Mathematical Biosciences and Engineering

MBE, 20(4): 7519–7547.
DOI: 10.3934/mbe.2023325
Received: 11 November 2022
Revised: 24 December 2022
Accepted: 05 February 2023
Published: 16 February 2023

http://www.aimspress.com/journal/MBE

Research article

An improved ant colony optimization for solving the flexible job shop scheduling problem with multiple time constraints

Shaofeng Yan[1], Guohui Zhang[1,*], Jinghe Sun[2] and Wenqiang Zhang[3]

[1] College of Management Engineering, Zhengzhou University of Aeronautics, Zhengzhou 450046, China
[2] College of Economics and Management, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China
[3] College of Information Science and Engineering, Henan University of Technology, Zhengzhou 450046, China

* **Correspondence:** Email: zgh_hust@qq.com.

Abstract: The flexible job shop scheduling problem is important in many research fields such as production management and combinatorial optimization, and it contains sub-problems of machine assignment and operation sequencing. In this paper, we study a many-objective FJSP (MaOFJSP) with multiple time constraints on setup time, transportation time and delivery time, with the objective of minimizing the maximum completion time, the total workload, the workload of critical machine and penalties of earliness/tardiness. Based on the given problem, an improved ant colony optimization is proposed to solve the problem. A distributed coding approach is proposed by the problem features. Three initialization methods are proposed to improve the quality and diversity of the initial solutions. The front end of the algorithm is designed to iteratively update the machine assignment to search for different neighborhoods. Then the improved ant colony optimization is used for local search of the neighborhood. For the searched scheduling set the entropy weight method and non-dominated sorting are used for filtering. Then mutation and closeness operations are proposed to improve the diversity of the solutions. The algorithm was evaluated through experiments based on 28 benchmark instances. The experimental results show that the algorithm can effectively solve the MaOFJSP problem.

Keywords: many-objective flexible job-shop scheduling problem; improved ant colony optimization; setup time; transportation time; delivery time

1. Introduction

The flexible job shop scheduling problem was originally developed by Brucker and Schlie [1], and since then the problem has received a lot of attention. In traditional shop scheduling, the machines for all operations of a job are unique, and only the sequence of operations on each machine needs to be determined. And flexible job shop scheduling eliminates the limitation of resource uniqueness [2], and each operation of a job can be processed on multiple machines. Scheduling requires not only sequencing the operation but also making decisions about the machines that will process the operation.

FJSP started with the main research objective of optimizing the maximum completion time. Later on, as the research continued, more objectives came into focus. There are optimization objectives regarding machine load that was studied in earlier years [3,4]. Different jobs in the flexible job shop are processed on the same machine requiring job setup that consumes a lot of time. Many scholars have considered the loading of workers and the change of tools, as well as the setup time that exists for machining on the same machine [5–8]. In addition, in flexible job shop scheduling, the same job often needs to be processed on different machines. This will greatly reduce the feasibility of the scheduling solution if the actual execution does not deliver the job or material to the corresponding station on time. Some scholars have added constraints on transport routes and transportation time considering the transport processes that exist in the middle of job processing in different machines [9–12]. In recent years, some scholars have also considered both transportation time and setup time [7,13]. The demand for punctuality in manufacturing orders is very high, and companies need to find a suitable production completion point according to the delivery date. A heterogeneous DAFJSP with the total cost and the earliness/tardiness as the objectives is studied [14]. The above literature has studied FJSP under different objectives, however, these papers are relatively scarce for considering multiple time constraints of setup time, transportation time, and delivery time simultaneously. [2] studied the flexible job shop scheduling problem with multiple time constraints. The paper treats transportation time and setup time as independent time factors, with the objectives of minimizing maximum completion time, minimizing total setup time, and minimizing total transportation time. It does not take into account the load capacity of individual machine and the delivery time of orders, which are important references for actual production. In [13], transportation and setup time are considered, while minimizing the maximum completion time, total delay, and total energy consumption. However, it ignores the limitations of the workload of each machine in a realistic shop.

As the demand for customized manufacturing continues to grow, multi-variety and low volume have become a very important aspect of modern manufacturing companies [15]. The problem studied in this paper is abstracted from a customized furniture production shop for panels. In this customized production shop, the diversity of products produced increases the setup time and transportation time. In this case, the scheduling shop that considers only the processing time schedule will have a large error with the real production completion time. Another requirement that cannot be ignored in custom processing orders is the delivery time. Different orders have different delivery time. Producing the product too early takes up storage space, while producing it too late violates the order requirements, so the right production completion time is important for customized production. Therefore, based on this customized production shop this paper considers processing time, setup time, transportation time, and order delivery time as independent time factors for FJSP.

The study of the multi-objective flexible job shop scheduling problem (MOFJSP) is of great engineering significance, but the difficulty of this type of problem lies in dealing with the existence of

multi-objective conflicts. The individual solutions found in this problem are not directly comparable and there are non-dominated solutions. For any two solutions S_1, and S_2. if all solutions of S_1 are better than S_2, it means S_1 dominates S_2. If there exists a solution within S_1 that is not dominated by other solutions, it means that S_1 is a non-dominated solution. The set of these non-dominated solutions is the Pareto Front. Zhang et al. [4] use a weighted summation method to construct the fitness function in solving MOFJSP, but the weights of each objective are difficult to determine. Non-dominated sorting is one of the more widely used sorting methods in recent years [16]. Therefore, this paper addresses these problems by proposing a combination of the entropy weight method and non-dominated sorting to solve the multi-objective problem.

It is well known that the traditional FJSP has proven to be an NP-hard problem. The FJSP, which considers multiple time constraints, is more complex and therefore requires an efficient algorithm to solve the problem. At present, many scholars in this field have used different optimization algorithms to solve FJSP. For example, genetic algorithms (GA) [17–19], ant colony optimization (ACO) [20], particle swarm optimization (PSO) [21], simulated annealing (SA) [22], imperialist competitive algorithm (ICA) [23], African buffalo optimization (ABO) [24,25], whale optimization algorithm (WOA) [26,27]. Ant colony optimization is a swarm intelligence algorithm. It is mainly used to solve traveling salesman problem (TSP) problem, assignment scheduling problem, and job-shop scheduling problem, and achieve better experimental results. Its search process uses distributed computing, with multiple individuals performing parallel computation at the same time, greatly improving the computational power and operational efficiency of the algorithm. In this paper, we choose ant colony optimization to solve the problem by combining the problem characteristics and algorithm features. Ant colony optimization has the characteristic of positive feedback. If the algorithm starts with a suboptimal solution, the positive feedback will make the suboptimal solution dominate quickly and make the algorithm fall into the local optimum, and it is difficult to jump out of the local optimum. Therefore, the use of ant colony optimization is more demanding for the initial solution.

Based on the characteristics of the ant colony optimization, this paper designs a distributed coding method and three ways to generate the initial solution to improve the quality of the initial solution. The search is performed using an improved ant colony optimization for operation sequencing. The scheduling set is then divided into two populations by combining the entropy weight method and the non-dominated sorting. Finally, mutation and closeness operations are proposed for the machine assignment of the population to improve the quality and diversity of the population, respectively.

In summary, there are relatively few studies that consider multiple time constraints on setup time, transportation time, and delivery time simultaneously. Therefore, this paper proposes an improved multi-objective ant colony optimization to solve the flexible job shop scheduling problem with multiple time constraints. The main contributions of this paper are as follows.

1) A flexible job shop that simultaneously considers multiple time constraints such as transportation time, setup time and delivery time, and takes into account machine load capacity, is studied from the perspective of actual production needs. Better guidance is provided for actual production.

2) To solve this problem effectively, an improved ant colony algorithm is proposed. The distributed coding approach designed in this paper expands the search range of the solution. Then the proposed three initialization strategies can generate high-quality initial solutions. In addition, the improved ant colony algorithm enhances the search accuracy of the algorithm. Finally, the population diversity is further improved by non-dominated selection combined with mutation and

closeness operations.

3) The feasibility of the proposed algorithm in solving flexible job shop scheduling under multiple time constraints is verified through extensive experiments.

The rest of this paper is as follows. Section 2 describes the flexible job shop scheduling problem with multiple time constraints in detail and provide an instance of a small-scale for understanding. Section 3 details the overall process of the algorithm in this paper. In Section 4, we compare the algorithm proposed in this paper with other algorithms on 28 benchmark instances. Section 5 presents our conclusions and plans for future research.

Table 1. Notations and indices for various parameters of the IACO.

Indices	Definition
i	Number of jobs, i = 1, 2, 3, ..., n
j	Operation number, j = 1, 2, 3,..., h_i
k,g	Machine number, k = 1, 2, 3, ..., m
Parameters	
O_{ij}	Operation j of Job J_i
$M_{O_{ij}}$	An optional set of machines for O_{ij}
C_i	Completion time of Job i
$PJ_{O_{ij}}$	O_{ij} for the immediate pre-operation
$SJ_{O_{ij}}$	O_{ij} for the immediate post-operation
$PT_{O_{ij,k}}$	O_{ij} processing time on machine k
$TT_{k,g}$	Transportation time from machine k to machine g
$ST_{O_{ij,k}}$	O_{ij} setup time on machine k
DS_i	Earliest delivery date
DE_i	Latest delivery date
α_s	Penalty for earliness
α_e	Penalty for tardiness
Variables	
$\partial_{O_{ij,k}}$	variable that is equal to 1 if operation O_{ij} is processed on machine k and 0
$\delta_{ij,k}$	variable that is equal to 1 if operation O_{ij} and $PJ_{O_{ij}}$ is processed on machine k and

2. Problem formulation

MaOFJSP is described as follows. There are *n* operations and *m* machines in a shop. Each job consists of h_i operations that must be processed in a specified sequence. Each operation can only be processed by one machine, which is any of the optional sets of machines. The processing time for each operation depends on which machine is processing the operation. When an operation of a job completes, it is moved to the next operation. There may be transportation time and setup time between the completion time of one operation and the start time of another consecutive operation. If two adjacent

operations of the same job are processed on different machines, there is a transfer process, which generates additional transportation time. Setup time exists on the machine when two consecutive operations on the same machine are different types of jobs. Note that each machine needs to be set up for a time before the first job can be processed. Transportation time and setup time are zero when two adjacent operations of the same job are processed on the same machine. The problem is to assign operations to machines and find the sequence of operations on all machines to create a feasible plan that minimizes makespan, total workload (TW), workload of critical machine (WCM), and penalties of earliness/tardiness (PET). The following assumptions exist in this paper for MaOFJSP.

1) Each machine can only process one operation at a time, and once the machine starts processing this operation it cannot be stopped.

2) At the moment 0 all machines are available for work and at the moment 0 all jobs are released at the same time.

3) The first job transportation time of the operation is 0, but its setup time is not 0.

4) Each operation requires only one machine for processing.

5) Each job has a fixed sequence of operations to be processed, and after each operation is completed the next operation will be processed.

6) When two adjacent operations of a job are processed on different machines, the transportation time is determined by the two machines. The transportation time between the machines is known.

7) The setup time before processing is different for different operations on different machines. The setup time before processing on different machines is also known for each operation.

For convenience of description, we define the notations in Table 1.

