processes

MDPI

Article

An Ant Colony Optimization-Simulated Annealing Algorithm for Solving a Multiload AGVs Workshop Scheduling Problem with Limited Buffer Capacity

Zishi Wang and Yaohua Wu *

School of Control Science and Engineering, Shandong University, Jinan 250061, China
* Correspondence: yaohua.wu@sdu.edu.cn; Tel.: +86-13906400525

Abstract: In this paper, we address a multiload AGVs workshop scheduling problem with limited buffer capacity. This has important theoretical research value and significance in the manufacturing field in considering the efficient multiload AGVs widely used today, and in the limited buffer area in production practice. To minimize the maximum completion time, an improved ant colony optimization-simulated annealing algorithm based on a multiattribute dispatching rule is proposed. First, we introduce a multiattribute dispatching rule, which combines two attributes, delivery completion time and input queue through dynamic weights that are determined by the information about the system, using the multiattribute dispatching rule to construct the initial solution. Then, with the ant colony optimization-simulated annealing algorithm as the basic framework, we propose a method for calculating transfer probability based on the multiattribute dispatching rule, which obtains heuristic information through the proposed rule. Further, we propose a path branch mechanism and dynamic equilibrium mechanism, aiming to efficiently construct the ant path and dynamically adjust ant path distribution. We propose a key job strategy and design a 2-opt neighborhood search method based on key jobs. Data experiments demonstrate the multiattribute dispatching rule is superior over other heuristic dispatching rules; the algorithm improvement strategies proposed are effective when used simultaneously or separately. Further, the proposed algorithm in this paper is superior over other heuristic algorithms and adapts to all kinds of instances.

Keywords: multiload AGV; limited buffer capacity; multiattribute dispatching rule; ant colony optimization algorithm; simulated annealing algorithm

Citation: Wang, Z.; Wu, Y. An Ant Colony Optimization-Simulated Annealing Algorithm for Solving a Multiload AGVs Workshop Scheduling Problem with Limited Buffer Capacity. *Processes* **2023**, *11*, 861. https://doi.org/10.3390/pr11030861

Academic Editor: Jean-Pierre Corriou

Received: 13 February 2023
Revised: 4 March 2023
Accepted: 10 March 2023
Published: 14 March 2023

1. Introduction

With the proposal of "Industry 4.0" and "Made in China 2025", intelligent manufacturing has become the general trend, and the manufacturing industry is gradually transforming to the direction of intelligence and automation. The automatic guide vehicle (AGV) can automatically transport jobs, which is an important support for the efficient operation of the intelligent manufacturing system. As a new type of AGV that can carry multiple jobs at a time, the multiload AGV (MLAGV) has many advantages over the traditional single load AGV (SLAGV), such as smaller fleet size, the better utilization of the vehicle and greater throughput. In machining processing, mold processing, sheet metal processing and other industries, MLAGV has been gradually used to replace SLAGV. Compared with SLAGV workshop scheduling, when performing their pickup and delivery tasks MLAGV workshop scheduling is more complex since MLAGVs have to consider several loads, including the loads that are currently on them, and the loads that they are going to pick up, instead of only one load as in SLAGVs [1]. Meanwhile, machine buffer capacity is limited in most practical manufacturing shops. Therefore, there are significant theoretical research values, and it is important to study the multiload AGVs workshop scheduling problem with limited buffer capacity.

Further, AGV workshop scheduling is one of the key research problems in the field of manufacturing scheduling optimization [2]. Research on SLAGV shop floor scheduling has made great progress and a large amount of research has been published [3–11]. With an increasing number of MLAGVs being used in manufacturing systems, MLAGV workshop scheduling has received extensive attention [1]. Reference [12] proposes a hybrid genetic algorithm and a particle swarm algorithm to solve the problem of the MLAGV transportation of materials in the weaving workshop environment, aiming to maximize the machine efficiency. Reference [13] proposes an improved iterative greedy algorithm for the problem of MLAGV real-time distribution, aiming to minimize the total cost. Reference [14] proposes an improved genetic particle swarm algorithm to solve the green MLAGV workshop logistics scheduling problem, aiming to reduce the energy consumption of AGV and optimize the AGV path. However, there are few studies on the MLAGV workshop scheduling problem, where there is more than one AGV in the workshop [15]. Reference [15] researches the MLAGV scheduling method for workshop logistics. Reference [16] proposes a genetic algorithm for the scheduling model of the hybrid SLAGV and MLAGV, to minimize the total logistics cost. Reference [17] proposes an improved differential evolution algorithm for the collaborative optimization problem of MLAGV scheduling and path planning, aiming at minimizing the maximum handling completion time. In terms of SLAGV workshop scheduling, and considering the limited buffer capacity, reference [18] uses a genetic algorithm for the problem of SLAGV transport jobs between machines with limited cache capacity, aiming to minimize the AGV travel distance. Reference [19] studies the configuration problem of the AGV and buffer capacity through the simulation method. In reference [20], a mathematical model of integer programming is established to solve the AGV scheduling problem, considering the buffer capacity. Reference [21] proposes a queuing network model and a decomposition of the state space method for a three-stage flexible flow shop with SLAGVs and limited buffer capacity, where every stage has a limited buffer. However, there are few studies on MLAGV shop scheduling with limited buffer capacity [22]; they mainly focus on heuristic rules. Reference [9] focusses on the first two problems and defines four MLAGV workshop scheduling problems with limited buffer capacity: task determination; delivery dispatching; pickup dispatching; and load selection. Reference [1] proposes a multiattribute method for concurrently solving pickup dispatching and the load selection above, which considers three attributes: slack time, waiting time and distance. Reference [22] proposes a hybrid gray wolf optimization and genetic algorithm for a MLAGV workshop scheduling problem with limited buffer capacity, aiming to minimize the completion time of all scheduled tasks. In addition, as a combinatorial optimization problem, AGV workshop scheduling problem can also be applicable to some novel algorithms, such as NSHFABC [23], AALO [24], etc. This paper studies the problem of MLAGV workshop scheduling with limited buffer capacity. As an AGV workshop scheduling problem, it can be solved by many heuristic scheduling rules and intelligent algorithms; however, the multiattribute dispatching rule (MADR) and the ant colony optimization-simulated annealing algorithm (ACOSA) are rarely used to solve it.

The common single attribute heuristic dispatching rules for a AGV workshop scheduling problem include the shortest travel time (STT) [25], and the smallest input queue (SIQ) [26], etc. In [27], Klein and Kim have demonstrated the superiority of multiattribute dispatching rules in different performance measures, e.g., queue length and load waiting time in each department, vehicle travel time and job completion time. Reference [25] proposes a multiattribute dispatching rule employing two attributes: output queue and travel time. Reference [28] proposes a multistage auction algorithm combining workload, the capacity occupancy of multiload AGV and the distance. However, it is difficult to determine the weights combining multiple attributes. In reference [26], the weights are determined by the information on the relative criticality of the processing and transportation subsystems. In addition, the weights are also determined by the neural network learning method [29]

and genetic algorithm [30], etc. Although the multiattribute dispatching rules have the same advantages, the optimization performance needs to be further improved.

The ant colony optimization algorithm (ACO) has the advantage of distributed computing, robustness and global optimization, and was first used to solve TSP [31]. At present, it has been applied in workshop scheduling problems [32–34] and AGV scheduling problems [35,36]. Reference [32] designs an improved ACO to solve flexible job shop scheduling problems (FJSP). Reference [33] designs a hybrid ant colony optimization algorithm (HACO) combined with a neighborhood search to solve job shop scheduling problems (JSP) and to study the influence of heuristic information on ACO. Reference [34] proposes a dynamic balance adaptive ACO for JSP, including the adaptive adjustment strategy of the evaporation coefficient, and the dynamic equilibrium mechanism of the search path. Reference [35] designs an ant colony genetic algorithm to solve the shelf task assignment problem considering AGVs, aiming to minimize all AGV total traveled distance. Reference [36] proposes a two-population ACO for AGV path planning for the problems of slow convergence, easy local optima and easy deadlock in basic ACO. A simulated annealing algorithm (SA) has simple efficiency and can avoid the issue of being too fast into the local optimum. It is one of the most commonly used metaheuristic algorithms used to solve combinatorial optimization problems, and is also applied to some workshop scheduling problems [37–39] and AGV scheduling problems [40,41]. Reference [37] designs a SA based on greedy strategy to solve the single person workshop scheduling problem considering the learning effect, aiming to minimize the maximum completion time. Reference [38] uses dynamic programming and SA to solve the single machine batch sorting problem considering the learning effect, aiming to minimize the total delay time. Reference [39] designs an improved distribution estimation algorithm based on the key path method and SA to solve JSP. Reference [40] proposed an improved genetic algorithm that combines the artificial potential field method and SA for the problems of traditional genetic algorithm, such as local optimal, slow convergence and nonshortest path length when planning the AGV path. In order to solve the problem of AGV path planning in the irregular intelligent manufacturing workshop, reference [41] proposes a two-layer environment modeling method to implement AGV path planning for population selection through SA. To combine the advantages of both ACO and SA, ACOSA is introduced in this paper. Reference [42] designs a multiobjective improvement ACOSA to solve the problem of periodic vehicle routing with time window and service selection. Reference [43] designs ACOSA for JSP. These studies show that ACOSA has some search advantage for separate VRP or workshop scheduling problems; however, ACOSA is rarely applied to workshop scheduling considering both MLAGV and limited buffer capacity.

This paper studies a MLAGV workshop scheduling problem with limited buffer capacity, aiming to minimize the maximum completion time. Further, it proposes an improved ant colony optimization-simulated annealing algorithm based on a multiattribute dispatching rule (MIACOSA), that is, an improved ACOSA based on MADR. First, the MADR, which combines delivery completion time and input queue through dynamic weights, is introduced to construct the initial solution. Then, based on ACOSA, the following algorithm improvement strategies are proposed: the transfer probability calculation method based on MADR; the path branch and dynamic equilibrium mechanism; and the key job strategy. Finally, through numerical experiments, the superiority of MADR, the effectiveness of the algorithmic improvement strategies and the superiority of MIACOSA, are all demonstrated.

