1, otherwise 0.
T_i^r	task i actual arrival time.
m	the number of AGV utilized.

Objective:

$$\min F(i,j,k) = c_t \sum_{k=1}^{m}\sum_{j=0}^{n}\sum_{i=0}^{n} x_{ijk} d_{ij} + c_a \sum_{k=1}^{m}\sum_{j=1}^{n} x_{0jk}$$

$$+ c_e \sum_{k=1}^{m}\sum_{j=1}^{n}\sum_{i=0}^{n} x_{ijk}\left(T_j^l - T_j^r\right) \tag{1}$$

Subject to:

$$d_{ij} = |x_i - x_j| + |y_i - y_j| \tag{2}$$

$$t_{ij} = d_{ij}/v \tag{3}$$

$$T_j^r = T_i^r + t_u + t_{ij}, \quad \forall i \in V, j \in V \setminus \{0\} \tag{4}$$

$$D_{xij} = \sum_{k=1}^{m}\sum_{j=0}^{n}\sum_{i=0}^{n} x_{ijk} d_{ij} \tag{5}$$

$$q_j = \left[\left(S^t - S_j^c\right) + \left\lceil \left(T_j^r - T_j^c\right)/t_m \right\rceil\right] * g, \quad \forall j \in V \setminus \{0\} \tag{6}$$

$$t_u = q_j/Q_t \tag{7}$$

$$k' = \lceil n/n' \rceil, n' = 12(8) \tag{8}$$

$$\sum_{k=1}^{m}\sum_{i=0}^{n} x_{ijk} = 1, \quad \forall j \in V \setminus \{0\} \tag{9}$$

$$\sum_{k=1}^{m}\sum_{j=0}^{n} x_{ijk} = 1, \quad \forall i \in V \setminus \{0\} \tag{10}$$

$$\sum_{i=0}^{n} x_{ijk} - \sum_{i=0}^{n} x_{jik} = 0, \quad \forall k \in K, j \in V \setminus \{0\} \tag{11}$$

$$\sum_{i=1}^{n} x_{i0k} = \sum_{j=1}^{n} x_{0jk} = 1, \quad \forall k \in K \tag{12}$$

$$x_{ijk}\left(T_i^r + t_u + t_{ij} - T_j^r\right) = 0, \quad \forall k \in K, j \in V \setminus \{0\}, i \in V \tag{13}$$

$$\sum_{j=1}^{n}\sum_{i=0}^{n} x_{ijk} \cdot q_j \le Q, \quad \forall k \in K \tag{14}$$

$$T_i^c \sum_{k=1}^{m}\sum_{j=0}^{n} x_{ijk} \le T_i^r \le T_i^l \sum_{k=1}^{m}\sum_{j=0}^{n} x_{ijk}, \quad \forall i \in V \setminus \{0\} \tag{15}$$

$$k' \le m \le k'', k'' = 6 \tag{16}$$

$$x_{ijk} \in \{0, 1\}, \quad \forall i, j \in V, \quad \forall k \in K \tag{17}$$

$$x_{ijk} = 0, i, j \in V \text{ and } i = j \tag{18}$$

$$T_i^r = T_0, i = 0 \tag{19}$$

In this model, the objective (1) is to minimize transportation cost which is composed of travel costs, vehicle costs, and penalty costs for earliness. Equation (2) and (3) represent the transportation distance and transportation time of each AGV on the route, and equation (4) indicates that it takes three parts of time for AGV to finish task i and task j, namely, the time for the AGV to reach task i, T_i^r, dynamic unloading time, t_u, and transport time, t_{ij}.Equation (5) represents the total distance traveled by all AGVs. Equation (6) represents the weight of material unloaded by the AGV for the task. Equation (7) represents the unloading time of each task. Equation (8) represents the minimum number of AGVs expected to be required. Constraints (9)-(11) ensure that every task must be serviced by an AGV once and that AGVs enter at most one task after leaving the depot. Constraint (12) denotes that every AGV route begins and finishes at the depot. Constraint (13) establishes the connection between arrival times of two adjacent tasks. Constraint (14) ensures that the overall demand of all tasks on a route cannot beyond the AGV capacity. Constraint (15) ensures that every task is completed within required time. Constraint (16) ensures that the overall number of AGVs can be within the range, and constraints (17)-(19) place limitations on the decision variables. The Q_t in the model represents the amount of AGV unloaded per unit time, which can be calculated to determine how long the AGV should unload the missing material for each task.

Take $n = 10$ as an example, the specific instances are shown in the Table 1 below, a sequence represents the task number, X-axis coordinates, Y-axis coordinates, the shortest distance to the depot, the call time, the inventory in the buffer at the call time, the latest delivery time, the type of demand, and the number of tools. Assuming Q =250, Q_t =3, c_t =1, c_a =200, c_e =0.1, AGV needs to distribute materials to 10 tasks in a certain order, and AGV distributes the tasks according to the proposed heuristic algorithm. The distribution rule is to select the task closes to the previous task and the task with the smallest call time among the remaining tasks. The original task sequence is {1,2,3,4,5,6,7,8,9,10}, and the later sequence is {1,6,4,9,2,7,3,8,10,5}, through calculation can get the total distance obtained is 367.4, the early arrival time is 2515.35, and only one AGV is used in this mission, so the final F=818.935.

TABLE 1. The specific instances.

(1)	{24,3,4,51.7,14,28,614,1,0}
(2)	{96,10,6,107.8,70,29,700,1,0}
(3)	{69,7,9,117.7,97,28,697,1,0}
(4)	{62,7,2,56.1,115,28,715,1,0}
(5)	{30,3,10,104.5,136,30,796,1,0}
(6)	{54,6,4,68.2,190,28,790,1,0}
(7)	{78,8,8,114.4,230,30,890,1,0}
(8)	{38,4,8,92.4,263,30,923,1,0}
(9)	{83,9,3,75.9,310,28,910,1,0}
(10)	{47,5,7,89.1,348,30,1008,1,0}

IV. BASIC JAYA ALGORITHM

Jaya algorithm is a meta-heuristic algorithm recently raised by Rao [26], which was initially used to solve continuous real-parameter optimization problems. Jaya means success in Sanskrit. Its key concept is to approach victory and avoid failure, always following the rule of constant improvement. The algorithm continuously improves the solution by moving the current solution towards optimal solution and away from worst solution until termination condition is met. The first step of algorithm is to construct a population of size (PSize). Next, the optimal and worst solutions are extracted from the initial population and used to update the other solutions for the next iteration. In each iteration, the optimal and worst solutions are updated. The Jaya algorithm is a single-stage algorithm that necessitates the evaluation of only one equation in each iteration to determine the value of the new solution. The Jaya algorithm also has the advantage that there are no particular algorithm parameters that will affect the outcome of the solution. Therefore, no external computational work is required to adjust the parameters. This enables the Jaya algorithm easier to interpret and execute than other metaheuristic algorithms. In Jaya algorithm, the optimal and worst solutions for each iteration are determined by the objective function values. The rest of solutions are varied in the population as follows in equation (20), where $r1$ and $r2$ are random numbers between 0 and 1. The term $r1 \times (S_{best} - |S_i|)$ brings the current solution closer to optimal solution and term $r2 \times (S_{worst} - |S_i|)$ brings the current solution away from worst solution. S_{best} is optimal solution for the current population and S_{worst} is worst solution for the current population. If the new solution obtained has a superior objective value, it is chosen, otherwise the former solution is retained. The restriction of Jaya's algorithm is that it does not efficiently explore regions in the solution space and tends to fall into local optima. To overcome this, the sequence insertion operator is used to generate a diversity of solutions.

$$S_{i+1} = S_i + r1 \times (S_{best} - |S_i|) - r2 \times (S_{worst} - |S_i|) \quad (20)$$

V. THE PROPOSED DJaya ALGORITHM

Since basic Jaya algorithm was initially introduced for successive optimization problems, it cannot be straightly used to the discrete case. For the purpose of solving the MAGVS$_{\mathrm{DUT}}$ with minimized transportation cost, a variant of Jaya algorithm called DJaya algorithm is proposed and will be presented in this section. Among the DJaya algorithm, the solution representation of the DJaya algorithm is first introduced in detail, a heuristic based on ant colony algorithm is given, followed by the initialization of the population, the update mechanism of the DJaya algorithm and finally the flowchart of the DJaya algorithm is given.

A. SOLUTION REPRESENTATION

In order to make the representation of the MAGVS$_{\mathrm{DUT}}$ solution more simplified, a simple one-dimensional description approach has been adopted, and the solution can be expressed as a one-dimensional vector with a length of $n + m$-1, where n and m stand for the number of tasks and the number of AGVs in matrix manufacturing workshop, respectively, and assuming that n_k is a sub-vector consisting of the tasks serviced by AGV k, and separating two neighboring sub-vectors by the number 0, which indicates the beginning of each AGV's route, then $(n_1, 0, n_2, 0, n_3, \ldots, 0, n_m)$ represents the solution. Here is a simple example, suppose that there are 8 tasks and 3 AGVs, the solution has a length of 10, AGV 1 delivers materials to tasks in order of 1 $\rightarrow$ 3 $\rightarrow$ 7, AGV 2 delivers materials to tasks in order of 2 $\rightarrow$ 4, and AGV 3 delivers materials to tasks in order of 5 $\rightarrow$ 6 $\rightarrow$ 8,so the solution expression is (1,3,7,0,2,4,0,5,6,8). The service order of the AGVs is shown as Fig.2.

