Computer Modeling in Engineering & Sciences

Tech Science Press

DOI: 10.32604/cmes.2022.020307

ARTICLE

An Improved Hyperplane Assisted Multiobjective Optimization for Distributed Hybrid Flow Shop Scheduling Problem in Glass Manufacturing Systems

Yadian Geng[1] **and Junqing Li**[1,2,*]

[1]College of Computer Science, Liaocheng University, Liaocheng, 252059, China

[2]School of Information Science and Engineering, Shandong Normal University, Jinan, 25014, China

*Corresponding Author: Junqing Li. Email: lijunqing@lcu-cs.com

Received: 16 November 2021 Accepted: 11 February 2022

ABSTRACT

To solve the distributed hybrid flow shop scheduling problem (DHFS) in raw glass manufacturing systems, we investigated an improved hyperplane assisted evolutionary algorithm (IhpaEA). Two objectives are simultaneously considered, namely, the maximum completion time and the total energy consumptions. Firstly, each solution is encoded by a three-dimensional vector, i.e., factory assignment, scheduling, and machine assignment. Subsequently, an efficient initialization strategy embeds two heuristics are developed, which can increase the diversity of the population. Then, to improve the global search abilities, a Pareto-based crossover operator is designed to take more advantage of non-dominated solutions. Furthermore, a local search heuristic based on three parts encoding is embedded to enhance the searching performance. To enhance the local search abilities, the cooperation of the search operator is designed to obtain better non-dominated solutions. Finally, the experimental results demonstrate that the proposed algorithm is more efficient than the other three state-of-the-art algorithms. The results show that the Pareto optimal solution set obtained by the improved algorithm is superior to that of the traditional multi-objective algorithm in terms of diversity and convergence of the solution.

KEYWORDS

Distributed hybrid flow shop; energy consumption; hyperplane-assisted multi-objective algorithm; glass manufacturing system

1 Introduction

The hybrid flow shop scheduling problem (HFS) has been investigated and employed in lots of realistic industrial applications [1], such as glass-making systems [1–3] and steelmaking systems [4]. In the classical HFS process, there are several jobs, machines, and stages. A certain number of parallel machines are in each stage, where each arriving job should choose exactly one available machine. And each job follows the same processing route with machine selection flexibility. Therefore, compared with the classical flow shop scheduling problem, in HFS, an additional task is selected to suit machines for each operation, which has been proven to be an NP-hard problem [1].

With the development of industries, more and more researches have focused on distributed scheduling problem, including the distributed flow shop scheduling problem (DFSSP) [5], as well as distributed hybrid flow shop scheduling problem (DHFS) [6]. However, there is less literature for DHFS, compared with the works in solving flow shop or distributed flow shop. Pan et al. [7] studied a distributed flowshop group scheduling problem (DFGSP), where the families are considered in manufacturing cells. Huang et al. [8] proposed three constructive heuristics and an effective discrete artificial bee colony (ABC) algorithm to solve the distributed permutation flowshop scheduling problem. Meng et al. [9] introduced the lot-streaming and carryover sequence-dependent setup time in the distributed permutation flowshop scheduling problem (DPFSP) with non-identical factories. Ying et al. [10] developed a hybrid algorithm with three versions of iterated greedy (IG) algorithm in order to minimize the makespan of the DHFS. Hao et al. [11] considered a DHFS with a brain storm optimization (BSO) algorithm, where the makespan is minimized. Cai et al. [12] proposed a new shuffled frog-leaping algorithm (SFLA) with memeplex quality (MQSFLA), which was used to minimize total tardiness and makespan simultaneously. Shao et al. [13] proposed a multi-neighborhood IG algorithm so as to solve the problem. Li et al. [14] investigated an improved IG algorithm to solve the DPFSP with both robotic transportation and order constraints. Jiang et al. [15] studied the energy-aware DHFS with multiprocessor tasks with considering total energy consumption and makespan. Niu et al. [16] developed an improved NSGA-II algorithm to solve an energy-efficient distributed assembly blocking flow shop problem. Qin et al. [17] utilized a realistic DHFS where a novel integrated production and distribution scheduling problem is focused.

In realistic industry system, including the glass manufacturing system, the improvement of glass raw materials processing has been studied by many researches [18–20], and therefore, the scheduling efficiency has become increasingly important [21]. Na et al. [22] addressed the glass optimization problems by using heuristic methods. Lozano et al. [23] proposed a two-phase heuristic that combines exact methods and searching heuristics. Wang et al. [24] proposed two heuristics based on decomposition idea to minimize total electricity cost and makespan. Wang et al. [25] formulated a mixed integer programming (MIP) for the problem. Typically, a highly energy-consuming stage (i.e., depreciation of machinery) exits in glass making process, which takes up a large part of the production cost. As a result, considering energy consumption in glass manufacturing system is practical as well as necessary.

Recently, multiobjective optimization algorithms have been applied and considered in many domains [26–31]. Shahvari et al. [32] considered a tabu search (TS) algorithm to minimize two different objectives. Zhang et al. [33] proposed a novel multiobjective multifactorial immune algorithm with a novel information transfer method to deal with multiobjective multitask optimization problems. Wang et al. [34] improved the overall efficiency of optimizing multiple tasks simultaneously by reusing the learned knowledge. Li et al. [35] solved flow shop scheduling problems with a novel multiobjective local search framework-based decomposition. Li et al. [36] developed a knowledge-based adaptive reference points multi-objective algorithm (KMOEA) to solve a DHFS with variable speed constraints. Du et al. [37] proposed a hybrid multi-objective optimization algorithm based on an estimation of distribution algorithm (EDA) and deep Q-network to solve a flexible job shop scheduling problem (FJSP) with time-of-use electricity price constraint. Mou et al. [38] developed an effective hybrid collaborative algorithm for energy-efficient distributed permutation flow-shop inverse scheduling. Li et al. [39] proposed an improved artificial immune system (IAIS) algorithm to solve a special case of the FJSP in flexible manufacturing systems. However, less literature investigated the multi-objective optimization in glass manufacturing systems.

Therefore, to solve DHFS in glass manufacturing systems, we propose an improved hyperplane assisted evolutionary algorithm (IhpaEA). The main contributions of this study are as follows: (1) each solution is represented with a three-dimensional vector, including the factory assignment, machine assignment, and operation scheduling; (2) an efficient initialization strategy is developed to increase the diversity of the population (3) an improved crossover operator is designed to enhance the global search abilities of the proposed algorithm; and (4) a cooperative search method is designed to enhance the local search abilities of the proposed algorithm deeply.

The structure of the rest paper is as follows. The problem descriptions are given in Section 2. Next, the developed IhpaEA framework is presented in Section 3. Then, the detailed components of the imposed algorithm are discussed in Section 4. Section 5 illustrates the experimental results to show the advantages of the algorithm. Finally, the last section shows the conclusion and future research directions.

2 Problem Description

The DHFS addressed in this study can be described as follows. There are n independent jobs to be assigned to f factories. Each factory consists of a series of π_i production stages (or processing centers) where there are k parallel machines in each stage. Moreover, each job can be completed in any factory with the same sequence. Each operation can be processed on any selected machine at the corresponding stage.

2.1 Problem Formulation

1) Assumptions:

- Each job should be released at time zero and be operated from the first stage to the next stage;
- All machines are available at time zero and remain continuously available over the entire production horizon;
- A job can be processed on exactly one machine at a time, and a machine can process exactly one job at a time;
- At each stage, one job can select one suitable machine from the parallel machine;
- There is unlimited buffer between stages;
- All machines belonging to the same stage have similar processing abilities.

2) Notations and variables:

Indices:

i	Index of the machines.
j	Index of the jobs.
f	Index of the factories.
k	Index of the stages

Parameters:

n	Number of jobs.
m	Number of machines.
w	Number of stages.
h	Number of factories.
m_k	Number of machines in k th stage, for $k = 1, 2, \ldots, w$.

J_j	Job j, for $i = 1, 2, \ldots, n$.
M_i	Machine i, for $i = 1, 2, \ldots m$.
O_{ij}	ith operation of job j.
n_i	Number of jobs that are processed on M_i. $i = 1, 2, \ldots m$.
J_{ir}	rth job that is processed on M_i. $i = 1, 2, \ldots m$, $r = 1, 2, \ldots, m$.

Variables:

S	$\left(O_{ij}\right)$ Starting time of O_{ij}. $i = 1, 2, \ldots, m, j = 1, 2, \ldots, n$.
$C\left(O_{ij}\right)$	Completion time of O_{ij}. $i = 1, 2, \ldots, m, j = 1, 2, \ldots, n$.
C_{max}	Makespan, i.e., the maximum completion time.
PM_{fki}	Machine power of M_i in stage k of factory f, $k = 1, 2, \ldots, w, i = 1, 2, \ldots, m$, $f = 1, 2, \ldots, h$.
TM_{fki}	Machine working time of M_i in stage k of factory f, $k = 1, 2, \ldots, w, i = 1, 2, \ldots, m$, $f = 1, 2, \ldots, h$.
TEC	Total energy consumption.

Decision Variables:

x_{jf}	A binary decision variable, which equals to 1 when job J_j is assigned to factory f and otherwise equals to 0.
z_{ij}	A binary decision variable, which equals to 1 when job J_j is processed on M_i and otherwise equals to 0.

3) Objective functions:

The makespan (C_{max}) and TEC are considered as two objectives. The first objective is to minimize makespan where $C_{max} = \max\left(CJ_{jk}\right)$. The second objective is to minimize TEC where TEC $= \sum_{f=1}^{h}\sum_{k=1}^{w}\sum_{i=1}^{m} PM_{fki} \cdot TM_{fki}$, i.e., the total energy consumptions during the processing time for all machines.