2. Problem Definition and Mathematical Model

2.1. Problem Definition

In this paper, the MLAGV workshop scheduling system with limited buffer capacity includes: MLAGV; machines with limited buffer capacity; loading point; and guide network. The construction of the system is shown in Figure 1. Each machine needs to process a fixed number of jobs. Each job contains only one operation, which can be processed on any machine. All jobs are initially located at a loading point. Several MLAGVs start from the

loading point and they are responsible for delivering each job to the machine. Meanwhile, we also know: the processing time of each machine; the transportation time between the machine and the machine, and between the machine and the loading point; the capacity of the machine input buffer; and the carrying capacity of the MLAGV. With the goal of minimizing the maximum completion time, this paper needs to allocate the MLAGV and machine for each job, determine delivery start time and delivery completion time, together with the processing start time and processing completion time for each job.

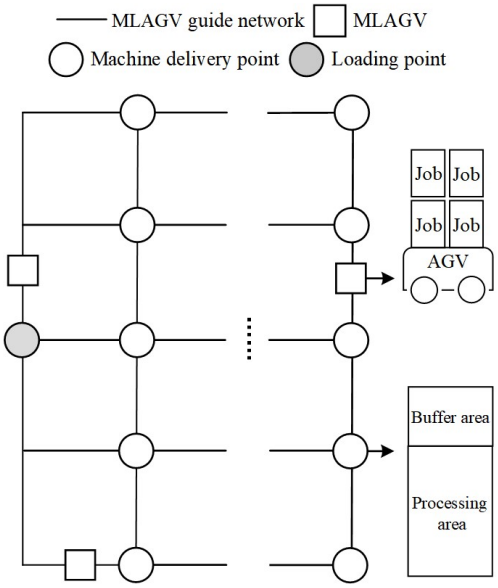

Figure 1. The construction of the MLAGV workshop scheduling system with limited buffer capacity.

Model assumptions: (1) Every job leaves the system immediately after being processed; therefore, the machine output buffer need not be considered. (2) The MLAGV always needs to be at a constant speed. (3) Each machine can only process one job at the same time. (4) The job cannot be interrupted during the processing process. (5) The mutual avoidance time of MLAGVs on the guide network can be ignored. (6) The loading and unloading time are not considered. (7) The MLAGV returns to the loading point to load the job when there is no job on its load; that is, the MLAGV gives priority to delivery over pickup. (8) Obviously, the MLAGV takes a longer time to return from the current machine to the loading point, and then to the next machine, compared to going directly to the next machine. In order to reduce the frequency of the MLAGV returning to the loading point, we assume that the MLAGV loads as many jobs as possible each time at the loading point.

2.2. Mathematical Model

To establish the problem model, the following symbols and variables are introduced in this paper: g denotes the job number; k denotes the machine number; i denotes delivery order of the jobs on the same MLAGV; a denotes the MLAGV number; M denotes the number of machines; A denotes the number of MLAGVs; n_k denotes the number of jobs required to be processed by machine k; N denotes the total number of jobs, $N = \sum_{k=1}^{M} n_k$; Q denotes the carrying capacity of the MLAGV; h denotes the capacity of the machine input buffer; p_k denotes the processing time of machine k; t_{kk^*} and t_{ok} denote the transportation time between machine k and machine k^* and between the loading point and machine k, respectively, and $t_{kk} = 0, t_{kk^*} = t_{k^*k}$, and $t_{ok} = t_{ko}$; z_{aikg} is the decision variable, if the job g is the ith job delivered by MLAGV a, and processed by machine k, $z_{aikg} = 1$, otherwise $z_{aikg} = 0$; N_a denotes the number of jobs delivered by MLAGV a; b_{akg} and d_{akg} denote delivery start time and delivery completion time, respectively, when MLAGV a delivers job g to machine k; s_{kg} and m_{kg} denote the processing start time and processing completion

time of the job g on the machine k, respectively; V denotes a large enough positive number. The mathematical model is constructed as follows:

Objective function, minimizing the maximum completion time:

$$minC_{max} = \max_{1 \le k \le M, 1 \le g \le N} \left(m_{kg} \right) \tag{1}$$

Constraints:

- Each job is allocated only one MLAGV and one machine, and can only correspond to one delivery order on a MLAGV.

$$\sum_{a=1}^{A} \sum_{k=1}^{M} \sum_{i=1}^{N_a} z_{aikg} = 1, g = 1, 2, \ldots, N; \tag{2}$$

- There must be a MLAGV that first delivers job 1.

$$\sum_{a=1}^{A} \sum_{k=1}^{M} z_{a1k1} = 1 \tag{3}$$

- If two jobs are on the same AGV, the job with the larger job number will have the larger delivery order.

$$\begin{gathered} (g - g^*)(i - i^*) + V\left(1 - z_{aikg} z_{ai^*k^*g^*}\right) \ge 0, \\ a = 1, 2, \ldots, A; i = 1, 2, \ldots, N_a; k = 1, 2, \ldots, M; \\ k^* = 1, 2, \ldots, M; g = 1, 2, \ldots, N; g^* = 1, 2, \ldots, N; \end{gathered} \tag{4}$$

- The number of jobs required for each machine is fixed.

$$\sum_{a=1}^{A} \sum_{g=1}^{N} \sum_{i=1}^{N_a} z_{aikg} = \mathrm{n}_k, k = 1, 2, \ldots, M; \tag{5}$$

- The MLAGV starts delivering after time 0.

$$d_{akg} \ge 0, a = 1, 2, \ldots, A; k = 1, 2, \ldots, M; g = 0, 1, \ldots, N; \tag{6}$$

- The first delivery of each MLAGV starts from the loading point.

$$d_{akg} \ge t_{ok} - V\left(1 - z_{a1kg}\right), a = 1, 2, \ldots, A; k = 1, 2, \ldots, M; g = 1, 2, \ldots, N; \tag{7}$$

- The delivery of the MLAGV for the job must be completed after the last delivery of the MLAGV and the transportation of the MLAGV for the job are both completed. If $mod((i-1)/Q) = 0$, the MLAGV will return to the loading point to pick up the jobs before delivering the ith job.

$$\begin{gathered} d_{akg} \ge d_{ak^*g^*} + t_{k^*o} + t_{ok} - V\left(1 - z_{aikg} z_{a(i-1)k^*g^*}\right), \\ a = 1, 2, \ldots, A; i = 2, 3, \ldots, N_a; mod((i-1)/Q) = 0; \\ k = 1, 2, \ldots, M; k^* = 1, 2, \ldots, M; g = 1, 2, \ldots, N; g^* = 1, 2, \ldots, N; \end{gathered} \tag{8}$$

$$\begin{gathered} d_{akg} \ge d_{ak^*g^*} + t_{kk^*} - V\left(1 - z_{aikg} z_{a(i-1)k^*g^*}\right), \\ a = 1, 2, \ldots, A; i = 2, 3, \ldots, N_a; mod((i-1)/Q) \ne 0; \\ k = 1, 2, \ldots, M; k^* = 1, 2, \ldots, M; g = 1, 2, \ldots, N; g^* = 1, 2, \ldots, N; \end{gathered} \tag{9}$$

- The delivery of the MLAGV for the job must be completed after the machine input buffer has reached the idle position.

$$\begin{gathered} d_{akg} \geq m_{kg^*} - h \times p_k - V\left(1 - z_{aikg}\right), \\ a = 1, 2, \ldots, A; i = 1, 2, \ldots, N_a; k = 1, 2, \ldots, M; \\ g = 2, 3, \ldots, N; a^* = 1, 2, \ldots, A; g^* = 1, 2, \ldots, g-1; \end{gathered} \tag{10}$$

In the formula, if $m_{kg^*} - h \times p_k \leq 0$, MLAGV can complete delivery without waiting for an idle location to appear in the machine input buffer. If $m_{kg^*} - h \times p_k > 0$, $m_{kg^*} - h \times p_k$ will represent the time when the machine k starts processing the job in the input buffer, that is, the input buffer of machine k starts to appear the idle position.

- The delivery start time of the job is the maximum delivery completion time of the MLAGV before the job.

$$b_{akg} = \max_{1 \leq k^* \leq M} \left(d_{ak^*(g-1)}\right), a = 1, 2, \ldots, A; k = 1, 2, \ldots, M; g = 1, 2, \ldots, N; \tag{11}$$

- The time that allows the machine to process is after time 0.

$$s_{kg} \geq 0, g = 1, 2, \ldots, N; k = 1, 2, \ldots, M; \tag{12}$$

- The processing start time of the job is after the delivery completion time of the job.

$$\begin{gathered} s_{kg} \geq d_{akg} - V\left(1 - z_{aikg}\right), a = 1, 2, \ldots, A; \\ i = 1, 2, \ldots, N_a; k = 1, 2, \ldots, M; g = 1, 2, \ldots, N; \end{gathered} \tag{13}$$

- The processing start time of the job is after the time that the machine allows for processing.

$$\begin{gathered} s_{kg} \geq m_{kg^*} - V\left(1 - z_{aikg}\right), a = 1, 2, \ldots, A; i = 1, 2, \ldots, N_a; \\ k = 1, 2, \ldots, M; g = 2, 3, \ldots, N; a^* = 1, 2, \ldots, A; g^* = 1, 2, \ldots, g-1; \end{gathered} \tag{14}$$

- The processing completion time of the job is the sum of its processing start time and the processing time.

$$m_{kg} = s_{kg} + p_k, k = 1, 2, \ldots, M; g = 1, 2, \ldots, N; \tag{15}$$

3. MIACOSA

In the basic ACO, ants search their path through a pheromone trail and heuristic information, and the higher the pheromone trail of the path, the more likely that ants will choose the path, thus gradually converging to the global optimal solution. The basic SA selectively accepts the neighborhood solutions with a variable temperature-dependent probability, and theoretically can find the global optimal solution in all random solution spaces. The ACOSA introduced in this paper alternates ACO and SA at each iteration. In each iteration, first, the ants complete their paths through ACO, and then select the optimal solution as the current solution among the ant colony path and the initial solution of this iteration. Then, the current solution enters the SA, takes the current solution updated by the SA as the initial solution of the next iteration, and finally updates the pheromone trail through the optimal solution currently searched by the algorithm, and update temperature. Based on ACOSA, MIACOSA constructs the initial solution through MADR, and proposes the following algorithm improvement strategies: (1) the transfer probability calculation method based on MADR; (2) the path branch and dynamic equilibrium mechanism; and (3) the key job strategy.

