Computer Modeling in Engineering & Sciences

Tech Science Press

DOI: 10.32604/cmes.2023.028098

ARTICLE

A Novel Collaborative Evolutionary Algorithm with Two-Population for Multi-Objective Flexible Job Shop Scheduling

Cuiyu Wang, Xinyu Li and Yiping Gao*

State Key Laboratory of Digital Manufacturing Equipment and Technology, Huazhong University of Science and Technology, Wuhan, 430074, China

*Corresponding Author: Yiping Gao. Email: gaoyiping@hust.edu.cn

Received: 29 November 2022 Accepted: 13 February 2023 Published: 28 June 2023

ABSTRACT

Job shop scheduling (JS) is an important technology for modern manufacturing. Flexible job shop scheduling (FJS) is critical in JS, and it has been widely employed in many industries, including aerospace and energy. FJS enables any machine from a certain set to handle an operation, and this is an NP-hard problem. Furthermore, due to the requirements in real-world cases, multi-objective FJS is increasingly widespread, thus increasing the challenge of solving the FJS problems. As a result, it is necessary to develop a novel method to address this challenge. To achieve this goal, a novel collaborative evolutionary algorithm with two-population based on Pareto optimality is proposed for FJS, which improves the solutions of FJS by interacting in each generation. In addition, several experimental results have demonstrated that the proposed method is promising and effective for multi-objective FJS, which has discovered some new Pareto solutions in the well-known benchmark problems, and some solutions can dominate the solutions of some other methods.

KEYWORDS

Multi-objective flexible job shop scheduling; Pareto archive set; collaborative evolutionary; crowd similarity

1 Introduction

Planning is a very crucial problem in modern industries [1]. Through the elimination of scheduling conflicts, the decrease of flow time, the improvement of production resource usage, and the adaptation to unpredictable shop floor disruptions, the optimization of production scheduling can lead to considerable improvements. Job shop scheduling (JS) is a challenging problem for production planning [2]. This problem is an NP-hard problem, which is difficult to solve [3]. However, traditional JS, which assumes no flexibility of the resources for each operation of every job, might not be able to fulfill the demand of modern industries, because manufacturing systems have become increasingly flexible, and a considerable amount of automation equipment has been used [4,5]. Therefore, flexible job shop scheduling (FJS) is increasingly interesting for both industry and research.

FJS is a kind of JS that enables any machine from a certain set to process an operation, and it was first introduced in the 1990s by Bruker et al. [6]. Compared with JS, FJS is more complex, which means that JSP is NP-hard as well. Recently, a considerable amount of research has been

examined, most of which has focused on a single object. However, different departments of a company expect to maximize their objectives, such as cost and efficiency. Therefore, a single objective is not sufficient to meet the requirements of realistic production, and there is a need for further research on multi-objective FJS (MOFJS). Recently, MOFJS has attracted increasing attention from both the industrial and academic, and several methods, including evolutionary algorithms [7], and ant colony algorithms [8]. Traditionally, MOFJS contains 2 subproblems: operation sorting (OS) and machine selection (MS). Most of the related work employs integrated encoding to solve MOFJS [9]. However, OS and MS are quite different, thereby limiting the global searching ability of the current methods and seriously affecting the solving effect. Therefore, a novel collaborative evolutionary algorithm with two populations based on Pareto optimality is proposed for MOFJS. The proposed method uses Pareto optimality to address the Mutli-objectives problems, and develops a collaborative optimization strategy, which simulates the optimization process of OS and MS wherein the subsystems interact with each other. The interaction with each other in every generation can improve the solution of FJS. Thus, the proposed method takes the merits of collaborative optimization and evolutionary algorithms, and combines them to prompt the solving. The experimental results also show the availability of the proposed method.

The rest of this paper is organized as follows. Section 2 is the literature review. Section 3 is the problem formulation. Section 4 is the proposed method and Section 5 presents the experimental results. Section 6 is the conclusion.

2 Literature Review

FJS has been a research hotspot for many years, and a considerable amount of research has been conducted in recent years. For the single objective, the FJS can be categorized by exact approaches, such as mathematical programming, and approximation approaches, such as GA [10]. Meng et al. [11] presented a novel integer linear programming for FJS. Alvarez-Valdes et al. [12] developed a heuristic algorithm and applied it to glass production.

Currently, MOFJS has attracted an increasing amount of attention from both academia and industry, and several approaches have been proposed, involving evolutionary algorithms, local search methods, and swarm intelligence. Tay et al. [13] evolved the dispatching rules by genetic programming. Baykasoğlu et al. [14] performed a deep analysis of the effects of dispatching rules. Rajkumar et al. [15] introduced a greedy adaptive search, which considered the maintenance and limited resource constraints. Caldeira et al. [16] developed a multi-objective discrete Jaya by maximizing the makespan, and workload. Li et al. [17] introduced a rescheduling method with a Monte Carlo tree search, and it considered the MOFJS with dynamic events.

In summary, most of the existing approaches use an integrated encoding. However, the two subproblems in FJS (MS and OS) are quite different, and the integrated encoding might cause the solution space to be more complex, and lead to a worse solution. Thus, this paper aims to address this drawback by proposing a novel collaborative evolutionary algorithm with two populations.

3 Problem Formulation

The formulation of an $n \times m$ FJS is defined as follows.

Given a set with n jobs $J = \{J_1, J_2, J_3, \ldots, J_n\}$ and a set of m $M = \{M_1, M_2, M_3, \ldots, M_m\}$, a job j_i has an operation series $\{O_{i1}, O_{i2}, O_{i3}, \ldots, O_{iOn_i}\}$. And On_i is the total number of operations of job J_i. $O_{ij}(i = 1, 2, \ldots, n; j = 1, 2, \ldots, On_i)$ must be handled by one in the given set M. In other words, FJS can be regarded as deciding the sequence of assignment and operation under the criteria.

In this research, the following criteria are involved:

Makespan: the maximal finish time;

Maximal machine workload (MMW): the maximal time on a machine.

Total workload of machines (TWM): the total running time of all machines.

Based on these criteria, the MOFJS in this research has three objectives, and the descriptions of these objectives are shown below. n denotes the number of jobs. m presents the number of machines. o_{ij} means the j-th operation of the job i. On_i is the total number of the operations of job i. c_{ijk} denotes the earliest finish time. c_i denotes the earliest finish time of job i. W_k denotes the workload of machine k. All the objectives are formulated below:

$$f_1 = Makespan = \max_{i=1}\{c_i\} \quad i \in [1,n]\, c_i = \max_{j=1}\left\{c_{ijk}\right\} j \in \{1,2,\ldots,On_i\} \tag{1}$$

$$f_2 = MMW = \max_{k=1}\{W_k\} \quad k \in \{1,2,\ldots,m\} \tag{2}$$

$$f_3 = TWM = \sum_{k=1}^{m} W_k \quad k \in \{1,2,\ldots,m\} \tag{3}$$

The model of FJS is followed to [18], and the summary of the notations is presented in Table 1.

Table 1: Summary of the notations in the proposed method

Symbol	Notation
J	Jobs
M	Machine
O	Operation
$Makespan$	The maximal finish time
MMW	Maximal machine workload
TWM	Total workload of machines
W_k	Workload of machine k
n	Number of Jobs
m	Number of machines

4 Proposed Method for MOFJS

4.1 Basic Concepts

Generally, the formulation of a multi-objective optimization [19,20] (contains n variables, k objectives and m constraints) can be defined as:

$$\begin{aligned} &Min\ y = f(x) = \{f_1(x), f_2(x), \ldots, f_k(x)\} \\ &s.t.\ e(x) = \{e_1(x), e_2(x), \ldots, e_m(x)\} \le 0 \\ &\qquad x = (x_1, x_2, \ldots, x_n) \in X \\ &\qquad y = (y_1, y_2, \ldots, y_n) \in Y \end{aligned} \tag{4}$$

x and X are the variable and variable space, respectively. y and Y denote the objective and objective space, respectively. $e(x)$ means the constraints, and $x_f = \{x \in X | e(x) \le 0\}$ is the feasible space.

