Applied Soft Computing 129 (2022) 109502

Contents lists available at ScienceDirect

Applied Soft Computing

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

# An effective iterative greedy algorithm for distributed blocking flowshop scheduling problem with balanced energy costs criterion

Xue Han [a], Yuyan Han [a,*], Biao Zhang [a,*], Haoxiang Qin [a], Junqing Li [b], Yiping Liu [c], Dunwei Gong [d]

[a] School of Computer Science, Liaocheng University, Liaocheng, 252000, China
[b] School of Computer Science, Shandong Normal University, Jinan, 252000, China
[c] The College of Computer Science and Electronic Engineering, Hunan University, 410082, China
[d] School of Information and Control Engineering, China University of Mining and Technology, Xuzhou 221116, China

ARTICLE INFO

*Article history:*
Received 1 December 2021
Received in revised form 2 August 2022
Accepted 9 August 2022
Available online 20 August 2022

*Keywords:*
Distributed flowshop scheduling problem
Blocking
Energy consumption cost
Local search algorithm
Variable neighborhood search

ABSTRACT

With the increase in production levels, a pattern of industrial production has shifted from a single factory to multiple factories, resulting in a distributed production model. The distributed flowshop scheduling problem (DPFSP) is of great research significance as a frequent pattern in real production activities. In this paper, according to real-world scenarios, we have added blocking constraints and sequence-dependent setup times (SDST) to the DFSP and proposed a distributed blocking flowshop scheduling problem with sequence-dependent setup times (DBFSP_SDST). In a distributed environment, the allocation of resources and utilization have become an urgent problem to be solved. In addition, scheduling problems related to resource conservation have also attracted increasing attention. Therefore, we study DBFSP_SDST and consider minimizing the energy consumption cost of the critical factory (critical factory is the factory with maximum energy consumption cost) under resource balance. To tackle this problem, an effective iterated greedy algorithm based on a learning-based variable neighborhood search algorithm (VNIG) is proposed. In VNIG, an efficient construction heuristic is well designed. Two different local searches based on the characteristics of the proposed problem are developed to enhance the local exploitation by neighborhood searching. A learning-based selection variable neighborhood search strategy is designed to avoid the solution trapping in local optima. By conducting extensive simulation experiments, the proposed VNIG shows superior performance compared with artificial chemical reaction optimization (CRO, 2017), the discrete artificial bee colony algorithm (DABC, 2018), the iterative greedy algorithm with a variable neighborhood search scheme (IGR, 2021), and the evolution strategy approach (ES, 2022).

© 2022 Elsevier B.V. All rights reserved.

## 1. Introduction

With the rapid growth of manufacturing, the production model of multiple factory operations has become the norm. Moreover, the emergence of globalization has also prompted factories to show multiregional, multimode characteristics [1]. Product processing is no longer limited to a factory but several factories, which makes processing much more efficient. The distributed permutation flowshop scheduling problem (DPFSP) has become the choice of many enterprises due to its simple structure and easy operation and has attracted much attention from scholars [2]. The characteristics of the DPFSP are as follows: there are $n$ jobs to be processed in $f$ identical factories, in which $m$ uncorrelated machines exist in each factory. Once a job is assigned to a factory, it cannot be assigned to another factory. The job must be processed on all machines of the factory. Based on the above characteristics, there are two key subproblems in the DPFSP to be solved: how to assign jobs to multiple factories and how to determine the scheduling sequence in each factory.

In the traditional DPFSP, factories are assumed to have infinite buffers. However, in real production activities, there are sometimes no buffers between neighboring machines due to the limitations of factory storage devices [3]. In this situation, a job must wait on the current machine until the next machine is available, which causes the blocking of jobs. For example, in the chemical industry, partially processed jobs (i.e., physical or chemical materials) are held in machines because there is no intermediate storage. In the case of the iron and steel industry, the blocking of ingot in the soaking pit will increase the extra consumption of energy since the blocked ingot requires a

* Corresponding authors.
*E-mail addresses:* hanyuyan@lcu-cs.com (Y. Han), zhangbiao@lcu-cs.com (B. Zhang).

https://doi.org/10.1016/j.asoc.2022.109502
1568-4946/© 2022 Elsevier B.V. All rights reserved.

high temperature [3]. In this paper, the blocking constraint is considered in the DPFSP, resulting in a new problem, i.e., the distributed blocking flowshop scheduling problem (DBFSP). Except for the above blocking constraint, the setup time of adjacent jobs is also an important factor in real-world scenarios [4], such as fixtures or tools of machines that may need to be changed when different jobs are processed on the same machine. The abovementioned setup time is related to the operation being processed and the previous operation in the sequence, generally called sequence-dependent setup times (SDST) [5,6]. According to the literature [7,8], both blocking constraints and SDST have been considered simultaneously in PFSP, and it is evident that the study of these two constraints on the flowshop scheduling problem is of practical importance. Therefore, based on the above analysis, combined with the distributed production environment, we investigate the distributed blocking flowshop scheduling problem with sequence-dependent setup time (DBFSP_SDST).

In the literature, most research on the above distributed flowshop scheduling problems only considers economic indicators, such as makespan, tardiness time or earliness time, and relatively few focus on environmental protection or energy consumption or energy consumption cost indicators from a sustainable manufacturing point of view. In practical production, idle machines, setup of machines, and blocking of jobs in each factory often lead to energy consumption csots [3]. For distributed production models, the factories may be located in different regions, which leads to the imbalanced reginons energy consumption csots. In the report of the Academician Conference of the Chinese Academy of Sciences in 2021, professor Zhongli Ding, Academician of the Chinese Academy of Sciences, pointed out that if different enterprises in a certain industry cannot coordinate and progress together, it will inevitably lead to cost savings for "doing not act as an enterprise", resulting in the phenomenon of "bad money drives out good money". Furthermore, efforts have also be made on minimizing the total energy consumption costs by many scholars [9,10]. Based on the above analysis, we find that it is more reasonable to optimize the total energy consumption costs of the critical factory than the total energy consumption of the critical factory.

Thus, it is necessary to expend much effort to ensure the energy costs balance between different regions and reduce the energy consumption costs of each factory. In view of this, in this paper, the objective is to minimize the total energy consumption costs of each critical factory by assigning jobs to the factories and reasonably scheduling sequence in each factory.

For flowshop scheduling problems, the iterated greedy algorithm (IG) shows great performance [11,12]. Compared with some swarm intelligence algorithms, IG has the characteristics of a simple frame, easy operations, and a powerful neighborhood search capability for deeper mining of solutions [13]. For the DBFSP_SDST with balanced energy costs criterion, there is no relevant literature to propose corresponding algorithms to solve this problem. Therefore, an expanded algorithm based on the IG algorithm is proposed in this paper.

This study has the following two novelties.

Based on the scheduling problem subject to blocking constraints. This study formulates a distributed blocking flowshop scheduling problem with setup times that optimize the balanced energy costs criterion for the first time. The formulation of the above scheduling problem can better reflect real-world applications, thus, compared to those in previous work, that have more practical significance.

Following that, an IG algorithm with variable neighborhoods (VNIG) is designed to solve the DBFSP_SDST. The proposed algorithm has the following trifold features. The first is that the initialization solution is obtained by a variant of MM and NEH2_en. The second is that three different local search strategies are proposed based on the properties of DBFSP_SDST with balanced energy costs criterion. Third, a variable neighborhood search strategy is added to avoid becoming trapped in local optima. The performances of the presented initialization, three local search, and variable neighborhood search strategies are empirically evaluated. The experimental results demonstrate that the proposed strategies can effectively tackle DBFSP_SDST with balanced energy costs criterion by obtaining a good scheduling sequence.

The remainder of this paper is organized as follows. Section 2 describes the literature related to energy-balance DBFSP_SDST. In Section 3, we explain and give examples of DBFSP_SDST with balanced energy costs criterion. The algorithms and innovations proposed in this paper are listed in Section 4. Section 5 gives the experimental results and analysis of the algorithms. In the last section, we conclude the paper and give an outlook on future research directions.

## 2. Literature review

Compared with the research on the classical DPFSP, little research has been done on the DFSP_SDST with blocking constraints and balanced energy costs criterion. Most studies have focused on optimizing the DPFSP or DPFSP_SDST with the makespan criterion. The following first reviews the DPFSP and then the DPFSP_SDST. Next, the blocking constraint in the flowshop scheduling problems is described. Finally, the characteristics of our addressed problem are presented.

The DPFSP has emerged in response to globalization and the continuous development of the manufacturing industry. The target of studying DPFSP is to save production materials and improve production efficiency. Since the emergence of the DPFSP, many scholars have developed a series of innovative algorithms to solve it. Heuristics and metaheuristics are studied by Hatami and Ruiz [14]. Naderi and Ruiz proposed a scatter search (SS) method to minimize makespan [15]. For the same objective, Bargaoui et al. designed an effectively improved chemical reaction optimization algorithm (CRO) [16]. Fernandez-Viagas et al. employed a bounded-search iterated greedy algorithm based on the specific structure of the DPFSP [17]. Ruiz and Pan proposed an improved iterative greedy algorithm that showed excellent performance in solving the DPFSP [18]. In addition to minimizing makespan, scholars have also studied the DPFSP with other objectives. Meng and Pan proposed three heuristics to optimize the customer satisfaction objective, i.e., the neighborhood descent, the artificial bee colony, and the iterative greedy methods [19]. For the DPFSP with the total flow time criterion, Fernandez-Viagas et al. proposed an iterative improvement algorithm [20]. Subsequently, Pan et al. designed three constructive heuristics and four metaheuristics based on the characteristics of the problem [21]. Recently, Zhang et al. proposed an innovative three-dimensional matrix-cube-based estimation of distribution algorithm (MCEDA) [22].

In real production activities of the DPFSP, many constraints limit the production of products, e.g., sequence-dependent setup time and blocking constraints. Regarding the sequence-dependent setup time, since the machine may generate operations with replacement parts, maintenance, etc., before processing a job, some extra time will be generated. The setup time is related to the job being machined as well as to the previous one. To solve this problem, Parthasarathy et al. [23] proposed an experimental evaluation of heuristics for scheduling in a real-life flowshop with sequence-dependent setup times (SDST) of jobs. Following that, to optimize the problems with SDST, some excellent algorithms, such as ant colony optimization techniques [24], variable domain constructive heuristics [25], constructive heuristics [26], and enhanced migratory bird optimization algorithms [27], have been proposed. Recently, Huang

**Table 1**
Review of the works on DPFSP.

| Authors | Setting | Objective(Minimizing) | Solution approach |
|---|---|---|---|
| Bargaoui et al. [16] | DPFSP | Makespan | Improved artificial chemical reaction optimization |
| Ruiz et al. [18] | DPFSP | Makespan | Effective Iterated Greedy methods |
| Huang et al. [28] | DPFSP | Setup times Constraint and makespan | An iterated greedy algorithm with a restart scheme |
| Zhao et al. [32] | DBFSP | Makespan | An ensemble discrete differential evolution algorithm |
| Chen et al. [33] | DBFSP | Makespan | An iterated greedy algorithm, an effective initialization method, and an enhanced construction method |
| Wang et al. [34] | DPFSP | Energy- Efficient and makespan | A knowledge-based cooperative algorithm |
| Rossi et al. [35] | Mixed No-idle DPFSP | Sequence-Dependent setup times constraint | A novel constructive heuristic and iterated greedy algorithms |
| Pan et al. [36] | DPFSP | Total flow time | A discrete artificial bee colony algorithm |
| This paper | DBFSP | Setup times constraint and Energy-balance | An effective iterative greedy with variable neighborhood search strategy |

et al. proposed an improved iterative greedy algorithm [28] and an effective discrete bee colony algorithm [29] to solve the DPFSP with SDST

The DPFSP can be classified into two categories concerning the size of buffers (either infinite or no buffers). The former does not result in job blocking since it has enough intermediate buffers to store uncompleted jobs. Here, the term "blocking" means maintaining a limited capacity of in-process inventories due to finite intermediate buffers. Once blocking occurs, it affects the overall production efficiency of the sequence and increases energy consumption. Therefore, determining job scheduling under blocking constraints becomes very important. To solve the blocking constraint, many researchers have proposed intelligent optimization algorithms. For the DPFSP_SDST, blocking time often arises in each factory due to no storage space between adjacent machines. In this paper, the blocking constraint is considered in DPFSP_SDST, and a new problem, called the distributed blocking flowshop scheduling problem with setup times (DBFSP_SDST), is formed. To solve the above DBFSP_SDST, Zhang et al. proposed a novel hybrid discrete differential evolution (DDE) algorithm for the DBFSP [30]. Next, a hybrid enhanced discrete fruit fly optimization algorithm (HEDFOA) was proposed by Shao [31]. Later, an ensemble discrete differential evolution [32] and an iterated greedy (IG) algorithm [33] were proposed.

