Egyptian Informatics Journal 27 (2024) 100509

Contents lists available at ScienceDirect

Egyptian Informatics Journal

journal homepage: www.sciencedirect.com

Full length article

A single-individual based variable neighborhood search algorithm for the blocking hybrid flow shop group scheduling problem

Zhongyuan Peng [a], Haoxiang Qin [b,*]

[a] Department of Basic Courses, Maoming Polytechnic, Guangdong, 525000, China
[b] School of Software Engineering, South China University of Technology, Guangdong, 510006, China

ARTICLE INFO

Keywords:
Hybrid flow shop group scheduling problem
Blocking constraints
Energy consumption
Single-individual based algorithm
Neighborhood search strategy

ABSTRACT

The Blocking Hybrid Flow Shop Group Scheduling Problem (BHFGSP) is prevalent within the manufacturing industry, where the ordering of groups poses a significant challenge for dispatchers. Moreover, the blocking constraints associated with jobs significantly influence energy consumption, yet these constraints are often overlooked in algorithm design. To address these issues effectively, a single-individual-based variable neighborhood search strategy is introduced. For the challenge of group ordering, a group-based neighborhood search strategy is proposed. This strategy is complemented by a job-based neighborhood search strategy to tackle the issues of blocking and job sequencing. These two neighborhood search strategies are designed to enhance the performance of the algorithm significantly. Furthermore, to augment the local search abilities of the proposed algorithm, the concept of a single-individual approach from the iterated greedy algorithm is integrated. The performance of the proposed algorithm is validated through 36 instances, demonstrating its efficiency in solving BHFGSPs compared to state-of-the-art algorithms. Notably, the proposed algorithm achieves a reduction in energy consumption by an average of 58% to 63.4% compared to previous best solutions.

1. Introduction

The hybrid flow shop scheduling problem (HFSP) is prevalent in manufacturing production [1–4]. Unlike the traditional Flow Shop Scheduling Problem (FSP), there exists at least one stage in HFSP with the number of machines greater than 1 [5–7]. Moreover, in FSP, the buffers between adjacent machines are infinite, and jobs can be stored infinitely in the buffers [8,9]. However, in real production scenarios, such as concrete block production [10], chemical production [11], steelmaking [12], there is no buffer between adjacent machines to store finished jobs due to production costs, product characteristics, or technical conditions [13,14]. This causes a situation where the job is blocked on the current machine, which in turn prolongs the completion time of the jobs and results in energy wastage [15–17]. The problem is also termed as blocking hybrid flow shop scheduling problem (BHFSP). In order to solve the BHFSP, it is important to use a sequencing strategy that can effectively reduce the blocking of jobs. By using this strategy, it can help companies to improve productivity and reduce unnecessary waste of resources.

With the continuous development and application of group and manufacturing unit technologies, the group scheduling problem has received extensive attention from researchers [18]. A group consists of multiple jobs with similar production requirements (e.g., setup, tooling, product attributes) [19]. In the real world, grouping technology has been widely used in many areas such as the automotive assembly industry [18,20], paint shops [21], printed circuit boards [22], and upholstered furniture [23]. The utilization of production in groups effectively simplifies the ordering process, shortens the completion time, reduces energy consumption, and thus improves the reliability of factory processing [24]. In the group scheduling problem, the ordering issue of the groups will directly affect the setting of machines, which in turn affects the objective value [25]. To improve the productivity as much as possible and reduce the unnecessary waste of resources, it is necessary to design efficient group ordering strategies [26].

HFSP and BHFSP have been demonstrated to be NP-hard problems [27–29]. When the size of the problem becomes progressively larger, traditional mathematical methods or exact algorithms will have difficulties in solving the problem [30,31]. Therefore, to solve the aforementioned problems, heuristic or meta-heuristic algorithms have been employed in most of the studies, such as Iterated Greedy Algorithm (IGA) [32], Artificial Bee Colony Algorithm [33], and Memetic Algorithm [34]. In addition, if the grouping of jobs is considered, the difficulty of solving the problem will be further increased. So far, there is no literature on problems that consider both blocking constraints and job grouping with the objective of minimizing energy consumption.

* Corresponding author.
E-mail addresses: 2004010041@mmpt.edu.cn (Z. Peng), 987352978@qq.com (H. Qin).

https://doi.org/10.1016/j.eij.2024.100509
Received 27 May 2024; Received in revised form 30 June 2024; Accepted 20 July 2024
Available online 30 July 2024

In the enterprise, how to reduce the energy consumption caused by group sorting and job blocking is an urgent concern for the dispatchers, and the existing techniques or strategies cannot cope with the above two situations at the same time. Therefore, it is necessary to design an effective optimization algorithm to solve this problem. This paper addresses the problem regarding blocking and group ordering that exists in today's organizations. At the same time, this paper not only fills the knowledge gap in the field of research, but also helps the dispatchers save their time in scheduling assignments.

In this paper, BHFGSP encoding and decoding rules are proposed with the objective of minimizing energy consumption. Then, a single-individual based variable neighborhood search algorithm (SIVNS) is designed to solve BHFGSP. Compared with the state-of-the-art optimization algorithms, this algorithm has the advantages of simpler structure, easy to implement, and better search ability.

The contributions of this paper are given as follows.

- This paper designs a neighborhood search strategy based on grouping of jobs, which solves the sorting challenges of grouping as well as the issues of high overhead of machine setting.
- This paper designs a neighborhood search strategy based on job blocking and job sorting, which reduces the impact of blocking constraints on energy consumption.
- The results demonstrate that the proposed algorithm consumes 58%-63.4% less average energy than the current state-of-the-art algorithms.

The rest of this paper is shown as follows. Section 2 reviews the literature related to the research problem of this paper. In Section 3, the BHFGSP with minimizing the objective of energy consumption is described. In Section 4, the proposed SIVNS is elaborated, which mainly consists of initialization, group-based neighborhood search strategy and job-based neighborhood search strategy. The simulation and analysis are carried out in Section 5. Section 6 concludes the paper and provides an outlook for future research.

2. Literature review

In this section, we review some studies related to the problems studied in this paper, such as HFSP, BHFSP and group shop scheduling. HFSP is a discrete combinatorial optimization problem. BHFSP is based on HFSP considering blocking constraints. Moreover, this section reviews some related studies on group shop scheduling problems.

HFSP has a very wide range of application scenarios in modern intelligent and smart manufacturing systems [2,3]. In these application scenarios, the utilization of parallel production mode can effectively improve the processing efficiency of factories and reduce unnecessary time cost or energy waste. Many optimization algorithms have been developed to solve HFSP effectively. In [35], a hybrid evolutionary algorithm using two solution representations is proposed to solve the HFSP for makespan minimization. Then, a multi-objective decomposition evolutionary algorithm based on ant colony optimization behavior is proposed to solve the problem from the point of view of factory production and management, and the duration and total electricity cost are considered as the optimization objectives [36]. In addition, there are some meta-heuristic algorithms with good performance, such as particle swarm optimization algorithm [37], artificial bee colony algorithm [38]. Although all the above algorithms find good solutions, some studies have argued that more efficient strategies should be introduced to solve the multi-objective optimization problem for HFSP [39–41]. From the above literature, it is clear that all meta-heuristic algorithms perform well in solving HFSPs, but they do not consider and study HFSPs with job blocking and group situations.

BHFSP is an extension of HFSP that is more complex and has no buffers between adjacent machines to store finished jobs. It has not been studied as much as HFSP. Specifically, after a job is completed in the previous stage, if there is no available machine in the next stage, then the job blocks on the current processing machine. So far, mathematic models of the problem has been proposed [42–44]. However, the aforementioned studies only addressed small-scale BHFSPs, and the large-scale problems were either unsolved or unsolvable. This is due to the fact that the computation time is exploding as the problem size continuously grows. Therefore, a better solution cannot be found in the limited time [45]. Consequently, a number of meta-heuristic algorithms that can solve large-scale problems have been proposed, such as the classical simulated annealing (SA) algorithm [46] and genetic algorithm (GA) [47]. The above-mentioned algorithms can solve the BHFSP efficiently. Moreover, some algorithms have been used to solve the problem with different objectives such as optimized energy consumption [48], due date window [49], total tardiness and earliness [15]. Although the meta-heuristic algorithms proposed in the above literature can solve both small and large scale BHFSPs. However, the case of groups of jobs is not considered in BHFSP, The setup time of the group on different machines is also not taken into account.

As the group technology of jobs continues to grow, the studies on its related applications are also increasing [50–52]. The methods in these studies mainly include mathematical models, heuristic and meta-heuristic algorithms. In [53], a polynomial optimization method was proposed to solve the job group problem. In [54], two mathematical models were proposed to optimize the total energy consumption. In the above-mentioned studies, grouping of jobs can effectively reduce the setup time of different machines and improve the processing efficiency of the factory. Additionally, the group technique can reduce thorny issues such as production lead time and raw material inventory [55]. In addition, heuristic and meta-heuristic algorithms have been carried out so far for solving the group problems. In [50], an improved constructive heuristic algorithm was proposed to minimize the makespan in group scheduling. In [56], a hybrid meta-heuristic genetic algorithm was considered to be introduced to solve the grouping scheduling problem of jobs. The researcher also considered blocking constraints in the group problem and devised effective meta-heuristics to solve the problem [57]. However, this paper did not consider the situations of parallel machines. Qin et al. [58] considered the situation of group in BHFSP. However, they only optimized makespan without energy consumption, and the structure of the algorithm was more complex and needed further improvement.

The above studies have proposed different algorithms that can address BHFSP or group scheduling. However, there are few strategies to change the job ordering based on job blocking. This paper bridges the gap in the design of strategies and proposes a single-individual based variable neighborhood search algorithm. In the proposed algorithm, the single-individual mechanism of the IG algorithm is introduced, which has the advantages of simple structure and easy implementation, and improves the quality of the solution by continuously optimizing a solution. Subsequently, a group-based and a job-based neighborhood search strategies are proposed, respectively. The group-based neighborhood search strategy (G_NS) can effectively solve the group ordering problem and reduce the machine setup time. The job-based neighborhood search strategy (J_NS) reduces the impact of blocking on energy consumption by improving the ordering of jobs in a group.

3. Problem statement

In this section, the symbol definition, encoding and decoding rules for BHFGSP are given.

3.1. Problem formulation

See [58], for ease of description, the following parameters and variables are given as follows. See [25,26], we also denote a collection of jobs grouped into one set as a family in our problem descriptions.

Notations:

S	Number of stages.
s	Index of stages, $s \in \{1,2,\ldots,S\}$.
M_s	Number of parallel machines at stage s.
m	Index of machines, $m \in \{1,2,\ldots,M_s\}$.
F	Number of families.
f, f'	Index of families, $f, f' \in \{0,1,\ldots,F\}$, 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, $j, j' \in \{1,\ldots,N\}$.
$p_{j,s}$	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.
$EC_s^{Process}$	Energy consumption per unit time of processing.
$EC_s^{Blocking}$	Energy consumption per unit time of blocking.
EC_s^{Idle}	Energy consumption per unit time of idle.
$EC_s^{Setting}$	Energy consumption per unit time of setting.
PEC	Total Energy consumption of processing.
BEC	Total Energy consumption of blocking.
IEC	Total Energy consumption of idle.
SEC	Total Energy consumption of setting.
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.
$E_{s,m}$	The shutdown time of machine m 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.
$z_{j,s,m}$	Binary decision variable, 1 if the job j is processed on the machine m of stage s, 0 otherwise.