2.2 Realistic Problem Example

A detailed illustration of the considered realistic DHFS is presented in a glass manufacturing casting system in Fig. 1. A specified quantity of molten glass can be provided by three processing stations. Many beam carriers (BCs) are used to transport those pouring molten glass. Each job or BC is transported to an available factory. The molten glass transported by BC will be operated through at least two stages in each factory: 1) glass forming; and 2) heat treatment stages. The processing operations are the same in all the factories for each BC. After processing in the designated factory, BC will be moved to the next stage, in which one machine will be selected for continuous casting procedure. The specified amount of charging shall be handled for each assigned machine. Fig. 1 presents that the complete working flows in the basic glass manufacturing casting system which can be considered as a typical DHFS with several stages in the last part. Moreover, the realistic processing systems should consider the deteriorating job constraint [33,34].

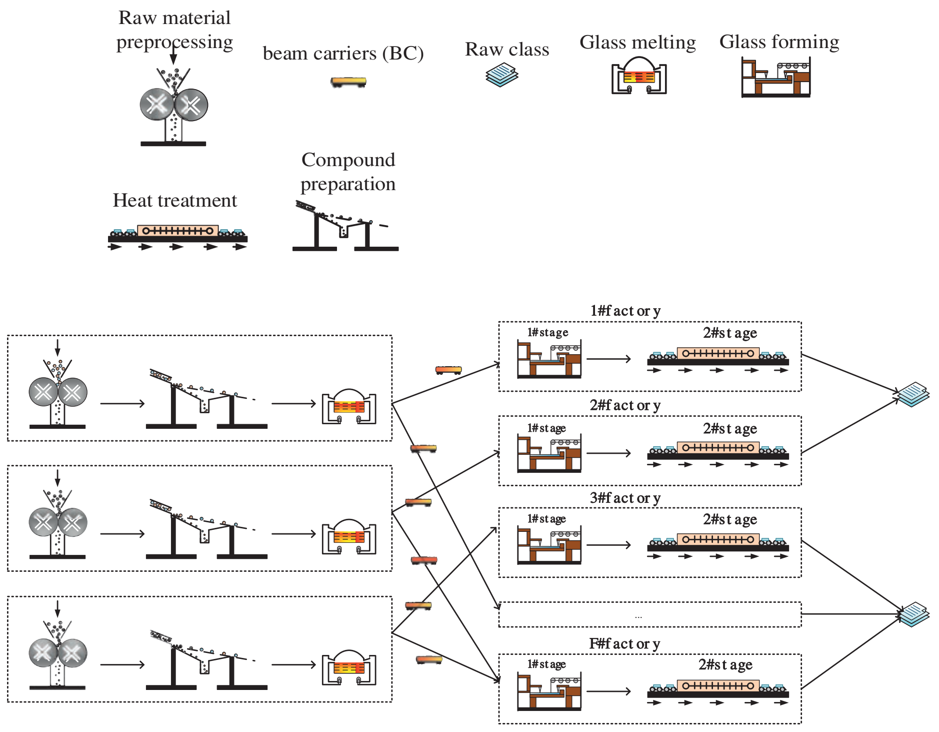


Figure 1: Realistic DHFS problem in a glass manufacturing system

A common production flow is shared by different glass manufacturing systems, i.e., raw glass should experience preprocessing, melting, and forming processes in sequence, as shown in Fig. 2. Specific processing requirements or features are in individual manufacturing systems with the considering that various types of glass are provided by different manufacturers. The detailed processing characteristics of this study are as below:

(1) Raw material preprocessing: Crush large raw materials (soda, quartz sand, feldspar, limestone, etc.) to dry raw materials which are wet, and then remove iron from raw materials to ensure glass quality.

(2) Compound preparation.

(3) Glass melting: In order to make the glass raw materials meet the forming requirements of uniform, bubble free and molten liquid glass, the glass raw materials need to be placed in the pool kiln or crucible kiln and heated at high temperature (1500–1600 degrees).

(4) Glass forming: Liquid glass is processed into the required shape of the specific products.

(5) Heat treatment: Through annealing, quenching and other processes to change the structural state of glass.

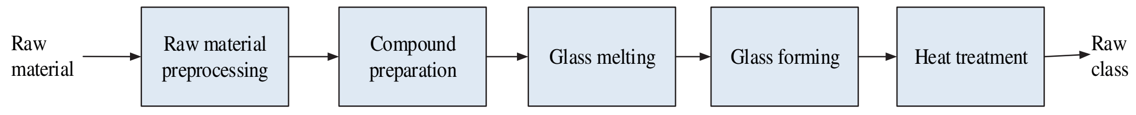


Figure 2: General procedure of a raw glass manufacturing system

3 Methodology

In this section, the proposed IhpaEA algorithm is presented to solve the considered DHFS problem. The first part describes the main framework of the proposed IhpaEA. Then the encoding, decoding, initialization, crossover, and other problem-specific heuristics are presented, respectively.

3.1 Framework of the Proposed IhpaEA

The main framework of the proposed IhpaEA algorithm is an enhanced inverted GA (genetic algorithm) indicator based hpaEA [40]. In IhpaEA, the main components, including the uniform reference point, mating selection and the environmental selection functions are directly included from hpaEA. The prominent solutions are retained by the environment selection strategy of hyperplane assisted evolutionary algorithm. Besides, it uses two criteria to select the size of population and the non-dominated solutions.

Algorithm 1: Framework of IhpaEA

```
Input: Objective optimization problem; maximum evaluations (MES);
       Population size $N$;
Output: Produce $N$ unit vectors as $V \leftarrow \left\{v^1, v^2, \ldots, v^N\right\}$;
1    The final population $P$;
2    The evaluation number is recorded as $FEs \leftarrow N$;
3    Population is randomly initialized as $P$ and $z \leftarrow z^{max}$;
4    The total number of prominent solutions are initialized as $K \leftarrow 0$;
5    while $FEs < MEs$ do
6    $I \leftarrow$ Randomly produce $N - K$ integers between $I$ and $|P|$; (c.f. [41])
7    $I \leftarrow I \cup \{1, 2, \ldots, K\}$; The elements in $I$; (c.f. [41])
8    $P' \leftarrow GenerateOffsprings\,(P\,(I))$; (c.f. Sections 3.5-3.7)
9    $Q \leftarrow P \cup P'$; (c.f. [41])
10     The solutions which cannot dominate $z$ will be deleted; (c.f. [41])
11     The evaluations will be updated as $FEs = FEs + N$ and the $z \leftarrow min\{z, z^{max}\}$; (c.f. [41])
12     $[P, K] \leftarrow popSelectionStrategy\,(Q, V, N)$ .(c.f. Algorithm 4)
     End
     Return $P$
```

Algorithm 1 represents the framework of IhpaEA, where the first step is to initialize four parameters (1) an initial population P (line 1); (2) vectors V (line 2); (3) the number of prominent solutions (line 3) and (4) the evaluation functions (line 4); and the loop of IhpaEA (lines 5–12). Each generation performs three steps in the algorithm: (1) mating selection; (2) offspring population generation, and (3) environmental selection. The mating selection tries to assign more evolutionary results to the prominent solutions, and select better solutions. The set $\{1, 2, \ldots, K\}$ represents the prominent solutions where K standing for the number of prominent solutions, which will be firstly chosen and located in the front of the population for environmental selection. The indexes of the solutions selected for mating are denoted as tan array I. $N - K$ solutions are first randomly selected (line 6) in the current populations to form the mating pool. Although some of the prominent solutions have been selected randomly in the former step, and all K promising solutions are chosen (line 7). Finally, rearrange all the elements in the array I (line 7). Next, an Improve Similar Job Order Crossover I (ISJOXI) is used by the proposed IhpaEA to produce the offspring population (line 8), which is different from the studies [40]. Besides, population Q (line 12) performs the environmental selection strategy.

3.2 *Representation and Encoding*

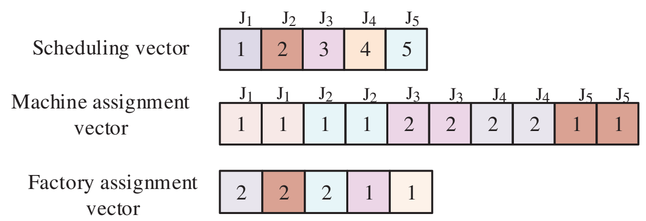
Each solution is represented by a three-dimensional vector as follows.

The first dimensional vector is called scheduling vector, and the length of it equals to the total number of operations $\Pi = \{\pi_1, \pi_2, .., \pi_n\}$. Each job number represents an element of the scheduling vector π_i, and the order of arrangement is the sequencing order.

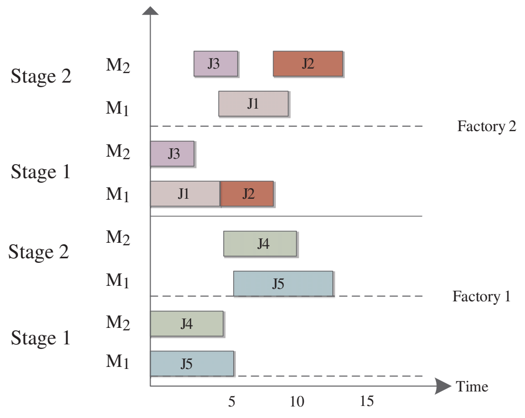
The name of the second dimensional vector is called the machine assignment vector $\delta = \{\delta_1, \delta_2, \ldots, \delta_k\}$, element δ_i of the vectors is represented by a machine number which tells the machine assigned to the corresponding job.

The third dimensional vector is named as the factory assignment vector, and the length of factory assignment vector equals to the total number of jobs $\varphi = \{\varphi_1, \varphi_2, \ldots, \varphi_n\}$, Each element of the factory assignment vector φ_i is represented by a factory number, which tells the factory assigned to the corresponding job.