Due to the gradual emphasis on resource conservation and environmental protection, research with the goal of green manufacturing has received increasing attention. Scholars have mostly presented their research on energy consumption objectives [37–42]. In recent years, for the energy-efficiency DPFSP, Wang et al. proposed a knowledge-based cooperative algorithm (KCA) [34]. Wang and Li studied the multiobjective whale swarm algorithm (MOWSA) [43]. Then, the collaborative optimization algorithm (COA) [44], the innovative 3D matrix cube distribution estimation algorithm (MCEDA) [22], the genetic programming hyperheuristic (GP-HH) algorithm [45], and the improved NSGAII algorithm (INSGAII) [46] were continuously designed and proposed to minimize the energy consumption objective in different research fields. For the distributed models, there are differences in the level of development between regions. To narrow the differences between factories and improve the utilization rate of factories, the resource balance between factories was considered for the first time.

The DBFSP_SDST with balanced energy costs criterion is first considered in this paper; therefore, some literature related to the DPFSP (see Table 1) must be described. Table 1 includes the characteristics and the main contributions of the DPFSP. For the DPFSP with makespan criteria, a improved iterated greedy algorithm [16] and CRO algorithm [18] were proposed. The work in [28] studied the DPFSP with makespan criteria and proposed an effective iterated greedy algorithm with a restart scheme (IGR). The works in [32,33] added the blocking constraint in DPFSP and proposed an ensemble discrete differential evolution (EDE) algorithm and an iterated greedy (IG) algorithm. In [34], Wang et al. considered the DPFSP with energy consumption and makespan criteria. The work in [35] addressed mixed no-idle DPFSP and considered sequence-dependent setup times. In [36], an effective discrete artificial bee colony algorithm is proposed to solve the DPFSP with total flowtime minimization.

In summary, for the above literature, we found that research on the DPFSP problem is a current hot topic. DPFSP is a frequent pattern in real production activities and has great research significance. However, the situation faced in real production activities is more complex [47]. To better approach real production activities, we consider the abovementioned constraints related to energy consumption cost indicators in the DPFSP and propose a simple and effective IG algorithm with a variable neighborhood (VNIG) for solving the above DBFSP_SDST with balanced energy costs criterion.

## 3. DBFSP_SDST with balanced energy costs criterion

In this section, the mathematical model of the DBFSP_SDST with balanced energy costs criterion is described, and we assume that there are $J$ jobs to be scheduled in $F$ factories that have $M$ machines. In addition, some restrictions may appear in the model: (1) A job can only be processed on one machine at a time. (2) Machines in a factory can only process one job at a time. (3) Each job must be scheduled in process order and cannot change the factory. (4) A job has to be blocked in the current machine until the downstream one is available. (5) An anticipatory and sequence-dependent job setup time is considered on each machine, and an initial setup time is needed if job $j$ is the first job on a machine. (6) The total energy consumption costs are the sum of the processing energy consumption cost, setup energy consumption cost, and standby energy consumption cost.

In this paper, the objective is to minimize the total energy consumption costs of each critical factory and balance the resources by assigning jobs to the factories and reasonably scheduling sequence in each factory. The calculation method and the related notations, decision variables, objective, and constraints in the proposed mathematical model are given below.

**Notations:**
$F$: The number of factories.
$f$: Index of factories, $f \in \{1, 2, \ldots, F\}$.
$M$: The number of machines in each factory.
$m$: Index of machines.
$J$: The number of jobs.
$j, j'$: Index of jobs, $j, j' \in \{0, 1, \ldots, J\}$, where 0 is the index of the dummy job, which represents the start and end of the job sequence in a factory.

$p_{j,m}$: Processing time of job $j$ on machine $m$.

$s_{j,j',m}$: Setup time from job $j$ to job $j'$ on machine $m$. An initial setup time $s_{0,j,m}$ is needed if job $j$ is the first job on machine $m$.

$EC_{j,m}^{Process}$: The energy consumption per unit time of machine $m$ when the machine processes job $j$.

$EC_{j,j',m}^{Setup}$: The energy consumption per unit time of machine $m$ when the machine stays in the setup state from job $j$ to job $j'$. When machine $m$ stays in the setup state for the first job $j$, its energy consumption per unit time is represented by $EC_{0,j,m}^{Setup}$.

$EC_{m}^{Idle}$: The energy consumption per unit time of machine $m$ when the machine stays in the idle or blocked state.

$ECC_f$: The cost per unit of energy consumption in factory $f$.

$h$: Sufficiently large positive number.

**Decision variables:**

$C_{j,m}$: The completion time of job $j$ on machine $m$.

$D_{j,m}$: The departure time of job $j$ on machine $m$.

$w_{j,f}$: Binary decision variable, 1 if job $j$ is assigned to factory $f$, 0 otherwise.

$x_{j,j',f}$: Binary decision variable, 1 if both jobs $j$ and $j'$ are assigned to factory $f$, and job $j'$ is an immediate successor of job $j$ in factory $f$, 0 otherwise.

$TPE_f$: Total energy consumption costs of all machines in factory $f$ when they stay in the processing state.

$TSE_f$: Total energy consumption costs of all machines in factory $f$ when they stay in the setup state.

$TIE_f$: Total energy consumption costs of all machines in factory $f$ when they stay in the idle or blocked state.

$EMAX$: Total energy consumption costs of all machines in each critical factory.

**Objective:**

$$Minimize\ (EMAX) \tag{1}$$

**Constraints:**

$$w_{0,f} = 1, \forall f \in \{1, 2, \ldots, F\} \tag{2}$$

$$\sum_{f=1}^{F} w_{j,f} = 1, \forall j \in \{1, 2, \ldots, J\} \tag{3}$$

$$\sum_{j'=0, j' \neq j}^{J} x_{j,j',f} = w_{j,f}, \forall j \in \{1, 2, \ldots, J\}, \forall f \in \{1, 2, \ldots, F\} \tag{4}$$

$$\sum_{j=0, j \neq j'}^{J} x_{j,j',f} = w_{j',f}, \forall j' \in \{1, 2 \ldots, J\}, \forall f \in \{1, 2, \ldots, F\} \tag{5}$$

$$\sum_{j'=0}^{J} x_{0,j',f} = 1, \forall f \in \{1, 2, \ldots, F\} \tag{6}$$

$$\sum_{j=0}^{J} x_{j,0,f} = 1, \forall f \in \{1, 2, \ldots, F\} \tag{7}$$

$$C_{j,m} - p_{j,m} \geqslant 0, \forall j \in \{1, 2, \ldots, J\}, \forall m \in \{1, 2, \ldots, M\} \tag{8}$$

$$D_{j,m} \geqslant C_{j,m}, \forall j \in \{1, 2, \ldots, J\}, \forall m \in \{1, 2, \ldots, M\} \tag{9}$$

$$C_{j,m} - p_{j,m} = D_{j,m-1}, \forall j \in \{1, 2, \ldots, J\}, \forall m \in \{2, 3, \ldots, M\} \tag{10}$$

$$C_{j',m} - p_{j',m} \geqslant D_{j,m} + s_{j,j',m} + \left(\sum_{f=1}^{F} x_{j,j',f} - 1\right) \cdot h, \forall j, j' \in \{1, 2, \ldots, J\}, j \neq j', \forall m \in \{1, 2, \ldots, M\} \tag{11}$$

$$C_{j,m} - p_{j,m} \geqslant s_{0,j,m} + \left(\sum_{f=1}^{F} x_{0,j,f} - 1\right) \cdot h, \forall j \in \{1, 2, \ldots, J\}, \forall m \in \{1, 2, \ldots, M\} \tag{12}$$

$$TPE_f = ECC_f \cdot \sum_{m=1}^{M} \sum_{j=1}^{J} \left(EC_{j,m}^{Process} \cdot p_{j,m} \cdot w_{j,f}\right), \forall f \in \{1, 2, \ldots, F\} \tag{13}$$

$$TSE_f = ECC_f \cdot \sum_{m=1}^{M} \sum_{j=1}^{J} \left( \sum_{j'=1, j' \neq j}^{J} \left(EC_{j,j',m}^{Setup} \cdot s_{j,j',m} \cdot x_{j,j',f}\right) + EC_{0,j,m}^{Setup} \cdot s_{0,j,m} \cdot x_{0,j,f} \right), \forall f \in \{1, 2, \ldots, F\} \tag{14}$$

$$TIE_f = ECC_f \cdot \sum_{m=1}^{M} \left( EC_{m}^{Idle} \cdot \left( \sum_{j=1}^{J} \left(C_{j,M} \cdot x_{j,0,f}\right) - \sum_{j=1}^{J} p_{j,m} \cdot w_{j,f} - \sum_{j=1}^{J} \left( \sum_{j'=1, j' \neq j}^{J} \left(s_{j,j',m} \cdot x_{j,j',f}\right) + s_{0,j,m} \cdot x_{0,j,f} \right) \right) \right), \forall f \in \{1, 2, \ldots, F\} \tag{15}$$

$$EMAX \geqslant TPE_f + TSE_f + TIE_f, \forall f \in \{1, 2, \ldots, F\} \tag{16}$$

Eq. (1) minimizes the energy consumption costs of each critical factory, which can balance the energy consumption costs of each factory. Constraint (2) defines that each factory contains a dummy job that represents the start and end of the job sequence in a factory. Constraint (3) guarantees that each job can only be assigned to one factory for processing. Constraints (4), (5), (6), and (7) ensure that each job must have only one immediate predecessor and successor. Constraint (8) enforces that the start processing time of each job on each machine must be greater than or equal to 0. Constraint (9) ensures that the departure time of each job on each machine must be greater than or equal to its completion time. Constraint (10) defines that the start processing time of each job in a machine is equal to its departure time on the previous machine. For job $j$ and its immediate successor $j'$ on machine $m$ in factory $f$, the start processing time of job $j'$ on machine $m$ is not less than the departure time of job $j$ on machine $m$ plus the setup time $s_{j,j',m}$, which is ensured by constraint (11). For the first job on machine $m$ in factory $f$, the start processing time must be greater than or equal to the initial setup time $s_{0,j,m}$ and is considered by constraint (12). Constraints (13), (14), and (15) calculate the total processing energy consumption costs of each factory, the total setup energy consumption costs of each factory, and the total standby energy consumption costs of each factory, respectively. Constraint (16) defines the total energy consumption costs of all machines in each critical factory. Our model contains $F$ job sequences, each starts from a dummy job and ends with another dummy job and represents the job scheduling in a factory.