Constraints:

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

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

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

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

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

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

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

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

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

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

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

$$E_{s,m} \geqslant d_{j,s},\ \forall j \in \{1,2,\ldots,N\},\ \forall s \in \{1,2,\ldots,S\},\ \forall m \in \{1,2,\ldots,M_s\} \tag{12}$$

$$PEC = \sum_{s=1}^{S}\sum_{j=1}^{J}\left(EC_s^{Process}\cdot p_{j,s}\right) \tag{13}$$

$$IEC = \sum_{s=1}^{S} EC_s^{Idle}\cdot\left(\sum_{m=1}^{M_s} E_{s,m} - \sum_{j=1}^{J} p_{j,s} - \sum_{j=1}^{J}\left(d_{j,s} - c_{j,s}\right)\right) \tag{14}$$

$$BEC = \sum_{s=1}^{S}\sum_{j=1}^{J}\left(EC_s^{Blocking}\cdot\left(d_{j,s} - c_{j,s}\right)\right) \tag{15}$$

$$SEC = \sum_{f=1}^{F}\sum_{f'=1}^{F}\sum_{s=1}^{S}\left(EC_s^{Setting}\cdot set_{f,f',s}\cdot x_{f,f',s}\right) \tag{16}$$

Objective:

$$Minimize\ TEC = (PEC + IEC + BEC + SEC) \tag{17}$$

Constraints (1) and (2) make sure that each family has a unique immediate predecessor and successor at every stage. Constraints (3) and (4) specify that the dummy family can only act as an immediate successor or predecessor to a family a maximum of M_s times at any given stage s. Constraint (5) mandates that the dummy family must have an equal count of immediate successors and predecessors across all stages. Constraint (6) dictates that for jobs j and j' belonging to the same family f, one must be processed before the other. If job j precedes job j' in processing, as per constraint (7), the completion time of job j' at stage s must be no earlier than the sum of the departure time of job j and its processing time $p_{j',s}$. For two consecutive families f and f' at stage s, where f' directly follows f, constraint (8) mandates that the completion time of job j' from family f' must be at least the sum of the departure time of job j from family f, the processing time $p_{j',s}$, and the setup time $set_{f,f',s}$. The initial setup time $set_{0,f,s}$ for the first job of a family at stage s is taken into account by constraint (9). Constraint (10) states that a job's departure time cannot be earlier than its completion time at the same stage. If the departure time $d_{j,s}$ equals the completion time $c_{j,s}$, job j is unobstructed at stage s. However, if $d_{j,s}$ exceeds $c_{j,s}$, job j is blocked at that stage. Constraint (11) establishes that a job's completion time is calculated by adding its processing time at the current stage to its departure time from the preceding stage. Constraint (12) guarantees that the shutdown time of the machine at a certain stage is not less than the departure time of the job on the same machine. Eq. (13) represents the processing energy consumption for all jobs. Eq. (14) indicates the total idle energy consumption. Eq. (15) denotes the total blocking energy consumption. Eq. (16) expresses the setup energy consumption of all machines. Eq. (17) represents the objective to be optimized: total energy consumption.

In the BHFGSP, a factory consists of S distinct stages, each equipped with $M_s\left(M_s = 1,2,\ldots,M_S\right)$ machines operating in parallel to process jobs. There are no intermediate buffers between any adjacent machines. The problem involves $F\,(1,2,\ldots,f,\ldots,F)$ families need to be processed on the machines, where the set of jobs in family of jobs, with the set of jobs belonging to family f represented as ω_f. Each job j within family f must traverse all stages without deviation. Furthermore, all jobs from

Table 1
The processing time of all jobs at every stage.

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

Table 2
The setup time of families at every stage.

Stage1						Stage2						Stage3					
Family	0	1	2	3	4	Family	0	1	2	3	4	Family	0	1	2	3	4
0	0	3	3	5	6	0	0	5	2	6	4	0	0	8	4	5	6
1	0	0	3	5	4	1	0	0	3	2	4	1	0	0	3	2	7
2	0	6	0	4	8	2	0	7	0	8	5	2	0	7	0	6	5
3	0	4	5	0	4	3	0	2	3	0	5	3	0	5	4	0	4
4	0	4	5	5	0	4	0	3	3	5	0	4	0	5	4	3	0

the same family are required to be processed on the same machine. Prior to processing, it is necessary to account for the family machine setup time denoted as set_{s,f_1,f_2} when transitioning between family f_1 and f_2 at stage s. The machine setup time for the initial family at stage s is represented by $set_{s,0,f_2}$. The processing time required for job j at stage s is designated as $p_{j,s}$. The objective of this paper is to minimize the energy consumption of BHFGSP. To further elucidate the problem, the following assumptions are made:

(1) Each job must be processed through all stages in sequence; skipping stages or completing the process before all stages are addressed is not permitted.
(2) The total processing time for each job encompasses both the actual processing time and any transportation time needed between stages.
(3) A machine can only process one job at a time, and similarly, a job is processed by only one machine at any given moment.
(4) Once processing of a job from a particular family begins on a machine, it must continue without any interruption from jobs of other families until completion.
(5) All jobs are ready to be processed at time zero.

3.2. Encoding and decoding procedure

To address large-scale and intricate combinatorial optimization problems, efficient encoding and decoding techniques are indispensable for diminishing computational demands and elevating the caliber of solutions [59]. See [26,60–62], in tackling the BHFGSP, we employ integer encoding to represent the solution. A solution is encoded as an ordered pair (π, τ), with π representing a collection of F families, expressed as $\pi = \{\pi_1, \pi_2, \ldots, \pi_l, \ldots, \pi_F\}$. τ signifies the sequence of jobs for each family, where $\tau = \{\tau_1, \ldots, \tau_l, \ldots, \tau_F\}$. Each τ encapsulates a sequence of jobs, $\tau_l(\tau_l = \{\tau_{l,1}, \ldots, \tau_{l,n_l}\}$, where $, n_l$ is the number of jobs in family π_l.

The decoding rule is shown in Algorithm 1. For convenience of description, we denote the time available to the machine m_s at stage s as $MIdle_{s,m_s}$. In this decoding rule (Lines 2–32), we first schedule the family. Subsequently, for each job in the family, we assign it a machine for each stage. In Lines 5–8 of step 1, if machine m_s has not processed a job, we will select m_s as the processing machine for jobs of the same family. In Lines 8–11 of step 1, if machine m_s has processed jobs, the machine with the minimum $MIdle_{s,m_s} + Set_{s,f_1,f_2}$ value will be selected as the processing machine. Additionally, after selecting the machine, the set energy consumption SEC of the machine is calculated. In step 2, after selecting the appropriate machine according to step 1, jobs in the selected family π_f are scheduled for processing in the first stage. The processing energy consumption PEC is calculated (Lines 15–18). In Lines 19–30, schedule all jobs in the selected family in stages 2 to S. Based on the completion time of the job in the previous stage and the idle start time of the selected machine in the current stage, the energy consumption of the job being blocked and the machine being idle is computed. In step 3, when all the jobs of all the families have been completed, the optimization objective of this paper will be calculated: the Total Energy Consumption TEC. This decoding rule is simplified to DR() when used in Section 4. In addition, one point to note is that all the algorithms in this paper are implemented on the basis of the aforementioned decoding rule.

Algorithm 1 Decoding rule of energy-efficient BHFGSP

Input: Solution (π, τ)
Output: $TEC = PEC + IEC + BEC + SEC$
1: $SEC = 0$
2: **for** $f = 1$ **to** F **do**
3: **for** $s = 1$ **to** S **do** ▷ *Step 1: Select the appropriate machine at each stage*
4: Find the family π_f in π
5: **if** m_s has not processed a job **then**
6: $MIdle_{s,m_s} = MIdle_{s,m_s} + Set_{s,0,f_2}$
7: $SEC = SEC + Set_{s,0,f_2} \cdot EC_s^{Setting}$
8: **else**
9: Find m_s that has the minimum $MIdle_{s,m_s} + Set_{s,f_1,f_2}$ value
10: $SEC = SEC + Set_{s,f_1,f_2} \cdot EC_s^{Setting}$
11: **end if**
12: **end for**
13: $PEC = 0$, $BEC = 0$, $IEC = 0$
14: **for** $j = \tau_{f,1}$ **to** τ_{f,n_f} **do** ▷ *Step 2: Schedule the jobs on the selected machines*
15: $c_{j,1} = MIdle_{1,m_1} + p_{j,1}$
16: $PEC = PEC + p_{j,1} \cdot EC_1^{Process}$
17: $d_{j,1} = c_{j,1}$
18: $MIdle_{1,m_1} = d_{j,1}$
19: **for** $s = 2$ **to** S **do**
20: $c_{j,s} = \max\{MIdle_{s,m_s}, c_{j,s-1}\} + p_{j,s}$
21: **if** $MIdle_{s,m_s} > c_{j,s-1}$ **then**
22: $BEC = BEC + (MIdle_{s,m_s} - c_{j,s-1}) \cdot EC_s^{Blocking}$
23: **else**
24: $IEC = IEC + (c_{j,s-1} - MIdle_{s,m_s}) \cdot EC_s^{Idle}$
25: **end if**
26: $d_{j,s} = c_{j,s}$
27: $MIdle_{s,m_s} = d_{j,s}$
28: $d_{j,s-1} = c_{j,s} - p_{j,s}$
29: $MIdle_{s-1,m_{s-1}} = d_{j,s-1}$
30: **end for**
31: **end for**
32: **end for**
33: $TEC = PEC + IEC + BEC + SEC$ ▷ *Step 3: Obtain the total energy consumption (objective value)*

3.3. An example of BHFGSP

In order to describe the decoding rule of BHFGSP more intuitively and conveniently, we give a concise and complete Gantt chart of one example: $F = 4$, $N = 8$, $S = 3$, $\tau_1 = \{1,2\}$, $\tau_2 = \{3,4\}$, $\tau_3 = \{5,6,7\}$, and $\tau_4 = \{8\}$, which are shown in Fig. 1. All these jobs are scheduled based on the $(\tau_1, \tau_2, \tau_3, \tau_4)$ arranged in advance. Tables 1 and 2 give the processing time of distinct jobs and setup time of families at every stage. The details are shown as follows:

As shown in Fig. 1, different jobs, blocking, and setup times are represented by rectangular blocks of different colors. The total energy consumption value to be optimized in this paper is the sum of machine idle energy consumption between all rectangular blocks and different rectangular blocks.

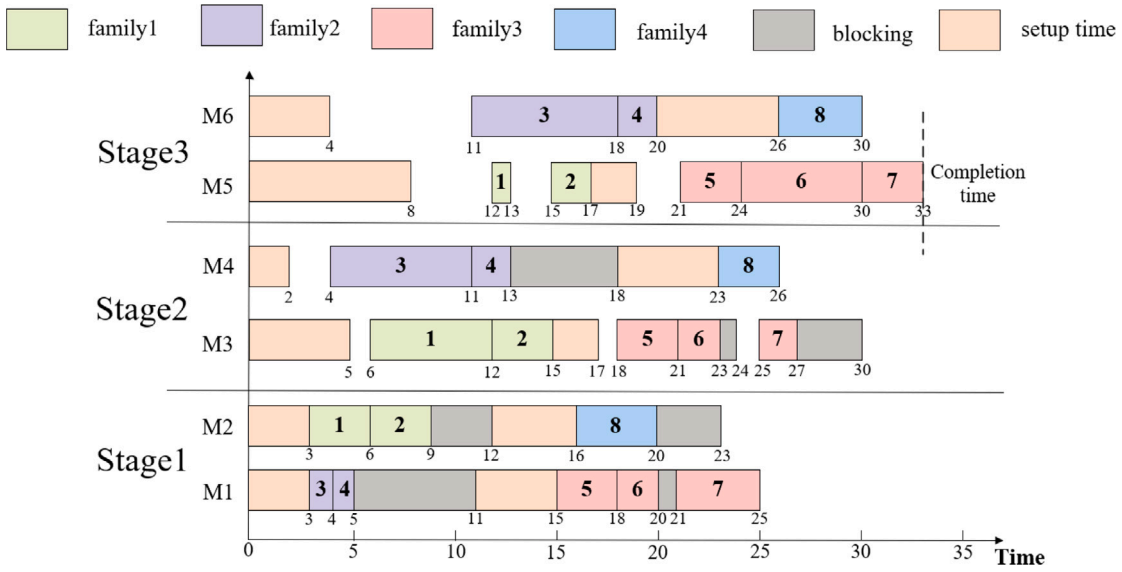


Fig. 1. The Gantt chart of the BHFGSP.

4. Proposed algorithm

This section presents the details of the SIVNS algorithm, which consists of the following parts, i.e., the framework of the proposed algorithm, the initialization method, the group-based neighborhood search strategy, and the job-based neighborhood search strategy.

Algorithm 2 The framework of SIVNS.

Input: Max_Iter, $Temperature$
Output: $(\pi^{best}, \tau^{best})$
1: $iter \leftarrow 1$
2: $(\pi^N, \tau^N) \leftarrow NEH\_Fam()$ // *Algorithm.3*
3: $(\pi, \tau) \leftarrow (\pi^N, \tau^N)$
4: $(\pi^{best}, \tau^{best}) \leftarrow (\pi, \tau)$
5: **while** $iter \leqslant Max\_Iter$ **do**
6: $(\pi^G, \tau^G) \leftarrow G\_NS((\pi^N, \tau^N))$ // *Algorithm.4*
7: $(\pi^J, \tau^J) \leftarrow J\_NS((\pi^G, \tau^G))$ // *Algorithm.5*
8: **if** $DR((\pi^J, \tau^J)) \leqslant DR((\pi, \tau))$ **then**
9: $DR((\pi, \tau)) \leftarrow DR((\pi^J, \tau^J))$
10: **if** $DR((\pi, \tau)) \leqslant DR((\pi^{best}, \tau^{best}))$ **then**
11: $DR((\pi^{best}, \tau^{best})) \leftarrow DR((\pi, \tau))$
12: **end if**
13: **else if** $random \leqslant exp\{-(DR((\pi^J, \tau^J)) - DR((\pi, \tau)))/Temperature\}$ **then** // *SA acceptance criteria*
14: $(\pi, \tau) \leftarrow (\pi^J, \tau^J)$
15: **end if**
16: **end while**
return $(\pi^{best}, \tau^{best})$

Inspired by the single-individual framework of the IG algorithm, we also propose a single-individual based variable neighborhood search algorithm. The neighborhood search strategies further improve the current solution by perturbing the scheduling order of groups, random jobs, and blocking jobs. This framework can effectively improve the local search ability of the algorithm without dispersing too much effort into many poorly performing solutions. A concise framework for this algorithm is shown in Algorithm 2.