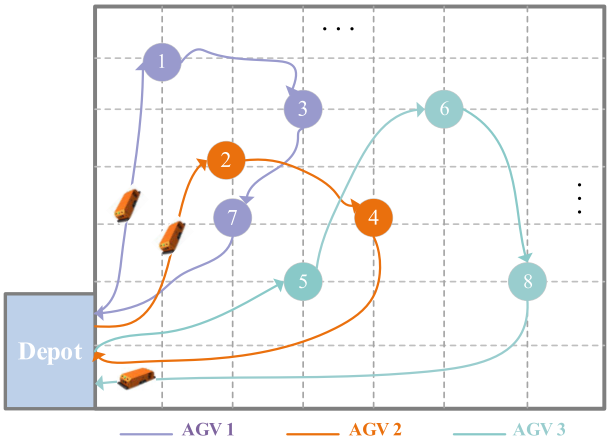

FIGURE 2. The service order of the AGVs.

B. A HEURISTIC BASED ON ANT COLONY ALGORITHM

Zou et al. [8] raised a Nearest Neighbor Heuristic (NNH) task search algorithm. The primary thought is to discover next task that is closest to the current task according to the Manhattan Distance. In this section, a heuristic based on ant colony algorithm is introduced. In heuristic, the distance to the remaining tasks and the call time of the remaining tasks are considered as two important factors to determine the next task, which is given in the following formula.

$$P_{i,j} = \frac{(\tau_{i,j})^{\alpha} \cdot (\eta_{i,j})^{\beta}}{\sum_{s\in A} (\tau_{i,s})^{\alpha} \cdot (\eta_{i,s})^{\beta}}, ifj \in A \tag{21}$$

where α stands for the heuristic factor of the pheromone, β represents the expected heuristic factor, $\tau_{i,j}$ stands for the pheromone strength from position i to position j, $\eta_{i,j}$ stands for the heuristic information value from position i to position j, A is the set of next points that can be selected and $P_{i,j}$ is the probability value of choosing the next point. Respectively, analogous to the problem we study, $\tau_{i,j}$

is equivalent to the call time to the remaining tasks, and $\eta_{i,j}$ represents the distance to the remaining tasks.

In the algorithm, the following notation is used: $U = 1, 2 \ldots n$ represents the collection of undistributed tasks, R denotes one of the present AGV route, X denotes the generated solution, j denotes the current task, and p is the probability value. Since AGV begins from the depot, the depot is taken as the current task, then the probability value from the current task to the remaining tasks is calculated by using the formula, the task with the minimum probability value is taken as the first task in the route, then the second task is sought from the remaining tasks and attempts to be inserted into the second location of the route, one point needs to be noted: the time window constraint and the loading constraint are satisfied each time, if the constraints are satisfied, then continue to search for the next task, otherwise, a new route is opened, this process will be repeated until all the tasks are assigned, the process of heuristic algorithm is given in Algorithm 1.

Algorithm 1 Heuristic Algorithm
Output: solution X

```
1: $U$: set of unassigned tasks; $R$: a route; $p$: value of probability; $j$: the current task
2:   Let R = 1 and $j = 0$
3:   while $U$ is not empty do
4:     Calculate $p$ i between task $j$ and each task $i$ in $U$
5:     $p_{\min} = \min_{i=1,2,...,N}(p_i)$, N = sizeof($U$)
6:     Test to append task $i$ with $p_{\min}$ to R
7:     if R satisfies capacity constraint and time constraint then
8:       Make task $j = i$ and remove $i$ from $U$
9:     else
10:      Close R, construct a new R=R+1
11:      Add 0 at the last of $X$, and make $j = 0$
12:  endif
13:  endwhile
14:  if R is unempty then
15:  Add R to solution $X$
16:  endif
17: return solution $X$
```

C. INITIAL POPULATION PHASE

An initial population that balances quality and diversity will always allow the algorithm to converge quickly to get a good result [36]. To raise the quality and variety of the initial population, the heuristic algorithm is used in this study for generating initial solution, and the remaining solutions are randomly created. The random approach is to randomly create a sequence that includes all tasks and then adds tasks to AGV routes from the front to the end of the permutation depending on the time and capacity constraints. After the heuristic method and random generation method to get PSize initial solutions. Due to the high cost of AGV, in order to reduce the use of AGV, Zou et al. [8] proposed a merge operator, which can make the quality of the solutions improve again, then we find the optimal and worst solutions in the current population for later evolution, P indicates initial population, PSize indicates population size, σ indicates generations of solutions, X_{best} indicates the optimal solution of the current population, X_{worst} indicates the worst solution of the current population, X represents the solution generated by the heuristic algorithm, X_{σ} represents a AGV route, and the process of initial population generation is given in Algorithm 2.

D. DJaya UPDATING MECHANISM

The prime thought of the Jaya algorithm is to move current solution towards optimal solution and away from worse ones. Based on the basic thought of Jaya algorithm, a discrete DJaya strategy is proposed. The related formula is as follows.

$$X_{tN} = X_t * (r1 \cdot X_{tB} + r2 \cdot X_{tW}) \tag{22}$$

Algorithm 2 Initial Population
Output: solution X_{best} and X_{worst}

```
1: Generate an initial solution $X$ by heuristic algorithm
2:      for $\sigma$ =2 to $PSize$
3:       Randomly generate sequences for all the tasks
4:      Make AGV route R= $\emptyset$
5:       for $i$=1 to $n$
6:          Test to append task $i$ to R
7:          if R satisfies capacity constraint and time constraints then
8:              Append task $i$ to R
9:          else
10:             Append R to solution $X_\sigma$ and empty R
11:             Add 0 at the last of $X_\sigma$
12:    endif
13:    endfor
14:    if R is unempty then
15:        Append R to solution$X_\sigma$ and empty R
16:    endif
17:        Add solution$X_\sigma$ to the initial population P
18:    endfor
19:    for $\sigma$ =1 to $PSize$
20:        merge operator
21:    endfor
22:    for $\sigma$ =1 to $PSize$
23:        find $X_{best}$and$X_{worst}$
24:     endfor
25: return solution $X_{best}$and$X_{worst}$
```

1) random coefficient generation: binary numbers $r1$ and $r2$ are generated, namely $r1$, $r2 \in \{0, 1\}$, and $r1+r2=1$;
2) generate new solution: the way of generating new solutions based on the DJaya algorithm is shown in the formula, X_{tB} stands for the optimal solution in the t iteration, X_{tW} represents the worst solution in the t iteration, X_{tN} is new solution, X_t is current solution, $*$ represents whether current solution is operated with optimal solution or worst solution, if $r1$ is 1, the current solution is operated with the optimal solution. Conversely, when $r2$ is 1, the current solution is operated with the worst solution. The DJaya updating mechanism steps involving a near optimal operator and an away worst operator are shown as follow.

1) A NEAR OPTIMAL OPERATOR

Since the objective of the problem is composed of three parts, namely, travel cost, penalty cost, and vehicle cost, the total of three costs of the optimal solution in each iteration is the minimum. Since the Jaya algorithm has the "optimal-oriented" property [26], we fully borrow the properties of the optimal solution and propose a near optimal operator. First of all, the optimal solution in population as a reference sequence, which is assumed to be X_{best}, and the current solution is X_{current}, then start from the first position of X_{best}, find the element in X_{current} that is the same as it, and try to re-insert this same element into all feasible positions of X_{current}, "feasible" means that both time constraints and capacity constraints are satisfied, and finally insert it into the position that obtains the minimum total cost. Next consider the element in the second position of X_{best}, repeat the above process until all elements in the X_{best} are considered, and finally obtain a new solution, suppose it is X. The following Fig.3 shows an example of a near optimal strategy, and the process is given in Algorithm 3.

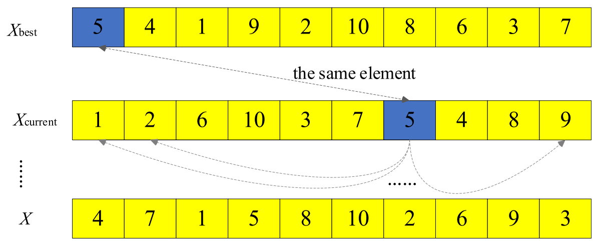

FIGURE 3. A near optimal operator.

Algorithm 3 A Near Optimal Operator
Output: solution X

```
1: Find the $X_{\text{best}}$, $X_{\text{current}}$
2:   for $i = 1$ to lengthOfArray($X_{\text{best}}$)
3:     for $j = 1$ to lengthOfArray($X_{\text{current}}$)
4:       if $X_{\text{current}}[i]$ == element then
5:         position = $j$
6:         break
7:         endif
8:       endfor
9:     for $k = 1$ to lengthOfArray($X_{\text{current}}$)
10:      test to insert element to position $k$
11:      if meet the time and capacity constraints then
12:        calculate the cost
13:      endif
14:   endfor
15:   for $k = 1$ to lengthOfArray(TotalCost)
16:      find the best position
17:    endfor
18:   endfor
19: return solution $X$
```

2) AN AWAY WORST OPERATOR

As mentioned above, Jaya algorithm not only has the characteristics of "optimal-oriented", but also has the characteristics of "error avoidance" [26], how to make current solution far away from worst solution is a problem that we should think about. An away worst operator is designed that moves current solution away from worst solution. First of all, worst solution in population is selected, which is assumed to be X_{worst}, the current solution is X_{current}, then start from the first position of X_{worst}, compare it with the element in the same position in X_{current}, if the element is same, put it in the X_{same}, otherwise put it in the $X_{\text{different}}$. Until elements in the X_{worst} are taken into account, finally elements in the X_{same} are inserted into $X_{\text{different}}$ in turn, so that the total cost of the inserted sequence is minimized, and the new solution X is obtained. BackCost and FrontCost represent the cost of new solution and the cost of old solution respectively. Comparing BackCost with FrontCost, the solution with the lower cost is kept. The following Fig.4 shows an example of an away worst strategy, the process is given in Algorithm 4, and the insertion-based local search operator is given in Algorithm 5.