The four optimization objectives of this paper are as follows:

1) Minimization of makespan (f_1):

$$\min f_1 = \max_{i=1}^{n} C_i \tag{1}$$

2) Minimization of TW (f_2):

$$\min f_2 = \sum_{i=1}^{n}\sum_{j=1}^{h_i}\sum_{k\in M_{O_{ij}}}\sum_{g\in M_{SJ_{O_{ij}}}}\left(PT_{O_{ij},k}\partial_{O_{ij},k} + TT_{k,g}\partial_{O_{ij},k}\partial_{SJ_{O_{ij},\,g}} + ST_{O_{ij},k}\partial_{O_{ij},k}\right) \tag{2}$$

3) Minimization of WCM (f_3):

$$\min f_3 = \max_{k=1}^{m}\sum_{i=1}^{n}\sum_{j=1}^{h_i}\sum_{g\in M_{PJ_{O_{ij}}}}\left(PT_{O_{ij},k}\partial_{O_{ij},k} + TT_{k,g}x_{PJ_{O_{i,j}},g}\partial_{O_{ij},k} + ST_{O_{ij},k}\partial_{O_{ij},k}\right) \tag{3}$$

4) Minimization of PET (f_4):

$$\min f_4 = \sum_{i=1}^{n}\begin{cases}\alpha_s(DS_i - C_i), & C_i < DS_i \\ 0, & DS_i \le C_i \le DE_i \\ \alpha_e(C_i - DE_i), & C_i > DE_i\end{cases} \tag{4}$$

To facilitate understanding, we now provide an instance of a small-scale flexible job shop scheduling problem. Table 2 shows an instance of a flexible job shop scheduling problem. The instance

consists of 3 jobs and 4 machines. Table 2 indicates the processing time and setup time required to handle different operations for different jobs on each machine. Table 3 indicates the transportation time between the different machines.

Table 2. Processing and setup times for the 3 × 4 instance.

Job	$O_{i,j}$	Processing time/Setup time			
		M_1	M_2	M_3	M_4
J_1	$O_{1,1}$	2/1	5/3	4/1	1/1
	$O_{1,2}$	5/3	4/1	5/2	7/6
	$O_{1,3}$	4/1	5/4	5/4	4/2
J_2	$O_{2,1}$	2/2	5/4	4/1	7/5
	$O_{2,2}$	5/1	6/6	9/7	8/4
	$O_{2,3}$	4/2	5/3	4/3	54/43
J_3	$O_{3,1}$	9/4	8/7	6/4	7/3
	$O_{3,2}$	6/6	1/1	2/2	5/3
	$O_{3,3}$	2/1	5/5	4/1	2/1
	$O_{3,4}$	4/4	5/1	2/2	1/1

Table 3. Transportation times for the 3 × 4 instance.

Machine	Transportation time			
	M_1	M_2	M_3	M_4
M_1	0	4	2	1
M_2	1	0	5	1
M_3	1	3	0	2
M_4	3	1	3	0

Figure 1(a) shows the scheduling Gantt chart without setup time and transportation time. In Figure 1(a) 101 is the first operation of the job. Operation 101 corresponds to 4 on the y-axis and 1 on the x-axis, indicating that the time required for operation 101 to be processed on machine 4 is 1-time unit. Figure 1(b) shows the scheduling Gantt chart considering setup time and transportation time. The brown box in Figure 1(b) represents the setup time and the light blue box represents the transportation time. The light blue 301 box indicates that the transportation time required for job 3 to be transported from machine 4 to machine 2 is the 1-time unit. As can be seen in Figure 1(b), the scheduling Gantt with setup time and transportation time has a maximum completion time of 5 more than the scheduling Gantt without setup time and transportation time. In the actual production process, the extra 5 make sense. The feasibility of the resulting plan is low if setup time and transportation time are not taken into account.

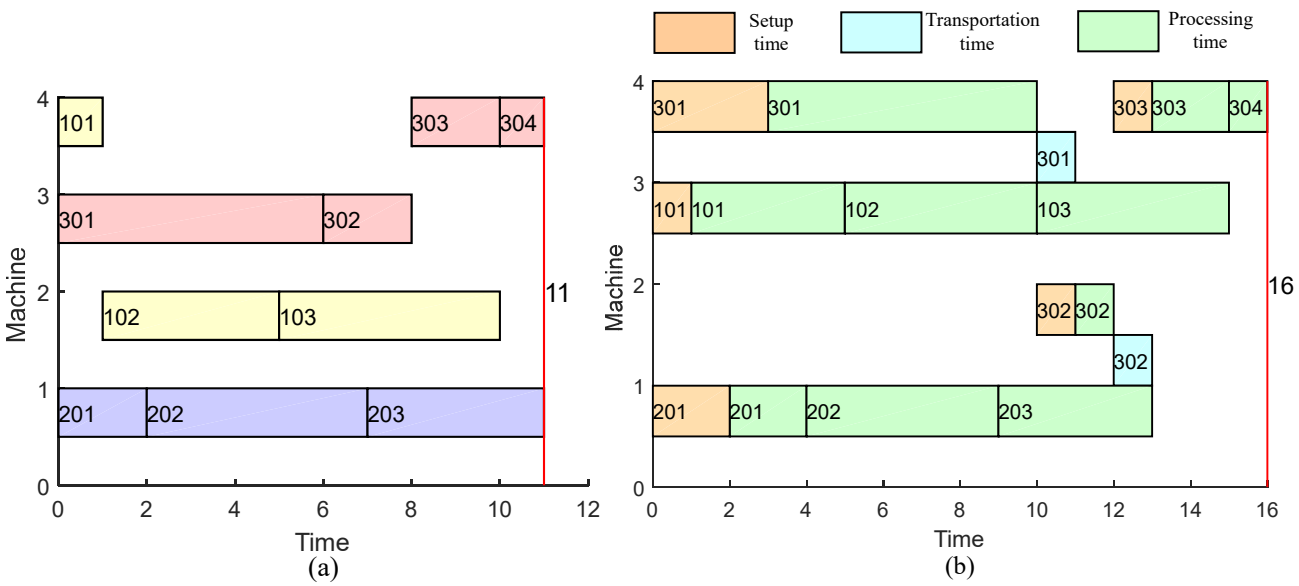

Figure 1. (a) The Gantt chart without transportation time and setup time. (b) The Gantt chart with transportation time and setup time.

3. The proposed IACO for MaOFJSP

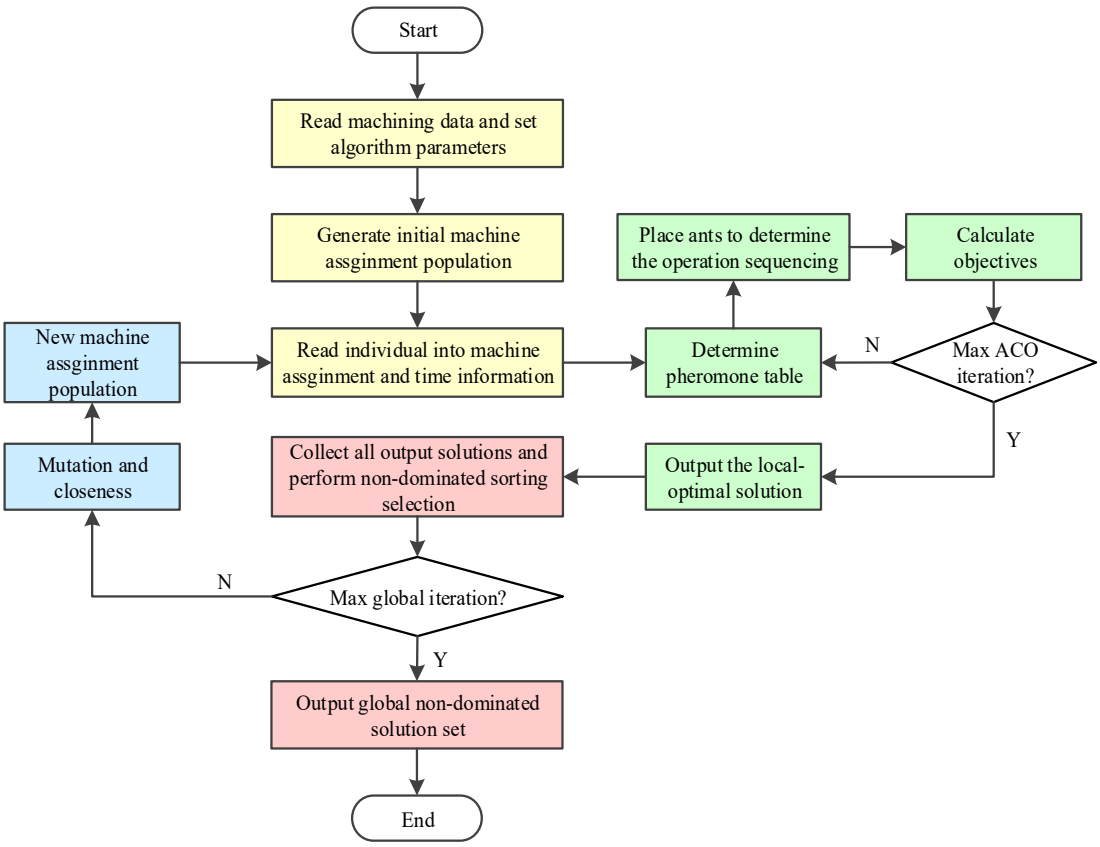

Figure 2. The framework of IACO.

In this paper, we propose an improved ant colony optimization to solve MaOFJSP. The framework of IACO is shown in Figure 2. The machine assignment is first determined and the solution space is decomposed into multiple mutually independent regions. Then an improved ant colony optimization is used to search for the optimal scheduling solution for each optimization objective in this region. In addition, a method combining entropy power method and non-dominated ranking method is presented for finding non-dominated solutions. Finally, mutation and closeness operations are proposed for the machine assignment part to improve the search accuracy. The algorithm is described in detail in Sections 3.1–3.6.

3.1. Chromosome coding

L Coding is the representation of a feasible solution in simple chromosomal form, converting the feasible solution into an array form that can be processed by a computer. An effective coding approach can better express the relationship between individuals and feasible solutions. FJSP contains two subproblems: machine assignment and operation sequencing. The two-stage integer coding method is one of the most common coding methods for FJSP. A sequence of two ends of length $L=\sum O_{ij}$ is used to represent machine assignment and operation sequencing, respectively. However, this coding is not sufficient for the search of individual solution spaces and can lead to an inefficient start of the search. In this paper, we propose a distributed coding approach, which is different from the traditional two-stage integer coding of the search framework. This coding approach first determines the machine assignment and decomposes the solution space into multiple mutually independent regions. This is then combined with an ant colony optimization to sort the operations in order to adequately search the solution space allocated to this machine. The final set of non-dominated solutions is obtained by continuously adjusting the machine assignment and operation sequencing.

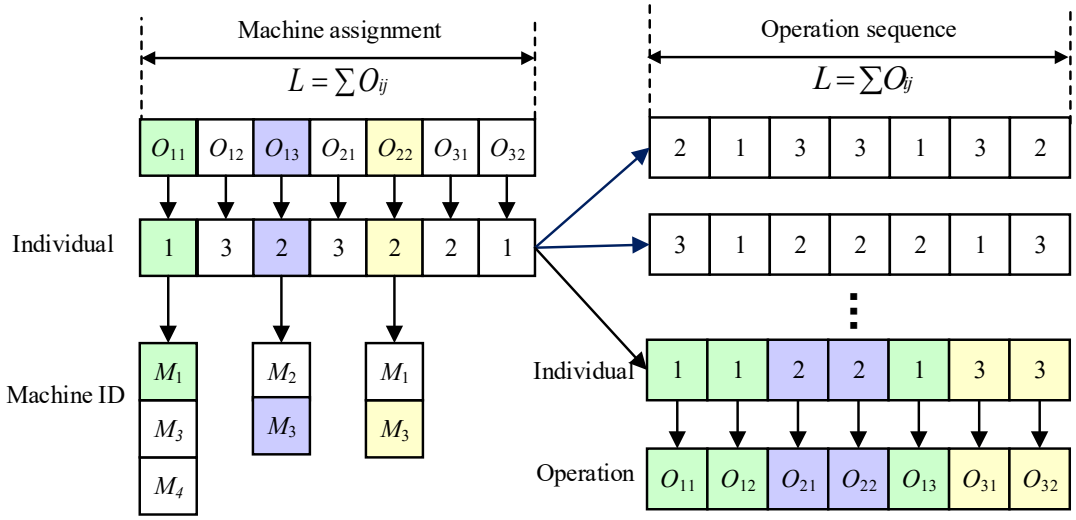

Figure 3 A chromosome representation.