In Algorithm 2, the proposed SIVNS preserves the main framework of the novel IG (NIG) algorithm in [58]. Its components are as follows: (1) The Nawaz, Enscore, and Ham initialization method related to the group (NEH_Group) is employed to create the initial sequence. (2) Group-based neighborhood search strategies are proposed to address the group ordering issue as well as the situation of machine settings to reduce the total energy consumption. (3) A job-based neighborhood search strategy is suggested, and thereby changing the sequence of blocked jobs as well as random jobs in groups, thus reducing the energy consumption of blocking and machine idle. 5) The SA acceptance criteria update the current solution based on the Temperature parameter, which is aimed at increasing the diversity of solutions. To some extent, it avoids the solution from falling into local optima. Since SA is not the innovation of this paper, the approach will not be described in detail in subsequent sections. For more details, please see [63].

4.1. NEH_Fam method

The ordering of groups directly affects the idle energy consumption of the scheduling sequence and the energy consumption of the machine settings. Moreover, a reduction in the setup time of families can potentially lead to a decrease in the objective value for the whole scheduling sequence [26]. NEH heuristic algorithms are widely used because of their high efficiency and simplicity of ideas. It is often used for initialization of meta-heuristic algorithms [64]. In [58], The *NEH_Fam* heuristic algorithm reduces the machine idle and machine setup time by changing the scheduling ordering between different groups and thus reducing the machine idle and machine setup time. In addition, it effectively avoids the problem of violating the constraint that jobs between different groups cannot cross over. Therefore, this initialization strategy is adopted in this paper to generate a high-quality solution. Algorithm3 gives the detailed steps.

Algorithm 3 *NEH_Fam* method

Input: $(\pi^{origin}, \tau^{origin})$, $\pi^{sub} = \{\}$
Output: (π^N, τ^N)
1: $P_l \leftarrow \sum_{s=1}^{S} \sum_{j=1}^{n_l} p_{j,s,l}$, $l = 1, ..., F$
2: Sort family sequence $\{\pi_1^{origin}, ..., \pi_l^{origin}, ..., \pi_F^{origin}\}$ based on the descending P_l, π^A is obtained.
3: **for** $l = 1$ **to** F **do**
4: **for** $i = 1$ **to** $|\pi^{sub}| + 1$ **do**
5: Retrieve family π_l^{origin} from π^A, and put it in the i^{th} position of π^{sub}. A new π_i^{sub} is obtained
6: **end for**
7: $(\pi^{sub}, \tau^{sub}) \leftarrow arg\ \min_{i=1}^{|\pi^{sub}|+1} DR(\pi_i^{sub}, \tau_i^{sub})$
8: **end for**
9: $(\pi^N, \tau^N) \leftarrow (\pi^{sub}, \tau^{sub})$

In Algorithm 3, $p_{j,s,l}$ denotes that the processing time of j belongs to the family l at stage s. In Lines 1–2, calculate the total processing time at all stages for all jobs belonging to the same group, arranging the families in descending order. In Lines 3–8, we retrieve the families one by one in the descending order and insert them into all positions of the remaining family sequence. For each retrieved family, the job sequence that results in the minimum energy consumption after all possible insertion positions have been considered will be saved. As in line 7, the energy consumption value is computed by using the decoding rule DR and finding the minimum energy consumption value for all current job sequences. In Line 9, we get a complete scheduling sequence.

4.2. Group-based neighborhood search strategy

In the scheduling process of the BHFGSP, the processing time for different jobs are pre-determined and immutable. The selection of machines is automatically determined by the decoding rule. Consequently, the main focus is on mitigating the impact of the setup time of machines and the order of group sequence. To achieve this, we have developed a group-based neighborhood search strategy that adjust the group sorting to diminish the setup time of machines, thereby further minimizing the total energy consumption. Moreover, all strategies are constructed around swap operators, which simplifies the process and reduces computational time complexity [65].

As shown in Algorithm 4, two group-based swap operators are employed to adjust the group sequence. Within the framework of the SIVNS algorithm, the incorporation of group-based neighborhood search strategy efficiently improves the order of group sorting. These operators are instrumental in broadening the search space and improving the solution quality. Specifically, the proposed strategy introduces two group-based swap operators tailored to reduce the energy consumption of machine setup time.

Algorithm 4 Group-based neighborhood search strategy

Input: (π^N, τ^N)
Output: (π^G, τ^G)
1: $rnd \leftarrow randi\,(1,2)$ ▷ A random integer in [1, 2]
2: **if** $rnd = 1$ **then**
3: $(\pi^G, \tau^G) \leftarrow$ ***Group-random_operator*** $((\pi^N, \tau^N))$
4: **else**
5: $(\pi^G, \tau^G) \leftarrow$ ***Group-perturb_operator*** $((\pi^N, \tau^N))$
6: **end if**

The details of the ***Group-random_operator*** () is shown as follows:

Step1: Randomly choose two families, denoted as π_a^{rand}, π_b^{rand}, from the set π^N and swap their positions. Then, a new sequence $(\pi^{rand}, \tau^{rand})$ is obtained.

Step2: If the dispatching performance $DR\,(\pi^{rand}, \tau^{rand})$ is less than or equal to the initial dispatching performance $DR\,(\pi^N, \tau^N)$, $(\pi^N, \tau^N) \leftarrow (\pi^{rand}, \tau^{rand})$; Otherwise, $(\pi^{rand}, \tau^{rand}) \leftarrow (\pi^N, \tau^N)$.

Step3: Repeat steps 1 and 2 for a total of R iterations. After the final iteration, $(\pi^G, \tau^G) \leftarrow (\pi^{rand}, \tau^{rand})$.

The details of the ***Group-perturb_operator*** () is shown as follows:

Step1: Set the counter $count \leftarrow 1$ and $count2 \leftarrow count + 1$.

Step2: While $count2$ is less than or equal to the total number of families F, perform a swap between the positions of $\pi_{count}^{perturb}$ and $\pi_{count2}^{perturb}$.

Step3: If the new dispatching performance $DR\,(\pi^{perturb}, \tau^{perturb})$ is less than or equal to the current dispatching performance $DR\,(\pi^N, \tau^N)$, $(\pi^N, \tau^N) \leftarrow (\pi^{perturb}, \tau^{perturb})$; Otherwise, $(\pi^{perturb}, \tau^{perturb}) \leftarrow (\pi^N, \tau^N)$.

Step4: Continue swapping the positions of $\pi_{count}^{perturb}$ and $\pi_{count2}^{perturb}$ until $count2$ reaches F.

Step5: Once $count2$ has reached F, $count{+}{+}$ and $count2 \leftarrow count + 1$.

Step6: Continue the operations described above until $count$ equals $F - 1$. At this point, $(\pi^G, \tau^G) \leftarrow (\pi^{perturb}, \tau^{perturb})$.

4.3. Job-based neighborhood search strategy

Due to the absence of buffers, jobs may be blocked on the current machine. The blocking constraint prevents the completion of the entire process and increases the ineffective energy consumption of the machine. Therefore, after adjusting the arrangement order between groups, it is necessary to further perform job adjustment to minimize the impact of blocking on energy consumption. In this subsection, a job-based neighborhood search strategy is proposed to improve the quality of the solution.

Algorithm 5 Job-based neighborhood search strategy

Input: (π^G, τ^G)
Output: (π^J, τ^J)
1: $Blocking_j \leftarrow \sum_{s=1}^{S} (d_{j,s} - c_{j,s}),\ j = \{1, 2, ..., N\}$ ▷ Store the blocking time of all jobs at each stage
2: Sort $\{Blocking\_1, ..., Blocking\_j, ..., Blocking\_N\}$ according to the descending sequence.
3: **for** $j = 1$ **to** N **do**
4: **if** $|JobInFam\,[Blocking_j]| = 1$ **then** ▷ this group that has only one job
5: $(\pi^J, \tau^J) \leftarrow$ ***Single-job_operator*** $((\pi^G, \tau^G), JobInFam\,[Blocking_j])$
6: **else**
7: $rnd \leftarrow randi\,(1,2)$ ▷ a random integer in [1, 2]
8: **if** $rnd = 1$ **then**
9: $(\pi^J, \tau^J) \leftarrow$ ***BlockingJob-random_operator***$((\pi^G, \tau^G), JobInFam\,[Blocking_j])$
10: **else**
11: $(\pi^J, \tau^J) \leftarrow$ ***Job-random_operator***$((\pi^G, \tau^G))$
12: **end if**
13: **end if**
14: **end for**

As shown in Algorithm 5, in the proposed strategy, we first calculate the total blocking time of all jobs in all stages (Line 1). Then, the sequence containing the job indexes is sorted in descending order based on the blocking time of all jobs, $Blocking\_j$ indicates the job that in position j of sequence $Blocking$. $JobInFam[j]$ stores the sort order number of jobs (Line 2). If a job has a blocking time of zero, it is positioned in the sequence subsequent to the jobs that cause blocking. This method ensures that the job with the longest blocking time is given the highest adjusted priority. In cases where a group comprises a single job, the operation on that job is redefined as a group operation. Specifically, a ***Single-job_operator*** is executed to rearrange the job order (Line 5). Additionally, operations such as ***BlockingJob-random_operator*** (Line 9) and ***Job-random_operator*** (Line 11) are also applied to jobs within the same sequence. Similar to the group-based neighborhood search strategy, the strategy for addressing blocking jobs and all these jobs is based on swap operations. These operations are highly effective in reducing energy consumption caused by machine blocking and idling. Furthermore, they compensate for the SIVNS algorithm's limitations in search ability.

The details of the ***Single-job_operator*** is shown as follows:

Step1: Identify the family $JobInFam\,[Blocking_j]$ containing the blocking job, denoted as π_{select}^G. Perform an initial swap of π_{select}^G with all other families.

Step2: Determine the family sequence with the minimum energy consumption, designated as π^{inter}.

Step3: If the dispatching performance $DR\,(\pi^{inter}, \tau^{inter})$ is less than or equal to the initial group's dispatching performance $DR\,(\pi^J, \tau^J)$, $(\pi^J, \tau^J) \leftarrow (\pi^{inter}, \tau^{inter})$; Otherwise, $(\pi^J, \tau^J) \leftarrow (\pi^G, \tau^G)$.

The details of the ***BlockingJob-random_operator*** is shown as follows:

Step1: Set the counter $count = 1$ and identify the job $\tau_{select}^G = Blocking_j$ within its family $JobInFam\,[Blocking_j]$, with position $Pos = Select$. Initialize the current sequences as $(\pi^J, \tau^J) \leftarrow (\pi^{inter}, \tau^{inter}) \leftarrow (\pi^G, \tau^G)$.

Step2: While the counter is less than or equal to a constant C, randomly select a job τ_{random}^{inter} from the same family as τ_{Pos}^{inter}, ensuring $\tau_{random}^{inter} != \tau_{Pos}^{inter}$.

Step3: Swap the positions of τ_{random}^{inter} with τ_{Pos}^{inter} to obtain a new sequence $(\pi^{inter}, \tau^{inter})$.

Table 3
Comparative results between different variants when F=20.

