Knowledge-Based Systems 258 (2022) 109962

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Intelligent optimization under blocking constraints: A novel iterated greedy algorithm for the hybrid flow shop group scheduling problem

Haoxiang Qin [a], Yuyan Han [a,*], Yuting Wang [a,*], Yiping Liu [b], Junqing Li [a,c], Quanke Pan [d]

[a] School of Computer Science, Liaocheng University, Liaocheng, 252059, China
[b] The College of Computer Science and Electronic Engineering, Hunan University, 410082, China
[c] School of Computer Science, Shandong Normal University, Jinan, 250014, China
[d] School of Mechatronic Engineering and Automation, Shanghai University, Shanghai 200072, China

ARTICLE INFO

Article history:
Received 23 August 2022
Received in revised form 27 September 2022
Accepted 28 September 2022
Available online 5 October 2022

Keywords:
Hybrid flow shop group scheduling problem
Blocking
Iterated greedy algorithm
Neighborhood probabilistic selection strategies
Makespan

ABSTRACT

This paper introduces a new flow shop combinatorial optimization problem, called the blocking hybrid flow shop group scheduling problem (BHFGSP). In the problem, no buffers exist between any adjacent machines, and a set of jobs with different sequence-dependent setup times needs to be scheduled and processed at organized manufacturing cells. We verify the correctness of the mathematical model of BHFGSP by using CPLEX. In this paper, we proposed a novel iterated greedy algorithm to solve the problem. The proposed algorithm has two key techniques. One is the decoding procedure that calculates the makespan of a job sequence, and the other is the neighborhood probabilistic selection strategies with families and blocking-based jobs. The performance of the proposed algorithm is investigated through a large number of numerical experiments. Comprehensive results show that the proposed algorithm is effective in solving BHFGSP.

1. Introduction

Intelligent optimization and scheduling for complex production flow shops play an important role in achieving smart manufacturing [1–3]. The hybrid flow shop scheduling problem (HFSP) is an extension of the traditional permutation flow shop scheduling problem (PFSP). It has been attracting much attention in recent years [4,5]. The relevant research mainly concentrated on designing rigorous scheduling mathematical models and effective algorithms for HFSP, and had obtained noteworthy achievements [6–8]. Unlike the PFSP, in HFSP, the number of machines in at least one stage is greater than one. In addition, the PFSP assumes infinite buffers between adjacent machines, where jobs can be stored for an unlimited time.

Blocking constraints generally exist in various production industries. Because multiple factors exist in real-world production, such as technical conditions, production costs, and product characteristics, this situation often causes the blocking of jobs. In jobs' blocking, a preceding machine holds its currently finished job until the machine in the next stage is available [9–11]. Some examples of blocking can be found due to the effects of product attributes. In the process of concrete block production [12], the concrete cannot be stored to avoid cracks. In the process of chemical production [13–15], blocking conditions also occur because there is no buffer to store the processed jobs. In order to solve the problem with blocking constraints, many industrial production plants have begun utilizing job sequencing strategies. By using these strategies, the production efficiency is improved [16,17]. It can be seen from the above-mentioned studies, that designing an efficient job scheduling algorithm can help enterprises improve their production efficiency.

Family constraints are also prevalent in manufacturing production. To improve the flexibility and efficiency of the production, machines in some workshops are divided into organized manufacturing units. The specific set of products produced by each manufacturing unit is called a job family. In order to handle job families with similar processing attributes, companies use some grouping scheduling techniques. By applying group scheduling techniques, the setup time of machine and the completion time of product can be effectively reduced [18]. In real world, group scheduling techniques have been widely used in many manufacturing plants, such as upholstered furniture [19], printed circuit boards [20], and automotive paint shops [21]. Moreover, in recent years, group scheduling techniques have been applied to solve distributed flow shop scheduling problems [22].

HFSP has been proved to be an NP-hard problem [23,24]. Traditional mathematical methods are difficult to solve this problem. Studies mentioned above show that in the HFSP, blocking and

* Corresponding authors.
E-mail addresses: hanyuyan@lcu-cs.com (Y. Han), wangyuting@lcu-cs.com (Y. Wang).

https://doi.org/10.1016/j.knosys.2022.109962
0950-7051/

family constraints are widespread in the production process of modern manufacturing. By decreasing the impact of the blocking and family constraints, companies can significantly increase their productivity and effectively reduce their production costs. In the current research, some intelligent optimization methods have been proposed to solve the problems related to the BHFGSP [10, 25]. However, no literature has designed a mathematical model with both blocking and family constraints in HFSP. In enterprises, the problem of blocking and the application of group technology are urgent issues that need the attention of decision makers in the production process. Existing technologies or strategies cannot cope with the above two scenarios at the same time. Therefore, it is necessary to develop an effective mathematical model for BHFGSP, and design an intelligent optimization algorithm to solve this problem. We hope this study can not only fill the knowledge gap in the scientific research field but also guide the production scheduling of enterprises.

In this paper, first, a mathematical model of BHFGSP is designed. Then, a novel iterated greedy (IG) algorithm is proposed to minimize the makespan of the BHFGSP. Compared to existing intelligent optimization algorithms, the basic IG algorithm [26] has the characteristics of fewer parameters, simple structure, and strong local search ability. Thus, in this study, we further improved the IG algorithm to solve the BHFGSP.

The main contributions of this paper are shown as follows.

(1) We propose the mathematical model of BHFGSP to minimize the makespan, then use the CPLEX to verify the correctness of the model.
(2) New encoding and decoding procedures are developed to illustrate the scheduling process. Through this process, the calculation of the makespan can be known.
(3) We design the neighborhood probabilistic selection strategies based on the family and blocking constraints of job sequence, simultaneously.
(4) Neighborhood probabilistic selection strategies with family and blocking-based jobs are designed to reduce the makespan of the job sequence, respectively.
(5) Eight swap operators are presented and embedded into the neighborhood probability selection strategies. Those operators can improve the global and local search abilities.

The rest of this paper is organized as follows. The related literature on this problem is reviewed in Section 2. In Section 3, a mathematical model of the BHFGSP is formulated. In Section 4, a novel IG algorithm which contains the neighborhood probabilistic selection strategies with family and blocking-based job is presented. The model validation, simulation, and analysis are implemented in Section 5. In Section 6, we give a conclusion of this paper and propose prospects for future research.

2. Literature review

In this section, we review related problems of the BHFGSP, i.e., HFSP, BHFSP, and FGSP. HFSP is a combinatorial optimization problem involving parallel machines. Then, the blocking constraint is considered in HFSP. In addition, considering the family constraint, the research of FGSP is reviewed. For solving the problems mentioned above, many scholars have proposed corresponding algorithms accordingly based on different optimization objectives. The details are as follows.

In the modern manufacturing system, HFSP has a wide range of applications [27,28]. In these applications, parallel machine production mode can effectively improve the throughput of enterprises and reduce the impact of the bottleneck stage. To solve the HFSP, researchers have developed many efficient intelligent optimization algorithms for different objectives. Tang and Wang [29], Pan and Wang [30] enjoyed more attention. They demonstrated the effectiveness of metaheuristic algorithms, i.e., particle swarm optimization (PSO), and discrete artificial bee colony (DABC) algorithm for solving the HFSP with minimizing the total weighted completion time and makespan, respectively. Although they have been shown good performance in finding optimal solutions for many instances, Li et al. [5,31], and Zhang [32] realized the need of introducing more effective strategies to solve multiobjective optimization problems. In particular, DABC, fruit fly optimization, and three-stage multiobjective algorithms select non-dominated solutions with good convergence and distribution through a staged mechanism. According to the above-mentioned literature, all algorithms perform well for solving HFSP, but they do not design corresponding strategies based on the family and blocking constraints. Refer to the relevant literature, constraint-based strategies are designed to further optimize the makespan criterion.

In BHFSP, there is no intermediate buffer between any adjacent machines. When a job is finished in the previous stage, the job will be blocked on the current machine if machines are not idle in the next stage. Trabelsi et al. [33] designed a mathematical model of BHFSP, and gave the lower bound of the problem. However, they did not design algorithms to test large-scale instances. Later, for solving the large-scale BHFSP, some classical intelligent optimization algorithms are presented, such as GA [17], DABC [34] and SA [35]. These algorithms can take advantage of parallel processing resources of machines and can effectively deal with blocking constraints. To optimize other objectives, i.e., due date window [36], total tardiness and earliness [10], and energy consumption [37], some algorithms are improved or modified for solving BHFSP. The improvement mainly develops the exploration capability of the proposed algorithm. Three common features of these algorithms are as the follows: (1) adopting the iterated greedy algorithm, (2) using the greedy adaptive search strategies, and (3) designing perturbation operators for improving the local and global abilities. These modifications make it possible to design a novel neighborhood structure constituting hybrid optimization algorithms [10]. Although the above-mentioned literature has proved that hybrid optimization algorithm is a good design idea and has successfully solved the HFSP with blocking constraints, the hybrid optimization algorithm has not solved well the HFSP with blocking and group conditions unconsidered in the above literature.

Studies mentioned above considered different optimization objectives of BHFSP and proposed many efficient methods to solve this problem. However, few strategies are designed to change the job ordering according to blocking characteristics. Most of the existing research optimized the goals at the level of job sequencing, with little consideration of the blocking constraints. This paper makes up for the deficiency in the strategy design. In the proposed neighborhood probabilistic selection strategy with blocking-based jobs, all operators are designed based on the blocking constraints.

In recent years, FGSP has caused more and more attention, and its applications are also increasing [38–40]. As seen from the research mentioned above, the grouping scheduling techniques can effectively reduce the machine setup time and shorten production time, thereby improving product productivity [41]. Up to now, many studies have been done for solving FGSP. Neufeld et al. [39] improved the constructive heuristic algorithms to reduce the makespan. Then, Costa et al. [42] considered the hybrid metaheuristic genetic algorithms to solve FGSP. Simulation results further confirm the effectiveness of the hybrid optimization algorithm, Later, the authors also introduce blocking constraints into FGSP and use a meta-heuristic approach to solve the problem [43], but the literature does not consider the parallel machine environment. Our paper further explores and extends the problem by considering parallel machines, blocking constraints, and grouping techniques. Liou et al. [44] also combined the PSO and GA algorithms to solve the FGSP. In literature [45], a hybrid genetic and a simulated annealing algorithm are proposed to minimize the total completion time. Keshavarz et al. [46] proposed a hybrid meta-heuristic algorithm based on PSO to minimize the objectives of tardiness and total weighted earliness. Inspired by the above ideas, Our paper also adopts the same idea of hybrid optimization algorithm and makes innovation on the basis of IG algorithm, which not only designs the corresponding local search strategy for the problem characteristics but also improves the computational efficiency of IG algorithm.

The algorithms mentioned above can solve the FGSP with different objectives and achieve good results. However, there is no literature to solve the HFGSP with blocking constraints. Therefore, this paper first proposes a mathematical model of BHFGSP. Then, the decoding method is developed to calculate the job sequence's makespan. Finally, a novel IG algorithm which contains eight swap operators is proposed for optimizing the BHFGSP.

3. Problem statement

3.1. Mathematical model of the BHFGSP

In BHFGSP, the factory contains S different stages. At each stage, $M_s\,(M_s = M_1, M_2, \ldots, M_S)$ unparallel identical machines are set to process the jobs. Between any adjacent machines, there are no intermediate buffers. For families and jobs, there are following definitions and rules: a set of $F\,(1, 2, \ldots, f, \ldots, F)$ families need to be processed on the machines, where the set of jobs in family f is denoted as ω_f. The job j in family f must pass through all the stages. Moreover, jobs in the same family need to be processed on the same machine. Before processing, family sequence dependent setup time (SDST) set_{s,f_1,f_2} should be considered between family f_1 and family f_2 at stage s. $set_{s,0,f_2}$ indicates the SDST of the first family at stage s. The processing time of job j at stage s is set as $p_{j,s}$. The objective of this paper is to minimize the makespan C_{max} of BHFGSP. To further describe the problem, we give the following assumptions.

(1) All jobs are available at time 0.
(2) Jobs in the same family must be processed continuously on the same machine, and once the processing is started, it cannot be interrupted by other families.
(3) At any time, a job can only be processed by one machine, and a machine can only process one job at any time.
(4) The transportation time is included in the processing time.
(5) All jobs must be processed at all stages consecutively. It is not allowed to skip a certain stage or end in advance.

The objective, notations, decision variables, and constraints of the BHFGSP are summarized as follows.

Notations:

S	Number of stages.
s	Index of stages, $s \in \{1, 2, \cdots, S\}$.
M_s	Number of parallel machines at stage s.
F	Number of families.
f, f'	Index of families, $f, f' \in \{0, 1, \cdots, F\}$, 0 is the index of the dummy family, which represents represents the start and end of the family sequence on a machine.
ω_f	Set of jobs in family f.
N	Number of jobs.
j, j'	Index of jobs, $j, j' \in \{1, \cdots, N\}$.
$p_{j,s}$	Processing time of job j at stage s.
$set_{f,f',s}$	Setup time from family f to family f' at stage s, $set_{s,f,f} = 0$. An initial setup time $set_{0,f,s}$ is needed if the family f is the first family at stage s.
h	Sufficiently large positive number.

Decision variables:

$c_{j,s}$	Completion time of job j at stage s.
$d_{j,s}$	Departure time of job j at stage s.
$x_{f,f',s}$	Binary decision variable, 1 if the family f' is an immediate successor of the family f at stage s, 0 otherwise.
$y_{j,j'}$	Binary decision variable, 1 if the job j precedes the job j' which belongs to the same family with the job j, 0 otherwise. The values of the decision variables are meaningful when the job j and j' are from the same family.
C_{max}	Makespan of the job sequence.

Constraints:

$$\sum_{f'=0, f'\neq f}^{F} x_{f,f',s} = 1,\ \forall f \in \{1, 2, \ldots, F\},\ \forall s \in \{1, 2, \ldots, S\} \tag{1}$$

$$\sum_{f=0, f\neq f'}^{F} x_{f,f',s} = 1,\ \forall f' \in \{1, 2, \ldots, F\},\ \forall s \in \{1, 2, \ldots, S\} \tag{2}$$

$$\sum_{f'=1}^{F} x_{0,f',s} \leq M_s,\ \forall s \in \{1, 2, \ldots, S\} \tag{3}$$

$$\sum_{f=1}^{F} x_{f,0,s} \leq M_s,\ \forall s \in \{1, 2, \ldots, S\} \tag{4}$$

$$\sum_{f'=1}^{F} x_{0,f',s} = \sum_{f=1}^{F} x_{f,0,s},\ \forall s \in \{1, 2, \ldots, S\} \tag{5}$$

$$y_{j,j'} + y_{j',j} = 1,\ \forall f \in \{1, 2, \ldots, F\},\ \forall j, j' \in \omega_f,\ j' > j,\ \forall s \in \{1, 2, \ldots, S\} \tag{6}$$

$$c_{j',s} \geq d_{j,s} + p_{j',s} + \left(y'_{j,j'} - 1\right) \cdot h,\ \forall f \in \{1, 2, \ldots, F\},\ \forall j, j' \in \omega_f,\ j' \neq j,\ \forall s \in \{1, 2, \ldots, S\} \tag{7}$$

$$c_{j',s} \geq d_{j,s} + set_{f,f',s} + p_{j',s} + \left(x_{f,f',s} - 1\right) \cdot h,\ \forall f \in \{1, 2, \ldots, F\},\ \forall f' \in \{1, 2, \ldots, F\},\ f \neq f',\ \forall j \in \omega_f,\ \forall j' \in \omega_{f'},\ \forall s \in \{1, 2, \ldots, S\} \tag{8}$$

$$c_{j,s} \geq set_{0,f,s} + p_{j,s} + \left(x_{0,f,s} - 1\right) \cdot h,\ \forall f \in \{1, 2, \ldots, F\},\ \forall j \in \omega_f,\ \forall s \in \{1, 2, \ldots, S\} \tag{9}$$

$$d_{j,s} \geq c_{j,s},\ \forall j \in \{1, 2, \ldots, N\},\ \forall s \in \{1, 2, \ldots, S\} \tag{10}$$

$$c_{j,s+1} = d_{j,s} + p_{j,s+1},\ \forall j \in \{1, 2, \ldots, N\},\ \forall s \in \{1, 2, \ldots, S-1\} \tag{11}$$

$$C_{max} \geq c_{j,S},\ \forall j \in \{1, 2, \ldots, N\} \tag{12}$$

Objective:

$$\text{Minimize } C_{max} \tag{13}$$

Constraints (1) and (2) ensure that each family must have only one immediate predecessor and successor at each stage.

Constraints (3) and (4) guarantee that the dummy family is an immediate successor and an immediate predecessor of the family less than or equal to M_s times at stage s, separately. Constraint (5) ensures that the dummy family has the same number of immediate successors and immediate predecessors at each stage. For the job j and j' from the same family f, constraint (6) ensure that either j is processed before j' or j' is processed before j, constraint (7) ensure that if the job j is processed before j', the completion time of the job j' at stage s is not less than the departure time of the job j plus the processing time $p_{j',s}$. For the family f and f', if the family f' is the immediate successor of the family f at stage s, the completion time of the job j' from the family f' at stage s is not less than the departure time of the job j from the group f plus the processing time $p_{j',s}$ and group setup time $set_{f,f',s}$, ensured by constraint (8). For the jobs in the first family processed at stage s, the initial setup time $set_{0,f,s}$ is considered by constraint (9). Constraint (10) represents that the departure time of one job is not less than its completion time at the same stage. If $d_{j,s} = c_{j,s}$, it indicates that job j is not blocked at stage s. Otherwise, if $d_{j,s} > c_{j,s}$, job j is blocked at stage s. Constraint (11) shows the completion time of one job equals to its processing time at the current stage plus its departure time at the last stage. Constraint (12) defines the makespan. Eq. (13) is the optimization objective of this paper.

Our model uses sequence-based variables with less than or equal to M_s dummy family at stage s. At each stage, the family sequence starts with a dummy family and ends with another family group. The remaining dummy families divide the family sequence into some subsequences and each subsequence represents one machine. The jobs in the same family have to be processed consecutively without interruption by jobs from a different family.

3.2. Encoding and decoding procedure

Efficient encoding and decoding procedures can reduce the computational complexity and improve the solution's quality, especially for large-scale complex combinatorial optimization problems [47]. For BHFGSP, we adopt the integer permutation encoding method to represent a solution [22,48]. That is, the solution is encoded as (π, τ), where$(\pi = \{\pi_1, \pi_2, \ldots, \pi_l, \ldots, \pi_F\})$ consists of F families, $\tau(\tau = \{\tau_1, \ldots, \tau_l, \ldots, \tau_F\})$ indicates the job sequence of each family, in which $\tau_l(\tau_l = \{\tau_{l,1}, \ldots, \tau_{l,n_l}\})$ contains n_l jobs in family π_l. Algorithm 1 gives the details of the decoding.

As can be seen from Algorithm 1, in step 1, if machine m_s is in idle state, we will choose m_s to process jobs belonging to the same family. If all machines have processed the families, we will select one machine with the minimum value of '$MIdle_{s,m_s} + Set_{s,f_1,f_2}$' to process the current family. In step 2, after choosing the appropriate machine, jobs in the current family will be scheduled consecutively from stage 1 to the last stage. In step 3, when all families are completed at the last stage, the objective value, C_{max}, of BHFGSP will be computed. Based on the decoding procedure, we propose a novel IG algorithm to minimize the makespan by reducing the blocking of jobs and SDST between different families.

3.3. Numerical illustration

To better describe the decoding procedure of BHFGSP, a numerical example of a small scale instance with $F = 4$, $S = 3$, $N = 8$, $\tau_1 = \{1, 2\}$, $\tau_2 = \{3, 4, 5\}$, $\tau_3 = \{6\}$, and $\tau_4 = \{7, 8\}$ is illustrated in Figs. 1(a), 1(b), 1(c), 1(d). All jobs are processed according to the sequence $(\tau_1, \tau_2, \tau_3, \tau_4)$ arranged in advance. Tables 1, 2 list the processing time of jobs and family SDSTs at each stage. The details are as follows:

Table 1
The processing time of jobs at each stage.

Family	Job	Stage1	Stage2	Stage3
1	1	3	6	4
	2	1	11	4
2	3	1	3	5
	4	1	3	4
	5	3	7	9
3	6	3	3	3
4	7	5	4	3
	8	5	3	4

Table 2
The family SDST at each stage.

Stage1						Stage2						Stage3					
Family	0	1	2	3	4	Family	0	1	2	3	4	Family	0	1	2	3	4
0	0	2	3	4	4	0	0	1	2	3	4	0	0	3	5	6	5
1	0	0	2	1	3	1	0	0	4	5	4	1	0	0	4	6	7
2	0	5	0	7	6	2	0	3	0	6	3	2	0	7	0	6	5
3	0	3	4	0	2	3	0	3	2	0	3	3	0	4	3	0	2
4	0	5	3	4	0	4	0	2	4	3	0	4	0	2	2	2	0

At stage 1: Family 1 and 2 are scheduled to the idle machines 1 and 2, respectively. The machine $m_1 = 1$ is assigned to family 3 according to $\min\left(MIdle_{1,1} + Set_{1,1,3}, MIdle_{1,2} + Set_{1,2,3}\right) = \min(11 + 1, 12 + 7) = 12$, and the completion and departure time of job 6 are calculated, i.e., $MIdle_{1,1} = 12$, $c_{6,1} = MIdle_{1,1} + p_{6,1} = 12 + 3 = 15$, $d_{6,1} = c_{6,1} = 15$, $MIdle_{1,1} = d_{6,1}$.

At stage 2: $m_2 = 2$ is assigned to family 3 according to $\min\left(MIdle_{2,1} + Set_{2,1,3}, MIdle_{2,2} + Set_{2,2,3}\right) = \min(22 + 5, 19 + 6) = 25$, $MIdle_{2,2} + Set_{2,2,3} = 19 + 6 = 25$. Then, calculate $MIdle_{2,2} = 25$, $c_{6,2} = \max\left\{MIdle_{2,2},\ c_{6,1}\right\} + p_{6,2} = \max\{25,\ 15\} + 3 = 28$, $d_{6,2} = c_{6,2} = 28$, $MIdle_{2,2} = c_{6,2} = 28$, $d_{6,1} = c_{6,2} - p_{6,2} = 28 - 3 = 25$, and $MIdle_{1,1} = d_{6,1} = 25$.