1) Machine assignment: For machine selection, the length *L* of the chromosome is the total number of operations. One machine is assigned to each operation. The gene strings of the machine part are represented as integers. Each integer is generated by the three methods of generating initial solutions proposed in this paper, and no illegal solutions are generated. As shown in Figure 3, the gene

string is 1-3-2-3-2-2-1. The 1 in the first light green box means that operation O_{11} selects the first machine to be processed from among the three machines that can be selected. Operation O_{13} selects the second machine for processing, and so on.

2) Operation sequencing: When sequencing the operations, the length L of the gene string is the total number of operations. The number of integer occurrences represents the total number of operations for this artifact. The order of occurrence of an integer is the order of operation. As shown in Figure 3, the gene strings are 1-1-2-2-1-3-3 and their processing order is O_{11}-O_{12}-O_{21}-O_{22}-O_{13}-O_{31}-O_{32}.

3.2. Chromosome decoding

Decoding is the process of converting chromosomal sequences into a number of scheduleable schedules. An effective decoding approach can lead to better results. For both sequences of machine assignment and operation sequencing, we first decode the machine part and then decode the operation sequence using full insertion decoding. For the machine assignment part, it is first converted into a machine selection matrix $J_m(i,j)$, a machine processing time matrix $T(i,j)$, and a machine setup time $HM(i,j)$. We use the four operations O_{11}, O_{12}, O_{21}, O_{22} of Tables 2 and 3 to decode the following equations.

$$J_M(i,j)=\begin{bmatrix}[1\ 2\ 3\ 4] & [1\ 2\ 3\ 4]\\ [1\ 2\ 3\ 4] & [1\ 2\ 3\ 4]\end{bmatrix} \tag{5}$$

$$T(i,j)=\begin{bmatrix}[2\ 5\ 4\ 1] & [5\ 4\ 5\ 7]\\ [2\ 5\ 4\ 7] & [5\ 6\ 9\ 8]\end{bmatrix} \tag{6}$$

$$HM(i,j)=\begin{bmatrix}[1\ 3\ 1\ 1] & [3\ 1\ 2\ 6]\\ [2\ 4\ 1\ 5] & [1\ 6\ 7\ 4]\end{bmatrix} \tag{7}$$

$J_m(i,j)$ represents the set of optional machines of O_{ij}. $T(i,j)$ denotes the processing time of O_{ij} by different optional machines. $HM(i,j)$ represents the setup time for processing O_{ij} by different machines. These three time matrices are interconnected. After decoding these three time matrices, the machine selects the chromosome part. The machine for the corresponding operation is selected from the $J_m(i,j)$ matrix based on the individual integer values of the chromosome sequence from left to right.

The operation part uses full insertion decoding to reduce the maximum completion time as much as possible. The principle of this method is to insert operations in scheduled machine idle time slots. The insertion operation will advance the start time of the operation, thus allowing the total completion time to be reduced. The insertion operation is performed as follows: When h operations are processed on a certain machine M, (h-1) idle time slots will be generated. When O_{ij} is not the first operation of the job and O_{ij} has already been processed by machine M. Find the processing machine corresponding to O_{ij} and find whether the free processing segment on that machine satisfies the early insertion of O_{ij}. The specific steps are as follows.

1) If the previous operation of this machine idle time period is the same job as O_{ij}, the setup time

and transportation time will be 0.

2) Calculate CT ($CJ_{Oi(j-1)}$ + transport time) with ET (completion time for the operation before the gap T_1 + setup time). Select Max (CT, ET) and O_{ij} processing time on this machine and add to get Q. If Q is less than T_2 (the start time of the operation after the gap) then the insertion is satisfied, as in type (a) in Figure 4, and O_{ij} is inserted into the gap. If Q is greater than T_2 then the insertion is not satisfied, as in type (b) in Figure 4, and the search for the next gap continues. If all gaps are not satisfied, O_{ij} is arranged in the original order. For each completed step, the start time and completion time of the operation will be calculated and recorded. Then the value of each target is calculated according to Eqs (1)–(4).

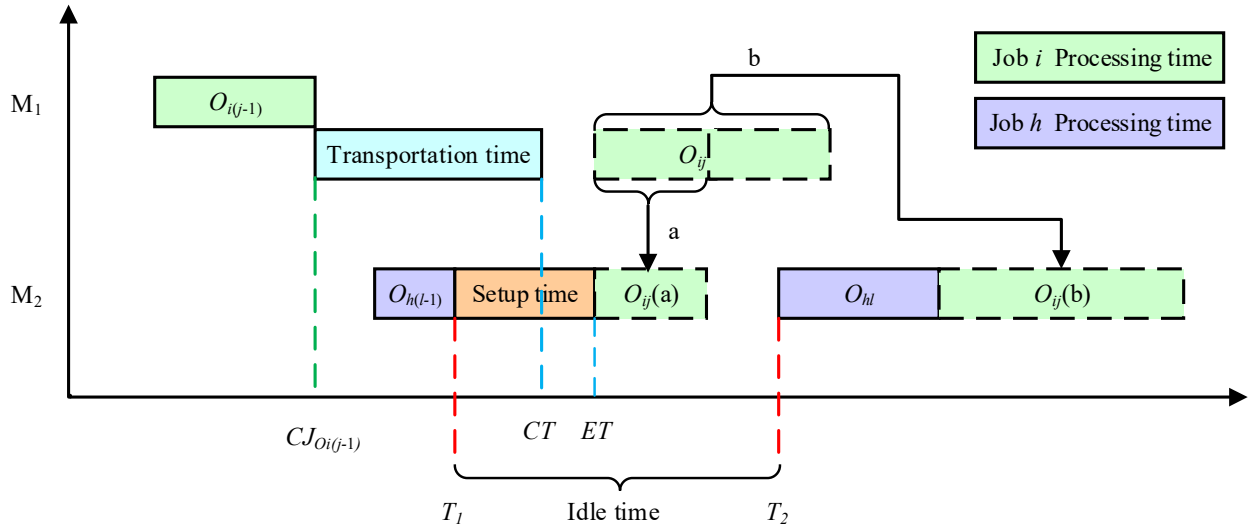

Figure 4. Full insertion decoding.

3.3. Initial population

The initialization of the population is a very important step in the algorithm. A good initial solution has an important impact on the speed and quality of the algorithm. The distributed encoding approach proposed in this paper requires more quality and diversity in the initial solution assignment by machines. Therefore, in this paper, three initialization methods are proposed to improve the quality and diversity of the initial solutions in response to this requirement.

1) Random selection method: We randomly generate a machine selection sequence. In the machine sequence, we randomly select a machine from among the machines that can be selected by the operation.

2) Least time selection method: Each operation may have more than one machine available for processing. From these, we select the machine that takes the least amount of time to process the operation.

3) Earliest start time selection method: The machine with the smallest start time is searched for in the operation of the optional machine.

Because of the need to balance the quality and diversity of the initial solutions, we search for these three different initialization methods using different probabilities. A random selection is made in 50% of a population, 30% choose the machine with the least processing operation time and 20% choose the machine with the least operation can start time.

3.4. Ant colony optimization based local search

Ant colony optimization is a probabilistic algorithm used to find the optimal path in the graph. According to the characteristics of FJSP, an improved ACO is proposed in this paper as the local search and calculation of the objective value.

In this part of the algorithm, the neighborhoods corresponding to the individuals in the machine assignment population are searched. For β objectives, β ant colonies are generated to search for a local-optimal solution on each objective. The specific steps are as follows:

1) Initialize the pheromone concentration table, and the concentration is all PC_{init}. The pheromone concentration table is shown in Figure 5, the number of rows is equal to the number of jobs and the number of columns is equal to the number of operations. The value in each grid indicates the pheromone of the selected job when machining the corresponding operation. For example, the value in grid (i, j) is to select job i as the jth processing operation.

2) Determine the operation sequencing of each ant according to the roulette method. When determining the jth processing operation, the job set is first to be processed, then the fitness value P_{ij}^k of each job is calculated by Eq (8) and composed a turntable. Each job is distributed on the turntable according to the P_{ij}^k. Each time the turntable rotates, the point indicated by the arrow is the selected job.

$$P_{ij}^k = \begin{cases} \dfrac{\tau_{ij}^{\alpha}(k) \bullet \eta_{ij}^{\beta}(k)}{\sum\limits_{s \in allwoed^k} \tau_{is}^{\alpha}(k) \bullet \eta_{is}^{\beta}(k)} & j \in allwoed^k \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

where i is the last determined job, and j is the job in the job set $allowed^k$ for jth processing, $\tau_{ij}(k)$ is the jth pheromone concentration from job i to j, $\eta_{is}(k)$ is the reciprocal of the distance d_{ij}^k from job i to j, as shown in Eq (9):

$$d_{ij}^k = \max\left\{\left[\max\left(S_J + TT, S_M + ST\right) - S_M\right], 0.1\right\} \tag{9}$$

where S_J is the allowable start time of the job, S_M is the allowable start time of the machine. T_2 is the transportation time of the job and T_3 is the setup time of the job. In fact, d_{ij}^k reflects the size of the machine idle gap that will be generated by job j at time k. If the gap is 0, it means that scheduling machining job j at this time can improve local productivity. Since 0 cannot be the denominator, d_{ij}^k is taken to be 0.1 at minimum.

Since each job has multiple operations, each job can be selected multiple times by ants, and when the maximum number of selections is reached (the number of operations for the job), the job is removed from the $allowed^k$ set. This ensures that when the $allowed^k$ set is empty, the processing order of all operations is determined, resulting in a complete scheduling solution. While determining the processing sequence of an operation, it is also necessary to calculate information such as the start time and end time of processing the operation, and with this information, the scheduling plan can guide the actual production. For each machine and each job, set its initial allowable start time S_{Mk} and S_{Ji} to 0. Whenever an operation is determined, the start time ts and the end time te are calculated for that operation, then the allowable start time S_M of the corresponding machine and the allowable start time S_J of the workpiece are updated. The equations for ts and te are shown below.

$$ts = \max\left(S_{Ji} + T_2, S_{Mk} + T_3\right) \tag{10}$$

$$te = ts + T_1 \tag{11}$$

3) When the ant selects all the jobs, the objective value of the corresponding scheduling scheme of the ant is calculated according to the Eqs (1)–(4). The time information of each operation has been calculated at the same time in the selection process, and there is no need to repeat the encoding and decoding.

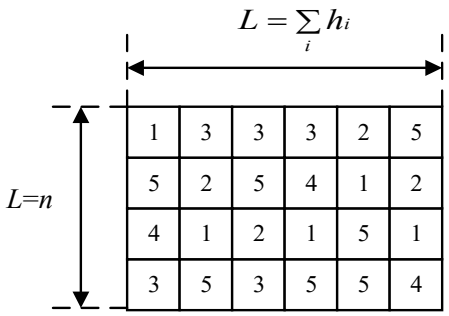

Figure 5. The pheromone concentration table.

