

Received March 3, 2021, accepted March 30, 2021, date of publication April 8, 2021, date of current version April 19, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3071729

# A Hybrid Multi-Objective Teaching-Learning Based Optimization for Scheduling Problem of Hybrid Flow Shop With Unrelated Parallel Machine

CUNLI SONG 

School of Software, Dalian Jiaotong University, Dalian 116052, China  
Artificial Intelligence Key Laboratory of Sichuan Province, Zigong 643000, China  
e-mail: scunli@163.com

This work was supported in part by the Open Fund Projects of Artificial Intelligence Key Laboratory of Sichuan Province under Grant 2020RYJ04, in part by the Liaoning Provincial Natural Science Foundation under Grant 20180551020, and in part by the Liaoning Educational Committee Program under Grant JDL2019011.

**ABSTRACT** Efficient scheduling benefits productivity promotion, energy savings and the customer's satisfaction. In recent years, with a growing concern about the energy saving and environmental impact, energy oriented scheduling is going to be a hot issue for sustainable manufacturing. In this study, we investigate an energy-oriented scheduling problem deriving from the hybrid flow shop with unrelated parallel machine. First, we formulate the scheduling problem with a mixed integer linear programming (MILP) model, which considers two objectives including minimizing the completion time and energy consumption. Second, a hybrid multi-objective teaching-learning based optimization (HMOTLBO) algorithm based on decomposition is proposed. In the proposed HMOTLBO, a new solution presentation and five decoding rules are designed for mining the optimal solution. To reduce the standby energy consumption and turning on/off energy consumption, a greedy shifting algorithm is developed without changing the completion time of a scheduling. To improve the converge speed of the algorithm, a weight matching strategy is designed to avoid randomly matching weight vectors with students. To enhance the exploration and exploitation capacities of the algorithm, A teaching operator based on crossover and a self-learning operator based on a variable neighborhood search(VNS) are proposed. Finally, fourth different experiments are performed on 15 cases, the comparison result verified the effectiveness and the superiority of the proposed algorithm.


**INDEX TERMS** Hybrid flow shop scheduling, teaching and learning based optimization, multi-objective, makespan, energy consumption.

## I. INTRODUCTION

In manufacturing shops, production scheduling plays an important role on achieving one or more manufacturing optimal objectives, such as minimizing the maximum completion time (makespan), mean flow time, earliness and tardiness. Flow-shop scheduling problem(FSP) is one of classical production scheduling problems and widely exists in chemical [1], metallurgy [2], micro-electronics [3], and so on. Hybrid flow shop with unrelated parallel machines (HFS) is an extension of FSP, but it is more complex than FSP, because at least in one stage there are multiple parallel

machines to choose from to process a job. In 1971, Arthanari and Ramamurthy [4] first studied a two-stage HFS scheduling problem. Since then, three-stage and m-stage HFS scheduling problems are widely studied by scholars. In 1988, Gupta and Jatinder [5] proved that HFS restricted to two processing stages is Non-deterministic Polynomial-time hard(NP-hard).

In recent years, with the environment pollution and the running out of the energy, more and more countries make policy on energy-saving and emission-reduction, high energy consumption enterprises face the risk of elimination. Therefore, saving energy conservation becomes one of objectives that many manufacturers pursue and an important issue for industry. The steel making and continuous casting process in the

The associate editor coordinating the review of this manuscript and approving it for publication was Kuo-Ching Ying .

steel industry is a typical HFS. Since there are many different machines at the same stage can process the same job with different energy consumption, energy efficient scheduling becomes more significant. On the other hand, productivity is related to the efficiency of enterprises. Under this background, it is of great significance to study the HFS scheduling problem to minimize the makespan and energy consumption. The twofold novelties of this paper are given in the following:

(1) To reduce energy consumption, the processing energy consumption, standby energy consumption and turning on/off energy consumption is considered simultaneously and a greedy shifting algorithm is proposed to reduce standby consumption and turning on/off energy consumption during the decoding phase.

(2) A hybrid multi-objective teaching-learning based optimization is proposed to solve the scheduling problem of HFS with unrelated machine. Three contributions of the proposed HMOTLBO lie in: (a) a new solution presentation method that support five different decoding rules is proposed in this paper; (b) at the preliminary stage of the algorithm, a weight matching algorithm is proposed to avoid the random matching between students and weight vectors and improve the convergence speed of the algorithm; (c) To enhance the exploration and exploitation capacities of the algorithm, a teaching operator based on crossover and a self-learning operator based on variable neighborhood search(VNS) are proposed.

The rest of this paper is organized as follows: Section II reports the relevant literature review. Section III formulates the HFS scheduling problem. Then, in section IV, the algorithm HMOTLBO is proposed. Section V analyzes the experimental results and compares them with those of other algorithms in the literature to evaluate the performance of the proposed algorithm. Finally, the last section presents the conclusions of our work.

## II. LITERATURE REVIEW

### A. SINGLE OBJECTIVE HFS SCHEDULING PROBLEMS

For HFS scheduling problem, most literature focus on the objective of makespan, such as Engin and Deyen [6] proposed an artificial immune system, Alaykran *et al.* [7] and Khalouli *et al.* [8] proposed an ant colony optimization (ACO) respectively. Liao *et al.* [9] proposed a particle swarm optimization(PSO). Li and Pan [10] proposed a hybrid artificial bee colony algorithm. Song [11] proposed an improved greedy genetical algorithm(IGA). The second hot objective is to minimize the sum of the earliness and tardiness costs, such as Han *et al.* [12] proposed a differential evolution (DE) algorithm based on multiple decision rules. M'Hallah and Alhajraf [13] proposed an Ant colony systems. Fei and Hui [14] proposed an allied genetic algorithm(GA), ect.

### B. MULTI-OBJECTIVE HFS SCHEDULING PROBLEMS

In recent years, with more and more manufacturing enterprises pursue multiple production objectives and the

proposition of multi-objective optimization algorithms, such as NSAG [15], NSGAI [16], MOEA/D[17], SPEA [18] and SPEA2 [19], more and more researches about HFS scheduling problem focus on the multi-objective. To minimize makespan and mean flow time, Solano-Charris *et al.* [20] presented an ACO; Marichelvam *et al.* [21] proposed a discrete firefly algorithm. To minimize makespan and mean tardiness, Mundim and Queiroz [22] and De Siqueira *et al.* [23] developed a variable neighborhood search algorithm(VNS) respectively; Ying *et al.* [24] proposed an iterated pareto greedy algorithm, Tran and Ng [25] proposed a hybrid water flow algorithm; Asefi *et al.* [26] considered the no-wait constraint and proposed a hybrid NSGA-II and VNS algorithm; Cho *et al.* [27] developed a Pareto genetic algorithm while considering reentrant. To minimize the total flow time and the number of tardy jobs, Wang *et al.* [28] developed a NSGAI. To minimize the production cost and maximize the customer's satisfaction, Shahvari and Logendran [29] proposed a tabu search enhanced with path-relinking. For more than two objectives, Zhu-Min [30] proposed a multi-objective master scheduling algorithm to minimize delay penalties, proceeding time, setup cost and holding cost; Lu *et al.* [31] proposed a novel multi-objective cellular grey wolf optimizer (MOCGWO) to minimize noise pollution, makespan and energy consumption, etc.

### C. ENERGY ORIENTED HFS SCHEDULING PROBLEMS

With the increasing awareness of environment protection, more and more industries pay more attention to the energy saving while manufacturing. Thus, more and more researchers begin considering the energy efficiency scheduling. Gao *et al.* [32] reviewed literature about production scheduling for intelligent manufacturing systems with the energy-related constraints and objectives in the past 10 years. To minimize the total electricity cost, Che *et al.* [33], [34] proposed an efficient greedy insertion heuristic while production scheduling under time-of-use electricity tariffs. Wu and Sun [35] developed a non-dominated sorted genetic algorithm to solve the flexible job shop scheduling (FJS) problem considering the makespan, the energy consumption and the numbers of turning-on/off machines simultaneously. Wang *et al.* [36] proposed a two stages optimization algorithm for the same problem. Meng *et al.* [37] integrate the process planning and scheduling of FJS problem, build a mixed integer linear programming(MILP), and solve it by CPLEX. Dai *et al.* [38] proposed a modified genetic algorithm to minimize the makespan and the total energy consumption in job shop horizons. For HFS scheduling problem, Luo *et al.* [39] presented a multi-objective ant colony optimization considering production efficiency and electric power cost. Dai *et al.* [40] presented a genetic simulated annealing algorithm to minimize the makespan and the total energy consumption. Zhang *et al.* [41] developed a time-indexed integer programming formulation to minimize the electricity cost and reduce CO2 emissions. Rao *et al.* [42] proposed

an improved particle swarm optimization for the dynamic scheduling. Li *et al.* [43] proposed an energy-aware multi-objective optimization algorithm (EA-MOA) consideration of the setup energy consumption.

**D. MOTIVATIONS**

In the actual HFS with unrelated parallel machines environment, there are multiple machines that can be selected to process a job in some stages, and the processing time and energy consumption of different machines are often different. At the same time, the machines can have three different states, such as processing state, standby state, and shutdown restart state. The unit energy consumption of a machine is different in different states. Therefore, it is of great significance to consider both the processing efficiency and the energy consumption of the machines when scheduling HFS problem. From the above review, there is less literature on minimizing both the makespan and the energy consumption for the HFS with unrelated machine scheduling problems, and there is no published literature in which the processing energy consumption, the standby energy consumption and the turning on/off energy consumption are considered simultaneously. Therefore, in our study, we consider to minimize the energy consumption and makespan simultaneously of the HFS scheduling problem.