Instance		V-G		V-J		SIVNS	
	N × S	RPD	p-value	RPD	p-value	RPD	p-value
F=20	80 × 3	0.733	**1.563E−12**	2.921	**1.202E−61**	**0.366**	\
	80 × 5	0.387	**3.761E−12**	2.235	**8.994E−70**	**0.156**	\
	80 × 8	1.218	**4.778E−38**	3.428	**9.022E−85**	**0.146**	\
	100 × 3	0.424	**1.484E−34**	2.031	**1.897E−84**	**0.049**	\
	100 × 5	0.615	**2.760E−17**	3.758	**4.275E−71**	**0.088**	\
	100 × 8	1.034	**1.165E−24**	3.851	**1.352E−66**	**0.143**	\
	120 × 3	0.365	**1.023E−09**	2.744	**1.076E−66**	**0.121**	\
	120 × 5	0.566	**1.031E−05**	3.983	**2.956E−59**	**0.229**	\
	120 × 8	1.299	**1.500E−18**	3.923	**1.834E−53**	**0.395**	\
	140 × 3	0.259	**1.481E−04**	2.794	**4.427E−69**	**0.123**	\
	140 × 5	0.645	**7.035E−08**	3.635	**2.878E−56**	**0.253**	\
	140 × 8	0.517	**1.402E−02**	3.686	**1.443E−53**	**0.329**	\
	160 × 3	0.343	**2.502E−03**	4.462	**2.441E−68**	**0.174**	\
	160 × 5	0.683	**3.273E−07**	4.172	**6.237E−59**	**0.258**	\
	160 × 8	0.327	3.397E−01	3.613	**9.894E−56**	**0.259**	\
	180 × 3	0.411	**3.471E−04**	4.498	**8.136E−66**	**0.189**	\
	180 × 5	0.434	**2.169E−02**	4.551	**1.567E−60**	**0.259**	\
	180 × 8	0.619	**1.926E−04**	3.805	**1.518E−54**	**0.319**	\
	200 × 3	0.563	**1.465E−04**	5.913	**4.672E−68**	**0.242**	\
	200 × 5	0.417	2.434E−01	5.153	**5.986E−59**	**0.313**	\
	200 × 8	0.384	4.355E−01	4.288	**5.319E−56**	**0.317**	\
	220 × 3	0.294	**1.118E−05**	3.108	**6.615E−68**	**0.120**	\
	220 × 5	0.479	**1.712E−02**	4.651	**1.145E−59**	**0.290**	\
	220 × 8	0.444	6.689E−02	4.017	**6.315E−55**	**0.295**	\
	240 × 3	0.545	**2.696E−03**	6.888	**9.496E−69**	**0.288**	\
	240 × 5	0.233	6.739E−01	3.464	**2.965E−59**	**0.207**	\
	240 × 8	0.507	1.797E−01	5.052	**2.317E−55**	**0.370**	\
	260 × 3	0.463	**8.472E−03**	5.976	**2.401E−67**	**0.247**	\
	260 × 5	0.499	1.203E−01	5.633	**1.307E−59**	**0.347**	\
	260 × 8	0.721	**7.313E−03**	5.179	**6.416E−54**	**0.408**	\
	280 × 3	0.358	9.103E−02	5.336	**1.532E−68**	**0.243**	\
	280 × 5	0.436	2.565E−01	5.163	**9.717E−58**	**0.333**	\
	280 × 8	0.476	2.105E−01	4.647	**3.126E−55**	**0.361**	\
	300 × 3	0.312	**5.210E−03**	4.491	**3.321E−70**	**0.166**	\
	300 × 5	0.648	**1.629E−02**	5.974	**3.659E−59**	**0.388**	\
	300 × 8	0.513	1.162E−01	4.724	**1.471E−55**	**0.366**	\

Table 4
Comparative results between different variants when F=40.

Instance		V-G		V-J		SIVNS	
	N × S	RPD	p-value	RPD	p-value	RPD	p-value
F=40	80 × 3	0.566	**8.702E−26**	1.811	**1.944E−65**	**0.132**	\
	80 × 5	1.061	**9.596E−36**	2.768	**3.267E−70**	**0.182**	\
	80 × 8	1.033	**1.272E−57**	3.443	**1.605E−107**	**0.040**	\
	100 × 3	0.485	**2.614E−17**	2.406	**1.135E−65**	**0.128**	\
	100 × 5	0.930	**1.172E−17**	3.133	**1.719E−67**	**0.118**	\
	100 × 8	2.259	**1.837E−20**	3.866	**4.218E−62**	**0.244**	\
	120 × 3	0.212	**3.961E−02**	2.341	**9.334E−54**	**0.122**	\
	120 × 5	0.536	**3.300E−02**	3.605	**3.806E−48**	**0.338**	\
	120 × 8	0.885	**1.145E−06**	3.119	**1.760E−46**	**0.362**	\
	140 × 3	0.535	**3.336E−04**	3.237	**3.784E−56**	**0.257**	\
	140 × 5	1.170	**1.491E−06**	5.006	**6.388E−51**	**0.487**	\
	140 × 8	0.543	4.443E−01	4.048	**1.584E−48**	**0.469**	\
	160 × 3	2.765	**2.101E−29**	4.124	**2.265E−56**	**0.318**	\
	160 × 5	1.148	**4.122E−08**	4.144	**8.306E−51**	**0.411**	\
	160 × 8	0.936	**9.117E−06**	3.193	**7.288E−47**	**0.347**	\
	180 × 3	0.626	**1.169E−04**	4.356	**4.479E−58**	**0.266**	\
	180 × 5	0.781	**1.508E−03**	4.705	**1.878E−51**	**0.380**	\
	180 × 8	0.856	**1.322E−06**	3.415	**1.138E−49**	**0.336**	\
	200 × 3	0.537	5.971E−02	5.360	**7.478E−58**	**0.360**	\
	200 × 5	0.624	**1.078E−05**	3.135	**4.003E−52**	**0.297**	\
	200 × 8	1.943	**9.098E−16**	4.751	**2.038E−51**	**0.487**	\
	220 × 3	0.493	6.147E−01	4.579	**5.511E−42**	**0.423**	\
	220 × 5	0.902	**4.926E−08**	3.443	**4.365E−48**	**0.359**	\
	220 × 8	0.591	**6.864E−03**	3.703	**5.462E−51**	**0.337**	\
	240 × 3	0.301	1.827E−01	3.348	**5.412E−55**	**0.206**	\
	240 × 5	0.579	1.280E−01	4.883	**2.973E−53**	**0.403**	\
	240 × 8	1.684	**1.041E−11**	4.435	**3.388E−51**	**0.439**	\
	260 × 3	0.782	**1.087E−06**	4.534	**6.180E−61**	**0.285**	\
	260 × 5	1.353	**7.887E−08**	5.408	**3.340E−52**	**0.539**	\
	260 × 8	0.871	**2.801E−02**	5.427	**4.363E−51**	**0.547**	\
	280 × 3	0.988	**4.699E−05**	5.562	**1.255E−55**	**0.397**	\
	280 × 5	0.882	**1.249E−03**	5.530	**2.966E−54**	**0.455**	\
	280 × 8	0.861	**2.611E−03**	4.574	**8.117E−50**	**0.487**	\
	300 × 3	0.630	**7.398E−06**	4.276	**2.936E−58**	**0.251**	\
	300 × 5	0.578	**1.178E−02**	3.499	**9.619E−49**	**0.309**	\
	300 × 8	0.922	**1.737E−03**	5.158	**2.772E−51**	**0.497**	\

Step4: If the new dispatching performance $DR\left(\pi^{inter}, \tau^{inter}\right)$ is less than or equal to the current best $DR\left(\pi^{J}, \tau^{J}\right)$, $\left(\pi^{J}, \tau^{J}\right) \leftarrow \left(\pi^{inter}, \tau^{inter}\right)$ and set Pos to a new random position; Otherwise, $\left(\pi^{inter}, \tau^{inter}\right) \leftarrow \left(\pi^{J}, \tau^{J}\right)$ and increment the counter.

Step5: Repeat steps 2 to 4 until the counter exceeds C. C is the number of jobs in the current family.

The details of the ***Job-random_operator*** is shown as follows:

Step1: Set the counter $count = 1$, $\left(\pi^{J}, \tau^{J}\right) \leftarrow \left(\pi^{G}, \tau^{G}\right)$.

Step2: Randomly select a job, denoted as π^{G}_{select}. In addition, choose another random job $\pi^{G}_{select2}$ that is different from Job π^{G}_{select}.

Step3: Swap the positions of π^{G}_{select} with $\pi^{G}_{select2}$. A new sequence $\left(\pi^{inter}, \tau^{inter}\right)$ is obtained.

Step4: If the new dispatching performance $DR\left(\pi^{inter}, \tau^{inter}\right)$ is less than or equal to the current best $DR\left(\pi^{G}, \tau^{G}\right)$, $\left(\pi^{J}, \tau^{J}\right) \leftarrow \left(\pi^{inter}, \tau^{inter}\right)$; Otherwise, $\left(\pi^{inter}, \tau^{inter}\right) \leftarrow \left(\pi^{J}, \tau^{J}\right)$ and increment the counter.

Step5: Repeat steps 2 to 4 until the counter exceeds C. C is the number of jobs in the current family.

5. Experiments and analysis

5.1. Simulation environment settings and evaluating metric

To evaluate the performance of SIVNS algorithm in addressing the energy-efficient BHFGSP, we have utilized a series of test instances. See [58,66,67], we set the same running time ($\rho \times F \times S$ milliseconds) for all comparison algorithms to serve as the algorithm termination criterion. we have considered the following parameters: $F = \{20, 40, 60\}$ for the number of families, $N = \{80, 100, 120, 140, 160, 180, 200, 220, 240, 260, 280, 300\}$ for the number of jobs, and $S = \{3, 5, 8\}$ for the number of stages. This results in a comprehensive set of 108 different combinations. For each instance, processing time $p_{j,s}$ and setup time set_{s,f_1,f_2} are randomly sampled from a uniform distribution in the ranges [50, 99] and [10, 20], respectively. All algorithms are conducted on a 64-bit Windows 11 operating system, using an Intel Core i7-13790F processor running at 2.10 GHz with 16.0 GB of RAM. The algorithms are implemented in the C++ programming language within the Visual Studio. The box plots of all the algorithms were drawn by origin software, while the evolutionary curve plots were drawn by matlab.

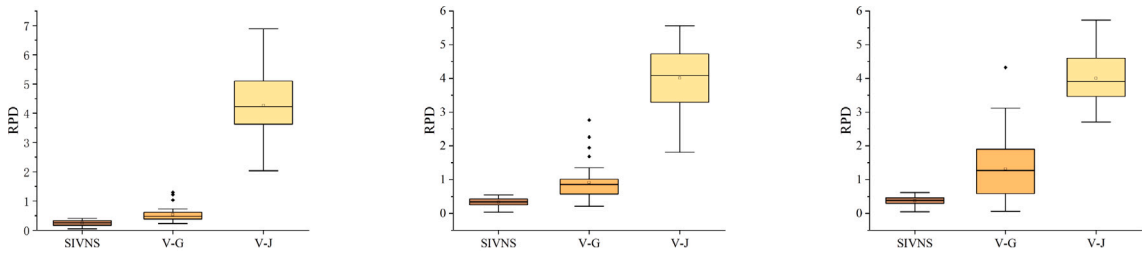
Following the approach of [58,68], we have set the termination criterion based on the maximum elapsed running time, calculated as (ρ×F×S) milliseconds. The parameter ρ is set to 5. If an algorithm's running time reaches the termination condition, the process is halted. Otherwise, the algorithm proceeds to the next iteration. It should be noted that in some large-scale instances, the running time may slightly exceed the set termination time, which is considered normal. For all algorithms, we used a metric of Relative Percentage Increment (RPD) to evaluate the overall performance of the algorithm. It measures the deviation of individual results from the mean and is widely used in much of numerous studies [69,70]. The equation is listed below:

$$RPD = \left(c_a - c_{best}\right) / c_{best} \times 100 \tag{18}$$

where c_a represents the total energy consumption obtained from algorithm a, and c_{best} is the minimum total energy consumption obtained from all these algorithms. A lower RPD value denotes better performance.

5.2. Evaluation of the SIVNS strategies

In this subsection, we delve into the performance of group-based and job-based neighborhood search strategies. As depicted in Tables 3,



(a) F=20 (b) F=40 (c) F=60

Fig. 2. The box plots of different variants.

4 and 5, the SIVNS algorithm encompasses the aforementioned dual neighborhood search strategies in 20, 40, and 60 groups of instances. A variant of the SIVNS that excludes the group-based neighborhood search strategy is designated as v-G. Similarly, a variant that excludes the job-based neighborhood search strategy is labeled as v-J. Each instance was subjected to 30 independent experimental runs with SIVNS, and the RPD is determined as the mean of these 30 trials.

Furthermore, we conducted Wilcoxon Test statistical analyses to ascertain whether there is a significant difference between the variants and SIVNS for each instance. In Tables 3, 4 and 5, should the *p*-value between any two algorithms be less than the 0.05 significance threshold, the null hypothesis is rejected, signifying a statistically significant difference between the compared algorithms. If not, the null hypothesis stands, suggesting that the difference between the algorithms is either negligible or nonexistent. Additionally, we have highlighted the best RPD values with a significant *p*-value in bold to emphasize these notable outcomes.

Examination of Tables 3, 4 and 5, reveals that neither v-G nor v-J achieved the optimal value in terms of RPD, whereas the SIVNS algorithm consistently obtained the best RPD values across all instances. Based on the *p*-value analysis, v-G showed significant differences from the SIVNS algorithm in 86 out of 108 instances. For v-J, the difference was significant in all 108 instances. This indicates that both search strategies significantly contribute to the enhancement of the algorithm's performance. To a certain extent, the group-based neighborhood search strategy reduces the setup time on machines between different groups, leading to a more rational and efficient group scheduling sequence. Notably, experimental comparisons indicate that the neighborhood search strategy tailored for addressing blocking and random jobs exerts a more pronounced influence on algorithm performance. This strategy significantly bolsters the algorithm's search ability, preventing it from becoming trapped in local optima and substantially reducing the energy consumption associated with machine blocking and idleness.

In addition, in order to visualize the variability of the solutions obtained between the different variants more intuitively, we draw the RPD box plots of the two variants with SIVNS at three different group sizes. As shown in Fig. 2, the solutions obtained by SIVNS are the most stable and both work better than v-G and v-J.