Non-dominated solution: if the objective function $f_i(a)$ is all better than $f_i(b)$, then solution b is dominated by solution a. For example, for a minimization problem, if $f_i(a) \le f_i(b)$, the solution b is

dominated by solution *a*. Otherwise, solution *b* is not dominated by solution *a*. This is a non-dominated solution. Fig. 1a shows a group of solutions for a two-objective problem.

Pareto-optimal: when the solution cannot be dominated, it is called Pareto-optimal (PO). Fig. 1b shows the Pareto front of a two objectives problem.

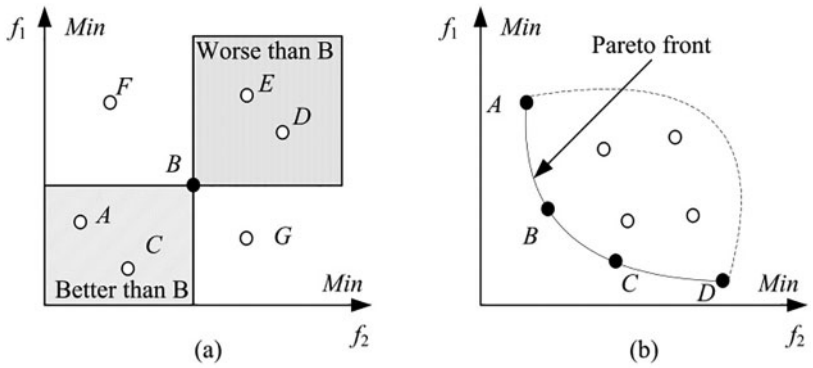


Figure 1: NDS and Pareto front

4.2 Proposed Method

4.2.1 The Optimization Strategy for Collaborative Evolution

The proposed method uses a collaborative strategy to search for a solution in both the OS and MS. The OS and MS are optimized alternatively, and evaluated together. Moreover, it should be noted that this strategy is generic, and it can be suitable to combine with various algorithms, such as GA. In this paper, the proposed method uses an evolutionary algorithm (EA) for both OS and MS, and the optimization strategy is shown in Fig. 2 and below:

***1)*: Parameter Initialization.** Assum given *n* jobs. Generating two initial populations for OS and MS individually;

***2)*: Fitness Evaluation.** Select the *Popsize* randomly, and calculate the solutions population. After that, evaluate each solution individually and choose the NDS (NDS);

***3)*: Condition judgement.** If it meets the end condition, just terminate and output the solution. Otherwise, going to 4);

***4)*: Collaborative evolution.** Generate some new individuals. Go back to 2).

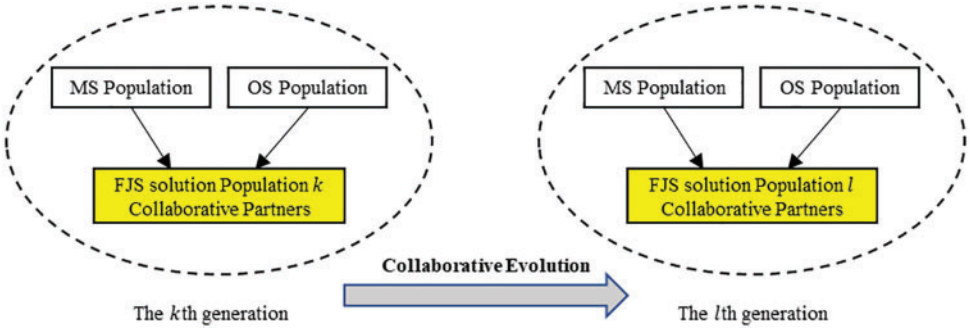


Figure 2: Optimization strategy of the proposed method

4.2.2 Pareto Archive Set

For the diversity, a Pareto archive set (PAS) is employed in the proposed method, which can retain the quantity of the NDS with the user requirement [21]. When optimizing, the PAS is moving to the Pareto-optimality front by replenishing some new NDS and eliminating some dominant solutions. Once the number of NDSs is sufficient, a crowding similarity in Eq. (5) is employed (CD) to remove the redundant solutions and guarantee diversity. Individuals with higher CD are preferentially kept in the PAS.

$$CD_p = \sum_{i=1}^{k} (f_{i(p+1)} - f_{i(p-1)}) \tag{5}$$

where p denotes the number of the individuals and i is the i-th objective.

4.2.3 Pareto Sort Algorithm

In MO problems, one objective is not enough to evaluate the quality of the solution individually, and all of the objectives must be involved. Thus, a sorting algorithm is essential, and the non-dominated sorting method [22] is adopted. The sorting algorithm separates the solutions into different levels. For example, as shown in Fig. 3, the solutions have three levels, and the lower level has the better fitness. For the same levels, the solutions that have higher CDs are also a priority. With this separation, the good solutions can be retained as a priority by the PAS.

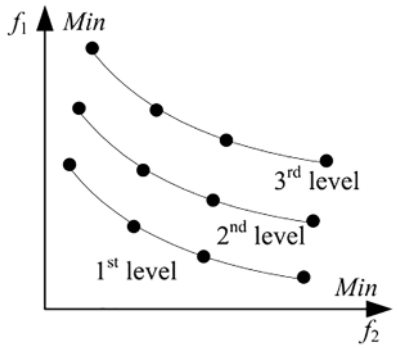


Figure 3: NDS levels

4.2.4 Encoding and Decoding Method

The encoding method is essential for an EA, and the proposed method adopts the method in [23] for encoding. Moreover, since the OS and MS are quite different, the details of the encoding method for each subproblem are also different. The encoding for OS is based on the operation representation. In addition, the representation employs an undivided arrangement. With this encoding, each job can be selected by On_i times. The f-th occurrence of a work number refers to the f-th operation of that work number, and thus, any arrangement of chromosomes can convert into a viable solution.

The MS chromosome refers to the M, and the length of this chromosome is $\sum On_i$. The i-th part means the chosen set of machines for the operation corresponding to job i. Assuming that O_h of J_i can be handled by a machine set $s_{ih} = \{m_{ih1}, m_{ih2}, \ldots, m_{ihc_{ih}}\}$, the i-th part is denoted as $\{g_{i1}g_{i2} \ldots g_{ih} \ldots g_{iOn_i}\}$. This denotes that O_h is assigned to the g_{ih}-th machine.

An encoding solution contains an OS chromosome and an MS chromosome, and the decoding manner involves semi-active, active, non-delay, and hybrid. In the proposed method, to achieve a better solution, an active schedule is used. The decoding process is shown below:

m: number of the machines.

o_{ij}: the j-th operation of the i-th job.

as_{ij}: the valid start time of o_{ij}.

s_{ij}: the earliest start time of o_{ij}.

k: the alternative machine corresponding to o_{ij}.

t_{ijk}: the processing time of operation o_{ij}on machine k.

c_{ij}: the earliest finish time of operation o_{ij}, i.e., $c_{ij} = s_{ij} + t_{ijk}$.

The process of decoding in the proposed method:

***1)*:** Generate the machine of each operation by the MS chromosome.

***2)*:** Choose a set of the operations for each machine: $m_{\mathrm{a}} = \{\mathrm{o_{ij}}\}\ 1 \le \mathrm{a} \le \mathrm{m}$.

***3)*:** Choose a set of machines for each job: $Jm_d = \{machine\}\ 1 \le d \le n$.

***4)*:** The allowable begin time for every operation: $as_{ij} = c_{i(j-1)}\ (o_{ij} \in m_a)$, $c_{i(j-1)}$ is the finish time of the pre-operation of o_{ij} for the same job.

***5)*:** Checking the unused time of the machine of o_{ij}, and getting the unused areas $[t_s, t_e]$, checking the areas in turn (if: $\max(as_{it}, t_s) + t_{ijk} \le t_e$, the earliest begin time is $s_{ij} = t_s$, else: check the next area), if the area cannot meet the condition: $s_{ij} = max\left(as_{ij}, c\left(o_{ij} - 1\right)\right)$, $c\left(o_{ij} - 1\right)$ is the finish time of the pre-operation of o_{ij} for the same machine.