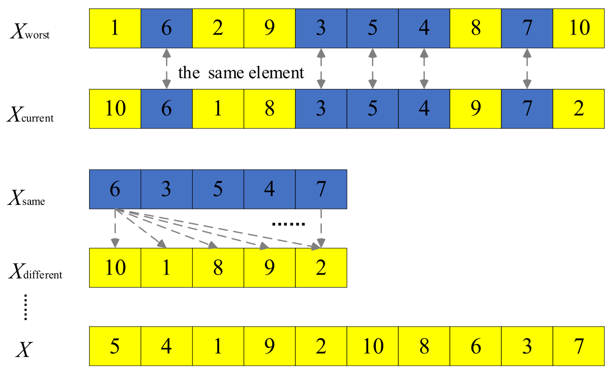

FIGURE 4. An away worst operator.

Algorithm 4 An away worst operator
Output: solution X

```
1: Find the $X_{worst}$, $X_{current}$
2:   **for**$i=$ 1tolengthOfArray($X$current)
3:      **if** $X_{current}[i]==X_{worst}[i]$ **then**
4:         $X_{same}[i]=X_{current}[i]$
5:      **else**
6:         $X_{different}[i]=X_{worst}[i]$
7:      **endif**
8:      **INSERT**
9:      **if** BackCost <FrontCost **then**
10:         chflag=true
11:     **else**
12:         chflag=false
13:     **endif**
14:  **endfor**
15:**return** solution $X$
```

Algorithm 5 INSERT
Output: solution X

```
1: Find the $X_{same}$, $X_{different}$
2:   **for** $i=1$ to lengthOfArray($X_{same}$)
3:      **for** $j=1$ to lengthOfArray($X_{different}$)
4:         insert $X_{same}[i]$ to position $j$
5:         **if** meet the time and capacity constraints **then**
6:            calculate the cost
7:         **endif**
8:      **endfor**
9:      **for** $k=1$ to lengthOfArray(TotalCost)
10:        find the best position
11:     **endfor**
12:  **endfor**
13:**return** solution $X$
```

3) SEQUENCE INSERTION OPERATOR

To prevent the solution from trapping in a local optimum, this paper proposes a neighborhood operator for sequence insertion. The process is divided into three main steps. Firstly, we randomly select three positions from the X, denoted as pos_1, pos_2 and pos_3, it should be noted that the three positions are sorted in ascending order, that is $pos_1 < pos_2 < pos_3$. Second, we select the elements between pos_1 and pos_2. Finally, we reinsert the selected elements after pos_3. In this way, a new solution X' can be generated, which greatly improves the diversity of the solution. The process is given in Fig.5, and the Sequence insertion operator flow is illustrated in Algorithm 6.

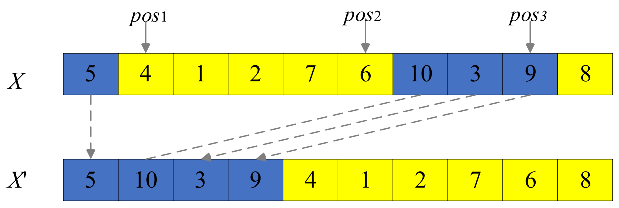

FIGURE 5. Sequence insertion operator.

Algorithm 6 Sequence Insertion Operator
Output: solutionX'

```
1: **Sort**$pos_1$, $pos_2$, $pos_3$ in ascending order so that $pos_1 < pos_2 < pos_3$
2: **Make**$seq$ for the elements between $pos_1$ and $pos_2$
3: **Insert** $seq$ after $pos_3$
4: return solution$X'$
```

E. SUMMARY OF THE PROPOSED DJaya ALGORITHM

As mentioned before, DJaya algorithm introduced in the paper, DJaya algorithm has several main components: Firstly, the population P is initialized to find optimal solution and worst solution. Then, new solutions are generated by DJaya updating mechanism. Finally, solutions are updated. The whole flowchart of DJaya algorithm is shown in Fig.6.

VI. COMPUTATIONAL AND STATISTICAL EXPERIMENTATION

In this section, the validity of the presented strategies and algorithms will be verified. All algorithms are coded using C++ programming language in Visual Studio 2019, and experiments are conducted on Windows 11 operating system using Intel(R) Core (TM) i7-12700 CPU @ 2.10 GHz processor and 16.0GB of RAM in the hardware environment. All algorithms stopped running when the predefined CPU runtime of 5 seconds was reached. We collected 110 instances form Foxconn Technology Group, one of China's foremost sophisticated electronic manufacturing companies, as testing benchmark, which are divided into two sets: a testing set

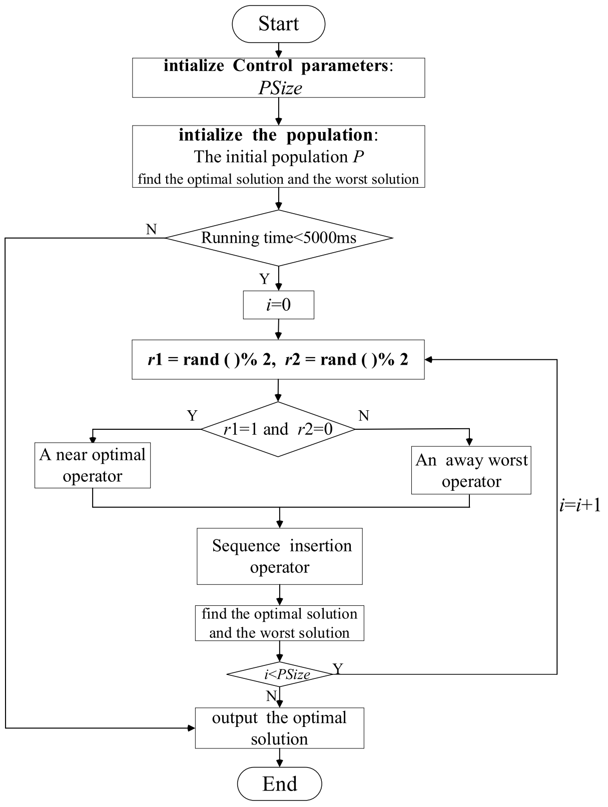
Start
intialize Control parameters: PSize
intialize the population: The initial population P
find the optimal solution and the worst solution
Running time<5000ms
N
Y
i=0
r1 = rand ( )% 2, r2 = rand ( )% 2
r1=1 and r2=0
Y
N
A near optimal operator
An away worst operator
i=i+1
Sequence insertion operator
find the optimal solution and the worst solution
i<PSize
Y
N
output the optimal solution
End

FIGURE 6. Whole flowchart of the DJaya.

consisting of 100 instances and a calibration set consisting of 10 instances. In the experiment, we consider tasks containing 10, 20, 30, 40, 50, each containing 22 instances, for a total of 110 instances. To ensure the reliability of parameter calibration, we assign 20 of the 22 instances in the same task to the testing set, while the remaining 2 are assigned to the calibration set. This method ensures that the testing set consists of a total of 100 instances (20∗5) and the calibration set consists of a total of 10 instances (2∗5). The instances in testing set and calibration set are independently repeated 30 times and 10 times, respectively. RPI (Relative Percentage Increase) is given to estimate the efficiency of all algorithms, and its expression is as follows:

$$\mathrm{RPI} = \frac{(C_i - C_{best})}{C_{best}} \times 100\% \tag{23}$$

where: C_i is the fitness obtained by a certain algorithm in the given instance. C_{best}: The minimum fitness obtained by all algorithms in the same instance. It is obvious that for a given instance, the lower the RPI value, the greater the result. Furthermore, the analysis of variance (ANOVA) technique is used to examine the statistical significance of discrepancies observed in the experimental results.

A. EXPERIMENTAL SETTINGS

In order to better distinguish test set and calibration set, the method of "T + number of task tasks + instance index" is used to represent the test set (e.g., T20I5), and the method of "C + number of task tasks + instance index" is used to represent the calibration set (e.g., C20I7). Detailed information is included in each instance, including task index, location, call time, buffer storage at the time of call, and delivery time. Assuming that the information for a task is {24, 3, 4, 51.7, 14, 28, 614}, it means that material delivery is requested from the control system by the task with index 24 at 14 seconds and reports that 28 pieces of material left in the buffer at that time. The AGV is required to arrive at the position (3, 4) at 614 seconds, and the shortest distance from this task to the warehouse is 51.7 units. Owing to space constraints, the instances utilized in the paper are not listed, and interested readers can request them from author. Table 2 lists the relevant parameters of the model.

TABLE 2. Parameters settings.

Items	Values	Items	Values
k''	5	n'	12
ΔT	5s	t_m	30s/slice
C	360s	g	0.75kg/slice
T_0	365s	S	48slice
Q	250kg	c_t	1
Q_t	3kg	c_e	0.1
v	1m/s	c_a	200
n	100		

TABLE 3. Parameters for the competitive algorithms.

Algorithms	Parameters
DABC	$PSize = 150, l = 800, r = 80, \tau = 20$
DIWO	$PS_0 = 50, PS_{\max} = 70, S_{\max} = 15, Plen = 2$
IG	$InitType = 0.8, T = 0.5, d = 5$
IIG	$d = 5, OpertIter = 60$
DJaya	$PSize = 150$

B. CALIBRATION OF THE PROPOSED AND COMPETING METHODS

This section calibrates parameters in DJaya algorithm and the comparison algorithms. In the proposed DJaya algorithm, we calibrated the population size (*PSize*) to achieve optimal performance. *PSize* has 5 levels: 30, 130, 150, 200, and 300. In other words, there are 5 options for the *PSize* parameter. When running the DJaya algorithm with different parameters, each calibration instance is performed 10 times independently, as a result 10 calibration instances produce a whole of 5∗10∗10=500 results. To ascertain the best level for the factor, *PSize* is treated as a factor and the RPI as the dependent variable to obtain the best configuration of algorithm. A multifactor Analysis of Variance (ANOVA)is performed on RPI. It should be noted that if the means of two

TABLE 4. Experimental outcomes for the instances comprising of 10 tasks.