3.1. The Expression of Solution

In this paper, we design two expressions of the solution. The first expression is used for MADR and SA; the solution is represented by two sets of sequences of length N, namely MLAGV sequence π^{agv} and machine sequence π^{mac}. Further, $\pi_g^{agv} = a$ and $\pi_g^{mac} = k$ denotes that the MLAGV and machine allocated for job g are a and k, respectively. The second expression is used for ACO, where the solution is the path of the ant walking

through each node in turn; each node is a number pair of AGV and the machine processing order uniqueness number of each job. For example, the gth node is node (a, j), indicating that the AGV and machine processing order uniqueness number of job g are a and j, respectively. The machine processing order uniqueness number is used to ensure the uniqueness of each node, and the mutual correspondence with machine number in π^{mac} is achieved by arranging the jobs in the order of machine number from small to large. Figure 2 shows the mutual correspondence between machine number and the machine processing order uniqueness number in a solution; job 1 is first processed on machine 2 according to the machine number, and the corresponding machine processing order uniqueness number is 4; the machine processing order uniqueness number of job 2 is 6, the corresponding machine number for the job is 3, and so on.

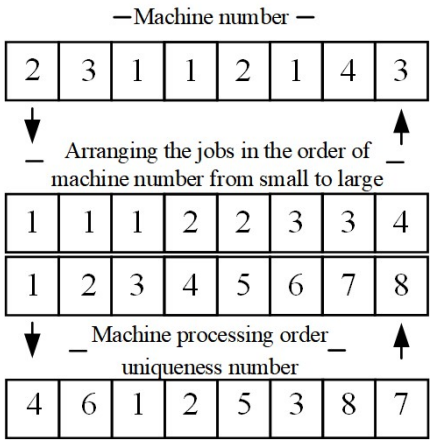

Figure 2. The mutual correspondence between machine number and the machine processing order uniqueness number in a solution.

3.2. Multiattribute Dispatching Rule

The MADR designed in this paper combines delivery completion time and input queue through dynamic weights. First, the smallest delivery completion time (SDCT) is designed, which considers the possible waiting time of MLAGV for the machine processing based on STT [25] by each time arranging the MLAGV to drive to the machine where the delivery completion time is the smallest. In SDCT, MLAGVs can complete the delivery of jobs as soon as possible, so as to shorten the maximum completion time. We also introduce SIQ [26], which is used to avoid the situation where the maximum completion time is extended because jobs wait for machine processing as much as possible. Then, the value function of each allocation scheme for each job is designed. If the allocation scheme can make the delivery completion time smaller, and the input queue of the machine in this scheme is smaller, the value function of this scheme will get a larger value. The value function D_{akg} of the allocation scheme of allocating MLAGV a and machine k for job g can be calculated as follows:

$$D_{akg} = \left(-W_g^V T_{akg}^d - W_g^P H_{kg}^k\right) / \left(W_g^V + W_g^P\right), g = 1, 2, \ldots, N; \tag{16}$$

$$T_{akg}^d = d_{akg} / \max_{1 \le a \le A, k \in allowed} \left(d_{akg}\right) \tag{17}$$

$$H_{kg}^k = h_{kg} / \max_{k \in allowed} \left(h_{kg}\right) \tag{18}$$

In the formulas, *allowed* is the set of machines that still need to process the job. The T_{akg}^d and H_{kg}^k are the normalized delivery completion time and input queue. The W_g^V and W_g^P are weights of T_{akg}^d and H_{kg}^k, initializing $W_1^V = W_1^P = 0.5$. h_{kg} is the input queue of the machine k when starting to determine the allocation scheme for the job g:

$$h_{k1} = 0, k = 1, 2, \ldots, M \tag{19}$$

$$h_{kg} = max(\lceil (\max_{1 \le g^* \le g-1} \left(m_{kg^*} \right) - \max_{1 \le a^* \le A, 1 \le g^* \le g-1} \left(d_{a^* kg^*} \right)) / p_k \rceil - 1, 0), \quad k = 1, 2, \ldots, M; g = 2, 3, \ldots, N. \tag{20}$$

By selecting the allocation scheme with the maximum value of the value function for each job, a complete scheme based on MADR can be obtained, as shown in Equation (21). This scheme is used as the initial solution of the algorithm in this paper. At the same time, the value function can improve the calculation method of transfer probability, which will be introduced in the next section.

$$\left(\pi_g^{agv}, \pi_g^{mac} \right) = arg \max_{1 \le a \le A, k \in allowed} \left(D_{akg} \right), g = 1, 2, \ldots, N \tag{21}$$

When the allocation scheme of job g is determined, the weights W_g^V and W_g^P will be updated dynamically, so that the system can "learn" its own feature throughout the update process [25]. Obviously, the longer time it takes for machines to reach their buffer capacity, the higher importance of delivery completion time is. Additionally, the longer time it takes for machines to process the jobs in their input buffer, the higher the importance of the input queue is. When starting to determine the allocation scheme for the job g ($g > 1$), the observations w_g^V and w_g^P for the importance of delivery completion time and input queue, can be calculated as follows:

$$w_g^V = \sqrt{(\sum_{k=1}^{M} p_k \left(h - h_{kg} \right) / (MN\overline{p})} \tag{22}$$

$$w_g^P = \sqrt{\left(\sum_{k=1}^{M} p_k h_{kg} \right) / (MN\overline{p})} \tag{23}$$

The weights W_g^V and W_g^P are calculated with a smooth function that reflect both observed value information and historical weight information. When $g > 1$, the weight calculation formula is:

$$W_g^V = \alpha_0 W_{g-1}^V + (1 - \alpha_0) w_g^V \tag{24}$$

$$W_g^P = \alpha_0 W_{g-1}^P + (1 - \alpha_0) w_g^P \tag{25}$$

In the formula, α_0 is the smoothing constant, and the experiments show that MADR works best when $\alpha_0 = 0.96$, so we have fixed $\alpha_0 = 0.96$.

3.3. Transfer Probability Calculation Method Based on MADR

In this paper, $(a^*, j^*) \to (a, j)$ denotes that the next node searched from the node (a^*, j^*) is the node (a, j). Further, $\tau_{a^* j^* aj}(g)$ and $\eta_{a^* j^* aj}(g)$ denote the pheromone trail and heuristic information for $(a^*, j^*) \to (a, j)$. Additionally, α and β are the control parameters of them. Further, *allowed* is the set of optional elements. In particular, the 0th node is a virtual node. The state transfer probability of $(a^*, j^*) \to (a, j)$ can be calculated as follows:

$$P_{a^* j^* aj}(g) = \begin{cases} \frac{\tau_{a^* j^* aj}^{\alpha}(g) \eta_{a^* j^* aj}^{\beta}(g)}{\sum_{a,j \in allowed} \tau_{a^* j^* aj}^{\alpha}(g) \eta_{a^* j^* aj}^{\beta}(g)}, a, j \in allowed \\ 0, otherwise \end{cases}, g = 1, 2, \ldots, N; \tag{26}$$

This paper improves the transfer probability calculation method through MADR where the value function of MADR is used as the heuristic information. For $\eta_{a^* j^* aj}(g)$, first convert the machine processing order uniqueness number j to the machine number k, and then calculate the value function D_{akg}, and let $\eta_{a^* j^* aj}(g) = D_{akg}$, and positive the heuristic information, that is, $\eta_{a^* j^* aj}(g) = \eta_{a^* j^* aj}(g) - min(\min_{a,j \in allowed} (\eta_{a^* j^* aj}(g), 0))$. Then, select the gth node based on $P_{a^* j^* aj}(g)$, and determine the MLAGV and machine allocated for job g, and then dynamically update the weights.

3.4. Path Branch and Dynamic Equilibrium Mechanism

3.4.1. Path Branch

The state node of classical ACO only contains one dimension of determining the city, while the state node of this paper contains two dimensions of determining MLAGV and the machine processing order uniqueness number. Therefore, selecting the state node according to the roulette method [36] will reduce the efficiency of the ant path search. This paper designs the path branch mechanism to build the ant path. Set the g^mth node of ant m $(m = 1, 2, \ldots, ant)$ as the branch node where ant is the number of ants. Ant m $(m = 1, 2, \ldots, ant)$ will select the node $\left(a^l, j^l\right)$ at the branch node and select nodes $\left(a^b, j^b\right)$ at the other $N - 1$ nodes. Set the path that ant select node $\left(a^b, j^b\right)$ at each node as the main path. Before the branch node, ants search along the main path; starting from the branch node, ants begin to explore new paths independently. Further, $\left(a^b, j^b\right)$ and $\left(a^l, j^l\right)$ are calculated as follows:

$$\left(a^b, j^b\right) = arg \max_{a,j \in allowed} \left\{P_{a^* j^* aj}(g)\right\} \tag{27}$$

$$\left(a^l, j^l\right) = arg \max_{\substack{a,\, j \,\in\, allowed \\ a \neq a^b \cup j \neq j^b}} \left\{P_{a^* j^* aj}(g)\right\} \tag{28}$$

In order to gradually reduce the guiding effect of MADR on the main path, β is calculated by Formula (29) where β_0 is the initial value of β, and ϕ is the decay factor of β, $\phi \in [0, 1]$. The β in the $iter$th iteration can be calculated as follows:

$$\beta = \begin{cases} \beta_0 \times \phi^{(iter-1)}, g < g^m \\ \beta_0, g \geq g^m \end{cases} \tag{29}$$

3.4.2. Dynamic Equilibrium Mechanism

To address the contradiction between fast convergence and precocious stagnation, this paper refers to reference [34] to design a dynamic equilibrium mechanism to dynamically adjust the degree of dispersion of ant colony paths. First, according to the degree of dispersion of the ant colony paths in the $iter - 1$th iteration, adjust the probability ε_g^{iter} that the g^m value of ant m $(m = 1, 2, \ldots, ant)$ in $iter$th iteration is g. Then, select a value for the g^m of ant m $(m = 1, 2, \ldots, ant)$ by the roulette method. Further, ε_g^{iter} is calculated as follows:

$$\varepsilon_g^{iter} = \frac{\sigma_{iter-1}\varepsilon_m}{\sigma_{iter-1}^{max}(N-1)}(g - N) + \varepsilon_m, g = 1, 2, \ldots, N; \tag{30}$$

In the formula: the parameter ε_m is the upper bound of ε_g^{iter}; σ_{iter} is the variance of the ant colony paths in the $iter$th iteration; and σ_{iter}^{max} is the maximum value of the variance of the ant colony paths before the $iter$th iteration. Initialized $\sigma_0 = 0$ and $\sigma_0^{max} = 1$.