**III. PROBLEM FORMULATION**

**A. PROBLEM DESCRIPTION**

In a HFS scheduling problem with unrelated machines, there are  $n$  jobs to be processed on  $s$  stages in a predefined same order. Jobs can be processed on one of several machines at each stage. Machines at each stage may have different speed and power. There are two tasks to schedule the HFS problem with unrelated machine. One task is to determine the jobs' processing sequence at each stage and the other is to assign machine for each job. Consequently, different processing routes may lead to different makespan and different energy consumption. As the two objectives are usually in conflict with each other, to solve this problem, some assumptions are made here, such as:

- All machines and jobs are available at time zero.
- Preemption is not allowed, that is, no task can be interrupted before the completion of its current operation.
- There is infinite buffer or storage between any two consecutive stages.
- Each machine can process at most one operation at a time and each operation is processed on at most one machine at a time.
- All operations are assigned strictly to one machine at each stage.

**B. NOTATIONS**

In order to analyse the problem easily, some notations are defined as follows:

Indexes:

- $i$  index of jobs, and  $i = 1, 2, \dots, n$ .
- $j$  index of stages, and  $j = 1, 2, \dots, s$ .
- $l$  index of jobs on one machine, and  $0 \leq l \leq n$ .
- $k$  index of machine at one stage, and  $k = 1, 2, \dots, m_j$ .

Parameters:

- $n$  total number of jobs.
- $s$  total number of the processing stages.
- $m_j$  total number of the machines at stage  $j$ , and  $m_j \geq 1$ .
- $T_{i,j,k}$  processing time that job  $i$  on machine  $k$  at stage  $j$ .
- $PE_{i,j,k}$  processing energy consumption per unit time of job  $i$  on machine  $k$  at stage  $j$ .
- $SB_{j,k}$  standby energy consumption per unit time of machine  $k$  at stage  $j$ .
- $SW_{j,k}$  turning on/off energy consumption per one time of machine  $k$  at stage  $j$ .

Decisions variables:

- $X_{i,j,k}$  if the processing of job  $i$  at stage  $j$  is assigned to machine  $k$ , then  $X_{i,j,k} = 1$ , otherwise,  $X_{i,j,k} = 0$ .
- $P_{i,j}$  complete time of job  $i$  at stage  $j$ .
- $B_{j,k,l}$  start time of the  $l$ -th job processed on machine  $k$  at stage  $j$ .
- $C_{j,k,l}$  complete time of the  $l$ -th job processed on machine  $k$  at stage  $j$ .
- $Jnumber_{j,k}$  total number of jobs that assigned to machine  $k$  at stage  $j$ .
- $Job_{j,k,l}$  job index of the  $l$ -th job on machine  $k$  at stage  $j$ ,  $l = 1, 2, \dots, Jnumber_{j,k}$ .
- $Ftime_{j,k,l}$  standby time between the  $l$ -th job on machine  $k$  at stage  $j$  and its predecessor,  $l = 2, \dots, Jnumber_{j,k}$ .
- $switch_{j,k,l}$  if machine  $k$  at stage  $j$  is shut down during the idle time between the  $l$ -th job and its predecessor, then  $switch_{j,k,l} = 1$ ; else  $switch_{j,k,l} = 0$ .
- $Cmax$  the completion time of the last job on the last machine at the last stage.
- $Energy$  total energy consumption.
- $ProcessE$  total processing energy consumption.
- $StandbyE$  total standby energy consumption.
- $SwitchE$  total on/off energy consumption
- $\pi$  a solution representation.

**C. MIXED-INTEGER LINEAR PROGRAMMING MODEL (MILP)**

$$Cmax = \min \{ Cmax(\pi^*) \mid \pi^* \in \pi \} \tag{1}$$

$$Energy = \min \{ energy(\pi^*) \mid \pi^* \in \pi \} \tag{2}$$

Subject to :

$$\sum_{k=1}^{m_j} x_{i,j,k} = 1 \tag{3}$$

$$B_{j,k,l} = \begin{cases} P_{Job_{j,k,l,j-1}}, & l = 1, j \neq 1 \\ C_{j,k,l-1}, & l \neq 1, j = 1 \\ \max \{C_{j,k,l-1}, P_{Job_{j,k,l,j-1}}\}, & l \neq 1, j \neq 1 \\ 0, & l = 1, j = 1 \end{cases} \tag{4}$$

$$C_{j,k,l} = B_{j,k,l} + T_{Job_{j,k,l,j,k}} \tag{5}$$

$$P_{i,j} = C_{j,k,l} = B_{j,k,l} + T_{i,j,k}, i = Job_{j,k,l} \tag{6}$$

$$Cmax(\pi) = \max \{P_{i,j} | i = 1, 2, \dots, N; j = 1, 2, \dots, S\} \tag{7}$$

$$Energy = ProcessE + StandbyE + SwitchE \tag{8}$$

$$ProcessE = \sum_{j=1}^S \sum_i^N \sum_{k=1}^{m_j} X_{i,j,k} \cdot T_{i,j,k} \cdot PE_{j,k} \tag{9}$$

$$StandbyE = \sum_{j=1}^S \sum_{k=1}^{m_j} \sum_{l=2}^{Jnumber_{j,k}} sb_{j,k} \cdot Ftime_{j,k,l} \cdot !switch_{j,k,l} \tag{10}$$

$$SwitchE = sw_{j,k} \cdot \sum_{j=1}^S \sum_{k=1}^{m_j} \sum_{l=1}^{Jnumber_{j,k}} switch_{j,k,l} \tag{11}$$

$$switch_{j,k,l} = \begin{cases} 1, & \text{if } sw_{j,k} < sb_{j,k} \cdot Ftime_{j,k,l} \text{ or } l = 1 \\ 0, & \text{else} \end{cases} \tag{12}$$

$$Ftime_{j,k,l} = B_{j,k,l} - C_{j,k,l-1}, l = 2, \dots, Jnumber_{j,k} \tag{13}$$

$$X_{i,j,k} = \begin{cases} 1, & i \in \{job_{j,k,l} | l = 1, \dots, Jnumber_{j,k}\} \\ 0, & \text{others} \end{cases} \tag{14}$$

$$\sum_{k=1}^{m_j} Jnumber_{j,k} = N \tag{15}$$

Eq(1) is the objective function of the makespan. Eq (2) is the objective function of the energy consumption. Eq(3) constraints that just allocate one machine to process job *i* at stage *j*. Eq (4) and Eq(5) is the formular for computing the start time and completion time of the *l*\_th job on machine *k* at stage *j* respectively. Eq(6) is the formular for computing the completion time of job *i* at state *j*. Eq(7) is the maximum completion time fromular of a scheduling and Eq(8) is the total energy consumption. Eq(9) is the total processing energy consumption formular. Eq(10) is the total standby consumption and Eq(11) is the total turning on/off energy consumption respectively. Eq(12) is the decision variables, the rule is turning off the machine when the standby energy consumption is bigger than a turning on/off energy consumption, else keep the machine in standby mode. Eq(13) is the formular for computing the idle time between two neighbor jobs on a machine. Eq(14)(15) show the distribution of job *i* on machines at stage *j*.

#### IV. THE PROPOSED ALGORITHM HMOTLBO

TLBO [42], proposed by Rao *et al.* in 2011. As a population-based algorithm, the performance of TLBO is not affected by specific parameters and is easy to implement. Since there are few TBLO algorithms is used to solve the multi-objective

HFS scheduling problem, this paper proposed a hybrid multi-objective TLBO algorithm(HMOTLBO). Inspired by MOEA/D, HMOTLBO is implemented based on objective decomposition, and uses an external archive set(EA) to record the non-dominated solutions. More details of the algorithm is introduced in the following subsections.

#### A. SOLUTION REPRESENTATION

For a solution representation of HFS, it must describe the jobs' processing sequence at each stage and the corresponding processing machine assigned at that stage. In this study, a novel encoding method is developed, which is a vector containing  $(s + 1) \times n$  integer elements. The first *n* integer elements is a sequence of integers from 1 to *n* to show the processing sequence of the jobs at the first stage, the processing sequence at other stages is determined by first come, first served. The elements from the  $(n + 1)$ \_th to the  $(n * 2)$ \_th of the vector correspond to the machines assigned to the jobs at the first stage, and the elements from  $(2 \times n + 1)$ \_th to  $(3 \times n)$ \_th of the vector correspond to the machines assigned at the second stage, and so on. To illustrate this encoding method, consider an instance with 3 jobs and 3 stages. Assume there are 2 unrelated machines at the first two stages and 3 at the last stage, that is,  $n = 3, s = 3, M_1 = 2, M_2 = 2$  and  $M_3 = 3$ . A solution representation is given in Figure 1, which means job 2 and 3 is processed by machine 1 and job 1 is processed by machine 2 at the first stage. Given the arrival sequence of jobs at the second stage is 1,2,3. Then job 1 and 3 is processed by machine 2 and job 2 is processed by machine 1 at this stage, etc. The advantage of this encoding method is that it can support 5 decoding methods described in the subsection B.

#### B. DECODING RULES

In order to guide the search direction of the algorithm, five different decoding rules based on different machine assignment rules are proposed: (1) Assign machine to jobs completely according to the solution representation, which is named SI and Figure 1 is an example. (2) When a job is scheduled, assign the machine which can complete the job earlier. If there are more than one machine satisfy this rule, then select the one with the lowest processing energy consumption. This decoding rule ignores the elements of the solution vector from the  $(n + 1)$ \_th to the last, which is named SII. (3) When a job is scheduled, assign the machine with the lowest processing energy consumption to the job. If there are more than one machine satisfy the rule

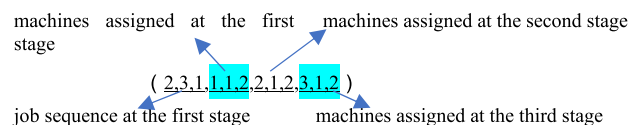


FIGURE 1. Example of one solution presentation.



simultaneously, then select the machine which can complete the job earlier, which is named SIII. Also, this strategy ignores the elements of the solution vector from the  $(n + 1)$ \_th to the last. (4) Hybrid strategy SIV. When one job is to be scheduled, the rules SII or SIII will be used to assign the machine. The selection of these two rules is based on the weight vector of the subproblem by roulette. (5) Hybrid strategy SV. When a solution is decoded, the decoding rule is chosen from SI, SII and SIII according to the probability of 0.9, 0.05, 0.05 respectively, and we got the ratio by orthogonal experiment on case j20c5a3.