Table 5
Comparative results between different variants when F=60.

Instance		V-G		V-J		SIVNS	
	N × S	RPD	p-value	RPD	p-value	RPD	p-value
F=60	80 × 3	0.063	2.725E−01	2.707	**2.858E−92**	**0.080**	\
	80 × 5	0.264	**1.014E−24**	3.058	**5.210E−91**	**0.050**	\
	80 × 8	1.480	**5.554E−48**	2.775	**2.912E−77**	**0.069**	\
	100 × 3	0.255	1.261E−01	4.670	**2.116E−67**	**0.170**	\
	100 × 5	1.616	**4.374E−25**	3.897	**2.898E−59**	**0.234**	\
	100 × 8	1.340	**5.053E−26**	2.866	**1.141E−61**	**0.199**	\
	120 × 3	0.250	9.783E−01	3.765	**4.174E−50**	**0.248**	\
	120 × 5	3.117	**2.791E−28**	4.600	**4.330E−54**	**0.412**	\
	120 × 8	1.396	**2.499E−17**	3.987	**1.378E−50**	**0.350**	\
	140 × 3	0.367	5.215E−01	3.827	**1.486E−53**	**0.312**	\
	140 × 5	3.015	**1.477E−18**	4.194	**5.055E−43**	**0.588**	\
	140 × 8	1.933	**1.951E−18**	3.566	**5.515E−46**	**0.402**	\
	160 × 3	0.636	**2.395E−03**	4.166	**2.603E−53**	**0.320**	\
	160 × 5	0.946	**1.050E−02**	5.209	**1.212E−47**	**0.550**	\
	160 × 8	1.588	**4.525E−12**	3.390	**3.924E−45**	**0.406**	\
	180 × 3	0.853	**3.749E−03**	5.466	**9.520E−52**	**0.486**	\
	180 × 5	2.010	**2.801E−14**	3.385	**4.440E−42**	**0.489**	\
	180 × 8	1.306	**2.190E−16**	2.843	**2.134E−46**	**0.308**	\
	200 × 3	1.947	**2.351E−14**	5.339	**6.532E−52**	**0.496**	\
	200 × 5	0.722	**1.413E−02**	3.902	**7.668E−47**	**0.435**	\
	200 × 8	1.451	**1.546E−12**	3.923	**1.952E−46**	**0.428**	\
	220 × 3	0.727	**2.315E−02**	5.402	**6.024E−54**	**0.410**	\
	220 × 5	1.919	**4.933E−12**	4.285	**3.703E−47**	**0.440**	\
	220 × 8	1.228	**5.612E−15**	3.038	**4.244E−49**	**0.317**	\
	240 × 3	0.539	**7.888E−03**	3.539	**4.765E−53**	**0.285**	\
	240 × 5	0.786	**1.043E−04**	3.777	**1.097E−50**	**0.338**	\
	240 × 8	1.552	**1.409E−15**	3.110	**8.796E−43**	**0.429**	\
	260 × 3	0.721	**1.289E−02**	4.691	**6.839E−55**	**0.367**	\
	260 × 5	2.849	**5.560E−18**	4.331	**7.078E−52**	**0.408**	\
	260 × 8	4.324	**6.345E−20**	4.706	**3.191E−47**	**0.614**	\
	280 × 3	0.361	5.970E−01	3.608	**1.841E−45**	**0.309**	\
	280 × 5	0.389	6.922E−01	4.024	**1.511E−52**	**0.353**	\
	280 × 8	2.047	**2.204E−07**	4.603	**2.115E−43**	**0.606**	\
	300 × 3	0.318	5.902E−01	3.681	**1.218E−53**	**0.274**	\
	300 × 5	1.207	**1.306E−04**	5.729	**6.318E−50**	**0.620**	\
	300 × 8	1.874	**2.189E−14**	3.997	**4.947E−43**	**0.615**	\

5.3. Effectiveness of the SIVNS algorithm

To the best of our knowledge, the NIG algorithm is the only one that addresses the BHFGSP [58]. Consequently, we have selected the NIG algorithm as one of the comparative algorithms in this study. Additionally, we have included classical single-individual metaheuristic algorithms for comparison: the IGA [71]) and the double level mutation iterated greedy algorithm (IGDLM) [45]. Both the IGA and IGDLM have been applied to solve the HFSP and the BHFSP, respectively, which are analogous to the problem examined in this paper. Furthermore, we have employed the SA and GA [72] as a classical swarm intelligence algorithm for comparison. To ensure the experiments are conducted fairly, all algorithms have been implemented following the methods outlined in their original research, with parameter settings aligned with the primary sources. We incorporate the *group-based neighborhood search strategy* proposed in this paper into their framework for perturbing the group ordering (which is executed before perturbing the job sequences). Moreover, uniform encoding and decoding methods are utilized across all algorithms, and the termination condition is uniformly set to the same running time to maintain consistency. Tables 6, 7 and 8 list the average RPD values obtained in 30 independent runs for the different algorithms when the number of groups is 20,40, and

Table 6
The results of all comparison algorithms when F=20. Bold font represents the best RPD or values with significant differences.

Instance		IGA[2020]		SA[2019]		GA[2019]		IGDLM[2021]		NIG[2022]		SIVNS	
	N × S	RPD	p-value	RPD	p-value	RPD	p-value	RPD	p-value	RPD	p-value	RPD	p-value
F=20	80 × 3	3.861	**1.375E−62**	5.424	**1.480E−50**	3.871	**2.045E−63**	3.458	**1.455E−63**	**0.370**	**5.891E−05**	0.538	\
	80 × 5	2.700	**3.265E−71**	3.354	**3.484E−62**	2.727	**2.034E−71**	2.480	**4.541E−71**	0.192	8.781E−01	**0.189**	\
	80 × 8	3.927	**2.454E−83**	5.433	**2.759E−53**	4.134	**9.757E−79**	3.662	**3.243E−85**	0.620	**5.588E−24**	**0.193**	\
	100 × 3	2.206	**1.684E−85**	2.505	**3.651E−77**	2.235	**2.618E−85**	2.121	**2.326E−84**	0.189	**1.543E−11**	**0.059**	\
	100 × 5	3.938	**5.497E−72**	4.692	**4.452E−58**	3.917	**1.621E−71**	3.832	**2.525E−71**	0.375	**5.275E−08**	**0.127**	\
	100 × 8	4.286	**2.582E−57**	5.275	**1.374E−55**	4.181	**6.896E−67**	4.020	**9.527E−67**	0.526	**7.765E−06**	**0.281**	\
	120 × 3	2.825	**2.830E−67**	3.013	**4.503E−67**	2.860	**1.396E−67**	2.801	**4.000E−67**	0.357	**6.938E−08**	**0.130**	\
	120 × 5	4.177	**1.800E−57**	4.750	**5.407E−59**	4.187	**3.219E−60**	4.057	**1.590E−59**	0.638	**1.579E−06**	**0.261**	\
	120 × 8	4.375	**7.562E−52**	4.862	**7.860E−54**	4.108	**1.882E−54**	3.989	**7.982E−54**	0.831	**1.185E−06**	**0.403**	\
	140 × 3	2.868	**1.029E−69**	3.048	**1.317E−69**	2.859	**1.409E−69**	2.813	**3.172E−69**	0.309	**3.192E−05**	**0.125**	\
	140 × 5	3.677	**9.127E−57**	3.981	**1.937E−56**	3.671	**7.086E−57**	3.606	**1.783E−56**	0.529	**8.356E−06**	**0.197**	\
	140 × 8	3.892	**5.928E−53**	4.238	**2.421E−54**	3.800	**3.323E−54**	3.737	**8.319E−54**	0.726	**6.291E−06**	**0.347**	\
	160 × 3	4.491	**2.939E−68**	4.705	**6.342E−67**	4.453	**1.889E−68**	4.440	**2.240E−68**	0.467	**3.287E−07**	**0.146**	\
	160 × 5	4.233	**1.584E−59**	4.650	**9.306E−60**	4.213	**2.581E−59**	4.146	**4.910E−59**	0.628	**2.648E−07**	**0.216**	\
	160 × 8	3.649	**3.708E−56**	3.821	**2.632E−55**	3.620	**6.093E−56**	3.603	**8.147E−56**	0.544	**5.945E−05**	**0.238**	\
	180 × 3	4.569	**6.425E−66**	4.740	**6.324E−65**	4.519	**6.025E−66**	4.501	**7.584E−66**	0.501	**1.187E−05**	**0.187**	\
	180 × 5	4.616	**1.073E−60**	4.888	**2.316E−60**	4.614	**9.900E−61**	4.591	**1.324E−60**	0.671	**6.012E−06**	**0.285**	\
	180 × 8	3.871	**8.941E−55**	4.246	**5.940E−55**	3.870	**9.273E−55**	3.844	**1.364E−54**	0.659	**5.500E−05**	**0.351**	\
	200 × 3	6.039	**2.327E−68**	6.484	**1.617E−66**	5.964	**3.343E−68**	5.922	**4.302E−68**	0.715	**8.130E−08**	**0.241**	\
	200 × 5	5.153	**4.260E−59**	5.375	**3.589E−59**	5.133	**4.598E−59**	5.119	**5.455E−59**	0.790	**6.643E−07**	**0.272**	\
	200 × 8	4.464	**6.301E−56**	4.568	**4.653E−56**	4.323	**3.391E−56**	4.300	**4.555E−56**	0.761	**1.227E−06**	**0.317**	\
	220 × 3	3.178	**3.737E−68**	3.264	**3.137E−68**	3.124	**3.614E−68**	3.112	**4.840E−68**	0.350	**2.069E−07**	**0.102**	\
	220 × 5	4.670	**6.666E−60**	4.853	**5.323E−60**	4.648	**8.375E−60**	4.639	**9.536E−60**	0.727	**7.696E−07**	**0.264**	\
	220 × 8	4.025	**4.234E−55**	4.252	**9.342E−56**	4.029	**4.131E−55**	4.012	**5.203E−55**	0.670	**6.923E−06**	**0.278**	\
	240 × 3	7.242	**4.813E−70**	7.360	**2.205E−67**	6.901	**7.322E−69**	6.888	**9.655E−69**	0.815	**2.085E−08**	**0.263**	\
	240 × 5	3.487	**2.091E−59**	3.506	**1.628E−59**	3.475	**2.571E−59**	3.473	**2.662E−59**	0.492	**1.231E−05**	**0.210**	\
	240 × 8	5.082	**1.605E−55**	5.416	**7.531E−56**	5.082	**1.631E−55**	5.065	**1.971E−55**	0.874	**2.626E−06**	**0.367**	\
	260 × 3	6.105	**6.123E−68**	6.250	**2.724E−66**	5.978	**2.175E−67**	5.974	**2.276E−67**	0.685	**9.606E−07**	**0.239**	\
	260 × 5	5.737	**5.412E−60**	5.916	**3.704E−60**	5.671	**9.414E−60**	5.652	**1.128E−59**	0.878	**9.089E−07**	**0.351**	\
	260 × 8	5.280	**3.111E−54**	5.749	**3.499E−55**	5.197	**3.698E−54**	5.172	**4.695E−54**	0.943	**6.475E−07**	**0.363**	\
	280 × 3	5.538	**1.478E−69**	5.658	**8.374E−68**	5.341	**1.472E−68**	5.333	**2.026E−68**	0.589	**2.623E−06**	**0.235**	\
	280 × 5	5.211	**5.516E−58**	5.370	**3.399E−58**	5.184	**7.685E−58**	5.176	**8.435E−58**	0.793	**3.305E−06**	**0.332**	\
	280 × 8	4.776	**4.676E−56**	4.944	**6.939E−56**	4.655	**2.207E−55**	4.646	**2.494E−55**	0.816	**9.779E−07**	**0.339**	\
	300 × 3	4.522	**2.366E−70**	4.676	**3.703E−70**	4.519	**2.565E−70**	4.513	**2.954E−70**	0.492	**1.843E−07**	**0.171**	\
	300 × 5	5.990	**2.695E−59**	6.338	**2.890E−59**	5.987	**2.704E−59**	5.991	**2.718E−59**	0.963	**2.163E−07**	**0.365**	\
	300 × 8	4.791	**5.045E−56**	4.977	**4.077E−56**	4.727	**1.138E−55**	4.728	**1.164E−55**	0.814	**3.341E−06**	**0.349**	\

Table 7
The results of all comparison algorithms when F=40. Bold font represents the best RPD or values with significant differences.