The method of dynamic adjustment of the degree of dispersion of ant colony paths is described as follow: the smaller σ_{iter-1} is, the flatter the line of ε_g^{iter} is, as shown in Figure 3; the greater the probability that g^m $(m = 1, 2, \ldots, ant)$ takes small values is, each ant will have a shorter path searched along the main path and explore a longer path independently; the greater degree of dispersion of ant colony paths is, thus the larger σ_{iter} is. Similarly, the larger σ_{iter-1} is, the smaller σ_{iter} is.

3.5. Key Job Strategy

3.5.1. Key Jobs

In order to improve the neighborhood search efficiency of SA, this paper refers to the key path [33,44,45] and designs the key job strategy. In a solution, the key jobs are a set of jobs which have the longest completion time and can continuously execute transportation or processing, or transportation and processing. The neighborhood solution is produced by the adjustment of the allocation scheme of key jobs in the current solution.

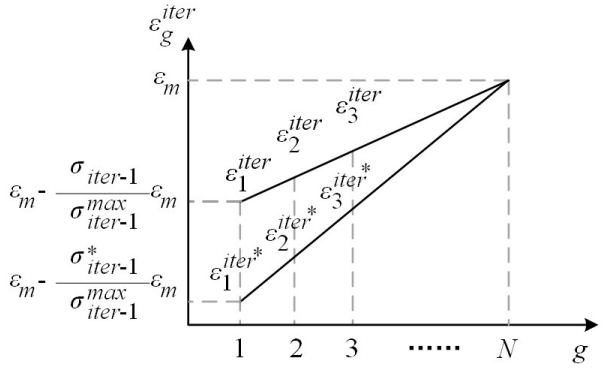

Figure 3. The line of ε_g^{iter}.

The key jobs in this paper consists of both the key jobs that do not consider MLAGV transportation, and the key jobs that do consider MLAGV transportation. First, make the following settings in a solution: Job f is the job whose processing completion time is maximum. Job g is the current key job. Note that s_g and e_g are the processing start time and processing completion time of job g, respectively. Further, s_g^a and e_g^a are the transportation start time and transportation completion time of job g, respectively. Job set J $\{j, j^*, \ldots, j^{**}\}$ is the set of jobs whose allocated MLAGV is the same as job g, and whose transportation completion time is the same as the transportation start time of job g, where $s_j < s_{j^*} < \ldots < s_{j^{**}}$. Job set R $\{r, r^*, \ldots, r^{**}\}$ is the set of jobs whose machine is the same as job set J; that is the last job group whose processing time is end to end before the delivery of job j is completed, where $s_{r^{**}} < \ldots < s_{r^*} < s_r$. u_1 is the time that the AGV may wait for an idle location to appear in the machine input buffer. Further, u_2 is the time that the job may wait for the machine to allow processing.

The key jobs that do not consider MLAGV transportation are all the jobs processed after the last idle of the machine on which job f is processed, that is, the sequence of jobs from job f to job g, as shown in Figure 4a. In Figure 4a, the processing interruption before job g is caused by the transportation of MLAGV; thus, if $s_g^a \neq 0$, the transportation of MLAGV needs to be considered when we continue searching for key jobs after job g.

The search method for the key jobs that consider MLAGV transportation is described as follow: Starting with job g, constantly search forward for the job that directly determines the transportation start time of job g, and make it as the new key job g, until $s_g^a = 0$. When $s_g^a > e_r$, as shown in Figure 4b, it means that the MLAGV transportation for job set J directly determines the transportation start time of job g. Since job j is the first delivered job in job set J, it is the most important for the transportation of job g, so job j will be regarded as the new key job g. When $s_g^a = e_r$, as shown in Figure 4c, it means that the machine processing job set R directly determines the transportation start time of job g, so the jobs in job set R will all be taken as the key jobs, and job r^{**} will be regarded as the new key job g. When $s_g^a < e_r$, that is, $s_g^a = e_j^a < e_r$, it will contradict the setting of job sets J and R, so this situation does not exist and will not be discussed.

3.5.2. Neighborhood Search Method based on Key Jobs

In this paper, a 2-opt neighborhood search method based on key jobs is designed, and the following steps are cycled L^m times: First, according to the proportion of the number of machine types M_K^π covered by the key jobs of the current solution to the number of machines M, select the job selection method: if $M_K^g/M > \mu$ where $\mu \in [0, 1]$, randomly select two jobs among the key jobs whose machines are different; otherwise, randomly select a job among the key jobs, and randomly select another job among all jobs whose machine are different from its. Then, exchange the machines of two jobs, and select the MLAGV combination that can make the objective function value optimal among all the possible combinations of the MLAGV allocating for the two jobs, thus producing a neighborhood solution. Finally, if $C_{max}^{\pi^*} < C_{max}^{\pi}$, set $\pi = \pi^*$, $C_{max}^{\pi} = C_{max}^{\pi^*}$, and if $C_{max}^{\pi^*} < C_{max}^{\pi^B}$, set $\pi^B = \pi^*$, $C_{max}^{\pi^B} = C_{max}^{\pi^*}$.

if $C_{max}^{\pi^*} \geq C_{max}^{\pi}$ and $e^{-(C_{max}^{\pi^*} - C_{max}^{\pi})/T} > rand$, set $\pi = \pi^*$, $C_{max}^{\pi} = C_{max}^{\pi^*}$, where π, π^* and π^B denote the current solution, the neighborhood solution of the current solution and the optimal solution currently searched by the algorithm, and C_{max}^{π}, $C_{max}^{\pi^*}$ and $C_{max}^{\pi^B}$ are the corresponding objective function values, respectively, with *rand* being a random number between 0 and 1. In particular, if in the key jobs of a solution, the key jobs that consider MLAGV transportation are an empty set, that is, $s_g^a = 0$ in Figure 4a, this means that the last machine to finish processing does not have idle time, thus, the solution will be the optimal solution and the algorithm will terminate.

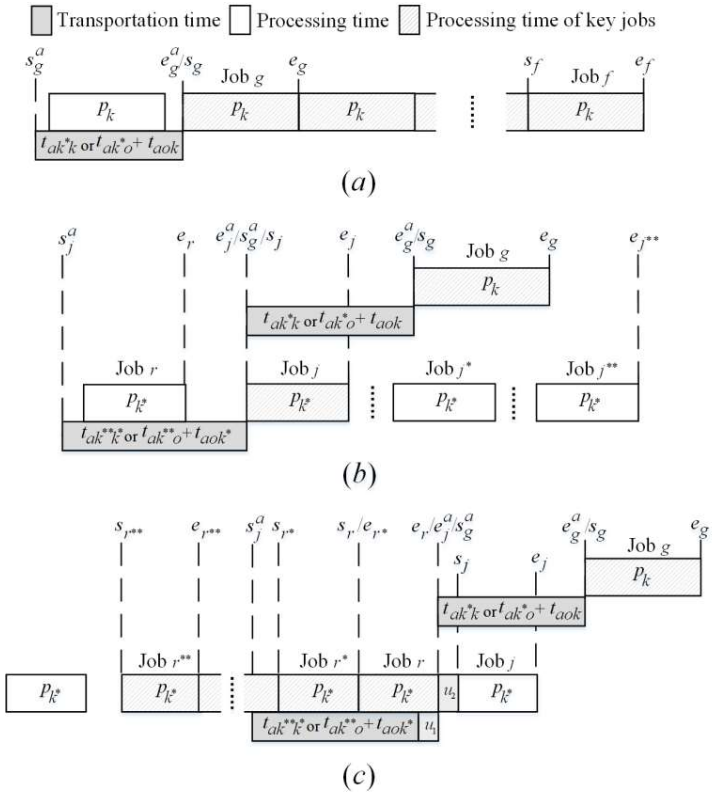

Figure 4. A schematic diagram of the key jobs. (**a**) the key jobs that do not consider MLAGV transportation. (**a**) shows the key jobs that do not consider MLAGV transportation. (**b**,**c**) show the key jobs that consider MLAGV transportation when $s_g^a > e_r$ and $s_g^a = e_r$.

3.6. Algorithm Steps

The specific description of the steps of MIACOSA is as follows:

Step 1: Construct the initial solution through MADR. Initialize $iter = 1$. Initialize $C_{max}^{\pi^B}$ as a large enough positive number and pheromone trail as 0.

Step 2: Perform ACO. Determine the branch nodes according to the dynamic equilibrium mechanism, calculate the transfer probability based on MADR, and construct the ant path according to the path branch mechanism. Then, select the optimal solution among the ant path and the initial path as the current solution, and update π^B with the optimal solution among the current solution and π^B.

Step 3: Perform the SA neighborhood search cycles based on the key jobs. In each cycle, if the current solution is judged to be the optimal solution, the algorithm is terminated and the optimal solution is output; otherwise, continue the execution as described in the above section.

Step 4: Take the current solution as the initial solution. Update the temperature $T = T \times \gamma$ and update pheromone trail.

Step 5: If the algorithm meets the termination condition, the running time of the algorithm reaches the termination time, or the number of iterations reaches the upper limit,

the algorithm will terminate, and output the optimal solution searched by the algorithm; otherwise, enter Step 2, set $iter = iter + 1$.

The update rule for pheromone trail is as follow:

$$\tau_{a^*j^*aj} = (1-\rho) \times \tau_{a^*j^*aj} + Q_m / C^{\pi^B}_{max}, (a^*, j^*) \to (a, j) \in \pi^B; \quad (31)$$

$$\tau_{a^*j^*aj} = (1-\rho) \times \tau_{a^*j^*aj}, (a^*, j^*) \to (a, j) \notin \pi^B; \quad (32)$$

4. Experimental Validation and Analysis

To demonstrate the superiority of MADR, the effectiveness of the algorithm improvement strategies and the superiority of MIACOSA, the following numerical simulation experiments are carried out in this paper: (1) comparative experiments of MADR and other heuristic rules; (2) comparative experiments of MIACOSA and other MIACOSA variant algorithms; and (3) comparative experiments of MIACOSA and other intelligent algorithms.