### C. GREEDY SHIFTING STRATEGY

During the decoding phase, only processing energy can be considered, more energy can be saved by reducing the machines' restart times, prolonging the restart interval or reducing the machines' standby time. Therefore, to achieve this purpose, a greedy shifting algorithm is proposed without changing the maximum completion time and the processing sequence of the jobs on each machine. The pseudo code is shown in Algorithm 1 and an instance is shown in Figure 2 with gantt chart.

---

#### Algorithm 1 . Greedy Shift Strategy

---

Input: a solution

Output: a solution with same complete time and less energy consumption

##### Step1: //right shifting

//From the last stage to the second stage

**for** (j=s; j<=2; j-)

// For each machine  $k$  at stage  $j$

**for** (k=1; k<=m<sub>j</sub>;k++)

**for each** job on machine  $k$ , from the penultimate to the first

{ **if** there are idle time behind the job, **then**  
 {move the job right as far as possible; }  
 }

##### step2: // left shifting

// From the second stage to the penultimate stage

**for**(j=2; j<=s-1;j++)

// For each machine  $k$  at stage  $j$

**for**(k=1; k<=m<sub>j</sub>;k++)

**for each** blocks on machine  $k$  from the second to the penultimate

**if** move it left can reduce the non-processing energy consumption, **then**  
 { **move** it left as far as possible; }

---

In Figure 2, gantt chart (a) shows the original decoding result of an instance, where, the makespan is 293 and the energy consumption is 9542. Gantt chart (b) shows the right shifting result of (a). Here, the makespan keeps unchanged while the energy consumption is reduced to 9294. To clearly

show the changed part, we use the red dotted line surrounded it. Gantt chart (c) shows the left shifting result of (b) and purple dotted line surrounds the changed part. In figure (c), the energy consumption is reduced to 9278 while the makespan unchanged.

### D. DECOMPOSITION MECHANISM

Inspired by MOEA/D, we convert the multi-objective problem into a series of scalar optimization problems and the weights of each problem can be decided by Eq (16). Here,  $ii$  is the index of the solutions,  $N$  is the total number of the solutions. It is notable that the target of the energy consumption and makespan have different value ranges, for easier tradeoff decisions, it is helpful to normalize their values with Eq (17). Here,  $f_o^{min}$  and  $f_o^{max}$  are the minimal and maximal value of the  $o$ \_th objective respectively. The fitness of the  $ii$ \_th scalar optimization problems is calculated according to Eq (18). Here,  $z_o^*$  is the reference point of the  $o$ \_th objective and is equal to zero in this question.

$$(\lambda_1^{ii}, \lambda_2^{ii}) = \left( \frac{ii-1}{Size-1}, \frac{Size-ii}{Size-1} \right), \quad 1 \leq ii \leq Size \quad (16)$$

$$\bar{f}_o = \frac{f_o - f_o^{min}}{f_o^{max} - f_o^{min}}, \quad 1 \leq o \leq 2 \quad (17)$$

$$\min g(x | \lambda^{ii}, z^*) = \max_{1 \leq o \leq 2} \{\lambda_o^{ii} |\bar{f}_o(x) - z_o^*|\} \quad (18)$$

### E. MATCHING STRATEGY

At the initialization phase of the MOEA/D, the weight vector is randomly allocated to a solution, if a weight vector can't match a solution well, the relationship between them will look like Figure 3. In Figure 3, weight vector  $\lambda^1$  is allocated to solution A,  $\lambda^3$  is allocated to solution B. Obviously, for solution A, the weight vector  $\lambda^5$  or  $\lambda^4$  is more suitable than  $\lambda^1$ . For solution B, weight  $\lambda^1$  and  $\lambda^2$  is more suitable than  $\lambda^3$ . It's like a student who is closer to the target in the east, but we assigned him a target in the west. As a result, the student will spend more time looking for the western target than the eastern target. To overcome this shortcoming of MOEA/D, a strategy of matching the weights and solutions is proposed. The idea of this strategy is to allocate a weight close to the objective of the solution. Algorithm 2 shows the pseudocode.

### F. TEACHING OPERATOR

During the teaching phase, the crossover operator is used to implement the teaching work. The specific crossover process is illustrated in Figure 4 and the pseudocode is provides in Algorithm 3. At this stage, teachers transfer knowledge among students, and strive to improve the score of the whole class. A good teacher will improve students' score greatly. Therefore, teachers will be selected from EA, the external archive set that record the nondominated solutions of the

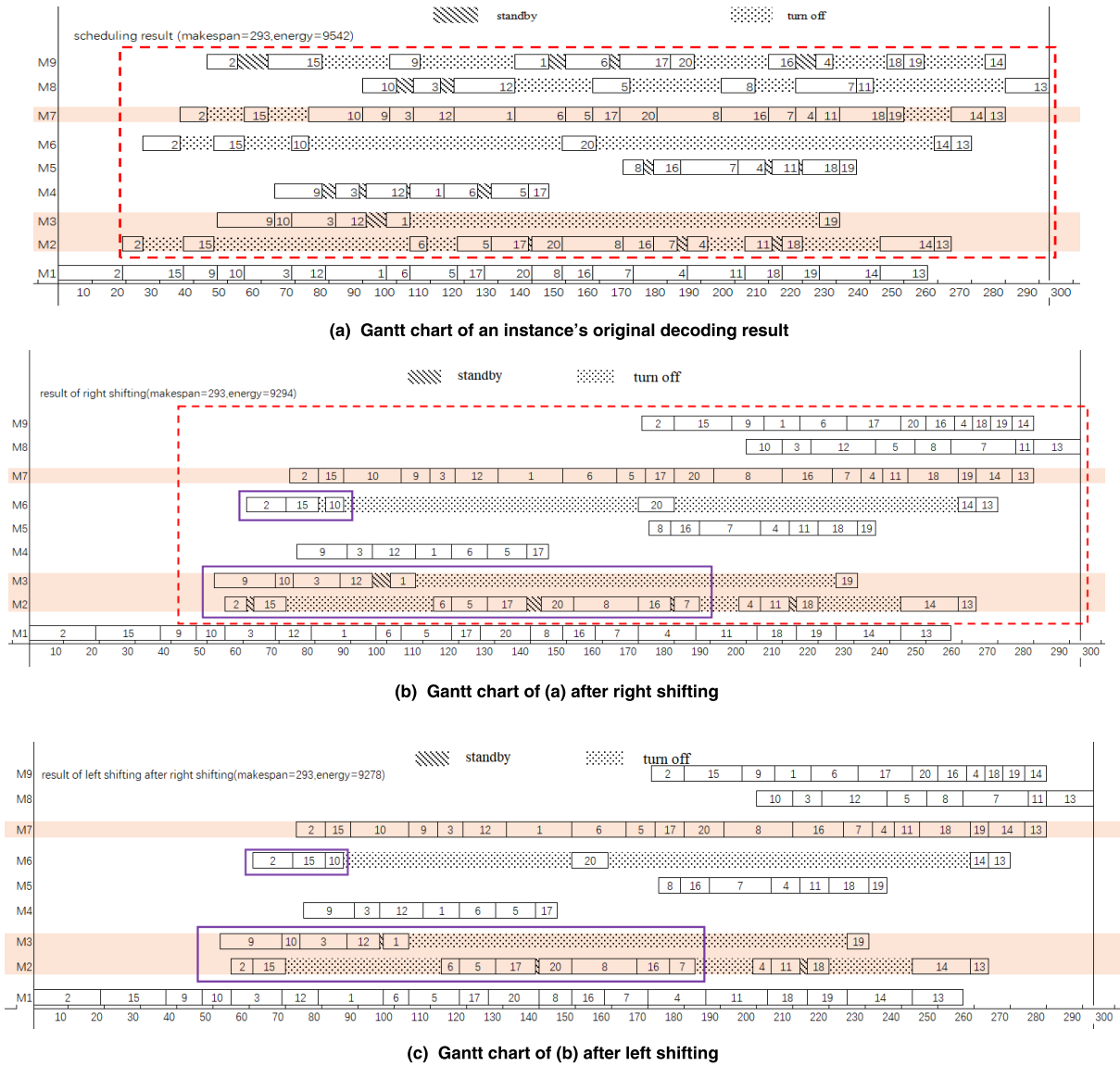


FIGURE 2. Gantt charts of an instance's scheduling result.