Fig. 3a gives a solution representation example, where there are five jobs. The total number of stages for each job is 2. The factory assignment vector tells the factory number for each job, the routing vector reports the machine number. Then, the scheduling vector represents the scheduling sequence for each job.


Scheduling vector
J1 J2 J3 J4 J5
1 2 3 4 5
Machine assignment vector
J1 J1 J2 J2 J3 J3 J4 J4 J5 J5
1 1 1 1 2 2 2 2 1 1
Factory assignment vector
2 2 2 1 1

(a) Encoding



(b) Decoding Gantt chart

Figure 3: Solution representation

3.3 *Decoding Heuristic*

Fig. 3b shows the Gantt chart. The detailed decoding introduces are described as follows:

Step 1: The assigned jobs are scheduled based on the sequence in the scheduling vector which is the first stage of each factory.

Step 2: After determining the factory, each job should select a suitable machine following the earliest available time rule.

Step 3: For the other stages, each job is scheduled as soon as possible after completing its previous operations. The first available suitable machine is also selected.

3.4 Initialization

To solve the considered problem, a solution is encoded with two dispatching rules. The longest processing time at the first stage (LPTF) rule, and the shortest processing time at the first stage (SPTF) rule. Based on the non-increasing total processing times, LPTF generates a permutation. Meanwhile, based on the non-decreasing total processing times, SPTF produces a permutation by sorting the jobs.

To produce an effect initial population, the following technique is used. Suppose the population size is N, the detailed steps are given as follows:

(1) The first $N-2$ individuals are generated by a random way. For the factory assignment vector, each job is assigned to a random selected factory. For the scheduling vector, all the jobs are sequenced in a random order.

(2) One individual is generated by LPTF. First, all the jobs' processing time are calculated in each stage. Then, every job in every stage has a processing time and the summation of these time is called total processing time. Finally, the individual is generated by permuting the total processing time in non-increasing order.

(3) SPTF generates the last individual. The first two steps are the same with LPTF. However, the third step is to permute the processing time in a non-decreasing order.

3.5 Crossover

Based on the encoding representation, we proposed a novel crossover heuristic including two parts.

(1) PTL crossover

The first type of crossover is PTL, which can be described as follows:

Step 1: Randomly select two different elements from the first parent.

Step 2: Copy the block of jobs which are cut by the two points from the first parent. And then move the block to the rightmost or leftmost part of the offspring.

Step 3: Place the empty elements of jobs which are remaining from the second parent.

The process of PTL for generating offspring is depicted in Fig. 4. Table 1 provided an example which can figure out the way element are updated.

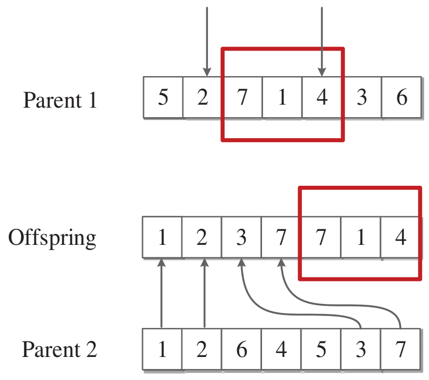


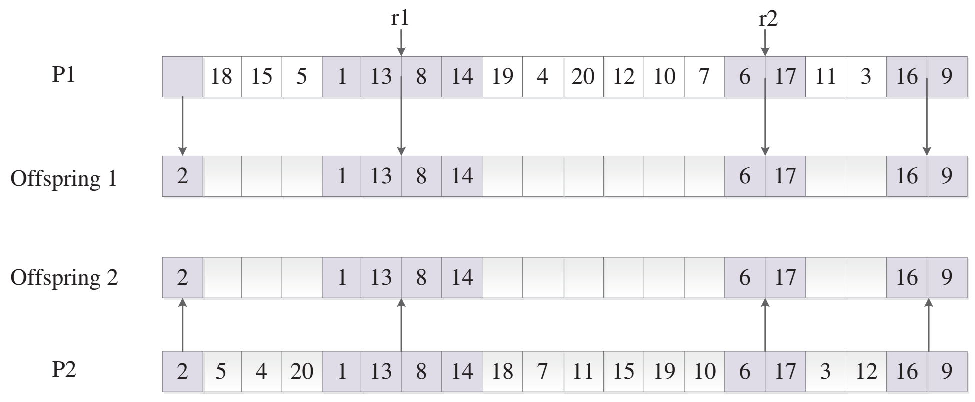
Figure 4: PTL crossover operator

Table 1: An example of PTL crossover operator

Two-cut PTL crossover						Two-cut PTL crossover						Two-cut PTL crossover					
P1	5	**1**	**4**	2	3	P1	**5**	**1**	**4**	2	3	P1	**5**	**1**	4	2	3
P2	3	5	$\underline{\mathbf{4}}$	2	**1**	P2	5	**1**	$\underline{\mathbf{4}}$	2	3	P2	3	$\underline{\mathbf{5}}$	4	2	$\underline{\mathbf{1}}$
O1	3	5	2	**1**	**4**	O1	5	2	3	**1**	**4**	O1	**5**	**1**	3	4	2
O2	**1**	**4**	3	5	2	O2	**1**	**4**	5	2	3	O2	**3**	**5**	1	4	2

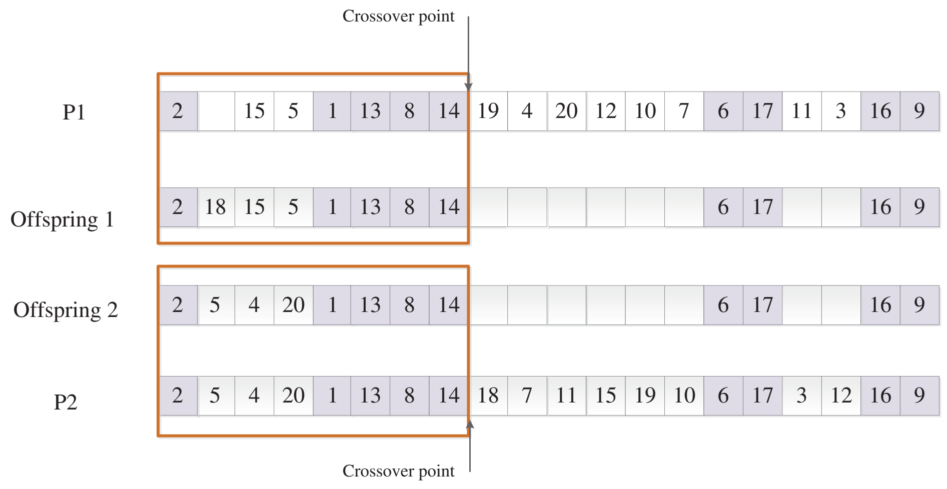
(2) ISJOXI crossover

The second type of crossover operator is Improve Similar Job Order Crossover I or ISJOXI, with which the building blocks of jobs are directly copied to the offspring. In Fig. 5a, a point is randomly selected and the elements before this cut point is copied to the offspring directly, shown in Fig. 5b. Furthermore, in order to maintain feasibility of the job sequence, the ISJOXI crossover operator copies the missing elements of each offspring which are in the relative order of the other parents, shown in Fig. 5c. Lastly, other elements which are not assigned are obtained by performing single point crossover operator on P_1 and P_2 which is chose a crossover point randomly between elements 2 and 3. In this example, r_1 and r_2 are selected as crossover points. Then, the elements between r_1 and r_2 are copied from P_1, and the elements after r_2 are copied from P_2. However, since J5 is absent, and J1 appears twice, J1 in offspring1 should be substituted with J5. As a result, offspring1 will be (1, 2, 5, 7, 4, 3, 6) and offspring 2 is generated in the same way, which is (5, 2, 4, 1, 7, 3, 6), as shown in Fig. 6.

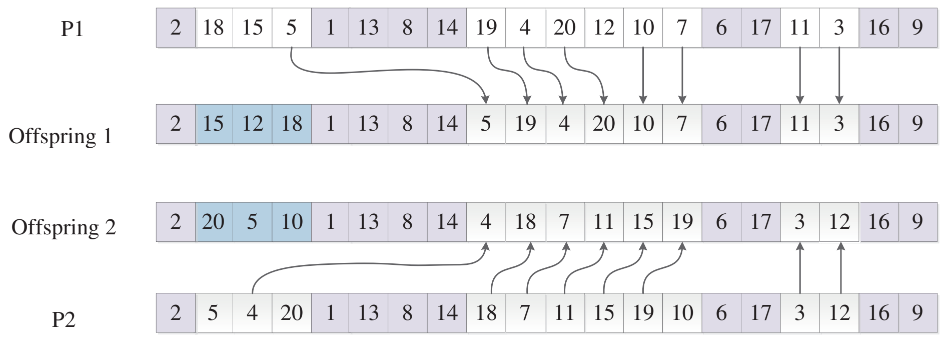


(a). Copy operator

Figure 5: (Continued)



(b). Randomly cut operator



(c). Copy missing elements

Figure 5: Process of performing ISJOXI

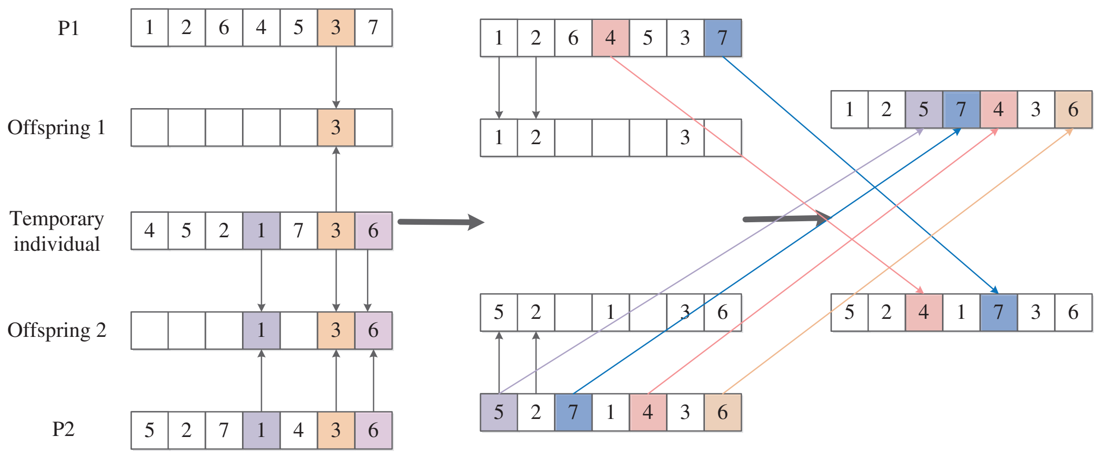


Figure 6: ISJOXI

The main steps of ISJOXI crossover are described in Algorithm 2.