***6)*:** Calculate the finish time of each operation by $c_{ij} = s_{ij} + t_{ijk}$.

***7)*:** Generate the sets of the begin time and finish time for each operation of each job by $T_d\left(s_{ij}, c_{ij}\right)\ 1 \le d \le n$.

In the decoding process, it can obtain the set of start time and finish time for each operation of each job, and it is a schedule solution for the workshop. Fig. 4 gives an example of the encoding and decoding in the proposed method. Fig. 4a is the background of this case, which involves 3 jobs and 3 machines. Fig. 4b presents an OS chromosome with repetitions of job numbers, and Fig. 4c is an MS chromosome. The OS and MS chromosomes in Fig. 4 form a solution of FJS. And this solution can be converted into a schedule by decoding, as shown in Fig. 4d. The decoding steps are as follows:

***1)*:** Generate the machine of each operation, O_{11} on M_2, O_{12} on M_1, O_{21} on M_3, O_{22} on M_1, O_{31} on M_3, O_{32} on M_2;

***2)*:** Choose the set of operations for each machine: $M_1 = \{O_{12}, O_{22}\}$, $M_2 = \{O_{11}, O_{32}\}$, $M_3 = \{O_{21}, O_{31}\}$;

***3)*:** Choose the set of machines for each job: $J_1 = \{M_2, M_1\}$, $J_2 = \{M_3, M_1\}$, $J_3 = \{M_3, M_2\}$;

***4)*:** The allowable begin time for each operation is calculated as follows: $as_{ij} = c_{i(j-1)}\ (o_{ij} \in m_a)$. $c_{i(j-1)}$ is the finish time of the pre-operation of o_{ij} for the same job. For example, the allowable start time of O_{12} is the finish time of O_{11}. This can guarantee that the obtained schedule is feasible. However, this cannot ensure that the schedule is active. Therefore, on the premise of avoiding destroying the feasibility of the schedule, Step 5 tries to insert the following operations into the earlier hole in schedule;

***5)*:** Check the unused time of the machine of o_{ij}, and give the idle areas $[t_s, t_e]$, check these areas in turn (if: $\max(as_{it}, t_s) + t_{ijk} \leq t_e$, the earliest begin time is $s_{ij} = t_s$, else: checking the another area), if there is no area that can meet this condition: $s_{ij} = max\left(as_{ij}, c\left(o_{ij} - 1\right)\right)$, $c\left(o_{ij} - 1\right)$ is the finish time of the pre-operation of o_{ij} for the same machine. For example, for arranging O_{22} on M_1, its allowable stating time is 6 ($as_{22} = 6$). However, before time 6 in M_1, there is a hole (from time 1 to 3). Now, we need to consider whether this operation can be inserted into the hole or not. Because this hole is smaller than the processing time of O_{22}, this operation cannot be inserted into this hole. Its earliest begin time is 6 ($s_{22} = 6$);

***Step 6*:** The finish time of each operation is calculated as follows: $c_{ij} = s_{ij} + t_{ijk}$. For example, for the O_{22}, its finish time is 13 ($c_{22} = s_{22} + t_{221} = 6 + 7 = 13$);

***Step 7*:** Generate the sets of begin time and finish time: $T_d\left(s_{ij}, c_{ij}\right) 1 \leq d \leq n$.

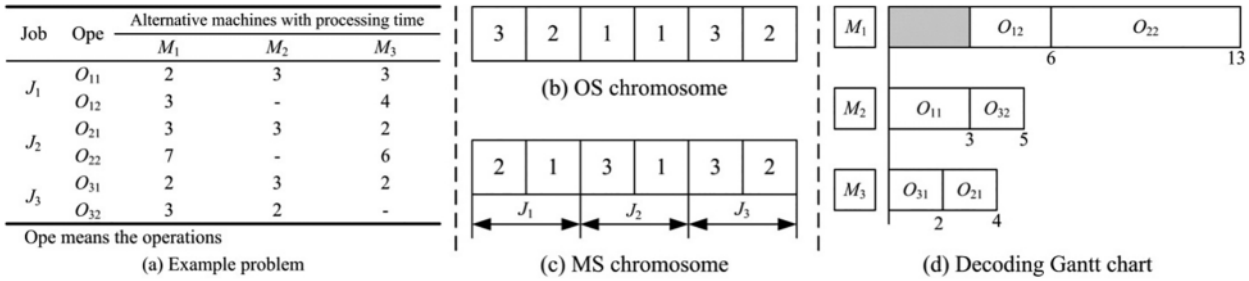


Figure 4: Encoding and decoding example

4.2.5 Genetic Operators

The genetic operator (GO) is important for good individuals and solutions. Generally, GO is classified into three categories: selection, crossover and mutation. In the proposed method, for a better solution, OS and MS use different genetic operators. The details of the genetic operations are shown below:

(1) Selection

OS and MS randomly select the individuals.

(2) Crossover

A precedence operation (PO) is used for the crossover in the OS. The PO crossover can retain the good of the parents and propagate it to the offspring. The flowchart of the PO is shown below (P1 and P2 are parents, and O1 and O2 are offspring).

1): Dividing the job set $J = \{J_1, J_2, J_3, \ldots, J_n\}$ into two parts *Jobset1* and *Jobset2*.

2): For an element belonging to *Jobset1* in P1, it is retained in the same position in O1 and removed in P1. For an element belonging to *Jobset2* in P1, it is retained in the same position in O2 and removed in P2.

3): Remains in P2 are moved to the empty positions in O1, while the remainder in P1 are moved to O1.

The MS population uses a two-point crossover, which selects two positions randomly first, and the two strings swap all elements.

Fig. 5a is an example of the PO for OS and Fig. 5b is for MS.

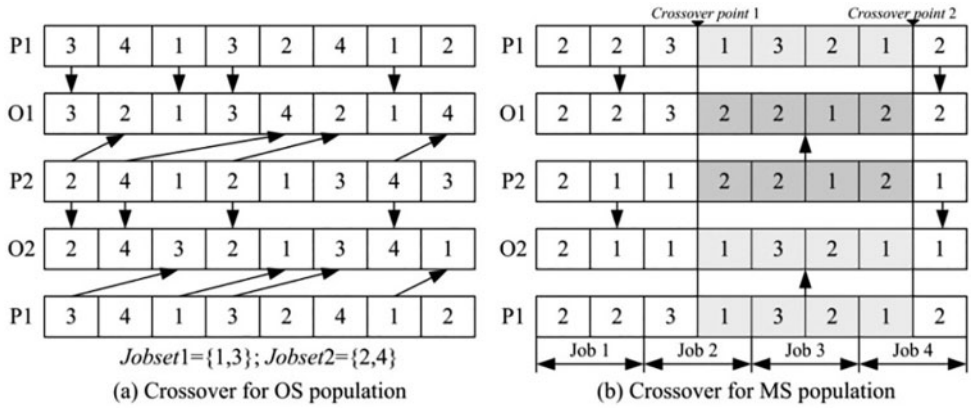


Figure 5: The crossover operations for OS and MS

(3) Mutation

The OS uses a neighborhood mutation operator.

***1)*:** Select three elements in a parent (each element has different value), and generate the neighborhood chromosomes.

***2)*:** Choose a chromosome randomly for the chromosome.

MS uses the mutation operator below.

***1)*:** Select r positions in a parent (r is the 1/2 of the chromosome);

***2)*:** For each position, change the value of the chosen position to the corresponding operations.

4.2.6 *Stop Condition*

If the iteration has reached the maximum epoch, the proposed method stops running.

4.2.7 *Framework of the Proposed Method*

Fig. 6 presents the flowchart, and the description of the proposed method is shown below:

***1)*:** Setting the parameters, including $Popsize_1$, $Popsize_2$, $Popsize_{SP}$, $Popsize_{AS}$, $maxGen$, crossover rate of OS (pc_1), crossover rate of MS population (pc_2), mutation probability of OS population (pm_1), mutation probability of MS population (pm_2).