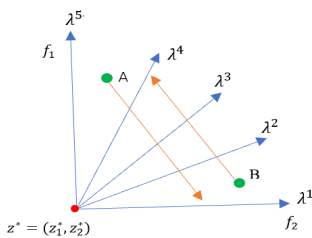


FIGURE 3. Matching randomly between weights and solutions.

problem. Inspired by the actual teaching activities, when the teacher's goal is consistent with the student's goal, students can learn knowledge from teacher faster and better. On the contrary, students may deviate from their learning goal or

**Algorithm 2** Weight Matching

- Input:** The weights vector  $W = \{\lambda^1, \lambda^2, \dots, \lambda^N\}$ , the initial solutions  $Q = \{\pi_1, \pi_2 \dots, \pi_N\}$ ;
- Output:** The solutions and their matched weight;
- Step 1:** Sort the solutions non-ascendingly according to the values of the first objective;
- Step 2:** Sort the weights of vector  $W$  ascendingly according to the values of first weight;
- Step 3:** for( $i=1; i \leq N; i++$ )  
Assign weight  $\lambda^i$  to solution  $\pi_i$ ;

be inefficient. In view of this, a teacher selection algorithm is proposed. When selecting a teacher for a student, the algorithm first calculates the selection probability of each

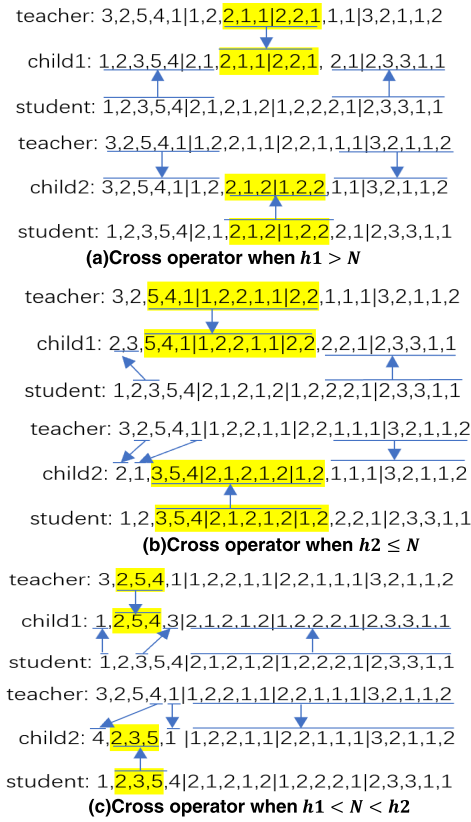


FIGURE 4. Cross operator.

teacher according to the proximity between the teacher’s goal and the student’s goal, then use the roulette strategy select a teacher for a student. The pseudocode is described in Algorithm 4.

$$\theta_o^t = 1 - \frac{f_o^t - f_o^{min}}{f_o^{max} - f_o^{min}} \Big/ \sum_{i=1}^2 \frac{f_i^t - f_i^{min}}{f_i^{max} - f_i^{min}}, \quad t = 1, 2, \dots, NT; \quad o = 1, 2 \quad (19)$$

$$D^t = \sqrt{(\lambda_1^i - \theta_1^t)^2 + (\lambda_2^i - \theta_2^t)^2}, \quad t = 1, 2, \dots, NT \quad (20)$$

$$P^t = D^t \Big/ \sum_{k=1}^{NT} D^k, \quad t = 1, 2, \dots, NT \quad (21)$$

### G. SELF-LEARNING OPERATOR

To speed up the learning process, students tend to study by themselves or learn from his neighbors. In this study, each student and teacher in EA will perform a self-learning operator based on variable neighborhood search(VNS) to complete the learning process. Three neighborhood structures are designed owing to their effectiveness, such as reverse, swap and insert. Algorithm 5 shows the pseudocode of student’s self-learning operator, the teacher’s self-learning operator is similar with Algorithm 5. Since the decoding method SII and SIII ignore the elements of the solution vector from  $(n + 1)_th$  to  $(s + 1) \cdot n_th$ , we will only search

### Algorithm 3 Teaching Operator

**Input:** the selected teacher and the current student;  
**Output:** the updated student and its neighbors;  
**Step1:** randomly generate two integers between 1 and  $s \times n$ , and put the smaller to variable  $h1$  while the other to  $h2$ .  
**Step2:** //Execute cross operation  
**if** ( $h1 > N$ ) **then**  
    **if** (the decoding mode is not SII and SIII) **then**  
        generate two children with the cross method of (a) in Figure 4;  
    **else**  
        goto Step1;  
    **else if** ( $h2 \leq N$ ) **then**  
        generate two children with the cross method of (b) in Figure 4;  
    **else**  
        generate two children with the cross method of (c) in Figure 4;  
**endif**  
**Step3:** evaluate the fitness of the two children.  
**Step4:** // update  
    choose the best solution from the two children and if it’s better than the current student, then update the student;  
    update the neighbors of the current student with the two children;

### Algorithm 4 Teacher Selection

**Input:** archive set EA, element number NT of set EA. the weight vector of the  $i_{th}$  student  $(\lambda_1^i, \lambda_2^i)$ ;  
**Output:** the teacher selected for the  $i_{th}$  student;  
**Step 1:** **for**( $t=1;t \leq NT;t++$ )  
    { compute the  $t_{th}$  teacher’s relative weight vector with Eq(19);  
    compute the  $i_{th}$  teacher’s Euclidean distances with the current student with Eq(20);  
**Step2:** **for**( $t=1;t \leq NT;t++$ )  
    compute the selection probability of the  $t_{th}$  teacher in set EA with Eq (21);  
**Step 3:** choose a teacher by roulette according the probability of each teacher in EA;

the neighborhood of the first  $n$  elements of the solution vector.

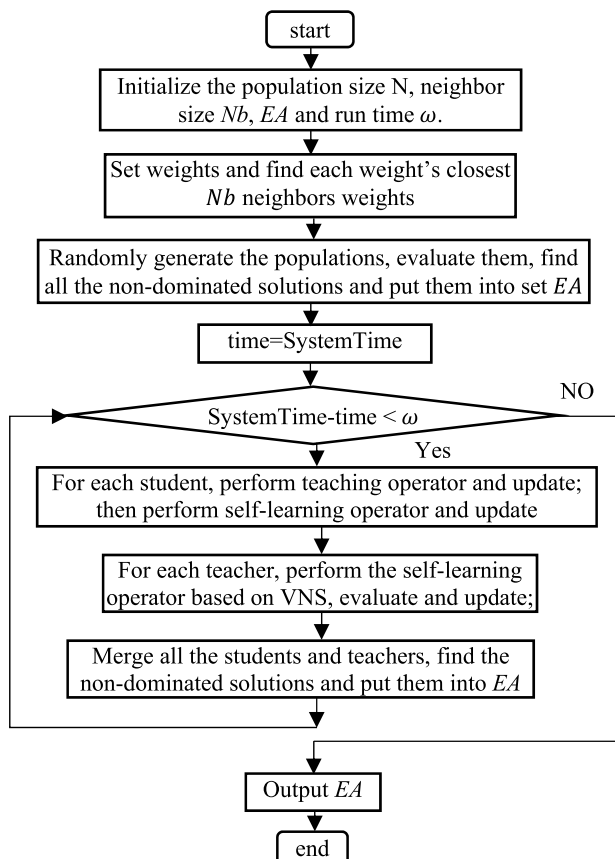
- **Reverse :** Randomly select two positions from 1 to  $n$  in the solution vector, then reverse the elements between these two positions of the solution.
- **Swap:** Randomly select two positions from 1 to  $n$  in the solution vector, swap the elements of these two positions.
- **Insert:** Randomly select two positions from 1 to  $n$  in the solution vector, insert an element in one position before the other.

**Algorithm 5** Self-Learning Based on VNS

**Input:** a solution  $\pi$   
**Output:** updated solution  $\pi_{new}$   
**Step 1:** initialize the maximum iteration number  $iternum$  of per neighborhood;  
**Step 2:** for each neighborhood, performs the follow operations  
**Step2.1:**  $k = 0$ ;  
**Step2.2:** while( $k < iternum$ )  
  { perform the neighborhood operations;  
  **if**( $f_{\pi} < f_{\pi_{new}}$ )**then**  
  {  $\pi = \pi_{new}$ ;  
  update the current student's neighbors with  $\pi_{new}$ ;  
   $k = k + 1$ ;  
  }  
  }

**H. FRAMEWORK OF THE PROPOSED ALGORITHM**

To show the proposed algorithm HMOTLBO more clearly, the pseudocode of it is described in Algorithm 6 and the flow chart is shown in Figure 5.

**FIGURE 5.** Flow chart of the HMOTLBO.**Algorithm 6** HMOTLBO

**Input:** the HFS problem;  
**Output:** external archive set  $EA$ ;  
**Step1:** //Initialization  
**Step1.1:** set the size of the population  $N$ ;  
  set the neighbor size of each weight  $NB$ ;  
  set the external archive set  $EA = \phi$ ;  
  set the maximal run time  $\omega$ ;  
**Step1.2:** for( $ii=1$ ;  $ii \leq N$ ;  $ii++$ )  
  { calculate weight  $\lambda^{ii}$  with Eq(16);  
  randomly generate solution  $\pi_{ii}$ ;  
  calculate the two objectives' value of  $\pi_{ii}$ ;  
**Step1.3:** for each weight  $\lambda^{ii}$   
  find its closest  $NB$  neighbor weights and put them into set  $B(ii)$ ;  
**Step1.4:** sort all the solutions with the non-dominated sorting algorithm in NSGAI [16];  
  find all the non-dominated solutions and put them into set  $EA$ ;  
**Step1.5:** for( $ii=1$ ;  $ii \leq N$ ;  $ii++$ )  
  Allocate weight to each student  $\pi_{ii}$  with Algorithm 2.  
**Step 2:** //evolutionary phase  
  TIME = system\_time; //record the current time  
  while(system\_time-TIME <  $\omega$ ) {  
  **Step2.1:** for( $ii=1$ ;  $ii \leq N$ ;  $ii++$ )  
  { **a):** //teaching  
  select a teacher from set  $EA$  with Algorithm 4 for student  $\pi_{ii}$ ;  
  execute teaching phase with Algorithm 4;  
  **b) //self-learning**  
  execute Algorithm 5 on  $\pi_{ii}$ ;  
  }  
  **Step 2.2:** // for each teacher, perform self-learning  
  for( $t=1$ ;  $t < |EA|$ ;  $t++$ )  
  execute self-learning with Algorithm 5 on  $\pi_t$ ;  
  **Step 2.3:** //  
  merge all the students and teachers, set  $EA = \emptyset$ , find all the non-dominated solutions and put them into  $EA$ .  
  }  
**Step3:** //output  
  output solutions of  $EA$ ;

Since HMOTLBO is a swarm intelligence and evolutionary algorithms, it's clearly that the computational complexity(CC) of it is mainly decided by its problem size, population size, evolution times, etc. For HFS scheduling problem, the problem size is mainly decided by the number of jobs  $n$ . At the same time, let us assume the population size of HMOTLBO is  $N$  and the evolution times is  $E$ . From the framework of HMOTLBO, we know the main algorithms include the decoding algorithm, the initialization algorithm and the evolutionary algorithm. Here, the CC of the decoding



algorithm is  $O(n^2 \cdot \log_2 n)$ . The CC of the initialization algorithm is mainly decided by the decoding operator, so the CC of initialization algorithm is  $O(N \cdot n^2 \cdot \log_2 n)$ . During the evolution phase, the CC of it is mainly decided by the teaching algorithm, self-learning algorithm and evolution times. Suppose that the number of searches in each neighborhood of the VNS algorithm is far less than  $n$ , then the total CC of evolution algorithm is  $O(E \cdot N \cdot n^2 \cdot \log_2 n)$ . Since  $E$  is much larger than 1, the complexity of the HMOTLBO is mainly determined by the evolutionary algorithm, that is  $O(E \cdot N \cdot n^2 \cdot \log_2 n)$ .