Instance		IGA[2020]		SA[2019]		GA[2019]		IGDLM[2021]		NIG[2022]		SIVNS	
	N × S	RPD	p-value	RPD	p-value	RPD	p-value	RPD	p-value	RPD	p-value	RPD	p-value
F=40	80 × 3	2.092	**5.895E−72**	2.810	**7.368E−57**	2.237	**2.335E−71**	2.048	**6.790E−71**	0.438	**7.010E−19**	**0.132**	\
	80 × 5	3.320	**2.297E−75**	4.038	**5.798E−59**	3.289	**1.696E−73**	2.978	**2.711E−71**	0.560	**7.632E−15**	**0.182**	\
	80 × 8	4.059	**7.412E−69**	5.485	**5.141E−45**	3.939	**9.531E−91**	3.549	**4.16E−105**	0.403	**8.116E−45**	**0.040**	\
	100 × 3	2.644	**4.594E−68**	3.471	**1.218E−51**	2.756	**9.592E−67**	2.495	**2.347E−66**	0.377	**4.171E−12**	**0.128**	\
	100 × 5	3.431	**1.674E−66**	4.603	**6.454E−49**	3.423	**3.417E−68**	3.195	**5.884E−68**	0.421	**2.285E−10**	**0.118**	\
	100 × 8	6.344	**4.918E−34**	6.907	**3.591E−36**	4.367	**1.404E−61**	3.944	**1.392E−62**	0.626	**1.490E−08**	**0.244**	\
	120 × 3	2.379	**3.546E−54**	2.494	**2.910E−55**	2.427	**1.078E−54**	2.368	**4.572E−54**	0.286	**1.714E−04**	**0.122**	\
	120 × 5	3.721	**6.381E−49**	4.933	**1.349E−41**	3.833	**1.142E−49**	3.663	**1.442E−48**	0.585	**3.696E−03**	**0.338**	\
	120 × 8	3.799	**3.435E−49**	5.028	**8.120E−37**	3.369	**2.987E−48**	3.166	**6.778E−47**	0.503	8.664E−02	**0.362**	\
	140 × 3	3.388	**2.397E−57**	4.214	**3.398E−42**	3.357	**5.214E−57**	3.269	**2.097E−56**	0.690	**1.774E−09**	**0.257**	\
	140 × 5	5.586	**7.460E−50**	7.798	**3.523E−33**	5.213	**1.291E−51**	5.030	**4.752E−51**	1.104	**6.960E−07**	**0.487**	\
	140 × 8	4.284	**6.198E−50**	6.640	**1.039E−32**	4.259	**2.083E−49**	4.077	**1.045E−48**	1.200	**9.862E−06**	**0.469**	\
	160 × 3	5.131	**1.747E−56**	6.954	**5.941E−43**	4.313	**8.157E−57**	4.151	**1.646E−56**	0.844	**3.704E−07**	**0.318**	\
	160 × 5	4.773	**1.163E−54**	6.496	**4.785E−38**	4.283	**1.649E−51**	4.168	**6.003E−51**	0.886	**1.446E−05**	**0.411**	\
	160 × 8	3.294	**1.155E−47**	4.237	**2.697E−37**	3.312	**9.208E−48**	3.219	**4.397E−47**	0.794	**6.734E−07**	**0.347**	\
	180 × 3	5.764	**7.828E−56**	5.871	**3.067E−40**	4.433	**1.728E−58**	4.367	**3.844E−58**	0.870	**2.314E−08**	**0.266**	\
	180 × 5	5.258	**3.613E−44**	6.166	**2.661E−36**	4.767	**8.677E−52**	4.723	**1.519E−51**	1.248	**4.830E−09**	**0.380**	\
	180 × 8	3.861	**4.255E−44**	4.609	**1.160E−37**	3.517	**2.200E−50**	3.434	**8.504E−50**	0.740	**5.698E−06**	**0.336**	\
	200 × 3	5.610	**6.389E−58**	7.112	**3.461E−37**	5.456	**2.764E−58**	5.387	**5.571E−58**	0.956	**3.784E−07**	**0.360**	\
	200 × 5	3.304	**3.433E−53**	3.552	**1.027E−54**	3.316	**1.453E−53**	3.201	**1.322E−52**	0.686	**1.471E−06**	**0.297**	\
	200 × 8	5.071	**3.281E−53**	7.437	**3.227E−35**	4.973	**3.889E−52**	4.845	**9.860E−52**	1.403	**2.261E−09**	**0.487**	\
	220 × 3	4.672	**1.593E−42**	5.137	**9.068E−41**	4.641	**2.405E−42**	4.607	**3.793E−42**	0.799	**1.109E−02**	**0.423**	\
	220 × 5	3.759	**3.014E−50**	4.008	**1.129E−50**	3.686	**7.560E−50**	3.565	**6.177E−49**	0.901	**3.046E−08**	**0.359**	\
	220 × 8	3.874	**4.127E−52**	5.312	**2.768E−36**	3.773	**2.012E−51**	3.728	**3.807E−51**	0.883	**2.282E−08**	**0.337**	\
	240 × 3	3.508	**3.197E−56**	3.994	**1.783E−44**	3.374	**3.394E−55**	3.369	**3.908E−55**	0.515	**3.111E−05**	**0.206**	\
	240 × 5	5.462	**3.765E−56**	6.586	**7.751E−38**	4.946	**1.655E−53**	4.926	**2.098E−53**	0.993	**3.668E−07**	**0.403**	\
	240 × 8	5.308	**1.560E−55**	6.982	**2.195E−31**	4.549	**1.164E−51**	4.461	**2.556E−51**	1.086	**6.773E−07**	**0.439**	\
	260 × 3	4.872	**4.835E−62**	5.107	**8.748E−61**	4.715	**8.444E−62**	4.632	**1.998E−61**	0.815	**7.146E−09**	**0.285**	\

(*continued on next page*)

Table 7 (*continued*).

Instance		IGA[2020]		SA[2019]		GA[2019]		IGDLM[2021]		NIG[2022]		SIVNS	
	N × S	RPD	p-value	RPD	p-value	RPD	p-value	RPD	p-value	RPD	p-value	RPD	p-value
	260 × 5	5.775	**1.173E−53**	7.790	**1.592E−38**	5.487	**1.653E−52**	5.476	**2.494E−52**	1.218	**2.344E−07**	**0.539**	\
	260 × 8	5.901	**3.540E−52**	8.409	**1.064E−36**	5.598	**9.596E−52**	5.522	**2.562E−51**	1.486	**7.570E−08**	**0.547**	\
	280 × 3	6.305	**1.963E−59**	7.026	**1.431E−40**	5.625	**1.743E−56**	5.593	**2.681E−56**	1.090	**1.442E−07**	**0.397**	\
	280 × 5	5.751	**1.941E−54**	7.204	**7.233E−40**	5.578	**1.884E−54**	5.565	**2.262E−54**	1.097	**1.548E−07**	**0.455**	\
	280 × 8	5.416	**1.107E−46**	7.339	**5.771E−34**	4.682	**2.621E−50**	4.651	**3.951E−50**	1.200	**4.495E−08**	**0.487**	\
	300 × 3	4.319	**1.602E−58**	4.738	**4.647E−54**	4.292	**2.360E−58**	4.295	**2.279E−58**	0.670	**8.796E−06**	**0.251**	\
	300 × 5	3.635	**4.263E−48**	4.163	**2.232E−41**	3.557	**3.497E−49**	3.523	**6.517E−49**	0.746	**1.227E−05**	**0.309**	\
	300 × 8	5.515	**1.177E−51**	8.414	**1.942E−34**	5.275	**7.476E−52**	5.239	**1.316E−51**	1.431	**8.914E−08**	**0.497**	\

Table 8
The results of all comparison algorithms when F=60. Bold font represents the best RPD or values with significant differences.

Instance		IGA[2020]		SA[2019]		GA[2019]		IGDLM[2021]		NIG[2022]		SIVNS	
	N × S	RPD	p-value	RPD	p-value	RPD	p-value	RPD	p-value	RPD	p-value	RPD	p-value
	80 × 3	3.635	**6.566E−93**	4.476	**6.989E−54**	3.063	**2.214E−89**	2.781	**8.247E−93**	0.444	**4.279E−42**	**0.038**	\
	80 × 5	3.375	**1.174E−93**	5.086	**2.884E−51**	3.592	**8.335E−81**	3.156	**2.974E−91**	0.282	**4.857E−26**	**0.050**	\
	80 × 8	4.246	**1.626E−73**	5.496	**1.692E−52**	3.587	**2.687E−73**	2.971	**3.008E−78**	0.508	**4.136E−26**	**0.069**	\
	100 × 3	5.155	**3.903E−69**	6.916	**2.655E−48**	5.011	**1.539E−68**	4.758	**7.801E−68**	0.605	**1.632E−10**	**0.170**	\
	100 × 5	4.453	**9.27E−48**	6.127	**1.279E−47**	4.467	**1.075E−60**	3.996	**6.662E−60**	0.646	**1.374E−09**	**0.234**	\
	100 × 8	3.086	**3.517E−62**	5.100	**1.092E−41**	3.355	**3.222E−63**	2.959	**1.762E−62**	0.512	**4.917E−10**	**0.199**	\
	120 × 3	4.665	**1.964E−47**	5.129	**1.141E−45**	4.001	**1.962E−51**	3.833	**1.696E−50**	0.462	**9.373E−03**	**0.248**	\
	120 × 5	5.401	**2.34E−54**	7.833	**3.129E−42**	5.196	**1.337E−54**	4.700	**1.285E−54**	1.115	**2.29E−08**	**0.412**	\
	120 × 8	6.936	**2.321E−65**	6.337	**1.099E−37**	4.398	**1.791E−52**	4.052	**5.489E−51**	0.858	**4.998E−07**	**0.350**	\
	140 × 3	3.955	**1.906E−54**	4.596	**6.705E−49**	3.942	**2.389E−54**	3.867	**7.82E−54**	0.729	**2.46E−05**	**0.312**	\
	140 × 5	4.819	**7.748E−46**	7.916	**3.608E−39**	4.696	**3.925E−45**	4.271	**1.657E−43**	1.520	**3.415E−06**	**0.588**	\
	140 × 8	4.428	**6.375E−52**	6.394	**1.033E−45**	3.942	**6.779E−48**	3.634	**1.861E−46**	1.154	**2.591E−06**	**0.402**	\
	160 × 3	4.643	**3.267E−56**	5.724	**1.003E−45**	4.308	**5.762E−54**	4.189	**1.913E−53**	1.032	**5.841E−09**	**0.320**	\
	160 × 5	6.895	**2.823E−55**	8.141	**4.682E−38**	5.389	**2.29E−48**	5.252	**7.417E−48**	1.279	**3.908E−07**	**0.550**	\
	160 × 8	5.293	**2.58E−57**	6.257	**2.312E−42**	3.664	**8.113E−47**	3.432	**1.808E−45**	1.418	**3.754E−05**	**0.406**	\
	180 × 3	7.139	**6.253E−59**	9.853	**2.875E−45**	5.754	**2.188E−52**	5.508	**6.192E−52**	1.668	**4.868E−05**	**0.486**	\
	180 × 5	3.582	**2.636E−43**	5.064	**2.716E−39**	3.561	**2.835E−43**	3.425	**2.182E−42**	0.933	**3.940E−04**	**0.489**	\
	180 × 8	3.161	**5.034E−48**	3.753	**2.608E−47**	2.974	**1.702E−47**	2.878	**1.006E−46**	0.765	**4.509E−07**	**0.308**	\
F=60	200 × 3	11.584	**1.369E−72**	9.660	**4.11E−35**	5.673	**1.014E−52**	5.442	**4.394E−52**	1.528	**3.202E−07**	**0.496**	\
	200 × 5	4.780	**2.123E−52**	5.402	**5.404E−40**	4.011	**1.739E−47**	3.937	**4.545E−47**	1.022	**1.473E−05**	**0.435**	\
	200 × 8	5.291	**7.341E−40**	6.489	**4.658E−34**	4.099	**2.418E−47**	3.976	**1.009E−46**	1.250	**5.675E−07**	**0.428**	\
	220 × 3	5.789	**1.03E−55**	8.002	**5.349E−39**	5.543	**2.144E−54**	5.490	**3.068E−54**	1.096	**5.471E−06**	**0.410**	\
	220 × 5	4.498	**2.358E−48**	6.127	**3.948E−44**	4.458	**4.893E−48**	4.363	**1.428E−47**	1.062	**1.933E−06**	**0.440**	\
	220 × 8	3.153	**4.06E−50**	4.099	**8.925E−45**	3.099	**1.265E−49**	3.063	**2.681E−49**	0.813	**6.792E−07**	**0.317**	\
	240 × 3	3.608	**1.428E−53**	4.519	**9.737E−40**	3.590	**2.154E−53**	3.587	**2.535E−53**	0.983	**2.445E−08**	**0.285**	\
	240 × 5	3.882	**2.026E−51**	4.185	**2.332E−50**	3.813	**6.109E−51**	3.801	**7.693E−51**	0.929	**4.663E−08**	**0.338**	\
	240 × 8	3.527	**3.241E−46**	4.541	**9.907E−41**	3.242	**8.593E−44**	3.161	**3.383E−43**	0.983	**1.792E−05**	**0.429**	\
	260 × 3	7.300	**1.463E−64**	6.742	**5.832E−45**	4.801	**2.004E−55**	4.790	**3.864E−55**	0.987	**3.051E−06**	**0.367**	\
	260 × 5	5.377	**1.373E−54**	6.905	**5.772E−41**	4.541	**1.181E−52**	4.454	**2.132E−52**	1.155	**7.201E−07**	**0.408**	\
	260 × 8	6.357	**9.508E−56**	9.077	**1.566E−37**	5.011	**2.064E−48**	4.894	**1.057E−47**	1.723	**3.07E−05**	**0.614**	\
	280 × 3	3.774	**1.296E−46**	3.846	**1.435E−46**	3.635	**1.165E−45**	3.628	**1.323E−45**	0.794	**9.616E−05**	**0.309**	\
	280 × 5	4.060	**8.485E−53**	4.699	**9.893E−51**	4.044	**1.103E−52**	4.057	**9.961E−53**	0.917	**8.259E−07**	**0.353**	\
	280 × 8	4.865	**7.889E−45**	7.224	**1.503E−36**	4.782	**2.413E−44**	4.679	**9.28E−44**	1.272	**9.265E−05**	**0.606**	\
	300 × 3	3.800	**2.083E−54**	4.314	**4.109E−48**	3.708	**7.827E−54**	3.716	**7.125E−54**	0.780	**2.795E−07**	**0.274**	\
	300 × 5	6.593	**5.606E−50**	8.754	**5.262E−39**	5.984	**1.107E−50**	5.868	**2.511E−50**	1.744	**2.2E−06**	**0.620**	\
	300 × 8	5.514	**1.764E−49**	5.652	**7.902E−37**	4.023	**5.841E−44**	4.057	**1.783E−43**	1.435	**4.185E−05**	**0.615**	\

60, respectively. In addition, Wilcoxon Test is conducted and all the algorithms are compared with SIVNS.