4.1. Data Set

In this paper, we refer to [14,26,37] and design experimental instances. In the experimental instances, the values of M of the number of machines are 10, 15 and 20; the ranges of value n_k of the number of jobs required to be processed by each machine are [1, 20], [11, 30] and [21, 40]; the range of value p_k of the processing time of each machine is [4, 12]; the values of h of the input buffer capacity of the machine are 2 and 4; the value of Q of the carrying capacity of MLAGV is 4; and the value A of the number of MLAGVs is 3. The system layout is shown as Figure 1, where the value of the transportation time of two adjacent nodes is 2. The experimental instances are named by m * n * h *, where * is a positive integer; for example, m10n1h2 is the instance whose M, the lower bound of n_k and h are 10, 1 and 2. Table 1 shows: the value of M, the range of values n_k; the values determined from the range of values n_k; and the total number N of jobs to be distributed and processed for each instance.

Table 1. The values of M, n_k and N for each instance.

Instances	M	n_k (Range of Values)	n_k (Values)	N
m10n1h2/4	10	[1, 20]	[3, 11, 12, 3, 19, 13, 15, 16, 10, 2]	104
m10n11h2/4	10	[11, 30]	[11, 14, 27, 28, 27, 27, 30, 17, 15, 20]	216
m10n21h2/4	10	[21, 40]	[22, 22, 31, 24, 25, 32, 21, 34, 29, 23]	263
m15n1h2/4	15	[1, 20]	[3, 20, 6, 17, 5, 17, 13, 7, 1, 13, 20, 13, 17, 14, 13]	179
m15n11h2/4	15	[11, 30]	[28, 11, 26, 30, 15, 13, 14, 18, 25, 28, 28, 17, 24, 27, 23]	327
m15n21h2/4	15	[21, 40]	[35, 22, 29, 36, 31, 38, 21, 28, 23, 35, 38, 40, 26, 21, 21]	444
m20n1h2/4	20	[1, 20]	[17, 4, 4, 5, 9, 18, 16, 8, 15, 7, 3, 6, 7, 17, 10, 11, 11, 4, 14, 11]	206
m20n11h2/4	20	[11, 30]	[16, 15, 28, 11, 16, 22, 29, 22, 21, 20, 16, 21, 13, 30, 12, 29, 24, 18, 16, 30]	409
m20n21h2/4	20	[21, 40]	[40, 39, 26, 25, 39, 39, 40, 34, 32, 38, 35, 25, 35, 25, 24, 30, 30, 30, 30, 34]	650

4.2. Parameter Sensitivity Analysis and Tuning

In this paper, we refer to [46,47] and use the design of experiments approach for the tuning of the main parameters in MIACOSA; that is, the number of ants ant, pheromone evaporation rate ρ and the temperature T. Other parameters are set empirically, as shown in Table 2. In the parameter tuning, ant is tested at level 0, 5, 10 and 15, ρ is tested at level 0.3, 0.5, 0.7 and 0.9, and T is tested at level 0.1, 0.3, 0.5 and 0.7. The MIACOSA of each parameter combination is calculated 10 times in each instance, with the running time of the algorithm limited to $N \times (M/2) \times 20$ milliseconds. The relative percentage deviation (RPD) of MIACOSA of a certain parameter combination in a certain instance is $(C_{max} - C^B_{max})/C^B_{max}$, where C^B_{max} is the objective function value obtained by the optimal algorithm for the instance, and C_{max} is the objective function value obtained by the algorithm of this parameter combination for the instance. Figure 5 shows the mean RPDs and 95% least significant difference (LSD) confidence intervals for MIACOSAs of different parameter values.

Table 2. The values of some parameters.

Parameters	α	β_0	ϕ	ε_m	L^m	γ	μ	Q_m
values	3	3	0.4	0.02	50	0.998	0.7	100

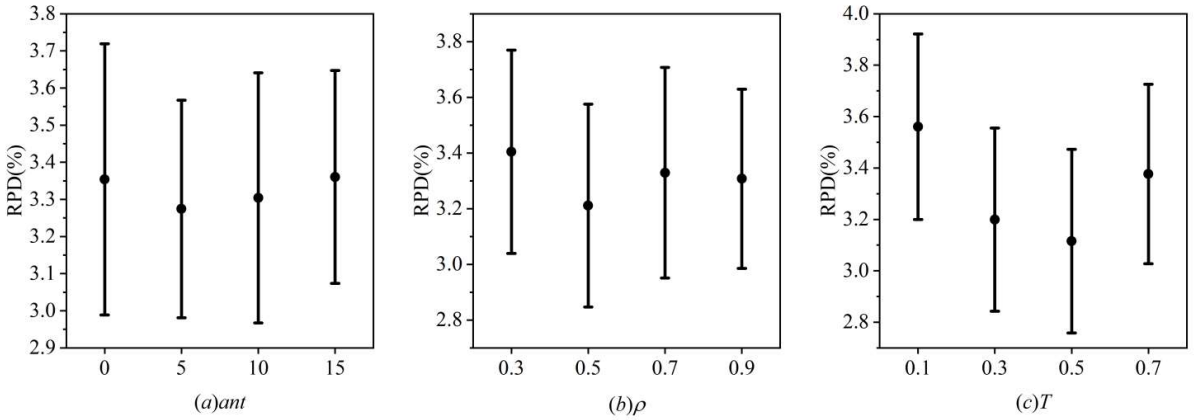

Figure 5. The mean RPDs and 95% LSD confidence intervals for MIACOSAs of different parameter values. (**a**–**c**) show the mean RPDs and 95% least significant difference (LSD) confidence intervals of different parameter values for parameter *ant*, ρ and T.

As shown in Figure 5a, the efficiency of MIACOSA when *ant* is 5 is higher than that when *ant* is 0, indicating that ACO has a certain role in MIACOSA. However, as *ant* increases, the calculation time of ACO will increase, and MIACOSA will not be able to perform a sufficient neighborhood search, so its efficiency will decrease. Thus, take *ant* as 5. As shown in Figure 5b, ρ adjusts the dependence of MIACOSA on past pheromones, with the highest efficiency when ρ is 0.5, so take ρ as 0.5. As shown in Figure 5c, with the increase of T, MIACOSA can more easily accept inferior solutions and expand the search space through SA. However, if T is too large, MIACOSA will frequently accept inferior solutions and reduce the efficiency. The highest efficiency is achieved when T is 0.5, so take T as 0.5. Note that the confidence intervals in Figure 5 overlap in many parts, indicating that different values of these parameters affect the algorithm to a limited extent.

Figure 6 shows a scheduling scheme obtained by MIACOSA in the instance m10n1h4. In this figure: M1 to M10 represent the machines; the numbers above each corresponding rectangle represent the processing start time and the processing completion time of the corresponding job; and the numbers in the rectangle indicate the job number and the MLAGV of the job; A1 to A3 represent MLAGVs; and the corresponding gray and white rectangles are the load driving time and no-load driving time. The numbers above each corresponding rectangle indicate the delivery start time and the delivery completion time, and the number in the rectangle indicates the target machine. Taking A1 as an example, its delivery route is M7 (for job 4) $\rightarrow$ M3 (for job 9 and job 10) $\twoheadrightarrow$ M2 (for job 12) $\rightarrow$ M8 (for job 15 and job 16) $\twoheadrightarrow$..... where $\rightarrow$ and $\twoheadrightarrow$ indicate the delivery after AGV delivers the job to the processing area and the buffer area, respectively. In addition, A1 needs no-load driving to the loading point before driving to M8.

4.3. Results and Analysis

4.3.1. Comparative Experiment of Heuristic Dispatching Rules

To demonstrate the superiority of MADR, different dispatching rules are independently run 50 times for each instance, including MADR, SDCT, STT [25], SIQ [26] and the random strategy RAND. The time of each run is less than 0.5 s, indicating there is no obvious time gap among them; therefore, this paper calculates the average objective function value of each instance solved by each rule.

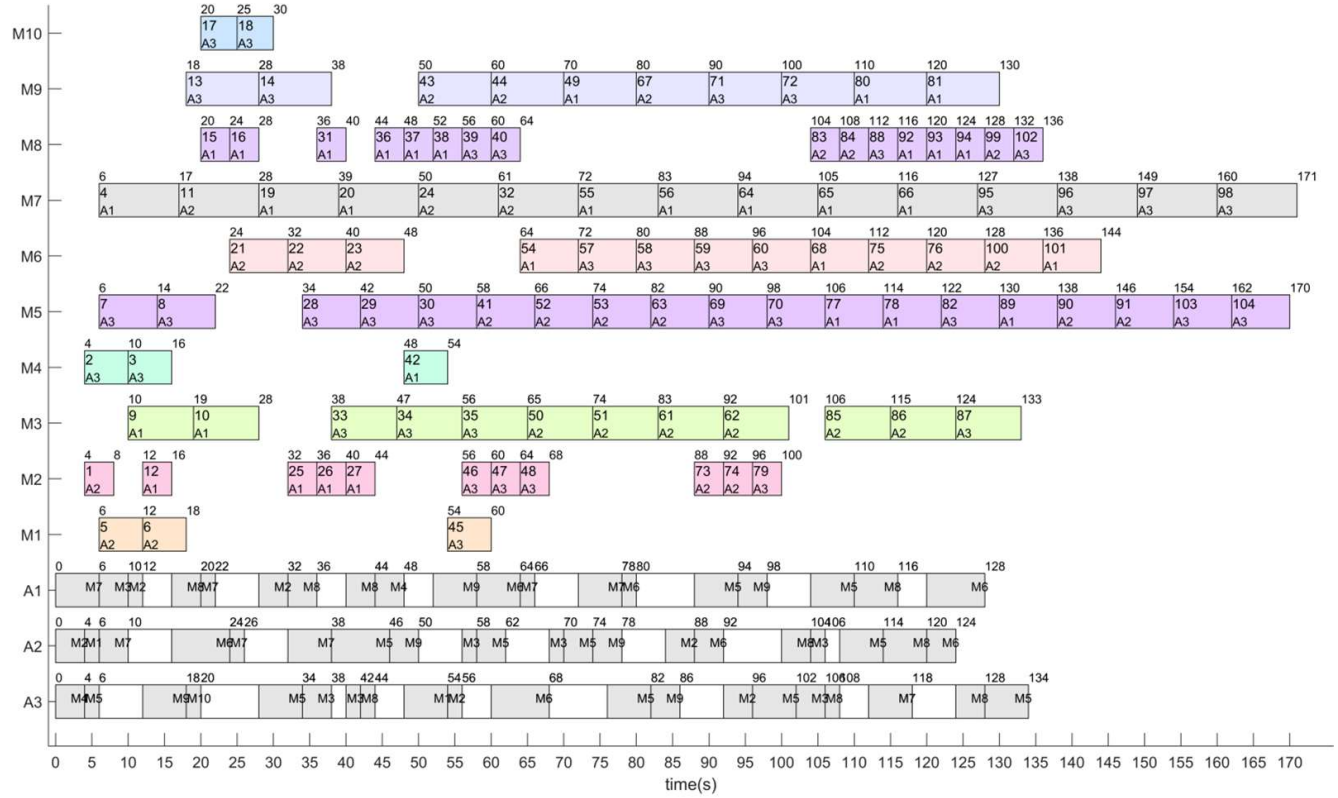

Figure 6. A scheduling scheme obtained by MIACOSA in the instance m10n1h4.