## V. EXPERIMENT RESULTS

### A. ENVIRONMENT AND TEST INSTANCE

In this section, we carry out the computational experiments to validate the performance of the proposed algorithm HMOTLBO. The algorithm is implemented in C++ language on an intel core i7-8550U, 1.88GHz PC with 16GB memory. All cases are named as  $j^*c^*a^*$ , here,  $*$  represents an integer. For example, case j15c5a1 represents that there are 15 jobs to be processed in a workshop with 5 processing stages, and 1 means the first case of this class. All cases are randomly generated, in which the processing time of each job is between 10 and 30, the number of machines at each stage is between 2 and 3, the processing energy consumption per unit time of a machine is between 5 and 10, the standby energy consumption per unit time of a machine is between 1 and 5, the energy consumption of a single start-up of the machine is between 20 and 30, and the number of jobs is 15, 20 and 30 respectively.

We set up four types of experiments, the first one is to verify the effectiveness of the decoding rules and then we will chose a better decoding rule for HMOTLBO. The second one is to verify the effectiveness of the weight matching strategy and teacher selection strategy. The third one is to verify the effectiveness of the self-learning operator based on VNS. The fourth one is to verify the effectiveness of the proposed algorithm in this paper through comparing with other two better algorithms.

Through orthogonal experiment, the parameters for the proposed HMOTLBO algorithm are set as follows:

- Population size is 60.
- The maximum running time  $\omega$ (stop condition) equal to  $s \times n \times \alpha$  (s), here,  $s$  is the number of processing stages,  $n$  is the number of jobs and  $\alpha$  is a parameter ant it's value is set as 1, 2, 3 based on the number of jobs 15,20,30 respectively.
- The maximum iterations of VNS of the self-learning for each neighbor is 5.
- The size of external Pareto archive set  $EA$  is 4 times of the population size.

To check the effectiveness of HMOTLBO, two better multi-objective algorithms about HFS scheduling problem, EA-MOA [43] and NIW-PSO [44] are selected for comparison. All three algorithms adopt the same population size,

the greedy shift strategy and the stopping condition, the other parameters are recommended by their original papers.

### B. EVALUATION METRICS

To evaluate the performance of the multi-objective algorithms on solving HFS, we firstly need to address the following issues:

(1) Inverted generation distance(IGD), the distance of the true Pareto frontier solutions with the obtained Pareto frontier solutions. It can reflect the distribution and convergence of the solution at the same time. The smaller the IGD is, the better the solution is. Eq(22) is the formula for calculating IGD. Here,  $A^*$  is the true Pareto frontier solution set. In this study, the true Pareto frontier solutions are found by running all the compared algorithms and merged all the non-dominated solutions they got.  $A$  is a set of Pareto frontier solutions obtained by an algorithm,  $d(v, A)$  is the minimum Euclidean distance of solution  $v$  to the solutions of  $A$ .

$$IGD(A, A^*) = \frac{\sum_{v \in A^*} d(v, A)}{|A^*|} \quad (22)$$

(2) The rate of the Pareto optimal solutions ( $V_r$ ):  $V_r$  is used to compute the ratio of non-dominated solutions in set  $A$ . Obviously, a larger value of  $V_r$  represents set  $A$  with a higher probability for the obtained solution to be a non-dominated solution. If  $V_r$  is close to one, almost all of the solutions in  $A$  are in  $A^*$ . Whereas, if  $V_r$  is close to zero, almost all of the obtained solutions are dominated by solutions in  $A^*$ .  $V_r$  is calculated by Eq (23).

$$V_r = \frac{|A|}{|A^*|} \quad (23)$$

### C. EFFECTIVENESS OF THE DECODING RULE

To verify which decoding rule of the five have a positive impact on the algorithm optimization results, five variants of HMOTLBO will be compared. Except for the different decoding rules, the five algorithms are same in the other aspects. According to the decoding methods adopted in the algorithm, the five algorithms are named as HMOTLBO-SI, HMOTLBO-SII, HMOTLBO-SIII, HMOTLBO-SIV and HMOTLBO-SV respectively. Figure 6 shows the scatter plot of the Pareto frontier solutions of each algorithm on 15 cases.

In Figure 6, it's clearly the solutions obtained by algorithm HMOTLBO-SI are far away from the Pareto frontier solution. None of these 15 cases can converge to the Pareto frontier solution. The reason is that decoding rule SI needs to use a vector of length  $(s+1) \cdot n$  to represent a solution and its solution space reaches  $n! \cdot \prod_{j=1}^s m_j^n$ , which is very large than the other decoding rule's. Therefore, the algorithm HMOTLBO-SI's search speed is relatively slow and the decoding rule SI is poor. For the Pareto frontier solutions obtained by HMOTLBO-SI, there are 12 out of 15 cases that can converge to the real Pareto frontier, but most are in the region with

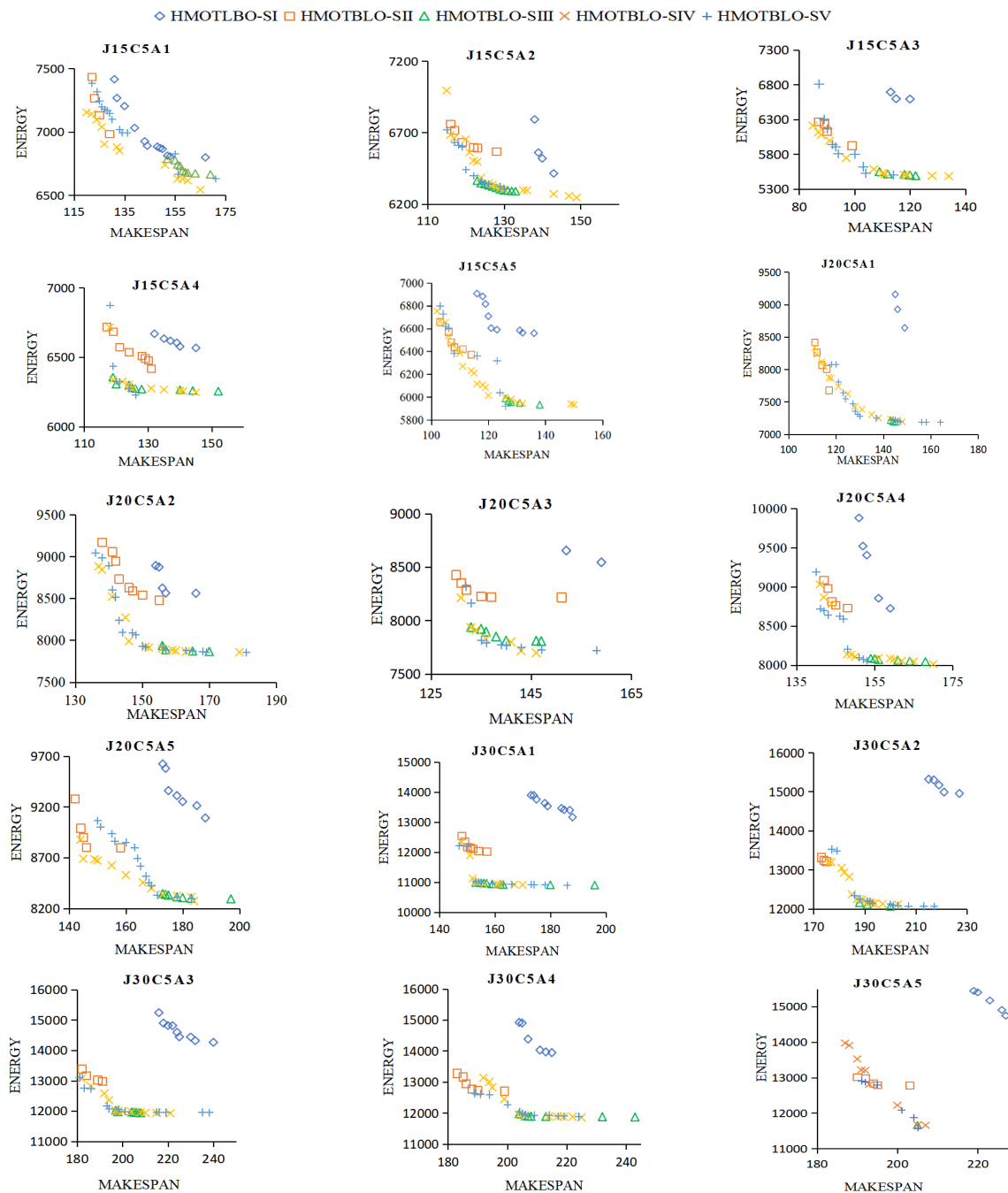


FIGURE 6. Scatter diagram of the Pareto front of the 15 instances.