At stage 3: $m_3 = 1$ is assigned to family 3 according to $\min\left(MIdle_{3,1} + Set_{3,1,3}, MIdle_{3,2} + Set_{3,2,3}\right) = \min(26 + 6, 28 + 6) = 32$, Next, calculate $c_{6,3} = \max\left\{MIdle_{3,1},\ c_{6,2}\right\} + p_{6,3} = \max\{32,\ 28\} + 3 = 35$, $d_{6,3} = c_{6,3} = 35$, $MIdle_{3,1} = d_{6,3} = 35$, $d_{6,2} = c_{6,3} - p_{6,3} = 32$, and $MIdle_{2,2} = d_{6,2} = 32$. Similarly, family 4 performs the above-mentioned steps. The specific scheduling process of family 4 is shown in Fig. 1(d). At last, the completion time C_{max} is 40.

To intuitively show the effects of blocking constraints and family SDSTs, we add a family 5 to compare the makespan of two sequences, i.e., $\pi_1 = \{1, 2, 3, 4, 5\}$ and $\pi_2 = \{1, 2, 3, 5, 4\}$. As shown in Fig. 2, comparing to sequence π_1, at the first stage, blocking, family SDSTs, and completion time of π_2 are 7, 10, and 37 that are less than those of π_1, respectively. Similarly, at the second stage, π_2 has less blocking time, family SDSTs, and completion time than those of π_1. At the last stage, the family SDSTs and C_{max} of π_2 reduce by 10 and 5, respectively. To reduce the impact caused by blocking constraints and family SDSTs, based on the decoding procedure, we develop the neighborhood probabilistic selection strategies based on family and blocking-based job. Through the simulation, the proposed strategies can effectively improve the quality of the solution. In addition, for the convenience of description, the 'decoding procedure' used in the later section is simplified as function *DP()*.

4. Proposed algorithm

When dealing with BHFGSP, the departure time of jobs could be extended if they are blocked on previous machines. An effective job scheduling method can reduce blocking time. At present,

Algorithm 1 Decoding procedure of BHFGSP

Input: Solution (π, τ)
Output: C_{max}

for $f = 1$ **to** F **do**
  for $s = 1$ **to** S **do** ▷ *Step 1: Select the appropriate machine at each stage*
    Find the critical factory π_c
    if m_s has not processed a job **then**
      $MIdle_{s,m_s} = MIdle_{s,m_s} + Set_{s,0,f_2}$
    else
      Find the machine m_s which has the minimum value $MIdle_{s,m_s} + Set_{s,f_1,f_2}$
    end if
  end for
  for $j = \tau_{f,1}$ **to** τ_{f,n_f} **do** ▷ *Step 2: Schedule the jobs on the selected machines*
    $c_{j,1} = MIdle_{1,m_1} + p_{j,1}$
    $d_{j,1} = c_{j,1}$
    $MIdle_{1,m_1} = d_{j,1}$
    for $s = 2$ **to** S **do**
      $c_{j,s} = \max\left\{MIdle_{s,m_s}, c_{j,s-1}\right\} + p_{j,s}$
      $d_{j,s} = c_{j,s}$
      $MIdle_{s,m_s} = d_{j,s}$
      $d_{j,s-1} = c_{j,s} - p_{j,s}$
      $MIdle_{s-1,m_{s-1}} = d_{j,s-1}$
    end for
  end for
end for
$C_{max} = \max\ d_{j,S},\ j = 1, 2, \cdots, N$ ▷ *Step 3: Obtain the objective value of the scheduling sequence*

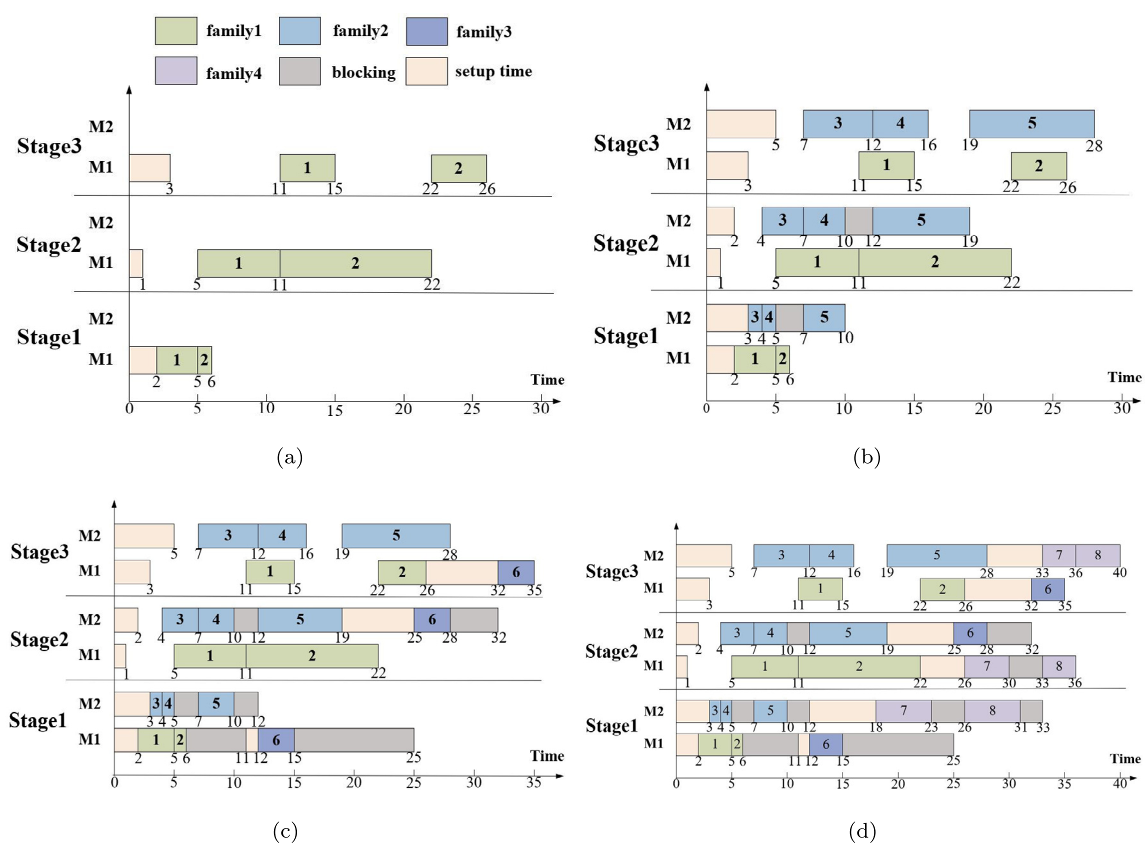


Fig. 1. Gantt chart for the scheduling numerical example: (a) Schedule the family 1. (b) Schedule the family 2. (c) Schedule the family 3. (d) Schedule the family 4.

many scholars have designed corresponding algorithms for flow shop scheduling problems with blocking constraints and family setup time constraints, respectively. If methods are designed based on the above two constraints, it will be more effective in optimizing the makespan from problem characteristics. However, most of the current algorithms do not develop strategies based on both the blocking and the family SDSTs constraints. In view of this, in this section, we design a novel IG (NIG) algorithm for two

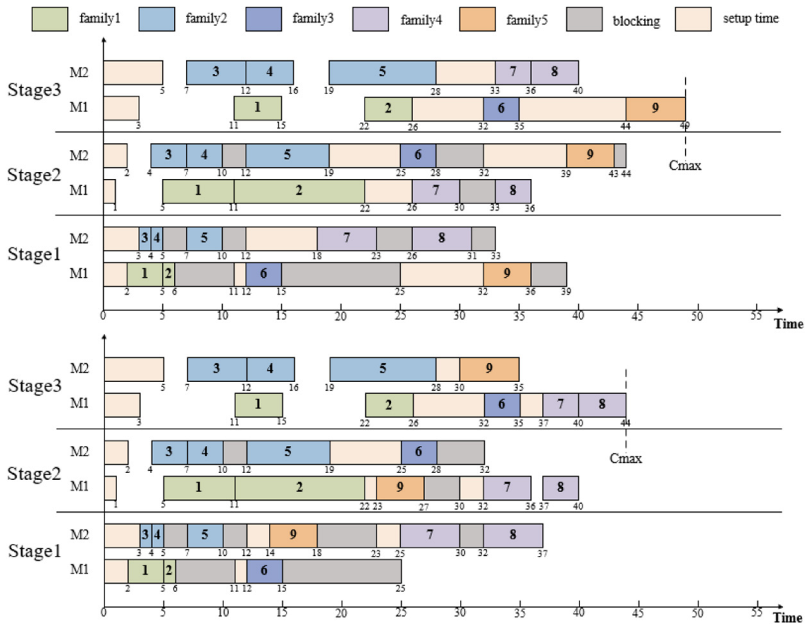


Fig. 2. Gantt chart for different scheduling sequences.

Table 3
Orthogonal array and RV values.

Case number	*d*	*R*	*C*	RV
1	2	*F*	*E*	0.64
2	2	2F	2E	0.58
3	2	3F	3E	0.52
4	2	4F	4E	0.5
5	3	*F*	2E	0.53
6	3	2F	*E*	0.52
7	3	3F	4E	0.53
8	3	4F	3E	0.51
9	4	*F*	3E	0.56
10	4	2F	4E	0.61
11	4	3F	*E*	0.59
12	4	4F	2E	0.59
13	5	*F*	4E	0.65
14	5	2F	3E	0.56
15	5	3F	2E	0.61
16	5	4F	*E*	0.6

Table 4
Mean RV values and rank of each parameter.

Level	*d*	*R*	*C*
1	0.56	0.59	0.59
2	**0.52**	0.57	0.58
3	0.59	0.56	**0.54**
4	0.6	**0.55**	0.57
Delta	0.08	0.04	0.05
Rank	1	3	2

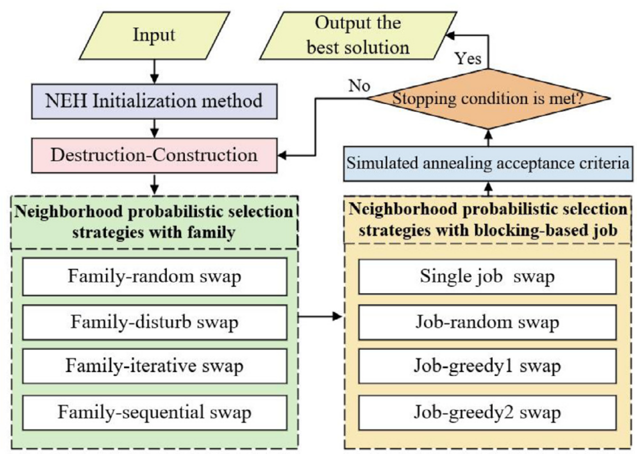


Fig. 3. The framework of the NIG algorithm.

constraints mentioned above to improve the processing efficiency of the plant and reduce unnecessary time costs.

There are two key subproblems for BHFGSP. One is the family sorting problem, and the other is the job sorting problem. Thus, for solving the above two problems, we design the corresponding strategies in the NIG algorithm (see Fig. 3).

In Fig. 3, the proposed NIG preserves the main structure of the basic IG algorithm in the literature [26]. Based on the basic IG algorithm, components of the NIG algorithm are as follows: (1) The Nawaz, Enscore, and Ham (NEH) heuristic strategy based on the family(NEH_Fam) is used to generate the initial scheduling sequence. (2) Destruction–construction strategy is adopted as a local intensification strategy to improve the local search ability. This strategy can further explore the promising solution regions. (3) Neighborhood probabilistic selection strategies with family is designed to change the arrangement order of families and reduce SDSTs of machines. (4) Neighborhood probabilistic selection strategy with the blocking-based job is developed to perturb jobs in each family. (5) SA acceptance criterion will update the current solution according to the temperature value. The criterion can improve the global and local search ability of NIG algorithm. Due to space limitations, this paper will not elaborate on the SA acceptance criterion in the following chapters. For specific details, please refer to [26].

4.1. NEH_Fam initialization strategy

When the SDST of families decreases, the makespan value of the whole scheduling sequence may also be reduced [22]. The NEH heuristic algorithm has been widely utilized and embedded into the initialization phase of the IG algorithm because of its high efficiency [49]. The NEH heuristic algorithm is performed based on the job sequence. Each job may be inserted at any position

Table 5
Evaluation of the mathematical model and NIG algorithm.

Mathematical model					NIG		
J/F/S/M	Constraints	Makespan	RPD	Times	Makespan	RPD	Times
8/4/2/3	346	37	0	0.59 s	37	0	0.80 s
10/5/2/3	516	48	0	1.10 s	48	0	1.00 s
12/6/2/3	720	55	0	1.86 s	55	0	1.20 s
14/7/2/3	958	71	0	21.17 s	71	0	1.40 s
16/8/2/3	1230	87	0	70.23 s	87	0	1.60 s
18/9/3/8	1895	45	0	137.31 s	53	17.18	2.70 s
20/10/3/8	2357	48	0	1000.00 s	57	18.75	3.00 s
22/11/3/8	2870	53	0	1000.00 s	62	16.98	3.30 s
24/12/3/8	3434	58	0	1000.00 s	66	13.79	3.60 s
26/13/3/8	4049	64	0	1000.00 s	65	1.56	3.90 s
50/10/3/8	15773	164	18.84	1000.00 s	138	0.94	3.00 s
60/12/3/8	22562	211	26.35	1000.00 s	167	0.92	3.60 s

Table 6
Comparison results of NIG_N_J, NIG_N_F, and NIG when $F = 20$.

	N×	NIG_N_J				NIG_N_F				NIG		
	S	MEAN	BEST	RPD	*p*-value	MEAN	BEST	RPD	*p*-value	MEAN	BEST	RPD
	80 × 3	3583	3576	7.52	**5.38E−92**	3363	3346	0.92	**1.93E−21**	**3338**	**3332**	**0.18**
	80 × 5	7342	7325	5.34	**1.57E−61**	7098	7052	1.83	**8.35E−22**	**7006**	**6970**	**0.52**
	80 × 8	11431	11420	6.38	**4.71E−63**	10992	10893	2.3	**9.60E−23**	**10825**	**10745**	**0.74**
	100 × 3	15175	15153	7.01	**2.18E−50**	14815	14643	4.47	**1.93E−28**	**14377**	**14181**	**1.39**
	100 × 5	57186	57182	6.18	**5.29E−101**	53914	53888	0.11	0.125135793	**53894**	**53855**	**0.07**
	100 × 8	66775	66771	6.37	**1.53E−85**	63007	62963	0.37	**6.63E−10**	**62847**	**62776**	**0.11**
	120 × 3	31755	31742	4.69	**6.97E−48**	31111	30692	2.56	**1.94E−13**	**30580**	**30333**	**0.81**
	120 × 5	109270	109265	7.19	**1.46E−84**	102307	102173	0.36	**1.97E−05**	**102095**	**101937**	**0.16**
	120 × 8	61090	61051	6.47	**3.48E−56**	58457	58169	1.89	**8.46E−17**	**57622**	**57375**	**0.43**
	140 × 3	77293	77245	6.63	**5.71E−57**	73721	72614	1.71	**7.86E−10**	**72892**	**72484**	**0.56**
	140 × 5	84749	84709	6.18	**1.60E−47**	80698	**79817**	1.1	0.232587102	**80506**	79899	**0.86**
	140 × 8	93690	93663	5.31	**1.19E−44**	90399	89684	1.61	**1.67E−06**	**89595**	**88968**	**0.7**
	160 × 3	222129	222123	7.86	**4.11E−74**	206364	206080	0.21	0.928215106	**206351**	**205942**	**0.2**
	160 × 5	121554	121528	6.19	**2.99E−40**	116352	114902	1.65	**0.000113453**	**115241**	**114466**	**0.68**
	160 × 8	265912	265899	6.52	**5.65E−63**	250539	249807	0.36	0.506821959	**250365**	**249634**	**0.29**
	180 × 3	300394	300387	8.2	**1.03E−61**	278762	277791	0.41	0.625715207	**278574**	**277620**	**0.34**
	180 × 5	162536	162508	5.61	**8.64E−28**	157548	156625	2.37	**3.05E−05**	**155817**	**153908**	**1.24**
	180 × 8	346908	346902	6.35	**4.83E−56**	327758	**326184**	0.48	0.842155784	**327669**	326239	**0.46**
F = 20	200 × 3	388560	388549	8.98	**3.26E−64**	**357765**	356620	**0.34**	0.868005603	357842	**356548**	0.36
	200 × 5	418192	418186	8.39	**5.92E−55**	387976	386117	0.56	0.856130273	**387855**	**385805**	**0.53**
	200 × 8	446965	446962	6.84	**1.87E−53**	**420265**	**418353**	**0.46**	0.87650234	420363	418358	0.48
	220 × 3	165021	164951	4.4	**6.44E−17**	162330	160341	2.69	**0.003390094**	**161076**	**158070**	**1.9**
	220 × 5	256401	256363	5.61	**1.79E−26**	**246604**	244386	**1.58**	0.829180508	246747	**242771**	1.64
	220 × 8	545756	545749	6.63	**5.70E−53**	**514269**	**511827**	**0.48**	0.627722974	514636	512097	0.55
	240 × 3	587244	587237	9.15	**5.07E−64**	**539952**	538244	**0.36**	0.955362056	539991	**538036**	**0.36**
	240 × 5	311582	311503	5.34	**4.84E−27**	**299358**	**295778**	**1.21**	0.170438384	300398	296560	1.56
	240 × 8	660115	660107	7.41	**3.98E−53**	**617737**	**614600**	**0.51**	0.783785401	618015	614721	0.56
	260 × 3	704782	704777	8.74	**7.01E−63**	650702	648349	0.39	0.932890377	**650624**	**648162**	**0.38**
	260 × 5	751055	751048	8.59	**1.02E−56**	695580	692008	0.57	0.736924625	**695183**	**691637**	**0.51**
	260 × 8	786828	786817	7.09	**1.21E−52**	739041	735014	0.58	0.902748855	**738897**	**734750**	**0.56**
	280 × 3	834999	834994	9.01	**3.48E−65**	769024	765989	0.4	0.810918129	**768771**	**765957**	**0.37**
	280 × 5	872658	872650	8.5	**9.27E−56**	808675	804494	0.54	0.930189089	**808554**	**804314**	**0.53**
	280 × 8	919273	919265	7.01	**8.31E−55**	863493	859107	0.52	0.882650435	**863300**	**859016**	**0.5**
	300 × 3	484274	484005	5.97	**3.38E−29**	**462426**	**456989**	**1.19**	0.191874249	464181	458368	1.57
	300 × 5	1026763	1026756	8.98	**1.80E−56**	**947301**	**942146**	**0.55**	0.931675544	947445	942254	0.56
	300 × 8	1066697	1066685	7.37	**9.16E−53**	999793	994118	0.64	0.64951689	**999037**	**993470**	**0.56**

in the sorting process. However, when using the NEH heuristic algorithm to arrange job sequence for BHFGSP, the jobs in the same family may be arranged to other families. Therefore, a new NEH heuristic based on the family (*NEH_Fam*) is proposed to obtain a feasible initial solution with high quality. We replace the job operation with the family operation to solve the problem that the job in different families cannot be crossed.

To describe the initialization process more conveniently, Algorithm 2 gives the procedure of *NEH_Fam* strategy. In Algorithm 2, notion $p_{j,s,l}$ indicates the processing time of job j belonging to the family l at stage s. At the beginning of *NEH_Fam* procedure, all jobs are randomly assigned to established families. Then, families are arranged in descending order by computing the total processing time of all jobs belonging to the same family at all stages. After that, we extract families one by one in descending order and insert them into the positions of another family sequence. The family sequence with the minimum makespan will be preserved and participate in the insertion test of new incoming jobs. Finally, a complete family sequence is obtained.

4.2. Destruction–construction strategy

After initialization, the destruction–construction strategy based on the family is executed to disturb job sequence and reduce the makespan. Similar to the NEH heuristic method, the destruction–construction strategy also uses greedy insertion operation to find a better solution in search regions. By using this strategy, the algorithm's performance has been dramatically improved [50,51]. However, the destruction–construction strategy only optimizes the objective by changing the arrangement order of jobs. If it is

Table 7
Comparison results of NIG_N_J, NIG_N_F, and NIG when $F = 40$.