Instance	DABC			DIWO			IG			IIG			DJaya		
	Min	Max	Ave	Min	Max	Ave	Min	Max	Ave	Min	Max	Ave	Min	Max	Ave
T10I1	29.09	29.09	29.09	29.05	30.89	30.22	29.09	29.09	29.09	29.09	29.09	29.09	**0.00**	**9.30**	**6.41**
T10I2	35.90	35.90	35.90	35.88	39.78	36.40	35.90	35.90	35.90	35.90	35.90	35.90	**0.00**	**3.18**	**2.15**
T10I3	33.32	39.54	35.59	33.18	39.43	33.39	33.32	33.32	33.32	33.32	33.32	33.32	**0.00**	**6.10**	**3.15**
T10I4	25.37	27.01	25.60	25.20	25.20	25.20	25.37	25.37	25.37	25.37	25.37	25.37	**0.00**	**4.32**	**2.00**
T10I5	27.39	27.39	27.39	27.39	27.39	27.39	27.39	27.39	27.39	27.39	27.39	27.39	**0.00**	**8.44**	**4.97**
T10I6	22.22	22.22	22.22	22.15	22.15	22.15	22.22	22.22	22.22	22.22	22.22	22.22	**0.00**	**5.73**	**2.18**
T10I7	28.24	28.24	28.24	28.22	28.22	28.22	28.24	28.24	28.24	28.24	28.24	28.24	**0.00**	**8.57**	**3.85**
T10I8	33.93	33.93	33.93	33.83	33.83	33.83	33.93	33.93	33.93	33.93	33.93	33.93	**0.00**	**8.55**	**5.14**
T10I9	22.94	22.94	22.94	22.90	22.90	22.90	22.94	22.94	22.94	22.94	22.94	22.94	**0.00**	**12.62**	**7.25**
T10I10	21.64	22.70	21.89	21.57	21.57	21.57	21.64	21.64	21.64	21.64	21.64	21.64	**0.00**	**6.31**	**2.74**
T10I11	22.90	22.90	22.90	22.86	22.86	22.86	22.90	22.90	22.90	22.90	22.90	22.90	**0.00**	**5.96**	**4.19**
T10I12	21.64	21.64	21.64	21.58	21.58	21.58	21.64	21.64	21.64	21.64	21.64	21.64	**0.00**	**7.80**	**5.17**
T10I13	33.12	33.12	33.12	33.08	33.08	33.08	33.12	33.12	33.12	33.12	33.12	33.12	**0.00**	**13.99**	**8.12**
T10I14	32.12	32.12	32.12	32.06	32.06	32.06	32.12	32.12	32.12	32.12	32.12	32.12	**0.00**	**9.61**	**6.50**
T10I15	21.16	23.57	21.80	21.12	21.12	21.12	21.16	21.16	21.16	21.16	21.16	21.16	**0.00**	**7.11**	**3.36**
T10I16	21.89	21.89	21.89	21.73	21.73	21.73	21.89	21.89	21.89	21.89	21.89	21.89	**0.00**	**2.08**	**1.25**
T10I17	25.55	25.55	25.55	25.54	25.54	25.54	25.55	25.55	25.55	25.55	25.55	25.55	**0.00**	**9.83**	**5.20**
T10I18	35.57	35.57	35.57	35.47	39.53	38.72	35.57	35.57	35.57	35.57	35.57	35.57	**0.00**	**7.21**	**3.51**
T10I19	19.05	19.05	19.05	18.99	18.99	18.99	19.05	19.05	19.05	19.05	19.05	19.05	**0.00**	**5.73**	**1.96**
T10I20	23.79	23.79	23.79	23.73	23.73	23.73	23.79	23.79	23.79	23.79	23.79	23.79	**0.00**	**6.26**	**3.17**
Average	26.84	27.41	27.01	26.78	27.58	27.03	26.84	26.84	26.84	26.84	26.84	26.84	**0.00**	**7.44**	**4.11**

TABLE 5. Experimental outcomes for the instances comprising of 20 tasks.

Instance	DABC			DIWO			IG			IIG			DJaya		
	Min	Max	Ave	Min	Max	Ave	Min	Max	Ave	Min	Max	Ave	Min	Max	Ave
T20I1	23.73	25.35	24.50	23.46	26.17	24.28	23.51	23.51	23.51	23.51	24.25	23.74	**0.00**	**13.44**	**7.22**
T20I2	37.61	40.15	38.69	37.15	40.01	37.62	37.61	37.61	37.61	37.61	37.90	37.67	**0.00**	**21.03**	**16.17**
T20I3	22.46	24.03	23.15	22.41	23.02	22.70	22.46	22.46	22.46	22.46	23.53	22.99	**0.00**	**13.69**	**5.57**
T20I4	31.76	34.09	32.56	31.73	31.73	31.73	31.75	32.30	31.76	31.77	33.63	32.32	**0.00**	**19.34**	**12.99**
T20I5	16.21	17.39	16.67	16.11	19.41	16.31	16.18	16.18	16.18	16.45	17.35	16.85	**0.00**	**9.01**	**5.24**
T20I6	23.12	24.23	23.47	22.78	23.12	22.99	23.02	23.08	23.02	23.02	23.02	23.02	**0.00**	**13.02**	**8.02**
T20I7	21.29	22.47	21.91	21.22	22.39	22.09	21.12	21.13	21.12	21.13	21.87	21.59	**0.00**	**9.71**	**3.83**
T20I8	23.78	26.00	24.64	24.19	24.62	24.45	23.78	23.99	23.78	23.78	24.70	24.08	**0.00**	**12.40**	**8.13**
T20I9	19.25	21.61	19.96	19.17	19.85	19.47	19.25	19.25	19.25	19.25	19.25	19.25	**0.00**	**9.17**	**6.36**
T20I10	12.69	13.80	12.99	12.62	14.16	13.19	12.69	12.69	12.69	12.92	14.18	13.69	**0.00**	**6.03**	**3.11**
T20I11	24.63	25.96	25.42	24.06	25.36	24.85	24.08	24.11	24.08	24.08	24.26	24.12	**0.00**	**11.89**	**7.68**
T20I12	21.02	22.21	21.51	20.96	21.64	21.18	21.02	21.27	21.03	21.02	21.68	21.37	**0.00**	**12.19**	**6.63**
T20I13	24.93	26.21	25.55	24.84	25.91	25.71	24.92	24.93	24.92	24.93	25.84	25.50	**0.00**	**8.85**	**4.71**
T20I14	23.92	28.31	26.41	24.88	27.71	26.75	23.92	23.92	23.92	23.92	25.90	24.47	**0.00**	**6.93**	**2.89**
T20I15	22.02	24.13	23.10	21.95	25.08	22.79	22.02	22.02	22.02	22.05	23.31	22.80	**0.00**	**12.40**	**6.06**
T20I16	22.50	23.61	23.04	22.46	23.55	22.77	22.50	22.50	22.50	22.60	23.15	22.78	**0.00**	**11.50**	**6.84**
T20I17	27.42	28.93	28.01	26.50	28.28	27.28	26.57	27.16	26.62	26.72	28.54	28.03	**0.00**	**12.05**	**5.17**
T20I18	18.01	20.46	19.02	17.95	19.04	18.19	18.01	18.01	18.01	18.01	18.01	18.01	**0.00**	**5.80**	**1.64**
T20I19	23.74	25.73	24.58	23.66	24.87	24.04	23.74	23.74	23.74	24.07	25.08	24.42	**0.00**	**16.48**	**11.83**
T20I20	21.61	22.95	22.02	21.61	22.65	21.97	21.61	21.66	21.61	21.66	22.05	21.93	**0.00**	**10.92**	**6.42**
Average	23.08	24.88	23.86	23.00	24.43	23.52	22.99	23.08	22.99	23.05	23.87	23.43	**0.00**	**11.79**	**6.83**

groups in the plot overlap, there is no statistically meaningful distinction between them. According to mean plot of the *PSize* parameter, we can learn that the levels 30, 130, 150, and 200 do not overlap with the level 300. This indicates that *PSize* has a statistically significant impact on RPI within the 95% confidence interval. Among these levels, *PSize*=150 has the lowest RPI value, making it the optimal level, as shown in the Fig.7, its RPI value is the lowest. Therefore, we finally set the value of the *PSize* parameter to 150. The similar calibration procedure is carried out for remaining comparison algorithms, and the calibrated parameters are shown in the Table 3. Fig.7 represents the pertinent means plots with 95% Tukey's Honest Significant Difference (HSD) confidence intervals for the five factors of DJaya.

PSize : population size; l: predetermined number of trials; r : predetermined number; τ: replications; PS_0: the number of initial population; $PS_{\max}$: the maximum number of population; $S_{\max}$: the maximum number of seeds. *Plen*: predefined number of times; *InitType*: the type selected by the heuristic; T: Temperature; d: Number of tasks removed; *OperIter* :The number of iterations in the local search stage.