As can be seen from Tables 6, 7, and 8, it clearly indicates that the SIVNS achieved the best RPD values in 107 out of 108 instances, while the NIG algorithm secured the best RPD value in just 1 out of 108 instances. In contrast, the IGA, SA, and GA did not attain the best RPD values. Based on the p-value analysis, there is a significant difference between SIVNS and NIG in 106 instances. Furthermore, all results from SIVNS demonstrate significant differences when compared to the IGA, SA, and GA. These findings underscore the superior performance of SIVNS over the other algorithms under comparison. To provide a visual representation of the RPD values among different algorithms across all instances, we have created box plots for each algorithm. As depicted in Fig. 3, each algorithm is represented by boxes of different colors, with the proposed SIVNS exhibiting the most consistent RPD range. This further substantiates the significant performance difference between the proposed algorithm and the others under comparison.

In summary, the reasons for the superior performance achieved by the proposed SIVNS are attributed:

- The single-individual algorithmic framework employed by the SIVNS effectively concentrates improvement efforts on a single solution, thereby significantly enhancing the local search ability of the algorithm.
- The group-based neighborhood search strategy, as introduced in this paper, addresses the challenges associated with grouping sorting and the high overhead of machine setup. It also effectively reduces the energy consumption associated with machine setup and idle times.
- The neighborhood search strategy designed to tackle job blocking and sorting greatly enhances the comprehension of the scheduling process and mitigates the impact of machine blocking and idling on energy consumption.

To assess the convergence of the algorithms across various instances, we randomly selected six instances of different scales: $100 \times 8 \times 40$, $160 \times 8 \times 20$, $200 \times 8 \times 20$, $240 \times 3 \times 40$, $260 \times 5 \times 60$, and $280 \times 5 \times 60$. For each instance, we plotted convergence curves

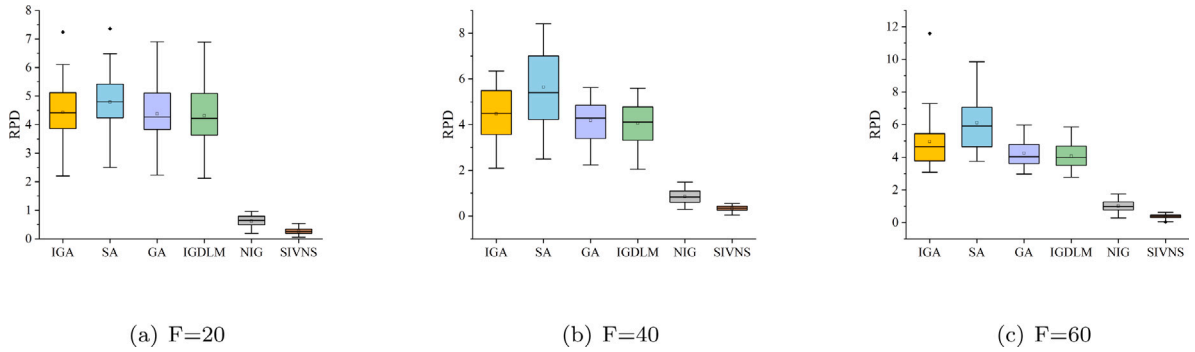


Fig. 3. RPD values of all algorithms, shown in the form of box plots.

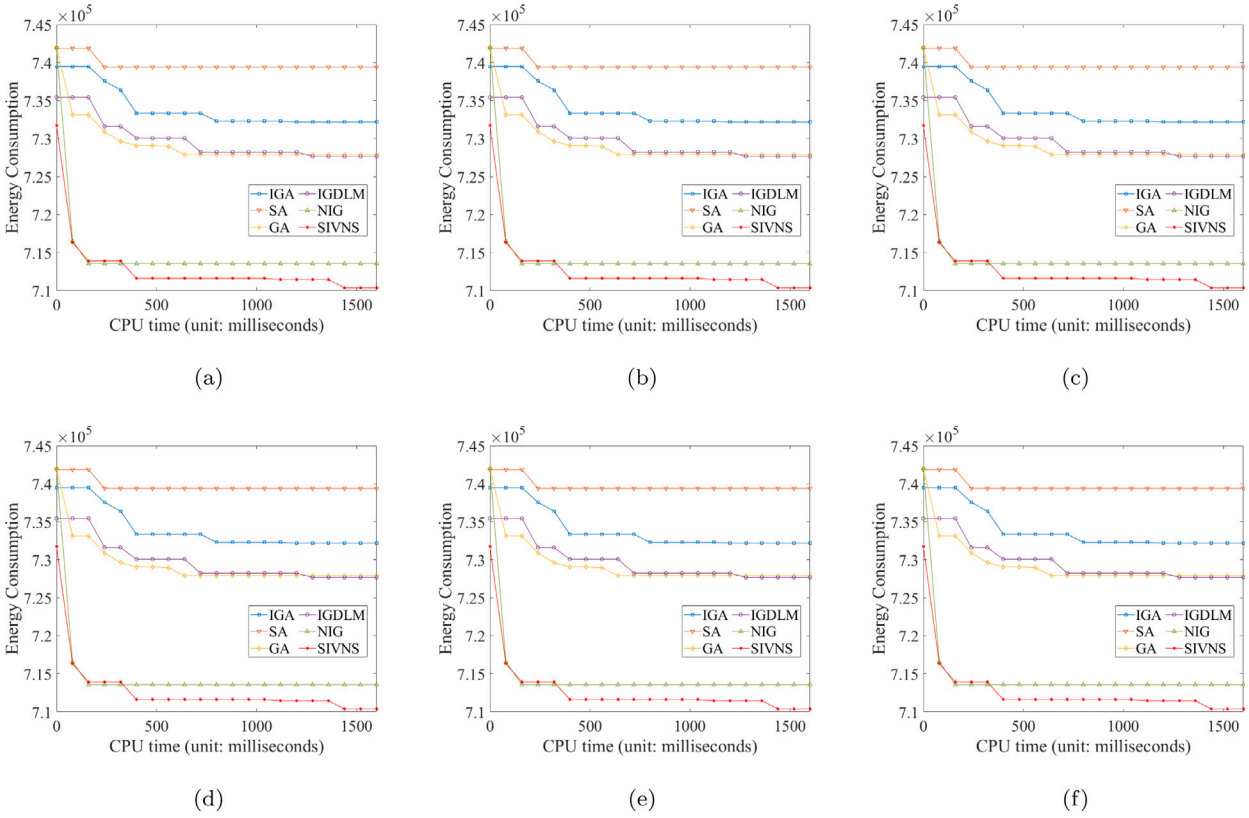


Fig. 4. The convergence curves of all these algorithms: (a) 100 × 8 × 40. (b) 160 × 8 × 20. (c) 200 × 8 × 20. (d) 240 × 3 × 40. (e) 260 × 5 × 60. (f) 280 × 5 × 60.

under identical termination conditions. At the commencement of each algorithmic iteration, we recorded the current objective function value and then tallied these values at regular time intervals. For instance, in the 100 × 8 × 40 instance, we divided the total time into 20 equal segments, using these segments as the units for the x-axis. The y-axis represents the range of objective function values. As illustrated by the six convergence plots in Fig. 4, the proposed SIVNS algorithm demonstrates superior performance in exploring the optimal solution compared to all other algorithms under comparison.

6. Conclusion

In this paper, we design a single-individual based variable neighborhood search algorithm to minimize the energy consumption of BHFGSP. Based on experimental and statistical analysis results, the proposed algorithm is proved to be very effective in solving the BHFGSP. In the proposed SIVNS, we propose a new decoding rule for BHFGSP with energy consumption as the optimization objective. Subsequently, a group-based neighborhood search strategy is designed to adjust the scheduling arrangement order of groups to further reduce the energy consumption of machine settings between different groups. Following this, a blocking job and job-based neighborhood search strategy is proposed to change the arrangement order of the job sequences to reduce the energy consumption caused by machine blocking and idling, and to further improve the productivity of the enterprise.

In the future, we will continue to design strategies based on the problem characteristics of BHFGSP. Similarly, we will study more real-world applications related to this problem. We will further consider more conditions such as distribution, processing time fuzzy, time window, and machine breakdown. In addition, some state-of-the-art machine learning techniques, such as reinforcement learning [73], etc., can be considered in combination with meta-heuristic algorithms.

CRediT authorship contribution statement

Zhongyuan Peng: Writing – original draft. **Haoxiang Qin:** Writing – original draft.

Declaration of competing interest

We declare that we have no personal relationships with other people or organizations that can inappropriately influence our work. We declare that we do not have any commercial or associative interest that represents a conflict of interest in connection with the work submitted.

References

[1] Wang Y-J, Wang G-G, Tian F-M, Gong D-W, Pedrycz W. Solving energy-efficient fuzzy hybrid flow-shop scheduling problem at a variable machine speed using an extended NSGA-II. Eng Appl Artif Intell 2023;121:105977. http://dx.doi.org/10.1016/j.engappai.2023.105977.

[2] Pan QK, Wang L, Mao K, Zhao JH, Zhang M. An effective artificial bee colony algorithm for a real-world hybrid flowshop problem in steelmaking process. IEEE Trans Autom Sci Eng 2013;10(2):307–22.

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

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

[5] Wang Y-J, Li J, Wang G-G. Fuzzy correlation entropy-based NSGA-II for energy-efficient hybrid flow-shop scheduling problem. Knowl-Based Syst 2023;277:110808. http://dx.doi.org/10.1016/j.knosys.2023.110808.

[6] Marichelvam MK, Prabaharan T, Yang XS. A discrete firefly algorithm for the multi-objective hybrid flowshop scheduling problems. IEEE Press; 2014.

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

[8] Fu Y, Zhou M, Guo X, Qi L. Scheduling dual-objective stochastic hybrid flow shop with deteriorating jobs via bi-population evolutionary algorithm. IEEE Trans Syst Man Cybern: Syst 2020;50(12):5037–48. http://dx.doi.org/10.1109/TSMC.2019.2907575.

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

[10] Riahi V, Newton M, Su K, Sattar A. Constraint guided accelerated search for mixed blocking permutation flowshop scheduling. Comput Oper Res 2018;102(FEB.):102–20.

[11] Ronconi DP. A note on constructive heuristics for the flowshop problem with blocking. Int J Prod Econ 2004;87(1):39–48.

[12] Li JQ, Pan QK, Mao K. A hybrid fruit fly optimization algorithm for the realistic hybrid flowshop rescheduling problem in steelmaking systems. IEEE Trans Autom Sci Eng 2016;932–49.

[13] Shao Z, Pi D, Shao W. A novel multi-objective discrete water wave optimization for solving multi-objective blocking flow-shop scheduling problem. Knowl-Based Syst 2019;165(FEB.1):110–31.

[14] Qin H, Wang Y, Han Y, Chen Q, Li J. Adapting a reinforcement learning method for the distributed blocking hybrid flow shop scheduling problem. In: 2021 5th Asian conference on artificial intelligence technology. 2021, p. 751–7. http://dx.doi.org/10.1109/ACAIT53529.2021.9731228.