As shown in Table 3, MADR is optimal in most instances; the final average objective function value is optimal, indicating the superiority of MADR. Both SIQ and STT can outperform RAND, which means that the single attribute scheduling strategy that considers input queue or travel distance can optimize the objective function value. The SDCT designed in this paper is better than STT and SIQ, and the objective function value can be optimized by 36.6% and 20.91%, respectively, which shows that selecting the allocation scheme that can minimize the delivery completion time of MLAGV for each job can bring a more obvious optimization effect. Compared with SDCT, STT and SIQ, MADR can optimize 3.81%, 39.01% and 23.92% on the objective function value, respectively. Additionally, MADR is better than SDCT in the vast majority of instances, especially for the instance whose h is 4; MADR is always better than SDCT. It means that MADR, which combines delivery completion time and the input queue through dynamic weights, can have a higher solution efficiency, especially for the case where the buffer capacity is lager.

4.3.2. Experiment on the Effectiveness of Algorithm Improvement Strategies

We designed different MIACOSA variant algorithms for comparative experiments in order to demonstrate the effectiveness of the three algorithm improvement strategies. These improvement strategies are: the transfer probability calculation method based on MADR; the path branch and dynamic equilibrium mechanism; and the key job strategy. On the basis of MIACOSA, MIACOSA_ND replaces the path branch and dynamic equilibrium mechanism with the roulette method for building ant paths; MIACOSA_NS replaces the MADR with the distance-based visibility [33] for calculating the heuristic information; MIACOSA_NK replaces the key job strategy with a random strategy for selecting jobs in SA; and MIACOSA_NA performs the above three substitutions at the same time. Each algorithm is run 50 times for each instance, and the upper limit of the number of iterations is fixed to 120. If the instance can obtain the optimal solution in a short time (such as 2 min), we will calculate the average time $\overline{T}$ for finding the optimal solution; otherwise, we will calculate the average objective function value $\overline{C_{max}}$. The results are shown in Table 4, where

$\overline{T}^{\downarrow} = \left(\overline{T} - \overline{T}^{*}\right)/\overline{T}$ and $\overline{C_{max}}^{\downarrow} = \left(\overline{C_{max}} - \overline{C_{max}}^{*}\right)/\overline{C_{max}}$ of a variant algorithm reflect the optimization degree of MIACOSA compared with the variant algorithm in $\overline{T}$ and $\overline{C_{max}}$, respectively; the larger the $\overline{T}^{\downarrow}$ and $\overline{C_{max}}^{\downarrow}$ are, the more obvious the optimization degree is.

Table 3. The objective function value of heuristic rules (s).

Instances	MADR	SDCT	STT	SIQ	RAND
m10n1h2	190	192.64	295.32	242.52	368.32
m10n11h2	350	355.22	670.82	491.60	763.96
m10n21h2	498	492.92	906.94	645.32	978.60
m15n1h2	266	310.26	457.00	410.52	776.32
m15n11h2	573	574.78	966.50	751.62	1445.24
m15n21h2	804	741.20	1305.18	1022.32	1887.90
m20n1h2	357	379.70	504.82	500.72	977.68
m20n11h2	802	827.04	1317.70	1014.04	1965.58
m20n21h2	1127	1036.8	1971.50	1528.34	2923.74
m10n1h4	179	190.04	241.92	218.32	368.58
m10n11h4	334	351.12	549.34	420.14	793.16
m10n21h4	434	511.04	760.16	538.90	918.20
m15n1h4	264	306.28	384.10	379.08	779.48
m15n11h4	506	574.72	800.90	646.34	1322.98
m15n21h4	676	719.74	1103.60	866.28	1838.22
m20n1h4	346	365.72	417.98	477.70	1890.06
m20n11h4	682	816.80	1081.02	891.66	2023.90
m20n21h4	989	1002.3	1640.98	1279.28	3139.98
Average	520.94	541.57	854.21	684.71	1397.88

Table 4. The running results of different MIACOSA variant algorithms (s).

Instances	MIACOSA	MIACOSA_ND		MIACOSA_NS		MIACOSA_NK		MIACOSA_NA	
	$\overline{T}^{*}$	$\overline{T}$	$\overline{T}^{\downarrow}$	$\overline{T}$	$\overline{T}^{\downarrow}$	$\overline{T}$	$\overline{T}^{\downarrow}$	$\overline{T}$	$\overline{T}^{\downarrow}$
m10n1h2	0.82	1.15	28.33%	1.78	53.67%	1.95	57.79%	5.60	85.29%
m10n11h2	2.08	5.49	62.05%	1.83	−13.82%	3.96	47.37%	10.43	80.03%
m10n21h2	51.67	>120	-	87.83	41.17%	>120	-	>120	-
m10n1h4	0.45	1.30	65.06%	1.02	55.35%	0.89	49.03%	3.86	88.25%
m10n11h4	0.44	0.50	11.69%	0.52	15.74%	0.81	45.48%	6.05	92.72%
m10n21h4	8.55	14.78	42.14%	10.75	20.48%	38.53	77.81%	73.22	88.32%
	$\overline{C_{max}}^{*}$	$\overline{C_{max}}$	$\overline{C_{max}}^{\downarrow}$	$\overline{C_{max}}$	$\overline{C_{max}}^{\downarrow}$	$\overline{C_{max}}$	$\overline{C_{max}}^{\downarrow}$	$\overline{C_{max}}$	$\overline{C_{max}}^{\downarrow}$
m15n1h2	256.94	258.63	0.65%	254.23	−1.07%	258.87	0.75%	272.71	5.78%
m15n11h2	467.86	476.67	1.85%	468.08	0.05%	491.53	4.82%	508.38	7.97%
m15n21h2	636.37	638.05	0.26%	632.11	−0.67%	670.92	5.15%	680.42	6.47%
m20n1h2	330.83	341.95	3.25%	329.39	−0.44%	339.14	2.45%	347.65	4.84%
m20n11h2	658.45	674.2	2.34%	668.4	1.49%	694.81	5.23%	721.7	8.76%
m20n21h2	1013.24	1048.51	3.36%	1007.98	−0.52%	1069.25	5.24%	1073.28	5.59%
m15n1h4	242.76	247.42	1.88%	242.84	0.03%	252.68	3.93%	274.32	11.50%
m15n11h4	427.00	436.28	2.13%	444.16	3.86%	448.94	4.89%	486.20	12.18%
m15n21h4	586.40	597.93	1.93%	607.03	3.40%	610.70	3.98%	666.53	12.02%
m20n1h4	300.88	307.80	2.25%	307.50	2.15%	315.10	4.51%	334.88	10.15%
m20n11h4	611.48	626.13	2.34%	660.37	7.40%	634.47	3.62%	727.53	15.95%
m20n21h4	936.20	959.90	2.47%	990.40	5.47%	966.40	3.12%	1011.30	7.43%

The $\overline{T}^{\downarrow}$ and $\overline{C_{max}}^{\downarrow}$ of MIACOSA_NA, MIACOSA_NK, MIACOSA_ND and MIACOSA_NS are positive in most instances, indicating that the algorithm improvement strategies are effective when used independently or simultaneously. The $\overline{T}^{\downarrow}$ and $\overline{C_{max}}^{\downarrow}$

of MIACOSA_NA are the largest, which indicates that the performance of MIACOSA is the best when the three strategies are used simultaneously. The $\overline{T}^{\downarrow}$ and $\overline{C_{max}}^{\downarrow}$ of MIACOSA_NK are large, which shows that the key job strategy can increase the efficiency of SA, so as to significantly improve the efficiency of MIACOSA. The small $\overline{T}^{\downarrow}$ and $\overline{C_{max}}^{\downarrow}$ of MIACOSA_ND suggest that the path branch and dynamic equilibrium mechanism can somewhat increase the efficiency of MIACOSA, by increasing the efficiency of ACO. The $\overline{T}^{\downarrow}$ and $\overline{C_{max}}^{\downarrow}$ of MIACOSA_NS are the smallest, but the values are large when h is 4, indicating that the improvement of MIACOSA through the transfer probability calculation method based on MADR is not obvious, but it is suitable for the case of large input buffer capacity.

4.3.3. Comparative Experiment of Intelligent Algorithms

To demonstrate the superiority of MIACOSA, different heuristic algorithms are independently run 50 times for each instance, including MIACOSA, ACOSA [42], HACO [33], SA [38] and ACO [32]. We calculate the average objective function value $\overline{C_{max}}$ and the average running time $\overline{T}$ in each instance solved by each algorithm. In all comparison algorithms, the initial solution is set as the solution result of MADR, and the parameters are set to the value that can make the algorithms show the best performance, in which the number of ants is 5, the control parameters of the pheromone trail and heuristic information are 3, the pheromone evaporation rate is 0.5, the pheromone increment is 100, the number of cycles in the SA is 50, the initial temperature is 0.5, and the temperature change coefficient is 0.998. In order to more fully reflect the superiority of MIACOSA in various examples, we add instances where the value ranges of n_k is [6,25] and [16,35].