To clearly understand the processing of calculating EMAX, we give the equations of the above model to calculate EMAX in the case of heuristics and metaheuristics. Suppose factory $f$ includes $\delta_f$ jobs that are processed according to job sequence $\pi_f = \left\{\pi_f^1, \pi_f^2, \ldots, \pi_f^j, \ldots, \pi_f^{\delta_f}\right\}$, where $\pi_f^j, j \in \{1, 2, \ldots, \delta_f\}$, is the job included in factory $f$. Denote $[f, j]$ as the job index of the $j$th job to be processed in factory $f$. For factory $f, f = 1, 2, \ldots, F$, the completion time and departure time of each job on each machine are calculated according to Eqs. (17)–(19). The energy consumptions costs $TPE_f$, $TSE_f$ and $TIE_f$ of factory $f$ are calculated according to Eqs. (20)–(22). Eq. (23) give the expression of $EMAX$.

$$C_{[f,j],0} = 0, j = 1, 2, \ldots, \delta_f \tag{17}$$

**Table 2**
Processing time of each job in different machines.

| Job | Machine | | |
|---|---|---|---|
| | 1 | 2 | 3 |
| 1 | 4 | 3 | 2 |
| 2 | 2 | 7 | 4 |
| 3 | 1 | 2 | 2 |
| 4 | 3 | 1 | 1 |
| 5 | 6 | 9 | 2 |
| 6 | 8 | 4 | 3 |
| 7 | 1 | 2 | 2 |
| 8 | 3 | 1 | 1 |
| 9 | 6 | 9 | 2 |

**Table 3**
Unit processing energy consumption of each job in different machines.

| Job | Machine | | |
|---|---|---|---|
| | 1 | 2 | 3 |
| 1 | 4 | 1 | 3 |
| 2 | 2 | 3 | 1 |
| 3 | 4 | 3 | 2 |
| 4 | 3 | 1 | 3 |
| 5 | 6 | 9 | 2 |
| 6 | 8 | 4 | 5 |
| 7 | 5 | 3 | 2 |
| 8 | 2 | 1 | 3 |
| 9 | 1 | 3 | 1 |

$$C_{[f,j],m} = \begin{cases} max\left(s_{0,[f,j],m}, C_{[f,j],m-1}\right) \\ +p_{[f,j],m}, j = 1, m = 1, 2, \ldots, M \\ max\left(D_{[f,j-1],m} + s_{[f,j-1],[f,j],m}, C_{[f,j],m-1}\right) \\ +p_{[f,j],m}, j = 2, 3, \ldots, \delta_f, \\ m = 1, 2, \ldots, M \end{cases} \tag{18}$$

$$D_{[f,j],m} = \begin{cases} max\left(C_{[f,j],m}, C_{[f,j],m+1} - p_{[f,j],m+1}\right), \\ j = 1, 2, \ldots, \delta_f, m = 1, 2, \ldots, M-1 \\ C_{[f,j],m}, j = 1, 2, \ldots, \delta_f, m = M \end{cases} \tag{19}$$

$$TPE_f = ECC_f \cdot \sum_{m=1}^{M} \sum_{j=1}^{\delta_f} \left(EC^{Process}_{[f,j],m} \cdot p_{[f,j],m}\right) \tag{20}$$

$$TSE_f = ECC_f \cdot \sum_{m=1}^{M} \left( EC^{Setup}_{0,[f,1],m} \cdot s_{0,[f,1],m} + \sum_{j=2}^{\delta_f} \left( EC^{Setup}_{[f,j-1],[f,j],m} \cdot s_{[f,j-1],[f,j],m} \right) \right) \tag{21}$$

$$TIE_f = ECC_f \cdot \sum_{m=1}^{M} \left( EC^{Idle}_{m} \cdot \left( C_{[f,\delta_f],M} - \sum_{j=1}^{\delta_f} p_{[f,j],m} - \left( s_{0,[f,1],m} + \sum_{j=2}^{\delta_f} s_{[f,j-1],[f,j],m} \right) \right) \right) \tag{22}$$

$$EMAX = max\left(TPE_f + TSE_f + TIE_f\right), f = 1, 2, \ldots, F \tag{23}$$

For ease of understanding, we give a scheduling case of DBFSP_SDST with balanced energy consumption costs having 9 jobs and 3 factories with 3 machines per factory. The processing time $p_{j,m}$ and the corresponding unit energy consumption $EC^{Process}_{j,m}$ of job $j$ on machine $m$ are shown in Tables 2 and 3, respectively. The cost per unit of energy consumption in factory $f$ is $EEC_f = [1, 3, 2]$ The energy consumption per unit idle $EC^{Idle}_m$ of machine $m$ is $EC^{idle}_m = [2 \quad 1 \quad 3]$. The sequence-dependent setup time $s_{j,j',m}$ and the corresponding unit energy consumption $EC^{Setup}_{j,j',m}$ are shown in Tables 4 and 5.

We adopt the Gurobi solver to solve the above example, and obtain the optimal schedule sequence. The job sequence in each factory is (7, 6, 9), (1, 5) and (4, 3, 8, 2), respectively. The Gantt chart of the above scheduling sequences is given in Fig. 1. The optimization objective of this paper is to minimize the energy consumption costs of the critical factory. The energy consumption costs of each factory is calculated as the sum of processing energy cost, setup energy cost, and standby energy cost (in the following, we will refer to the sum of blocking time and idle time as standby time).

**Table 4a**
Sequence-dependent setup time of jobs on machine 1.

| Job | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 4 | 8 | 8 | 5 | 8 | 9 | 2 | 6 | 4 |
| 1 | – | 3 | 6 | 1 | 2 | 4 | 1 | 2 | 7 |
| 2 | 7 | – | 2 | 6 | 1 | 2 | 7 | 5 | 1 |
| 3 | 4 | 5 | – | 7 | 5 | 1 | 1 | 3 | 7 |
| 4 | 6 | 1 | 4 | – | 8 | 1 | 2 | 7 | 1 |
| 5 | 3 | 1 | 4 | 5 | – | 9 | 1 | 6 | 9 |
| 6 | 3 | 6 | 9 | 1 | 6 | – | 2 | 6 | 4 |
| 7 | 6 | 1 | 4 | 5 | 8 | 1 | – | 7 | 1 |
| 8 | 3 | 1 | 4 | 5 | 8 | 9 | 1 | – | 9 |
| 9 | 3 | 6 | 9 | 1 | 6 | 9 | 2 | 6 | – |

**Table 4b**
Sequence-dependent setup time of jobs on machine 2.

| Job | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 2 | 3 | 2 | 3 | 3 | 3 | 6 |
| 1 | – | 4 | 6 | 7 | 3 | 2 | 7 | 3 | 1 |
| 2 | 5 | – | 8 | 6 | 7 | 3 | 1 | 7 | 8 |
| 3 | 2 | 3 | – | 1 | 7 | 8 | 4 | 6 | 3 |
| 4 | 6 | 7 | 2 | – | 2 | 7 | 3 | 1 | 4 |
| 5 | 6 | 7 | 2 | 3 | – | 3 | 8 | 3 | 3 |
| 6 | 4 | 5 | 3 | 8 | 3 | – | 1 | 2 | 6 |
| 7 | 6 | 7 | 2 | 3 | 2 | 7 | – | 1 | 4 |
| 8 | 6 | 7 | 2 | 3 | 2 | 3 | 8 | – | 3 |
| 9 | 4 | 5 | 3 | 8 | 3 | 3 | 1 | 2 | – |

**Table 4c**
Sequence-dependent setup time of jobs on machine 3.

| Job | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 2 | 2 | 6 | 4 | 2 | 6 | 3 | 6 | 2 |
| 1 | – | 3 | 5 | 2 | 3 | 5 | 2 | 3 | 2 |
| 2 | 3 | – | 3 | 5 | 2 | 3 | 2 | 6 | 5 |
| 3 | 4 | 4 | – | 2 | 6 | 5 | 2 | 1 | 2 |
| 4 | 5 | 2 | 4 | – | 6 | 2 | 3 | 2 | 2 |
| 5 | 5 | 2 | 4 | 4 | – | 6 | 5 | 6 | 6 |
| 6 | 2 | 4 | 6 | 5 | 6 | – | 2 | 1 | 2 |
| 7 | 5 | 2 | 4 | 4 | 6 | 2 | – | 2 | 2 |
| 8 | 5 | 2 | 4 | 4 | 2 | 6 | 5 | – | 6 |
| 9 | 2 | 4 | 6 | 5 | 6 | 6 | 2 | 1 | – |

From the Gantt chart, we can see that the maximum completion time of the sequence is 40. We set the end time of the last machine in each factory to be the end time of all machines. The energy consumptions costs $TPE_1$, $TSE_1$ and $TIE_1$ in factory 1 are 260, 77, and 112, respectively. The energy consumptions costs $TPE_2$, $TSE_2$ and $TIE_2$ in factory 2 are 105, 90 and 237, respectively. The energy consumptions $TPE_3$, $TSE_3$ and $TIE_3$ in factory 3 are 124, 196 and 96, respectively. The total energy consumptions costs of factory 1, factory 2 and factory 3 are 449, 432 and 416, respectively, in which the critical factory is the factory with the maximum energy consumption cost, that is, $EMAX$ is 449.

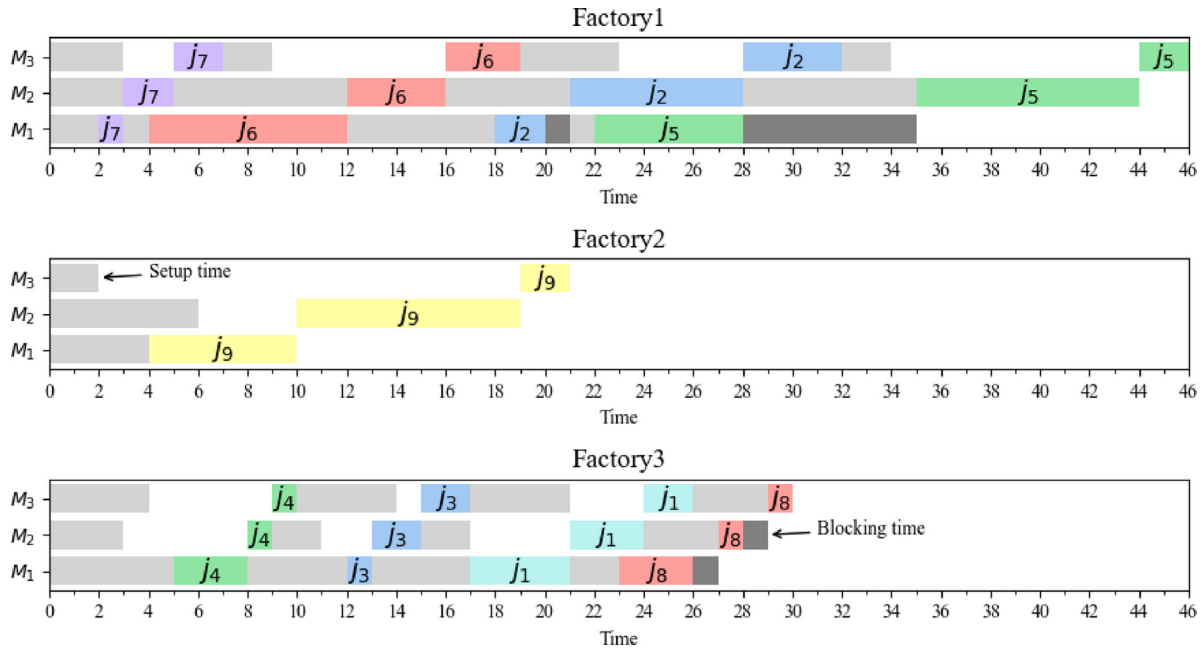



**Fig. 1.** Gantt chart for a solution to the example problem.

**Table 5a**
Unit sequence-dependent setup time energy consumption of jobs on machine 1.

| Job | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 2 | 1 | 4 | 4 | 4 | 1 | 2 | 1 | 1 |
| 1 | – | 5 | 8 | 9 | 1 | 6 | 1 | 4 | 3 |
| 2 | 4 | – | 6 | 9 | 3 | 1 | 4 | 5 | 8 |
| 3 | 1 | 1 | – | 5 | 8 | 3 | 2 | 1 | 3 |
| 4 | 1 | 2 | 3 | – | 5 | 8 | 3 | 1 | 1 |
| 5 | 3 | 5 | 2 | 8 | – | 3 | 1 | 1 | 1 |
| 6 | 3 | 1 | 5 | 8 | 8 | – | 3 | 4 | 3 |
| 7 | 1 | 1 | 4 | 5 | 8 | 3 | – | 1 | 3 |
| 8 | 1 | 2 | 3 | 4 | 5 | 8 | 3 | – | 1 |
| 9 | 3 | 5 | 2 | 8 | 4 | 3 | 1 | 1 | – |

**Table 5b**
Unit sequence-dependent setup time energy consumption of jobs on machine 2.

| Job | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 8 | 2 | 2 | 2 | 2 | 1 | 1 | 3 |
| 1 | – | 3 | 2 | 3 | 8 | 3 | 2 | 1 | 1 |
| 2 | 1 | – | 3 | 3 | 1 | 4 | 2 | 3 | 2 |
| 3 | 1 | 4 | – | 3 | 2 | 1 | 1 | 2 | 3 |
| 4 | 2 | 1 | 1 | – | 3 | 2 | 1 | 1 | 3 |
| 5 | 1 | 3 | 1 | 2 | – | 1 | 1 | 8 | 3 |
| 6 | 1 | 3 | 3 | 2 | 2 | – | 3 | 2 | 1 |
| 7 | 1 | 4 | 2 | 3 | 2 | 1 | – | 2 | 3 |
| 8 | 2 | 1 | 1 | 2 | 3 | 2 | 1 | – | 3 |
| 9 | 1 | 3 | 1 | 2 | 2 | 1 | 1 | 8 | – |

**Table 5c**
Unit sequence-dependent setup time energy consumption of jobs on machine 3.

| Job | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 5 | 5 | 4 | 4 | 4 | 5 | 3 | 6 | 4 |
| 1 | – | 4 | 2 | 6 | 5 | 6 | 4 | 3 | 2 |
| 2 | 3 | – | 6 | 6 | 2 | 8 | 4 | 4 | 6 |
| 3 | 2 | 8 | – | 4 | 6 | 4 | 3 | 5 | 2 |
| 4 | 6 | 7 | 2 | – | 4 | 6 | 4 | 6 | 4 |
| 5 | 3 | 4 | 7 | 6 | – | 4 | 6 | 5 | 4 |
| 6 | 4 | 2 | 4 | 6 | 6 | – | 2 | 4 | 4 |
| 7 | 2 | 8 | 4 | 4 | 6 | 4 | – | 5 | 2 |
| 8 | 6 | 7 | 2 | 4 | 4 | 6 | 4 | – | 4 |
| 9 | 3 | 4 | 7 | 6 | 4 | 4 | 6 | 5 | – |

## 4. Iterative greedy algorithm with a variable neighborhood search strategy

For DBFSP_SDST, two problems should be considered simultaneously: how to allocate the sequence to the factories and how to arrange the sequence of jobs within each factor. Therefore, considering the characteristics of DBFSP_SDST with balanced energy consumption costs criterion, we propose a simple iterative greedy algorithm with variable neighborhood search (VNIG). The VNIG is a variant of the IG algorithm. Within the while loop, we design two different local search strategies to adjust the optimization solution. In addition, a learning-based variable neighborhood strategy is designed to avoid the solution falling into local optima.