	N× S	NIG_N_J				NIG_N_F				NIG		
		MEAN	BEST	RPD	*p*-value	MEAN	BEST	RPD	*p*-value	MEAN	BEST	RPD
	80 × 3	7034	7004	6.59	**1.83E−42**	6764	6690	2.49	**1.86E−09**	**6676**	**6599**	**1.17**
	80 × 5	17870	17830	6.29	**1.13E−62**	17119	17049	1.83	**2.10E−23**	**16920**	**16812**	**0.65**
	80 × 8	43187	43180	5.14	**2.96E−109**	41287	41277	0.52	**2.86E−46**	**41097**	**41075**	**0.05**
	100 × 3	24547	24496	7.63	**1.35E−59**	23533	23299	3.19	**4.31E−24**	**23094**	**22806**	**1.26**
	100 × 5	84779	84765	6.43	**1.92E−104**	79877	79779	0.28	**7.75E−14**	**79712**	**79658**	**0.07**
	100 × 8	94945	94930	5.38	**2.72E−79**	90658	90522	0.62	**7.40E−18**	**90240**	**90096**	**0.16**
	120 × 3	63286	63246	6.26	**7.01E−76**	60218	59952	1.11	**3.55E−19**	**59748**	**59556**	**0.32**
	120 × 5	139509	139496	6.44	**4.11E−84**	131480	131370	0.31	**8.83E−06**	**131224**	**131070**	**0.12**
	120 × 8	154494	154471	5.19	**1.25E−83**	147442	147375	0.39	**5.54E−12**	**147063**	**146868**	**0.13**
	140 × 3	94719	94648	6.57	**6.28E−54**	91229	90184	2.64	**2.89E−19**	**89447**	**88883**	**0.63**
	140 × 5	211116	211109	6.95	**2.92E−81**	198289	198072	0.45	**8.14E−08**	**197675**	**197392**	**0.14**
	140 × 8	224800	224791	5.87	**1.74E−75**	213193	213040	0.4	**1.66E−05**	**212712**	**212341**	**0.17**
	160 × 3	265976	265964	7.55	**1.26E−75**	248132	247765	0.33	0.081600077	**247832**	**247306**	**0.21**
	160 × 5	287120	287106	6.94	**2.70E−71**	**269089**	268617	**0.23**	0.892697627	269118	**268481**	0.24
	160 × 8	153306	153242	5.85	**2.66E−38**	148400	147636	2.46	**2.47E−10**	**146542**	**144833**	**1.18**
	180 × 3	347672	347662	8.12	**6.51E−68**	322861	322111	0.41	0.166855122	**322422**	**321555**	**0.27**
	180 × 5	374189	374182	7.01	**1.15E−61**	350945	350021	0.36	0.892936131	**350894**	**349674**	**0.35**
	180 × 8	394788	394778	5.79	**8.18E−61**	375001	373657	0.48	0.238106163	**374484**	**373197**	**0.34**
F = 40	200 × 3	220300	220252	6.33	**9.26E−36**	**209849**	**207176**	**1.29**	0.371162287	210361	207953	1.54
	200 × 5	156288	156199	2.6	**5.01E−26**	156881	154069	2.99	**1.75E−19**	**153469**	**152330**	**0.75**
	200 × 8	495140	495130	6.38	**3.14E−58**	467331	465612	0.4	0.535270885	**466933**	**465464**	**0.32**
	220 × 3	135426	135275	2.38	**3.28E−19**	135184	133199	2.19	**8.16E−09**	**133813**	**132284**	**1.16**
	220 × 5	192551	192417	2.33	**0.001245437**	192258	191012	2.18	**0.011899575**	**191007**	**188164**	**1.51**
	220 × 8	608278	608267	6.12	**4.67E−50**	576461	573483	0.57	0.614982104	**576001**	**573190**	**0.49**
	240 × 3	328153	328072	5.21	**5.69E−31**	316598	313184	1.5	0.053995907	**315201**	**311917**	**1.05**
	240 × 5	693686	693677	7.61	**2.42E−51**	648679	644847	0.62	0.812271073	**648382**	**644652**	**0.58**
	240 × 8	722626	722611	6.16	**7.92E−50**	684586	680883	0.57	0.941720482	**684505**	**680705**	**0.56**
	260 × 3	260450	260286	3.01	**1.80E−11**	263024	259131	4.03	**9.40E−16**	**256400**	**252838**	**1.41**
	260 × 5	822004	821995	7.46	**3.16E−52**	769222	**764916**	0.56	0.992396367	**769208**	764924	**0.56**
	260 × 8	856628	856618	6.91	**7.25E−48**	806436	801574	0.64	0.930500353	**806305**	**801279**	**0.63**
	280 × 3	457123	457046	5.1	**3.03E−30**	**440141**	437149	**1.19**	0.667767265	440524	**434945**	1.28
	280 × 5	959708	959694	8.46	**9.45E−52**	**890928**	**884889**	**0.68**	0.91178574	891133	885490	0.71
	280 × 8	993217	993208	6.41	**5.30E−50**	**938758**	**933348**	**0.58**	0.932651183	938892	933489	0.59
	300 × 3	515708	515674	5.08	**5.49E−28**	**496316**	**490772**	**1.13**	0.328976125	497458	492160	1.36
	300 × 5	273676	273462	1.43	**3.46E−25**	272660	270804	1.05	4.04E−08	**271030**	**269819**	**0.45**
	300 × 8	1145416	1145405	7.12	**9.93E−51**	**1075620**	1069316	**0.59**	0.934351803	1075773	**1069291**	0.61

Algorithm 2 *NEH_Fam* initialization strategy

Input: $\left(\pi^{origin}, \tau^{origin}\right)$, $\pi^{sub} = \{\}$
Output: $\left(\pi^{N}, \tau^{N}\right)$
$P_l \leftarrow \sum_{s=1}^{S} \sum_{j=1}^{n_l} p_{j,s,l},\ \ l = 1, \ldots, F$
$\pi^{\Delta} \leftarrow$ Sort families $\left\{\pi_1^{origin}, \ldots, \pi_l^{origin}, \ldots, \pi_F^{origin}\right\}$ according to descending sequence P_l
for $l = 1$ **to** F **do**
  for $i = 1$ **to** $\left|\pi^{sub}\right| + 1$ **do**
    $\pi_i^{sub} \leftarrow$ Extract family π_l^{origin} from π^{Δ}, and insert it into the ith position of π^{sub}
  end for
  $\left(\pi^{sub}, \tau^{sub}\right) \leftarrow arg\ \min_{i=1}^{\left|\pi^{sub}\right|+1} DP\left(\pi_i^{sub}, \tau_i^{sub}\right)$
end for
$\left(\pi^{N}, \tau^{N}\right) \leftarrow \left(\pi^{sub}, \tau^{sub}\right)$

used directly for BHFGSP, it may cause the cross restriction of jobs between different families. Thus, in this study, we improve the destruction–construction strategy based on the family sequence.

In the proposed destruction–construction strategy, the procedure mainly consists of the following three parts: (1) d random families are selected and removed from the original sequence and put into an empty sequence π^{remove}; (2) d families in π^{remove} are extracted one by one and inserted into all positions of the current family; (3) the makespan of the current sequence is calculated, and the sequence with the lowest objective value is recorded. Algorithm 3 gives the procedure of the destruction–construction strategy.

4.3. Neighborhood probabilistic selection strategies with family

For the scheduling process of BHFGSP, the processing time of jobs is determined in advance and cannot be changed. The machine selection is decided by the decoding procedure automatically. Therefore, the primary way is to reduce the impact of family SDSTs and blocking constraints of jobs. We design the neighborhood probabilistic selection strategies with family to reduce the SDST, further minimizing the objective value. In addition, all strategies are designed based on the swap operators, which reduces the time complexity [37].

In the framework of NIG algorithm, neighborhood probabilistic selection strategies with family are helpful to increase the global

Table 8
Comparison results of NIG_N_J, NIG_N_F, and NIG when $F = 60$.

	N× S	NIG_N_J				NIG_N_F				NIG		
		MEAN	BEST	RPD	*p*-value	MEAN	BEST	RPD	*p*-value	MEAN	BEST	RPD
	80 × 3	48172	48166	4.67	**8.50E−116**	**46034**	**46024**	**0.02**	**2.91E−16**	46073	46054	0.11
	80 × 5	56388	56379	5.19	**4.78E−103**	**53680**	53679	0.14	**0.00195635**	53659	**53604**	**0.1**
	80 × 8	64473	64456	3.98	**2.31E−87**	62139	62124	0.22	**4.81E−10**	**62051**	**62006**	**0.07**
	100 × 3	100245	100240	6.17	**6.35E−118**	94504	94485	0.09	**1.85E−05**	**94460**	**94423**	**0.04**
	100 × 5	110900	110877	5.62	**2.35E−93**	105199	105138	0.19	**0.000101343**	**105086**	**104997**	**0.08**
	100 × 8	121332	121308	4.43	**1.83E−89**	116641	116614	0.4	**3.18E−21**	**116310**	**116181**	**0.11**
	120 × 3	78339	78269	6.93	**1.71E−73**	74086	73694	1.13	**1.74E−12**	**73602**	**73261**	**0.47**
	120 × 5	86048	85993	7.27	**3.68E−56**	82017	81666	2.24	**5.47E−20**	**80566**	**80217**	**0.43**
	120 × 8	185090	185084	5.62	**1.23E−88**	175899	175694	0.38	**8.37E−12**	**175447**	**175240**	**0.12**
	140 × 3	226706	226692	7.18	**6.56E−85**	212296	212077	0.36	**1.11E−06**	**211809**	**211528**	**0.13**
	140 × 5	245851	245827	6.08	**1.30E−77**	232944	232682	0.51	**2.51E−07**	**232257**	**231759**	**0.21**
	140 × 8	261318	261300	5.22	**4.69E−72**	249426	249134	0.43	**0.000163706**	**248895**	**248356**	**0.22**
	160 × 3	306191	306183	7.82	**1.42E−82**	284839	284479	0.3	**0.045594026**	**284507**	**283996**	**0.18**
	160 × 5	331205	331196	6.76	**8.42E−78**	311142	310852	0.29	**0.021893305**	**310727**	**310242**	**0.16**
	160 × 8	349372	349353	5.72	**3.71E−66**	331902	331498	0.43	**0.005590021**	**331240**	**330469**	**0.23**
	180 × 3	397746	397740	8.08	**1.91E−71**	369268	368609	0.34	0.296480178	**368941**	**367999**	**0.26**
	180 × 5	209917	209713	5.67	**1.36E−32**	201735	200024	1.55	0.190802921	**201062**	**198653**	**1.21**
	180 × 8	223048	222883	3.98	**5.57E−38**	217346	216287	1.32	**4.54E−05**	**216062**	**214516**	**0.72**
F = 60	200 × 3	491703	491699	7.5	**2.76E−65**	458930	457635	0.33	0.648231836	**458689**	**457408**	**0.28**
	200 × 5	520254	520226	6.91	**1.79E−60**	488763	487094	0.43	0.425671132	**488296**	**486647**	**0.34**
	200 × 8	549905	549895	6.14	**5.52E−58**	520073	518467	0.38	0.988870792	**520065**	**518079**	**0.38**
	220 × 3	603385	603376	8.25	**1.12E−58**	559901	557876	0.45	0.545212105	**559406**	**557398**	**0.36**
	220 × 5	637609	637587	7.05	**1.72E−52**	**598452**	595886	0.48	0.908639542	598560	**595614**	0.49
	220 × 8	665867	665841	5.75	**7.21E−53**	**632336**	**629662**	**0.42**	0.909203369	632432	629924	0.44
	240 × 3	354620	354488	5.92	**6.02E−31**	345569	341423	3.22	**1.25E−08**	**339599**	**334797**	**1.43**
	240 × 5	375334	375230	5.68	**2.00E−30**	**359181**	**355165**	**1.13**	0.054796481	361141	357828	1.68
	240 × 8	397080	396975	3.41	**6.51E−14**	393074	386326	2.37	**0.003812433**	**389238**	**383985**	**1.37**
	260 × 3	834976	834967	8.58	**2.66E−60**	774809	770032	0.75	0.103142985	**772400**	**769009**	**0.44**
	260 × 5	893735	893720	7.32	**8.24E−55**	839259	834056	0.78	0.178693588	**837262**	**832782**	**0.54**
	260 × 8	934767	934763	6.41	**2.60E−54**	884250	880050	0.66	0.118630305	**882335**	**878460**	**0.44**
	280 × 3	486013	485933	4.87	**2.63E−35**	469312	**463462**	1.26	0.470776135	**468501**	464161	**1.09**
	280 × 5	1036616	1036594	7.91	**4.80E−48**	**967587**	**960641**	**0.72**	0.93554151	967763	961375	0.74
	280 × 8	1073307	1073295	6.03	**8.97E−46**	**1018358**	**1012225**	**0.61**	0.659819935	1019143	1012769	0.68
	300 × 3	555522	555384	5.07	**2.00E−25**	540440	533969	2.22	**0.013554354**	**536813**	**528726**	**1.53**
	300 × 5	1181436	1181418	7.87	**2.18E−48**	**1102406**	1095572	**0.65**	0.871957291	1102769	**1095260**	0.69
	300 × 8	609999	609808	3.01	**2.78E−24**	604965	**592183**	2.16	**1.57E−06**	**597482**	593013	**0.89**

Algorithm 3 Destruction–construction strategy

Input: $\left(\pi^N, \tau^N\right)$, $\pi^{remove} = \{\}$
Output: $\left(\pi^d, \tau^d\right)$
for $g = 1$ **to** d **do**
  Randomly extract a family from $\pi^{initial}$, and insert it into π^{remove}
end for
for $e = 1$ **to** d **do**
  for $i = 1$ **to** $\left|\pi^{initial}\right| + 1$ **do**
    $\pi_i^{sub} \leftarrow$ Extract the first family of π^{remove}, and insert it into the ith position of π^{sub}
  end for
  $\pi^{sub} \leftarrow arg\min_{i=1}^{\left|\pi^{sub}\right|+1} DP\left(\pi_i^{sub}, \tau_i^{sub}\right)$
end for
$\left(\pi^d, \tau^d\right) \leftarrow \left(\pi^{sub}, \tau^{sub}\right)$

search capability. In the proposed neighborhood probabilistic selection strategies, four family-based swap operators are designed for the optimized solution. The framework of neighborhood probabilistic selection strategies is shown in Algorithm 4.

The steps of ***Family-random swap operator*** $\left(\left(\pi^d, \tau^d\right)\right)$ are given as follows:

Step1: Randomly select two families π_a^{inter}, π_b^{inter} from π^{inter} and swap their positions;

Step2: If $DP\left(\pi^{inter}, \tau^{inter}\right) <= DP\left(\pi^d, \tau^d\right)$, $\left(\pi^d, \tau^d\right) \leftarrow \left(\pi^{inter}, \tau^{inter}\right)$;

Otherwise, $\left(\pi^{inter}, \tau^{inter}\right) \leftarrow \left(\pi^d, \tau^d\right)$;

Step3: Repeat steps 1–2, after R iterations, $(\pi^g, \tau^g) \leftarrow \left(\pi^{inter}, \tau^{inter}\right)$.

The steps of ***Family-disturb swap operator*** $\left(\left(\pi^d, \tau^d\right)\right)$ are given as follows:

Step1: $count = 1$, $count2 = count + 1$;

Step2: While $count2 <= F$, swap the positions of π_{count}^{inter} and π_{count2}^{inter};

Step3: If $DP\left(\pi^{inter}, \tau^{inter}\right) <= DP\left(\pi^d, \tau^d\right)$, $\left(\pi^d, \tau^d\right) \leftarrow \left(\pi^{inter}, \tau^{inter}\right)$;

Otherwise, $\left(\pi^{inter}, \tau^{inter}\right) \leftarrow \left(\pi^d, \tau^d\right)$, and $count2++$;

Step4: Continuously swap the positions of π_{count}^{inter} and π_{count2}^{inter} until the termination $count2 = F$ is met;

Step5: When the number of $count2$ reaches F, let $count++$, $count2 = count + 1$;

Step6: Continue to perform above-mentioned operations until $count = F - 1$, $(\pi^g, \tau^g) \leftarrow \left(\pi^{inter}, \tau^{inter}\right)$.

The steps of ***Family-iterative swap operator*** $\left(\left(\pi^d, \tau^d\right)\right)$ are given as follows:

Table 9
Comparison results with the neighborhood probabilistic selection strategies with blocking-based job when $\rho = 100$, $F = 20$.

	N× S	IGA				IGDLM				NIG		
		MEAN	BEST	RPD	*p*-value	MEAN	BEST	RPD	*p*-value	MEAN	BEST	RPD
	80 × 3	3592	3585	7.59	**0**	3590	3582	7.5	**0**	**3348**	**3339**	**0.28**
	80 × 5	7392	7345	5.53	**0**	7355	7335	5	**0**	**7035**	**7005**	**0.43**
	80 × 8	11538	11459	6.7	**0**	11451	11439	5.9	**0**	**10869**	**10813**	**0.52**
	100 × 3	15219	15201	6.42	**0**	15197	15157	6.27	**0**	**14499**	**14300**	**1.39**
	100 × 5	57201	57188	6.12	**0**	57197	57188	6.11	**0**	**53923**	**53901**	**0.04**
	100 × 8	66791	66783	6.47	**0**	66789	66779	6.47	**0**	**62810**	**62732**	**0.12**
	120 × 3	31834	31791	4.33	**0**	31793	31749	4.2	**0**	**30897**	**30512**	**1.26**
	120 × 5	109290	109275	6.89	**0**	109292	109282	6.89	**0**	**102303**	**102247**	**0.05**
	120 × 8	61195	61082	5.82	**0**	61081	61059	5.62	**0**	**58185**	**57829**	**0.62**
	140 × 3	77439	77343	6.94	**0**	77283	77259	6.72	**0**	**72889**	**72416**	**0.65**
	140 × 5	84883	84754	6.24	**0**	84770	84732	6.1	**0**	**80443**	**79896**	**0.68**
	140 × 8	93869	93686	5.36	**0**	93725	93675	5.2	**0**	**89766**	**89095**	**0.75**
	160 × 3	222153	222148	7.86	**0**	222138	222132	7.85	**0**	**206302**	**205971**	**0.16**
	160 × 5	121663	121551	6.13	**0**	121561	121531	6.04	**0**	**115744**	**114636**	**0.97**
	160 × 8	265960	265932	6.49	**0**	265940	265925	6.48	**0**	**250505**	**249754**	**0.3**
	180 × 3	300409	300404	8.2	**0**	300407	300397	8.2	**0**	**278264**	**277651**	**0.22**
	180 × 5	162820	162523	5.98	**0**	162590	162513	5.83	**0**	**155471**	**153635**	**1.2**
	180 × 8	346935	346914	6.36	**0**	346921	346913	6.35	**0**	**327249**	**326197**	**0.32**
F = 20	200 × 3	388570	388565	9	**0**	388572	388568	9	**0**	**357692**	**356474**	**0.34**
	200 × 5	418205	418193	8.39	**0**	418207	418196	8.39	**0**	**387846**	**385836**	**0.52**
	200 × 8	446983	446972	6.79	**0**	446963	446961	6.79	**0**	**420406**	**418561**	**0.44**
	220 × 3	165399	165117	3.52	**0**	165197	164976	3.39	**0**	**162606**	**159779**	**1.77**
	220 × 5	256638	256387	5.58	**0**	256402	256369	5.49	**0**	**246357**	**243067**	**1.35**
	220 × 8	545786	545768	6.64	**0**	545787	545760	6.64	**0**	**514384**	**511802**	**0.5**
	240 × 3	587250	587244	9.09	**0**	587251	587248	9.09	**0**	**540151**	**538294**	**0.34**
	240 × 5	312050	311566	5.52	**0**	311637	311523	5.38	**0**	**299173**	**295714**	**1.17**
	240 × 8	660151	660122	7.41	**0**	660132	660122	7.4	**0**	**617769**	**614620**	**0.51**
	260 × 3	704790	704785	8.68	**0**	704795	704782	8.68	**0**	**650794**	**648528**	**0.35**
	260 × 5	751061	751055	8.55	**0**	751070	751063	8.55	**0**	**695308**	**691882**	**0.5**
	260 × 8	786863	786831	7.08	**0**	786847	786837	7.07	**0**	**739055**	**734863**	**0.57**
	280 × 3	835004	834996	8.99	**0**	835003	835002	8.99	**0**	**769061**	**766127**	**0.38**
	280 × 5	872683	872654	8.47	**0**	872680	872658	8.47	**0**	**808825**	**804507**	**0.54**
	280 × 8	919293	919268	6.95	**0**	919287	919277	6.95	**0**	**863682**	**859529**	**0.48**
	300 × 3	484586	484023	6.25	**0**	484268	484017	6.18	**0**	**463111**	**456088**	**1.54**
	300 × 5	1026776	1026758	8.94	**0**	1026775	1026769	8.94	**0**	**947670**	**942500**	**0.55**
	300 × 8	1066707	1066697	7.34	**0**	1066718	1066696	7.34	**0**	**999444**	**993741**	**0.57**

Algorithm 4 Neighborhood probabilistic selection strategies with family

Input: (π^d, τ^d)
Output: (π^g, τ^g)
$rnd \leftarrow randi(1, 4)$ ▷ A random integer in [1, 4]
if $rnd = 1$ **then**
  $(\pi^g, \tau^g) \leftarrow$ Execute the ***Family-random swap operator*** $((\pi^d, \tau^d))$
else if $rnd = 2$ **then**
  $(\pi^g, \tau^g) \leftarrow$ Execute the ***Family-disturb swap operator*** $((\pi^d, \tau^d))$
else if $rnd = 3$ **then**
  $(\pi^g, \tau^g) \leftarrow$ Execute the ***Family-iterative swap operator*** $((\pi^d, \tau^d))$
else
  $(\pi^g, \tau^g) \leftarrow$ Execute the ***Family-sequential swap operator*** $((\pi^d, \tau^d))$
end if

Step1: $(\pi^{inter}, \tau^{inter}) \leftarrow (\pi^d, \tau^d)$. $count = 1$, $count2 = count + 1$;

Step2: While $count2 <= F$, swap the positions of π^{inter}_{count} and π^{inter}_{count2}, $(\pi^{inter}, \tau^{inter}) \leftarrow (\pi^d, \tau^d)$, $count2 + +$;

Step3: Constantly implement the steps until the $count2 = F$.

Step4: After execute above mentioned operations, the objective value of sequence $(\pi^{inter}, \tau^{inter})$ is calculated, then the position of the family with the minimum completion time is recorded as *pos*.

Step5: Swap the positions of π^{inter}_{count} and π^{inter}_{pos}, if $DP(\pi^{inter}, \tau^{inter}) <= DP(\pi^d, \tau^d)$, $(\pi^d, \tau^d) \leftarrow (\pi^{inter}, \tau^{inter})$;

Otherwise, $(\pi^{inter}, \tau^{inter}) \leftarrow (\pi^d, \tau^d)$, count++, count2 = count+1;

Step6: Continue to perform operations mentioned above until $count = F - 1$, $(\pi^g, \tau^g) \leftarrow (\pi^{inter}, \tau^{inter})$.

The steps of ***Family-sequential swap operator*** $((\pi^d, \tau^d))$ are given as follows:

Step1: $(\pi^{inter}, \tau^{inter}) \leftarrow (\pi^d, \tau^d)$. $count = 1$, $count2 = count$.