4) Record the scheduling scheme corresponding to the optimal individual in the ant colony.

5) Update pheromone. In this paper, the pheromone update of ant colony optimization with elite strategy is used to update the pheromone secreted during the traversal of ants. Each completed cycle gives an additional pheromone increment to the path searched by the elite ant. This approach accelerates the increase of pheromones on the optimal path, which leads the subsequent ants to find the global optimal path and the global optimal solution quickly. The pheromone update with elite strategy is as follows.

$$\tau_{ij}(k+1) = (1-\rho)\tau_{ij}(k) + \Delta\tau_{ij}(k) + \Delta\tau_{ij}^{\varphi}(k) \tag{12}$$

$$\Delta\tau_{ij}(k) = \sum_{t=1}^{h_i} \Delta\tau_{ij}^{t}(k) \tag{13}$$

$$\Delta\tau_{ij}^{t}(k) = \begin{cases} Y & \text{If ant passed } (i,j) \\ 0 & \text{otherwise} \end{cases} \tag{14}$$

$$\Delta\tau_{ij}^{\varphi}(k) = \begin{cases} \omega^{\varphi}U & \text{If } (i,j) \text{ is on the optimal path} \\ 0 & \text{otherwise} \end{cases} \tag{15}$$

$\Delta\tau_{ij}(k)$ denotes the sum of the pheromone increments left by route (i, j) during the traversal of the ant. $\Delta\tau_{ij}(k)$ denotes the sum of the pheromone increments left by the elite ant pathway (i, j). ρ is

the pheromone evaporation rate. ω^{φ} is the number of elite ants.

Based on the proposed paper when giving additional pheromone increments to elite ants, it may lead to the pheromone stacking on a certain path and falling into a locally optimal solution. We control the amount of pheromones on each path by setting a maximum value and a minimum value for the pheromone values on the ant search path. Limit the amount of pheromones to $[\tau_{\min}, \tau_{\max}]$. When the pheromone value of (i, j) is greater than $\tau_{\max}$, $\tau_{ij}(k) = \tau_{\max}$. When the pheromone value of (i, j) is less than $\tau_{\max}$, $\tau_{ij}(k) = \tau_{\min}$. This results in better control of the amount of pheromones on each path.

6) Judge whether the ant colony optimization is terminated. If the algorithm is terminated, the current optimal solution will be output. If not, return to step 2) continue to execute.

3.5. Non-dominated sorting selection

Due to the existence of four objectives in this paper, the obtained objective values cannot be compared with each other. And each objective is not able to obtain an optimal solution by constraining each other. Thus the scheduling set obtained by the IACO algorithm above is a non-dominated sorting selection [16]. First, the four target values are assigned weights by the entropy weighting method, and the comprehensive score of this scheduling scheme is calculated. After the combined score of each solution is obtained, the solutions are then ranked non-dominated and the solution set is divided into different dominated layers. The solutions of different dominating layers are selected in turn up to the F_n layer. When the number of selected solutions is greater than or equal to N_{pop}, the population consisting of the first N_{pop} solutions is treated as F_1, and the population consisting of the remaining solutions is F_{2n}. The machine assignment information of the solutions in F_1 and F_{2n} is extracted to form population F_1 and population F_2. This process is shown in Figure 6(a).

The basic concept of Pareto theory is domination. Domination determines the relationship between two solutions. A condition for a solution X_1 to completely dominate another solution X_2 is that all objectives of X_1 $f_m(X_1)$ (m = 1, 2, …, N) are all better than X_2. If both solutions directly have a better objective than the other one, the two do not dominate each other. The process of delineating the different non-dominated layers is shown in Figure 6(b). The three solutions in Rank1, i, i-1, and i+1 are not dominated by each other.

The basic idea of the entropy weighting method is to determine the objective weights based on the magnitude of the variability of the indicators. According to the definition of information entropy, for a certain index, the entropy value can be used to judge the dispersion degree of a certain index. The smaller its information entropy value is, the greater the dispersion degree of the indicator, and the greater the influence of the indicator on the comprehensive evaluation. The specific steps of the entropy method are as follows.

Normalization of the target values using Eq (16).

$$y_{ij} = \frac{x_{ij} - x_{j\min}}{x_{j\max} - x_{\min}} \tag{16}$$

where x_{ij} is the original value of the jth indicator data for the ith object, y_{ij} is the jth indicator value for the ith object, $i = 1, 2, ..., n$, $n = 4 \times N_{pop}$.

Equation (17) is used to calculate the share of the ith object in this indicator under the jth indicator Y_{ij}:

$$Y_{ij} = \frac{y_{ij}}{\sum_{i=1}^{n} y_{ij}} \tag{17}$$

Calculate the entropy value e_j and the information utility value d_j for the jth indicator with the formulas shown in Eqs (18) and (19).

$$e_j = -\frac{1}{\ln m} \sum_{i=1}^{n} Y_{ij} \ln Y_{ij} \tag{18}$$

$$d_j = 1 - e_j \tag{19}$$

The weights W_j of the values of j indicators can be obtained as shown in Eq (20).

$$w_j = \frac{d}{\sum_{j=1}^{n} d_j} \tag{20}$$

Calculate the composite score F, as shown in Eq (21).

$$F = \sum w_j y_{ij} \tag{21}$$

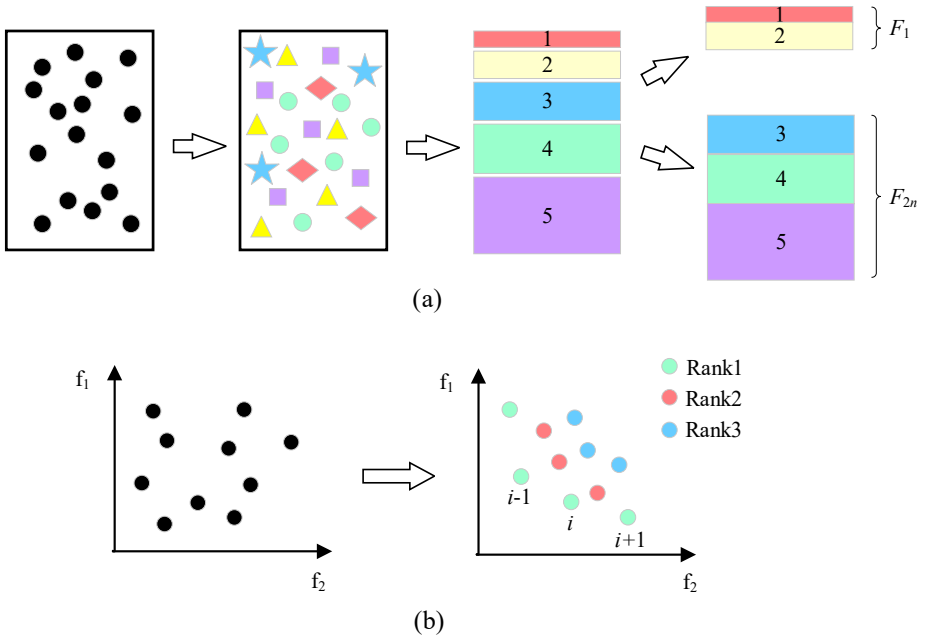

Figure 6. (a) Non-dominated sorting process; (b) Stratification process based on Pareto theory.

3.6. Mutation and closeness

Mutation operator: we design the mutation operator for the individuals in F_1 to increase the search

scope. As shown in Figure 7, let the mutation position is 1, 3, 5 and 6, and the values are randomly selected and transformed into any number of optional machines at this position. After mutation, the value in position 1 is transformed into 3 and value in position 3 is transformed into 2, etc., to generate a new individual in the machine assignment population.

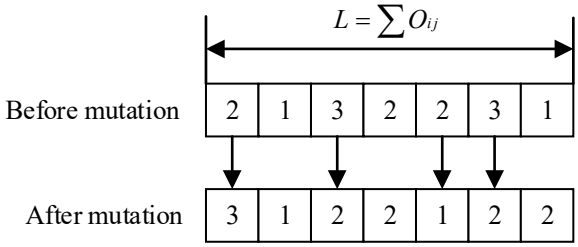

Figure 7. The mutation operator.

Closeness operator: The closeness operation is performed on the individuals in F_{2n} to approach the individuals in F_1. First, calculating the Hamming distance between the individuals in F_{2n} and F_1. The Hamming distance is the number of different corresponding positions in the two individuals. As shown in Figure 8, the information of position 1, 2 and 4 of individual 1 and individual 2 is different, so the Hamming distance between them is equal to 3. Then, each individual in F_{2n} randomly approaches to one nearest individual in F_1, approaches the machine part in the way of multi-point cross mutation, randomly generates multiple positions of cross, replaces the individual information in F_1 to the corresponding individual in F_{2n} to be approached, and the information in other positions of individuals in F_1 remains unchanged.

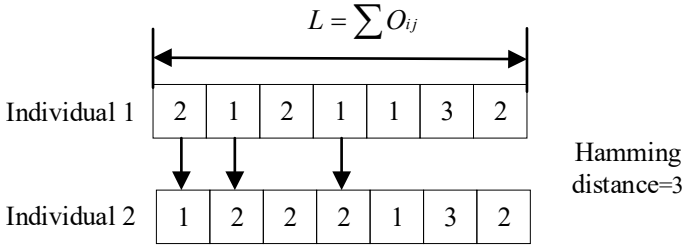

Figure 8. The Hamming distance.

The next generation with the information of machine assignment is composed by merging the two population after mutation and closeness.

4. Experimental results

4.1. Test instances

We use Matlab 2016a to perform simulation experiments for this algorithm, running on an Intel(R) Core(TM) i5-7300HQ CPU @ 2.50 GHz and 2.50 GHz. In this paper, we use the

Brandimarte dataset [28] containing 10 instances (MK01-MK10) and the Dauzere-Peres and Paulli dataset [29] containing 18 instances (01a–18a) for our comparative experiments. Where the data in the MK sets are small-medium scale problems, while the data in (01a–18a) are large-scale problems. The transportation and setup times in the instances are randomly generated. The equations for generating the start delivery date and end delivery date are

$$DE_i = \theta_i \sum_{j=1}^{h_i} \max_{k \in M_{O_{i,j}}} \left(ST_{O_{i,j},k} + PT_{O_{i,j},k} \right) \tag{22}$$

$$DS_i = DE_i - \gamma \tag{23}$$

where θ_i is the random number. The range of values in Mk01-Mk04 is [1, 1.2], the range of values in Mk06 is [0.8, 1], the value range in MK10 is [1.6, 1.8], and the value range in MK10 is [1.6, 1.8]. γ is the delivery interval, Mk1-Mk10 is 10, 01a-18a is 100. All detailed data for this paper can be found in the following websites: https://pan.baidu.com/s/16_zjfTGb6A2JMEsLVRELrA code: ysf1.

4.2. Performance metrics

To evaluate the IACO proposed in this paper, we use the set coverage (SC) [30], Hypervolume (HV) [30], the metrics of the number of non-dominated solutions (NONS), and the convergence of the algorithm are used to evaluate the algorithm proposed in this paper.