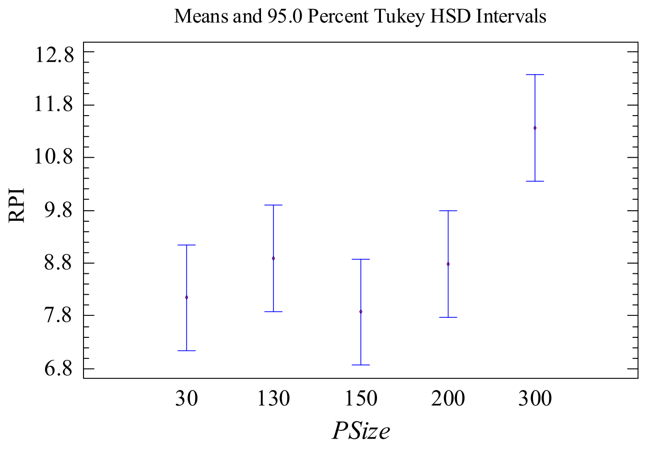

FIGURE 7. Means plots of parameter of the DJaya.

C. COMPARISON OF METHODS

To validate the efficiency of the new strategies presented in this algorithm, two comparative experiments are carried out. Next, effectiveness of the proposed heuristic algorithm and DJaya operators will be demonstrated through practical experiments.

1) THE EFFECTIVENESS OF THE PROPOSED HEURISTIC ALGORITHM

To ensure the validity of the presented heuristic algorithm, DJaya algorithm and DJaya_$_{no}$ algorithm are compared. The difference between the two algorithms is that the DJaya_$_{no}$ algorithm only uses a random method to initialize the population, while the other aspects of the algorithm are the same as the DJaya algorithm. Both algorithms are executed in the same experimental environment for testing same test instances. After running independently for 30 times, the RPI values obtained are compared and analyzed using ANOVA analysis tool. The experimental results are given in the Fig.8. As you can see, the DJaya algorithm yielded smaller RPI values, indicating that the heuristic algorithm is effective in improving the performance of DJaya algorithm and helping it find better solutions.

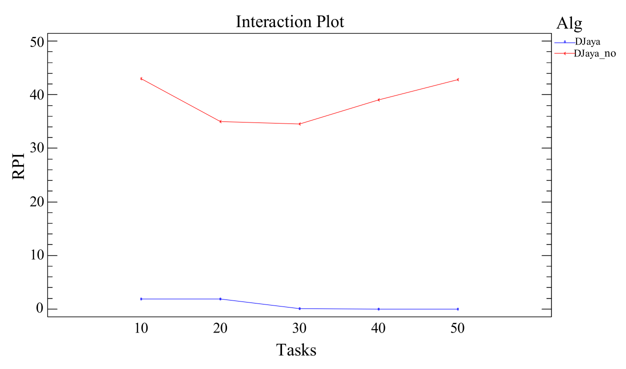

FIGURE 8. The effectiveness of the heuristic algorithm.

2) THE EFFECTIVENESS OF THE DJaya OPERATORS

To improve the search capability of the algorithm, academics typically incorporate local search operators into the algorithm [37]. In the proposed DJaya algorithm, a local search that references the optimal solution is introduced, during operations on the worst and current solutions, the parts that are identical to the current solution and the worst solution are removed, followed by a local search algorithm based on insertion. By comparing DJaya algorithm with the local search strategies and DJaya_no algorithm without the local search strategies, the effectiveness of the presented DJaya algorithm is confirmed. The experimental results is given in the Fig.9. It is evident that DJaya algorithm with local search operators achieved smaller RPI values, indicating that algorithm with local search operators has a noticeable effect on improving the behavior of DJaya algorithm.

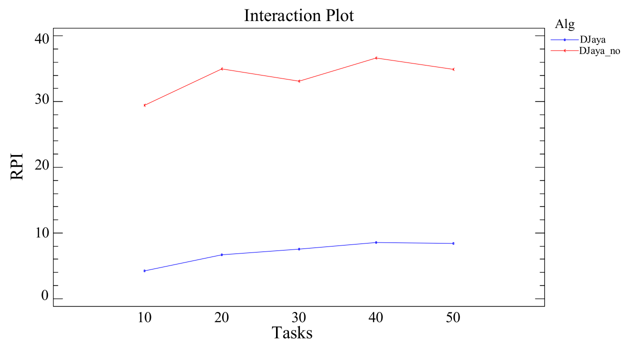

FIGURE 9. The effectiveness of DJaya operators.

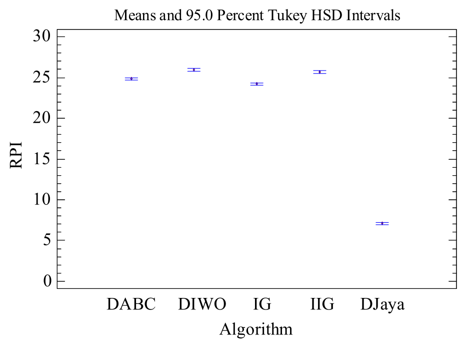

FIGURE 10. Means plots with 95% Tukey's HSD confidence intervals for all the comparison algorithms.

D. COMPARISON WITH OTHER ALGORITHMS

To ensure the validity of the presented algorithm, 100 test instances of different sizes are used for verification, and then the scalability of the algorithm was verified by analyzing the instances of different sizes. The experimental setup in this section is the same as the one mentioned above, and the evaluation index utilized is RPI. In order to obtain

TABLE 6. Experimental outcomes for the instances comprising of 30 tasks.

Instance	DABC			DIWO			IG			IIG			DJaya		
	Min	Max	Ave	Min	Max	Ave	Min	Max	Ave	Min	Max	Ave	Min	Max	Ave
T30I1	22.93	25.20	24.19	22.53	24.34	23.77	22.04	22.96	22.42	23.78	25.65	24.90	**0.00**	**11.81**	**6.63**
T30I2	20.87	23.49	22.27	22.29	25.92	24.16	20.45	21.76	20.98	21.03	22.34	21.87	**0.00**	**12.08**	**5.17**
T30I3	17.44	20.10	19.11	18.59	21.54	20.02	16.97	18.48	17.73	19.83	20.40	20.13	**0.00**	**6.49**	**4.02**
T30I4	23.10	25.43	24.36	25.36	26.99	26.03	22.83	23.99	23.35	25.89	27.49	26.94	**0.00**	**12.61**	**9.29**
T30I5	12.55	14.21	13.43	13.08	15.69	14.57	11.76	12.61	12.12	12.49	13.05	12.95	**0.00**	**7.55**	**3.24**
T30I6	18.76	20.92	19.98	18.86	22.44	20.64	18.25	19.69	18.79	21.01	22.46	22.09	**0.00**	**12.41**	**5.83**
T30I7	19.33	21.33	20.40	21.27	24.02	22.54	18.14	19.41	18.86	20.67	22.54	21.78	**0.00**	**14.38**	**8.72**
T30I8	17.52	19.30	18.38	19.37	22.15	20.64	16.90	17.85	17.45	19.52	21.14	20.30	**0.00**	**13.39**	**7.28**
T30I9	19.32	21.45	20.25	20.45	23.11	21.67	18.05	18.82	18.42	20.44	21.23	20.88	**0.00**	**14.50**	**7.53**
T30I10	21.20	22.58	21.91	22.21	25.19	24.04	20.13	21.16	20.56	22.69	23.93	23.40	**0.00**	**12.17**	**5.45**
T30I11	20.19	22.33	21.53	21.34	24.79	23.16	19.56	20.76	20.04	21.04	22.21	21.98	**0.00**	**13.97**	**6.47**
T30I12	17.89	18.78	18.23	18.38	19.85	19.04	17.62	18.20	17.84	17.97	18.05	18.03	**0.00**	**11.58**	**5.96**
T30I13	25.59	28.30	26.74	24.97	28.63	26.47	24.77	25.88	25.31	24.89	28.22	25.82	**0.00**	**10.47**	**6.96**
T30I14	22.17	23.69	23.09	23.15	26.39	24.89	21.38	22.65	22.10	23.87	25.68	24.95	**0.00**	**13.08**	**4.50**
T30I15	25.82	27.90	26.99	27.46	31.45	29.49	25.44	26.78	26.07	27.38	28.46	27.90	**0.00**	**17.13**	**12.20**
T30I16	15.80	17.15	16.45	16.18	19.47	18.19	14.97	16.21	15.79	16.26	18.02	17.14	**0.00**	**8.74**	**5.35**
T30I17	23.74	26.31	25.06	24.95	27.79	25.95	22.34	24.42	23.54	24.26	25.22	25.01	**0.00**	**14.80**	**9.12**
T30I18	23.61	25.54	24.45	23.37	25.27	24.19	22.63	23.64	23.04	24.51	25.82	25.24	**0.00**	**11.48**	**8.53**
T30I19	12.78	14.78	13.61	13.26	15.44	14.03	12.56	13.65	13.10	12.82	13.84	13.22	**0.00**	**10.18**	**5.06**
T30I20	23.59	25.21	24.50	24.89	25.73	25.49	22.75	23.83	23.25	25.31	26.03	25.72	**0.00**	**9.96**	**5.47**
Average	20.21	22.20	21.25	21.10	23.81	22.45	19.48	20.64	20.04	21.28	22.59	22.01	**0.00**	**11.94**	**6.64**

TABLE 7. Experimental outcomes for the instances comprising of 40 tasks.