Step2: While $count2 <= F - 1$, swap the positions of π^{inter}_{count2} and $\pi^{inter}_{count2+1}$, $count2 + +$;

Step3: Repeat the above procedure. When satisfy the termination condition $count2 = F - 1$, the best sequence is recorded as $(\pi^{inter}, \tau^{inter})$;

Step4: If $DP(\pi^{inter}, \tau^{inter}) <= DP(\pi^d, \tau^d)$, $(\pi^d, \tau^d) \leftarrow (\pi^{inter}, \tau^{inter})$;

Otherwise, $(\pi^{inter}, \tau^{inter}) \leftarrow (\pi^d, \tau^d)$; $count + +$, $count2 = count$;

Step5: Continue to carry out the above-mentioned steps until $count = F - 1$, $(\pi^g, \tau^g) \leftarrow (\pi^{inter}, \tau^{inter})$.

Table 10
Comparison results with the neighborhood probabilistic selection strategies with blocking-based job when $\rho = 100$, $F = 40$.

	N× S	NIG_N_J				NIG_N_F				NIG		
		MEAN	BEST	RPD	*p*-value	MEAN	BEST	RPD	*p*-value	MEAN	BEST	RPD
	80 × 3	6780	6641	2.74	**1.04E−10**	6727	6604	1.94	**0.000190035**	**6676**	**6599**	**1.17**
	80 × 5	17122	17053	**1.84**	**3.77E−23**	16994	16868	1.08	**2.98E−06**	**16920**	**16812**	**0.65**
	80 × 8	41148	41130	0.18	**2.40E−16**	41189	41152	0.28	**3.03E−23**	**41097**	**41075**	**0.05**
	100 × 3	23495	23290	3.02	**3.09E−21**	23183	22942	1.65	**0.001715813**	**23094**	**22806**	**1.26**
	100 × 5	79837	79771	0.23	**9.89E−10**	79804	79789	0.18	**1.24E−10**	**79712**	**79658**	**0.07**
	100 × 8	90455	90387	0.4	**3.41E−09**	90419	90353	0.36	**2.48E−07**	**90240**	**90096**	**0.16**
	120 × 3	60132	59857	0.97	**5.76E−17**	59821	59599	0.44	**0.038845799**	**59748**	**59556**	**0.32**
	120 × 5	131439	131323	0.28	**5.03E−05**	131321	131155	0.19	**0.049254482**	**131224**	**131070**	**0.12**
	120 × 8	147183	146962	0.21	**0.025874204**	147207	147084	0.23	**0.003782263**	**147063**	**146868**	**0.13**
	140 × 3	90996	90313	2.38	**2.92E−19**	**89295**	88955	**0.46**	0.166658561	89447	**88883**	0.63
	140 × 5	197931	197784	0.27	**0.005386341**	198189	198074	0.4	**7.24E−08**	**197675**	**197392**	**0.14**
	140 × 8	212916	212648	0.27	**0.049961787**	**212661**	212447	**0.15**	0.634106315	212712	**212341**	0.17
	160 × 3	**247671**	**247301**	0.15	0.317855221	247967	247570	0.27	0.393842849	247832	247306	0.21
	160 × 5	**268837**	**268371**	**0.17**	0.172109355	269182	268817	0.3	0.739613005	269118	268481	0.28
	160 × 8	148889	147542	2.8	**3.76E−11**	**146195**	145353	**0.94**	0.23233226	146542	**144833**	1.18
	180 × 3	322449	321765	0.28	0.932316732	322471	321900	0.29	0.868081967	**322422**	**321555**	**0.27**
	180 × 5	**350696**	349777	**0.29**	0.593088091	350772	349977	0.31	0.727684838	350894	**349674**	0.35
	180 × 8	**374419**	373416	**0.33**	0.85931233	374694	373522	0.4	0.55168819	374484	**373197**	0.34
F = 40	200 × 3	212192	209163	**2.7**	**0.000316588**	**209800**	**206605**	**1.55**	0.362187062	210361	207953	1.82
	200 × 5	158690	154657	4.18	1.03E−22	155042	153454	1.78	1.95E−07	**153469**	**152330**	**0.75**
	200 × 8	466986	465910	0.33	0.92345102	467137	465608	0.36	0.723174371	**466933**	**465464**	**0.32**
	220 × 3	134315	133152	**1.54**	**0.002769707**	133864	132593	1.19	0.761775162	**133813**	**132284**	**1.16**
	220 × 5	195046	191619	3.66	**3.08E−09**	191619	189022	1.84	0.270595298	**191007**	**188164**	**1.51**
	220 × 8	574455	**572295**	0.38	0.05839993	**574434**	572368	**0.37**	0.058238831	576001	573190	0.65
	240 × 3	316877	314004	**1.59**	**0.012665618**	316583	312268	1.5	0.140247912	**315201**	**311917**	**1.05**
	240 × 5	648484	644985	0.62	0.930009212	648461	**644486**	0.62	0.947534735	**648382**	644652	**0.6**
	240 × 8	685135	681330	0.65	0.559808155	684873	681331	0.61	0.728888011	**684505**	**680705**	**0.56**
	260 × 3	262950	257293	4	**1.39E−14**	258610	255942	2.28	**0.00079726**	**256400**	**252838**	**1.41**
	260 × 5	769470	765346	0.59	0.84568644	**769136**	764995	**0.55**	0.956233118	769208	**764924**	0.56
	260 × 8	806615	801577	0.67	0.836698496	**806249**	801774	**0.62**	0.96978819	806305	**801279**	0.63
	280 × 3	440899	437590	1.37	0.649500066	442122	435644	1.65	0.17665878	**440524**	**434945**	**1.28**
	280 × 5	890818	885397	0.65	0.85664255	**890653**	**885043**	**0.63**	0.78695066	891133	885490	0.69
	280 × 8	**938827**	933533	**0.57**	0.965905677	938900	933741	0.58	0.995621734	938892	**933489**	0.58
	300 × 3	**496240**	490705	**1.34**	0.294432781	496289	**489655**	1.35	0.396885822	497458	492160	1.59
	300 × 5	273270	271104	1.28	**1.30E−09**	272678	271077	1.06	**4.32E−10**	**271030**	**269819**	**0.45**
	300 × 8	**1075634**	1069386	**0.59**	0.939716657	1076288	1069708	0.65	0.784352018	1075773	**1069291**	0.61

4.4. Neighborhood probabilistic selection strategies with blocking-based job

Due to the limitation of no buffers, jobs may be blocked. Blocking constraints prevent the entire process from progressing and extend the completion time of the scheduling sequence. Therefore, after adjusting the family SDST, it is necessary to sort the job sequence in families to reduce the impact of blocking on makespan. As far as we know, there is no corresponding strategy for blocking constraints in BHFGSP. Thus, in this subsection, we design neighborhood probabilistic selection strategies with blocking-based jobs to improve the solution.

In the proposed strategy, we firstly compute the blocking time of every job at each stage. Then, the sequences including job indexes are sorted in descending order according to blocking time. If the blocking time of jobs is 0, they are arranged after the blocking jobs. By doing this, the job blocked with the longest time will be paid attention to for the first time. If the family only contains one job, the operation on one job will be transformed into a family, i.e., executing the ***Single job swap operator*** to arrange the job sequence. In addition, the ***Job-random***, ***Job-greedy1*** and ***Job-greedy2 swap operators*** are executed for job sequences in the same family. Similar to neighborhood probabilistic selection strategies with family, the strategy for blocking jobs is also based on those swap operations. These operations can effectively reduce the overall movement time of the job sequence and make up for the lack of global search ability of the NIG algorithm.

Next, we will give the framework of neighborhood probabilistic selection strategies with blocking-based jobs and specific execution procedures of each operator. The details are shown in Algorithm 5.

The steps of ***Single job swap operator*** are given as follows:

Step1: find the family $JobInFam\left[Blocking_j\right]$, noted as π^g_{select}, swap the π^g_{select} with all other families in π^g_{select}.

Step2: Family sequence with the minimum objective value is denoted as π^{inter};

Step3: If $DP\left(\pi^{inter}, \tau^{inter}\right) <= DP\left(\pi^g, \tau^g\right)$, $\left(\pi^B, \tau^B\right) \leftarrow \left(\pi^{inter}, \tau^{inter}\right)$;

Otherwise, $\left(\pi^B, \tau^B\right) \leftarrow \left(\pi^g, \tau^g\right)$;

The steps of ***Job-random swap operator*** are given as follows:

Step1: Set $count = 1$, find a job $\tau^g_{select} = Blocking_j$, which belongs to the family $JobInFam\left[Blocking_j\right]$, $Pos = Select$. $\left(\pi^B, \tau^B\right) = \left(\pi^{inter}, \tau^{inter}\right) = \left(\pi^g, \tau^g\right)$;

Step2: While $count <= C$, randomly select a job τ^{inter}_{random} belonging to the same family as τ^{inter}_{Pos}, and $\tau^{inter}_{random} != \tau^{inter}_{Pos}$;

Step3: Swap the position of τ^{inter}_{random} with τ^{inter}_{Pos}, a new sequence $\left(\pi^{inter}, \tau^{inter}\right)$ is obtained;

Step4: If $DP\left(\pi^{inter}, \tau^{inter}\right) <= DP\left(\pi^B, \tau^B\right)$, $\left(\pi^B, \tau^B\right) \leftarrow \left(\pi^{inter}, \tau^{inter}\right)$, $Pos \leftarrow random$;

Otherwise, $\left(\pi^{inter}, \tau^{inter}\right) \leftarrow \left(\pi^B, \tau^B\right)$; $count++$;

Step5: Repeat steps 2–4 until $count > C$.

The steps of ***Job-greedy1 swap operator*** are given as follows:

Step1: Set $Pos = 1$, find a job $\tau^g_{select} = Blocking_j$ belonging to the family $JobInFam\left[Blocking_j\right]$, $\left(\pi^B, \tau^B\right) = \left(\pi^{inter}, \tau^{inter}\right) = \left(\pi^g, \tau^g\right)$;

Step2: While $Pos <= |JobInFam\left[Blocking_j\right]|$, swap the position of τ^{inter}_{select} with τ^{inter}_{Pos};

Step3: If $DP\left(\pi^{inter}, \tau^{inter}\right) <= DP\left(\pi^B, \tau^B\right)$, $\left(\pi^B, \tau^B\right) \leftarrow \left(\pi^{inter}, \tau^{inter}\right)$, $select \leftarrow Pos$;

Otherwise, $\left(\pi^{inter}, \tau^{inter}\right) \leftarrow \left(\pi^B, \tau^B\right)$;

Table 11
Comparison results with the neighborhood probabilistic selection strategies with blocking-based job when ρ = 100, F = 60.

	N× S	IGA				IGDLM				NIG		
		MEAN	BEST	RPD	p-value	MEAN	BEST	RPD	p-value	MEAN	BEST	RPD
	80 × 3	**46069**	46054	**0.05**	0.228982297	46085	**46048**	0.08	0.044088084	46073	46054	**0.05**
	80 × 5	53740	53730	0.25	**6.14E−18**	53700	53678	0.18	**2.24E−07**	**53659**	**53604**	**0.1**
	80 × 8	62198	62114	0.31	**6.19E−11**	62146	62121	0.23	**3.31E−11**	**62051**	**62006**	**0.07**
	100 × 3	94493	94457	0.07	**0.002716253**	94641	94571	0.23	**1.50E−26**	**94460**	**94423**	**0.04**
	100 × 5	105261	105224	0.25	**3.37E−10**	105343	105281	0.33	**2.49E−15**	**105086**	**104997**	**0.08**
	100 × 8	116480	116426	0.26	**3.57E−09**	116545	116349	0.31	**5.52E−11**	**116310**	**116181**	**0.11**
	120 × 3	73908	73453	0.88	**8.18E−07**	**73586**	73280	**0.44**	0.740935367	73602	**73261**	0.47
	120 × 5	82049	81495	2.28	**1.13E−20**	80926	80565	0.88	**0.000216945**	**80566**	**80217**	**0.43**
	120 × 8	175911	175845	0.38	**9.26E−16**	175730	175675	0.28	**6.91E−08**	**175447**	**175240**	**0.12**
	140 × 3	211994	211807	0.22	0.03142007	212169	212061	0.3	**3.16E−05**	**211809**	**211528**	**0.13**
	140 × 5	232750	232410	0.43	**8.21E−06**	232355	232129	0.26	0.367013518	**232257**	**231759**	**0.21**
	140 × 8	249360	248885	0.4	**0.000324203**	249056	248512	0.28	0.23845276	**248895**	**248356**	**0.22**
	160 × 3	284749	284341	0.27	0.120807624	**284444**	284143	**0.16**	0.679806689	284507	**283996**	0.18
	160 × 5	311498	311228	0.4	**8.67E−06**	311044	310818	0.26	0.036845466	**310727**	**310242**	**0.16**
	160 × 8	331602	331116	0.34	0.11480995	331606	331088	0.34	0.109037748	**331240**	**330469**	**0.23**
	180 × 3	**368597**	368109	**0.16**	0.241277886	369041	368620	0.28	0.722448471	368941	**367999**	0.26
	180 × 5	204461	202824	3.63	**8.82E−11**	**199304**	**197294**	**1.02**	**0.001776381**	201062	198653	1.91
	180 × 8	217711	216204	2.58	**3.29E−06**	**214759**	**212238**	**1.19**	**0.001119074**	216062	214516	1.8
F = 60	200 × 3	**458440**	457416	**0.23**	0.590364221	458630	457711	0.27	0.896607697	458689	**457408**	0.28
	200 × 5	488345	487032	0.35	0.929300051	**488229**	487129	**0.33**	0.901301421	488296	**486647**	0.34
	200 × 8	519651	517957	0.34	0.489178226	**519294**	**517899**	**0.27**	0.16303501	520065	518079	0.42
	220 × 3	559265	557410	0.33	0.863324247	**559137**	557532	**0.31**	0.733124962	559406	**557398**	0.36
	220 × 5	598030	595835	0.53	0.565502912	**597715**	**594851**	**0.48**	0.39763481	598560	595614	0.62
	220 × 8	632457	**629756**	0.43	0.975667277	632487	629913	0.43	0.943561272	**632432**	629924	**0.42**
	240 × 3	341807	338195	2.09	**0.023962747**	341381	337283	1.97	0.050959863	**339599**	**334797**	**1.43**
	240 × 5	362711	358357	2.35	0.089889385	**358338**	**354375**	**1.12**	**0.009129664**	361141	357828	1.91
	240 × 8	390546	384181	2.23	0.243159126	**385747**	**382032**	**0.97**	**0.000753833**	389238	383985	1.89
	260 × 3	775240	771059	0.81	**0.046501667**	775490	770569	0.84	**0.043341569**	**772400**	**769009**	**0.44**
	260 × 5	839085	834285	0.76	0.191519348	839078	834110	0.76	0.191582514	**837262**	**832782**	**0.54**
	260 × 8	884467	879893	0.68	0.079840249	886087	880486	0.87	**0.013900825**	**882335**	**878460**	**0.44**
	280 × 3	469686	465935	1.58	0.182943072	470092	**462381**	1.67	0.205326708	**468501**	464161	**1.32**
	280 × 5	**967441**	**960791**	**0.69**	0.875500388	967810	961035	0.73	0.982307923	967763	961375	0.73
	280 × 8	**1018316**	**1012227**	**0.6**	0.64290689	1018515	1012648	0.62	0.721454994	1019143	1012769	0.68
	300 × 3	540380	533007	2.2	**0.028588756**	539994	533509	2.13	0.021638306	**536813**	**528726**	**1.53**
	300 × 5	1103278	1095666	0.78	0.829973637	**1102466**	**1094752**	**0.7**	0.897072076	1102769	1095260	0.73
	300 × 8	601601	593393	1.73	**0.016761446**	**596162**	**591361**	**0.81**	0.225353785	597482	593013	1.03

Algorithm 5 Neighborhood probabilistic selection strategies with blocking-based job

Input: $JobInFam\,[j]$ ▷ record the family that the job j belongs to.
Output: $\left(\pi^B, \tau^B\right)$

$BlockingTime_j \leftarrow \sum_{s=1}^{S}\left(d_{j,s} - c_{j,s}\right),\ j = \{1, 2, ..., N\}$ ▷ Record the blocking duration of each job at all stages
Sort {*BlockingTime_1*, ..., *BlockingTime_j*, ..., *BlockingTime_N*} according to the descending order. If the blocking time of some jobs is 0, they are arranged sequentially behind the jobs whose blocking time is greater than 0. *Blocking_j* indicates the job that in position j of sequence *Blocking*.
for $j = 1$ **to** N **do** ▷ the position index of the job in sequence *Blocking*
  if $|JobInFam\left[Blocking_j\right]| = 1$ **then** ▷ the family which has only one job
    $\left(\pi^B, \tau^B\right) \leftarrow$ Execute the ***Single job swap operator*** $((\pi^g, \tau^g)$*JobInFam* [*Blocking_j*])
  else
    $rnd \leftarrow randi\,(1, 3)$ ▷ a random integer in [1, 3]
    if $rnd = 1$ **then**
      $\left(\pi^B, \tau^B\right) \leftarrow$ Execute the ***Job-random swap operator*** $((\pi^g, \tau^g), JobInFam\left[Blocking_j\right])$
    else if $rnd = 2$ **then**
      $\left(\pi^B, \tau^B\right) \leftarrow$ Execute the ***Job-greedy1 swap operator*** $((\pi^g, \tau^g), JobInFam\left[Blocking_j\right])$
    else
      $\left(\pi^B, \tau^B\right) \leftarrow$ Execute the ***Job-greedy2 swap operator*** $((\pi^g, \tau^g), JobInFam\left[Blocking_j\right])$
    end if
  end if
end for

Step4: $Pos++$, repeat steps 2–3 until $Pos > |JobInFam\left[Blocking_j\right]|$.

The steps of ***Job-greedy2 swap operator*** are given as follows:

Step1: Set $Pos = 1$, $\min Pos = -1$, find the job $\tau^g_{select} = Blocking_j$ belonging to the family $JobInFam\left[Blocking_j\right]$, $\left(\pi^B, \tau^B\right) = \left(\pi^{inter}, \tau^{inter}\right) = (\pi^g, \tau^g)$;

Step2: While $Pos <= |JobInFam\left[Blocking_j\right]|$, swap the position of τ^{inter}_{select} with τ^{inter}_{Pos}.

Step3: If $DP\left(\pi^{inter}, \tau^{inter}\right) <= DP\left(\pi^B, \tau^B\right)$, $\left(\pi^{inter}, \tau^{inter}\right) \leftarrow \left(\pi^B, \tau^B\right)$, $\min Pos \leftarrow Pos$;

Otherwise, $\left(\pi^{inter}, \tau^{inter}\right) \leftarrow \left(\pi^B, \tau^B\right)$;

Step4: $Pos++$, repeat the steps 2–3 until $Pos = JobInFam\left[Blocking_j\right].size\,()$;

Step5: If $\min Pos\ != -1$, swap positions of τ^B_{select} and $\tau^B_{\min Pos}$.

Table 12
Comparison results with the neighborhood probabilistic selection strategies with blocking-based job when $\rho = 200$, $F = 20$.