Algorithm 2: ISJOXI crossover

```
Input: The current Pareto Set, the current population;
Output: Two newly-generated solutions $c_1$ and $c_2$ ;
1    Calculate the occurrence number of each job at each scheduling position;
2    Find the job with the maximum occurrence number at each scheduling position;
3    Generate a new vector $\pi_i$ , by using the job number with the maximum occurrence times at
     each position;
4    for $ind = 1$ to $PS$ do
5    Randomly select two parent individuals, named $p_1$ and $p_2$ from the current population;
6    Randomly generate two positions for the scheduling vector, named $r_1$ and $r_2$;
7    For each position $r$ between $r_1$ and $r_2$ of child individual $c_1$, let $c_1[r] = \pi_t[r]$, and the
     repeated elements are ignored. Note that $c_1[r]$ represents the element at the $r\ th$ position in
     $c_1$;
8    For each position $r$ between $r_1$ and $r_2$ of child individual $c_2$, let $c_2[r] = \pi_t[r]$, and the
     repeated elements are ignored. Note that $r'$ represents the blank position between $r_1$ and $r_2$
     of $c_2$ from left to right;
9    The blank positions before $r_1$ of the individuals $c_1$ and $c_2$ are collected directly from $p_1$ and
     $p_2$, respectively;
10     For each position $r$ before $r_1$ and after $r_2$, let $c_1[r] = c_2[r] = p[r]$, if p $[r] = p_2[r]$;
11     For the remaining blank positions of $c_1$ and $c_2$, fill it with the nonscheduled job number one
     by one from $p_1$ and $p_2$.
   end for
```

3.6 Mutation

Suppose F_c is the factory with the maximum makespan, and F_e is the factory with the maximum TEC. The mutation method for the DHFS problem is described as follows:

(1) Mutation for the factory assignment

FA_{cs}: Select two jobs J_1 and J_2 randomly, where J_1 from F_c and J_2 from a different factory, then swap the two positions of them.

FA_{es}: Select two jobs J_1 and J_2 randomly, where J_1 from F_e and J_2 from a different factory, then swap two positions of them.

FA_{ci}: Randomly insert a job which is removed from F_c into a location in a randomly selected factory.

FA_{ei}: Randomly insert a job which is removed from F_e into a location in a randomly selected factory.

(2) Mutation for the scheduling vector

JS_{cs}: Randomly choose two different jobs from F_c and then swap them.

JS_{es}: Randomly choose two different jobs from F_e and then swap them.

JS_{ci}: Insert a job which is randomly selected into a random location in F_c.

JS_{ei}: Insert a job which is randomly selected into a random location in F_e.

(3) Mutation for the machine vector

The procedure of the mutation operator is as follows. Firstly, a position r_1 is randomly selected in the machine vector. Then for the element in r_1, select a different machine.

3.7 Local Search

The following multi-objective cooperation local search operator is embedded to achieve good diversity and convergence.

First, in each generation, the maximum completion time of solution a is denoted as $\bar{C}_{max}(a) = \left(C_{max}(a) - C_{max}^{min}\right) / \left(C_{max}^{max} - C_{max}^{min}\right)$ and the TEC of each solution a is denoted as $\overline{TEC}(a) = (TEC(a) - TEC^{min}) / (TEC^{max} - TEC^{min})$ where TEC^{min}, TEC^{max}, C_{max}^{min}, *and* C_{max}^{max} represent the minimum TEC, maximum TEC, minimum makespan and maximum makespan of the solutions in current population.

Second, the value $\gamma(a) = \overline{TEC}(a) / \bar{C}_{max}(a)$ is used to calculate each solution. Then, the population is divided into two sets P_c and P_e according to the γ which are sorted in an ascending order. And the size of P_c is the same with P_e. The search operators related to F_c (factory F_c with the maximum completion time) are used because the maximum completion time of solutions in P_c (small γ) are large. The search operators related to F_e (factory F_e with the maximum TEC) are used because TEC of solutions in P_e (large γ) are large. The detailed process is described in Algorithm 3.

Algorithm 3: Cooperative Search

```
For $k = 1$ to $PS$
   Calculate $\bar{C}_{max}(a_k)$ and $\overline{\mathrm{TEC}}_{max}(a_k)$;
   Calculate $\gamma(a_k) = \overline{\mathrm{TEC}}_{max}(a_k) / \bar{C}_{max}(a_k)$;
End For
For $k = 1$ to $PS$
  If $\gamma(a_k)$ is smaller than the median of all $\gamma$
     Set $a_k \in P_c$;
     If energy consumption of $F_c$ is the largest among all factories
        Perform $FA_{cs}(a_k)$ and $FA_{ci}(a_k)$;
     Else Perform $JS_{cs}(a_k)$ and $JS_{ci}(a_k)$;
End If
Else If $\gamma(a_k)$ is larger than the median of all $\gamma$
   Set $a_k \in P_e$;
   If makespan of $F_e$ is the largest among all factories
      Perform $FA_{es}(a_k)$ and $FA_{ei}(a_k)$;
   Else Perform $JS_{es}(a_k)$ and $JS_{ei}(a_k)$;
End If
End For
```

An example is provided in Table 2, where the objective values of a population with four solutions are listed, including the C_{max} and TEC of each solution. According to the γ, these solutions are divided into P_c and P_e. From Table 2, a_1 and a_3 are put into P_c, a_2 and a_4 are put into P_e. For solution a_1, $F_e = 1$ and the makespan of F_e is medium among all factories. As a result, it performs JS_{es} and JS_{ei} in turn for solution a_1. For solution a_2, $F_c = 3$ and the TEC of F_c is the largest among all factories. Thus, it performs FA_{cs} and FA_{ci} in turn for solution a_2.

Table 2: Example for collaborative search

Solutions	Objective	f = 1	f = 2	f = 3	C_{max}	TEC	γ
a_1	Makespan	66	98	117	117	8936	76.3760
	TEC	1265	1648	6023			
a_2	Makespan	122	102	94	122	22144	181.5081
	TEC	9744	11984	416			
a_3	Makespan	290	64	71	290	26339	90.8241
	TEC	24784	532	1023			
a_4	Makespan	98	152	154	154	18502	120.1428
	TEC	3728	5112	9662			

4 Experimental Results

The computational experiments to test the performance of IhpaEA algorithm is discussed in this section. The improved algorithm was implemented in the PlatEMO v3.0 on an Intel Core i7 3.4-GHz PC with 16 GB of memory. To test the performance of IhpaEA algorithm, 20 different scales of instances are generated according to the realistic flow shop.

All the compared algorithms are used to solving the considered problem, including the encoding, and decoding method, and the initialization procedure. The parameters are set according to their literatures. For each instance, the stop condition is set to 3000 iterations.

30 independent runs are used to test the performance of the proposed algorithm, the results of non-dominated solutions found by all the compared algorithms were collected for performance comparisons. The relative percentage increase (RPI) is used for the ANOVA comparison, which is formulated as follows:

$$\text{RPI (c)} = \frac{C_c - C_b}{C_b} * 100\%$$

where C_b represents the best solution that has been calculated by all the compared algorithms and C_c is the best solution to the tested algorithm.

4.1 Experiment Parameters

20 large-scale test instances of DHFS problem are randomly generated to solve the DHFS problem and test the validity of the hpaEA algorithm based on the actual production data. For example, instance 1 can be denoted with 20 jobs, 2 stages, as well as 3 parallel machines in the first stages as well as 5 parallel machines in the second stages wherein the index of jobs are {20, 30, 50, 80, 100}, the parameter of machines are {2, 3, 4, 5}, the parameter of stages are {2, 3, 5, 10}, and the parameter of factories are {2, 3, 4, 5, 6}, respectively. The four algorithms ran 30 times.

4.2 *Efficiency of the Initialization Heuristic*

Two types of IhpaEA algorithms are coded to test the initialization heuristic discussed in Section 3.4: a random initialization heuristic named IhpaEA -NI, and IhpaEA with all components. All other components of the two comparison algorithms are the same.

Table 3 reports the comparison results between IhpaEA -NI and IhpaEA. Instance numbers are given in the first column, the HV results collected from the two compared algorithms are listed in the following two columns, respectively. The last two columns illustrate the IGD values by IhpaEA -NI and IhpaEA, respectively.