The SC is described as follows:

$$SC(A,B) = \frac{\left| \left\{ \mathbf{x} \in B \middle| \exists \mathbf{y} \in A : \mathbf{y} \succ \mathbf{x} \right\} \right|}{|B|} \tag{24}$$

A and B are the two solution sets that we are to compare. $SC(A, B)$ represents the proportion of solutions in B that are dominated by solutions in A. When $SC(A, B) = 1$ means that the solutions in B are all dominated by A. When $SC(A, B) = 0$ means that none of the solutions in B is dominated by the solutions of A. It is worth noting that $SC(A, B)$ is not necessarily equal to 1- $SC(B, A)$. If $SC(A, B) > SC(B, A)$ then it means that the number of solutions in A dominates the number of solutions in B, and the solution set A is better than the solution set B.

HV is used to represent the volume of the hypercube enclosed by the individuals in the solution set and the reference points in the target space. The convergence and distribution of the solution set S are evaluated by calculating the HV value, which is defined in Eq (25).

$$HV(S, P) = volume \left(\bigcup_{s \in S}^{S} v(s,P) \right) \tag{25}$$

S is the approximate solution set of the Pareto optimal frontier. P is the reference point corresponding to the Pareto frontier. $v(s, P)$ denotes the hypervolume of the space formed between the solution s and the reference point P in the non-prevailing solution set S. Larger HV represents better diversity of the solution set S.

4.3. Parameter settings

The algorithm proposed in this paper has three key parameters: evaporation rate (E), mutation probability (M), and pheromone factor weight (P). In order to choose the best combination of parameters for the algorithm, the currently widely used design of experiment (DOE) method of Taguchi [31] is used on a medium-sized problem (MK01). The parameter level table as in Table 4 has three parameters, each with three levels, and a total of nine different combinations of the three parameters can be seen in the orthogonal table of parameter combinations in Table 5. The HV (Hypervolume) index is used to evaluate different combinations of parameters, and the accuracy of HV is highly dependent on the choice of reference points. The reference set for this experiment is the optimal set filtered by the algorithm under MK01 instance using the first three coefficient combinations after non-dominated sorting. Figure 9 shows the parameter level trends chart. Based on the results of DOE, the optimal values of the parameters are set to $E = 0.2$, $M = 0.4$, and $P = 2$.The other parameter values of the algorithm were set as in Table 6.

Table 4. Factor level table.

Parameter	Factor level		
	1	2	3
E	0.1	0.2	0.3
M	0.4	0.5	0.6
P	1	1.5	2

Table 5. Factor parameter combination table.

Experiment number	Parameter			HV
	E	M	P	
1	1	1	1	0.2949
2	1	2	2	0.5545
3	1	3	3	0.2949
4	2	1	2	0.5545
5	2	2	3	0.6176
6	2	3	1	0.5545
7	3	1	3	0.8645
8	3	2	1	0.2949
9	3	3	2	0.5545

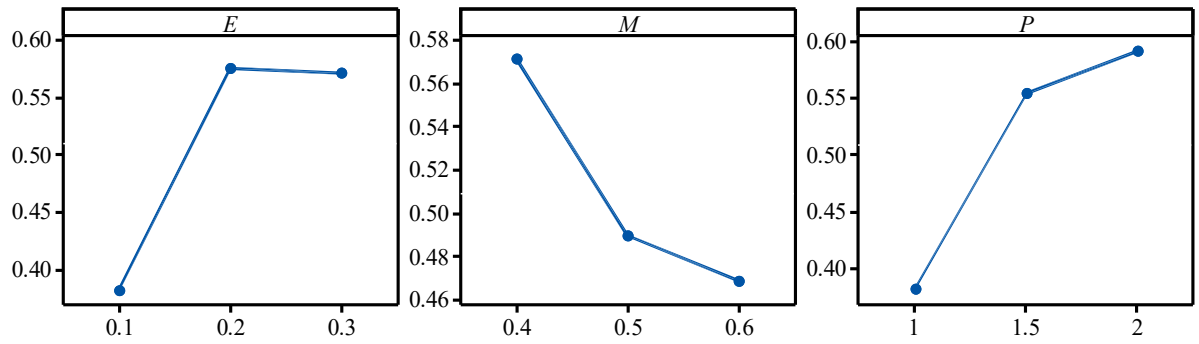

Figure 9. Parameter level trends chart.

Table 6. Parameter settings of the IACO.

Parameters	Value
population size (N)	200
maximum of global iteration	10
ant colony size	10
maximum of IACO iteration	10
initial pheromone concentration PC_{init}	1
heuristic function weight η	1
closeness probability	1
penalty for earliness (α_s)	1
penalty for tardiness (α_e)	5
Y	3
U	10

4.4. Comparison with other classical algorithms

In order to evaluate the effectiveness of the proposed algorithm in this paper more comprehensively, we compare IACO with NSGA-II [32] and IGA [15]. The reasons for choosing these two algorithms as comparison algorithms are: (i) These algorithms are multi-objective optimization algorithms, which are easy to reproduce. (ii) Both of these algorithms are very classical and have been proven to be effective. Considering the randomness of the algorithm. For each instance, each algorithm was run 10 times independently with a stopping condition of 200 iterations. We take the average of 10 times results for comparison and analysis.

The sc-based results obtained for IACO, IGA, and NSGA-II are given in Table 7. As can be seen from Table 7, IACO obtains a larger number of non-dominated solutions in the instance than the other algorithms. The bold black font in the table represents the better data between the two. In almost all instances, $SC(HA, A1) > SC(A1, HA)$, $SC(HA, A2) > SC(A2, HA)$. For example, in Mk01, Mk03, Mk04, MK05, Mk06, Mk07, Mk09, Mk10, 01a, 02a, 03a, 04a, 05a, 06a, 07a, 08a, 12a, 14a, 15a, 18a. According to the box plot in Figure 10, these relationships are even more evident. The red line in the graph is the mean value line and the small blue squares are the outliers.

Table 7. SC results (mean) between IACO, IGA, and NSGA-II.

Benchmark	IACO(*HA*) vs IGA(*A1*)		IACO(*HA*) vs NSGA-II(*A*2)		NSGA-II(*A*2) vs IGA(*A1*)	
instances	*SC*(*HA,A1*)	*SC*(*A1,HA*)	*SC*(*HA,A*2)	*SC*(*A*2,*HA*)	*SC*(*A2,A1*)	*SC*(*A1,A2*)
Mk01	**0.5556**	0.0526	**0.7632**	0.0000	**1.0000**	0.0000
Mk02	**0.9545**	0.0000	0.0000	0.0000	0.0000	0.0000
Mk03	**1.0000**	0.0000	**0.4922**	0.0000	**0.4844**	0.0000
Mk04	**0.8214**	0.0000	**0.3803**	0.0000	**0.3803**	0.0000
Mk05	**0.9565**	0.0000	**0.0455**	0.0000	0.0000	**0.7174**
Mk06	**0.6700**	0.0110	**1.0000**	0.0000	**1.0000**	0.0000
Mk07	**0.9750**	0.0000	**1.0000**	0.0000	**1.0000**	0.0000
Mk08	**1.0000**	0.0000	0.0000	**0.0308**	0.0000	**0.5000**
Mk09	**0.4737**	0.0576	**1.0000**	0.0000	**1.0000**	0.0000
Mk10	**0.6486**	0.0159	**1.0000**	0.0000	**1.0000**	0.0000
01a	**1.0000**	0.0000	**0.0817**	0.0000	0.0000	0.0000
02a	**0.8000**	0.0000	**0.4235**	0.0622	0.0000	0.0000
03a	**1.0000**	0.0000	**1.0000**	0.0000	0.0000	**0.1799**
04a	**1.0000**	0.0000	**0.7254**	0.1253	**0.5624**	0.0000
05a	**1.0000**	0.0000	**1.0000**	0.0000	**0.1642**	0.0000
06a	**0.7736**	0.0000	**0.4250**	0.0000	0.0000	**0.2611**
07a	**0.9531**	0.0000	**1.0000**	0.0000	0.0000	0.0000
08a	**1.0000**	0.0000	**1.0000**	0.0000	0.0000	0.0000
09a	**0.3846**	0.0000	**1.0000**	0.0000	**0.2425**	0.0000
10a	**0.5769**	0.0000	**0.0133**	0.0000	**0.0152**	0.0000
11a	**1.0000**	0.0000	**1.0000**	0.0000	0.0000	0.0000
12a	**1.0000**	0.0000	**1.0000**	0.0000	**0.6553**	0.0000
13a	**1.0000**	0.0000	**1.0000**	0.0043	0.0000	0.0000
14a	**1.0000**	0.0000	**1.0000**	0.0000	**0.0113**	0.0000
15a	**1.0000**	0.0000	**1.0000**	0.0000	**0.0580**	0.0000
16a	**1.0000**	0.0000	**1.0000**	0.0187	0.0000	**0.0213**
17a	**0.8333**	0.0000	0.0000	0.0000	0.0000	0.0000
18a	0.0000	**1.0000**	**0.6574**	0.1554	0.0000	**0.0072**
+/-/=	27/1/0		25/1/2		14/6/8	

Table 8 shows the non-based results from IACO, IGA, and NSGA-II. In Table 8, IACO is much better at finding non-dominated solutions than the other two algorithms, except for the two examples MK03 and MK08. According to the box plot in Figure 11, these relationships are even more evident. The red line in the graph represents the mean value line. To ensure more transparent comparison results, Pareto front plots were drawn to demonstrate the performance of IACO, NSGA-II and IGA. To confirm our experiments, Figure 12 shows the Pareto front plots for the three algorithms based on the MK02, MK07, 03a instances. These three instances were chosen because they are three examples of different scales. We can clearly see the ability of the algorithm to find non-dominated solutions from the comparison of these three instances of different scales. Each colored dot in the Figure 12 represents a non-dominated solution, and from the figure we can clearly see that IACO has many more dots than the other two algorithms. Thus it shows that the algorithm proposed in this paper is stronger than the

other two algorithms in finding non-dominated solutions on this instance.

Table 8. NONS results (mean) of IACO, NSGA-II, and IGA.

Benchmark instances	IACO *HA*	NSGA-II *A2*	IGA *A1*
Mk01	**38**	29	9
Mk02	**59**	27	22
Mk03	8	**128**	11
Mk04	**90**	71	28
Mk05	**145**	88	46
Mk06	**365**	126	100
Mk07	**163**	24	40
Mk08	65	**103**	22
Mk09	**382**	156	114
Mk10	**314**	155	37
01a	**154**	76	14
02a	**264**	134	35
03a	**289**	165	47
04a	**55**	30	43
05a	**262**	65	19
06a	**206**	136	53
07a	**195**	98	64
08a	**163**	66	22
09a	**190**	95	39
10a	**260**	97	26
11a	**137**	62	15
12a	**187**	68	23
13a	**159**	64	28
14a	**123**	63	19
15a	**159**	55	32
16a	**120**	49	26
17a	**122**	85	18
18a	**150**	76	24
+/-	26/2	2/26	0/28

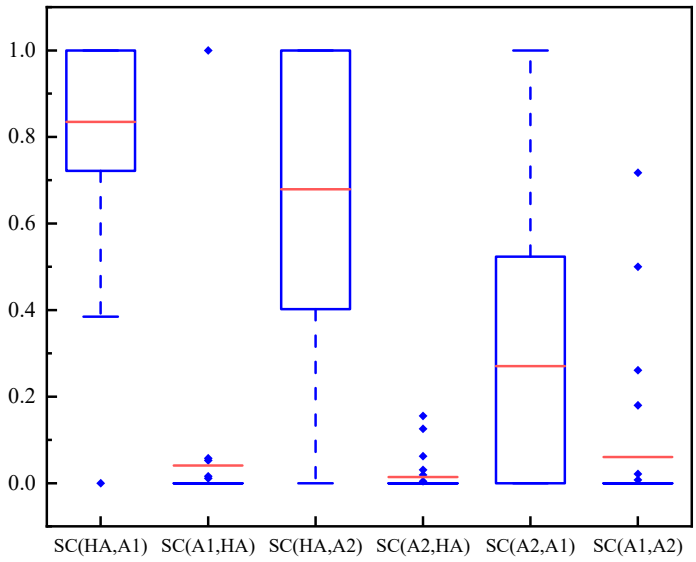

Figure 10. Box plots based on the SC results in Table 7.