	N× S	IGA				IGDLM				NIG		
		MEAN	BEST	RPD	*p*-value	MEAN	BEST	RPD	*p*-value	MEAN	BEST	RPD
	80 × 3	3357	3344	0.91	**3.13E−21**	3347	3340	0.61	**2.95E−14**	**3337**	**3327**	**0.29**
	80 × 5	7081	7021	1.38	**4.85E−21**	7039	7007	0.78	**2.96E−09**	**7008**	**6984**	**0.35**
	80 × 8	10904	10842	1.67	**2.29E−13**	10870	10802	1.36	**8.09E−06**	**10821**	**10725**	**0.89**
	100 × 3	14663	14466	3.57	**1.34E−20**	14478	14369	2.27	**4.23E−10**	**14342**	**14157**	**1.3**
	100 × 5	53864	53840	0.1	**1.20E−06**	**53841**	**53809**	**0.06**	**3.75E−12**	53909	53881	0.19
	100 × 8	63062	63027	0.55	**1.40E−37**	**62768**	**62715**	**0.08**	**9.80E−17**	62901	62877	0.3
	120 × 3	31250	30876	3.51	**1.31E−26**	30813	30547	2.06	**2.34E−15**	**30503**	**30190**	**1.04**
	120 × 5	102229	102171	0.21	**1.49E−06**	102278	102209	0.26	**1.59E−10**	**102086**	**102014**	**0.07**
	120 × 8	58542	58310	1.66	**6.80E−22**	57942	57661	0.61	0.054470015	**57859**	**57588**	**0.47**
	140 × 3	73177	72634	2.04	**4.66E−08**	72679	72329	1.35	0.085606657	**72546**	**71711**	**1.16**
	140 × 5	80740	80311	1.35	**7.96E−07**	80432	79918	0.96	**0.005442362**	**80113**	**79664**	**0.56**
	140 × 8	89782	**88510**	1.44	**0.016936744**	89463	88828	1.08	0.361938189	**89368**	88819	**0.97**
	160 × 3	206119	205877	0.13	0.335040171	**206048**	**205846**	**0.1**	0.114923925	206217	205885	0.18
	160 × 5	115259	113745	2.05	**0.001455183**	114604	112983	1.47	0.216271919	**114197**	**112940**	**1.11**
	160 × 8	250096	249600	0.28	0.700851821	**250053**	**249390**	**0.27**	0.580178978	250171	249656	0.31
	180 × 3	278135	277645	0.28	0.67487345	278093	277609	0.26	0.815659051	**278035**	**277360**	**0.24**
	180 × 5	155823	153719	2.82	**0.00834422**	**153993**	**151554**	**1.61**	0.2793346	154588	152233	2
	180 × 8	326773	325939	0.33	0.694098921	326791	325886	0.34	0.663310832	**326647**	**325700**	**0.29**
F = 20	200 × 3	**357357**	356557	**0.22**	0.832828675	358045	356900	0.42	0.178983374	357437	**356555**	0.25
	200 × 5	387913	386037	0.51	0.899299465	**387895**	**385956**	**0.5**	0.87946115	387998	386063	0.53
	200 × 8	420454	418527	0.51	0.955957396	420516	418490	0.52	0.879816981	**420420**	**418337**	**0.5**
	220 × 3	161978	160338	2.69	**0.000153206**	161569	159727	2.43	**0.006226092**	**160589**	**157728**	**1.81**
	220 × 5	**245253**	242413	1.35	**0.04586923**	246245	**241981**	**1.76**	0.532425192	246735	243252	1.96
	220 × 8	514629	**512059**	0.5	0.939981736	514751	512165	0.53	0.823469128	**514572**	512090	**0.49**
	240 × 3	**539807**	538060	**0.33**	0.946199977	539870	538029	0.35	0.984674341	539856	**538005**	0.34
	240 × 5	299263	296490	1.41	0.992426481	299996	**295112**	1.66	0.385063108	**299257**	296828	**1.4**
	240 × 8	617828	614662	0.53	0.999547938	**617634**	614677	**0.5**	0.844073926	617827	**614558**	0.53
	260 × 3	650604	648376	0.39	0.803834004	650726	648518	0.41	0.701004978	**650389**	**648077**	**0.36**
	260 × 5	695503	691763	0.57	0.921271035	**695127**	**691529**	**0.52**	0.680526746	695623	691859	0.59
	260 × 8	739148	735310	0.58	0.813269836	739004	735096	0.56	0.906787586	**738867**	**734864**	**0.54**
	280 × 3	768906	766162	0.39	0.993745208	**768698**	766288	**0.36**	0.842460478	768898	**765910**	0.39
	280 × 5	808849	804732	**0.53**	0.990334254	808992	804888	0.55	0.904820551	**808832**	**804594**	**0.53**
	280 × 8	863658	859595	0.59	0.733384925	863715	859679	0.59	0.693669394	**863212**	**858619**	**0.53**
	300 × 3	462942	459392	2.18	0.91098638	**460560**	**453052**	**1.66**	0.065083819	463060	457415	2.21
	300 × 5	947732	942147	0.59	0.960656375	**947388**	**942143**	**0.56**	0.879002819	947647	942325	0.58
	300 × 8	999194	993805	**0.58**	0.978559569	999644	994044	0.63	0.765387036	**999149**	**993401**	**0.58**

4.5. The computational complexity of NIG

Assume that there are N jobs, F families and S stages. The computational complexity of the whole NIG algorithm mainly consists of *initialization*, *destruction–construction*, *neighborhood probabilistic selection strategies with family* and *blocking-based job*. The time complexity of the *initialization strategy* is $O(F^*F^*S)$ that approaches $O(F^2 * S)$. The time complexity of the *destruction–construction* strategy is $O(w_1 * d * F * S)$, where w_1 is the number of iterations of *destruction–construction* and d is the parameter of the strategy. In the *neighborhood probabilistic selection strategies with family*, the time complexities of the ***Family-random swap operator***, ***Family-disturb swap operator***, ***Family-iterative swap operator***, and ***Family-sequential swap operator*** are $O(R * S)$, $O(F * F * S)$, $O(F * F * S)$, and $O(F * F * S)$, respectively. According to the following experimental parameter settings, the maximum number of times of R is $4 * F$. Therefore, the computational complexity of the *neighborhood probabilistic selection strategies with family* is $O(w_2 * F^2 * S)$, where w_2 is the number of iterations of the family operator. Similarly, in the *neighborhood probabilistic selection strategies with blocking-based job*, the time complexities of the ***Single job swap operator***, the ***Job-random***, ***Job-greedy1*** and ***Job-greedy2 swap operators*** are $O(N^*N^*S)$, $O(4^*N^*N^*S)$, $O(N^*N^*S)$, and $O(N^*S)$, respectively. The computational complexity of the *neighborhood probabilistic selection strategies with blocking-based job* is $O(4 * w_3 * N^2 * S)$, where w_3 is the number of iterations of the job operator. Thus, for the whole NIG algorithm, the time complexity of the NIG is $O(S(F^2 + w_1 * d * F + w_2 * F^2 + 4 * w_3 * N^2))$.

5. Experimental results and analysis

5.1. Experimental environment settings and evaluating indicators

To test the performance of NIG algorithm for solving BHFGSP, the following test instances are utilized. Refer to DFGSP [22], HFSP [52], and BHFSP [10,53], we consider the scale settings: $S = \{3, 5, 8\}$, $F = \{20, 40, 60\}$, and $N = \{80, 100, 120, 140, 160, 180, 200, 220, 240, 260, 280, 300\}$. This leads to a set of $12 \times 3 \times 3 = 108$ different combinations. For each combinational instance, integer processing time $p_{j,s}$ and SDST set_{s,f_1,f_2} are randomly sampled in uniformly range of [50, 99] and [10, 20], respectively. All strategies are coded by C++ programming language in the Visual Studio 2019 experiment. The host configuration is Windows 10 Operation System in 64-bit with Intel Core i7 CPU, 2.60 GHZ, 16 GB RAM.

Refer to [22,54], we set the termination criterion using maximum elapsed running time ($\rho \times$ F $\times$ S milliseconds) for test instances and comparisons. Parameter ρ has two levels: 100 and 200. According to two different time frames, the influence of different termination time on results can be observed more intuitively and comprehensively. If running time of algorithms reach the termination condition, the procedure ends. Otherwise, algorithms will continue to execute in the next iteration. It is noteworthy that in some large-scale instances, running time of algorithms may be slightly greater than that of the set termination time, but it is a normal phenomenon. For all instances, the Relative Percentage Deviation (RPD) is computed as a Response Variable (RV) in Table 3. It measures the deviation of individual measurement results from the average value, and has been widely used in many literature [55–57]. In order to enable the reader to

Table 13
Comparison results with the neighborhood probabilistic selection strategies with blocking-based job when ρ = 200, F = 40.

	N× S	IGA				IGDLM				NIG		
		MEAN	BEST	RPD	p-value	MEAN	BEST	RPD	p-value	MEAN	BEST	RPD
	80 × 3	6744	6549	4.65	**5.50E−06**	6716	6634	4.23	**0.000167092**	**6664**	**6444**	**3.42**
	80 × 5	17115	17060	1.87	**8.45E−29**	16975	16867	1.04	**4.05E−09**	**16895**	**16801**	**0.56**
	80 × 8	**41076**	**41061**	**0.04**	**2.05E−47**	41175	41171	0.28	**1.74E−47**	41249	41243	0.46
	100 × 3	23388	23172	3.68	**5.72E−18**	23077	22773	2.3	**0.00828988**	**22975**	**22559**	**1.84**
	100 × 5	79855	79772	0.27	**1.57E−18**	**79700**	79670	**0.07**	0.983520695	79701	**79643**	**0.07**
	100 × 8	90338	90306	0.2	**1.03E−05**	90497	90435	0.38	**5.57E−18**	**90243**	**90154**	**0.1**
	120 × 3	60067	59667	1.41	**1.80E−16**	59739	59360	0.86	**0.000282401**	**59593**	**59232**	**0.61**
	120 × 5	131346	131255	0.32	**2.90E−05**	131518	131411	0.45	**6.46E−13**	**131170**	**130933**	**0.18**
	120 × 8	147290	147183	0.4	**4.11E−14**	147240	147135	0.37	**6.06E−12**	**146891**	**146702**	**0.13**
	140 × 3	90583	89830	2.33	**1.02E−19**	**89051**	**88519**	**0.6**	0.287871692	89144	88741	0.71
	140 × 5	197885	197798	0.22	**0.013345846**	**197554**	**197455**	**0.05**	**0.013094302**	197724	197515	0.14
	140 × 8	212905	212791	0.18	**0.014275365**	**212722**	212586	**0.09**	0.820865674	212736	**212527**	0.1
	160 × 3	247725	247501	0.2	0.984163758	**247565**	**247236**	**0.13**	0.203531995	247722	247355	0.2
	160 × 5	268966	268628	0.4	**0.001746092**	269019	268620	0.42	**0.001826441**	**268489**	**267897**	**0.22**
	160 × 8	148746	147457	2.99	**5.38E−23**	**145567**	**144434**	**0.78**	0.750496304	145649	145060	0.84
	180 × 3	322183	321697	0.29	0.121253063	322276	321836	0.32	**0.0345957**	**321841**	**321251**	**0.18**
	180 × 5	**350107**	349661	**0.21**	0.213344485	350123	**349382**	**0.21**	0.3013809	350422	349565	0.3
	180 × 8	374379	373795	0.29	0.409660435	374234	**373315**	0.25	0.84357598	374171	**373317**	**0.23**
F = 40	200 × 3	212014	210223	3.23	**4.65E−08**	**207481**	**205380**	**1.02**	**0.023206806**	208952	205717	1.74
	200 × 5	159128	155510	4.21	**2.85E−21**	154543	153167	1.2	**0.018703316**	**153702**	**152705**	**0.65**
	200 × 8	466128	464902	0.3	0.777934152	466330	465284	0.34	0.468950873	**466000**	**464735**	**0.27**
	220 × 3	134254	132683	1.73	**2.16E−06**	133775	132495	1.37	**0.001185676**	**133154**	**131965**	**0.9**
	220 × 5	194600	190278	3.44	**1.52E−14**	191451	189134	1.76	**3.06E−08**	**189313**	**188137**	**0.62**
	220 × 8	574493	572061	0.44	0.852486069	**574027**	**571956**	**0.36**	0.401976625	574628	572180	0.47
	240 × 3	315759	312914	2.81	**3.15E−05**	317366	313117	3.34	**4.14E−07**	**311465**	**307120**	**1.41**
	240 × 5	647163	644010	0.68	0.126774543	647520	644326	0.73	0.05928952	**645581**	**642809**	**0.43**
	240 × 8	684740	681182	0.69	0.205896122	685024	681269	0.74	0.12606826	**683360**	**680025**	**0.49**
	260 × 3	259972	256395	3.42	**3.24E−08**	255794	252058	1.76	0.92677503	**255729**	**251379**	**1.73**
	260 × 5	769690	765286	0.66	0.732489191	769375	764792	0.62	0.910540264	**769212**	**764635**	**0.6**
	260 × 8	806708	**801924**	0.6	0.842814588	806676	801992	0.59	0.856721603	**806421**	801955	**0.56**
	280 × 3	**439274**	**433492**	**1.33**	0.061343667	442660	437892	2.11	0.556087712	441923	434927	1.94
	280 × 5	**890456**	885171	**0.62**	0.823237978	890735	885058	0.66	0.950058627	890846	**884935**	0.67
	280 × 8	**938699**	**933591**	**0.55**	0.762918005	939208	933942	0.6	0.981613001	939172	933673	0.6
	300 × 3	**495556**	**489499**	**1.24**	**0.006594044**	497364	492065	1.61	0.125098096	499559	493921	2.06
	300 × 5	274124	270773	1.76	**3.98E−09**	271793	270035	0.89	0.283219824	**271437**	**269388**	**0.76**
	300 × 8	1075543	1069100	**0.62**	0.997225878	1075763	1069573	0.64	0.908428263	**1075536**	**1068901**	**0.62**

better reproduce the algorithms used in this paper, we have made the code publicly available and uploaded it to Github "https://github.com/klaette/BHFGSP".

5.2. Parameter settings and sensitivity analysis

NIG algorithm includes 3 parameters to be tested: (1) number of jobs, d, extracted by destruction–construction strategy; (2) number of executions, R, in ***Family-random swap operator***. (3) number of executions, C, in ***Job-random swap operator***. F indicates the number of families, E means the number of jobs in selected family. To calibrate the effect of different parameters on the proposed algorithm, we utilized the Taguchi method [58] to conduct the experiment tests.

Four levels are set for above-mentioned three parameters, i.e. $d = \{2, 3, 4, 5\}$, $R = \{F, 2F, 3F, 4F\}$, $C = \{E, 2E, 3E, 4E\}$. Then, as shown in Table 3, we give an orthogonal array L_{16} (4^3) with 16 (d, R, C) combinations to test the performance of parameters separately. In order to reduce the difference of results, we use four large span examples, i.e., 80 × 3 × 20, 120 × 3 × 20, 160 × 5 × 40, and 200 × 5 × 40 to investigate effect of parameters. For each instance of (d, R, C) combination, 30 independent replications are performed and corresponding RV values are computed. According to RV values which are listed in Table 3, Table 4 shows the mean RV values and significance rank of each parameter. In Table 4, Delta represents the gap of RV among different levels of the parameters, rank indicates the influence level of parameters on algorithm performance. In addition, Fig. 4 displayed the trend of the factor level of each parameter.

As can be seen from Table 4 and Fig. 4, parameter d is the most influential and important parameter, followed by C and R. From obtained results, the minimum RV value is obtained when $d = 3$. The reason may be that too small d value (d<3) reduces the local search ability of the algorithm, resulting in weak intensification. However, too large d value (d>3) may disturb the current solution too much and spend more time, resulting in the performance degradation of NIG algorithm. For parameters C and R, they are not particularly influential on the performance of NIG algorithm. Because these are only two parameters in the neighborhood search strategies with family and blocking-based job, they are only selected according to the probability in each iteration of the algorithm, thus, the influence of parameter value is small. According to comparison results shown in Table 4 and Fig. 4, we finally set the parameter values as: $d = 3$, $C = 3E$, $R = 4F$.

5.3. Evaluation of the MILP model

To verify correctness and effectiveness of the mathematical model and NIG algorithm, we selected 10 small-scale instances in the subsection. Mathematical model of BHFGSP is coded in the mathematical programming solver CPLEX Studio IDE. We set the maximum elapsed running time of mathematical model as 1000s. For the proposed NIG algorithm, we also choose the termination criterion "$\rho \times$ F × S milliseconds" mentioned in Section 5.1. The parameter setting of NIG algorithm refers to Section 5.2: $d = 3$, $C = 3E$, $R = 4F$. For each instance, NIG algorithm repeated independently for 30 times and average values of results are computed as "Makespan". Table 5 lists the relevant operation results of the mathematical model and the NIG algorithm. For

Table 14
Comparison results with the neighborhood probabilistic selection strategies with blocking-based job when ρ = 200, F = 60.

	N× S	IGA				IGDLM				NIG		
		MEAN	BEST	RPD	*p*-value	MEAN	BEST	RPD	*p*-value	MEAN	BEST	RPD
	80 × 3	46003	45989	0.12	**9.12E−08**	**45959**	**45946**	**0.03**	**1.74E−33**	46018	46005	0.16
	80 × 5	53785	53771	0.29	**2.71E−64**	53760	53732	0.25	**3.62E−43**	**53634**	**53627**	**0.01**
	80 × 8	62150	62113	0.48	**8.23E−29**	62049	61992	0.32	**2.88E−20**	**61903**	**61850**	**0.09**
	100 × 3	94552	94501	0.15	**1.09E−13**	94600	94560	0.2	**8.23E−19**	**94445**	**94407**	**0.04**
	100 × 5	105239	105209	0.38	**2.57E−18**	105260	105242	0.4	**3.73E−21**	**104941**	**104842**	**0.09**
	100 × 8	116526	116443	0.41	**4.39E−20**	116387	116308	0.29	**1.02E−12**	**116152**	**116052**	**0.09**
	120 × 3	73731	73434	1.15	**1.07E−13**	73398	**72896**	0.69	0.235447443	**73333**	73001	**0.6**
	120 × 5	81228	80763	1.04	**4.30E−09**	80822	**80388**	**0.54**	0.681577417	**80841**	80618	0.56
	120 × 8	175774	175745	0.26	**3.40E−17**	175646	175551	0.18	**6.04E−08**	**175428**	**175327**	**0.06**
	140 × 3	212118	212000	0.26	**3.07E−06**	211965	211781	0.19	**0.024956376**	**211819**	**211566**	**0.12**
	140 × 5	232494	232322	0.25	**0.000432827**	**232170**	231981	**0.11**	0.620316909	232212	**231919**	0.13
	140 × 8	249152	249053	0.36	**4.92E−08**	248859	248569	0.24	**0.048977746**	**248676**	**248256**	**0.17**
	160 × 3	284613	**284234**	0.13	0.688522501	284735	284577	0.18	0.102554882	**284566**	284245	**0.12**
	160 × 5	310764	310497	0.19	0.006297581	310796	310564	0.2	**0.004171728**	**310475**	**310173**	**0.1**
	160 × 8	331570	331230	0.4	**9.18E−05**	330971	330446	0.22	0.694221303	**330897**	**330242**	**0.2**
	180 × 3	368783	368384	0.23	0.390657404	368662	368394	0.2	0.674353823	**368569**	**367922**	**0.18**
	180 × 5	200621	198808	1.86	0.436722213	**198625**	**196950**	**0.85**	**0.000337376**	200297	198152	1.7
	180 × 8	216087	214455	1.84	**0.001605839**	**214012**	**212181**	**0.86**	**0.026116468**	214925	213034	1.29
F = 60	200 × 3	458374	457722	0.32	0.457827623	458474	457675	0.34	0.30538649	**458087**	**456899**	**0.26**
	200 × 5	487840	486715	0.28	0.699677263	488099	487177	0.33	0.328956145	**487665**	**486484**	**0.24**
	200 × 8	519349	518151	0.43	0.566346624	**518658**	**517128**	**0.3**	0.349210929	519101	517976	0.38
	220 × 3	558878	557385	0.38	0.393015029	559075	557549	0.41	0.242843075	**558345**	**556784**	**0.28**
	220 × 5	**597393**	595258	**0.37**	0.79756142	597514	595351	0.39	0.911494144	597604	**595174**	0.41
	220 × 8	**630763**	628797	**0.33**	0.591379267	630839	**628685**	0.34	0.667339896	631150	628825	0.39
	240 × 3	342002	338476	1.54	0.024105857	340478	**336802**	1.09	0.616934534	**340057**	337153	**0.97**
	240 × 5	360269	356867	1.35	0.938539633	361793	**355485**	1.77	0.115267508	**360201**	356387	**1.33**
	240 × 8	385465	383019	0.88	0.352065277	385846	382158	0.98	0.200188377	**384879**	**382113**	**0.72**
	260 × 3	772628	769676	0.46	0.805087972	772420	769330	0.43	0.94971694	**772343**	**769113**	**0.42**
	260 × 5	**835261**	**831430**	**0.46**	0.894578323	836213	832642	0.58	0.508490487	835419	831977	0.48
	260 × 8	883028	879507	0.57	0.494564781	**882178**	**878036**	**0.47**	0.992089642	882191	878447	**0.47**
	280 × 3	466939	461342	2.96	0.371973097	**464234**	**453503**	**2.37**	0.031029319	468204	461322	3.24
	280 × 5	**963544**	958831	**0.51**	0.951834659	963743	959291	0.53	0.94408362	963635	**958677**	0.52
	280 × 8	1014427	**1009742**	0.46	0.831375352	1014722	**1009986**	0.49	0.667308285	1014122	1009934	**0.43**
	300 × 3	535822	532042	**1.01**	0.666540872	540251	**533739**	1.85	0.003179708	536377	**530446**	1.12
	300 × 5	1103136	1095267	0.72	0.792770162	**1102507**	**1095239**	**0.66**	0.592307602	1103767	1095645	0.78
	300 × 8	**595564**	592373	**0.7**	0.253868185	597433	593146	1.02	0.442155235	596674	**591426**	0.89

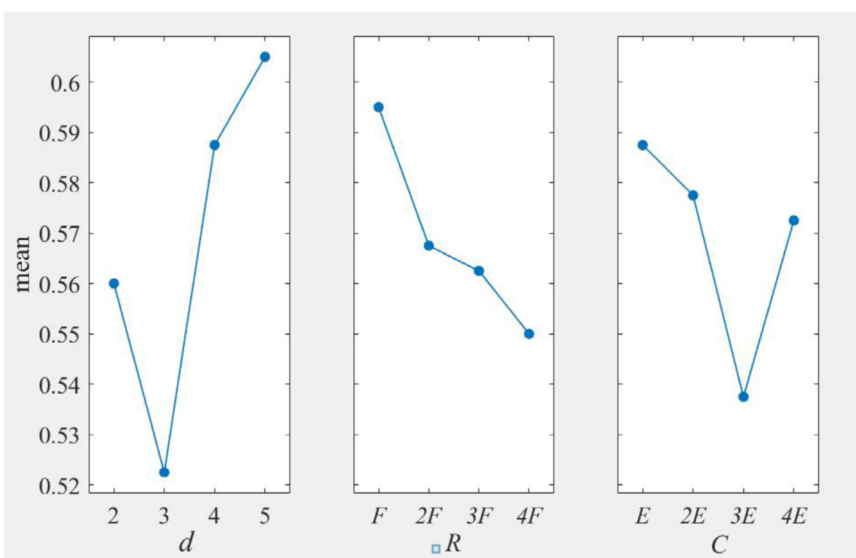


Fig. 4. The trend of the factor level.

mathematical model, we counted and calculated the number of the constraints, makespan, RPD values, and actual running time. For NIG algorithm, we counted makespan, RPD values and actual running time.

For small-scale instances, with the increasing of the problem size, the number of mathematical model constraints continues to grow. Mathematical model of BHFGSP gets the best makespan and RPD values in all instances, while the NIG algorithm only get best values in the first five instances. However, in first five instances, except for the first instance, NIG algorithm takes less time than the mathematical model. For other instances, except for the last two instances, results obtained by mathematical model are better than those obtained by NIG algorithm, but the running time of the mathematical model is very long. It can be seen that with the continuous increase of problem scale, time cost of the mathematical model will also increase sharply. In last two

Table 15
Comparison results without the neighborhood probabilistic selection strategies with blocking-based job when $\rho = 100$, $F = 20$.