### 4.1. Initial solution (MME_en)

NEH2_en has good performance when solving the distributed permutation flowshop scheduling problem [16]. MME is used to generate a good initialization solution for the blocking flowshop scheduling problem [3]. Inspired by the above two heuristics, we proposed the MME_en heuristic, which is the combination of NEH2_en and MME, to solve the distributed blocking flowshop scheduling problem.

The MM algorithm is used to generate a heuristic solution, $\pi == \{\pi^1, \pi^2, \ldots, \pi^k, \ldots, \pi^n\}$, by minimizing the critical path length, where $n$ is the number of jobs. Let $\varphi = \{1, 2, \ldots, n\}$ be the set of initial jobs. First, the job with the shortest processing time on the first machine is chosen as $\pi^1$, and the job with the shortest processing time on the last machine is chosen as $\pi^n$.

The remaining jobs, $\{\pi^2, \ldots, \pi^k, \ldots, \pi^{n-1}\}$ are obtained using function (24).

$$\pi^k = \underset{j \in \varphi \setminus \pi}{\operatorname{argmin}} (\phi \times \sum_{i=1}^{m-1} |p_{j,i} - p_{[k-1],i+1}| + (1-\phi) \times \sum_{i=1}^{m} p_{j,i}), \quad k = 2, 3, \ldots, n-1 \tag{24}$$

where $[k]$ is the job index of the $k$th job of the sequence $\pi$. $\phi$ is a random number between 0 and 1. After obtaining $\pi = \{\pi^1, \pi^2 \ldots, \pi^n\}$ yielded by the MM heuristic, the jobs in $\pi$ are assigned to factories according to the following process. Put first $f$ jobs of the sequence, $\pi$, into each factory one by one. The above operations ensure the uniformity of the allocated quantity. For the remaining jobs of the sequence, we take out the jobs one by one and try to test them in all positions of all factories, and the position $pos_{l*}$ in factory $F_{l*}$ with the minimal objective is selected. At this time, we can guarantee that the energy consumption cost of each factory is as low as possible, but we cannot guarantee that the number of jobs in all factories is the same. Thus, in our paper, we attempt to optimize the scheduling sequences of each factory and search for the optimal solution with minimal energy consumption costs. After insertion, remove a job from $pos_{l*} - 1$ or $pos_{l*} + 1$ of factory $F_{l*}$, then try to insert it at all locations in $F_{l*}$ until you find the lowest energy cost $E'_{l*}$. Algorithm 1 gives the steps of MME_en.

**Algorithm 1 MME_en**

**Input:** $\pi = \phi$, $n$ // $\pi$ is an empty scheduling sequence, and $n$ is the number of jobs

01: $\pi = \{\pi^1, \pi^2 \ldots, \pi^n\} \leftarrow$ MM strategy
02: ***for*** $i = 1$ ***to*** $f$
03: Take job $\pi^i$ from $\pi$ and assign it to factory $F_i$
04: ***endfor***
05: ***for*** $i = f+1$ ***to*** $n$ ***do***
06: ***for*** $l = 1$ ***to*** $f$
07: Test $\pi^i$ in all possible positions in $\pi_l$ // $\pi_l$ *is the job sequence of factory* $l$
08: $E_l$ is the lowest energy cost of factory $F_l$ obtained // $E_l = TPE_l + TSE_l + TIE_l$
09: $pos_l$ is the position with $E_l$
10: ***endfor***
11: $l^* = \arg(\min_{l=1}^{f} E_l)$
12: Insert $\pi^i$ at the position $pos_{l^*}$ of the sequence $\pi_{l^*}$
13: Extract a job from position $pos_{l^*} - 1$ or $pos_{l^*} + 1$ of $\pi_{l^*}$
14: Insert the selected job in all possible positions of $\pi_{l^*}$
15: Put the selected job to the best position with minimal $E'_{l^*}$ of $\pi_{l^*}$
16: ***endfor***

**Output:** $EMAX$, $\pi$

### 4.2. Effective local search strategies

Neighborhood-based local search has an essential role in improving the quality of solutions. Insert and swap operations are widely used as local search strategies because of their simplicity and efficiency. When we solve the problem with the continuous expansion of the flowshop scheduling, the advantages of the iterative improvement strategy based on insertion may be gradually declined compared to the one based on swap operation. This is because the insertion operation needs to move a series of jobs. The time complexity of multilayer loops is very high, i.e., $O(n^3)$. However, the time complexity of the swap operation is $O(n^2)$ [48]. If the termination time is the same, the number of insert operations will be smaller than that of swap operations, which will reduce the number of algorithm iterations and make it difficult to further seek a potential solution. Obviously, if the optimization objective is further enhanced, the insert operator will spend too much execution time. Therefore, in this paper, to reduce the time complexity, local perturbation strategies based on swaps are adopted to improve the local search ability of the solution.

Based on the distributed feature of DBFSP_SDST, two local search strategies based on swap permutation are proposed, i.e., critical and other factories job swapping (Exter_CriticalFactory_Swap) [29] and random factory job block swapping (LS_JBS). The first one randomly selects two jobs from critical and other factories to perform the swap operator. The last one is to swap the job block with minimal energy consumption cost and a random job block and avoid destroying the job block. Based on the above description, the proposed local search strategies can be given as follows.

(1) ***Local search based on critical and other factory job swapping***

The energy consumption cost of the critical factory has a direct influence on the total energy consumption costs. Thus, in this paper, we propose a local search based on single job swapping within the two factories, named Exter_CriticalFactory_Swap. Exter_CriticalFactory_Swap is a swap operation of critical factory and other factory. Two jobs, $job_1$ and $job_2$ are randomly selected from a critical factory and random factory, respectively, and swap the two jobs. (See Algorithm 2)

**Algorithm 2 Exter_CriticalFactory_Swap**

**Input:** $Cnt = 0$, $\pi$, $\pi^{Initial}$ is an initial solution

01: Find a critical factory $F_{E1}$ with $EMAX$ and a random factory $F_2$ selected from $\{F_1, F_2, \ldots, F_n\} / F_{E1}$
02: ***While*** $Cnt < n$ ***do*** *% n is the number of jobs in factory* $F_{E1}$
03: $\pi^{Initial} = \pi$
04: $job_1$=a job randomly selected from $F_{E1}$
05: $job_2$=a job randomly selected from $F_2$
06: Swap job $job_1$ and $job_2$
07: $EMAX^*$=the energy consumption cost of $F_{E1}$ after swapping
08: ***if*** $EMAX^* < EMAX$
09: $EMAX = EMAX^*$
10: Find a critical factory $F_{E1}$ with $EMAX$
11: Random select a factory $F_2$ from $\{F_1, F_2, \ldots, F_n\} / F_{E1}$
12: $Cnt = 0$
13: ***else***
14: $\pi = \pi^{Initial}$ and $Cnt = Cnt + 1$
15: ***endif***
16: ***endwhile***

**Output:** $EMAX$, $\pi$

(2) ***Local search based on random factory job block swapping***

The Exter_CriticalFactory_Swap strategy implemented in the previous step leads to an improvement in the quality of the solution by the interaction between two factories. The improved solution has many excellent job blocks that should be retained. Thus, to avoid destroying the job block, we utilize the good job block to disturb the sequence of a random factory to enhance the quality of the solution. The proposed job block swapping is named LS_JBS. Compared with the above job swapping, job block swapping can retain the integrity of the job block with minimum energy consumption cost. It is noted that LS_JBS is applied to a random factory, guaranteeing that the energy consumption cost of each factory may be reduced and further achieve the balance of the energy consumption cost from each factory.

In Algorithm 3, define the length $d$ of the job block, randomly select the factory $F_k$ from $\{F_1, F_2, \ldots, F_n\}$, and record the energy consumption cost $E_k$ of $F_k$($E_k = TPE_k + TSE_k + TIE_k$). When the number of jobs $n_{F_k}$ in $F_k$ is less than or equal to $3 * d$, a single job exchange (see lines 3–5) is carried out. Otherwise, job block $block_1$ and job block $block_2$ ($block_1 \cap block_2 = \varnothing$) are randomly selected from sequence $\pi_0$. Swap $block_1$ and $block_2$ (see lines 6–9). Finally, the acceptance criterion is executed (see lines 10–16).

**Algorithm 3** ***LS_JBS***

**Input:** $d$, $count = 0$, $\pi_k$ ($\pi_k$ is the sequence of $F_k$)

01: $F_k$ = random selected a factory from $\{F_1, F_2, ..., F_f\}$, $\pi_0 = \pi_k$, $E_k$ is the energy consumption cost of factory $F_k$
02: ***while*** ($count$ < $Factory\_number$)
03: ***if*** $n_{F_k} <= 3*d$ ***do***
04: $job_1, job_2$ =random selected a job from $\pi_0$, respectively // $job_1 \neq job_2$
05: swap $job_1$ and $job_2$, $E_k$ is the energy consumption cost of $F_k$ after swapping
06: ***else***
07: $block_1, block_2$ = random selected a job from $\pi_0$, respectively// $block_1 \cap block_2 = \varnothing$
08: swap $block_1$ and $block_2$, $E_k$ is the energy consumption cost of $F_k$ after swapping
09: ***endif***
10: ***if*** $E_{k'} < E_k$ ***do***
11: $\pi_k = \pi_0$
12: $E_k = E_{k'}$
13: ***else***
14: $\pi_0 = \pi_k$
15: $count = count + 1$;
16: ***endif***
17: ***endwhile***
**Output:** $E_k$, $\pi_k$

### 4.3. Improved variable neighborhood search strategies

In the local search, the solution easily falls into the local optima because of the slight disturbance. In view of this, if the quality of the solution is not improved after *num* generations, we will execute an improved variable neighborhood search (*IVNS*, for short) strategy to avoid the solution trapping into local optima. Furthermore, to enhance the diversity or globality of the solution, we first use destruction and reconstruction strategies to disturb the current solution. After this, we try to adopt different operators to produce a promising solution, which can improve the quality of the solution. In this paper, one of the three strategies, i.e., Exter_CriticalFactory_Swap (shown in Algorithm 2), Insert_K, and Insert_D, is chosen by the learning-based selection strategy in the *IVNS* strategy shown in Algorithm 4.

In Algorithm 4, *cnt* records the number of cycles of *IVNS*, *R*0, *R*1, and R2 record the number of executions of *Exter_CriticalFactory_Swap*, *Insert_K*, and *Insert_D*, respectively. *PL is* the number of jobs that are removed and reinserted in destruction and reconstruction. The first step is to perform destruction and reconstruction strategies to disturb the current solution. For the first $z$ times of IVNS, the random strategy is adopted (see lines 21–30). If the solution is improved, the corresponding $Ri = Ri + 1$, ($i = 0, 1, 2$). After applying IVNS several times, the strategy will be selected according to the value of the probability $P_i$.