smaller makespan and higher energy consumption. On cases j15c5a4, j20c5a2 and j20c5a3, the Pareto frontier solutions obtained by the algorithm HMOTLBO-SII can't converge to the real Pareto frontier solutions. Therefore, decoding rule SII is more advantageous to the makespan, but not to the minimum energy consumption. On the contrary, the Pareto frontier solutions obtained by the algorithm HMOTLBO-SIII are all concentrated on the Pareto frontier with smaller energy consumption and higher makespan on 15 instances. There-

fore, decoding rule SIII is more advantageous to minimum energy consumption but not conducive to the objective of improving production efficiency. The biased decoding rule is the main reason that SII and SIII fall into a certain solution interval. But compared with SI, decoding rules SII and SIII only use the first  $n$  elements of the solution vector, their solution space is only  $n!$ , therefore, their convergence speed is faster than SI. The algorithms HMOTLBO-SIV and HMOTLBO-SV obtained the better Pareto frontier solutions

and well distributed on all 15 instances, there is no clearly different between them, so we can draw the conclusion that the decoding method SIV and SV are equally better than the other three decoding rules. SIV is better than SII and SIII because it uses the decoding rules of them in probability and avoids their eccentricity. For the hybrid decoding rule SV, 90% of the solutions are decoded by SI, but it converges faster than SI. One of the main reasons is that the solution decoded by SII and SIII transfer their good machine allocation genes to other solutions through the teaching operator, thus speeding up the convergence of the algorithm as a whole. Later in this paper, if there is no clear illustration, the algorithm HMOTLBO in this paper will use SIV to decode the solutions.

#### D. EFFECTIVENESS OF MATCHING STRATEGY AND TEACHER SELECTION

To check the effectiveness of the proposed weight matching strategy and teacher selection strategy, two algorithms, HMOTLBO with the randomly strategy named HMOTLBO-I and HMOTLBO with the proposed strategy named HMOTLBO-II, are compared. Each algorithm will be run 10 times, the average IGD and the average  $V_r$  will be compared. For ease of analysis, better values are shown in bold. The experiment results are shown in Table 1.

TABLE 1. Comparison of 15 instances.

instance	machine layout	HMOTLBO-I		HMOTLBO-II	
		IGD	$V_r$	IGD	$V_r$
j15c5a1	2,3,2,2,2	7.676087E-02	21.4%	<b>7.063712E-03</b>	<b>86.6%</b>
j15c5a2	2,2,3,3,2	3.751357E-02	33.3%	<b>2.228446E-02</b>	<b>75%</b>
j15c5a3	3,3,3,3,3	1.464377E-02	33.3%	<b>3.001182E-03</b>	<b>91.7%</b>
j15c5a4	2,3,2,2,3	0.2393579	25%	<b>5.281094E-02</b>	<b>75%</b>
j15c5a5	3,3,2,3,2	0.120763	33.3%	<b>2.417223E-02</b>	<b>77.8%</b>
j20c5a1	3,3,2,3,3	5.368575E-02	35.7%	<b>1.774827E-02</b>	<b>85.7%</b>
j20c5a2	3,2,2,3,2	<b>1.553168E-02</b>	<b>92.3%</b>	0.1055932	37.8%
j20c5a3	3,2,2,3,3	0.3161123	37.5%	<b>0.1315721</b>	<b>62.5%</b>
j20c5a4	2,3,2,3,3	<b>2.136446E-02</b>	<b>91.7%</b>	0.1194534	16.7%
j20c5a5	2,2,2,2,3	8.053818E-02	38.5%	<b>6.050022E-02</b>	<b>84.6%</b>
j30c5a1	3,3,2,3,3	5.384076E-02	12.5%	<b>0</b>	<b>100%</b>
j30c5a2	3,3,3,2,2	4.491777E-02	35.7%	<b>0.0149897</b>	<b>64.3%</b>
j30c5a3	2,3,2,3,3	0.1030563	50%	<b>5.267221E-02</b>	50%
j30c5a4	2,2,3,3,3	<b>1.957469E-02</b>	<b>53.8%</b>	4.795547E-02	46.2%
j30c5a5	3,3,2,3,3	0.161538	0%	<b>0</b>	<b>100%</b>

In Table 1, in the comparisons of the IGD, HMOTLBO-II has 12 better out of the given 15 test cases, which is significantly better than HMOTLBO-I. Hence, the Pareto frontier solutions obtained by HMOTLBO-II has the better distribution and convergence. In the comparison of the  $V_r$ , HMOTLBO-II has 12 better out of the given 15 test cases, while HMOTLBO-I just gets three. So the conclusion is that the proposed strategy is effective.

#### E. EFFECTIVENESS OF THE SELF-LEARNING OPERATOR BASED ON VNS

To check the effectiveness of the proposed self-learning operator based on VNS, in this section, we compared four different HMOTLBO algorithms. The algorithm with self-learning operator based on insert search named HMOTLBO-I, based on reverse search named HMOTLBO-II, based on swap search named HMOTLBO-III and the algorithm based on VNS named HMOTLBO. Each algorithm will be run 10 times, the average IGD and average  $V_r$  will be compared. The experiment results are shown in Table 2, for ease of analysis, better values are shown in bold.

In Table 2, the results of IGD show that HMOTLBO obtained 11 best values for the 15 cases. The average IGD that HMOTLBO got on 15 instances is approximately equal to 0.045, this is smaller than the average IGD obtained by the other three algorithms, this illustrates that the proposed VNS has the strongest local exploration ability compared with the reverse search, insert search and swap search. In the comparison of the  $V_r$ , HMOTLBO obtained 12 best values for the 15 cases and the average  $V_r$  on 15 instances is 46.7%, bigger than the other three algorithms, this also shows the effectiveness of the VNS.

#### F. COMPARISONS WITH OTHER ALGORITHMS

After the previous experiments, the matching strategy, decoding method SV and the self-learning operator based on VNS are assembled in algorithm HMOTLBO. In the following comparison experiments, two better multi-objective algorithms EA-MOA [43] and NIW-PSO [44] are selected to compare. The parameters of each algorithm are described in the subsection A of this section. Here, we use IGD to measure the algorithms. Each algorithm is run 10 times independently on each case and the ideal Pareto frontier solutions set  $A^*$  are the dominated solutions from the Pareto frontier solutions that those three algorithms got on that case. The results are shown in Figure 7 and abscissa 1 in the figure represents the algorithm HMOTLBO, 2 represents EA-MOA and 3 represents NIW-PSO.

The following conclusion can be drawn from Figure 7, in the comparison of the mid-value value of IGD, HMOTLBO gets 11 better out of the given 15 cases, which is significantly better than the other compared algorithms. EA-MOA gets 4 best out of the given 15 cases and NIW-PSO gets none of them. Hence, HMOTLBO has the best convergence ability and the best distribution of solutions, EA-MOA is the second, and NIW-PSO is the third. In the comparison of the height of boxes, HMOTLBO obtains the box with smaller height on 8 cases out of the 15 test instances, EA-MOA gets 7 out of the 15 test cases, and NIW-PSO gets zero. These shows that the stability of HMOTLBO is the best, EA-MOA is the second, and NIW-PSO is the third.

To compare the computation time and the convergence of HMOTLBO, EA-MOA and NIW-PSO, set the number of iteration of each algorithm 500 and the other parameters is not

TABLE 2. Comparison of 15 instances.

instance	HMOTLBO-I		HMOTLBO-II		HMOTLBO-III		HMOTLBO	
	IGD	$V_r$	IGD	$V_r$	IGD	$V_r$	IGD	$V_r$
j15c5a1	4.515295E-02	25%	8.837123E-02	21.1%	3.335571E-02	30%	<b>0.1377006E-02</b>	<b>40%</b>
j15c5a2	6.277855E-02	26.7%	5.838967E-02	25%	5.938575E-02	20%	<b>2.652844E-02</b>	<b>58.1%</b>
j15c5a3	<b>4.499588E-02</b>	39.6%	6.128944E-02	33.3%	6.603258E-02	9.2%	4.760357E-02	<b>41.3%</b>
j15c5a4	0.1470671	21.1%	0.194783	14%	0.1170145	26.7%	<b>9.860413E-02</b>	<b>53.7%</b>
j15c5a5	0.1050358	11.1%	8.376975E-03	28.2%	7.616755E-02	16.7%	<b>4.796298E-02</b>	<b>66.7%</b>
J20c5a1	4.636852E-02	<b>33.3%</b>	0.048564	31.1%	5.981852E-02	29.7%	<b>4.202433E-02</b>	<b>33.3%</b>
J20c5a2	5.882316E-02	14.5%	<b>4.131597E-02</b>	<b>33.3%</b>	4.483866E-02	<b>33.3%</b>	4.671205E-02	30%
J20c5a3	0.1668624	33.3%	9.433841E-02	11.1%	0.0708392	44.8%	<b>3.603953E-02</b>	<b>56.5%</b>
J20c5a4	8.834033E-02	13.3%	8.318991E-02	25%	4.370958E-02	<b>42.7%</b>	<b>0.0225044</b>	0.41.1%
J20c5a5	0.1119991	20%	0.1337489	24.4%	0.1300563	23.3%	<b>2.737369E-02</b>	<b>66.7%</b>
J30c5a1	0.1009227	16.7%	8.710619E-02	29.5%	<b>8.135621E-02</b>	36.3%	8.575317E-02	<b>42%</b>
J30c5a2	0.1483034	23.3%	0.0870332	31.1%	9.192136E-02	28%	<b>2.607352E-02</b>	<b>46.7%</b>
J30c5a3	0.0945886	10%	8.829624E-02	25%	6.606433E-02	44.7%	<b>2.436885E-02</b>	<b>63.3%</b>
J30c5a4	8.688192E-02	22.3%	7.628866E-02	29%	0.1138582	14.5%	<b>5.236413E-02</b>	<b>38.2%</b>
J30c5a5	0.0828857	31.3%	<b>3.541702E-02</b>	<b>53%</b>	9.286912E-02	13.3%	9.512606E-02	16.7%
Average	0.09273374	22.8%	0.07900158	27.6%	0.09286912	27.5%	<b>0.04536106</b>	<b>46.7%</b>