	N× S	IGA				IGDLM				NIG		
		MEAN	BEST	RPD	*p*-value	MEAN	BEST	RPD	*p*-value	MEAN	BEST	RPD
	80 × 3	3592	3585	7.59	**0**	3590	3582	7.5	**0**	**3348**	**3339**	**0.28**
	80 × 5	7392	7345	5.53	**0**	7355	7335	5	**0**	**7035**	**7005**	**0.43**
	80 × 8	11538	11459	6.7	**0**	11451	11439	5.9	**0**	**10869**	**10813**	**0.52**
	100 × 3	15219	15201	6.42	**0**	15197	15157	6.27	**0**	**14499**	**14300**	**1.39**
	100 × 5	57201	57188	6.12	**0**	57197	57188	6.11	**0**	**53923**	**53901**	**0.04**
	100 × 8	66791	66783	6.47	**0**	66789	66779	6.47	**0**	**62810**	**62732**	**0.12**
	120 × 3	31834	31791	4.33	**0**	31793	31749	4.2	**0**	**30897**	**30512**	**1.26**
	120 × 5	109290	109275	6.89	**0**	109292	109282	6.89	**0**	**102303**	**102247**	**0.05**
	120 × 8	61195	61082	5.82	**0**	61081	61059	5.62	**0**	**58185**	**57829**	**0.62**
	140 × 3	77439	77343	6.94	**0**	77283	77259	6.72	**0**	**72889**	**72416**	**0.65**
	140 × 5	84883	84754	6.24	**0**	84770	84732	6.1	**0**	**80443**	**79896**	**0.68**
	140 × 8	93869	93686	5.36	**0**	93725	93675	5.2	**0**	**89766**	**89095**	**0.75**
	160 × 3	222153	222148	7.86	**0**	222138	222132	7.85	**0**	**206302**	**205971**	**0.16**
	160 × 5	121663	121551	6.13	**0**	121561	121531	6.04	**0**	**115744**	**114636**	**0.97**
	160 × 8	265960	265932	6.49	**0**	265940	265925	6.48	**0**	**250505**	**249754**	**0.3**
	180 × 3	300409	300404	8.2	**0**	300407	300397	8.2	**0**	**278264**	**277651**	**0.22**
	180 × 5	162820	162523	5.98	**0**	162590	162513	5.83	**0**	**155471**	**153635**	**1.2**
	180 × 8	346935	346914	6.36	**0**	346921	346913	6.35	**0**	**327249**	**326197**	**0.32**
F = 20	200 × 3	388570	388565	9	**0**	388572	388568	9	**0**	**357692**	**356474**	**0.34**
	200 × 5	418205	418193	8.39	**0**	418207	418196	8.39	**0**	**387846**	**385836**	**0.52**
	200 × 8	446983	446972	6.79	**0**	446963	446961	6.79	**0**	**420406**	**418561**	**0.44**
	220 × 3	165399	165117	3.52	**0**	165197	164976	3.39	**0**	**162606**	**159779**	**1.77**
	220 × 5	256638	256387	5.58	**0**	256402	256369	5.49	**0**	**246357**	**243067**	**1.35**
	220 × 8	545786	545768	6.64	**0**	545787	545760	6.64	**0**	**514384**	**511802**	**0.5**
	240 × 3	587250	587244	9.09	**0**	587251	587248	9.09	**0**	**540151**	**538294**	**0.34**
	240 × 5	312050	311566	5.52	**0**	311637	311523	5.38	**0**	**299173**	**295714**	**1.17**
	240 × 8	660151	660122	7.41	**0**	660132	660122	7.4	**0**	**617769**	**614620**	**0.51**
	260 × 3	704790	704785	8.68	**0**	704795	704782	8.68	**0**	**650794**	**648528**	**0.35**
	260 × 5	751061	751055	8.55	**0**	751070	751063	8.55	**0**	**695308**	**691882**	**0.5**
	260 × 8	786863	786831	7.08	**0**	786847	786837	7.07	**0**	**739055**	**734863**	**0.57**
	280 × 3	835004	834996	8.99	**0**	835003	835002	8.99	**0**	**769061**	**766127**	**0.38**
	280 × 5	872683	872654	8.47	**0**	872680	872658	8.47	**0**	**808825**	**804507**	**0.54**
	280 × 8	919293	919268	6.95	**0**	919287	919277	6.95	**0**	**863682**	**859529**	**0.48**
	300 × 3	484586	484023	6.25	**0**	484268	484017	6.18	**0**	**463111**	**456088**	**1.54**
	300 × 5	1026776	1026758	8.94	**0**	1026775	1026769	8.94	**0**	**947670**	**942500**	**0.55**
	300 × 8	1066707	1066697	7.34	**0**	1066718	1066696	7.34	**0**	**999444**	**993741**	**0.57**

instances, results obtained by NIG algorithm are better than those obtained by the mathematical model. For large scale problems, mathematical model may not even be able to get a feasible result within limited time. Therefore, from the overall results, the proposed NIG algorithm can better solve the BHFGSP.

5.4. Evaluation of the NIG algorithm strategies

In this subsection, we will investigate the performance of neighborhood probabilistic selection strategies with family and blocking-based job. As shown in Tables 6, 7, and 8, NIG algorithms without family and blocking-based job selection strategies are denoted as NIG_N_F and NIG_N_J, respectively. NIG algorithm includes both neighborhood probabilistic selection strategies. As shown in Tables 6, 7, and 8, for each instance, 30 independently repeated experiments were performed by all algorithms. MEAN and BEST values are the mean and best values of makespan counted by 30 numerical results. RPD measures the deviation of individual measurement results from average values. In addition, we performed statistical tests and analyses to verify whether there is a significant difference between comparison algorithms and NIG in each instance. In Tables 6, 7, and 8, *p*-value are obtained by comparing results of 30 replications in NIG_N_F/NIG and NIG_N_J/NIG respectively. If *p*-value is less than the significance level of 0.05, reject the original hypothesis, indicating that there is a significant difference between the two compared algorithms; Otherwise, accept the hypothesis, and difference between above algorithms is not obvious or there is no difference. Besides, we marked black bold for best value and *p*-value with significant difference.

From Tables 6, 7, and 8, we can observe that NIG_N_J has no optimal value in MEAN, BEST and RPD values for all 108 instances, and results of *p*-value are close to 0, suggesting that the NIG_N_J are significantly different from NIG algorithm. It can be seen that neighborhood probabilistic selection strategy designed for blocking constraints has a great impact on the performance of algorithms. This strategy effectively improves the global search ability of algorithms, further increasing the diversity of solution, and reducing the impact of blocking on the completion time of scheduling sequence. In addition, the strategy designed for job blocking constraints can effectively improve the quality of solutions by changing the arrangement position of blocked jobs in the group, thereby greatly reducing the completion time of the sequence. The above results prove that the proposed family and blocking-based job selection strategies are feasible and effective. From the results listed in Tables 6, 7, and 8, NIG_N_F(NIG) gets 24/108(84/108), 20/108(88/108), and 22/108(86/108) with respect to MEAN, BEST, RPD indicators, respectively. Furthermore, there are 49 results with significant differences between the two algorithms. It can be seen that NIG algorithm outperforms the NIG_N_F algorithm. To a certain extent, neighborhood probabilistic selection strategies with family reduce the SDST between different families and makes the arrangement of jobs more reasonable. The completion time of the scheduling sequence is also reduced by this strategy. This may be that: after the sequence between different groups is changed, the blocking of jobs within the group will change, and the machine setting time may be reduced. Therefore, objective value of scheduling sequence is further optimized. However, through the experimental comparison, it can be seen that the impact of neighborhood probability selection strategy designed for blocking constraints is greater than the

Table 16
Comparison results without the neighborhood probabilistic selection strategies with blocking-based job when $\rho = 100$, $F = 40$.

	N× S	IGA				IGDLM				NIG		
		MEAN	BEST	RPD	p-value	MEAN	BEST	RPD	p-value	MEAN	BEST	RPD
	80 × 3	7114	7085	7.72	**0**	7062	7022	6.93	**0**	**6727**	**6604**	**1.86**
	80 × 5	17931	17887	6.3	**0**	17902	17863	6.13	**0**	**16994**	**16868**	**0.74**
	80 × 8	43216	43210	5.02	**0**	43219	43207	5.02	**0**	**41189**	**41152**	**0.09**
	100 × 3	24575	24549	7.12	**0**	24562	24525	7.06	**0**	**23183**	**22942**	**1.05**
	100 × 5	84840	84804	6.33	**0**	84836	84826	6.33	**0**	**79804**	**79789**	**0.02**
	100 × 8	94991	94971	5.13	**0**	94998	94979	5.14	**0**	**90419**	**90353**	**0.07**
	120 × 3	63365	63322	6.32	**0**	63330	63297	6.26	**0**	**59821**	**59599**	**0.37**
	120 × 5	139556	139546	6.41	**0**	139557	139549	6.41	**0**	**131321**	**131155**	**0.13**
	120 × 8	154526	154505	5.06	**0**	154545	154528	5.07	**0**	**147207**	**147084**	**0.08**
	140 × 3	94803	94725	6.57	**0**	94718	94656	6.48	**0**	**89295**	**88955**	**0.38**
	140 × 5	211144	211114	6.6	**0**	211123	211120	6.59	**0**	**198189**	**198074**	**0.06**
	140 × 8	224811	224811	5.82	**0**	224806	224806	5.82	**0**	**212661**	**212447**	**0.1**
	160 × 3	266005	265993	7.45	**0**	265997	265980	7.44	**0**	**247967**	**247570**	**0.16**
	160 × 5	287159	287147	6.82	**0**	287168	287168	6.83	**0**	**269182**	**268817**	**0.14**
	160 × 8	153395	153354	5.53	**0**	153310	153262	5.47	**0**	**146195**	**145353**	**0.58**
	180 × 3	347714	347686	8.02	**0**	347694	347683	8.01	**0**	**322471**	**321900**	**0.18**
	180 × 5	374238	374237	6.93	**0**	374218	374200	6.93	**0**	**350772**	**349977**	**0.23**
	180 × 8	394847	394803	5.71	**0**	394818	394811	5.7	**0**	**374694**	**373522**	**0.31**
F = 40	200 × 3	220424	220294	6.69	**0**	220301	220227	6.63	**0**	**209800**	**206605**	**1.55**
	200 × 5	156858	156421	2.22	**0**	156396	156225	1.92	**0**	**155042**	**153454**	**1.04**
	200 × 8	495183	495166	6.35	**0**	495157	495142	6.35	**0**	**467137**	**465608**	**0.33**
	220 × 3	135777	135519	2.4	**0**	135487	135344	2.18	**0**	**133864**	**132593**	**0.96**
	220 × 5	193111	192665	2.16	**0**	192693	192481	1.94	**0**	**191619**	**189022**	**1.37**
	220 × 8	608345	608343	6.29	**0**	608301	608293	6.28	**0**	**574434**	**572368**	**0.36**
	240 × 3	328397	328166	5.17	**0**	328186	328106	5.1	**0**	**316583**	**312268**	**1.38**
	240 × 5	693720	693717	7.64	**0**	693712	693694	7.64	**0**	**648461**	**644486**	**0.62**
	240 × 8	722666	722660	6.07	**0**	722680	722663	6.07	**0**	**684873**	**681331**	**0.52**
	260 × 3	261617	260802	2.22	**0**	261046	260643	1.99	**0**	**258610**	**255942**	**1.04**
	260 × 5	822035	822033	7.46	**0**	822028	822017	7.46	**0**	**769136**	**764995**	**0.54**
	260 × 8	856642	856642	6.84	**0**	856636	856631	6.84	**0**	**806249**	**801774**	**0.56**
	280 × 3	457308	457084	4.97	**0**	457098	457064	4.92	**0**	**442122**	**435644**	**1.49**
	280 × 5	959763	959720	8.44	**0**	959733	959719	8.44	**0**	**890653**	**885043**	**0.63**
	280 × 8	993260	993225	6.37	**0**	993252	993237	6.37	**0**	**938900**	**933741**	**0.55**
	300 × 3	516014	515806	5.38	**0**	515748	515705	5.33	**0**	**496289**	**489655**	**1.35**
	300 × 5	274247	273864	1.17	**0**	273832	273512	1.02	**0**	**272678**	**271077**	**0.59**
	300 × 8	1145455	1145443	7.08	**0**	1145410	1145410	7.08	**0**	**1076288**	**1069708**	**0.62**

one designed for family constraints, which is a more important strategy for NIG algorithm. In the next subsection, we will further compare and analyze the neighborhood probabilistic selection strategies with blocking-based job to verify the performance of NIG algorithm.

5.5. Effectiveness of the NIG algorithm

To the best knowledge of us, no existing algorithm is designed for solving BHFGSP. Therefore, in this paper, we select the IGA [59] (2020) and IGDLM [53] (2021) algorithms as the compared algorithms. IGA and IGDLM are used to solve HFSP and BHFSP, respectively, that are similar to BHFGSP. For the sake of comparison, all algorithms adopt the same encoding and decoding method. In addition, for IGA and IGDLM, all strategies are designed for job sequence. If they are applied directly to solve the BHFGSP, it would inevitably break the constraint that jobs cannot be manipulated across families. To solve this problem mentioned above, we change all strategies designed for jobs to those for families, which ensures reducibility of the original algorithm and also satisfies the condition of families well. However, to verify the performance of neighborhood probabilistic selection strategies with blocking-based job mentioned above and ensure the fairness of algorithms, we also embed the strategy designed in this research into IGA and IGDLM for an additional comparative experiment.

In this way, we have two groups of comparative experiments with IGA and IGDLM: (1) Algorithm execution process in the original literature is preserved, but strategy for jobs is transformed to families. (2) Neighborhood probabilistic selection strategies with blocking-based job are embedded into the IGA and IGDLM to verify their performance, which can make the experiment more fairness. Then, for best performance of algorithms, all parameters in IGA and IGDLM are set according to the original literature. Furthermore, in order to observe the performance of algorithms in different time ranges, we also set the termination condition '$\rho \times$ F $\times$ S' as two levels: $\rho = 100$, $\rho = 200$, respectively. Then, we got such tables, which have the combination of different strategies and termination conditions. All these Tables have the same evaluation indicators, i.e., MEAN, BEST, RPD, and p-value.

As can be seen from Tables 9, 10, and 11, for MEAN, BEST, and RPD, IGA obtains 17/108, 11/108, 22/108 minimum values, respectively. IGDLM obtains 32/108, 23/108, 22/108 minimum values. NIG obtains 59/108, 75/108, 54/108 minimum values. In p-value results, IGA, IGDLM have 51 and 35 significantly different results from NIG, respectively. From Tables 12, 13, and 14, IGA gets 14/108, 10/108, 16/108 minimum MEAN, BEST, and RPD values. IGDLM gets 37/108, 33/108, 30/108 minimum values. NIG gets 67/108, 67/108, 67/108 minimum values. There are 52 and 38 significant difference results in IGA and IGDLM. It can be seen from these results, NIG outperforms other two comparison algorithms in MEAN, BEST, RPD values. NIG has the largest number of best solutions, and with the extension of termination conditions, overall performance of NIG algorithm is also improving. In fact, it can be seen from the p-value, though the performance of NIG algorithm is better than that of IGA and IGDLM, the difference of solutions is not large. The reason is not only the strong local search ability of the IG series algorithms, but also the embedding of neighborhood probabilistic selection strategies with blocking-based job. Therefore, it effectively reduces the blocking time of scheduling sequence.

Table 17
Comparison results without the neighborhood probabilistic selection strategies with blocking-based job when $\rho = 100$, $F = 60$.

	N× S	IGA				IGDLM				NIG		
		MEAN	BEST	RPD	*p*-value	MEAN	BEST	RPD	*p*-value	MEAN	BEST	RPD
	80 × 3	48178	48178	4.63	**0**	48179	48179	4.63	**0**	**46085**	**46048**	**0.08**
	80 × 5	56421	56408	5.11	**0**	56414	56404	5.1	**0**	**53700**	**53678**	**0.04**
	80 × 8	64504	64500	3.84	**0**	64519	64519	3.86	**0**	**62146**	**62121**	**0.04**
	100 × 3	100252	100252	6.01	**0**	100256	100256	6.01	**0**	**94641**	**94571**	**0.07**
	100 × 5	110945	110921	5.38	**0**	110922	110922	5.36	**0**	**105343**	**105281**	**0.06**
	100 × 8	121403	121384	4.34	**0**	121388	121382	4.33	**0**	**116545**	**116349**	**0.17**
	120 × 3	78463	78341	7.07	**0**	78370	78302	6.95	**0**	**73586**	**73280**	**0.42**
	120 × 5	86175	86090	6.96	**0**	86098	86068	6.87	**0**	**80926**	**80565**	**0.45**
	120 × 8	185092	185086	5.36	**0**	185085	185085	5.36	**0**	**175730**	**175675**	**0.03**
	140 × 3	226729	226728	6.92	**0**	226770	226754	6.94	**0**	**212169**	**212061**	**0.05**
	140 × 5	245941	245916	5.95	**0**	245910	245900	5.94	**0**	**232355**	**232129**	**0.1**
	140 × 8	261412	261390	5.19	**0**	261386	261381	5.18	**0**	**249056**	**248512**	**0.22**
	160 × 3	306229	306229	7.77	**0**	306218	306205	7.77	**0**	**284444**	**284143**	**0.11**
	160 × 5	331202	331198	6.56	**0**	331194	331194	6.56	**0**	**311044**	**310818**	**0.07**
	160 × 8	349401	349398	5.53	**0**	349436	349426	5.54	**0**	**331606**	**331088**	**0.16**
	180 × 3	397757	397752	7.9	**0**	397739	397739	7.9	**0**	**369041**	**368620**	**0.11**
	180 × 5	210176	210062	6.53	**0**	209978	209920	6.43	**0**	**199304**	**197294**	**1.02**
	180 × 8	223213	223168	5.17	**0**	223127	223090	5.13	**0**	**214759**	**212238**	**1.19**
F = 60	200 × 3	491731	491730	7.43	**0**	491732	491722	7.43	**0**	**458630**	**457711**	**0.2**
	200 × 5	520271	520267	6.8	**0**	520297	520283	6.81	**0**	**488229**	**487129**	**0.23**
	200 × 8	549925	549924	6.18	**0**	549933	549923	6.19	**0**	**519294**	**517899**	**0.27**
	220 × 3	603376	603376	8.22	**0**	603381	603381	8.22	**0**	**559137**	**557532**	**0.29**
	220 × 5	637692	637692	7.2	**0**	637702	637685	7.2	**0**	**597715**	**594851**	**0.48**
	220 × 8	665930	665928	5.72	**0**	665945	665934	5.72	**0**	**632487**	**629913**	**0.41**
	240 × 3	354839	354749	5.21	**0**	354699	354544	5.16	**0**	**341381**	**337283**	**1.22**
	240 × 5	375595	375432	5.99	**0**	375404	375319	5.93	**0**	**358338**	**354375**	**1.12**
	240 × 8	397229	397135	3.98	**0**	397124	397036	3.95	**0**	**385747**	**382032**	**0.97**
	260 × 3	835001	834967	8.36	**0**	835001	834994	8.36	**0**	**775490**	**770569**	**0.64**
	260 × 5	893764	893742	7.15	**0**	893751	893742	7.15	**0**	**839078**	**834110**	**0.6**
	260 × 8	934783	934783	6.17	**0**	934774	934774	6.17	**0**	**886087**	**880486**	**0.64**
	280 × 3	486222	486167	5.16	**0**	486077	485982	5.12	**0**	**470092**	**462381**	**1.67**
	280 × 5	1036647	1036621	7.87	**0**	1036665	1036643	7.87	**0**	**967810**	**961035**	**0.7**
	280 × 8	1073346	1073303	5.99	**0**	1073334	1073323	5.99	**0**	**1018515**	**1012648**	**0.58**
	300 × 3	555735	555623	4.17	**0**	555569	555457	4.13	**0**	**539994**	**533509**	**1.22**
	300 × 5	1181457	1181431	7.92	**0**	1181487	1181460	7.92	**0**	**1102466**	**1094752**	**0.7**
	300 × 8	610159	610027	3.18	**0**	610063	609842	3.16	**0**	**596162**	**591361**	**0.81**

To further investigate the performance of the proposed neighborhood probabilistic selection strategies with blocking-based job, we carried out experiments without embedding the strategy in IGA and IGDLM under different termination conditions. As shown in Tables 15, 16, 17, and 18, 19, 20, When IGA and IGDLM do not use this strategy, their performance will decline sharply. NIG achieves best values in all indicators and are significantly different from the IGA and IGDLM in all instances. It can be seen that the strategy developed according to characteristics of BHFGSP is very important, i.e., blocking constraints play an important role for the design of the algorithm. This strategy effectively improves the diversity of solution, and its' advantages are as follows: On one hand, it makes up for the deficiency of NIG algorithm in exploration ability. On the other hand, it balances the overall local and global search ability of the NIG algorithm.

To observe the convergence of all algorithms in different cases, we randomly selected three different instances, i.e., $100 \times 3 \times 20$, $200 \times 5 \times 40$, and $280 \times 8 \times 60$ scales, and draw corresponding figures. For all algorithms, at the beginning of each loop, the current objective function value is recorded, and then we record the target value in each fixed time period. For example, in $100 \times 3 \times 20$ instance, we divide time equally into 30 subparts and use them as the scale of the x-axis. The y-axis represents the range of objective function values. Different algorithms are represented by curves of different colors. Similarly, to check the convergence of different algorithms with and without neighborhood probabilistic selection strategies with blocking-based job, we draw two groups of different convergence curves. Figs. 5(a), 5(b), 5(c), 5(d) represent that all algorithms use the strategy proposed in this paper. Figs. 5(d), 5(e), 5(f) represent that IGA and IGDLM algorithms do not use blocking-based job strategies. From 5(a), 5(b), 5(c), 5(d), 5(e), 5(f), all algorithms are convergent, convergence speed of NIG algorithm and ability to explore the optimal solution are superior to IGA and IGDLM. However, once IGA and IGDLM remove the neighborhood probabilistic selection strategies with blocking-based job, their search performance will decline significantly. It can also prove that this strategy has a deep impact on the performance of algorithms. That is, it reduces the completion time of whole scheduling sequence by changing the arrangement position of blocking jobs in the same family. All in all, from the results, the idea designed for blocking constraints is reasonable and effective.