As shown in Table 5, for all instances, the objective function value and running time of MIACOSA are both smaller than those of other algorithms, suggesting the superiority of the MIACOSA. The ACO has the longest running time and the worst running results because of the large dimension of each state node in the ant path, which reduces the search efficiency. The SA has better running results than the ACO, and the running time is improved significantly, indicating that SA is more efficient than ACO in this problem. The HACO combines neighborhood search on the basis of ACO, so it has a significant improvement in running results and running time compared to ACO, and HACO is also better than SA. The ACOSA can be further improved compared to HCO, indicating that on the basis of the neighborhood search of HACO, the ACOSA can selectively accept inferior solutions through SA and improve the performance of the algorithm. Through the combination of ACO and SA, the MIACOSA can be significantly improved compared with ACO and SA, which can improve objective function values by 10.95% and save 88.28% in run time compared to ACO, and can improve objective function values by 7.65% and save 47.99% in run time compared to SA. Additionally, it can also be significantly improved compared to ACOSA and HACO by using the algorithm improvement strategies, which can improve objective function values by 7.06% and save 22.8% in run time, compared to ACOSA, and can improve objective function values by 7.47% and save 33.8% in run time compared to SA.

Figure 7 shows the optimization ratios of the average objective function value and average running time of MIACOSA, relative to other heuristic algorithms in the instances of various number of jobs. With the increase of job number, both optimization ratios have some upward trend, indicating that MIACOSA is still applicable when the number of jobs is large. When the number of jobs is 300 to 400 and 600 to 700, the optimization ratios of MIACOSA in objective function value and running time are high.

Table 5. The objective function value and running time of intelligent algorithms (s).

Instances	MIACOSA		ACOSA		HACO		SA		ACO	
	$\overline{C_{max}}$	$\overline{T}$	$\overline{C_{max}}$	$\overline{T}$	$\overline{C_{max}}$	$\overline{T}$	$\overline{C_{max}}$	$\overline{T}$	$\overline{C_{max}}$	$\overline{T}$
m10n1h2	171	0.82	175.00	1.60	175.80	1.92	172.00	1.14	177.12	3.16
m10n11h2	330	2.08	335.10	4.34	337.72	4.84	338.20	7.50	334,32	6.20
m10n21h2	378	51.67	405.90	69.79	409.80	70.72	409.61	67.29	429.73	77.45
m15n1h2	254.37	184.77	262.27	204.35	258.38	233.93	262.60	317.79	262.19	1265.10
m15n6h2	313.68	274.78	347.32	327.34	351.74	323.05	336.15	417.89	350.48	1982.40
m15n11h2	451.54	406.39	502.86	407.66	507.23	476.42	511.24	600.20	533.99	3112.20
m15n16h2	572.17	433.52	574.35	546.22	587.14	619.04	578.53	762.45	601.23	3840.40
m15n21h2	614.02	447.74	692.15	566.94	700.26	693.37	680.45	827.05	719.32	3060.10
m20n1h2	298.15	191.04	335.05	232.51	331.40	301.68	345.21	355.83	349.42	691.06
m20n6h2	428.39	272.29	476.34	384.07	471.59	470.96	476.86	550.78	489.78	2790.90
m20n11h2	614.72	469.87	719.09	540.77	710.22	470.70	728.94	763.43	744.80	2635.30
m20n16h2	782.37	645.30	858.64	716.65	859.65	690.04	865.34	1042.40	871.60	4085.90
m20n21h2	930.93	937.40	1097.1	1028.3	1093.80	922.64	1090.63	1330.30	1100.2	7119.04
m10n1h4	171	0.45	176.60	0.90	177.48	1.03	173.60	1.59	179	2.22
m10n11h4	330	0.44	330.60	1.41	330.12	1.18	330.85	1.74	334	3.74
m10n21h4	378	8.55	391.90	12.29	395.60	16.59	395.60	10.79	434	20.34
m15n1h4	236.84	153.23	254.40	213.92	254.48	210.02	252.60	288.27	262.12	1120.30
m15n6h4	291.80	163.05	312.50	294.23	313.12	235.33	308.85	402.73	313.20	1753.40
m15n11h4	422.00	266.59	443.38	440.00	460.60	478.24	476.21	595.17	478.75	2491.90
m15n16h4	505.70	313.72	514.76	485.42	533.39	604.21	529.55	769.78	530.42	3298.80
m15n21h4	581.40	324.42	600.45	497.91	618.10	686.13	628.13	852.90	635.50	3621.90
m20n1h4	295.88	127.66	334.30	190.53	329.64	293.75	319.56	358.89	330.84	1643.20
m20n6h4	420.00	217.35	440.20	308.53	436.40	469.90	440.28	560.50	448.60	2638.50
m20n11h4	606.48	287.63	651.40	443.76	645.71	644.21	646.30	763.19	676.57	3724.60
m20n16h4	749.40	493.84	780.12	673.29	778.90	936.28	790.70	1066.90	796.35	4085.90
m20n21h4	925.20	604.19	956.90	835.36	958.34	1138.4	963.32	1278.30	965.70	7023.00
Average	463.58	279.95	498.80	362.62	501.02	422.87	501.97	538.26	520.60	2388.35

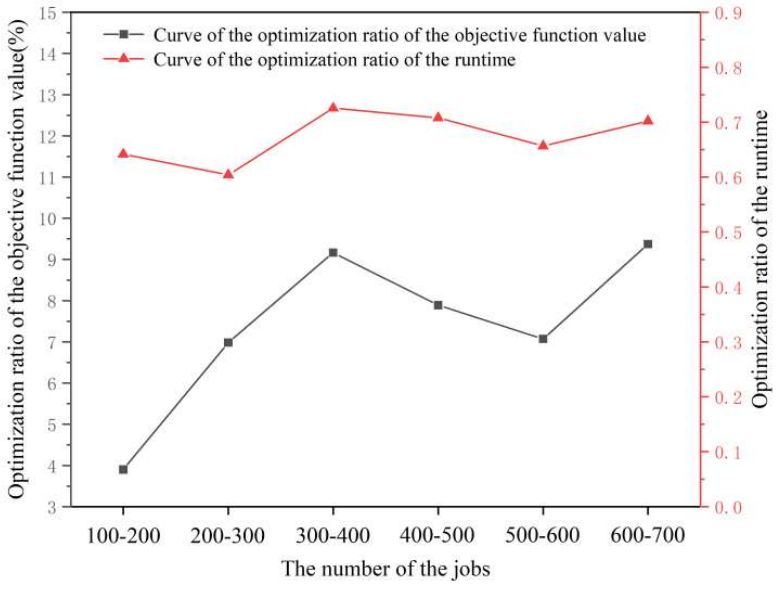

Figure 7. Optimization ratios of the average objective function value and average running time of MIACOSA, relative to other heuristic algorithms in the instances of various number of jobs.

5. Conclusions

We studied a MLAGV workshop scheduling problem with limited buffer capacity, aiming to minimize the maximum completion time and propose MIACOSA. First, design MADR, which combines delivery completion time and the input queue through dynamic

weights that are determined by the information about the system. We use MADR to construct the initial solution. Then, with ACOSA as the basic framework, we propose a method for calculating the transfer probability based on MADR, which obtains heuristic information through MADR; we propose a path branch mechanism and dynamic equilibrium mechanism, aiming to efficiently construct an ant path and dynamically adjust ant paths distribution. Further, we propose the key job strategy and design a 2-opt neighborhood search method based on the key jobs. Data experiments demonstrate MADR is superior over other heuristic dispatching rules. The algorithm improvement strategies proposed are effective when used simultaneously or separately. The MIACOSA is superior over other heuristic algorithms and adapts to all kinds of instances. Meanwhile, there are more complex workshop scheduling problems, such as the MLAGV workshop scheduling problem, with limited buffer capacity and multiprocesses, which can be used as a further research direction.

Author Contributions: Writing–original draft, Z.W.; supervision, Y.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China (No. 62273204), Natural Science Foundation of Shandong Province (No. ZR2020MF085), and the Science and Technology Innovation Committee of Shenzhen (No. JCYJ20220530141210023).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Readers can obtain the raw datasets used in this paper through the data sources described in Section 4, or by contacting the authors.

Acknowledgments: We gratefully acknowledge the anonymous reviewers for their insightful comments and suggestions for this paper.

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

References

1. Ho, Y.C.; Liu, H.C.; Yih, Y. A multiple-attribute method for concurrently solving the pickup-dispatching problem and the load-selection problem of multiple-load AGVs. *J. Manuf. Syst.* **2012**, *31*, 288–300. [CrossRef]
2. Wu, X.; Zhai, J.J.; Lou, P.H.; Hu, Y.; Xiao, H.N. Deadlock-free task scheduling with task traveling time for a multi-load AGV system. *Chin. J. Mech. Eng.* **2021**, *32*, 2840–2849. (In Chinese) [CrossRef]
3. Fu, J.L.; Zhang, H.Z.; Zhang, J.; Jiang, L.K. Review on AGV scheduling optimization. *J. Syst. Simul.* **2020**, *32*, 1664–1675. (In Chinese) [CrossRef]
4. Li, Z.F.; Liu, Y.Y. Job shop scheduling considering multiple AGVs with charging. *Comput. Integr. Manuf. Syst.* **2021**, *27*, 2872–2879. (In Chinese) [CrossRef]
5. Dang, Q.V.; Thanh, N.C.; Hana, R. Scheduling of mobile robots for transportation and manufacturing tasks. *J. Heuristics* **2019**, *25*, 175–213. [CrossRef]
6. Li, G.; Zeng, B.; Liao, W.; Li, X.; Gao, L. A new AGV scheduling algorithm based on harmony search for material transfer in a real-world manufacturing system. *Adv. Mech. Eng.* **2018**, *10*, 3. [CrossRef]
7. Zhang, P.; Song, S.; Niu, S.; Zhang, R. A Hybrid Artificial Immune-Simulated Annealing Algorithm for Multiroute Job Shop Scheduling Problem With Continuous Limited Output Buffers. *IEEE Trans. Cybern.* **2021**, *52*, 12112–12125. [CrossRef]
8. Dang, V.Q.; Nielsen, I.E.; Steger-Jensen, K.; Madsen, O. Scheduling a single mobile robot for part-feeding tasks of production lines. *J. Intell. Manuf.* **2014**, *25*, 1271–1287. [CrossRef]
9. Zhou, E.D. Single AGV Optimal Path Planning and Its System Development. Master's Thesis, Henan University of Technology, Zhenzhou, China, 2017. (In Chinese).
10. José, A.V.; Brian, Q. Optimal location of dwell points in a single loop AGV system with time restrictions on vehicle availability. *Eur. J. Oper. Res.* **2007**, *192*, 93–104. [CrossRef]
11. Mustafa, S.; Özdemirel, N.E.; Sinan, K. A self-organizing neural network approach for the single AGV routing problem. *Eur. J. Oper. Res.* **2000**, *121*, 124–137. [CrossRef]
12. Du, L.Z.; Ke, S.; Wang, Z.; Tao, J.; Yu, L. Research on multi-load AGV path planning of weaving workshop based on time priority. *Math. Biosci. Eng.* **2019**, *16*, 2277–2292. [CrossRef] [PubMed]
13. Zou, W.Q.; Pan, Q.K.; Tasgetiren, M.F. An effective iterated greedy algorithm for solving a multi-compartment AGV scheduling problem in a matrix manufacturing workshop. *Appl. Soft Comput.* **2020**, *99*, 3. [CrossRef]