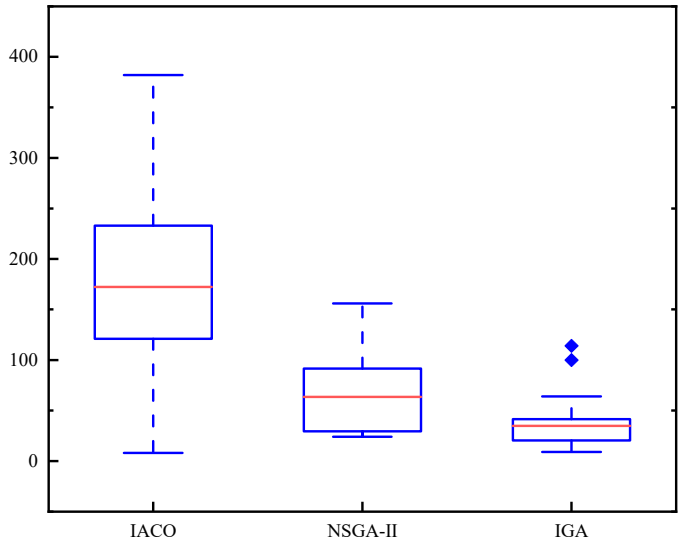

Figure 11. Box plot based on the NONS results in Table 8.

To find out the specific cause of SC(*HA*, *A1*) > SC(*A1*, *HA*), SC(*HA*, *A2*) > SC(*A2*, *HA*), We further compare the convergence of IACO and IGA, and NSGA-II. As shown in Table 9, the value in the table is the optimal value for each objective in the first of 10 experiments, where f_1 represents the makespan, f_2 represents the total workload, f_3 represents the workload of the critical machine, and f_4 represents the penalties of earliness/tardiness. As can be seen from Table 9, the f_1 of IACO is better than IGA, and NSGA-II on 20 instances; The f_2 of IACO is better than IGA, and NSGA-II on 24 instances; The f_3 of IACO is better than IGA, and NSGA-II on 24 instances; The f_4 of IACO is better than IGA, and NSGA-II on 25 instances. All four target values of IACO in these instances outperform the other two algorithms. For example, on MK05, MK09, MK10, 01a, 02a, 03a, 06a, 08a, 09a, 12a, 14a, 15a, 18a. The convergence curves of the three algorithms for the four objectives on instance MK07

and 03a are given in Figures 13 and 14. We can see from the Figure 13 that IACO outperforms the other two algorithms in terms of convergence on the other three objectives, except for f_1 where it performs slightly worse than IGA. From the Figure 14, we can see that the IACO proposed in this paper is better than the other two algorithms in terms of convergence of all four objectives in 03a instance. To demonstrate our experimental process, we give the Gantt charts obtained by applying the algorithm proposed in this paper at three different scale instances. The light blue boxes in the Gantt chart represent transportation time and the light brown boxes represent setup time. The numbers above the different boxes represent different job numbers, and the different operation numbers are indicated in the order in which they appear. The results show that the IACO structure is more suitable for minimizing the makespan, workload of the critical machine, and penalties of earliness/tardiness.

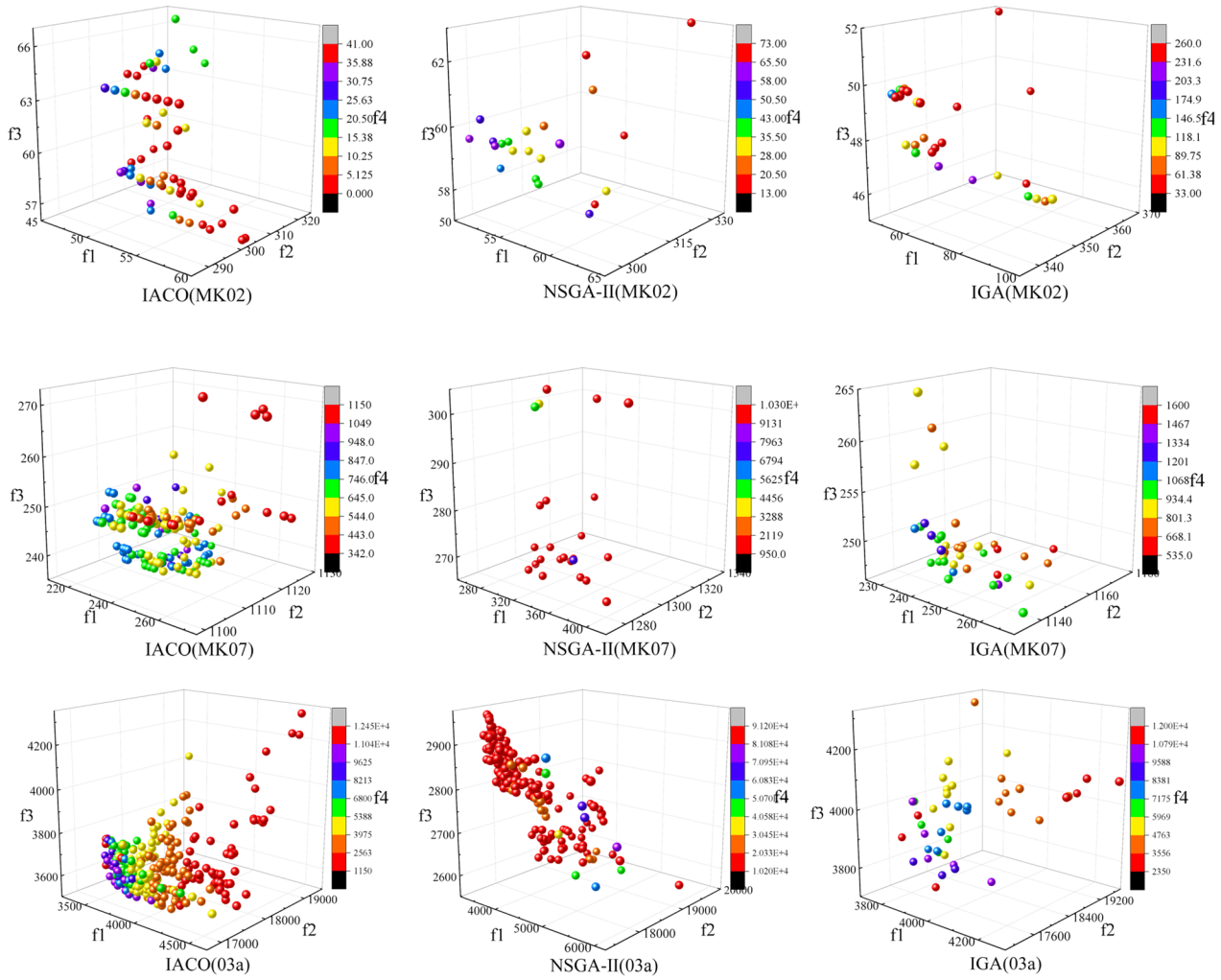

Figure 12. Pareto front plot of three algorithms based on MK02, MK07, 03a.

In summary, these results show that the proposed IACO has better performance than IGA and NSGA-II in solving the MaOFJSP.

Table 9 Convergence of IACO, NSGA-II, and IGA.

Benchmark instances	IGA				NSGA-II				IACO			
	f_1	f_2	f_3	f_4	f_1	f_2	f_3	f_4	f_1	f_2	f_3	f_4
Mk01	66	345	71	236	66	**337**	71	147	**61**	339	**54**	**71**
Mk02	54	**301**	61	61	59	315	56	42	**46**	333	**46**	**33**
Mk03	348	2227	338	1085	345	1712	331	**946**	**260**	**1033**	**246**	2441
Mk04	135	717	134	927	126	**654**	110	814	**115**	663	**103**	**343**
Mk05	269	1051	275	2191	265	1029	275	1683	**257**	**1023**	**238**	**1545**
Mk06	**183**	1178	184	**268**	184	1180	187	334	189	**1163**	**113**	2136
Mk07	**223**	1154	241	715	238	1189	235	992	229	**1092**	**233**	**276**
Mk08	775	4213	838	5072	777	4194	**833**	4832	**761**	**4116**	704	**2461**
Mk09	614	4390	580	2019	655	4382	656	2036	**606**	**4121**	**500**	**1976**
Mk10	535	3929	450	2085	537	3864	467	1906	**534**	**3839**	**394**	**1776**
01a	4126	21,555	4905	11,672	4213	24,235	5316	8952	**3985**	**21,550**	**4905**	**8500**
02a	3794	19,189	4013	2895	3955	19,555	4021	1855	**3702**	**18,791**	**3833**	**1627**
03a	3491	18,170	3684	4949	3552	17,775	3621	10720	**3417**	**16,958**	**3471**	**1627**
04a	4010	21,239	4941	15,227	**3985**	21,513	4521	13554	4021	**21,194**	**4941**	**11,779**
05a	3621	19,118	3883	3723	**3523**	19,567	3995	2698	3654	**19,059**	3903	**1670**
06a	3651	18,238	3740	4450	3699	17,569	3621	2692	**3503**	**16,585**	**3499**	**1285**
07a	**4088**	34,240	4841	32,518	4284	35,421	4787	25681	4160	**34,067**	**4693**	**20,903**
08a	3816	31,257	4192	8762	3955	32,659	4047	6632	**3739**	**30,354**	**3886**	**4558**
09a	3712	30,067	4032	5735	3788	29,621	3896	4552	**3644**	**27,637**	**3620**	**2189**
10a	4179	34,384	4733	25,905	4197	36,496	**4553**	20,318	**4133**	**34,243**	4675	**18,882**
11a	3707	31,078	4001	6173	3882	30,226	**3756**	4569	**3698**	**29,892**	3836	**3099**
12a	3675	30,419	3934	7750	3699	29,634	3873	5597	**3636**	**28,267**	**3816**	**3381**
13a	4262	45,440	5008	43,611	**4033**	**44,653**	5562	36,654	4172	44,938	**4815**	**30,585**
14a	3855	41,406	4368	11,657	3927	40,398	4273	5932	**3842**	**39,990**	**4150**	**3318**
15a	3960	41,130	4396	9919	3884	40,336	4558	6217	**3777**	**38,141**	**3949**	**5563**
16a	4212	45,837	5063	40,684	**4124**	46,682	4933	**26,423**	4155	**45,492**	**4846**	27,186
17a	**3844**	40,845	4362	15,694	3975	41,389	4258	9663	3849	**40,205**	**4136**	**5672**
18a	3927	41,706	4436	8126	4226	39,558	4365	6397	**3830**	**37,518**	**4004**	**5785**
#better	4	1	1	1	4	3	3	2	20	24	24	25