***2)*:** Initialization. For n jobs, the encoding methods are used to generate two populations for OS with $Popsize_1$ individuals and MS with $Popsize_2$ individuals.

***3)*:** Evaluation. Randomly select $Popsize_{SP}$ individuals from every population separately to form an FJSP solution population with $Popsize_{SP}$ FJSP solutions. The considered objectives of each solution are calculated, and the Pareto sort algorithm is used to sequence the population and divide these solutions into several levels. If $Gen = 1$, generate the PAS, copy all the NDS to the PAS. If the number of solutions $Size_{SL} \geq Popsize_{AS}$, selecting the solutions with bigger CD to copy into the PAS until the PAS is full; if $Size_{SL} < Popsize_{AS}$, select the NDS to the PAS and set the other positions in PAS is NULL; If $Gen > 1$, update the PAS: compare each new

non-dominant solution with the PAS in turn. If the new solution dominates the PAS, it can be retained in the PAS.

4)*:** If the model meets the stop condition, go to ***6), otherwise, go to ***Step 5)***.

***5)*:** Collaboration evaluating. Upgrade the populations: generate new individuals for each population;

***5.1)*:** For the q-th population, use the genetic operator to generate the new population;

5.2)*:** If $q \leq 2$, go to ***5.3), else, set $Gen = Gen + 1$ and go to ***3)***.

5.3)*:** Set $q = q + 1$ and go to ***5.1).

***6)*:** Output the NDS in PAS.

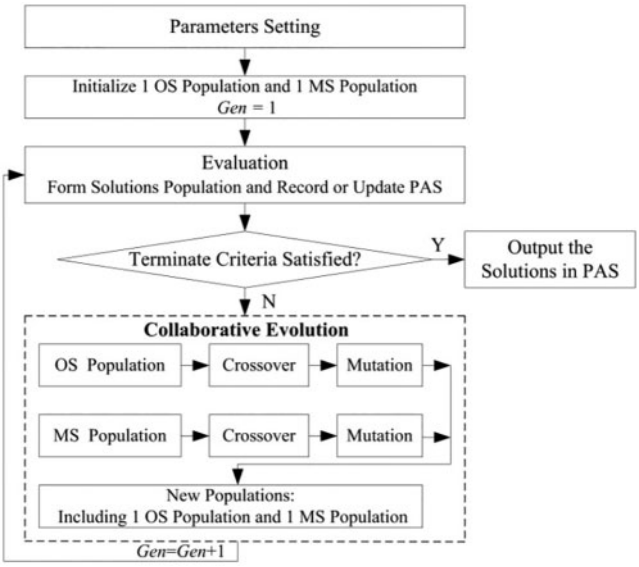


Figure 6: Work flow of the proposed method

5 Experimental Results of the Proposed Method

5.1 Experimental Setting and Results

The proposed method codes in C++ on a laptop. To evaluate the proposed method, six experiments were selected. The five experiments are adopted from the related papers, and the last experiment is employed by the author themselves. For the parameters, the sizes of the OS and MS populations are $Popsize_1 = 400$ and $Popsize_2 = 400$. The $Popsize_{SP}$ is 400 and $Popsize_{AS}$ is 5, and the $maxGEN$ is set to 200. The crossover rates of OS and MS are 0.9 and 0.9, and the crossover rate of OS and MS are 0.9. The mutation rate of OS and MS are 0.1.

5.1.1 Experiment 1

This experimental problem is from [24], which includes 5 kinds of problems. Tables 2 and 3 show the comparison results. From Table 2, the proposed method finds another new Pareto solution for problem 4 × 5, which is (13, 7, 33), as shown in Fig. 7a. For problem 10 × 7, compared with other algorithms, the proposed algorithm also finds two Pareto solutions. Based on Table 2, the proposed method finds two solutions that dominate the P-DABC. One solution (11, 11, 61) is shown in Fig. 7b.

Table 2: The experimental results of 4 × 5 and 10 × 7

	Makespan	MMW	TWM	Makespan	MMW	TWM
Problem	4×5			10 × 7		
HPSO [25]	11	10	32	–	–	–
SM [26]	11	10	32	11	11	61
	11	9	34	11	10	62
	12	8	32	12	12	60
SACO [27]	12	8	32	11	11	61
				11	10	62
GAIE [28]	11	10	32	–	–	–
	11	9	34			
	12	8	32			
P-DABC [29]	11	10	32	*12*	*11*	*61*
	12	8	32	*11*	*11*	*63*
	13	7	33	12	12	60
Proposed method	11	10	32	11	10	62
	12	8	32	11	11	61
	13	7	33			

Note: - means the result was not given by the author.

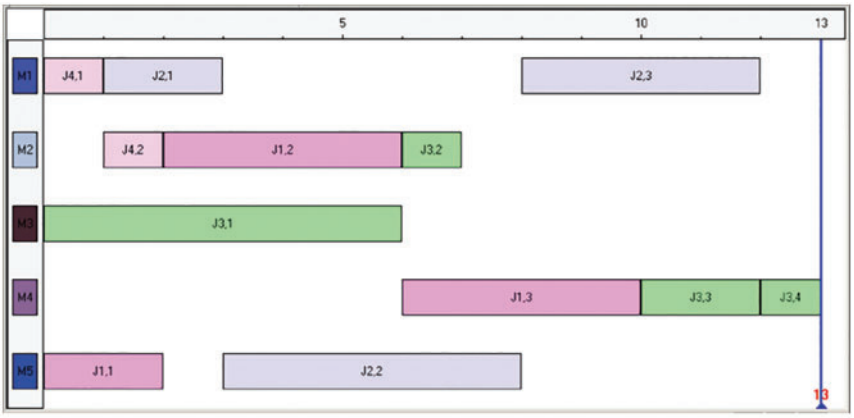
Table 3: Experimental results of the proposed method in Experiment 1

	Makespan	MMW	TWM	Makespan	MMW	TWM	Makespan	MMW	TWM
Problem	8 × 8			10 × 10			15 × 10		
MOEA-GLS	16	13	73	8	7	41	11	10	93
[30]	15	12	75	8	5	42	11	11	91
	14	12	77	7	5	43			
	16	11	77	7	6	42			
PSO-SA [31]	16	13	73	*7*	*6*	*44*	*12*	*11*	*91*
	15	12	75						
moGA [32]	15	14	73	–	–	–	–	–	–
hGA [33]	15	12	75	7	5	43	11	11	91
HPSO [25]	15	12	75	6	7	43	*11*	*11*	*93*
	14	12	77						
SM [34]	16	13	73	8	5	42	11	11	91
	16	11	77	7	6	42			
	14	12	77	8	7	41			
GA-IE [28]	15	11	81	8	5	42	11	11	91
	15	12	75	7	6	42	12	10	95
	16	13	73	8	7	41	11	10	98

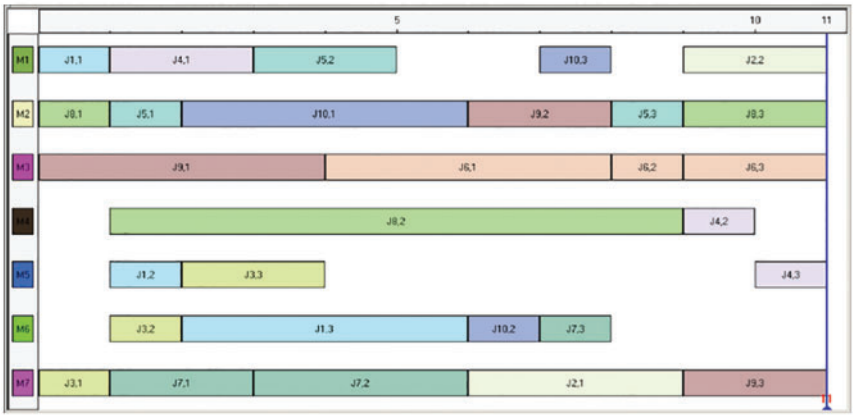
(Continued)