5.6. Statistical experiments and analysis

In this subsection, we also perform an ANOVA analysis for obtained numerical results. Figs. 6(a), 6(b), 6(c) display Mean Plots with 0.95 HSD intervals of all comparison algorithms when the termination parameter $\rho = 100$. The analysis is mainly used to check the difference level of overall performance between different algorithms under all scales and conditions. We draw confidence intervals with different strategy algorithms in Section 4 to see the impact. Figs. 6(a), 6(b), 6(c) are divided into three subgraphs according to the family scale. From Figs. 6(a), 6(b), 6(c), it can be seen from that difference among the NIG_N_J, NIG_N_F and NIG is significant, and NIG shows the best performance in different family sizes. From the distribution of confidence intervals, we can also observe that the influence of neighborhood probabilistic selection strategies with blocking-based job on the NIG algorithm is much greater than that of the neighborhood probabilistic selection strategies with family. It also shows the

Table 18
Comparison results without the neighborhood probabilistic selection strategies with blocking-based job when $\rho = 200$, $F = 20$.

	N× S	IGA				IGDLM				NIG		
		MEAN	BEST	RPD	*p*-value	MEAN	BEST	RPD	*p*-value	MEAN	BEST	RPD
	80 × 3	3592	3584	7.95	**0**	3589	3581	7.87	**0**	**3337**	**3327**	**0.29**
	80 × 5	7393	7340	5.85	**0**	7349	7337	5.23	**0**	**7008**	**6984**	**0.35**
	80 × 8	11508	11451	7.3	**0**	11448	11437	6.74	**0**	**10821**	**10725**	**0.89**
	100 × 3	15256	15209	7.76	**0**	15186	15156	7.27	**0**	**14342**	**14157**	**1.3**
	100 × 5	57196	57191	6.15	**0**	57195	57189	6.15	**0**	**53909**	**53881**	**0.05**
	100 × 8	66789	66785	6.22	**0**	66788	66776	6.22	**0**	**62901**	**62877**	**0.04**
	120 × 3	31824	31772	5.41	**0**	31781	31746	5.27	**0**	**30503**	**30190**	**1.04**
	120 × 5	109287	109282	7.13	**0**	109288	109278	7.13	**0**	**102086**	**102014**	**0.07**
	120 × 8	61156	61145	6.2	**0**	61066	61057	6.04	**0**	**57859**	**57588**	**0.47**
	140 × 3	77463	77390	8.02	**0**	77268	77259	7.75	**0**	**72546**	**71711**	**1.16**
	140 × 5	84851	84750	6.51	**0**	84755	84723	6.39	**0**	**80113**	**79664**	**0.56**
	140 × 8	93899	93837	5.72	**0**	93699	93671	5.49	**0**	**89368**	**88819**	**0.62**
	160 × 3	222158	222155	7.9	**0**	222137	222131	7.89	**0**	**206217**	**205885**	**0.16**
	160 × 5	121676	121581	7.74	**0**	121539	121532	7.61	**0**	**114197**	**112940**	**1.11**
	160 × 8	265944	265939	6.52	**0**	265932	265912	6.52	**0**	**250171**	**249656**	**0.21**
	180 × 3	300412	300402	8.31	**0**	300403	300397	8.31	**0**	**278035**	**277360**	**0.24**
	180 × 5	162857	162783	6.98	**0**	162532	162513	6.77	**0**	**154588**	**152233**	**1.55**
	180 × 8	346961	346925	6.53	**0**	346919	346908	6.51	**0**	**326647**	**325700**	**0.29**
F = 20	200 × 3	388561	388554	8.98	**0**	388572	388569	8.98	**0**	**357437**	**356555**	**0.25**
	200 × 5	418203	418194	8.33	**0**	418205	418195	8.33	**0**	**387998**	**386063**	**0.5**
	200 × 8	446972	446964	6.84	**0**	446982	446973	6.85	**0**	**420420**	**418337**	**0.5**
	220 × 3	165527	165044	4.94	**0**	165181	164994	4.73	**0**	**160589**	**157728**	**1.81**
	220 × 5	256674	256637	5.52	**0**	256398	256371	5.4	**0**	**246735**	**243252**	**1.43**
	220 × 8	545782	545769	6.58	**0**	545781	545762	6.58	**0**	**514572**	**512090**	**0.48**
	240 × 3	587246	587242	9.15	**0**	587250	587245	9.15	**0**	**539856**	**538005**	**0.34**
	240 × 5	311994	311538	5.11	**0**	311601	311525	4.98	**0**	**299257**	**296828**	**0.82**
	240 × 8	660122	660115	7.41	**0**	660131	660119	7.42	**0**	**617827**	**614558**	**0.53**
	260 × 3	704788	704781	8.75	**0**	704795	704788	8.75	**0**	**650389**	**648077**	**0.36**
	260 × 5	751073	751060	8.56	**0**	751065	751059	8.56	**0**	**695623**	**691859**	**0.54**
	260 × 8	786885	786882	7.08	**0**	786838	786832	7.07	**0**	**738867**	**734864**	**0.54**
	280 × 3	835010	835002	9.02	**0**	835001	834997	9.02	**0**	**768898**	**765910**	**0.39**
	280 × 5	872676	872675	8.46	**0**	872675	872659	8.46	**0**	**808832**	**804594**	**0.53**
	280 × 8	919283	919276	7.07	**0**	919278	919276	7.06	**0**	**863212**	**858619**	**0.53**
	300 × 3	484583	484013	5.94	**0**	484150	484006	5.84	**0**	**463060**	**457415**	**1.23**
	300 × 5	1026774	1026768	8.96	**0**	1026774	1026763	8.96	**0**	**947647**	**942325**	**0.56**
	300 × 8	1066735	1066729	7.38	**0**	1066726	1066693	7.38	**0**	**999149**	**993401**	**0.58**

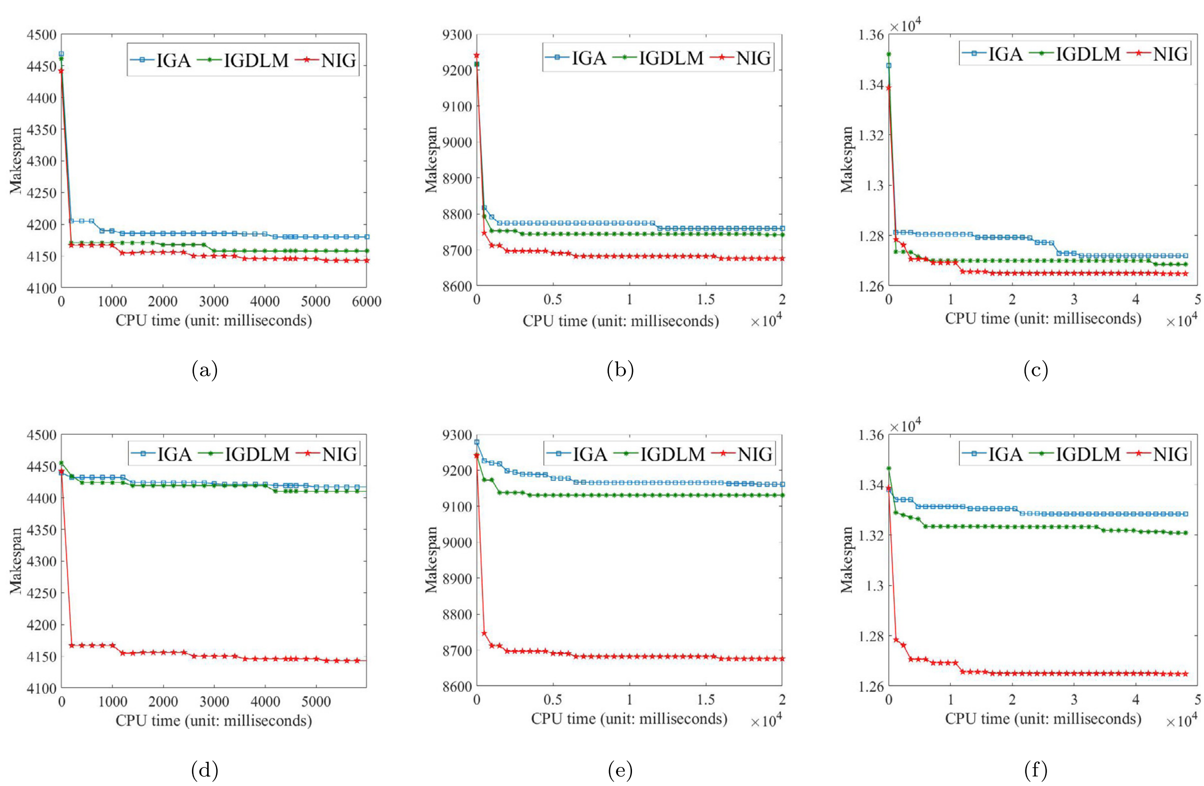
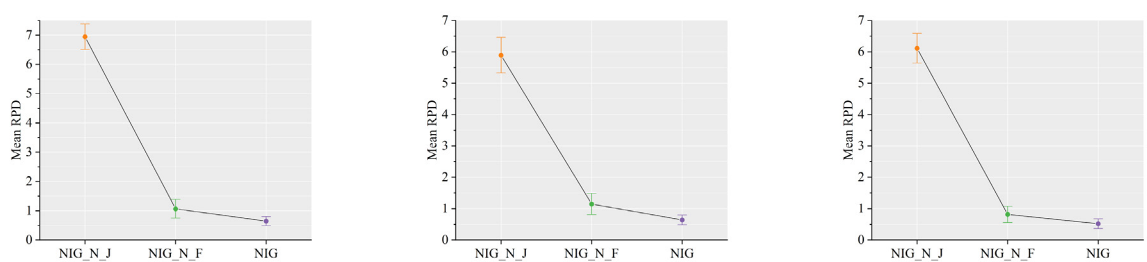


Fig. 5. The convergence curve of IGA, IGDLM and NIG algorithms: (a) 100 × 3 × 20. (b) 200 × 5 × 40. (c) 280 × 8 × 60. (d) N_100 × 3 × 20. (e) N_200 × 5 × 40. (f) N_280 × 8 × 60.

Table 19
Comparison results without the neighborhood probabilistic selection strategies with blocking-based job when $\rho = 200$, $F = 40$.

	N× S	IGA				IGDLM				NIG		
		MEAN	BEST	RPD	*p*-value	MEAN	BEST	RPD	*p*-value	MEAN	BEST	RPD
	80 × 3	7083	7071	9.92	**0**	7058	7037	9.53	**0**	**6664**	**6444**	**3.42**
	80 × 5	17976	17900	6.99	**0**	17896	17872	6.52	**0**	**16895**	**16801**	**0.56**
	80 × 8	43225	43209	4.8	**0**	43216	43205	4.78	**0**	**41249**	**41243**	**0.02**
	100 × 3	24631	24580	9.18	**0**	24548	24514	8.82	**0**	**22975**	**22559**	**1.84**
	100 × 5	84846	84834	6.53	**0**	84824	84810	6.51	**0**	**79701**	**79643**	**0.07**
	100 × 8	94970	94960	5.34	**0**	94983	94971	5.36	**0**	**90243**	**90154**	**0.1**
	120 × 3	63392	63334	7.02	**0**	63314	63286	6.89	**0**	**59593**	**59232**	**0.61**
	120 × 5	139541	139541	6.57	**0**	139539	139534	6.57	**0**	**131170**	**130933**	**0.18**
	120 × 8	154539	154527	5.34	**0**	154505	154497	5.32	**0**	**146891**	**146702**	**0.13**
	140 × 3	94761	94736	6.78	**0**	94697	94650	6.71	**0**	**89144**	**88741**	**0.45**
	140 × 5	211142	211139	6.9	**0**	211140	211121	6.9	**0**	**197724**	**197515**	**0.11**
	140 × 8	224844	224844	5.8	**0**	224825	224812	5.79	**0**	**212736**	**212527**	**0.1**
	160 × 3	266007	266001	7.54	**0**	265995	265983	7.54	**0**	**247722**	**247355**	**0.15**
	160 × 5	287155	287143	7.19	**0**	287141	287135	7.18	**0**	**268489**	**267897**	**0.22**
	160 × 8	153413	153335	5.76	**0**	153300	153256	5.68	**0**	**145649**	**145060**	**0.41**
	180 × 3	347696	347694	8.23	**0**	347688	347676	8.23	**0**	**321841**	**321251**	**0.18**
	180 × 5	374207	374207	7.05	**0**	374197	374190	7.05	**0**	**350422**	**349565**	**0.25**
	180 × 8	394852	394823	5.77	**0**	394833	394808	5.76	**0**	**374171**	**373317**	**0.23**
F = 40	200 × 3	220420	220222	7.15	**0**	220271	220219	7.07	**0**	**208952**	**205717**	**1.57**
	200 × 5	156644	156309	2.58	**0**	156372	156225	2.4	**0**	**153702**	**152705**	**0.65**
	200 × 8	495150	495150	6.54	**0**	495160	495149	6.55	**0**	**466000**	**464735**	**0.27**
	220 × 3	135716	135569	2.84	**0**	135426	135288	2.62	**0**	**133154**	**131965**	**0.9**
	220 × 5	193008	192736	2.59	**0**	192680	192442	2.41	**0**	**189313**	**188137**	**0.62**
	220 × 8	608285	608281	6.31	**0**	608299	608286	6.31	**0**	**574628**	**572180**	**0.43**
	240 × 3	328374	328267	6.92	**0**	328154	328086	6.85	**0**	**311465**	**307120**	**1.41**
	240 × 5	693721	693710	7.92	**0**	693701	693673	7.92	**0**	**645581**	**642809**	**0.43**
	240 × 8	722682	722674	6.27	**0**	722664	722632	6.27	**0**	**683360**	**680025**	**0.49**
	260 × 3	261605	261234	4.07	**0**	260955	260412	3.81	**0**	**255729**	**251379**	**1.73**
	260 × 5	822037	822022	7.51	**0**	822022	822004	7.51	**0**	**769212**	**764635**	**0.6**
	260 × 8	856648	856641	6.82	**0**	856633	856613	6.82	**0**	**806421**	**801955**	**0.56**
	280 × 3	457279	457124	5.14	**0**	457108	457058	5.1	**0**	**441923**	**434927**	**1.61**
	280 × 5	959782	959755	8.46	**0**	959729	959717	8.45	**0**	**890846**	**884935**	**0.67**
	280 × 8	993229	993228	6.38	**0**	993235	993215	6.38	**0**	**939172**	**933673**	**0.59**
	300 × 3	516011	515732	4.47	**0**	515735	515701	4.42	**0**	**499559**	**493921**	**1.14**
	300 × 5	274235	274044	1.8	**0**	273689	273494	1.6	**0**	**271437**	**269388**	**0.76**
	300 × 8	1145480	1145473	7.16	**0**	1145435	1145420	7.16	**0**	**1075536**	**1068901**	**0.62**



(a) (b) (c)

Fig. 6. Means plots for the NIG_N_J, NIG_N_F, and NIG algorithms with different family scales: (a) $F = 20$. (b) $F = 40$. (c) $F = 60$.

importance of strategy designed for blocking constraints and internal arrangement order of jobs.

Then, we draw interactions plots with the 0.95 HSD confidence intervals. Figs. 7(a), 7(b), 7(c) show the confidence intervals of all comparison algorithms in different family scales: i.e., $F = 20$, 40, and 60. Among them, IGA and IGDLM used neighborhood probabilistic selection strategies with blocking-based job to explore better solutions. For Figs. 7(d), 7(e), 7(f), IGA and IGDLM do not use the proposed strategies to search the scheduling scheme. According to Fig. 6, we can observe that whatever the scale of families is, NIG is also the best algorithm, then followed by the IGDLM and IGA. When the IGA and IGDLM remove the neighborhood probabilistic selection strategies with blocking-based job, their performance will become much worse, and difference between the NIG is significant. In addition, Figs. 8(a), 8(b), 8(c), 8(d), 8(e), 8(f), illustrate the violin plots with 95% confidence interval of three random selected instances obtained by all comparison algorithms, i.e., 80 × 3 × 20, 140 × 5 × 40, 120 × 8 × 60 scales. Similarly, we divide the experiment into two groups. One group is that the comparison algorithms carry the proposed strategy with blocking-based job (Figs. 8(a), 8(b), 8(c)), and another group is without this strategy (Figs. 8(d), 8(e), 8(f)). The results of all instances are repeated for 30 times. It illustrates that RPD values obtained by NIG algorithm are much smaller than that of IGA and IGDLM algorithms. It also indicates that NIG algorithm has potential to get a better scheduling sequence.

Through above figures, we can find that, without the neighborhood probabilistic selection strategies with blocking-based job, IGA and IGDLM lead to a great significant difference from NIG algorithm. It further indicates the great impact of the blocking

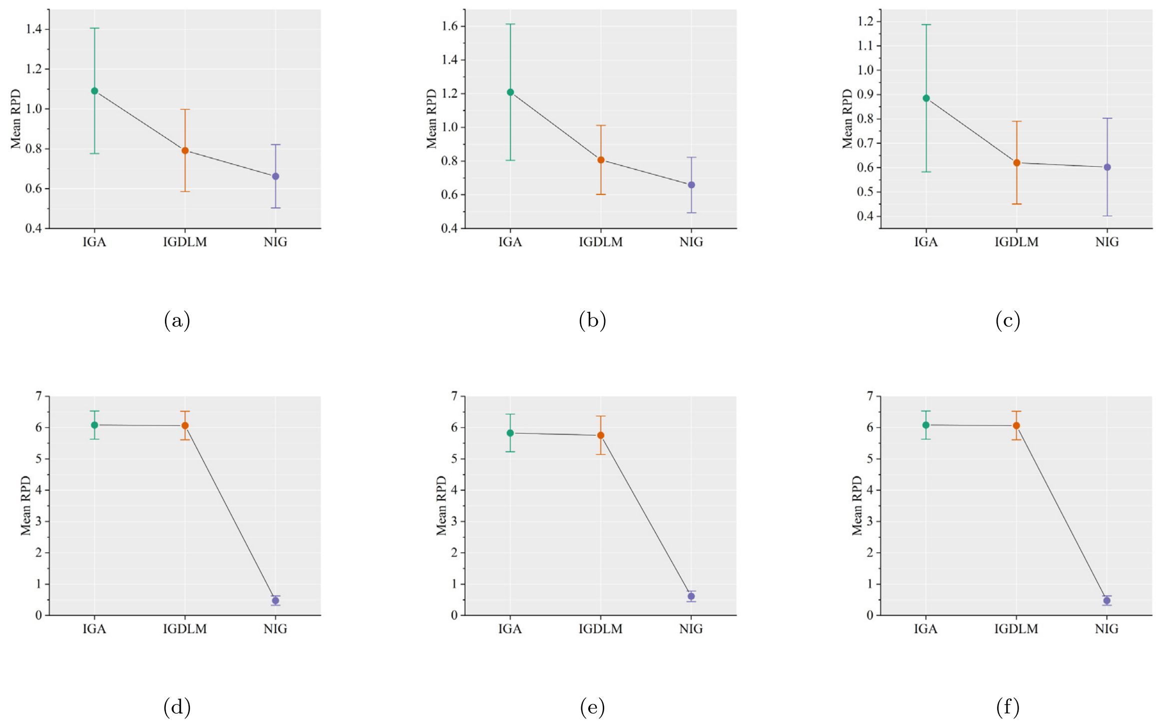


Fig. 7. Interactions plots for the IGA, IGDLM, and NIG algorithms with different family scales: (a) Mean $F = 20$. (b) Mean $F = 40$. (c) Mean $F = 60$. (d) N_Mean $F = 20$. (e) N_Mean $F = 40$. (f) N_Mean $F = 60$.

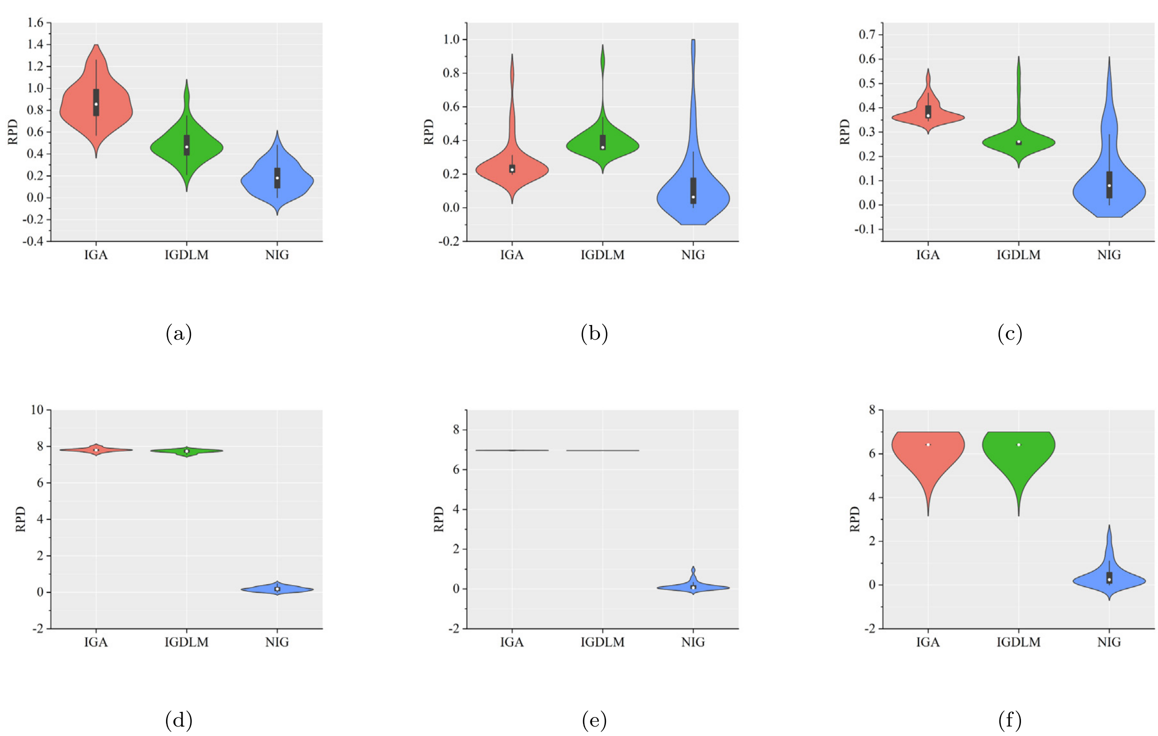


Fig. 8. Violin plots for the IGA, IGDLM, and NIG algorithms in different scale instances: (a) $80 \times 3 \times 20$. (b) $140 \times 5 \times 40$. (c) $120 \times 8 \times 60$. (d) $80 \times 3 \times 20$N. (e)$140 \times 5 \times 40$N. (f) $120 \times 8 \times 60$N.

constraints on completion time. Besides, when IGA and IGDLM utilized the neighborhood probabilistic selection strategies with blocking-based job to solve the BHFGSP, their performance is also inferior to NIG algorithm. The reason may be that the adjustment ability of IGA and IGDLM to SDST is not as good as neighborhood probabilistic selection strategies with family. NIG is still the best

Table 20
Comparison results without the neighborhood probabilistic selection strategies with blocking-based job when $\rho = 200$, $F = 60$.