**Algorithm 4** ***IVNS***

**Input:** $\pi$, $cnt$, $R0$, $R1$, $R2$, $PL$, $E$ // $E$ *is the energy comsumptioncost of* $\pi$
01: $\pi'' \leftarrow Destruction\_Construction(\pi', PL)$
02: ***if (cnt*** < $z$ ***)*** // $z$ *is a threshold value*
03: $R \leftarrow$ randomly obtain value from 0-2
04: $cnt=cnt+1$
05: ***if*** $R = 0$ ***do***
06: $\pi \leftarrow$ ***Exter_CriticalFactory_Swap*** ($\pi''$)
07: if $E$ is improve
08: R0=R0+1;
09: ***elseif*** $R = 1$ ***do***
10: $\pi \leftarrow$ ***Insert_K*** ($\pi''$)
11: if $E$ is improve
12: R1=R1+1;
13: ***elseif*** $R = 2$ ***do***
14: $\pi \leftarrow$ ***Insert_D*** ($\pi''$, $l$)
15: if $E$ is improve
16: R2=R2+1;
17: ***endif***
18: ***else***
19: Calculate the probability $P_i = Ri / (R0 + R1 + R2)$
20: $r = random[0,1]$.
21: ***if*** ($r \le p_0$)
22 $R = 0$, R0=R0+1;
23: $\pi \leftarrow$ ***Exter_CriticalFactory_Swap*** ($\pi''$)
24: ***else if*** ($p_0 < r \le p_0 + p_1$)
25: $R = 1$, R1=R1+1;
26: $\pi \leftarrow$ ***Insert_K*** ($\pi''$)
27: ***else if*** ($p_0 + p_1 < r$)
28: $R = 2$, R2=R2+1;
29: $\pi \leftarrow$ ***Insert_D*** ($\pi''$, $l$)
30: ***endif***
31: ***endif***
**Output:** $\pi$

In the IG algorithm, the destructive and reconstruction operations are used to largely disturb the current solution and effectively avoid the algorithm from falling into a local optimum. In our algorithm, the destructive and reconstruction (DR, for short) strategies are applied in IVNS as the first step. The framework of DR is shown in Algorithm 5.

**Algorithm 5** Destruction and reconstruction

**Input:** $\pi'$, $PL$ // *PL is* the number of jobs that be removed and reinserted in DR
**Define:** $\pi^{temp} = \varnothing$, $Cnt = 0$ // *Cnt* is a counter
01: ***While*** $Cnt < PL$ ***do*** // *Destruction*
02: $F_l$ = random select factory from $\{F_1, F_2, ..., F_f\}$
03: ***if (***$n > 1$***)*** // $n = |F_l|$ *is the number of jobs in* $F_l$
04: $job$ =random select job from $F_l$
05: Put $job$ into $\pi^{temp}$
06: Delete $job$ from $\pi'$
07: $Cnt++$
08: ***endif***
09: ***endwhile***
10: ***for i=0 to*** $PL$ // *reconstruction*
11: $job_i$ = the $i$th job in $\pi^{temp}$
12: ***for l=0 to f***
13: Test $job_i$ in all possible positions in $\pi_l$ // $\pi_l$ *is the job sequence of factory l*
14: $E_l$ is the lowest energy cost of factory $F_l$ // $E_l = TPE_l + TSE_l + TIE_l$
15: $pos_l$ is the position with $E_l$
16: ***endfor***
17: $l^* = \arg(\min_{l=1}^{f} E_l)$
18: Insert $job$ in the sequence $\pi_{l^*}$ at position $pos_{l^*}$
19: ***endfor***
**Output:** $\pi'$

The DR strategy is divided into two parts: destruction and reconstruction. For the destruction, factory $F_l$ is randomly selected from $\{F_1, F_2, \ldots, F_f\}$. If factory $F_l$ has more than one job, a job will be randomly selected from $F_l$ and put into the empty collection $\pi^{temp}$, and correspondingly, the selected job is deleted from the original sequence. The above operation is performed *PL*

times so that there are $PL$ jobs in set $\pi^{temp}$ (see lines 1–9). For the reconstruction operation, we remove $job_i$ from $\pi^{temp}$ one by one, try to insert it at all positions of all factories, and find the best position $pos_{l*}$ with the minimum energy consumption cost (see lines 12–16). The above step is repeated until all the jobs in sequence $\pi^{temp}$ have been removed.

***Insert_K*** $(\pi)$ refers to the insert operation in a factory. The factory $F_k$ is randomly selected from the set $\{F_1, F_2, \ldots, F_f\}$, and the job sequence of $F_k$ is $\pi_k$. Calculate the sum of the energy consumption cost of each job processing at all machines in the selected factory, and the formula is $JobE_{[F_k,j]}$. We believe that the movement of a job with large processing energy consumption cost is more likely to have an impact on the quality of the solution. Therefore, the job with the maximum energy consumption cost is selected from the factory, denoted as *MaxJob*. Next, *MaxJob* is inserted into all positions of the solution from all the factories, and the position with the minimal value of total energy consumption cost is selected.

**Algorithm 6 Insert based on critical job (Insert_K)**

**Input:** $\pi_k$

01: $F_k$ = random selected factory from $\{F_1, F_2, ..., F_f\}$

02: ***for*** $j = 0$ ***to*** $n$ ***do*** //$n$is the job number of $\pi_k$allocated in $F_k$

03: calculated $JobE_{[F_k,j]} = \sum_{i=1}^{m} p_{[F_k,j],i} * EC^{Process}_{[F_k,j],i}$

04: ***endfor***

05: Find the job, $MaxJob$, with maximum energy cost from $\pi_k$

06: Test insert $MaxJob$ in all possible positions of all factories

07: Put $MaxJob$ in the best position $h$ with the lowest energy cost $E_l$

**Output:** $EMAX, \pi_k$

***Insert_D*** $(\pi, l)$ states a job block dispersal insertion method to increase the diversity of the solution. Compared with the job block insertion operation, *Insert_D* $(\pi, l)$ makes it easier to jump out of the local optimum. The job block with maximum energy consumption cost is obtained from a random factory. Next, each job in the block is reinserted in the best position of all the factories one by one. In Algorithm 7, first, define the length $l$ of the job block. Then, randomly select the factory $F_r$ in $\{F_1, F_2, \ldots, F_f\}$, find the block of jobs with maximum energy consumption cost from $F_r$ and store it in the array *Block*. In lines 3–7, the jobs are taken from *Block* and reinserted at all positions in all the factories. The position with the minimal total energy consumption cost is found. The above steps are repeated until all the jobs in the job block are inserted.

**Algorithm 7 Job block dispersal insertion strategy (Insert_D)**

**Input:** $\pi_k, l$ // $l$ *is the number of the job block*

01: $F_r$=random select factory from $\{F_1, F_2, ..., F_f\}$

02: $Block$ = the job block with most energy cost in $F_r$

03: ***for*** $i = 0$ ***to*** $l$ ***do***

04: Get the job $Block[i]$ in the $Block$

05: Test insert $Block[i]$ in all possible positions of all factories

06: Insert job $Block[i]$ in the position resulting with the lowest energy cost $E_l$

07: ***endfor***

**Output:** $EMAX = \max_{l=0}^{f} E_l, \pi_k$

## 5. Experiments and analysis

### 5.1. Experiment settings

In this section, we validate the performance of the VNIG. Because there are few algorithms for solving DBFSP_SDST with balanced energy cost criterion. Therefore, we selected some algorithms that were used to solve the DFSP that are closely related to the DBFSP_SDST. The compared algorithms are artificial chemical reaction optimization (CRO) [16], the discrete artificial bee colony algorithm (DABC) [36], the iterative greedy algorithm with a restart scheme (IGR) [28], and an evolution strategy approach (ES) [49]. In our experiments, the following comparison was performed:

(1) Verification of the MILP model
(2) Sensitivity study on three parameters
(3) Comparison of the proposed MME_en and NEH2_en.
(4) Validate the effectiveness of IVNS
(5) Comparison results between the VNIG algorithm and the existing four compared algorithms.

In this paper, we choose 90 instances, where the number of jobs $n$ comes from the set {100, 200, 300, 400, 500}, the number of machines $m$ comes from the set {5, 8, 10}, and the number of factories $f$ is {2, 3, 4, 5, 6, 7}. Thus, 90 different combinations can be obtained by combining $m$, $n$, and $f$. The processing data are generated as follows. The values of setup time and processing time are in the range of [1,99), and we set the energy consumption per unit of processing, setup, and standby as in the range of [4, 6], [1, 3], [1, 2], respectively. The cost per unit of energy consumption in each factory is in the range of [1, 10].

For the termination criterion of all the compared algorithms, the same maximal elapsed CPU time of $TimeLimit = t \times n \times m$ milliseconds, where $n$ represents the number of jobs, $m$ refers to the number of machines, and $t$ is equal to 2 and 3. Thus, the computation time can be adjusted for different sizes of instances and the value of $t$. To evaluate the performance of the proposed algorithm, we choose the energy consumption cost objective and the relative percentage increase (RPI) as evaluation indicators. The formula for calculating RPI is as follows.

$$RPI = \frac{M_i - M_{best}}{M_{best}} \times 100 \tag{25}$$

where $M_i$ is the average energy consumption cost value of the $i$th algorithm and $M_{best}$ is the optimal value obtained by all the algorithms. A smaller value of RPI means a better performance.

We also used normalization to process the results. Normalization reduces the values to [0,1] in equal proportions, aiming to make the differences between the values clearer. The normalization formula is as follows.

$$y' = \frac{x - \min(x_i)}{\max(x_i) - \min(x_i)}, 1 < i < \sigma \tag{26}$$

where $y'$ is the value after normalization and $x$ is the value that needs to be processed. $\sigma$ is the number of compared algorithms. $\min(x_i)$ and $\max(x_i)$ denote the minimum and maximum of the values to be processed, respectively.

In this study, all the algorithms adopt the same maximal elapsed CPU time with the unit of milliseconds as the termination criterion. All the experiments should be conducted and compared under the same or stricter conditions. Variety algorithms are implemented on different PCs, and the implementation settings will be different. In this circumstance, using execution time as the stopping criterion will no longer be reliable since execution time may be affected by the operating system and other applications running during the experiments. Thus, all the algorithms are written in Visual C++ 2019, and the same library functions are adopted in this study to make a fair comparison. For their implementations, all the algorithms are realized on a PC with Pentium (R) Dual 2.9 GHz and 8 G memory, in which the operating system is Microsoft Windows 7 X 64. In addition, the same background running environment is employed, the background processes that may occupy system resources are closed, and no other programs are executed in parallel while implementing an algorithm.

**Table 6**
Result for the MILP model.

| F_J_M | Gap by Gurobi | Energy consumption cost by Gurobi (time) | Energy consumption cost by VNIG (time) |
|---|---|---|---|
| 3_6_3 | 0% | **514** (0.07 s) | **514** (0.036 s) |
| 3_7_3 | 0% | **574** (1 s) | **574** (0.105 s) |
| 3_9_3 | 0% | **652** (32.64 s) | **652** (0.81 s) |
| 3_10_3 | 0% | **656** (669.82 s) | **656** (1.5 s) |
| 3_12_3 | 23.40% | **752** (1000 s) | 782 (1.8 s) |
| 3_14_3 | 26.04% | 845 (1000 s) | **825** (2.1 s) |
| 3_16_3 | 21.68% | 895 (1000 s) | **890** (2.4 s) |
| 3_20_3 | 21.76% | 1190 (3600 s) | **1110** (6 s) |
| 3_30_3 | 21.05% | 1910 (3600 s) | **1904** (13.5 s) |
| 3_40_3 | 19.84% | 2424 (3600 s) | **2400** (18 s) |

### 5.2. Verification of the MILP model

In this section, we select eight instances with small sizes to verify the effectiveness of MILP using the Gurobi solver [48]. In Table 6, F_J_M represents the numbers of factories, jobs and machines. We set the running time to 1000 s and 3600 s to ensure sufficient time to search for the solution. Table 6 gives the values of the gap, running time, and energy consumption cost obtained by Gruobi. In addition, the values of energy consumption cost and running time obtained by our VNIG algorithm are listed in Table 6. Gap = 0 means that the optimal solution is found for the problem. For a minimization model, Gap is computed as (ObjVal-ObjBound)/ObjVal, where ObjVal is the objective value for the current solution, and ObjBound is the lower bound that gives a bound of the best possible objective. Thus, if the gap is not equal to 0, it does not mean that no optimal solution is found.