Table 3 (continued)

Problem	Makespan	MMW	TWM	Makespan	MMW	TWM	Makespan	MMW	TWM
	8 × 8			10 × 10			15 × 10		
				7	*5*	*45*			
PSO-LS [35]	15	12	75	8	7	41	12	10	93
	16	13	73	7	6	42	11	11	91
	14	12	77	8	5	42			
	16	11	78	7	5	43			
	17	11	77						
GA-VND [35]	14	12	77	7	5	43	11	11	91
Proposed	16	13	73	7	5	43	11	11	91
method	14	12	77	7	6	42	11	10	93
	15	12	75						



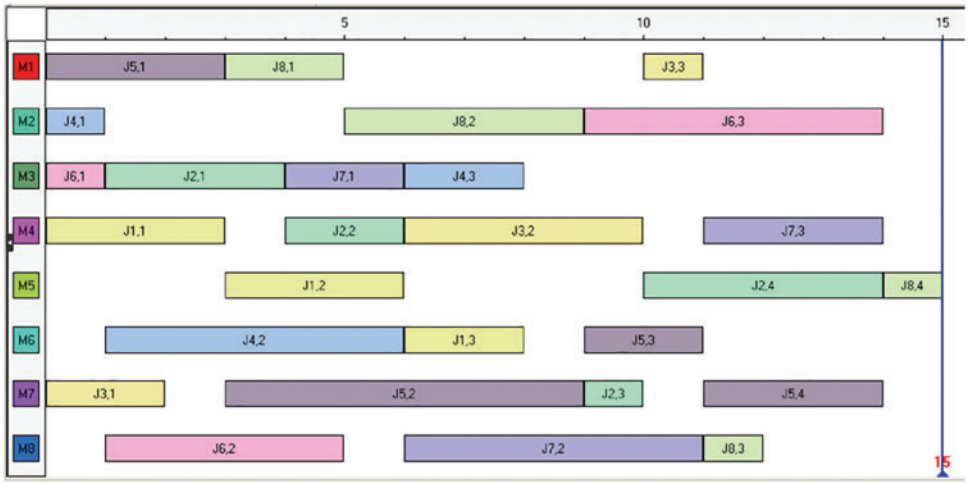
(a) Gantt chart for problem 4×5 (13, 7, 33)



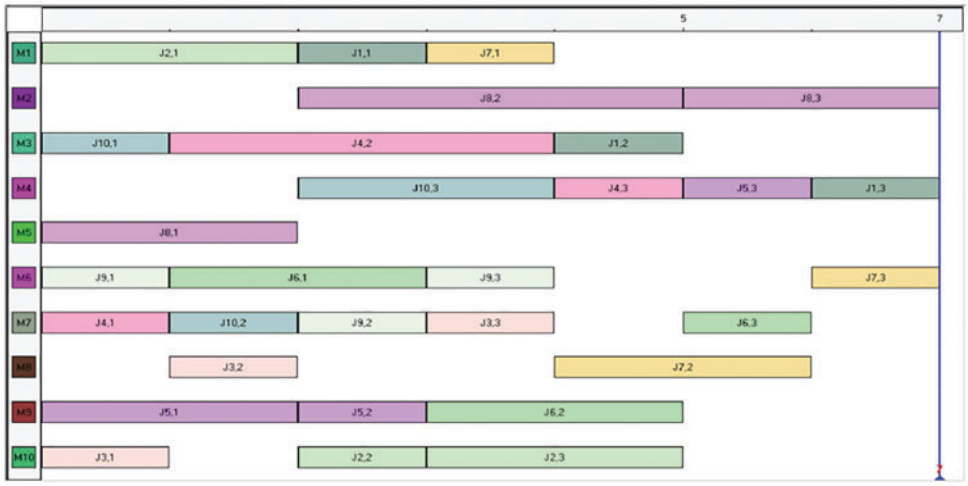
(b) Gantt chart for problem 10×7 (11, 11, 61)

Figure 7: Gantt charts for problem 4 × 5 and problem 10 × 7

Table 3 is the comparisons with the other methods for problems 8 × 8, 10 × 10 and 15 × 10. From Table 3, the proposed method finds several solutions that dominate the other methods. Fig. 8 is the Gantt chart with different solutions that are solved by the proposed method. From the experimental results, it can be seen that the proposed method achieves the best performance. Compared with the other methods, the results of the proposed method are improved for the makespan, MMW and TWM, which means that the proposed method can find new Pareto solutions.



(a) Gantt chart for problem 8×8 (15, 12, 75)



(b) Gantt chart for problem 10×10 (7, 5, 43)

Figure 8: Gantt charts solved by the proposed method

5.1.2 Experiment 2

This experimental problem is from [15], which includes 3 kinds of problems. Table 4 is the comparison with the other algorithms. From Table 4, the proposed method finds several solutions that dominate the other methods. Fig. 9 is the Gantt chart of a solution for problem 12 × 5. Similar to the experimental results in Section 5.1.1, the experimental results also suggest that the proposed method

has improved performance. Compared with the other methods, the proposed method can find a better solution.

Table 4: Experimental results of the proposed method in Experiment 2

	Makespan	MMW	TWM	Makespan	MMW	TWM	Makespan	MMW	TWM
Problem	8 × 5			12 × 5			8 × 8		
GA [15]	*27*	*27*	*109*	*33*	*33*	*145*	–	–	–
GRASP [15]	24	24	101	*33*	*33*	*138*	16	13	73
Proposed method	24	24	101	33	33	137	14	12	77
	27	25	100	31	30	140	16	13	73

Note: - in the table denotes that the result was not given in the related work.

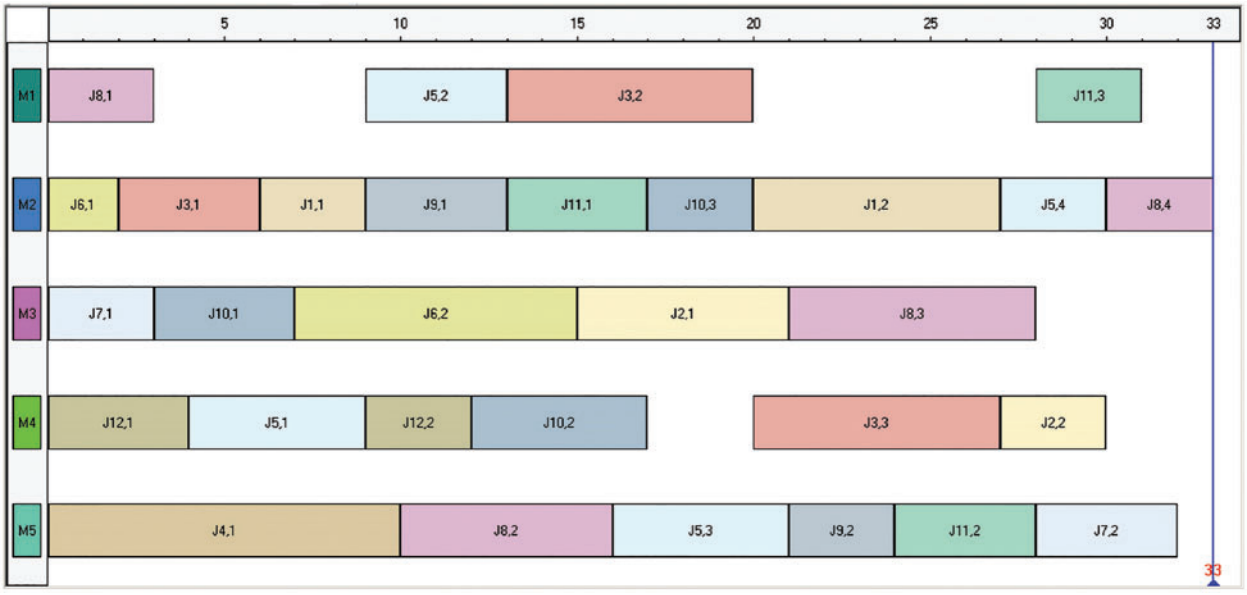


Figure 9: Gantt chart of one solution for problem 12 × 5 (33, 33, 137)

5.1.3 Experiment 3

This experimental problem is from [36], which is the famous benchmark instance for the single objective FJS. This data contains 10 problems. Table 5 is the comparison with the other method, which also shows that the proposed method finds several dominant solutions.