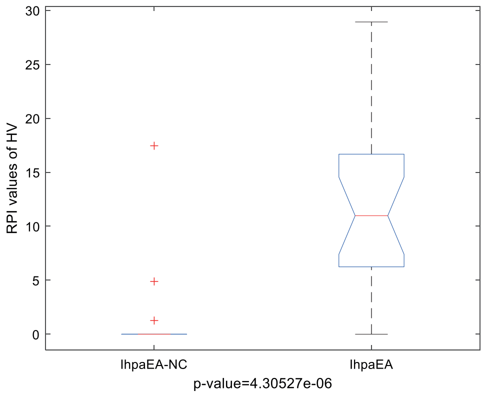
Table 3: Comparisons between IhpaEA –NI and IhpaEA

Instances	HV		IGD	
	IhpaEA –NI	IhpaEA	IhpaEA –NI	IhpaEA
Instance 1	0.4931	**0.5036**	14.2508	**12.2757**
Instance 2	0.5198	**0.5428**	3.3700	**1.8011**
Instance 3	0.5002	**0.5218**	90.9069	**2.9069**
Instance 4	0.5113	**0.5236**	38.6131	**8.6444**
Instance 5	**0.4811**	0.4809	8.5562	**2.2810**
Instance 6	0.4642	**0.4973**	8.8448	**8.7355**
Instance 7	**0.4989**	0.4964	2.8100	**2.0933**
Instance 8	0.5209	**0.5715**	844.9814	**116.680**
Instance 9	0.5029	**0.5238**	11.5812	**9.9977**
Instance 10	0.5250	**0.5373**	176.0458	**3.2249**
Instance 11	**0.5116**	0.5018	34.5704	**13.1165**
Instance 12	0.4981	**0.5148**	398.4254	**58.8273**
Instance 13	0.4963	**0.5077**	13.5969	**1.70870**
Instance 14	0.5081	**0.5205**	166.6750	**37.9270**
Instance 15	0.5037	**0.5381**	6.7975	**0.0000**
Instance 16	0.5034	**0.5259**	263.8726	**63.1717**
Instance 17	**0.5075**	0.5463	125.8685	**14.2800**
Instance 18	0.5029	**0.5232**	205.5073	**10.8245**
Instance 19	0.4954	**0.5168**	112.0908	**26.2619**
Instance 19	0.4926	**0.5137**	169.1495	**97.1709**
Instance 20	0.5229	**0.5247**	3.9088	**0.0000**
Mean	0.502852	**0.520595**	128.5916	**23.4251**

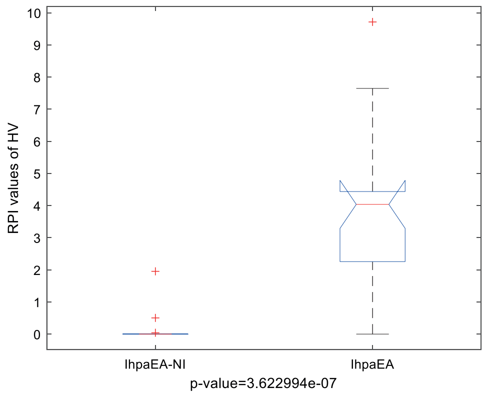
It can be concluded from the comparison results that: (1) IhpaEA algorithm obtains 16 better results by considering the HV values of the IhpaEA-NI algorithm, and the slightly worse results for the other two instances; (2) for the IGD values, IhpaEA obtains 20 better results out of the given 20 different scale instances; and (3) from the average performance in HV and IGD given in the last line and the ANOVA results from Fig. 7a, it can be seen that IhpaEA is significantly better than the IhpaEA -NI algorithm, which verify the efficiency of the proposed initialization heuristic.

4.3 *Efficiency of the Crossover Operator*

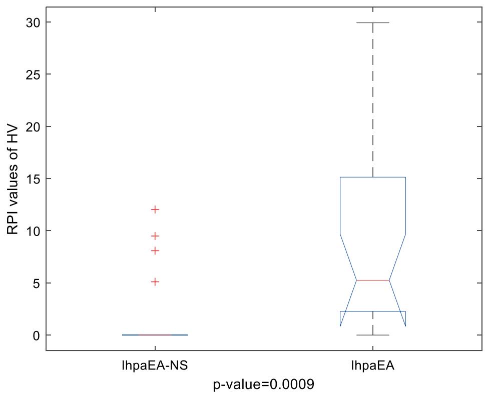
In order to test the performance of the crossover operators discussed in Section 3.5, two different types of IhpaEA algorithms are coded, i.e., the IhpaEA-NC algorithm with the classical two-point crossover, and the IhpaEA algorithm with all the two crossover operators.



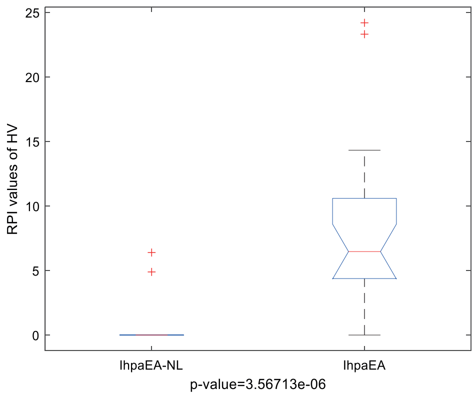
(a) ANOVA results for IhpaEA-NC and IhpaEA



(b) ANOVA results for IhpaEA-NI and IhpaEA



(c) ANOVA results for IhpaEA-NS and IhpaEA



(d) ANOVA results for IhpaEA -NL and IhpaEA

Figure 7: Means and 95% LSD interval for pairs of compared algorithms

From the comparison results given in Table 4, it can be observed that: (1) Compared with the IhpaEA -NC, IhpaEA algorithm obtains 17 better results based on the HV values; (2) the ANOVA results from Fig. 7b shows that the IhpaEA obtains significant better results based on the HV results, where the *p*-value equals to 4.30527e-06 (3) for the IGD values, IhpaEA obtains 18 better results; and (4) from the average performance in HV and IGD given in the last line, the efficiency of the proposed crossover heuristic can be verified.

Table 4: Comparisons between IhpaEA -NC and IhpaEA

Instances	HV		IGD	
	IhpaEA -NC	IhpaEA	IhpaEA -NC	IhpaEA
Instance 1	**0.6534**	0.5563	8.2956	**0.0000**
Instance 2	0.4924	**0.5647**	7.5307	**3.2319**
Instance 3	0.5455	**0.6110**	10.1593	**7.7628**
Instance 4	0.4209	**0.5236**	25.0598	**20.0341**
Instance 5	0.4673	**0.5437**	32.3624	**8.3308**
Instance 6	0.5931	**0.6582**	19.3699	**0.0000**
Instance 7	0.6289	**0.6964**	14.6482	**9.8876**
Instance 8	**0.5275**	0.5209	10.6421	**5.2935**
Instance 9	**0.6565**	0.6259	60.2584	**9.9977**
Instance 10	0.5350	**0.5670**	30.1638	**3.2249**
Instance 11	0.5316	**0.5918**	10.1558	**13.1165**
Instance 12	0.4381	**0.5648**	38.4254	**58.8273**
Instance 13	0.4963	**0.5277**	7.2531	**1.7087**
Instance 14	0.5081	**0.5565**	**12.5093**	37.927
Instance 15	0.5437	**0.5928**	5.1988	**0.0000**
Instance 16	0.5345	**0.6289**	11.2876	**5.2110**
Instance 17	0.4715	**0.5636**	**13.9689**	14.2800
Instance 18	0.5295	**0.5902**	12.3323	**9.8783**
Instance 19	0.4485	**0.5593**	7.0617	**2.3466**
Instance 19	0.5265	**0.5673**	2.1003	**0.0000**
Instance 20	0.6565	**0.6747**	15.3439	**0.0000**
Mean	0.5335	**0.5850**	16.8632	**10.05041**

4.4 Efficiency of the Mutation Operator

Two different types of IhpaEA algorithms are coded to test the performance of the mutation operator discussed in Section 3.6, i.e., the proposed IhpaEA-NS algorithm without mutation operator, and the IhpaEA algorithm with all the components.

From the comparison results given in Table 5, it can be concluded that: (1) compared with the IhpaEA-NS algorithm, IhpaEA obtains 18 better results, based on the HV values; (2) the ANOVA results from Fig. 7c shows that the IhpaEA obtains significant better results based on the HV results, where the p-value equals to 0.0009; (3) for the IGD values, IhpaEA obtains 18 better results; and (4) from the average performance in HV and IGD given in the last line, the efficiency of the proposed mutation operator can be verified.

Table 5: Comparisons between IhpaEA -NS and IhpaEA

Instances	HV		IGD	
	IhpaEA -NS	IhpaEA	IhpaEA -NS	IhpaEA
Instance 1	**0.5757**	0.5258	12.0756	**6.5329**
Instance 2	0.5801	**0.6379**	54.039	**4.3242**
Instance 3	0.4906	**0.6012**	10.6876	**3.4552**
Instance 4	0.6444	**0.6131**	23.2886	**12.0974**
Instance 5	0.4281	**0.5562**	16.7376	**0.0000**
Instance 6	0.5355	**0.5448**	7.0794	**5.9346**
Instance 7	0.5933	**0.6381**	13.0640	28.934
Instance 8	0.4768	**0.5981**	4.0355	**2.7526**
Instance 9	0.5997	**0.6312**	15.6516	**0.0000**
Instance 10	0.5249	**0.5758**	20.4162	**3.5004**
Instance 11	**0.6165**	0.5704	4.5155	**3.1654**
Instance 12	0.6273	**0.6544**	18.0674	**3.8187**
Instance 13	0.5708	**0.5969**	11.6355	**0.0000**
Instance 14	0.5927	**0.6775**	10.4286	**3.8125**
Instance 15	0.5975	**0.6289**	9.0578	**4.3771**
Instance 16	0.6717	**0.6926**	8.0385	**3.4569**
Instance 17	0.5728	**0.5868**	**4.0471**	7.1544
Instance 18	0.4824	**0.5673**	2.9527	**0.0000**
Instance 19	0.6619	0.5908	5.0999	**4.8534**
Instance 19	0.5709	**0.6495**	**2.0368**	2.6596
Instance 20	0.5088	**0.6587**	6.3669	**5.6507**
Mean	0.5677	**0.6093**	12.3486	**5.0704**

4.5 Efficiency of the Local Search Operator

To evaluate the performance of the local search heuristic discussed in Section 3.7, two types of IphaEA algorithms are coded: IhpaEA-NL without the local search heuristic and IhpaEA with all components that mentioned in Section 3.7.