Instance	DABC			DIWO			IG			IIG			DJaya		
	Min	Max	Ave	Min	Max	Ave	Min	Max	Ave	Min	Max	Ave	Min	Max	Ave
T40I1	24.89	26.63	25.84	27.12	30.03	28.94	24.03	25.78	24.98	26.86	28.95	28.20	**0.00**	**12.18**	**7.51**
T40I2	23.10	25.75	24.47	24.43	30.22	27.41	23.52	24.99	24.22	25.55	27.10	26.50	**0.00**	**16.77**	**8.09**
T40I3	26.96	29.36	28.32	29.15	32.82	30.88	27.13	28.49	27.80	28.75	30.20	29.49	**0.00**	**10.57**	**6.58**
T40I4	25.79	28.23	27.21	26.84	30.09	28.23	26.52	27.77	26.91	28.73	30.98	30.25	**0.00**	**4.43**	**2.40**
T40I5	22.99	25.14	24.31	24.35	28.33	25.81	22.55	23.65	23.30	23.75	25.12	24.34	**0.00**	**19.76**	**10.38**
T40I6	23.66	25.63	24.70	26.42	29.91	27.64	22.90	24.97	24.18	25.97	27.84	27.07	**0.00**	**14.06**	**6.96**
T40I7	20.89	23.66	22.29	23.49	25.34	24.58	20.71	22.63	21.48	23.14	24.05	23.59	**0.00**	**15.46**	**8.84**
T40I8	23.35	24.87	24.25	26.06	42.71	32.96	23.41	24.48	23.89	25.74	27.85	27.06	**0.00**	**15.74**	**9.05**
T40I9	21.32	23.29	22.58	23.34	26.13	24.76	21.25	22.38	21.74	22.58	24.09	23.34	**0.00**	**11.92**	**8.09**
T40I10	30.28	31.79	31.12	30.95	34.73	32.45	29.79	31.54	30.66	31.54	33.11	32.56	**0.00**	**19.18**	**11.71**
T40I11	25.46	27.59	26.28	25.61	29.02	27.53	24.69	26.32	25.64	27.25	29.07	28.49	**0.00**	**14.90**	**10.28**
T40I12	17.73	18.86	18.21	19.33	23.37	20.98	17.76	18.74	18.29	19.06	20.69	19.91	**0.00**	**14.97**	**8.30**
T40I13	33.10	35.23	34.16	34.04	35.96	35.11	32.21	34.51	33.52	34.23	36.58	35.44	**0.00**	**25.21**	**10.22**
T40I14	25.31	27.17	26.16	27.34	31.25	29.49	24.10	26.20	25.46	26.65	28.35	27.58	**0.00**	**15.12**	**8.78**
T40I15	20.57	22.20	21.25	22.72	25.51	24.17	20.36	21.85	21.01	22.58	23.48	23.01	**0.00**	**14.97**	**8.44**
T40I16	25.36	27.10	26.36	27.31	30.04	28.68	24.99	26.73	25.91	26.57	27.82	27.36	**0.00**	**15.84**	**11.15**
T40I17	38.16	40.94	40.12	41.61	46.72	44.31	39.01	40.73	39.78	41.57	43.79	42.96	**0.00**	**25.98**	**19.27**
T40I18	26.49	28.25	27.32	28.02	30.65	29.26	25.82	28.15	26.99	28.61	30.99	30.56	**0.00**	**16.17**	**7.89**
T40I19	17.80	19.27	18.72	19.21	22.85	20.49	17.91	19.16	18.72	19.42	20.23	19.90	**0.00**	**15.19**	**8.38**
T40I20	24.85	26.06	25.58	25.78	27.99	26.75	23.80	25.33	24.66	26.39	28.24	27.43	**0.00**	**11.70**	**5.58**
Average	24.90	26.85	25.96	26.66	30.68	28.52	24.62	26.22	25.46	26.75	28.43	27.75	**0.00**	**15.51**	**8.89**

accurate experimental results, each algorithm operates on 100 instances, and each instance is repeated for 30 times. The experimental results of each algorithm are obtained after the CPU reaches a predefined 5s. The experimental results consist of minimum (Min), maximum (Max) and average values (Ave) of RPI. We choose four comparison algorithms, namely Discrete Artificial Bee Colony algorithm (DABC) [8], Discrete Invasive Weed Optimization algorithm (DIWO) [19], Greedy Iterative algorithm (IG) [21], Improved Greedy Iterative algorithm (IIG) [18].

In this section, all the algorithms are compared using instances with a task size of 10 to validate the effectiveness of the algorithms in handling small-sized tasks. The average RPI

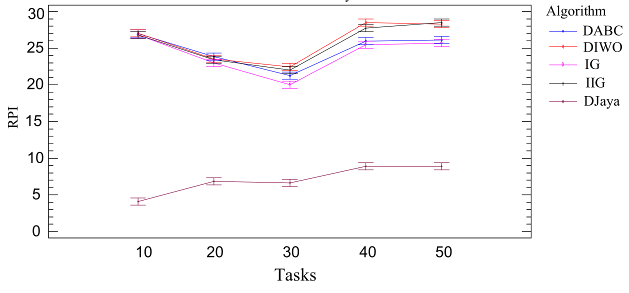

FIGURE 11. The RPI of algorithms on all tasks.

values for DABC, DIWO, IG, IIG, and DJaya are 27.01%, 27.03%, 26.84%, 26.84%, and 4.11%, respectively in Table 4.

TABLE 8. Experimental outcomes for the instances comprising of 50 tasks.

Instance	DABC			DIWO			IG			IIG			DJaya		
	Min	Max	Ave	Min	Max	Ave	Min	Max	Ave	Min	Max	Ave	Min	Max	Ave
T50I1	28.67	29.97	29.22	28.59	30.56	29.48	26.86	29.04	28.23	30.17	32.79	31.50	**0.00**	**14.30**	**8.78**
T50I2	25.74	27.79	26.73	27.24	28.57	28.06	25.28	27.34	26.36	26.71	27.99	27.42	**0.00**	**14.98**	**8.62**
T50I3	22.32	23.87	23.24	24.51	28.03	26.47	22.09	24.16	22.99	26.46	28.51	27.47	**0.00**	**8.15**	**2.88**
T50I4	31.05	32.54	31.94	32.86	35.59	34.09	30.84	32.37	31.63	33.71	34.89	34.34	**0.00**	**22.91**	**10.30**
T50I5	25.08	26.75	25.63	26.01	29.93	27.36	24.22	26.21	25.10	25.91	26.85	26.50	**0.00**	**18.35**	**11.67**
T50I6	22.50	24.70	24.11	24.03	27.47	25.85	22.86	24.69	23.75	24.09	24.97	24.63	**0.00**	**15.30**	**7.60**
T50I7	20.31	22.15	21.40	21.05	24.95	23.21	20.04	22.34	21.22	21.61	23.58	22.78	**0.00**	**11.79**	**6.36**
T50I8	31.05	32.41	31.85	34.14	36.60	34.89	30.78	32.61	31.52	34.61	37.18	36.27	**0.00**	**22.96**	**15.94**
T50I9	26.42	28.18	27.12	28.75	30.49	29.61	25.74	27.17	26.52	29.27	31.14	30.50	**0.00**	**21.40**	**11.17**
T50I10	26.15	27.88	26.80	28.58	31.84	30.35	25.41	26.97	26.25	27.46	29.57	28.60	**0.00**	**13.78**	**7.68**
T50I11	25.67	27.74	26.87	27.65	30.58	29.26	25.36	27.13	26.39	29.25	31.21	30.41	**0.00**	**19.65**	**10.92**
T50I12	21.52	22.87	22.16	23.26	26.43	25.24	21.37	22.91	22.09	23.60	25.41	24.58	**0.00**	**16.20**	**11.81**
T50I13	31.10	33.23	31.98	32.06	36.21	33.12	30.58	32.04	31.41	32.74	36.48	35.61	**0.00**	**21.54**	**10.16**
T50I14	27.32	28.87	28.31	28.46	31.40	29.57	26.82	28.47	27.73	29.85	30.68	30.27	**0.00**	**15.68**	**8.23**
T50I15	19.79	21.20	20.57	21.33	26.20	23.19	19.41	21.05	20.19	22.57	23.45	23.06	**0.00**	**11.96**	**6.45**
T50I16	20.54	21.89	21.21	22.07	25.18	24.16	19.33	21.73	20.62	22.81	24.34	23.70	**0.00**	**16.23**	**7.46**
T50I17	30.32	32.22	31.58	31.38	34.76	32.93	30.09	32.15	31.11	32.38	33.63	33.10	**0.00**	**22.96**	**11.92**
T50I18	26.37	28.49	27.51	28.96	33.61	31.38	26.52	28.03	27.42	27.93	30.91	29.50	**0.00**	**14.25**	**7.15**
T50I19	20.97	22.60	21.72	21.48	25.27	22.61	20.68	22.25	21.58	22.32	24.10	23.09	**0.00**	**18.69**	**8.01**
T50I20	22.14	23.42	22.75	23.90	24.94	24.52	21.18	22.95	22.22	24.93	27.36	26.58	**0.00**	**11.33**	**5.09**
Average	25.25	26.94	26.13	26.82	29.93	28.27	24.77	26.58	25.72	27.42	29.25	28.49	**0.00**	**16.62**	**8.91**