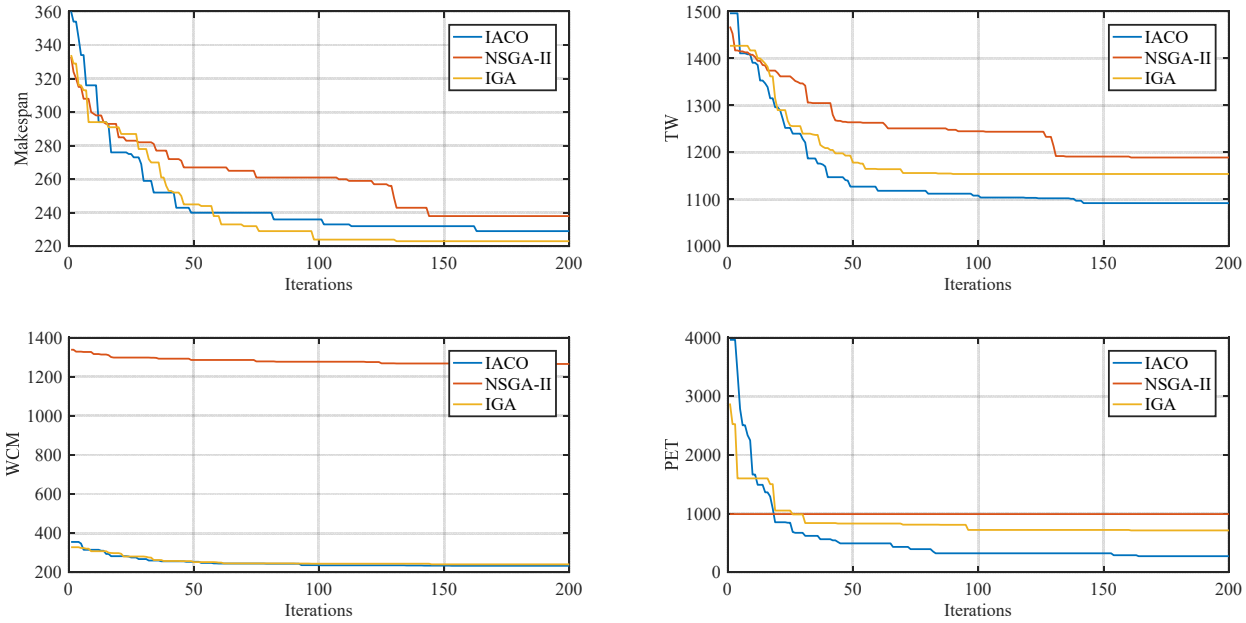

Figure 13. Convergence curves of the three algorithms based on MK07.

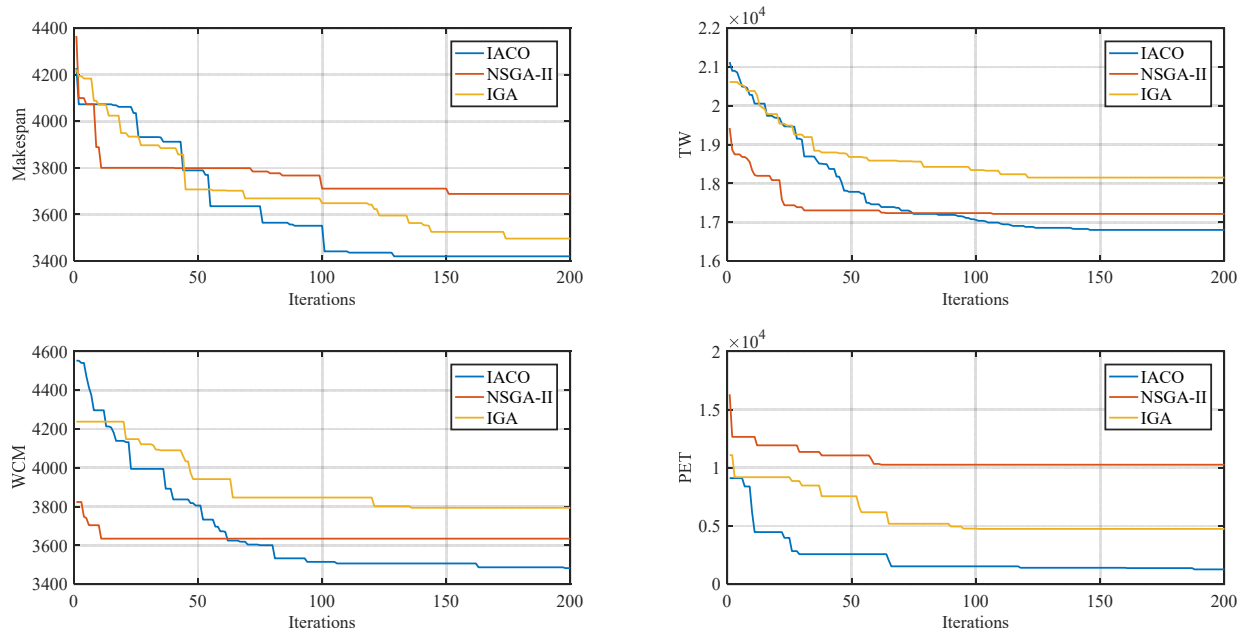

Figure 14. Convergence curves of the three algorithms based on 03a.

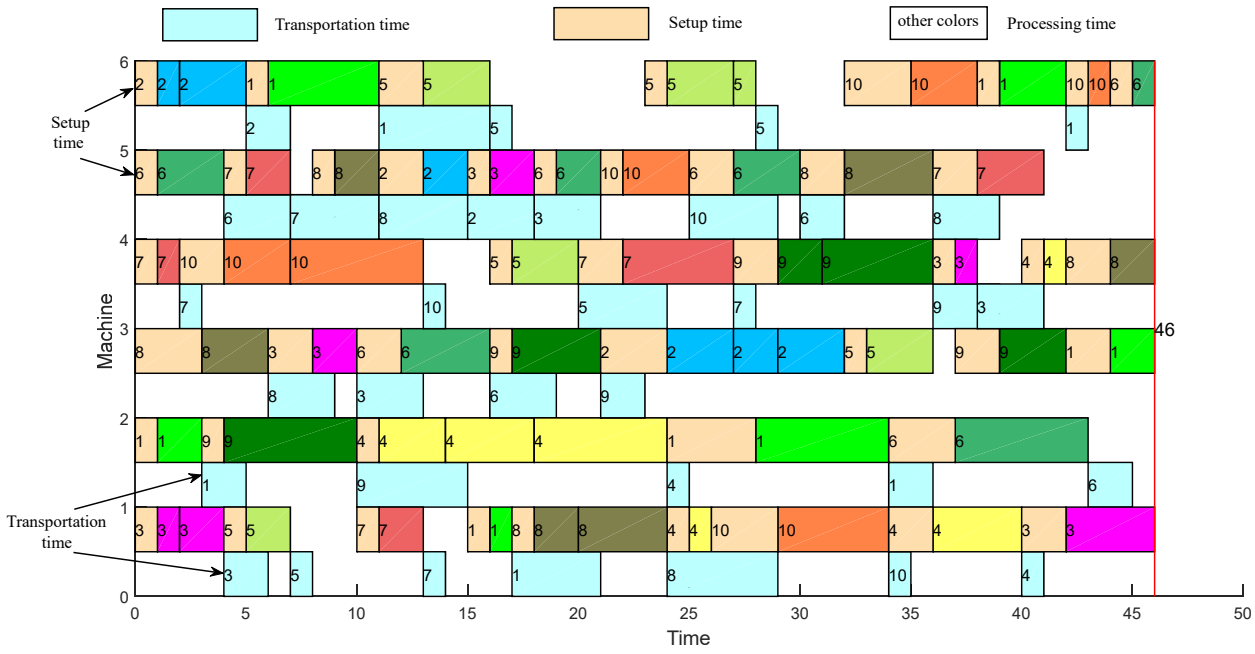

Figure 15. Gantt chart based on MK02 considering transportation time and setup time.

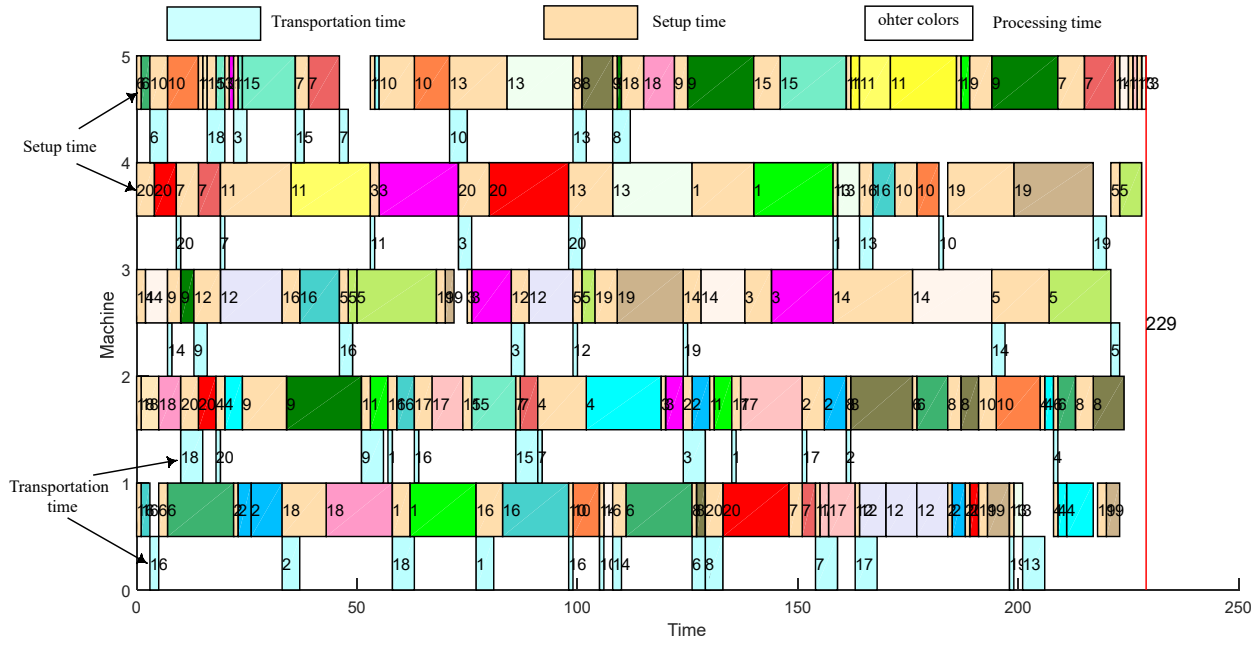

Figure 16. Gantt chart based on MK07 considering transportation time and setup time.