As seen from Table 6, the optimal solution will be found by Gurobi when the size of instance is small, i.e., 3_6_3, 3_7_3 3_9_3, and 3_10_3 instances. For 4 out of 10 instances, the values of the energy consumption cost obtained by Gurobi and VNIG are the same. However, the computation time of VNIG is far less than that of Gurobi. For 5 out of 10 instances, the values of the energy consumption cost yielded by Gurobi are good, suggesting that the Gurobi solver can obtain better solutions in small-scale examples than those of VNIG. However, with the increasing scale of the instances, the solutions obtained by VNIG are gradually better than those obtained by the Gurobi solver, and it takes less time. Therefore, we believe that the VNIG solver is more suitable than the Gurobi solver for solving large-scale and complicated instances.

### 5.3. Parameters calibration

To demonstrate the effectiveness of the proposed algorithm, sensitivity analyses of the five parameters are first conducted. The five parameters are $d$ (the length of the job block swapping), $l$ (the length of the job block insertion), $num$ (the execution number of the unimproved solution), $PL$ (the number of jobs that are removed and reinserted in destruction and reconstruction), and $z$ (the threshold value used for the random selection strategy in Algorithm 5). To better determine the values of the parameters, we adopt the Taguchi experimental method to calibrate them.

Table 7 shows the five factor levels of each parameter, i.e., $d \in \{1, 2, 3, 5, 7\}$, $l \in \{1, 3, 4, 6, 9\}$, $num \in \{1, 3, 5, 8, 10\}$, $PL \in \{2, 3, 4, 5, 9\}$, and $z \in \{10, 30, 50, 70, 90\}$. We obtained 25 combinations by the orthogonal table listed in Table 8. For the sake of fairness, we select eight instances with different sizes, e.g., $100 \times 10 \times 2$, $200 \times 8 \times 3$, $200 \times 10 \times 6$, $300 \times 5 \times 4$, $300 \times 8 \times 7$, $400 \times 10 \times 4$, $500 \times 5 \times 2$ and $500 \times 8 \times 5$. Each instance is run 20 times independently under the same conditions, and the corresponding RPI values are obtained. Subsequently, the mean RPI values of the five instances with different combinations of parameters are integrated. According to the obtained RPI, the trend of the factor level is plotted in Fig. 2.

**Table 7**
Parameters level.

| Parameters | Parameters level | | | | |
|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 |
| $d$ | 1 | 2 | 3 | 5 | 7 |
| $l$ | 1 | 3 | 4 | 6 | 9 |
| $num$ | 1 | 3 | 5 | 8 | 10 |
| $PL$ | 2 | 3 | 4 | 5 | 9 |
| $z$ | 10 | 30 | 50 | 70 | 90 |

**Table 8**
Orthogonal array and response value.

| Combination | Parameters | | | | | Response (RPI) |
|---|---|---|---|---|---|---|
| | $d$ | $l$ | $num$ | $PL$ | $z$ | |
| 1 | 1 | 1 | 1 | 2 | 10 | 0.477362 |
| 2 | 1 | 3 | 3 | 3 | 30 | 0.513544 |
| 3 | 1 | 4 | 5 | 4 | 50 | 0.495309 |
| 4 | 1 | 6 | 8 | 5 | 70 | 0.550468 |
| 5 | 1 | 9 | 10 | 9 | 90 | 0.676341 |
| 6 | 2 | 1 | 3 | 4 | 70 | 0.513624 |
| 7 | 2 | 3 | 5 | 5 | 90 | 0.525749 |
| 8 | 2 | 4 | 8 | 9 | 10 | 0.569902 |
| 9 | 2 | 6 | 10 | 2 | 30 | 0.487203 |
| 10 | 2 | 9 | 1 | 3 | 50 | 0.509608 |
| 11 | 3 | 1 | 5 | 9 | 30 | 0.571204 |
| 12 | 3 | 3 | 8 | 2 | 50 | 0.539073 |
| 13 | 3 | 4 | 10 | 3 | 70 | 0.529518 |
| 14 | 3 | 6 | 1 | 4 | 90 | 0.557715 |
| 15 | 3 | 9 | 3 | 5 | 10 | 0.571028 |
| 16 | 5 | 1 | 8 | 3 | 90 | 0.530301 |
| 17 | 5 | 3 | 10 | 4 | 10 | 0.574164 |
| 18 | 5 | 4 | 1 | 5 | 30 | 0.581162 |
| 19 | 5 | 6 | 3 | 9 | 50 | 0.617788 |
| 20 | 5 | 9 | 5 | 2 | 70 | 0.55007 |
| 21 | 7 | 1 | 10 | 5 | 50 | 0.593352 |
| 22 | 7 | 3 | 1 | 9 | 70 | 0.604799 |
| 23 | 7 | 4 | 3 | 2 | 90 | 0.497132 |
| 24 | 7 | 6 | 5 | 3 | 10 | 0.569272 |
| 25 | 7 | 9 | 8 | 4 | 30 | 0.561151 |

**Table 9**
The mean RPI response values and rank of each parameter.

| Level | $d$ | $l$ | $num$ | $PL$ | $z$ |
|---|---|---|---|---|---|
| 1 | 0.5426 | 0.5372 | 0.5461 | 0.5102 | 0.5523 |
| 2 | 0.5212 | 0.5515 | 0.5426 | 0.5304 | 0.5429 |
| 3 | 0.5537 | 0.5346 | 0.5423 | 0.5404 | 0.551 |
| 4 | 0.5707 | 0.5565 | 0.5502 | 0.5644 | 0.5497 |
| 5 | 0.5651 | 0.5736 | 0.5721 | 0.608 | 0.5574 |
| Delta | 0.0495 | 0.039 | 0.0298 | **0.0978** | 0.0146 |
| Rank | **2** | **3** | 4 | **1** | 5 |

Table 9 lists the significance level of each parameter according to the average RPI value of each scale instance, where Delta measures the size of the effect by taking the difference between the maximum and minimum average RPI of the four factors. A larger value of Delta generally indicates a more significant influence. In addition, a smaller Rank value means a larger difference for different values of the parameter.

As seen from Table 9, the parameter $PL$ has the greatest impact on the algorithm, followed by $d$, $l$, $num$ and $z$. From Fig. 2, the parameter $PL$ has the best effect when the value is 2. As the number of broken jobs increases, the efficiency of the algorithm becomes low. We believe that too much job block destruction may lead to the destruction of excellent job sequences, which degrades the performance of the algorithm. When $d = 2$, the performance of VNIG is great. With the increase in job block length, the algorithm becomes progressively less effective. The

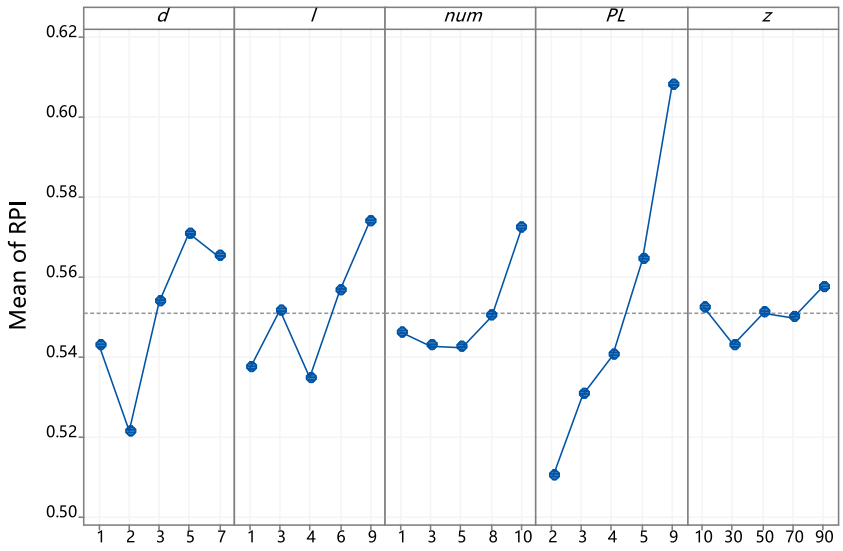
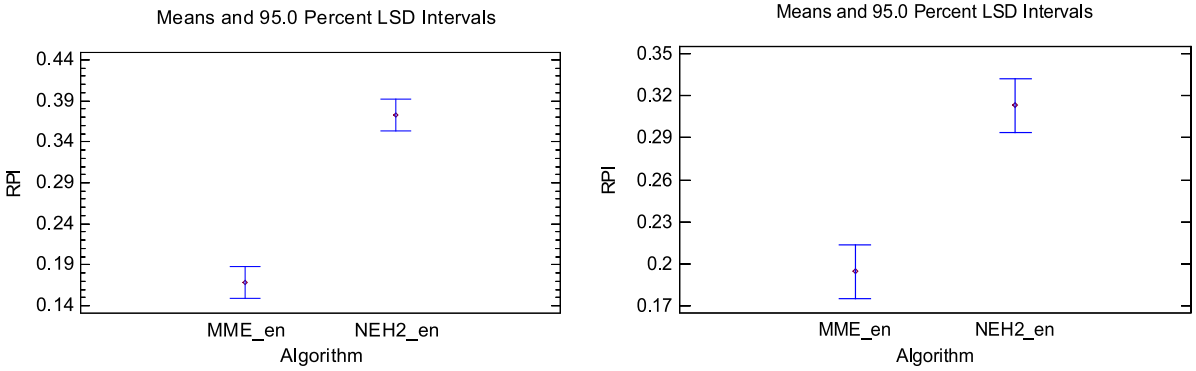



**Fig. 2.** The trend of the factor level.




(a) $t$ =2 (b) $t$ =3

**Fig. 3.** Confidence intervals for NEH2_en and MME_en.

reason may be that the current solution in a factory may be an approximated optimal solution before disturbing; thus, a slight disturbing should be done. However, if the length of the job block is large, the disturbance to the sequence will be very large by swapping the job block, which leads to the generation of a bad solution. For the length $l$ of the job block insertion, the algorithm obtains the minimal mean RPI value when the value of $l$ is 4. The reason may be that if the length of the job block is small, the disturbance to the sequence will be small, which leads to poor diversity. When $num = 5$, the VNIG shows great performance. We believe that an $num$ value that is too small will make the local search strategy not fully work, and an $num$ value that is too large will not jump out in time when the solution falls into a local optimum, which will take more time. Therefore, it is reasonable that the value of $num$ is 5. Parameter $z$ has a small impact on the algorithm. The performance of the algorithm is good when $z = 30$. According to the above experimental results and analysis, we set the parameters as $d = 2$, $l = 4$, $num = 5$, $PL = 2$ and $z = 30$.

In addition, we calibrated the parameters of the four compared algorithms using the Taguchi experimental method. The calibration results are shown in the supplementary data.

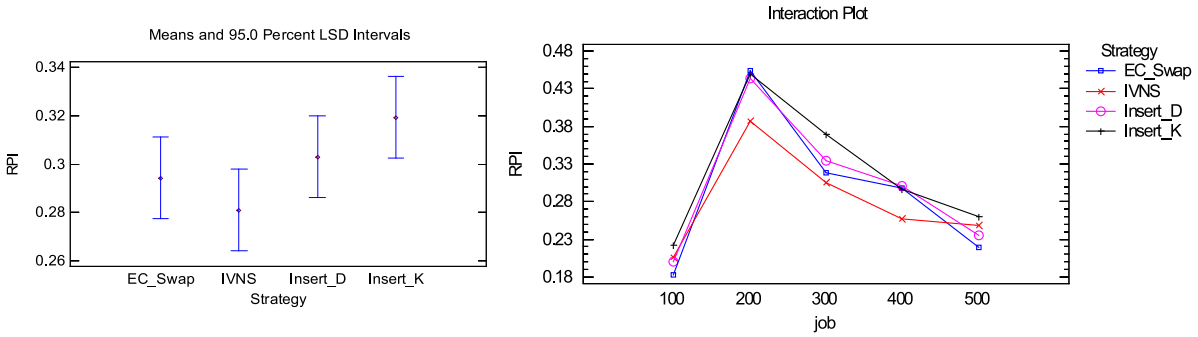
### 5.4. Comparison of the results of NEH2_en and MME_en

NEH2_en proposed by Ruiz and Pan can generate high-quality solutions [18] and show better performance than NEH. Therefore, based on this, we combined the MM and NEH2_en to propose an efficiency MME_en strategy. To evaluate the performance of the proposed initialization strategy, we equip the developed algorithm with NEH2_en and MME_en heuristics. The experimental results are shown in Fig. 3 when $t = 2$ and 3.

From Fig. 3, the RPI value of MME_en is lower than that of NEH2_en, which indicates that the former is more efficient in seeking promising solutions for the DBFSP. The results suggest that the MM heuristic has a better performance in optimizing the makespan of BFSP than other heuristics. The reason is that MM selects the job based on the blocking time on these machines by utilizing the shortest critical path, which can effectively deal with the blocking constraint.