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

[16] Zhao F, Zhang H, Wang L. A Pareto-based discrete jaya algorithm for multiobjective carbon-efficient distributed blocking flow shop scheduling problem. IEEE Trans Ind Inf 2023;19(8):8588–99. http://dx.doi.org/10.1109/TII.2022.3220860.

[17] Qin H-x, Han Y, Li J, Sang H, Chen Q, Meng L, et al. A quick and effective iterated greedy algorithm for energy-efficient hybrid flow shop scheduling problem with blocking constraint. In: 2021 11th international conference on information science and technology. 2021, p. 325–31. http://dx.doi.org/10.1109/ICIST52614.2021.9440648.

[18] Logendran R. Group scheduling problem: Key to flexible manufacturing systems. Comput Ind Eng 1992;23(1):113–6.

[19] Zhao F, Zhou G, Wang L. A cooperative scatter search with reinforcement learning mechanism for the distributed permutation flowshop scheduling problem with sequence-dependent setup times. IEEE Trans Syst Man Cybern: Syst 2023;53(8):4899–911. http://dx.doi.org/10.1109/TSMC.2023.3256484.

[20] Zhao F, Zhu B, Wang L. An estimation of distribution algorithm-based hyper-heuristic for the distributed assembly mixed no-idle permutation flowshop scheduling problem. IEEE Trans Syst Man Cybern: Syst 2023;53(9):5626–37. http://dx.doi.org/10.1109/TSMC.2023.3272311.

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

[22] Schaller JE, Gupta J, Vakharia AJ. Scheduling a flowline manufacturing cell with sequence dependent family setup times. European J Oper Res 2000;125(2):324–39.

[23] Wilson AD, King RE, Hodgson TJ. Scheduling non-similar groups on a flow line: multiple group setups. Robot Comput-Integr Manuf 2004;20(6):505–1513.

[24] Wang Y, Han Y, Wang Y, Tasgetiren MF, Li J, Gao K. Intelligent optimization under the makespan constraint: Rapid evaluation mechanisms based on the critical machine for the distributed flowshop group scheduling problem. European J Oper Res 2023;311(3):816–32.

[25] He X, Pan Q-k, Gao L, Wang L, Suganthan PN. 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.

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

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

[28] Qin H-X, Han Y-Y, Zhang B, Meng L-L, Liu Y-P, Pan Q-K, et al. An improved iterated greedy algorithm for the energy-efficient blocking hybrid flow shop scheduling problem. Swarm Evol Comput 2022;69:100992.

[29] Zhang B, Pan QK, Meng LL, Lu C, Mou JH, Li JQ. An automatic multi-objective evolutionary algorithm for the hybrid flowshop scheduling problem with consistent sublots. Knowl-Based Syst 2022;238:107819.

[30] Wang Y, Han Y, Wang Y, Pan Q-k, Wang L. Sustainable scheduling of distributed flow shop group: A collaborative multi-objective evolutionary algorithm driven by indicators. IEEE Trans Evol Comput 2023;1. http://dx.doi.org/10.1109/TEVC.2023.3339558.

[31] Wang Y, Han Y, ke Pan Q, Li H, Wang Y. Redefining hybrid flow shop group scheduling: Unveiling a novel hybrid modeling paradigm and assessing 48 MILP and CP models. Swarm Evol Comput 2023;83:101416. http://dx.doi.org/10.1016/j.swevo.2023.101416.

[32] Qin H-X, Han Y-Y, Liu Y-P, Li J-Q, Pan Q-K, Xue-Han. A collaborative iterative greedy algorithm for the scheduling of distributed heterogeneous hybrid flow shop with blocking constraints. Expert Syst Appl 2022;201:117256.

[33] Li J-q, Pan Q-k, Duan P-y. An improved artificial bee colony algorithm for solving hybrid flexible flowshop with dynamic operation skipping. IEEE Trans Cybern 2016;46(6):1311–24. http://dx.doi.org/10.1109/TCYB.2015.2444383.

[34] Wang J-j, Wang L. A bi-population cooperative memetic algorithm for distributed hybrid flow-shop scheduling. IEEE Trans Emerg Top Comput Intell 2021;5(6):947–61. http://dx.doi.org/10.1109/TETCI.2020.3022372.

[35] Fan J, Li Y, Xie J, Zhang C, Shen W, Gao L. A hybrid evolutionary algorithm using two solution representations for hybrid flow-shop scheduling problem. IEEE Trans Cybern 2023;53(3):1752–64. http://dx.doi.org/10.1109/TCYB.2021.3120875.

[36] Shao W, Shao Z, Pi D. An ant colony optimization behavior-based MOEA/D for distributed heterogeneous hybrid flow shop scheduling problem under nonidentical time-of-use electricity tariffs. IEEE Trans Autom Sci Eng 2022;19(4):3379–94. http://dx.doi.org/10.1109/TASE.2021.3119353.

[37] Tang L, Wang X. 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 2010;18(6):1303–14.

[38] Pan QK, Wang L, Li JQ, Duan JH. A novel discrete artificial bee colony algorithm for the hybrid flowshop scheduling problem with makespan minimisation. Omega 2014;45(jun.):42–56.

[39] Chen F, Luo C, Gong W, Lu C. Two-stage adaptive memetic algorithm with surprisingly popular mechanism for energy-aware distributed hybrid flow shop scheduling problem with sequence-dependent setup time. Complex Syst Model Simul 2024;4(1):82–108. http://dx.doi.org/10.23919/CSMS.2024.0003.

[40] Li J-Q, Chen X-L, Duan P-Y, Mou J-H. KMOEA: A knowledge-based multiobjective algorithm for distributed hybrid flow shop in a prefabricated system. IEEE Trans Ind Inf 2022;18(8):5318–29. http://dx.doi.org/10.1109/TII.2021.3128405.

[41] Zhang B, Pan Q-K, Gao L, Meng L-L, Li X-Y, Peng K-K. A three-stage multiobjective approach based on decomposition for an energy-efficient hybrid flow shop scheduling problem. IEEE Trans Syst Man Cybern: Syst 2020;50(12):4984–99. http://dx.doi.org/10.1109/TSMC.2019.2916088.

[42] Trabelsi W, Sauvey C, Sauer N. Mathematical model and lower bound for hybrid flowshop problem with mixed blocking constraints. IFAC Proc Vol 2012;45(6):1475–80.

[43] Mollaei A, Mohammadi M, Naderi B. A bi-objective MILP model for blocking hybrid flexible flow shop scheduling problem: Robust possibilistic programming approach. Int J Manag Sci Eng Manag 2018;14:137–46, URL https://api.semanticscholar.org/CorpusID:69511921.

[44] Leilei M. MILP models and an improved BSA for hybrid flow shop scheduling problems with blocking. China Mech Eng 2018;29(22):2647–58.

[45] Qin HX, Han YY, Chen QD, Li JQ, Sang HY. 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.

[46] Elmi A, Topaloglu S. A scheduling problem in blocking hybrid flow shop robotic cells with multiple robots. Comput Oper Res 2013;40(10):2543–55.

[47] Nakkaew P, Kantanantha N, Wongthatsanekorn W. 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 2016;43(2).
[48] Qin H, Han Y, Chen Q, Wang L, Wang Y, Li J, et al. Energy-efficient iterative greedy algorithm for the distributed hybrid flow shop scheduling with blocking constraints. IEEE Trans Emerg Top Comput Intell 2023;7(5):1442–57. http://dx.doi.org/10.1109/TETCI.2023.3271331.
[49] Missaoui A, Boujelbene Y. An effective iterated greedy algorithm for blocking hybrid flow shop problem with due date window. RAIRO - Oper Res 2021;55(3):1603–16.
[50] Neufeld JS, Gupta JND, Buscher U. Minimising makespan in flowshop group scheduling with sequence-dependent family set-up times using inserted idle times. Int J Prod Res 2015;53(6):1791–806.
[51] Neufeld JS, Gupta J, Buscher U. A comprehensive review of flowshop group scheduling literature. Comput Oper Res 2016;70(Jun.):56–74.
[52] Lin SW, Ying KC. Makespan optimization in a no-wait flowline manufacturing cell with sequence-dependent family setup times. Comput Ind Eng 2019;128(FEB.):1–7.
[53] Baker K. Scheduling groups of jobs in the two-machine flow shop. Math Comput Modelling 1990;13(3):29–36. http://dx.doi.org/10.1016/0895-7177(90)90368-W.
[54] Ghorbanzadeh M, Ranjbar M. Energy-aware production scheduling in the flow shop environment under sequence-dependent setup times, group scheduling and renewable energy constraints. European J Oper Res 2023;307(2):519–37. http://dx.doi.org/10.1016/j.ejor.2022.09.034.
[55] Feng H, Xi L, Xiao L, Xia T, Pan E. Imperfect preventive maintenance optimization for flexible flowshop manufacturing cells considering sequence-dependent group scheduling. Reliab Eng Syst Saf 2018;176(aug.):218–29.
[56] Costa A, Cappadonna FA, Fichera S. A hybrid genetic algorithm for minimizing makespan in a flow-shop sequence-dependent group scheduling problem. J Intell Manuf 2017.
[57] Costa A, Cappadonna FV, Fichera S. Minimizing makespan in a flow shop sequence dependent group scheduling problem with blocking constraint. Eng Appl Artif Intell 2020;89(Mar.):103413.1–103413.15.
[58] Qin H, Han Y, Wang Y, Liu Y, Li J, Pan Q. Intelligent optimization under blocking constraints: A novel iterated greedy algorithm for the hybrid flow shop group scheduling problem. Knowl-Based Syst 2022;258:109962. http://dx.doi.org/10.1016/j.knosys.2022.109962.
[59] Wang JJ, Wang L. 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.
[60] Fernandez-Viagas V, Perez-Gonzalez P, Framinan JM. Efficiency of the solution representations for the hybrid flow shop scheduling problem with makespan objective. Comput Oper Res 2019;109(SEP.):77–88.
[61] Li R, Gong W, Wang L, Lu C, Dong C. Co-evolution with deep reinforcement learning for energy-aware distributed heterogeneous flexible job shop scheduling. IEEE Trans Syst Man Cybern: Syst 2024;54(1):201–11. http://dx.doi.org/10.1109/TSMC.2023.3305541.
[62] Li R, Gong W, Wang L, Lu C, Zhuang X. Surprisingly popular-based adaptive memetic algorithm for energy-efficient distributed flexible job shop scheduling. IEEE Trans Cybern 2023;53(12):8013–23. http://dx.doi.org/10.1109/TCYB.2023.3280175.
[63] Ruiz R, Stützle T. A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. European J Oper Res 2007;177(3):2033–49.
[64] Ham JI. A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. Omega 1983.
[65] Qin H-X, Han Y-Y, Zhang B, Meng L-L, Liu Y-P, Pan Q-K, et al. An improved iterated greedy algorithm for the energy-efficient blocking hybrid flow shop scheduling problem. Swarm Evol Comput 2022;69:100992.
[66] Zhang B, Pan QK, Gao L, Zhang XL, Sang HY, Li JQ. An effective modified migrating birds optimization for hybrid flowshop scheduling problem with lot streaming. Appl Soft Comput 2017;52:14–27.
[67] Qin H, Bai W, Xiang Y, Liu F, Han Y, Wang L. A self-adaptive collaborative differential evolution algorithm for solving energy resource management problems in smart grids. IEEE Trans Evol Comput 2023;1. http://dx.doi.org/10.1109/TEVC.2023.3312769.
[68] Huang J-P, Pan Q-K, Gao L. An effective iterated greedy method for the distributed permutation flowshop scheduling problem with sequence-dependent setup times. Swarm Evol Comput 2020;59:100742.
[69] Li Y, Li X, Gao L, Meng L. An improved artificial bee colony algorithm for distributed heterogeneous hybrid flowshop scheduling problem with sequence-dependent setup times. Comput Ind Eng 2020;147:106638. http://dx.doi.org/10.1016/j.cie.2020.106638.
[70] Meng T, Pan QK. A distributed heterogeneous permutation flowshop scheduling problem with lot-streaming and carryover sequence-dependent setup time. Swarm Evol Comput 2021;60:100804.
[71] Aqil S, Allali K. Local search metaheuristic for solving hybrid flow shop problem in slabs and beams manufacturing. Expert Syst Appl 2020;162:113716.
[72] Meng L, Zhang C, Shao X, Ren Y, Ren C. Mathematical modelling and optimisation of energy-conscious hybrid flow shop scheduling problem with unrelated parallel machines. Int J Prod Res 2019;57(4):1119–45. http://dx.doi.org/10.1080/00207543.2018.1501166.
[73] Zhao F, Zhuang C, Wang L, Dong C. An iterative greedy algorithm with Q-learning mechanism for the multiobjective distributed no-idle permutation flowshop scheduling. IEEE Trans Syst Man Cybern: Syst 2024;54(5):3207–19. http://dx.doi.org/10.1109/TSMC.2024.3358383.