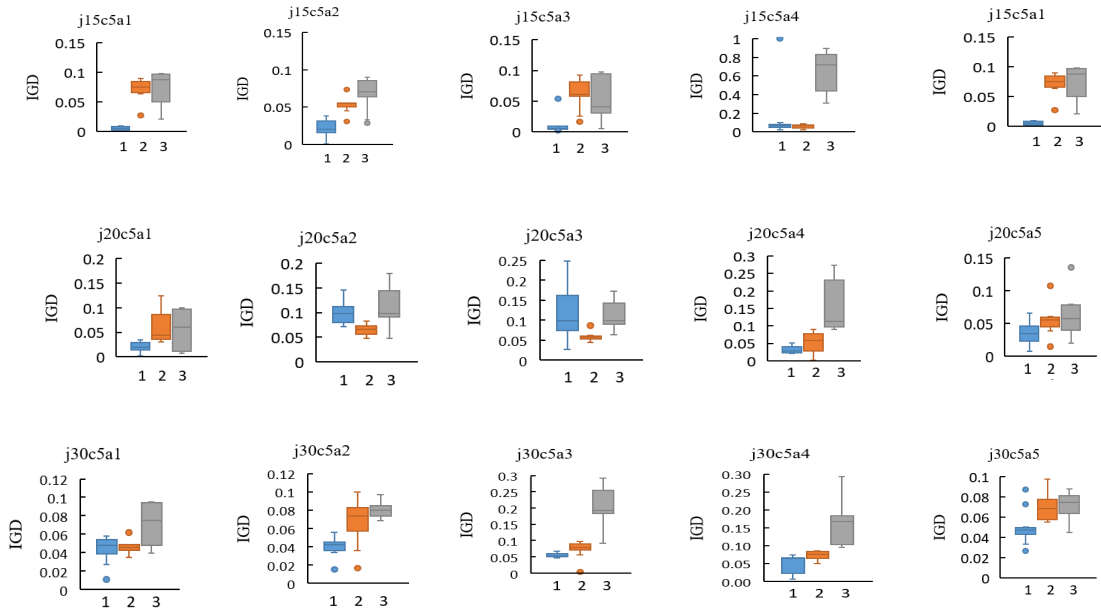


FIGURE 7. Boxplot of IGD for the three compared algorithm.

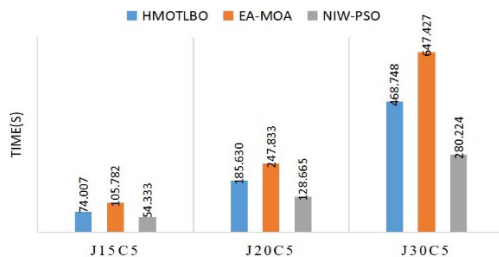


FIGURE 8. Comparison of the computation time.

changed. Figure 8 shows the average computation time histogram of the three algorithms on three scale cases. Figure 9 shows the convergence curves of the three algorithms on cases j15c5a5, j20c5a5 and j30c5a4 respectively.

From Figure 8, we can draw the conclusion that under the same number of iterations, algorithm NIW-PSO has the shortest computation time on three scale cases, followed by the algorithm HMOTLBO, and algorithm EA-MOA has the longest computation time. Moreover, with the increase of the scale of the cases, the computation time of the three algorithms increase. From Figure 9, we can draw the conclusion that under the same number of iterations, the algorithm HMOTLBO can converge to the Pareto front solutions faster and better, followed by EA-MOA. NIW-PSO has the slowest convergence speed and the worst convergence quality.

For test case j15c5a1, Figure 10 reports the Gantt chart, here, figure (a) has the minimum makespan and figure (b) has



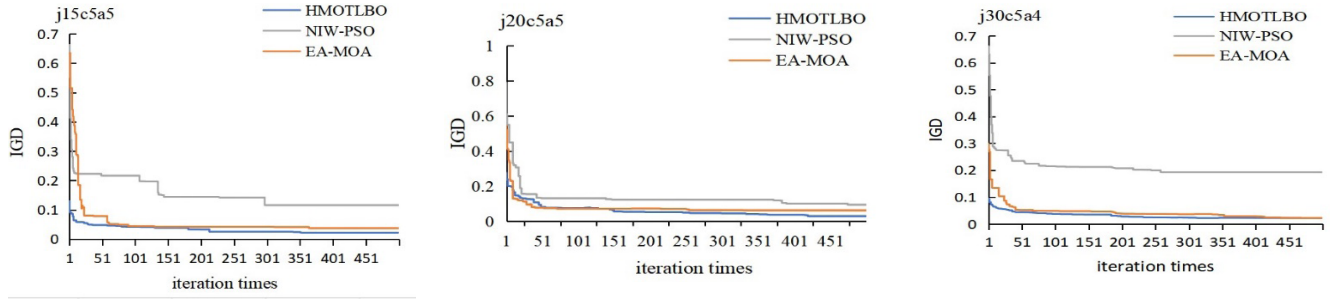


FIGURE 9. Convergence curves of the three algorithms on three cases.

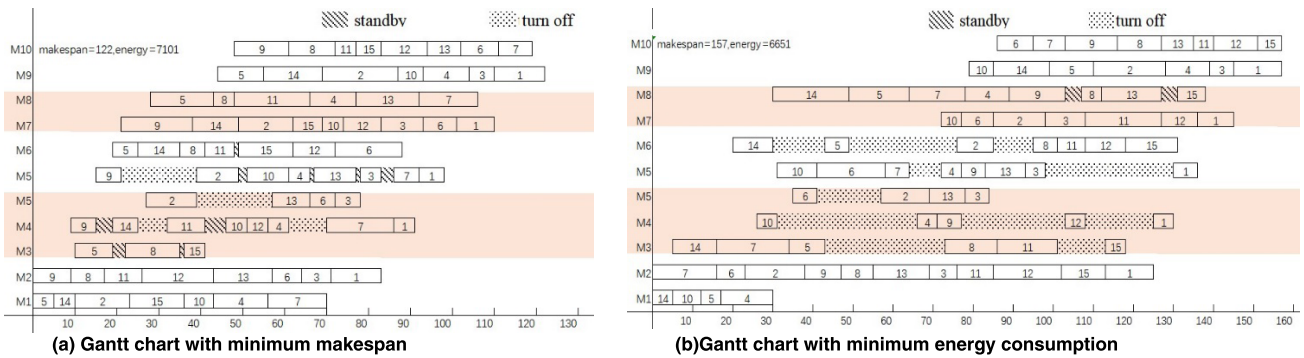


FIGURE 10. Gantt chart for case j15c5a1.

the minimum energy consumption. This shows the effectiveness of the proposed HMOTLBO algorithm.

VI. CONCLUSION

In this study, the scheduling problem of HFS with unrelated machine considering the makespan and the minimum energy consumption objectives is investigated. To solve this bi-objective scheduling problem effectively, a mixed integer linear programming (MILP) model is formulated and an efficient multi-objective optimization algorithm based on decomposition is proposed.

In the outline of the proposed algorithm HMOTLBO, we devised a new encoding method to represent a solution, which can cover the most solution space and support the five different decoding method as discussed in this paper. An external archive(EA) set is used to record the Pareto frontier and all teachers come from this set in view of good teachers can teach excellent students. To enhance the exploration and exploitation ability of the algorithm during each iteration, besides the normal teaching process, each students and teachers will perform a self-learning operator based on variable neighborhood search(VNS). To improve the efficiency of the algorithm, for each student, the weight is assigned by a weight matching algorithm rather than randomly assigned during the initialize phase. Based on the experimental comparisons, we can conclude that the proposed algorithm exhibits high capabilities in handling the multi-objective scheduling problem of HFS.

Since the HFS is a workshop with strong industrial background, we believe that scheduling HFS with energy considerations will be an important research topic for a long time, which is a very meaningful tool to achieve the sustainable manufacturing goal of the energy-intensive companies. Our future work will extend the proposed HFS scheduling problem to a more complex environment, such as dynamic environment or an uncertain environment.

REFERENCES

- [1] C. Cheng, Y. J. Tan, R. J. He, and Y. Feng, "Chemical reaction optimization for earliness-tardiness scheduling problem," *Appl. Mech. Mater.*, vols. 513–517, pp. 2594–2598, Feb. 2014.
- [2] X.-F. Pang, S.-P. Yu, Z.-Y. Zhang, B.-L. Zheng, and T.-Y. Chai, "Optimal rescheduling method for steelmaking-continuous casting," *J. Syst. Eng.*, vol. 14, no. 5, pp. 98–103, 2010.
- [3] A. Costa, F. A. Cappadonna, and S. Fichera, "A dual encoding-based meta-heuristic algorithm for solving a constrained hybrid flow shop scheduling problem," *Comput. Ind. Eng.*, vol. 64, no. 4, pp. 937–958, Apr. 2013.
- [4] T. S. Arthanari and K. G. Ramamurthy, "An extension of two machine sequencing problem," *Opsearch*, vol. 8, no. 1, pp. 10–22, 1971.
- [5] J. N. D. Gupta, "Two-stage, hybrid flowshop scheduling problem," *J. Oper. Res. Soc.*, vol. 39, no. 4, pp. 359–364, Apr. 1988.
- [6] O. Engin and A. Döyem, "A new approach to solve hybrid flow shop scheduling problems by artificial immune system," *Future Gener. Comput. Syst.*, vol. 20, no. 6, pp. 1083–1095, 2004.
- [7] K. Alaykýran, O. Engin, and A. Döyem, "Using ant colony optimization to solve hybrid flow shop scheduling problems," *Int. J. Adv. Manuf. Technol.*, vol. 35, nos. 5–6, pp. 541–550, Nov. 2007.
- [8] S. Khalouli, F. Ghedjati, and A. Hamzaoui, "An ant colony system algorithm for the hybrid flow-shop scheduling problem," *Int. J. Appl. Meta-heuristic Comput.*, vol. 2, no. 1, pp. 29–43, Jan. 2011.