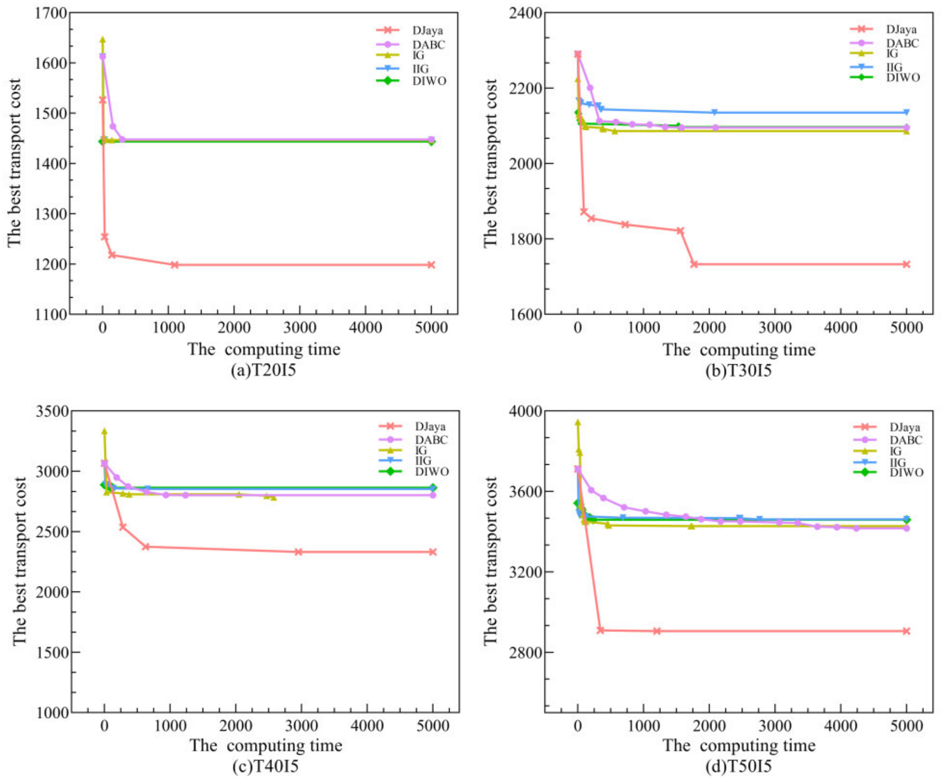

FIGURE 12. The convergence curves of the algorithms.

It can be seen that the DJaya algorithm exhibits the lowest RPI value, which is highlighted in black in the table and outperforms the other algorithms.

Table 5-8 correspond to instances with task sizes of 20, 30, 40, and 50, respectively. The average RPI values for the DJaya algorithm in these instances are 6.83%, 6.64%, 8.89% and 8.91% respectively. It can be observed that the DJaya algorithm has the lowest RPI value compared to the other algorithms, which is highlighted in black in the tables. To further prove the effectiveness of the presented algorithm, a statistical method called multi-factor analysis of variance (ANOVA) is adopted, considering the RPI values obtained from all algorithms and the factors of comparison algorithms and task scales. The Fig.10 presents 95% confidence interval mean values for the five comparison algorithms. It is evident that the DJaya algorithm has the smallest RPI value among the five algorithms, indicating its superior performance. The Fig.11 also illustrates the correlation between five comparison algorithms and task scales. The horizontal axis stands for task scale, and the vertical axis stands for the RPI value. It can be observed that the DJaya algorithm consistently achieves the lowest RPI value across the five task sizes. While the differences in RPI values among the other four algorithms are not significant, they all exhibit a considerable gap compared to the DJaya algorithm, highlighting its effectiveness in addressing the proposed problem. Therefore, the DJaya algorithm is feasible

The above content can be summarized as follows:

(1) The DJaya algorithm outperforms the other four algorithms in the task size range of 10-50.

(2) The DJaya algorithm consistently achieves the lowest RPI value across all task sizes.

(3) The DJaya algorithm exhibits excellent performance and effectiveness in addressing the proposed problem.

E. COMPARISON OF CONVERGENCE

To evaluate the convergence performance of the algorithm presented, an evolution plot can effectively illustrate the changes in transportation costs for AGVs at different time points within a cycle. It was mentioned earlier that the CPU running time is 5 seconds. It is shown in Fig.12(a)-(d), the x-axis represents various time points in a single production cycle, and the y-axis indicates the attained minimum transportation cost.

These four instances correspond to task sizes of 20, 30, 40, and 50 tasks. From the Fig.12, it is evident that the performance of the DABC, DIWO, IIG, and IG algorithms exhibits certain differences compared to the DJaya algorithm across the four different scale instances.

In summary, by comparing the maximum, minimum, and average RPI values of different algorithms at different scales, as well as convergence graphs and algorithm comparison graphs, we have validated that presented DJaya algorithm is more valid than the other four comparison algorithms in handling our problem.

VII. CONCLUSION AND FUTURE RESEARCH

This paper has investigated the AGV scheduling problem with dynamic unloading time in a matrix manufacturing workshop. As far as we know, there are relatively few studies on this problem. The objective is to minimize the overall transportation cost, which contains travel cost, penalty cost, and vehicle cost. To solve this problem, this paper provides a comprehensive solution, proposes an efficient Jaya (DJaya) algorithm, and comprehensively evaluates all algorithms on 110 examples based on Foxconn Technology Group. In the experimental part, the parameters of the algorithm are first calibrated, in order to determine under which parameter the algorithm can achieve the best effect. Secondly, the effectiveness of the proposed heuristic algorithm and the DJaya operators is proven, and it is proven that the proposed strategies are effective. Then, all the algorithms are tested under different task sizes, and it is proven that the proposed algorithms can achieve the best effect under different task sizes. Finally, the convergence performance of all the algorithms is compared. It can intuitively see the change of AGV transportation cost at different time points in a cycle. The experimental results show that DJaya algorithm is superior to the other four algorithms in solving the problem studied. Overall, main contributions are as follows:

(1) Formulated the MAGVSDUT and established a mixed-integer linear programming model.

(2) Proposed a heuristic based on ant colony algorithm to generate high quality initial solutions.

(3) Proposed an efficient DJaya algorithm with advanced techniques, such as two DJaya operators in the updating mechanism for balancing algorithmic exploitation and exploration, and a sequence insertion operator for facilitating the population find better solutions.

This study focuses on the normal production conditions of the workshop. In future research work, we will further focus on the following topics:

(1) We should consider other characteristics of the studied problems, such as the occurrence of uncertain factors like AGV failures, AGV variable speeds, AGV collisions, the problem of multi-objective [38], [39], [40], [41], a limited number of AGVs [42], and energy saving problems [43].

(2) We will also explore more problem-oriented strategies to enhance DJaya algorithm, such as collaborative strategy [44], [45].and local search methods.

(3) Applying the DJaya algorithm to closely related scheduling problems such as flow shop scheduling problem, job shop scheduling problem, and dynamic production scheduling.

REFERENCES

[1] J. Zhang, Y. Lv, Y. Li, and J. Liu, "An improved QMIX-based AGV scheduling approach for material handling towards intelligent manufacturing," in *Proc. IEEE 20th Int. Conf. Embedded Ubiquitous Comput. (EUC)*, Dec. 2022, pp. 54–59.

[2] Y. Xiong and J. L. Mao, "Research on scheduling of multi-load AGV system with limited buffer capacity," *Electron. Sci. Technol.*, vol. 31, pp. 72–77, 2018.

[3] Y. Xi, N. B. Ahmad, and A. Al Mamun, "Research on improving e-commerce logistics service customer satisfaction through application of AGV in intelligent warehouse," in *Proc. Int. Conf. Emerg. Technol. Intell. Syst.*, vol. 2, 2022, pp. 61–73.

[4] L. Ou, J. Peng, J. Chen, X. Zou, F. Sun, and B. Yang, "Research on multi AGV control system and scheduling algorithm of warehouse automation," in *Proc. Int. Conf. Frontier Computing.*, 2021, pp. 1710–1716.

[5] I. Aziez, J.-F. Cǒté, and L. C. Coelho, "Fleet sizing and routing of healthcare automated guided vehicles," *Transp. Res. E, Logistics Transp. Rev.*, vol. 161, May 2022, Art. no. 102679.

[6] 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, Dec. 2018.

[7] C. Kui, B. Li, and W. Wenya, "Research on integrated scheduling of AGV and machine in flexible job shop," *J. Syst. Simul.*, vol. 34, no. 3, pp. 461–469, 2022.

[8] 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," *Exp. Syst. Appl.*, vol. 161, Dec. 2020, Art. no. 113675.

[9] 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, Nov. 2011.

[10] X. Ge, L. Li, and H. Chen, "Research on online scheduling method for flexible assembly workshop of multi-AGV system based on assembly island mode," in *Proc. IEEE 7th Int. Conf. Cloud Comput. Intell. Syst. (CCIS)*, Nov. 2021, pp. 371–375.

[11] L. Meng, C. Zhang, Y. Ren, B. Zhang, and C. Lv, "Mixed-integer linear programming and constraint programming formulations for solving distributed flexible job shop scheduling problem," *Comput. Ind. Eng.*, vol. 142, Apr. 2020, Art. no. 106347.

[12] M. Tian, H. Sang, W. Zou, Y. Wang, M. Miao, and L. Meng, "Joint scheduling of AGVs and parallel machines in an automated electrode foil production factory," *Exp. Syst. Appl.*, vol. 238, Mar. 2024, Art. no. 122197.

[13] D. B. M. M. Fontes and S. M. Homayouni, "Joint production and transportation scheduling in flexible manufacturing systems," *J. Global Optim.*, vol. 74, no. 4, pp. 879–908, Aug. 2019.

[14] Y. Hu, H. Yang, and Y. Huang, "Conflict-free scheduling of large-scale multi-load AGVs in material transportation network," *Transp. Res. E, Logistics Transp. Rev.*, vol. 158, Feb. 2022, Art. no. 102623.

[15] H. Tang, X. Cheng, W. Jiang, and S. Chen, "Research on equipment configuration optimization of AGV unmanned warehouse," *IEEE Access*, vol. 9, pp. 47946–47959, 2021.

[16] H. Niu, W. Wu, Z. Xing, X. Wang, and T. Zhang, "A novel multi-tasks chain scheduling algorithm based on capacity prediction to solve AGV dispatching problem in an intelligent manufacturing system," *J. Manuf. Syst.*, vol. 68, pp. 130–144, Jun. 2023.