From the comparison results given in Table 6, it can be observed that: (1) considering the HV values, compared with the IhpaEA -NL algorithm, IhpaEA obtains 18 better results; (2) the ANOVA results from Fig. 7d shows that the IhpaEA obtains significant better results considering the HV results, where the *p*-value equals to 3.56712e-06 (3) for the IGD values, IhpaEA obtains 20 better results; and (4) from the average performance in HV and IGD given in the last line, the efficiency of the proposed heuristic can be verified.

Table 6: Comparisons between IhpaEA -NL and IhpaEA

Instances	HV		IGD	
	IhpaEA -NL	IhpaEA	IhpaEA -NL	IhpaEA
Instance 1	0.6162	**0.6416**	14.2508	**12.2757**
Instance 2	0.5895	**0.6158**	3.3700	**1.8011**
Instance 3	0.5879	**0.6721**	90.9069	**2.9069**
Instance 4	0.6068	**0.6479**	38.6131	**8.6444**
Instance 5	0.5673	**0.5970**	8.5562	**2.2810**
Instance 6	0.5932	**0.6547**	8.8448	**8.7355**
Instance 7	0.5489	**0.6817**	2.8100	**2.0933**
Instance 8	0.5526	**0.6148**	844.9814	**116.680**
Instance 9	**0.6048**	0.5766	11.5812	**9.9977**
Instance 10	0.6107	**0.6890**	176.0458	**3.2249**
Instance 11	0.6015	**0.6557**	34.5704	**13.1165**
Instance 12	0.6049	**0.6580**	398.4254	**58.8273**
Instance 13	0.6523	**0.6861**	13.5969	**1.70870**
Instance 14	0.5873	**0.6204**	166.6750	**37.9270**
Instance 15	0.5671	**0.6993**	6.7975	**0.0000**
Instance 16	0.6040	**0.6429**	263.8726	**63.1717**
Instance 17	0.5722	**0.6242**	125.8685	**14.2800**
Instance 18	0.6153	**0.6337**	205.5073	**10.8245**
Instance 19	**0.5942**	0.5585	112.0908	**26.2619**
Instance 19	0.5798	**0.6173**	169.1495	**97.1709**
Instance 20	0.6544	**0.6747**	3.9088	**0.0000**
Mean	0.0606	**0.3911**	128.5916	**23.4251**

4.6 Comparisons of the Efficient Algorithms

Three algorithms are selected, namely, NSGAII [37], GFMOEA [38], BiGE [39], to test the effectiveness of the IhpaEA algorithm. Table 7 presents the HV and IGD results after 30 independent runs.

Table 7: Comparisons results of the HV values (NSGAII, BiGE, GFMMOEA, IhpaEA)

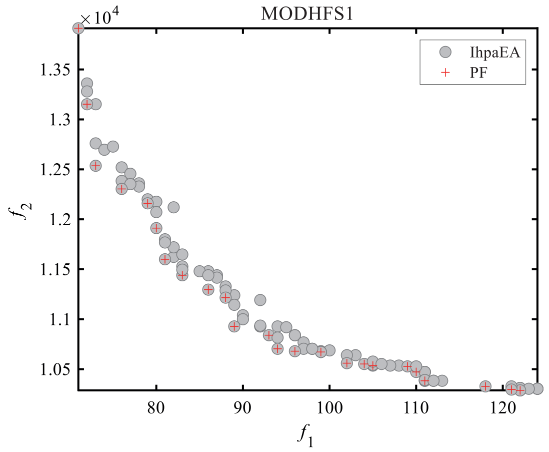
Instance	HV			
	NSGAII	BiGE	GFMMOEA	IhpaEA
Instance 1	0.0653	0.0000	0.0653	**0.4763**
Instance 2	0.0432	0.0118	0.0440	**1.8447**
Instance 3	0.0345	0.0000	0.0387	**12.1810**
Instance 4	0.1209	11.0856	0.1288	**6.5236**
Instance 5	0.1673	0.0000	0.1795	**7.2437**

(Continued)

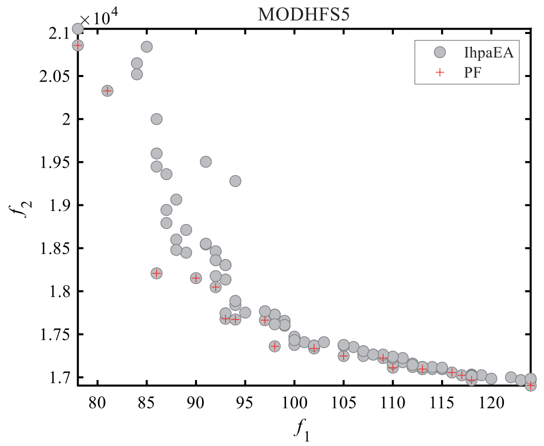
Table 7 (continued)

Instance	HV			
	NSGAII	BiGE	GFMMOEA	IhpaEA
Instance 6	0.0593	0.0000	0.0594	**0.2582**
Instance 7	0.0289	0.0000	0.0340	**17.6964**
Instance 8	0.0275	7.6034	0.0255	0.0000
Instance 9	0.1565	0.0000	0.1648	**5.3259**
Instance 10	0.035	**14.3026**	0.0306	0.0000
Instance 11	0.0316	0.0000	0.0341	**7.9018**
Instance 12	0.0381	0.3611	0.0374	**2.0480**
Instance 13	0.1163	0.0000	0.1168	**0.4277**
Instance 14	0.0381	**6.312**	0.0358	0.0000
Instance 15	0.0437	0.0118	**0.0468**	0.0000
Instance 16	0.0345	0.0000	0.0345	**2.8302**
Instance 17	0.0715	**5.0329**	0.0681	0.9585
Instance 18	0.0295	**2.8113**	0.0303	0.0000
Instance 19	0.0485	0.0000	0.0489	**12.6747**
Instance 20	**0.0265**	0.0000	0.0258	0.0000
Mean	0.063269	2.4609	0.064	**3.8171**

Figs. 8a–8c shows the Pareto front charts for solving three different problem scale instances, i.e., "Instance 1", "Instance 5", and "Instance 20". It can be observed from Fig. 8 that, the solutions obtained by the IhpaEA algorithm are close to the Pareto front and well-distributed.

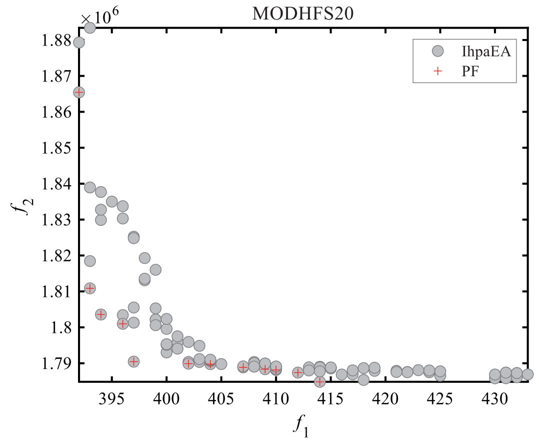


(a) Pareto front for "Instance 1"



(b) Pareto front for "Instance 5"

Figure 8: (Continued)



(c) Pareto front for "Instance 20"

Figure 8: Pareto front results

Table 7 reports the comparison results of the HV values for the given 20 different scale instances. The first column represents the number of the instances. Then, the results collected by NSGAII, GFMMOEA, BiGE, and IhpaEA, are illustrated in the following four columns, respectively. Table 8 reveals that: (1) 13 better values obtained by the proposed IhpaEA algorithm for the given 20 instances perform significantly better than other 3 comparison algorithms; and (2) the average values of the last line further evaluate the efficiency of the IhpaEA. In addition, from the compared results of the IGD values reported in Table 8, it can be known that the proposed IhpaEA algorithm gets 19 better values, which further test the superiority of the IhpaEA algorithm.

Table 8: Comparisons of the IGD values (NSGAII, BiGE, GFMMOEA, IhpaEA)

Instance	IGD			
	NSGAII	BiGE	GFMMOEA	IhpaEA
Instance 1	116.2298	158.8989	36.7109	**0.0000**
Instance 2	8.1941	34.8277	32.5033	**0.0000**
Instance 3	83.8725	22.5379	**0.0000**	59.8702
Instance 4	19.8748	94.4182	31.8385	**10.8258**
Instance 5	47.9635	179.0395	273.2809	**3.0955**
Instance 6	12.8869	170.6979	36.779	**1.2823**
Instance 7	19.4737	18.5112	277.2231	**4.7008**
Instance 8	75.6480	67.4810	28.3924	**25.1631**
Instance 9	73.9556	70.0808	265.1931	**53.1107**
Instance 10	64.1465	41.8955	16.3510	**1.4966**
Instance 11	53.4103	24.1171	81.9518	**8.2834**
Instance 12	19.5608	36.0141	112.583	**0.0000**
Instance 13	41.2228	33.5263	96.2000	**20.7798**
Instance 14	93.9044	78.7471	137.8545	**38.0666**

(Continued)

Table 8 (continued)

Instance	IGD			
	NSGAII	BiGE	GFMMOEA	IhpaEA
Instance 15	57.8876	50.251	98.1652	**4.2185**
Instance 16	13.8127	8.3406	56.0629	**5.8542**
Instance 17	80.9432	79.2301	41.8200	**40.6400**
Instance 18	83.1895	31.1598	18.7015	**3.4795**
Instance 19	23.9740	43.5960	157.5631	**23.2828**
Instance 20	37.4045	16.7215	29.9731	**0.0000**
Mean	51.3776	63.0046	91.4573	**15.2074**

A multifactor analysis of variance (ANOVA) is also performed to verify the difference from the above tables, based on three compared algorithms namely, NSGAII, GFMMOEA and BiGE. Fig. 9 reveals the means and the 95% LSD (Fisher's Least Significant Difference) intervals for the best values of the three compared algorithms. The result of the three comparison algorithms shows statistically significant difference. From Fig. 9 we can conclude that the IhpaEA algorithm performs better than other three compared algorithms obviously.