- [9] C.-J. Liao, E. Tjandradjaja, and T.-P. Chung, "An approach using particle swarm optimization and bottleneck heuristic to solve hybrid flow shop scheduling problem," *Appl. Soft Comput.*, vol. 12, no. 6, pp. 1755–1764, Jun. 2012.
- [10] J.-Q. Li and Q.-K. Pan, "Solving the large-scale hybrid flow shop scheduling problem with limited buffers by a hybrid artificial bee colony algorithm," *Inf. Sci.*, vol. 316, pp. 487–502, Sep. 2015.
- [11] C. Song, "Improved greedy genetic algorithm for solving the hybrid flow-shop scheduling problem," *Syst. Eng. Electron.*, vol. 41, no. 5, pp. 1079–1086, 2019.
- [12] Z. Han, H. Shi, F. Qiao, and L. Yue, "Multiple rules decision-based DE solution for the earliness-tardiness instance of hybrid flow-shop scheduling problem," *Int. J. Model. Identificat. Control*, vol. 16, no. 2, pp. 97–107, 2012.
- [13] R. M'Hallah and A. Alhajraf, "Ant colony systems for the single-machine total weighted earliness tardiness scheduling problem," *J. Scheduling*, vol. 19, no. 2, pp. 191–205, Apr. 2016.
- [14] L. U. Fei and T. G. Hui, "Solution to earliness and tardiness hybrid flowshop scheduling problem using allied genetic algorithm," *Comput. Appl.*, vol. 24, no. 7, pp. 122–124, 2004.
- [15] N. Srinivas and K. Deb, "Multiobjective optimization using nondominated sorting in genetic algorithms," *Evol. Comput.*, vol. 2, no. 3, pp. 221–248, 1995.
- [16] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, Apr. 2002.
- [17] Q. Zhang and H. Li, "MOEA/D: A multiobjective evolutionary algorithm based on decomposition," *IEEE Trans. Evol. Comput.*, vol. 11, no. 6, pp. 712–731, Dec. 2007.
- [18] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: A comparative instance study and the strength Pareto approach," *IEEE Trans. Evol. Comput.*, vol. 3, no. 4, pp. 257–271, Nov. 1999.
- [19] G. Rudolph and A. Agapie, "Convergence properties of some multi-objective evolutionary algorithms," in *Proc. IEEE Congr. Evol. Comput.*, Jul. 2000, vol. 1, no. 20, pp. 1010–1016.
- [20] E. L. Solano-Charris, J. R. Montoya-Torres, and C. D. Paternina-Arboleda, "Ant colony optimization algorithm for a bi-criteria 2-stage hybrid flow-shop scheduling problem," *J. Intell. Manuf.*, vol. 22, no. 5, pp. 815–822, 2011.
- [21] M. K. Marichelvam, T. Prabaharan, and X. S. Yang, "A discrete firefly algorithm for the multi-objective hybrid flowshop scheduling problems," *IEEE Trans. Evol. Comput.*, vol. 18, no. 2, pp. 301–305, Apr. 2014.
- [22] L. R. Mundim and T. A. D. Queiroz, "Using a variable neighborhood search to solve a bi-objective identical parallel machine scheduling problem," *Electron. Notes Discrete Math.*, vol. 66, pp. 127–134, 2018.
- [23] E. C. de Siqueira, M. J. F. Souza, and S. R. de Souza, "A multi-objective variable neighborhood search algorithm for solving the hybrid flow shop problem," *Electron. Notes Discrete Math.*, vol. 66, pp. 87–94, Apr. 2018.
- [24] K.-C. Ying, S.-W. Lin, and S.-Y. Wan, "Bi-objective reentrant hybrid flowshop scheduling: An iterated Pareto greedy algorithm," *Int. J. Prod. Res.*, vol. 52, nos. 19–20, pp. 5735–5747, 2014.
- [25] T. H. Tran and K. M. Ng, "A hybrid water flow algorithm for multi-objective flexible flow shop scheduling problems," *Eng. Optim.*, vol. 45, nos. 19–20, pp. 483–502, 2013.
- [26] H. Asefi, F. Jolai, M. Rabiee, and M. E. T. Araghi, "A hybrid NSGA-II and VNS for solving a bi-objective no-wait flexible flowshop scheduling problem," *Int. J. Adv. Manuf. Technol.*, vol. 75, nos. 5–8, pp. 1017–1033, 2014.
- [27] H.-M. Cho, S.-J. Bae, J. W. Kim, and I.-J. Jeong, "Bi-objective scheduling for reentrant hybrid flow shop using Pareto genetic algorithm," *Comput. Ind. Eng.*, vol. 61, no. 3, pp. 529–541, Oct. 2010.
- [28] H. Wang, Y. Fu, M. Huang, G. Q. Huang, and J. Wang, "A NSGA-II based memetic algorithm for multiobjective parallel flowshop scheduling problem," *Comput. Ind. Eng.*, vol. 113, pp. 185–194, Nov. 2017.
- [29] O. Shahvari and R. Logendran, "Hybrid flow shop batching and scheduling with a bi-criteria objective," *Int. J. Prod. Econ.*, vol. 179, pp. 239–258, Sep. 2016.
- [30] C. Zhu-Min, "A heuristic algorithm for hybrid flow shop's master scheduling," *Ind. Eng. Manage.*, vol. 14, no. 5, pp. 69–78, 2009.
- [31] C. Lu, L. Gao, Q. Pan, X. Li, and J. Zheng, "A multi-objective cellular grey wolf optimizer for hybrid flowshop scheduling problem considering noise pollution," *Appl. Soft Comput. J.*, vol. 75, pp. 728–749, Feb. 2019.
- [32] K. Gao, Y. Huang, A. Sadollah, and L. Wang, "A review of energy-efficient scheduling in intelligent production systems," *Complex Intell. Syst.*, vol. 6, no. 2, pp. 237–249, 2020.
- [33] A. Che, S. Zhang, and X. Wu, "Energy-conscious unrelated parallel machine scheduling under time-of-use electricity tariffs," *J. Cleaner Prod.*, vol. 156, no. 10, pp. 688–697, Jul. 2017.
- [34] A. Che, Y. Zeng, and K. Lyu, "An efficient greedy insertion heuristic for energy-conscious single machine scheduling problem under time-of-use electricity tariffs," *J. Cleaner Prod.*, vol. 129, no. 15, pp. 565–577, Aug. 2016.
- [35] X. Wu and Y. Sun, "A green scheduling algorithm for flexible job shop with energy-saving measures," *J. Clean. Prod.*, vol. 172, no. 3, pp. 3249–3264, Jan. 2018.
- [36] H. Wang, Z. Jiang, Y. Wang, H. Zhang, and Y. Wang, "A two-stage optimization method for energy-saving flexible job-shop scheduling based on energy dynamic characterization," *J. Cleaner Prod.*, vol. 188, pp. 575–588, Jul. 2018.
- [37] L. Meng, C. Zhang, and Y. Ren, "Mathematical modeling of energy-efficient integration of process planning and scheduling," *J. Mech. Eng.*, vol. 55, no. 16, pp. 186–196, 2018.
- [38] M. Dai, D. B. Tang, Y. C. Xu, and W. D. Li, "Energy-aware integrated process planning and scheduling for job shops," *Proc. Inst. Mech. Eng. B, J. Eng. Manuf.*, vol. 229, no. 1, pp. 13–26, 2019.
- [39] H. Luo, B. Du, G. Q. Huang, H. Chen, and X. Li, "Hybrid flow shop scheduling considering machine electricity consumption cost," *Int. J. Prod. Econ.*, vol. 146, no. 2, pp. 423–439, 2013.
- [40] M. Dai, D. B. Tang, A. Giret, M. A. Salido, and W. D. Li, "Energy-efficient scheduling for a flexible flow shop using an improved genetic-simulated annealing algorithm," *Robot. Comput.-Integr. Manuf.*, vol. 29, no. 5, pp. 418–429, 2013.
- [41] H. Zhang, F. Zhao, K. Fang, and J. W. Sutherland, "Energy-conscious flow shop scheduling under time-of-use electricity tariffs," *CIRP Ann.-Manuf. Technol.*, vol. 63, no. 1, pp. 37–40, 2014.
- [42] R. V. Rao, V. J. Savsani, and D. P. Vakharia, "Teaching-learning-based optimization: A novel method for constrained mechanical design optimization problems," *Comput.-Aided Des.*, vol. 43, no. 3, pp. 303–315, Mar. 2011.
- [43] J.-Q. Li, H.-Y. Sang, Y.-Y. Han, C.-Q. Wang, and K.-Z. Gao, "Efficient multi-objective optimization algorithm for hybrid flow shop scheduling problems with setup energy consumptions," *J. Clean. Prod.*, vol. 181, pp. 584–598, Apr. 2018.
- [44] D. Tang, M. Dai, M. A. Salido, and A. Giret, "Energy-efficient dynamic scheduling for a flexible flow shop using an improved particle swarm optimization," *Comput. Ind.*, vol. 81, pp. 82–95, Sep. 2016.



**CUNLI SONG** was born in Shanxi, China, in 1975. She received the M.S. degree from the School of Software, Dalian Jiaotong University, China, in 2003, and the Ph.D. degree in control theory and control engineering from the Dalian University of Technology, China, in 2011. Since 2011, she has been an Associate Professor with the School of Software, Dalian Jiaotong University. She has authored more than 20 refereed articles. Her current research interests include evolutionary computation, multi-objective optimization, and flow shop scheduling.

• • •