Table 5: The experimental results of problem MK01-MK10

	Makespan	MMW	TWM	Makespan	MMW	TWM
Problem	MK01 (10 × 6)			MK02 (10 × 6)		
Efficient search [26]	*42*	*42*	*162*	*28*	*28*	*155*
HGA [33]	40	36	167	26	26	151
HTSA [37]	40	36	167	26	26	151
Proposed method	40	37	165	31	31	141
	42	38	160	26	26	151
	46	46	153			
Problem	MK03 (15 × 8)			MK04 (15 × 8)		
Efficient search [26]	*204*	*204*	*852*	*68*	*67*	*352*
HGA [33]	204	204	850	60	60	375
HTSA [37]	*204*	*204*	*852*	61	61	366
Proposed method	204	204	850	65	63	349
				66	66	345
Problem	MK05 (15 × 4)			MK06 (10 × 10)		
Efficient search [26]	*177*	*177*	*702*	*75*	*67*	*431*
HGA [33]	172	172	687	58	56	427
HTSA [37]	172	172	687	65	62	398
Proposed method	173	173	685	61	55	427
Problem	MK07 (20 × 5)			MK08 (20 × 10)		
Efficient search [26]	*150*	*150*	*717*	523	523	2524
HGA [33]	139	139	693	523	523	2524
HTSA [37]	*140*	*140*	*695*	523	523	2524
Proposed method	140	140	686	523	523	2524
	139	139	693	526	524	2521
Problem	MK09 (20 × 10)			MK10 (20 × 15)		
Efficient search [26]	*311*	*299*	*2374*	227	221	1989
HGA [33]	307	299	2312	197	197	2029
HTSA [37]	310	301	2294	*214*	*210*	*2053*
Proposed method	328	316	2276	200	199	2020
				240	214	1953

5.1.4 Experiment 4

This experimental problem is from [38], which is another famous benchmark instance for the single objective FJS and is very hard to solve. The data contain 18 problems, and the range of the problems is from 8 × 5 to 20 × 10. Table 6 shows the experimental results. The results indicate that the proposed method can address the multi-objective FJS well.

Table 6: Experimental results of problem 01a–18a

Problem	Makespan	MMW	TWM
01a (10 × 5)	2528	2505	11137
02a (10 × 5)	2236	2234	11137
03a (10 × 5)	2232	2232	11137
04a (10 × 5)	2510	2503	11088
05a (10 × 5)	2228	2222	11054
06a (10 × 5)	2219	2219	11037
07a (15 × 8)	2301	2288	16485
08a (15 × 8)	2082	2072	16485
09a (15 × 8)	2084	2072	16485
10a (15 × 8)	2295	2276	16536
11a (15 × 8)	2089	2077	16439
12a (15 × 8)	2045	2045	16202
13a (20 × 10)	2270	2253	21610
14a (20 × 10)	2173	2172	21610
15a (20 × 10)	2176	2174	21610
16a (20 × 10)	2255	2239	21579
17a (20 × 10)	2159	2159	21407
18a (20 × 10)	2148	2147	21374

5.1.5 Experiment 5

This experimental problem is from [39], which is another famous benchmark instance for the single objective FJSP and is very hard to solve. The experiment contains 21 problems and Table 7 presents the experimental results. The experimental results indicate the proposed method has significant improvement.

Table 7: The results of Experiment 5

Problem	Makespan	MMW	TWM
mt10c1 (10 × 11)	928	631	5109
mt10cc (10 × 12)	910	631	5109
mt10x (10 × 11)	918	556	5109
mt10xx (10 × 12)	918	556	5109

(Continued)

Table 7 (continued)

Problem	Makespan	MMW	TWM
mt10xxx (10 × 13)	918	556	5109
mt10xy (10 × 12)	906	548	5109
mt10xyz (10 × 13)	851	534	5109
setb4c9 (15 × 11)	914	857	7727
setb4cc (15 × 12)	916	857	7727
setb4x (15 × 11)	925	846	7727
setb4xx (15 × 12)	925	846	7727
setb4xxx (15 × 13)	925	846	7727
setb4xy (15 × 12)	916	845	7727
setb4xyz (15 × 13)	905	838	7727
seti5c12 (15 × 16)	1174	1027	11472
seti5cc (15 × 17)	1136	888	11472
seti5x (15 × 16)	1204	938	11472
seti5xx (15 × 17)	1199	938	11472
seti5xxx (15 × 18)	1199	938	11472
seti5xy (15 × 17)	1136	888	11472
seti5xyz (15 × 18)	1126	835	11472

5.1.6 Experiment 6

The experimental problems are built by the authors. The range of the problems involves 10 × 8 and 16 × 8, which are shown in Table 8. The data of problem 16 × 8 are shown in Table 9. Table 10 shows the experimental results. From Table 10, the proposed method provides several Pareto solutions for each problem.

Table 8: The data of problem 10 × 8

Job	Operations	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8
1	$O_{1,1}$	5	4	5	7	8	6	4	4
	$O_{1,2}$	3	3	3	4	3	2	5	2
	$O_{1,3}$	7	6	8	6	5	7	7	8
2	$O_{2,1}$	2	1	3	2	1	1	2	3
	$O_{2,2}$	8	9	8	7	6	9	7	6
	$O_{2,3}$	3	3	2	3	4	5	4	3
3	$O_{3,1}$	7	5	8	9	6	8	5	7
	$O_{3,2}$	5	4	6	4	5	6	3	3
	$O_{3,3}$	7	5	6	8	5	6	4	5
4	$O_{4,1}$	2	3	1	3	2	1	2	2
	$O_{4,2}$	4	5	6	4	4	6	5	4
	$O_{4,3}$	8	5	8	6	7	6	5	6

(Continued)

Table 8 (continued)

Job	Operations	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8
5	$O_{5,1}$	6	7	8	6	7	8	5	6
	$O_{5,2}$	3	4	5	4	6	4	3	5
	$O_{5,3}$	9	10	9	8	9	11	9	10
	$O_{5,4}$	8	9	7	8	9	9	8	9
6	$O_{6,1}$	10	11	12	9	8	9	7	10
	$O_{6,2}$	7	6	7	8	5	9	6	5
	$O_{6,3}$	12	13	10	9	8	9	7	7
	$O_{6,4}$	5	6	4	6	4	5	7	4
7	$O_{7,1}$	3	4	5	3	4	2	3	4
	$O_{7,2}$	4	5	6	3	4	3	5	4
	$O_{7,3}$	8	5	6	8	9	10	6	7
	$O_{7,4}$	1	2	3	2	1	1	2	1
8	$O_{8,1}$	3	6	4	7	5	7	5	6
	O_{82}	2	3	2	4	5	1	1	2
	O_{83}	3	3	3	2	3	4	4	2
	O_{84}	9	10	12	11	13	12	10	11
9	$O_{9,1}$	9	8	9	10	14	12	9	10
	$O_{9,2}$	2	2	1	3	2	1	1	1
	$O_{9,3}$	5	6	4	6	4	6	5	4
	$O_{9,4}$	9	8	9	8	7	10	9	12
	$O_{9,5}$	2	1	3	2	2	2	1	3
10	$O_{10,1}$	5	4	6	7	4	5	6	4
	$O_{10,2}$	7	8	6	9	7	6	6	7
	$O_{10,3}$	4	4	4	3	5	6	5	5
	$O_{10,4}$	3	4	2	4	2	2	1	3
	$O_{10,5}$	5	6	7	5	4	5	6	5