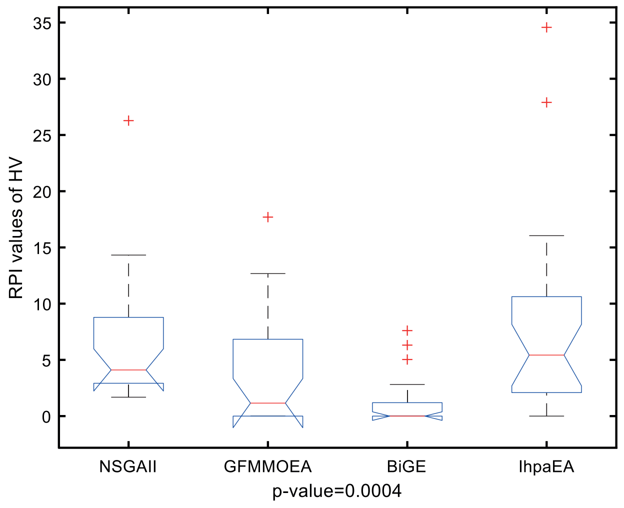


Figure 9: Multi-compare results for NSGAII, GFMMOEA, BiGE, and IhpaEA

Fig. 10 shows the Gantt chart for one of the optimal solutions for "Instance 7". It can be concluded from Fig. 10 that the proposed algorithm can obtain some feasible and efficient solutions for the considered problem.

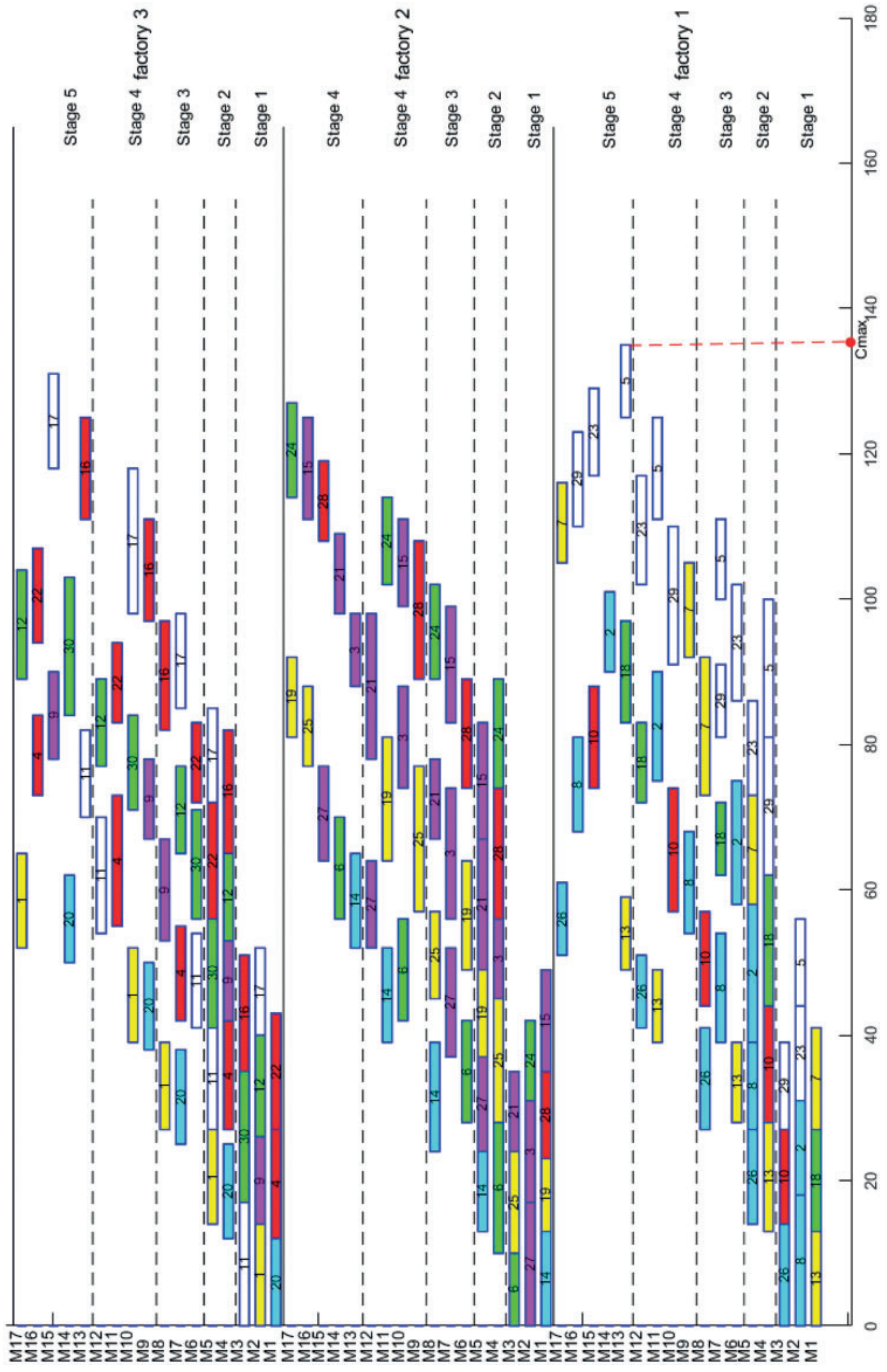


Figure 10: Gantt charts for instance 7

4.7 Experimental Analysis

From the above discussed comparison results, the efficient performance of the proposed IhpaEA algorithm has been tested. The main advantages of IhpaEA are as follows: (1) the proposed initialization heuristic, which can enhance the population diversity and quality; (2) the Pareto-based crossover operator enhance the global search abilities; (3) the mutation operator enhance the convergence of optimization process; and (4) a cooperation of search operators improve the local search abilities.

5 Conclusion

This paper studies a DHFS problem with makespan and total energy consumption. To solving the problem, a multiobjective optimization algorithm is proposed. The contributions are as follows: (1) an efficient encoding and decoding mechanism is embedded; (2) in the initialization phase of the algorithm, considering the constraints in the model, two heuristics are developed; (3) a Pareto-based crossover operator is designed; and (4) a cooperation of search operator is developed to further improve the quality of the solution and accelerate the convergence speed of the algorithm. To further illustrate the effectiveness of the proposed algorithm, IhpaEA is compared with three other multi-objective algorithms, including NSGA-II, BiGE, and GFMMOEA, The Pareto frontier is closer to the optimal solution than the other three algorithms.

We test the IhpaEA algorithm with different scales and compare several efficient algorithms with the IhpaEA algorithm. The robustness as well as efficiency is shown by experimental results. There are some works need to be focused as follows: (1) to improve the search capabilities, more local optimization methods or other efficient heuristics need to be introduced; (2) some useful dynamic and rescheduling strategies should be considered in flow shop scheduling problem; (3) more conflict objectives such as maximum workload and parallel batch workload need to be focused on.

Funding Statement: The authors received no specific funding for this study.

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

References

1. Ruiz, R., Vázquez-Rodríguez, J. A. (2010). The hybrid flow shop scheduling problem. *European Journal of Operational Research, 205(1),* 1–18. DOI 10.1016/j.ejor.2009.09.024.
2. Liu, M., Yang, X., Chu, F., Zhang, J., Chu, C. (2020). Energy-oriented bi-objective optimization for the tempered glass scheduling. *Omega, 90,* 101995. DOI 10.1016/j.omega.2018.11.004.
3. Chung, T. P., Sun, H., Liao, C. J. (2017). Two new approaches for a two-stage hybrid flowshop problem with a single batch processing machine under waiting time constraint. *Computers & Industrial Engineering, 113,* 859–870. DOI 10.1016/j.cie.2016.11.031.
4. Pan, Q. K., Gao, L., Wang, L. (2019). A Multi-objective hot-rolling scheduling problem in the compact strip production. *Applied Mathematical Modelling, 73,* 327–348. DOI 10.1016/j.apm.2019.04.006.
5. Xu, Y., Wang, L., Wang, S., Liu, M. (2014). An effective hybrid immune algorithm for solving the distributed permutation flow-shop scheduling problem. *Engineering Optimization, 46(9),* 1269–1283. DOI 10.1080/0305215X.2013.827673.
6. Bargaoui, H., Driss, O. B., Ghédira, K. (2017). A novel chemical reaction optimization for the distributed permutation flowshop scheduling problem with makespan criterion. *Computers & Industrial Engineering, 111,* 239–250. DOI 10.1016/j.cie.2017.07.020.

7. Pan, Q. K., Gao, L., Wang, L. (2021). An effective cooperative co-evolutionary algorithm for distributed flowshop group scheduling problems. *IEEE Transactions on Cybernetics*, *52(7),* 5999–6012. DOI 10.1109/tcyb.2020.3041494.

8. Huang, J. P., Pan, Q. K., Miao, Z. H., Gao, L. (2021). Effective constructive heuristics and discrete bee colony optimization for distributed flowshop with setup times. *Engineering Applications of Artificial Intelligence, 97,* 104016. DOI 10.1016/j.engappai.2020.104016.