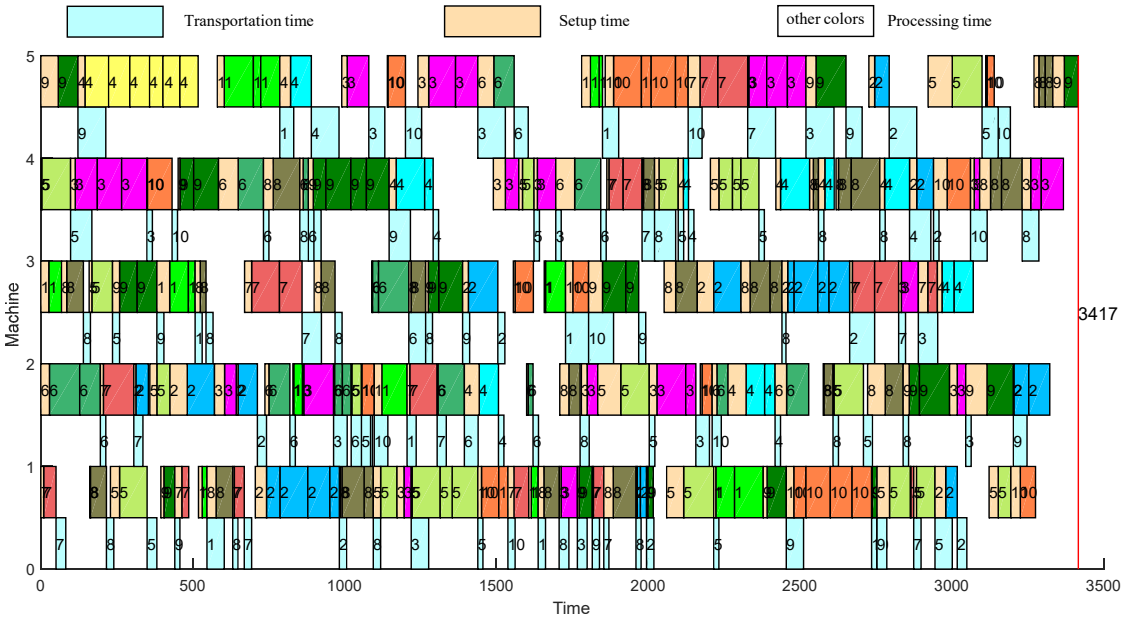

Figure 17. Gantt chart based on 03a considering transportation time and setup time.

5. Conclusions and future work

In this paper, an improved ant colony optimization is proposed to solve the multi-time-constrained multi-objective flexible job shop scheduling problem (MaOFJSP) with transportation time, setup time and delivery time. The four objectives the makespan, total workload (TW), workload of the critical machine (WCM), and penalties of earliness/tardiness (PET) were also optimized. Combining the problem characteristics, we design a distributed coding approach. The different machine neighborhoods are searched by iteratively updating the machine distribution part in the early stage of the algorithm. The local search is later performed by the improved ant colony optimization in this paper for the different neighborhoods assigned by the machine. The resulting scheduling set is sorted and filtered using entropy weight method and non-dominated sorting. In addition, we again proposed mutation and closeness operations for the machine assignment in order to improve the diversity of the population. Finally, the algorithm is evaluated and the effectiveness of the proposed algorithm is verified by experimenting with 28 benchmark instances of different sizes.

The study in this paper shows that scheduling with multiple time constraints will take more time to complete production. And this extra time is not negligible. Therefore, we hope that the research in this paper can be a good guide for the managers of shop production. In the future, our research will focus on the application of multi-objective FJSP in dynamic production scheduling environments. In addition, we can also use some emerging machine learning techniques to further improve the performance of the algorithm, such as generating high-quality initial populations by a two-stage method with the help of recurrent neural networks (RNN). Adjusting the algorithm parameters by dynamic control of reinforcement learning (RL) allows the algorithm to be self-adjusting.

Acknowledgments

This paper presents work funded by the National Natural Science Foundation of China (Nos.

U1904167, 51905494), Innovative Research Team (in Science and Technology) at the University of Henan Province (No. 21IRTSTHN018), Henan Province Philosophy and Social Sciences Planning Project (No. 2019BZX017), and Zhengzhou Science and Technology Collaborative Innovation Project (21ZZXTCX19).

Conflict of interest

The authors declare no conflict of interest.

References

1. P. Brucker, R. Schlie, Job-shop scheduling with multi-purpose machines, *Computing*, **45** (1990), 369–375. https://doi.org/10.1007/BF02238804
2. H. Zhang, G. Xu, R. Pan, H. Ge, A novel heuristic method for the energy-efficient flexible job-shop scheduling problem with sequence-dependent set-up and transportation time, *Eng. Optim.*, **54** (2022), 1646–1667. https://doi.org/10.1080/0305215X.2021.1949007
3. X. Shao, W. Liu, Q. Liu, C. Zhang, Hybrid discrete particle swarm optimization for multi-objective flexible job-shop scheduling problem, *Int. J. Adv. Manuf. Technol.*, **67** (2013), 2885–2901. https://doi.org/10.1007/s00170-012-4701-3
4. G. Zhang, X. Shao, P. Li, L. Gao, An effective hybrid particle swarm optimization algorithm for multi-objective flexible job-shop scheduling problem, *Comput. Ind. Eng.*, **56** (2009), 1309–1318. https://doi.org/10.1016/j.cie.2008.07.021
5. F. M. Defersha, D. Obimuyiwa, A. D. Yimer, Mathematical model and simulated annealing algorithm for setup operator constrained flexible job shop scheduling problem, *Comput. Ind. Eng.*, **171** (2022), 108487. https://doi.org/10.1016/j.cie.2022.108487
6. D. Kress, D. Müller, J. Nossack, A worker constrained flexible job shop scheduling problem with sequence-dependent setup times, *OR Spectrum*, **41** (2019), 179–217. https://doi.org/10.1007/s00291-018-0537-z
7. M. Li, D. Lei, An imperialist competitive algorithm with feedback for energy-efficient flexible job shop scheduling with transportation and sequence-dependent setup times, *Eng. Appl. Artif. Intell.*, **103** (2021), 104307. https://doi.org/10.1016/j.engappai.2021.104307
8. Y. Wang, Q. Zhu, A hybrid genetic algorithm for flexible job shop scheduling problem with sequence-dependent setup times and job lag times, *IEEE Access*, **9** (2021), 104864–104873. https://doi.org/10.1109/ACCESS.2021.3096007
9. Y. Du, J. Li, C. Luo, L. Meng, A hybrid estimation of distribution algorithm for distributed flexible job shop scheduling with crane transportations, *Swarm Evol. Comput.*, **62** (2021), 100861. https://doi.org/10.1016/j.swevo.2021.100861
10. J. Q. Li, Y. Du, K. Z. Gao, P. Y. Duan, D. W. Gong, Q. K. Pan, et al., A hybrid iterated greedy algorithm for a crane transportation flexible job shop problem, *IEEE Trans. Autom. Sci. Eng.*, **19** (2021), 2153–2170. https://doi.org/10.1109/TASE.2021.3062979
11. W. Ren, Y. Yan, Y. Hu, Y. Guan, Joint optimisation for dynamic flexible job-shop scheduling problem with transportation time and resource constraints, *Int. J. Prod. Res.*, **60** (2022), 5675–5696. https://doi.org/10.1080/00207543.2021.1968526
12. J. Yan, Z. Liu, C. Zhang, T. Zhang, Y. Zhang, C. Yang, Research on flexible job shop scheduling under finite transportation conditions for digital twin workshop, *Rob. Comput. Integr. Manuf.*, **72** (2021), 102198. https://doi.org/10.1016/j.rcim.2021.102198

13. J. Li, J. Deng, C. Li, Y. Han, J. Tian, B. Zhang, et al., An improved Jaya algorithm for solving the flexible job shop scheduling problem with transportation and setup times, *Knowl.-Based Syst.*, **200** (2020), 106032. https://doi.org/10.1016/j.knosys.2020.106032
14. X. Wu, X. Liu, N. Zhao, An improved differential evolution algorithm for solving a distributed assembly flexible job shop scheduling problem, *Memet. Comput.*, **11** (2019), 335–355. https://doi.org/10.1007/s12293-018-00278-7
15. G. Zhang, Y. Hu, J. Sun, W. Zhang, An improved genetic algorithm for the flexible job shop scheduling problem with multiple time constraints, *Swarm Evol. Comput.*, **54** (2020), 100664. https://doi.org/10.1016/j.swevo.2020.100664
16. K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, *IEEE Trans. Evol. Comput.*, **6** (2002), 182–197. https://doi.org/10.1109/4235.996017
17. R. Chen, B. Yang, S. Li, S. Wang, A self-learning genetic algorithm based on reinforcement learning for flexible job-shop scheduling problem, *Comput. Ind. Eng.*, **149** (2020), 106778. https://doi.org/10.1016/j.cie.2020.106778
18. X. Li, L. Gao, An effective hybrid genetic algorithm and tabu search for flexible job shop scheduling problem, *Int. J. Prod. Econ.*, **174** (2016), 93–110. https://doi.org/10.1016/j.ijpe.2016.01.016
19. G. Zhang, L. Gao, Y. Shi, An effective genetic algorithm for the flexible job-shop scheduling problem, *Expert Syst. Appl.*, **38** (2011), 3563–3573. https://doi.org/10.1016/j.eswa.2010.08.145
20. L. Wang, J. Cai, M. Li, Z. Liu, Flexible job shop scheduling problem using an improved ant colony optimization, *Sci. Program.*, **2017** (2017), 9016303. https://doi.org/10.1155/2017/9016303
21. H. Ding, X. Gu, Improved particle swarm optimization algorithm based novel encoding and decoding schemes for flexible job shop scheduling problem, *Comput. Oper. Res.*, **121** (2020), 104951. https://doi.org/10.1016/j.cor.2020.104951
22. F. M. Defersha, D. Obimuyiwa, A. D. Yimer, Mathematical model and simulated annealing algorithm for setup operator constrained flexible job shop scheduling problem, *Comput. Ind. Eng.*, **171** (2022), 108487. https://doi.org/10.1016/j.cie.2022.108487
23. G. Zhang, X. Lu, X. Liu, L. Zhang, S. Wei, W. Zhang, An effective two-stage algorithm based on convolutional neural network for the bi-objective flexible job shop scheduling problem with machine breakdown, *Expert Syst. Appl.*, **203** (2022), 117460. https://doi.org/10.1016/j.eswa.2022.117460
24. T. Jiang, H. Zhu, G. Deng, Improved African buffalo optimization algorithm for the green flexible job shop scheduling problem considering energy consumption, *J. Intell. Fuzzy Syst.*, **38** (2020), 4573–4589. https://doi.org/10.3233/JIFS-191370
25. H. Zhu, T. Jiang, Y. Wang, Discrete african buffalo optimization algorithm for the low-carbon flexible job shop scheduling problem, *J. Adv. Manuf. Syst.*, **19** (2020), 837–854. https://doi.org/10.1142/S0219686720500390
26. T. Jiang, C. Zhang, Q. M. Sun, Green job shop scheduling problem with discrete whale optimization algorithm, *IEEE Access*, **7** (2019), 43153–43166. https://doi.org/10.1109/ACCESS.2019.2908200
27. T. Jiang, C. Zhang, H. Zhu, J. Gu, G. Deng, et al., Energy-efficient scheduling for a job shop using an improved whale optimization algorithm, *Mathematics*, **6** (2018), 220. https://doi.org/10.3390/math6110220
28. P. Brandimarte, Routing and scheduling in a flexible job shop by tabu search, *Ann. Oper. Res.*, **41** (1993), 157–183. https://doi.org/10.1007/BF02023073

29. S. Dauzère-Pérès, J. Paulli, An integrated approach for modeling and solving the general multiprocessor job-shop scheduling problem using tabu search, *Ann. Oper. Res.*, **70** (1997), 281–306. https://doi.org/10.1023/A:1018930406487
30. E. Zitzler, L. Thiele, Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach, *IEEE Trans. Evol. Comput.*, **3** (1999), 257–271. https://doi.org/10.1109/4235.797969
31. D. C. Montgomery, R. H. Myers, W. H. Carter Jr, G. G. Vining, The hierarchy principle in designed industrial experiments, *Qual. Reliab. Eng. Int.*, **21** (2005), 197–201. https://doi.org/10.1002/qre.615
32. M. Yuan, Y. Li, L. Zhang, F. Pei, Research on intelligent workshop resource scheduling method based on improved NSGA-II algorithm, *Rob. Comput. Integr. Manuf.*, **71** (2021), 102141. https://doi.org/10.1016/j.rcim.2021.102141

AIMS Press