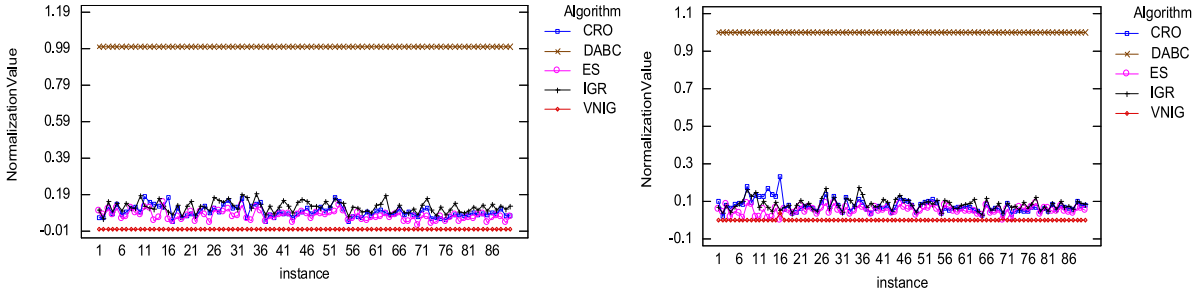
### 5.5. Validation of the effectiveness of the variable neighborhood search strategy based on the learning method

To comprehensively evaluate the performance of the proposed improved learning-based variable neighborhood search strategy, we test the performance of learning-based IVNS and IVNIG with only applying Exter_CriticalFactory_Swap, Insert_K, and Insert_D



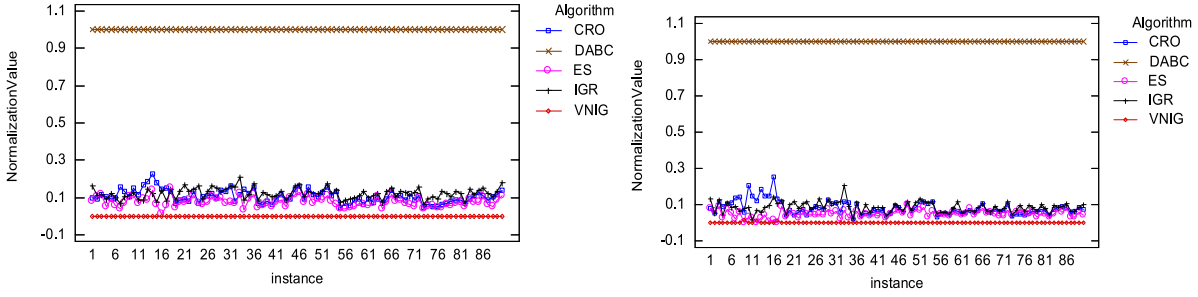

(a) Confidence interval (b) Interaction plot

**Fig. 4.** RPI for disturbing strategies and IVNS.




(a). minimum values after normalization (b). maximum values after normalization

**Fig. 5.** Minimum and Maximum values after normalization of the compared algorithms when t = 2.




(a). minimum values after normalization (b). maximum values after normalization

**Fig. 6.** Minimum and Maximum values after normalization of the compared algorithms when t = 3.

strategies. For the sake of convenience, the IVNS with only applying Exter_CriticalFactory_Swap, Insert_K, and Insert_D strategies is named as EC_swap, Insert_K, and Insert_D strategies, respectively. They run the same test instances under the same conditions.

From Fig. 4(a), we observe that the difference in the performance between EC_Swap and Insert_D is not significant except that the performance of Insert_K was slightly worse, and the performance of IVNS is best among the above compared operators. Except for that, we test the scenario with jobs of different sizes. From Fig. 4(b), EC_Swap shows great performance when the numbers of jobs are 100, 300, 400 and 500. Insert_K can obtain the great solutions when the number of jobs is 200, and the performance of Insert_D is excellent when the number of jobs is 400 and 500. However, considering the above three operators in the variable neighborhood search strategy, the variable neighborhood search strategy based on the learning method can obtain superior results in most situations. The reason might be that the different operators can generate more promising solutions, which can improve the exploitation of an algorithm by disturbing the current solution.

### *5.6. Comparison results between the VNIG algorithm and the five compared algorithms*

In this section, the five algorithms, i.e., CRO [16], DABC [36], IGR [28], ES [49] (the parameters of the comparison algorithm were calibrated according to the method in Section 5.3.), and the proposed VNIG is compared for $t = 2$ and $t = 3$ ($t$ is the variable used to adjust the running time; the larger the value of $t$ is, the longer the running time of the algorithm), respectively. The experimental results are listed in Tables 10 and 11, respectively, where a row represents the results obtained by different algorithms in terms of the average and RPI values of the energy consumption cost objective. In addition, Figs. 5 and 6 display the maximum and minimum values for 90 instances obtained by all the compared algorithms. The confidence intervals and the evolutionary curve of all the compared algorithms are given

**Table 10**
Energy consumption costs of the compared algorithms when $t = 2$.

| Job | m ∗ f | DABC[2018] | | VNIG | | IGR[2020] | | ES[2022] | | CRO[2017] | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | **avg** | RPI | **avg** | RPI | **avg** | RPI | **avg** | RPI | **avg** | RPI |
| 100 | 5 ∗ 2 | 107 683.7 | 9.33 | **98 812.1** | **0.32** | 99 538.8 | 1.06 | 99 425.9 | 0.95 | 99 523.1 | 1.04 |
| | 8 ∗ 2 | 241 849.8 | 11.99 | **217 059.2** | **0.51** | 218 203.9 | 1.04 | 218 326.8 | 1.10 | 218 433.9 | 1.15 |
| | 10 ∗ 2 | 263 729.6 | 9.75 | **241 179.7** | **0.36** | 243 571.4 | 1.36 | 243 191 | 1.20 | 243 370.3 | 1.27 |
| | 5 ∗ 3 | 97 356.1 | 10.07 | **89 076.8** | **0.71** | 89 521.2 | 1.21 | 89 311.5 | 0.97 | 89 595.9 | 1.29 |
| | 8 ∗ 3 | 259 707.4 | 10.72 | **235 782.3** | **0.52** | 237 808.2 | 1.38 | 237 585.9 | 1.28 | 238 043.1 | 1.48 |
| | 10 ∗ 3 | 375 025.8 | 9.85 | **342 621** | **0.36** | 345 494.1 | 1.20 | 344 115.7 | 0.79 | 345 473.3 | 1.19 |
| | 5 ∗ 4 | 67 717.1 | 11.00 | **61 304.9** | **0.49** | 62 149.7 | 1.87 | 61 643.3 | 1.04 | 61 857.9 | 1.39 |
| | 8 ∗ 4 | 274 612.6 | 10.68 | **248 730.8** | **0.25** | 251 984.5 | 1.56 | 251 540.4 | 1.38 | 252 053.8 | 1.59 |
| | 10 ∗ 4 | 446 432.7 | 10.65 | **405 018.9** | **0.38** | 410 550.4 | 1.76 | 408 739.4 | 1.31 | 409 327.3 | 1.45 |
| | 5 ∗ 5 | 122 285.8 | 10.48 | **111 363.8** | **0.61** | 113 212.6 | 2.28 | 111 934.8 | 1.13 | 112 817.6 | 1.93 |
| | 8 ∗ 5 | 413 736.2 | 11.04 | **374 732.2** | **0.57** | 379 404.7 | 1.83 | 377 488.5 | 1.31 | 380 435.6 | 2.10 |
| | 10 ∗ 5 | 220 447.1 | 11.38 | **198 995.1** | **0.54** | 201 062.5 | 1.58 | 200 681.8 | 1.39 | 201 636.1 | 1.87 |
| | 5 ∗ 6 | 70 789.5 | 11.29 | **63 906.5** | **0.47** | 64 523.9 | 1.44 | 64 067.3 | 0.72 | 64 897 | 2.03 |
| | 8 ∗ 6 | 246 534.2 | 11.60 | **221 701.5** | **0.36** | 224 615 | 1.68 | 223 008.1 | 0.95 | 224 973 | 1.84 |
| | 10 ∗ 6 | 218 040.1 | 10.96 | **197 194.5** | **0.35** | 199 564.2 | 1.56 | 199 234.4 | 1.39 | 200 290.5 | 1.93 |
| | 5 ∗ 7 | 99 387.4 | 12.41 | **89 111.1** | **0.79** | 89 783 | 1.55 | 89 183.8 | 0.87 | 90 514.2 | 2.37 |
| | 8 ∗ 7 | 741 817.6 | 10.46 | **672 502.1** | **0.14** | 678 004.2 | 0.95 | 675 931.6 | 0.65 | 675 920.7 | 0.64 |
| | 10 ∗ 7 | 974 158.9 | 9.41 | **892 617.2** | **0.25** | 901 197.6 | 1.21 | 897 511.6 | 0.80 | 900 861 | 1.17 |
| | Mean | 291 184.0 | 10.72 | **264 539.4** | **0.44** | 267 232.8 | 1.47 | 266 273.4 | 1.07 | 267 223.6 | 1.54 |
| 200 | 5 ∗ 2 | 287 350.2 | 10.73 | **260 245** | **0.29** | 261 644.2 | 0.83 | 261 293.7 | 0.69 | 261 538.1 | 0.79 |
| | 8 ∗ 2 | 871 425 | 9.55 | **797 772.2** | **0.29** | 805 428.2 | 1.25 | 801 757.6 | 0.79 | 801 577.2 | 0.77 |
| | 10 ∗ 2 | 2 265 412 | 8.03 | **2 102 349** | **0.26** | 2 123 410 | 1.26 | 2 116 478 | 0.93 | 2 115 328 | 0.88 |
| | 5 ∗ 3 | 221 064.5 | 10.76 | **199 961.3** | **0.19** | 201 688.5 | 1.05 | 200 914.3 | 0.67 | 201 415.7 | 0.92 |
| | 8 ∗ 3 | 558 230.4 | 8.85 | **514 281.6** | **0.28** | 518 947 | 1.19 | 517 639.2 | 0.93 | 517 832.4 | 0.97 |
| | 10 ∗ 3 | 640 871.5 | 9.12 | **589 411.5** | **0.36** | 593 505 | 1.05 | 592 180.1 | 0.83 | 593 868 | 1.12 |
| | 5 ∗ 4 | 234 815.2 | 11.30 | **211 770.3** | **0.38** | 213 447.8 | 1.17 | 212 609.3 | 0.78 | 213 480.1 | 1.19 |
| | 8 ∗ 4 | 690 437.4 | 10.04 | **630 088.1** | **0.42** | 638 704.8 | 1.79 | 634 769.2 | 1.16 | 636 029.5 | 1.36 |
| | 10 ∗ 4 | 1 088 798 | 8.22 | **1 008 091** | **0.20** | 1 021 621 | 1.54 | 1 015 898 | 0.97 | 1 018 893 | 1.27 |
| | 5 ∗ 5 | 172 276.2 | 9.80 | **157 598** | **0.45** | 159 080.6 | 1.39 | 158 464.1 | 1.00 | 159 068.2 | 1.38 |
| | 8 ∗ 5 | 729 530.4 | 10.65 | **661 189.6** | **0.28** | 671 643.7 | 1.87 | 668 243.3 | 1.35 | 671 378.3 | 1.83 |
| | 10 ∗ 5 | 974 158.9 | 9.41 | **892 617.2** | **0.25** | 901 197.6 | 1.21 | 897 511.6 | 0.80 | 900 861 | 1.17 |
| | 5 ∗ 6 | 166 245.2 | 11.07 | **150 116.2** | **0.30** | 151 617.4 | 1.30 | 151 298.2 | 1.09 | 151 675.2 | 1.34 |
| | 8 ∗ 6 | 539 975.5 | 11.99 | **485 414.3** | **0.67** | 493 767 | 2.40 | 490 197.1 | 1.66 | 493 096.5 | 2.26 |
| | 10 ∗ 6 | 417 041.1 | 9.21 | **383 379.9** | **0.40** | 388 295.6 | 1.69 | 385 193.6 | 0.87 | 385 607 | 0.98 |
| | 5 ∗ 7 | 99 039.3 | 9.85 | **90 397.9** | **0.27** | 91 183.1 | 1.14 | 90 822.2 | 0.74 | 91 196 | 1.15 |
| | 8 ∗ 7 | 171 544.6 | 9.43 | **157 529.8** | **0.49** | 159 924.4 | 2.01 | 158 823.6 | 1.31 | 159 091.7 | 1.48 |
| | 10 ∗ 7 | 421 748.6 | 10.29 | **383 586.8** | **0.31** | 389 470.5 | 1.85 | 386 885.7 | 1.18 | 388 382.6 | 1.57 |
| | Mean | 586 109.1 | 9.91 | **537 544.4** | **0.34** | 543 587.6 | 1.45 | 541 165.5 | 0.99 | 542 239.9 | 1.25 |
| 300 | 5 ∗ 2 | 741 817.6 | 10.46 | **672 502.1** | **0.14** | 678 004.2 | 0.95 | 675 931.6 | 0.65 | 675 920.7 | 0.64 |
| | 8 ∗ 2 | 3 346 741 | 9.44 | **3 068 009** | **0.32** | 3 095 932 | 1.24 | 3 081 558 | 0.77 | 3 083 167 | 0.82 |
| | 10 ∗ 2 | 2 970 733 | 8.71 | **2 736 407** | **0.14** | 2 752 796 | 0.74 | 2 751 408 | 0.69 | 2 751 547 | 0.69 |
| | 5 ∗ 3 | 1 192 222 | 12.95 | **1 058 036** | **0.24** | 1 071 903 | 1.55 | 1 067 497 | 1.13 | 1 070 352 | 1.40 |
| | 8 ∗ 3 | 969 296.6 | 10.03 | **883 490.9** | **0.29** | 894 131.6 | 1.50 | 889 083.9 | 0.93 | 889 532.8 | 0.98 |
| | 10 ∗ 3 | 2 103 154 | 8.45 | **1 943 054** | **0.19** | 1 960 924 | 1.12 | 1 955 334 | 0.83 | 1 957 410 | 0.93 |
| | 5 ∗ 4 | 338 870.7 | 11.20 | **305 277.6** | **0.17** | 307 893.4 | 1.03 | 306 857.8 | 0.69 | 307 616.5 | 0.94 |
| | 8 ∗ 4 | 520 527.7 | 9.98 | **474 733** | **0.30** | 480 403.1 | 1.50 | 477 478.3 | 0.88 | 478 268.7 | 1.05 |
| | 10 ∗ 4 | 1 343 634 | 9.06 | **1 234 593** | **0.21** | 1 250 594 | 1.51 | 1 244 236 | 0.99 | 1 245 565 | 1.10 |
| | 5 ∗ 5 | 612 725 | 10.60 | **555 647.4** | **0.29** | 563 234.9 | 1.66 | 559 909.8 | 1.06 | 561 886.5 | 1.42 |
| | 8 ∗ 5 | 375 199.9 | 8.85 | **345 010.8** | **0.10** | 348 676.5 | 1.16 | 346 968.7 | 0.66 | 347 729.6 | 0.88 |
| | 10 ∗ 5 | 818 390.9 | 8.64 | **755 444.2** | **0.28** | 761 889.3 | 1.14 | 760 663.1 | 0.98 | 761 011 | 1.02 |
| | 5 ∗ 6 | 450 991.6 | 12.00 | **404 488.6** | **0.45** | 408 478 | 1.44 | 407 057.5 | 1.09 | 408 522.1 | 1.45 |
| | 8 ∗ 6 | 371 006.1 | 10.50 | **336 657.2** | **0.27** | 340 758 | 1.49 | 338 915.4 | 0.94 | 340 261 | 1.34 |
| | 10 ∗ 6 | 869 756 | 9.89 | **793 212.7** | **0.22** | 801 750.6 | 1.30 | 799 061.6 | 0.96 | 801 820.3 | 1.30 |
| | 5 ∗ 7 | 174 999.7 | 10.11 | **159 557.7** | **0.40** | 161 537.7 | 1.64 | 160 713.9 | 1.12 | 161 554.9 | 1.65 |
| | 8 ∗ 7 | 822 552.1 | 10.14 | **748 492** | **0.22** | 756 747.7 | 1.33 | 756 112.6 | 1.24 | 757 062.6 | 1.37 |
| | 10 ∗ 7 | 560 438 | 10.68 | **507 871.9** | **0.30** | 514 602 | 1.63 | 511 973.7 | 1.11 | 513 579.2 | 1.42 |
| | Mean | 1 032 392.0 | 10.09 | **943 471.4** | **0.25** | 952 792.0 | 1.33 | 949 486.7 | 0.93 | 950 711.5 | 1.14 |
| 400 | 5 ∗ 2 | 1 430 559 | 11.13 | **1 289 738** | **0.19** | 1 297 237 | 0.77 | 1 295 463 | 0.63 | 1 296 189 | 0.69 |
| | 8 ∗ 2 | 2 748 351 | 9.39 | **2 517 102** | **0.18** | 2 542 494 | 1.19 | 2 531 784 | 0.77 | 2 530 290 | 0.71 |
| | 10 ∗ 2 | 4 955 614 | 8.73 | **4 565 990** | **0.18** | 4 607 398 | 1.09 | 4 596 773 | 0.86 | 4 591 031 | 0.73 |
| | 5 ∗ 3 | 1 198 829 | 10.28 | **1 089 260** | **0.20** | 1 097 582 | 0.96 | 1 094 224 | 0.66 | 1 096 504 | 0.87 |
| | 8 ∗ 3 | 2 368 007 | 9.12 | **2 174 403** | **0.20** | 2 189 725 | 0.91 | 2 183 414 | 0.61 | 2 189 302 | 0.89 |
| | 10 ∗ 3 | 1 360 247 | 8.91 | **1 250 931** | **0.15** | 1 260 089 | 0.89 | 1 257 246 | 0.66 | 1 258 415 | 0.75 |
| | 5 ∗ 4 | 847 639.4 | 11.91 | **758 813.4** | **0.18** | 768 866.8 | 1.51 | 764 446.7 | 0.93 | 767 075.1 | 1.27 |
| | 8 ∗ 4 | 1 269 549 | 9.99 | **1 156 808** | **0.22** | 1 170 118 | 1.37 | 1 164 083 | 0.85 | 1 166 934 | 1.10 |
| | 10 ∗ 4 | 493 735.4 | 8.42 | **456 892.4** | **0.33** | 462 161.6 | 1.48 | 459 109.7 | 0.81 | 459 621.6 | 0.93 |
| | 5 ∗ 5 | 262 584.6 | 11.90 | **235 277.2** | **0.26** | 237 214.2 | 1.09 | 236 553.6 | 0.81 | 237 092.4 | 1.04 |