	N× S	IGA				IGDLM				NIG		
		MEAN	BEST	RPD	p-value	MEAN	BEST	RPD	p-value	MEAN	BEST	RPD
	80 × 3	48191	48180	4.75	**0**	48194	48181	4.76	**0**	**46018**	**46005**	**0.03**
	80 × 5	56442	56419	5.25	**0**	56411	56405	5.19	**0**	**53634**	**53627**	**0.01**
	80 × 8	64519	64516	4.31	**0**	64519	64516	4.32	**0**	**61903**	**61850**	**0.09**
	100 × 3	100261	100256	6.2	**0**	100258	100256	6.2	**0**	**94445**	**94407**	**0.04**
	100 × 5	110962	110949	5.84	**0**	110934	110931	5.81	**0**	**104941**	**104842**	**0.09**
	100 × 8	121396	121392	4.6	**0**	121396	121367	4.61	**0**	**116152**	**116052**	**0.09**
	120 × 3	78428	78344	7.43	**0**	78369	78309	7.35	**0**	**73333**	**73001**	**0.46**
	120 × 5	86163	86118	6.88	**0**	86086	86031	6.78	**0**	**80841**	**80618**	**0.28**
	120 × 8	185095	185091	5.57	**0**	185092	185092	5.57	**0**	**175428**	**175327**	**0.06**
	140 × 3	226741	226712	7.17	**0**	226748	226745	7.18	**0**	**211819**	**211566**	**0.12**
	140 × 5	245931	245895	6.04	**0**	245934	245928	6.04	**0**	**232212**	**231919**	**0.13**
	140 × 8	261427	261372	5.31	**0**	261358	261358	5.28	**0**	**248676**	**248256**	**0.17**
	160 × 3	306226	306212	7.73	**0**	306207	306197	7.73	**0**	**284566**	**284245**	**0.11**
	160 × 5	331207	331207	6.78	**0**	331205	331205	6.78	**0**	**310475**	**310173**	**0.1**
	160 × 8	349414	349406	5.81	**0**	349413	349404	5.81	**0**	**330897**	**330242**	**0.2**
	180 × 3	397747	397747	8.11	**0**	397744	397744	8.11	**0**	**368569**	**367922**	**0.18**
	180 × 5	210163	210050	6.06	**0**	209971	209928	5.96	**0**	**200297**	**198152**	**1.08**
	180 × 8	223221	223113	4.78	**0**	223116	223029	4.73	**0**	**214925**	**213034**	**0.89**
F = 60	200 × 3	491726	491713	7.62	**0**	491715	491715	7.62	**0**	**458087**	**456899**	**0.26**
	200 × 5	520296	520257	6.95	**0**	520281	520274	6.95	**0**	**487665**	**486484**	**0.24**
	200 × 8	549937	549910	6.17	**0**	549920	549912	6.17	**0**	**519101**	**517976**	**0.22**
	220 × 3	603383	603375	8.37	**0**	603388	603388	8.37	**0**	**558345**	**556784**	**0.28**
	220 × 5	637684	637640	7.14	**0**	637700	637667	7.15	**0**	**597604**	**595174**	**0.41**
	220 × 8	665941	665903	5.9	**0**	665961	665937	5.91	**0**	**631150**	**628825**	**0.37**
	240 × 3	354785	354612	5.23	**0**	354620	354524	5.18	**0**	**340057**	**337153**	**0.86**
	240 × 5	375538	375449	5.37	**0**	375378	375314	5.33	**0**	**360201**	**356387**	**1.07**
	240 × 8	397243	397089	3.96	**0**	397089	397010	3.92	**0**	**384879**	**382113**	**0.72**
	260 × 3	834985	834967	8.56	**0**	834972	834971	8.56	**0**	**772343**	**769113**	**0.42**
	260 × 5	893754	893728	7.43	**0**	893773	893753	7.43	**0**	**835419**	**831977**	**0.41**
	260 × 8	934777	934767	6.41	**0**	934788	934787	6.41	**0**	**882191**	**878447**	**0.43**
	280 × 3	486177	486054	5.39	**0**	486054	485979	5.36	**0**	**468204**	**461322**	**1.49**
	280 × 5	1036656	1036634	8.13	**0**	1036667	1036645	8.14	**0**	**963635**	**958677**	**0.52**
	280 × 8	1073329	1073328	6.28	**0**	1073323	1073321	6.28	**0**	**1014122**	**1009934**	**0.41**
	300 × 3	555659	555545	4.75	**0**	555535	555419	4.73	**0**	**536377**	**530446**	**1.12**
	300 × 5	1181470	1181436	7.83	**0**	1181478	1181457	7.83	**0**	**1103767**	**1095645**	**0.74**
	300 × 8	610185	610061	3.17	**0**	609994	609790	3.14	**0**	**596674**	**591426**	**0.89**

algorithm in the current comparison algorithms. It is attributed to the balance between its global and local search capabilities, and its own strategies for problem characteristics.

6. Conclusion

This is the first reported research work of designing a novel IG algorithm for solving BHFGSP with minimizing the makespan. According to comprehensive and adequate numerical and statistical results, the proposed NIG algorithm is proved to be very effective to solve BHFGSP. In summary, we provide four contributions: (1) we proposed a novel mathematical model of BHFGSP. Then, the objective is to minimize the makespan. (2) New decoding procedure is presented to represent and calculate the optimization objective of the job sequence. (3) Neighborhood probabilistic selection strategies with family are designed to adjust the positions of families, and further reducing SDST between different families. (4) Neighborhood probabilistic selection strategies with blocking-based job are proposed to change blocking conditions of job sequence, further improving the production efficiency of enterprise.

In the future, we will continue to design corresponding algorithm strategies according to characteristics of the problem. Similarly, other optimization objectives, such as total flow time, energy consumption, and tardiness, are also worth studying. We will research more real-world applications that are similar to this problem. We will consider more constraints, for example, distributed environment, dynamic scheduling environment, machine breakdown, job deterioration and assembly, etc. In addition, knowledge, collaboration and machine learning can also be combined with IG algorithm to solve different flow shop scheduling problems. These above-mentioned topics are interesting and worth investigating.

CRediT authorship contribution statement

Haoxiang Qin: Conception and design of study, Acquisition of data, Analysis and/or interpretation of data, Writing – original draft, Writing – review & editing, Technical, Writing assistance, Revised the manuscript. **Yuyan Han:** Conception and design of study, Acquisition of data, Analysis and/or interpretation of data, Writing – original draft, Writing – review & editing, Technical, Writing assistance, Revised the manuscript. **Yuting Wang:** Technical, Editing, Writing assistance, Interpretation of data, Revised the manuscript. **Yiping Liu:** Technical, Editing, Writing assistance, Interpretation of data, Revised the manuscript. **Junqing Li:** Technical, Editing, Writing assistance, Interpretation of data, Revised the manuscript. **Quanke Pan:** Technical, Editing, Writing assistance, Interpretation of data, Revised the manuscript.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgments

This work was supported by the National Natural Science Foundation of China under grant numbers 61803192, 61973203, 61966012, 62106073, and 71533001. We are grateful for Guangyue Young Scholar Innovation Team of Liaocheng University under grant number LCUGYTD2022-03, the Natural Science Foundation of Hunan Province of China under grant number 2021JJ40116, and the Natural Science Foundation of Shandong Province under grant numbers ZR2021QE195 and ZR2021QF036.

References

[1] W. Shao, D. Pi, Z. Shao, Optimization of makespan for the distributed no-wait flow shop scheduling problem with iterated greedy algorithms, Knowl.-Based Syst. 137 (dec.1) (2017) 163–181.

[2] M. Qin, R. Wang, Z. Shi, L. Liu, L. Shi, A genetic programming-based scheduling approach for hybrid flow shop with a batch processor and waiting time constraint, IEEE Trans. Autom. Sci. Eng. PP (99) (2019) 1–12.

[3] T. Meng, Q.K. Pan, L. Wang, A distributed permutation flowshop scheduling problem with the customer order constraint, Knowl.-Based Syst. 184 (Nov.15) (2019) 104894.1–104894.17.

[4] D. Lei, L. Gao, Y. Zheng, A novel teaching-learning-based optimization algorithm for energy-efficient scheduling in hybrid flow shop, IEEE Trans. Eng. Manage. (2017) 1–11.

[5] J.Q. Li, Q.K. Pan, K. Mao, A hybrid fruit fly optimization algorithm for the realistic hybrid flowshop rescheduling problem in steelmaking systems, IEEE Trans. Autom. Sci. Eng. (2016) 932–949.

[6] M.K. Marichelvam, T. Prabaharan, X.S. Yang, A discrete firefly algorithm for the multi-objective hybrid flowshop scheduling problems, IEEE Press (2014).

[7] Y. Fu, M. Zhou, X. Guo, L. Qi, Scheduling dual-objective stochastic hybrid flow shop with deteriorating jobs via bi-population evolutionary algorithm, IEEE Trans. Syst., Man, Cybern.: Syst. 50 (12) (2020) 5037–5048, http://dx.doi.org/10.1109/TSMC.2019.2907575.

[8] J.J. Wang, L. Wang, A bi-population cooperative memetic algorithm for distributed hybrid flow-shop scheduling, IEEE Trans. Emerg. Top. Comput. Intell. PP (99) (2020) 1–15.

[9] Z. Shao, D. Pi, W. Shao, A novel multi-objective discrete water wave optimization for solving multi-objective blocking flow-shop scheduling problem, Knowl.-Based Syst. 165 (FEB.1) (2019) 110–131.

[10] S. Aqil, K. Allali, Two efficient nature inspired meta-heuristics solving blocking hybrid flow shop manufacturing problem, Eng. Appl. Artif. Intell. 100 (104196) (2021).

[11] X. Han, Y.Y. Han, B. Zhang, H.X. Qin, J.Q. Li, Y.P. Liu, D.W. Gong, An effective iterative greedy algorithm for distributed blocking flowshop scheduling problem with balanced energy costs criterion, Appl. Soft Comput. 129 (109502) (2022).

[12] V. Riahi, M. Newton, K. Su, A. Sattar, Constraint guided accelerated search for mixed blocking permutation flowshop scheduling, Comput. Oper. Res. 102 (FEB.) (2018) 102–120.

[13] J. Grabowski, J. Pempera, Sequencing of jobs in some production system, European J. Oper. Res. 125 (3) (2000) 535–550.

[14] D.P. Ronconi, Lower bounding schemes for flowshops with blocking in-process, J. Oper. Res. Soc. 52 (2001) 1289–1297.

[15] D.P. Ronconi, A note on constructive heuristics for the flowshop problem with blocking, Int. J. Prod. Econ. 87 (1) (2004) 39–48.

[16] G. Hua, L. Tang, C.W. Duin, A two-stage flow shop scheduling problem on a batching machine and a discrete machine with blocking and shared setup times, Comput. Oper. Res. 37 (5) (2010) 960–969.

[17] L. Hao, G.Q. Huang, Y. Zhang, Q. Dai, C. Xin, Two-stage hybrid batching flowshop scheduling with blocking and machine availability constraints using genetic algorithm, Robot. Comput.-Integr. Manuf. 25 (6) (2009) 962–971.

[18] X. He, Q.-k. Pan, L. Gao, L. Wang, P.N. Suganthan, A greedy cooperative co-evolution ary algorithm with problem-specific knowledge for multi-objective flowshop group scheduling problems, IEEE Trans. Evol. Comput. (2021) 1, http://dx.doi.org/10.1109/TEVC.2021.3115795.

[19] A.D. Wilson, R.E. King, T.J. Hodgson, Scheduling non-similar groups on a flow line: multiple group setups, Robot. Comput.-Integr. Manuf. 20 (6) (2004) 505–51513.

[20] J.E. Schaller, J. Gupta, A.J. Vakharia, Scheduling a flowline manufacturing cell with sequence dependent family setup times, European J. Oper. Res. 125 (2) (2000) 324–339.

[21] N. Salmasi, R. Logendran, M.R. Skandari, Total flow time minimization in a flowshop sequence-dependent group scheduling problem, Comput. Oper. Res. 37 (1) (2010) 199–212.

[22] Q.K. Pan, L. Gao, L. Wang, An effective cooperative co-evolutionary algorithm for distributed flowshop group scheduling problems, IEEE Trans. Cybern. PP (99) (2020) 1–14.

[23] W. Shao, Z. Shao, D. Pi, Modeling and multi-neighborhood iterated greedy algorithm for distributed hybrid flow shop scheduling problem, Knowl.-Based Syst. (2020) 105527.

[24] B. Zhang, Q.K. Pan, L.L. Meng, C. Lu, J.H. Mou, J.Q. Li, An automatic multi-objective evolutionary algorithm for the hybrid flowshop scheduling problem with consistent sublots, Knowl.-Based Syst. 238 (2022) 107819.

[25] I. Ribas, R. Leisten, J.M. Framinan, Review and classification of hybrid flow shop scheduling problems from a production system and a solutions procedure perspective, Comput. Oper. Res. 37 (8) (2010) 1439–1454.

[26] R. Ruiz, T. Stützle, A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem, European J. Oper. Res. 177 (3) (2007) 2033–2049.

[27] Q.K. Pan, L. Wang, K. Mao, J.H. Zhao, M. Zhang, An effective artificial bee colony algorithm for a real-world hybrid flowshop problem in steelmaking process, IEEE Trans. Autom. Sci. Eng. 10 (2) (2013) 307–322.

[28] J. Zheng, L. Wang, J.J. Wang, A cooperative coevolution algorithm for multi-objective fuzzy distributed hybrid flow shop, Knowl.-Based Syst. (2020) 105536.

[29] L. Tang, X. Wang, An improved particle swarm optimization algorithm for the hybrid flowshop scheduling to minimize total weighted completion time in process industry, IEEE Trans. Control Syst. Technol. 18 (6) (2010) 1303–1314.

[30] Q.K. Pan, L. Wang, J.Q. Li, J.H. Duan, A novel discrete artificial bee colony algorithm for the hybrid flowshop scheduling problem with makespan minimisation, Omega 45 (jun.) (2014) 42–56.

[31] J.Q. Li, Q.K. Pan, P.Y. Duan, An improved artificial bee colony algorithm for solving hybrid flexible flowshop with dynamic operation skipping, IEEE Trans. Cybern. 46 (6) (2016) 1311–1324.

[32] B. Zhang, Q.-K. Pan, L. Gao, L.-L. Meng, X.-Y. Li, K.-K. Peng, A three-stage multiobjective approach based on decomposition for an energy-efficient hybrid flow shop scheduling problem, IEEE Trans. Syst., Man, Cybern.: Syst. 50 (12) (2020) 4984–4999, http://dx.doi.org/10.1109/TSMC.2019.2916088.

[33] W. Trabelsi, C. Sauvey, N. Sauer, Mathematical model and lower bound for hybrid flowshop problem with mixed blocking constraints, IFAC Proc. Vol. 45 (6) (2012) 1475–1480.

[34] P. Nakkaew, N. Kantanantha, W. Wongthatsanekorn, A comparison of genetic algorithm and artificial bee colony approaches in solving blocking hybrid flowshop scheduling problem with sequence dependent setup/changeover times, KKU Eng. J. 43 (2) (2016).

[35] A. Elmi, S. Topaloglu, A scheduling problem in blocking hybrid flow shop robotic cells with multiple robots, Comput. Oper. Res. 40 (10) (2013) 2543–2555.

[36] A. Missaoui, Y. Boujelbene, An effective iterated greedy algorithm for blocking hybrid flow shop problem with due date window, RAIRO - Oper. Res. 55 (3) (2021) 1603–1616.

[37] H.-X. Qin, Y.-Y. Han, B. Zhang, L.-L. Meng, Y.-P. Liu, Q.-K. Pan, D.-W. Gong, An improved iterated greedy algorithm for the energy-efficient blocking hybrid flow shop scheduling problem, Swarm Evol. Comput. 69 (2022) 100992, http://dx.doi.org/10.1016/j.swevo.2021.100992.

[38] S.W. Lin, K.C. Ying, Makespan optimization in a no-wait flowline manufacturing cell with sequence-dependent family setup times, Comput. Ind. Eng. 128 (FEB.) (2019) 1–7.

[39] J.S. Neufeld, J.N.D. Gupta, U. Buscher, Minimising makespan in flowshop group scheduling with sequence-dependent family set-up times using inserted idle times, Int. J. Prod. Res. 53 (6) (2015) 1791–1806.

[40] J.S. Neufeld, J. Gupta, U. Buscher, A comprehensive review of flowshop group scheduling literature, Comput. Oper. Res. 70 (Jun.) (2016) 56–74.

[41] H. Feng, L. Xi, L. Xiao, T. Xia, E. Pan, Imperfect preventive maintenance optimization for flexible flowshop manufacturing cells considering sequence-dependent group scheduling, Reliab. Eng. Syst. Saf. 176 (aug.) (2018) 218–229.

[42] A. Costa, F.A. Cappadonna, S. Fichera, A hybrid genetic algorithm for minimizing makespan in a flow-shop sequence-dependent group scheduling problem, J. Intell. Manuf. (2017).

[43] A. Costa, F.V. Cappadonna, S. Fichera, Minimizing makespan in a flow shop sequence dependent group scheduling problem with blocking constraint, Eng. Appl. Artif. Intell. 89 (Mar.) (2020) 103413.1–103413.15.

[44] C.D. Liou, Y.C. Hsieh, A hybrid algorithm for the multi-stage flow shop group scheduling with sequence-dependent setup and transportation times, Int. J. Prod. Econ. 170 (DEC.PT.A) (2015) 258–267.

[45] B. Naderi, N. Salmasi, Permutation flowshops in group scheduling with sequence-dependent setup times, Eur. J. Ind. Eng. 6 (2) (2012) 177.

[46] T. Keshavarz, N. Salmasi, M. Varmazyar, Flowshop sequence-dependent group scheduling with minimisation of weighted earliness and tardiness, Eur. J. Ind. Eng. 13 (1) (2019) 54.

[47] J.J. Wang, L. Wang, A cooperative memetic algorithm with learning-based agent for energy-aware distributed hybrid flow-shop scheduling, IEEE Trans. Evol. Comput. (2021) http://dx.doi.org/10.1109/TEVC.2021.3106168.
[48] V. Fernandez-Viagas, P. Perez-Gonzalez, J.M. Framinan, Efficiency of the solution representations for the hybrid flow shop scheduling problem with makespan objective, Comput. Oper. Res. 109 (SEP.) (2019) 77–88.
[49] J.I. Ham, A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem, Omega (1983).
[50] A. Jph, B. Qkpa, G.C. Liang, An effective iterated greedy method for the distributed permutation flowshop scheduling problem with sequence-dependent setup times, Swarm Evol. Comput. 59 (2020).
[51] Y.Y. Huang, Q.K. Pan, J.P. Huang, P.N. Suganthan, L. Gao, An improved iterated greedy algorithm for the distributed assembly permutation flowshop scheduling problem, Comput. Ind. Eng. 152 (3) (2020) 107021.
[52] B. Zhang, Q.K. Pan, L. Gao, X.L. Zhang, H.Y. Sang, J.Q. Li, An effective modified migrating birds optimization for hybrid flowshop scheduling problem with lot streaming, Appl. Soft Comput. 52 (2017) 14–27.
[53] H.X. Qin, Y.Y. Han, Q.D. Chen, J.Q. Li, H.Y. Sang, A double level mutation iterated greedy algorithm for blocking hybrid flow shop scheduling, Control Decis. (2021) 1–10, http://dx.doi.org/10.13195/j.kzyjc.2021.0607.
[54] J.-P. Huang, Q.-K. Pan, L. Gao, An effective iterated greedy method for the distributed permutation flowshop scheduling problem with sequence-dependent setup times, Swarm Evol. Comput. 59 (2020) 100742.
[55] S.-Y. Wang, L. Wang, An estimation of distribution algorithm-based memetic algorithm for the distributed assembly permutation flow-shop scheduling problem, IEEE Trans. Syst., Man, Cybern.: Syst. 46 (1) (2016) 139–149, http://dx.doi.org/10.1109/TSMC.2015.2416127.
[56] Y. Li, X. Li, L. Gao, L. Meng, An improved artificial bee colony algorithm for distributed heterogeneous hybrid flowshop scheduling problem with sequence-dependent setup times, Comput. Ind. Eng. 147 (2020) 106638, http://dx.doi.org/10.1016/j.cie.2020.106638.
[57] T. Meng, Q.K. Pan, A distributed heterogeneous permutation flowshop scheduling problem with lot-streaming and carryover sequence-dependent setup time, Swarm Evol. Comput. 60 (2021) 100804.
[58] V.N. Nair, B. Abraham, J. Mackay, J.A. Nelder, G. Box, M.S. Phadke, R.N. Kacker, J. Sacks, W.J. Welch, T.J. Lorenzen, Taguchiš parameter design: A panel discussion, Technometrics 34 (2) (1992) 127–161.
[59] S. Aqil, K. Allali, Local search metaheuristic for solving hybrid flow shop problem in slabs and beams manufacturing, Expert Syst. Appl. 162 (2020) 113716.