[17] Q. Liu, N. Wang, J. Li, T. Ma, F. Li, and Z. Gao, "Research on flexible job shop scheduling optimization based on segmented AGV," *Comput. Model. Eng. Sci.*, vol. 134, no. 3, pp. 2073–2091, 2023.

[18] X.-J. Zhang, H.-Y. Sang, J.-Q. Li, Y.-Y. Han, and P. Duan, "An effective multi-AGVs dispatching method applied to matrix manufacturing workshop," *Comput. Ind. Eng.*, vol. 163, Jan. 2022, Art. no. 107791.

[19] Z.-K. Li, H.-Y. Sang, J.-Q. Li, Y.-Y. Han, K.-Z. Gao, Z.-X. Zheng, and L.-L. Liu, "Invasive weed optimization for multi-AGVs dispatching problem in a matrix manufacturing workshop," *Swarm Evol. Comput.*, vol. 77, Mar. 2023, Art. no. 101227.

[20] 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, Dec. 2022, Art. no. 108755.

[21] 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, Feb. 2021, Art. no. 106945.

[22] Y.-Z. Li, J.-Z. Zou, Y.-L. Jia, L.-L. Meng, and W.-Q. Zou, "An improved genetic algorithm for multi-AGV dispatching problem with unloading setup time in a matrix manufacturing workshop," *Int. J. Ind. Eng. Comput.*, vol. 14, no. 4, pp. 767–784, 2023.

[23] W. Zou, J. Zou, H. Sang, L. Meng, and Q. Pan, "An effective population-based iterated greedy algorithm for solving the multi-AGV scheduling problem with unloading safety detection," *Inf. Sci.*, vol. 657, Feb. 2024, Art. no. 119949.

[24] W.-Q. Zou, Q.-K. Pan, L.-L. Meng, H.-Y. Sang, Y.-Y. Han, and J.-Q. Li, "An effective self-adaptive iterated greedy algorithm for a multi-AGVs scheduling problem with charging and maintenance," *Exp. Syst. Appl.*, vol. 216, Apr. 2023, Art. no. 119512.

[25] X. Wang, W. Zou, L. Meng, B. Zhang, J. Li, and H. Sang, "Effective metaheuristic and rescheduling strategies for the multi-AGV scheduling problem with sudden failure," *Exp. Syst. Appl.*, vol. 250, Sep. 2024, Art. no. 123473.

[26] R. Venkata Rao, "Jaya: A simple and new optimization algorithm for solving constrained and unconstrained optimization problems," *Int. J. Ind. Eng. Comput.*, vol. 7, no. 1, pp. 19–34, 2016.

[27] J. Mumtaz, Z. Guan, L. Yue, Z. Wang, S. Ullah, and M. Rauf, "Multi-level planning and scheduling for parallel PCB assembly lines using hybrid spider monkey optimization approach," *IEEE Access*, vol. 7, pp. 18685–18700, 2019.

[28] M. Rauf, Z. Guan, L. Yue, Z. Guo, J. Mumtaz, and S. Ullah, "Integrated planning and scheduling of multiple manufacturing projects under resource constraints using raccoon family optimization algorithm," *IEEE Access*, vol. 8, pp. 151279–151295, 2020.

[29] Q. S. Khalid, M. Arshad, S. Maqsood, M. Jahanzaib, A. R. Babar, I. Khan, J. Mumtaz, and S. Kim, "Hybrid particle swarm algorithm for products' scheduling problem in cellular manufacturing system," *Symmetry*, vol. 11, no. 6, p. 729, May 2019.

[30] G. Wang, Y. Chen, J. Mumtaz, and L. Zhu, "A study of mixed-flow human-machine collaborative disassembly line balancing problem based on improved artificial fish swarm algorithm," *Eng. Proc.*, vol. 45, no. 1, p. 40, 2023.

[31] K. Gao, Y. Zhang, A. Sadollah, A. Lentzakis, and R. Su, "Jaya, harmony search and water cycle algorithms for solving large-scale real-life urban traffic light scheduling problem," *Swarm Evol. Comput.*, vol. 37, pp. 58–72, Dec. 2017.

[32] R. Buddala and S. S. Mahapatra, "An integrated approach for scheduling flexible job-shop using teaching–learning-based optimization method," *J. Ind. Eng. Int.*, vol. 15, no. 1, pp. 181–192, Mar. 2019.

[33] R. H. Caldeira and A. Gnanavelbabu, "Solving the flexible job shop scheduling problem using an improved Jaya algorithm," *Comput. Ind. Eng.*, vol. 137, Nov. 2019, Art. no. 106064.

[34] J. Fan, W. Shen, L. Gao, C. Zhang, and Z. Zhang, "A hybrid Jaya algorithm for solving flexible job shop scheduling problem considering multiple critical paths," *J. Manuf. Syst.*, vol. 60, pp. 298–311, Jul. 2021.

[35] K. Gao, A. Sadollah, Y. Zhang, R. Su, and K. G. J. Li, "Discrete Jaya algorithm for flexible job shop scheduling problem with new job insertion," in *Proc. 14th Int. Conf. Control, Autom., Robot. Vis. (ICARCV)*, Nov. 2016, pp. 1–5.

[36] R. Ruiz and C. Maroto, "A genetic algorithm for hybrid flowshops with sequence dependent setup times and machine eligibility," *Eur. J. Oper. Res.*, vol. 169, no. 3, pp. 781–800, Mar. 2006.

[37] H.-Y. Sang, Q.-K. Pan, P.-Y. Duan, and J.-Q. Li, "An effective discrete invasive weed optimization algorithm for lot-streaming flowshop scheduling problems," *J. Intell. Manuf.*, vol. 29, no. 6, pp. 1337–1349, Aug. 2018.

[38] J.-Q. Li, X.-R. Tao, B.-X. Jia, Y.-Y. Han, C. Liu, P. Duan, Z.-X. Zheng, and H.-Y. Sang, "Efficient multi-objective algorithm for the lot-streaming hybrid flowshop with variable sub-lots," *Swarm Evol. Comput.*, vol. 52, Feb. 2020, Art. no. 100600.

[39] B. Zhang, Q.-K. Pan, L. Gao, L.-L. Meng, X.-Y. Li, and K.-K. Peng, "A three-stage multiobjective approach based on decomposition for an energy-efficient hybrid flow shop scheduling problem," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 50, no. 12, pp. 4984–4999, Dec. 2020.

[40] L. Meng, C. Zhang, B. Zhang, K. Gao, Y. Ren, and H. Sang, "MILP modeling and optimization of multi-objective flexible job shop scheduling problem with controllable processing times," *Swarm Evol. Comput.*, vol. 82, Oct. 2023, Art. no. 101374.

[41] X. Han, W. Cheng, L. Meng, B. Zhang, K. Gao, C. Zhang, and P. Duan, "A dual population collaborative genetic algorithm for solving flexible job shop scheduling problem with AGV," *Swarm Evol. Comput.*, vol. 86, Apr. 2024, Art. no. 101538.

[42] L. Meng, W. Cheng, B. Zhang, W. Zou, W. Fang, and P. Duan, "An improved genetic algorithm for solving the multi-AGV flexible job shop scheduling problem," *Sensors*, vol. 23, no. 8, p. 3815, Apr. 2023.

[43] L. Meng, P. Duan, K. Gao, B. Zhang, W. Zou, Y. Han, and C. Zhang, "MIP modeling of energy-conscious FJSP and its extended problems: From simplicity to complexity," *Expert Syst. Appl.*, vol. 241, May 2024, Art. no. 122594.

[44] 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 Trans. Evol. Comput.*, vol. 27, no. 3, pp. 430–444, Jun. 2023.

[45] 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, Jul. 2022.

YINGYING CUI received the Bachelor of Engineering degree from the Department of Mathematics and Information Engineering, Dongchang College, Liaocheng University, Liaocheng, China. She is currently pursuing the master's degree with Liaocheng University. Her research interests include intelligent optimization and scheduling.

LEILEI MENG received the B.S. degree in mechanical engineering from Chang'an University, Xi'an, China, in 2014, and the Ph.D. degree from the School of Mechanical Science and Engineering, Huazhong University of Science and Technology, Wuhan, China, in 2020. He is currently a Lecturer with the School of Computer Science, Liaocheng University, Liaocheng, China. His research interests include modeling, optimization of scheduling problems, tool wear prediction, and sustainable manufacturing.

BAOXIAN JIA received the Ph.D. degree in educational big data from Tsinghua University, in 2019. He is currently an Associate Professor with the School of Computer Science, Liaocheng University. His research interests include education big data and e-commerce.

BIAO ZHANG received the B.S. degree from Shandong University of Technology, Zibo, China, in 2012, and the Ph.D. degree from Huazhong University of Science and Technology, Wuhan, China, in 2019. He is currently a Lecturer with the School of Computer Science, Liaocheng University, Liaocheng, China. He has authored more than 40 refereed articles. His research interests include machine learning and intelligent optimization. He is a Guest Editor of *Symmetry*.

HONGYAN SANG received the M.S. degree from the School of Computer Science, Liaocheng University, Liaocheng, China, in 2010, and the Ph.D. degree in industrial engineering from Huazhong University of Science Technology, Wuhan, China, in 2013. Since 2003, she has been with the School of Computer Science, Liaocheng University, where she became a Professor, in 2021.

WENQIANG ZOU received the M.S. degree in computer application technology from Liaoning University of Science and Technology, in 2011. He studied for a Ph.D. degree with the School of Mechatronic Engineering and Automation, Shanghai University, Shanghai, China. He is working with the School of Computer Science, Liaocheng University. His research interests include AGV scheduling and routing, intelligent optimization, and scheduling algorithms.