14. Wang, Z.; Wang, C.X.; Wang, H.; Du, L.Z.; Yu, L. On multi-load AGV green logistics scheduling in knitting workshop. *Math. Biosci. Eng.* **2020**, *43*, 1. (In Chinese) [CrossRef]
15. Li, G.M. Research on Multi-Load AGVs Scheduling Methods for Shopfloor Logistics. Ph.D. Thesis, Guangdong University of Technology, Guangzhou, China, 2020. (In Chinese). [CrossRef]
16. Zhang, L.; Hu, Y.; Guan, Y. Research on hybrid-load AGV dispatching problem for mixed-model automobile assembly line. *Procedia CIRP* **2019**, *81*, 1059–1064. [CrossRef]
17. Yu, N.N.; Li, T.K.; Mao, N.; Wang, B.L.; Yuan, S.P. Multi-AGVs scheduling and path planning algorithm in automated sorting warehouse. *Comput. Integr. Manuf. Syst.* **2020**, *26*, 171–179. (In Chinese) [CrossRef]
18. Zhou, Q. Single AGV Scheduling for Limited Buffer of Flexible Manufacturing System. Master's Thesis, Guangdong University of Technology, Guangzhou, China, 2014. (In Chinese). [CrossRef]
19. Chen, G.Z.; Chen, Q.X.; Mao, N.; Yu, A.L. Simulation optimization of AGV and buffer capacity in intelligent manufacturing system. *J. Ind. Eng. Int.* **2016**, *19*, 77–84. (In Chinese) [CrossRef]
20. Blazewicz, J. Scheduling tasks and vehicles in a flexible manufacturing system. *Int. J. Flex. Manuf. Sys.* **1991**, *4*, 5–16. [CrossRef]
21. Zhang, H.Y.; Kang, W.C.; Ning, M.; Chen, Q.G. Buffer Optimization Configuration Method for Customized Manufacturing System. *J. Ind. Eng. Int.* **2014**, *17*, 85. [CrossRef]
22. Xiong, Y.; Mao, J.L. Research on scheduling of multi-load AGV system with limited buffer capacity. *J. Electron. Sci. Technol.* **2018**, *31*, 72–77. (In Chinese) [CrossRef]
23. Mallala, B.; Papana, V.P.; Sangu, R.; Palle, K.; Chinthalacheruvu, V.K.R. Multi-Objective Optimal Power Flow Solution Using a Non-Dominated Sorting Hybrid Fruit Fly-Based Artificial Bee Colony. *Energies* **2022**, *15*, 4063. [CrossRef]
24. Balasubbareddy, M.; Dwivedi, D.; Venkata, G.; Murthy, K.; Kumar, K. Optimal Power Flow Solution with Current Injection Model of Generalized Interline Power Flow Controller using Ameliorated Ant Lion Optimization. *IJECE* **2023**, *13*, 1060–1077. [CrossRef]
25. Bilge, U.; Esenduran, G.; Varol, N.; Oeztuerk, Z.; Aydin, B.; Alp, A. Multi-attribute responsive dispatching strategies for automated guided vehicles. *Int. J. Prod. Econ.* **2006**, *100*, 65–75. [CrossRef]
26. Ho, Y.-C.; Chien, S.H. A simulation study on the performance of task-determination rules and delivery-dispatching rules for multiple-load AGVs. *Int. J. Prod. Res.* **2006**, *44*, 4193–4222. [CrossRef]
27. Klei, C.M.; Kim, J. AGV dispatching. *Int. J. Prod. Res.* **1996**, *34*, 95–110. [CrossRef]
28. Guan, J.; Qiu, Z.; Zhou, Y.; Yu, J.; Gui, W. Scheduling of multi-load automated guided vehicles for material distribution based on multi-stage auction algorithm. In Proceedings of the 13th WCICA, Changsha, China, 4–8 July 2018. Available online: https://ieeexplore.ieee.org/document/8630350 (accessed on 31 January 2019).
29. Jeong, B.H.; Randhawa, S.U. A multi-attribute dispatching rule for automated guided vehicle systems. *Int. J. Prod. Res.* **2001**, *39*, 2817–2832. [CrossRef]
30. Naso, D.; Turchiano, B. Multicriteria meta-heuristics for AGV dispatching control based on computational intelligence. *IEEE Trans. Syst. Man Cybern. Syst.* **2005**, *35*, 208–226. [CrossRef]
31. Xiao, Y.Q.; Jiao, J.Q.; Qiao, D.P.; Du, J.H.; Zhou, K. Summary of the basic principles and application of ant colony algorithm. *J. Light Ind.* **2018**, *34*, 69–72. (In Chinese). Available online: https://kns.cnki.net/kcms/detail/45.1385.TS.20180302.1710.064.html (accessed on 2 March 2018).
32. Huang, M.; Huang, Y. Application of improved ant colony algorithm in manufacturing shop scheduling. *China Comput. Commun.* **2021**, *33*, 144–146. (In Chinese) [CrossRef]
33. Heinonen, J.; Pettersson, F. Hybrid ant colony optimization and visibility studies applied to a job-shop scheduling problem. *Appl. Math. Comput.* **2007**, *187*, 989–998. [CrossRef]
34. Wang, Y.H.; Wang, W.X.; Yu, H.X.; Chen, L. Dynamic balance adaptive colony algorithm solving job-shop scheduling. *Comput. Integr. Manuf. Syst.* **2013**, *19*, 2521–2527. (In Chinese) [CrossRef]
35. Mei, Q.; Dong, B.L. Mult-AGV task assignment based on hybrid ant colony genetic algorithm. *Logist. Eng. Manag.* **2022**, *44*, 1–5+9. (In Chinese) [CrossRef]
36. Liu, R.; Yang, C.W.; Gao, C.S.; Li, X.D. Research on path planning of AGV based on dual population ant colony algorithm. *Comput. Meas. Control* **2023**. (In Chinese). Available online: http://kns.cnki.net/kcms/detail/11.4762.tp.20221229.1739.004.html (accessed on 2 January 2023).
37. Hu, J.C.; Wu, Y.Y.; Wang, Y.Y.; Wu, Y.H. Worker job shop scheduling algorithm considering learning effect. *J. Control. Decis.* **2022**, *37*, 37–46. (In Chinese) [CrossRef]
38. Hu, J.C.; Wu, Y.H.; Wu, Y.Y.; Yang, D. Batch scheduling with learning effect on single-machine to minimize the total tardiness. *J. Control Decis.* **2019**, *34*, 2708–2712. (In Chinese) [CrossRef]
39. Dai, M.; Zhang, Y.W.; Zeng, L.; Zhu, Z.D.; Zhang, F. Enhanced estimation of distribution algorithm for solving job-shop scheduling problem. *J. Yangzhou Univ.* **2020**, *23*, 26–30. (In Chinese) [CrossRef]
40. Niu, Q.Y.; Li, B. Omnidirectional AGV path planning based on simulated annealing genetic algorithm. *Comput. Integr. Manuf. Syst.* **2023**. (In Chinese). Available online: http://kns.cnki.net/kcms/detail/11.5946.TP.20220517.1105.014.html (accessed on 17 May 2022).
41. Niu, H.; Jin, S.; Luo, L.; Liu, S. AGV path planning algorithm and optimization based on topological grid modeling. *Comput. Eng. Des.* **2022**, *43*, 101–109. (In Chinese) [CrossRef]

42. Wang, Y.; Wang, L.; Chen, G.; Cai, Z.; Xing, L. An improved ant colony optimization algorithm to the periodic vehicle routing problem with time window and service choice. *Swarm Evol. Comput.* **2020**, *55*, 100675. [CrossRef]
43. Zhang, X.J.; Gao, H.M. Application of ant colony optimization based on simulated annealing to job-shop problem. *Comput. Appl. Softw.* **2008**, *25*, 77–79. (In Chinese) [CrossRef]
44. Liu, F.J.; Xue, R.Z. Hybrid particle swarm optimization for solving job-shop scheduling problem. *J. Sci. Teach. Coll. Univ.* **2022**, *42*, 38–43. (In Chinese) [CrossRef]
45. Xu, B.Z.; Ji, J.; Fei, X.L. Problems and solutions of flexible job-shop scheduling with operation lot-spliting. *Chin. J. Mech. Eng.* **2016**, *27*, 3221–3228. (In Chinese) [CrossRef]
46. Ruiz, R.; Stützle, T. A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. *Eur. J. Oper. Res.* **2007**, *177*, 2033–2049. [CrossRef]
47. Ruiz, R.; Pan, Q.K.; Naderi, B. Iterated greedy methods for the distributed permutation flowshop scheduling problem. *Omega* **2019**, *83*, 213–222. [CrossRef]