Table 9: The data of problem 16×8

Job	Operations	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8
1	$O_{1,1}$	6	7	5	6	8	5	6	8
	$O_{1,2}$	8	7	4	5	7	4	5	7
	$O_{1,3}$	3	5	6	3	5	6	5	3
2	$O_{2,1}$	8	6	5	8	5	6	7	4
	$O_{2,2}$	2	3	1	1	2	2	1	1
	$O_{2,3}$	10	12	11	13	12	14	9	10
3	$O_{3,1}$	3	3	4	2	1	1	2	1
	$O_{3,2}$	8	7	9	7	6	9	7	7
	$O_{3,3}$	6	6	5	3	4	6	6	5
4	$O_{4,1}$	10	9	8	9	8	7	6	9

(Continued)

Table 9 (continued)

Job	Operations	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8
	$O_{4,2}$	1	1	2	3	1	2	3	1
	$O_{4,3}$	3	4	3	4	2	2	3	4
5	$O_{5,1}$	7	6	9	6	9	9	8	8
	$O_{5,2}$	5	6	4	6	6	3	3	4
	$O_{5,3}$	6	5	3	7	3	5	4	3
6	$O_{6,1}$	2	1	2	3	4	2	1	3
	$O_{6,2}$	9	8	10	11	12	8	9	7
	$O_{6,3}$	4	4	5	3	5	3	4	3
7	$O_{7,1}$	6	5	7	8	5	7	6	5
	$O_{7,2}$	5	4	6	7	5	4	7	5
	$O_{7,3}$	5	4	3	6	4	4	3	5
	$O_{7,4}$	10	11	9	8	10	9	7	8
8	$O_{8,1}$	10	11	13	12	10	11	14	13
	$O_{8,2}$	2	3	3	2	1	2	1	1
	$O_{8,3}$	2	3	4	2	5	5	4	3
	$O_{8,4}$	1	2	1	2	3	4	2	1
9	$O_{9,1}$	10	7	8	9	7	8	7	8
	$O_{9,2}$	10	12	13	15	12	10	11	12
	$O_{9,3}$	2	3	1	3	2	1	2	1
	$O_{9,4}$	6	7	5	7	6	5	6	6
10	$O_{10,1}$	3	4	5	3	4	5	3	3
	$O_{10,2}$	2	1	3	2	1	2	2	2
	$O_{10,3}$	9	8	7	9	7	10	7	8
	$O_{10,4}$	4	3	6	4	5	3	6	7
11	$O_{11,1}$	11	12	15	16	17	14	15	13
	$O_{11,2}$	2	1	1	2	3	4	2	1
	$O_{11,3}$	5	4	5	4	6	4	6	4
	$O_{11,4}$	8	9	8	9	8	9	7	7
12	$O_{12,1}$	7	8	5	4	9	10	11	8
	$O_{12,2}$	4	6	7	3	7	5	3	7
	$O_{12,3}$	2	2	2	1	3	4	1	1
	$O_{12,4}$	3	4	3	2	4	2	4	2
13	$O_{13,1}$	6	5	4	3	5	7	4	5
	$O_{13,2}$	4	6	5	7	4	5	5	7
	$O_{13,3}$	4	6	8	5	4	5	4	6
	$O_{13,4}$	2	3	4	5	1	3	1	1
	$O_{13,5}$	7	8	5	5	6	5	6	5
14	$O_{14,1}$	10	11	12	13	14	15	16	10
	$O_{14,2}$	2	3	4	1	5	1	3	1
	$O_{14,3}$	4	4	6	7	2	4	7	2
	$O_{14,4}$	3	5	6	2	6	6	2	5

(Continued)

Table 9 (continued)

Job	Operations	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8
	$O_{14,5}$	4	2	2	3	3	2	3	3
15	$O_{15,1}$	5	7	7	5	8	5	5	4
	$O_{15,2}$	1	2	1	2	3	1	4	1
	$O_{15,3}$	4	5	3	5	3	6	3	3
	$O_{15,4}$	10	9	8	7	8	9	7	8
	$O_{15,5}$	13	12	12	11	13	13	14	15
16	$O_{16,1}$	9	8	10	8	7	8	7	9
	$O_{16,2}$	10	11	13	12	10	11	9	12
	$O_{16,3}$	9	8	7	9	10	11	9	9
	$O_{16,4}$	2	1	1	3	2	4	2	3
	$O_{16,5}$	4	3	2	5	6	3	4	5

Table 10: The experimental results of problem 10×8 and problem 16×8

	Makespan	MMW	TWM	Makespan	MMW	TWM
Problem		10×8			16×8	
Proposed method	28	24	153	33	33	259
	24	21	160	35	33	257

5.2 Discussion

In the experimental results, the proposed method finds some dominant solutions of the other methods. These results suggest that the proposed method addresses the MOFJS well. The main reasons for the good performance are because the OS and MS are quite different, and the integrated encoding might cause solution space intricacy and complexity. The complex space might prevent the search ability of the existing methods. Moreover, the proposed method evolves OS and MS separately, which can improve the overall searching ability. With this improvement, the proposed method achieves the more effective results. Furthermore, the proposed method uses PAS to save the solutions, and uses crowd similarity for diversity, which is also effective for the improvement of MOFJS.

6 Conclusions

FJS is very important for production, and a novel collaborative evolutionary algorithm with two-population based on Pareto optimality is proposed for MOFJS. The experimental results suggest that the proposed method is feasible for improving the solution of MOFJS, and several dominant solutions are found by the proposed method, which achieved significant improvement. The main contributions of this paper are as follows:

- With the problem features of MOFJS, a collaborative evolutionary method is designed. The proposed method reflects the essential feature of MOFJS, and the PAS is used to save the Pareto solutions. The comparison demonstrates that the proposed method can address MOFJS successfully with a promising result.

- The proposed method combines the merits of collaborative optimization and EA. It uses crowd similarity to guarantee diversity. It provides a novel way to solve MO problems by containing several sub-problems. The experimental results of the multi-objective FJSP show that the proposed method may solve these problems effectively.

Funding Statement: This research work is the Key R&D Program of Hubei Province under Grant No. 2021AAB001, and National Natural Science Foundation of China under Grant No. U21B2029.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

1. Zhou, J., Li, P., Zhou, Y., Wang, B., Zang, J. et al. (2018). Toward new-generation intelligent manufacturing. *Engineering, 4(1),* 11–20. https://doi.org/10.1016/j.eng.2018.01.002
2. Wang, J., Wang, L. (2022). A cooperative memetic algorithm with feedback for the energy-aware distributed flow-shops with flexible assembly scheduling. *Computers & Industrial Engineering, 168(4),* 108126. https://doi.org/10.1016/j.cie.2022.108126
3. Liu, Q., Li, X., Gao, L. (2021). A novel MILP model based on the topology of a network graph for process planning in an intelligent manufacturing system. *Engineering, 7(6),* 807–817. https://doi.org/10.1016/j.eng.2021.04.011
4. Gao, Y., Gao, L., Li, X. (2023). A hierarchical training-convolutional neural network with feature alignment for steel surface defect recognition. *Robotics and Computer-Integrated Manufacturing, 81,* 102507. https://doi.org/10.1016/j.rcim.2022.102507
5. Gao, Y., Gao, L., Li, X., Yan, X. (2020). A semi-supervised convolutional neural network-based method for steel surface defect recognition. *Robotics and Computer-Integrated Manufacturing, 61(1),* 101825. https://doi.org/10.1016/j.rcim.2019.101825
6. Brucker, P., Schlie, R. (1990). Job-shop scheduling with multi-purpose machines. *Computing, 45(4),* 369–375. https://doi.org/10.1007/BF02238804
7. Sun, J., Zhang, G., Lu, J., Zhang, W. (2021). A hybrid many-objective evolutionary algorithm for flexible job-shop scheduling problem with transportation and setup times. *Computers & Operations Research, 132(3),* 105263. https://doi.org/10.1016/j.cor.2021.105263
8. Gu, X. (2021). Application research for multiobjective low-carbon flexible job-shop scheduling problem based on hybrid artificial bee colony algorithm. *IEEE Access, 9,* 135899–135914. https://doi.org/10.1109/ACCESS.2021.3117270
9. Zhang, P., Song, S., Niu, S., Zhang, R. (2021). A hybrid artificial immune-simulated annealing algorithm for multiroute job shop scheduling problem with continuous limited output buffers. *IEEE Transactions on Cybernetics, 52(11),* 12112–12125. https://doi.org/10.1109/TCYB.2021.3081805
10. Park, J. S., Ng, H. Y., Chua, T. J., Ng, Y. T., Kim, J. W. (2021). Unified genetic algorithm approach for solving flexible job-shop scheduling problem. *Applied Sciences, 11(14),* 6454. https://doi.org/10.3390/app11146454
11. Meng, L., Zhang, C., Ren, Y., Zhang, B., Lv, C. (2020). Mixed-integer linear programming and constraint programming formulations for solving distributed flexible job shop scheduling problem. *Computers & Industrial Engineering, 142(19),* 106347. https://doi.org/10.1016/j.cie.2020.106347
12. Alvarez-Valdes, R., Fuertes, A., Tamarit, J. M., Giménez, G., Ramos, R. (2005). A heuristic to schedule flexible job-shop in a glass factory. *European Journal of Operational Research, 165(2),* 525–534. https://doi.org/10.1016/j.ejor.2004.04.020