9. Meng, T., Pan, Q. K. (2021). A distributed heterogeneous permutation flowshop scheduling problem with lot-streaming and carryover sequence-dependent setup time. *Swarm and Evolutionary Computation, 60,* 100804. DOI 10.1016/j.swevo.2020.100804.

10. Ying, K. C., Lin, S. W., Cheng, C. Y., He, C. D. (2017). Iterated reference greedy algorithm for solving distributed no-idle permutation flowshop scheduling problems. *Computers & Industrial Engineering, 110,* 413–423. DOI 10.1016/j.cie.2017.06.025.

11. Hao, J. H., Li, J. Q., Du, Y., Song, M. X., Duan, P. et al. (2019). Solving distributed hybrid flowshop scheduling problems by a hybrid brain storm optimization algorithm. *IEEE Access, 7,* 66879–66894. DOI 10.1109/Access.6287639.

12. Cai, J., Zhou, R., Lei, D. (2020). Dynamic shuffled frog-leaping algorithm for distributed hybrid flow shop scheduling with multiprocessor tasks. *Engineering Applications of Artificial Intelligence, 90,* 103540. DOI 10.1016/j.engappai.2020.103540.

13. Shao, Z., Shao, H., S, W., Pi, D. C. (2021). Effective constructive heuristic and iterated greedy algorithm for distributed mixed blocking permutation flow-shop scheduling problem. *Knowledge-Based Systems, 221,* 106959. DOI 10.1016/j.knosys.2021.106959.

14. Li, W., Chen, X., Li, J., Sang, H., Han, Y. et al. (2022). An improved iterated greedy algorithm for distributed robotic flowshop scheduling with order constraints. *Computers & Industrial Engineering, 164,* 107907. DOI 10.1016/j.cie.2021.107907.

15. Jiang, E., Wang, L., Wang, J. (2021). Decomposition-based multi-objective optimization for energy-aware distributed hybrid flow shop scheduling with multiprocessor tasks. *Tsinghua Science and Technology, 26(5),* 646–663. DOI 10.26599/TST.2021.9010007.

16. Niu, W., Li, J. Q., Jin, H., Qi, R., Sang, H., Y. (2022). Bi-objective optimization using improved NSGA-II for energy-efficient scheduling of distributed assembly blocking flow shop. *Engineering Optimization* (in press). DOI 10.1080/0305215X.2022.2032017.

17. Qin, H., Li, T., Teng, Y., Wang, K. (2021). Integrated production and distribution scheduling in distributed hybrid flow shops. *Memetic Computing, 13(2),* 185–202. DOI 10.1007/s12293-021-00329-6.

18. Khan, U., Ahmed, N., Mohyud-Din, S. T., Alharbi, S. O., Khan, I. (2022). Thermal improvement in magnetized nanofluid for multiple shapes nanoparticles over radiative rotating disk. *Alexandria Engineering Journal, 61(3),* 2318–2329. DOI 10.1016/j.aej.2021.07.021.

19. Alqahtani, A. M., Khan, U., Ahmed, N., Mohyud-Din, S. T., Khan, I. (2020). Numerical investigation of heat and mass transport in the flow over a magnetized wedge by incorporating the effects of cross-diffusion gradients: Applications in multiple engineering systems. *Mathematical Problems in Engineering*, *2020,* 2475831. DOI 10.1155/2020/2475831.

20. Ahmed, N., Khan, U., Mohyud-Din, S. T., Chu, Y. M., Khan, I., Nisar, K. S. (2020). Radiative colloidal investigation for thermal transport by incorporating the impacts of nanomaterial and molecular diameters (dNanoparticles, dFluid): Applications in multiple engineering systems. *Molecules, 25(8),* 1896. DOI 10.3390/molecules25081896.

21. Pan, Q. K., Wang, L. (2012). Effective heuristics for the blocking flowshop scheduling problem with makespan minimization. *Omega, 40(2),* 218–229. DOI 10.1016/j.omega.2011.06.002.

22. Na, B., Ahmed, S., Nemhauser, G., Sokol, J. (2013). Optimization of automated float glass lines. *International Journal of Production Economics, 145(2),* 561–572. DOI 10.1016/j.ijpe.2013.04.024.

23. Lozano, A. J., Medaglia, A. L. (2014). Scheduling of parallel machines with sequence-dependent batches and product incompatibilities in an automotive glass facility. *Journal of Scheduling, 17(6),* 521–540. DOI 10.1007/s10951-012-0308-7.

24. Wang, S., Liu, M., Chu, F., Chu, C. (2016). Bi-objective optimization of a single machine batch scheduling problem with energy cost consideration. *Journal of Cleaner Production, 137,* 1205–1215. DOI 10.1016/j.jclepro.2016.07.206.

25. Wang, S., Wang, X., Chu, F., Yu, J. (2020). An energy-efficient two-stage hybrid flow shop scheduling problem in a glass production. *International Journal of Production Research, 58(8),* 2283–2314. DOI 10.1080/00207543.2019.1624857.

26. Li, J., Sang, H., Pan, Q., Duan, P., Gao, K. (2017). Solving multi-area environmental/economic dispatch by pareto-based chemical-reaction optimization algorithm. *IEEE/CAA Journal of Automatica Sinica, 6(5),* 1240–1250. DOI 10.1109/JAS.6570654.

27. Marichelvam, M. K., Prabaharan, T., Yang, X. S. (2014). Improved cuckoo search algorithm for hybrid flow shop scheduling problems to minimize makespan. *Applied Soft Computing, 19,* 93–101. DOI 10.1016/j.asoc.2014.02.005.

28. Huang, R. H., Yang, C. L., Hsu, C. T. (2015). Multi-objective two-stage multiprocessor flow shop scheduling–a subgroup particle swarm optimisation approach. *International Journal of Systems Science, 46(16),* 3010–3018. DOI 10.1080/00207721.2014.886742.

29. Wang, S., Liu, M. (2014). Two-stage hybrid flow shop scheduling with preventive maintenance using multi-objective tabu search method. *International Journal of Production Research, 52(5),* 1495–1508. DOI 10.1080/00207543.2013.847983.

30. Li, J. Q., Liu, Z. M., Li, C., Zheng, Z. X. (2020). Improved artificial immune system algorithm for type-2 fuzzy flexible job shop scheduling problem. *IEEE Transactions on Fuzzy Systems, 29(11),* 3234–3248. DOI 10.1109/TFUZZ.2020.3016225.

31. Li, J. Q., Song, M. X., Wang, L., Duan, P. Y., Han, Y. Y. et al. (2019). Hybrid artificial bee colony algorithm for a parallel batching distributed flow-shop problem with deteriorating jobs. *IEEE Transactions on Cybernetics, 50(6),* 2425–2439. DOI 10.1109/TCYB.6221036.

32. Shahvari, O., Logendran, R. (2016). Hybrid flow shop batching and scheduling with a bi-criteria objective. *International Journal of Production Economics, 179,* 239–258. DOI 10.1016/j.ijpe.2016.06.005.

33. Zhang, G., Xing, K., Cao, F. (2018). Scheduling distributed flowshops with flexible assembly and set-up time to minimise makespan. *International Journal of Production Research, 56(9),* 3226–3244. DOI 10.1080/00207543.2017.1401241.

34. Wang, H., Huang, M., Wang, J. (2019). An effective metaheuristic algorithm for flowshop scheduling with deteriorating jobs. *Journal of Intelligent Manufacturing, 30(7),* 2733–2742. DOI 10.1007/s10845-018-1425-8.

35. Li, J. Q., Sang, H. Y., Han, Y. Y., Wang, C. G., Gao, K. Z. (2018). Efficient multi-objective optimization algorithm for hybrid flow shop scheduling problems with setup energy consumptions. *Journal of Cleaner Production, 181,* 584–598. DOI 10.1016/j.jclepro.2018.02.004.

36. Li, J., Chen, X. L., Duan, P., Mou, J. H. (2021). KMOEA: A knowledge-based multi-objective algorithm for distributed hybrid flow shop in a prefabricated system. *IEEE Transactions on Industrial Informatics, 18(8),* 5318–5329. DOI 10.1109/TII.2021.3128405.

37. Du, Y., Li, J. Q., Chen, X. L., Duan, P. Y., Pan, Q. K. (2022). Knowledge-based reinforcement learning and estimation of distribution algorithm for flexible job shop scheduling problem. *IEEE Transactions on Emerging Topics in Computational Intelligence,* 1–15. DOI 10.1109/TETCI.2022.3145706.

38. Mou, J., Duan, P., Gao, L., Liu, X., Li, J. (2022). An effective hybrid collaborative algorithm for energy-efficient distributed permutation flow-shop inverse scheduling. *Future Generation Computer Systems, 128,* 521–537. DOI 10.1016/j.future.2021.10.003.

39. Li, J. Q., Du, Y., Gao, K. Z., Duan, P. Y., Gong, D. W. et al. (2021). A hybrid iterated greedy algorithm for a crane transportation flexible job shop problem. *IEEE Transactions on Automation Science and Engineering, 19(3),* 2153–2170. DOI 10.1109/TASE.2021.3062979.
40. Chen, H., Tian, Y., Pedrycz, W., Wu, G., Wang, R. et al. (2019). Hyperplane assisted evolutionary algorithm for many-objective optimization problems. *IEEE Transactions on Cybernetics, 50(7),* 3367–3380. DOI 10.1109/TCYB.6221036.
41. Han, Y., Gong, D., Jin, Y., Pan, Q. (2017). Evolutionary multiobjective blocking lot-streaming flow shop scheduling with machine breakdowns. *IEEE Transactions on Cybernetics, 49(1),* 184–197. DOI 10.1109/TCYB.2017.2771213.