13. Tay, J. C., Ho, N. B. (2008). Evolving dispatching rules using genetic programming for solving multi-objective flexible job-shop problems. *Computers & Industrial Engineering, 54(3),* 453–473. https://doi.org/10.1016/j.cie.2007.08.008
14. Baykasoğlu, A., Özbakır, L. (2010). Analyzing the effect of dispatching rules on the scheduling performance through grammar based flexible scheduling system. *International Journal of Production Economics, 124(2),* 369–381. https://doi.org/10.1016/j.ijpe.2009.11.032
15. Rajkumar, M., Asokan, P., Anilkumar, N., Page, T. (2011). A GRASP algorithm for flexible job-shop scheduling problem with limited resource constraints. *International Journal of Production Research, 49(8),* 2409–2423. https://doi.org/10.1080/00207541003709544
16. Caldeira, R. H., Gnanavelbabu, A. (2021). A Pareto based discrete Jaya algorithm for multi-objective flexible job shop scheduling problem. *Expert Systems with Applications, 170(1),* 114567. https://doi.org/10.1016/j.eswa.2021.114567
17. Li, K., Deng, Q., Zhang, L., Fan, Q., Gong, G. et al. (2021). An effective MCTS-based algorithm for minimizing makespan in dynamic flexible job shop scheduling problem. *Computers & Industrial Engineering, 155(1),* 107211. https://doi.org/10.1016/j.cie.2021.107211
18. Fattahi, P., Saidi Mehrabad, M., Jolai, F. (2007). Mathematical modeling and heuristic approaches to flexible job shop scheduling problems. *Journal Intelligent Manufacturing, 18(3),* 331–342. https://doi.org/10.1007/s10845-007-0026-8
19. Lei, D., Yuan, Y., Cai, J. (2021). An improved artificial bee colony for multi-objective distributed unrelated parallel machine scheduling. *International Journal of Production Research, 59(17),* 5259–5271. https://doi.org/10.1080/00207543.2020.1775911
20. Xia, W., Wu, Z. (2005). An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems. *Computers & Industrial Engineering, 48(2),* 409–425. https://doi.org/10.1016/j.cie.2005.01.018
21. Lei, D., Yuan, Y., Cai, J. (2021). An improved artificial bee colony for multi-objective distributed unrelated parallel machine scheduling. *International Journal of Production Research, 59(17),* 5259–5271. https://doi.org/10.1080/00207543.2020.1775911
22. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation, 6(2),* 182–197. https://doi.org/10.1109/4235.996017
23. Gao, L., Peng, C., Zhou, C., Li, P. (2006). Solving flexible job-shop scheduling problem using general particle swarm optimization. *Proceedings of the 36th CIE Conference on Computers & Industrial Engineering*, pp. 3018–3027. Taipei, China.
24. Kacem, I., Hammadi, S., Borne, P. (2002). Approach by localization and multiobjective evolutionary optimization for flexible job-shop scheduling problems. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 32(1),* 1–13. https://doi.org/10.1109/TSMCC.2002.1009117
25. Zhang, G., Shao, X., Li, P., Gao, L. (2009). An effective hybrid particle swarm optimization algorithm for multi-objective flexible job-shop scheduling problem. *Computers & Industrial Engineering, 56(4),* 1309–1318. https://doi.org/10.1016/j.cie.2008.07.021
26. Xing, L., Chen, Y., Yang, K. (2009). An efficient search method for multi-objective flexible job shop scheduling problems. *Journal Intelligent Manufacturing, 20(3),* 283–293. https://doi.org/10.1007/s10845-008-0216-z
27. Xing, L., Chen, Y., Yang, K. (2009). Multi-objective flexible job shop schedule: Design and evaluation by simulation modeling. *Applied Soft Computing, 9(1),* 362–376. https://doi.org/10.1016/j.asoc.2008.04.013
28. Wang, X., Gao, L., Zhang, C., Shao, X. (2010). A multi-objective genetic algorithm based on immune and entropy principle for flexible job-shop scheduling problem. *International Journal Advanced Manufacturing Technology, 51(5–8),* 757–767. https://doi.org/10.1007/s00170-010-2642-2

29. Li, J., Pan, Q., Gao, K. (2011). Pareto-based discrete artificial bee colony algorithm for multi-objective flexible job shop scheduling problems. *International Journal Advanced Manufacturing Technology, 55(9–12),* 1159–1169. https://doi.org/10.1007/s00170-010-3140-2
30. Ho, N., Tay, J. (2008). Solving multiple-objective flexible job shop problems by evolution and local search. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 38(5),* 674–685. https://doi.org/10.1109/TSMCC.2008.923888
31. Xia, W. J., Wu, Z. M. (2005). An effective hybrid optimization approach for multi-objective flexible job shop scheduling problems. *Computers & Industrial Engineering, 48(2),* 409–425. https://doi.org/10.1016/j.cie.2005.01.018
32. Zhang, H., Gen, M. (2005). Multistage-based genetic algorithm for flexible job-shop scheduling problem. *Journal of Complexity International, 11,* 223–232.
33. Gao, J., Gen, M., Sun, L., Zhao, X. (2007). A hybrid of genetic algorithm and bottleneck shifting for multiobjective flexible job shop scheduling problems. *Computers & Industrial Engineering, 53(1),* 149–162. https://doi.org/10.1016/j.cie.2007.04.010
34. Xing, L. N., Chen, Y. W., Yang, K. W. (2009). An efficient search method for multi-objective flexible job shop scheduling problems. *Journal of Intelligent Manufacturing, 20(3),* 283–293. https://doi.org/10.1007/s10845-008-0216-z
35. Moslehi, G., Mahnam, M. (2011). A Pareto approach to multi-objective flexible job-shop scheduling problem using particle swarm optimization and local search. *International Journal of Production Economics, 129(1),* 14–22. https://doi.org/10.1016/j.ijpe.2010.08.004
36. Brandimarte, P. (1993). Routing and scheduling in a flexible job shop by taboo search. *Annals of Operations Research, 41(3),* 157–183. https://doi.org/10.1007/BF02023073
37. Li, J., Pan, Q., Liang, Y. (2010). An effective hybrid tabu search algorithm for multi-objective flexible job-shop scheduling problems. *Computers & Industrial Engineering, 59(4),* 647–662. https://doi.org/10.1016/j.cie.2010.07.014
38. Peres, S. D., Paulli, J. (1997). An integrated approach for modeling and solving the general multiprocessor job shop scheduling using tabu search. *Annals of Operations Research, 70,* 281–306. https://doi.org/10.1023/A:1018930406487
39. Barnes, J. (1996). *Flexible job shop scheduling by tabu search (Ph.D. Thesis)*. University of Texas at Austin.