(*continued on next page*)

in Figs. 7 and 8 to show their identification and convergence, respectively.

Tables 10 and 11 list the PRI and average energy consumption cost value of the algorithms for 90 different instances. From the

**Table 10** (*continued*).

| Job | m ∗ f | DABC[2018] | | VNIG | | IGR[2020] | | ES[2022] | | CRO[2017] | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | **avg** | RPI | **avg** | RPI | **avg** | RPI | **avg** | RPI | **avg** | RPI |
| | 8 ∗ 5 | 322 469.8 | 9.78 | **294 732** | **0.34** | 296 413 | 0.91 | 296 114.8 | 0.81 | 296 330 | 0.88 |
| | 10 ∗ 5 | 1 185 483 | 9.16 | **1 087 866** | **0.17** | 1 100 967 | 1.38 | 1 096 256 | 0.94 | 1 097 633 | 1.07 |
| | 5 ∗ 6 | 413 679.9 | 12.50 | **368 378.2** | **0.18** | 371 856.4 | 1.12 | 370 489.9 | 0.75 | 371 756.5 | 1.10 |
| | 8 ∗ 6 | 523 239.2 | 9.03 | **480 807.6** | **0.19** | 484 267.5 | 0.91 | 482 739.3 | 0.59 | 484 231.3 | 0.90 |
| | 10 ∗ 6 | 520 846.3 | 9.82 | **475 209.4** | **0.20** | 480 008.9 | 1.21 | 477 888.9 | 0.76 | 478 398.3 | 0.87 |
| | 5 ∗ 7 | 275 393.1 | 9.75 | 251 794 | 0.35 | 252 909.4 | 0.79 | **251 496.8** | **0.23** | 252 900.5 | 0.79 |
| | 8 ∗ 7 | 721 726.7 | 10.03 | **657 116** | **0.18** | 664 268.3 | 1.27 | 661 723 | 0.88 | 664 147.9 | 1.25 |
| | 10 ∗ 7 | 629 229.4 | 9.41 | **577 151.7** | **0.35** | 583 903.3 | 1.53 | 579 826 | 0.82 | 581 284.8 | 1.07 |
| | Mean | 1 195 954.6 | 9.96 | **1 093 776.3** | **0.22** | 1 103 748.9 | 1.13 | 1 099 996.1 | 0.75 | 1 101 063.1 | 0.94 |
| 500 | 5 ∗ 2 | 3 034 919 | 12.80 | **2 695 152** | **0.17** | 2 725 020 | 1.28 | 2 709 918 | 0.72 | 2 713 449 | 0.85 |
| | 8 ∗ 2 | 5 832 302 | 9.25 | **5 346 469** | **0.15** | 5 376 079 | 0.71 | 5 371 911 | 0.63 | 5 374 745 | 0.68 |
| | 10 ∗ 2 | 5 741 000 | 8.64 | **5 294 060** | **0.18** | 5 342 580 | 1.10 | 5 322 976 | 0.73 | 5 319 768 | 0.67 |
| | 5 ∗ 3 | 1 508 880 | 9.39 | **1 381 425** | **0.15** | 1 390 310 | 0.79 | 1 386 630 | 0.53 | 1 387 709 | 0.61 |
| | 8 ∗ 3 | 3 368 735 | 10.34 | **3 057 950** | **0.16** | 3 087 004 | 1.11 | 3 078 550 | 0.83 | 3 079 745 | 0.87 |
| | 10 ∗ 3 | 3 262 112 | 7.91 | **3 026 719** | **0.12** | 3 058 660 | 1.18 | 3 045 876 | 0.76 | 3 043 731 | 0.69 |
| | 5 ∗ 4 | 1 379 045 | 11.15 | **1 244 062** | **0.27** | 1 253 434 | 1.03 | 1 248 487 | 0.63 | 1 252 282 | 0.93 |
| | 8 ∗ 4 | 1 478 380 | 7.82 | **1 373 001** | **0.13** | 1 381 548 | 0.75 | 1 379 960 | 0.64 | 1 379 952 | 0.64 |
| | 10 ∗ 4 | 904 553.3 | 10.39 | **821 462.8** | **0.25** | 827 572.9 | 1.00 | 827 015 | 0.93 | 826 508.8 | 0.87 |
| | 5 ∗ 5 | 1 406 530 | 11.37 | **1 265 693** | **0.21** | 1 280 320 | 1.37 | 1 272 129 | 0.72 | 1 278 006 | 1.19 |
| | 8 ∗ 5 | 1 083 803 | 9.61 | **991 399.4** | **0.26** | 999 105.6 | 1.04 | 997 567 | 0.89 | 998 298.6 | 0.96 |
| | 10 ∗ 5 | 1 657 984 | 9.89 | **1 510 551** | **0.12** | 1 527 980 | 1.27 | 1 522 145 | 0.89 | 1 523 218 | 0.96 |
| | 5 ∗ 6 | 451 254 | 11.72 | **404 623.4** | **0.18** | 408 509 | 1.14 | 406 703.5 | 0.69 | 407 927.4 | 1.00 |
| | 8 ∗ 6 | 1 365 419 | 9.05 | **1 253 950** | **0.15** | 1 266 751 | 1.17 | 1 262 061 | 0.80 | 1 263 565 | 0.92 |
| | 10 ∗ 6 | 1 318 780 | 9.00 | **1 211 800** | **0.16** | 1 221 142 | 0.93 | 1 218 381 | 0.70 | 1 220 216 | 0.85 |
| | 5 ∗ 7 | 619 427.1 | 11.78 | **555 419.6** | **0.23** | 562 908.8 | 1.58 | 559 308.1 | 0.93 | 561 918.3 | 1.40 |
| | 8 ∗ 7 | 659 570.8 | 11.23 | **593 452.9** | **0.08** | 600 215 | 1.22 | 597 319.7 | 0.73 | 598 713.5 | 0.97 |
| | 10 ∗ 7 | 749 013.5 | 8.58 | **691 048** | **0.18** | 697 259.2 | 1.08 | 694 894.2 | 0.74 | 696 149.1 | 0.92 |
| | Mean | 1 990 094.9 | 9.99 | **1 817 679.9** | **0.18** | 1 833 688.8 | 1.10 | 1 827 879.5 | 0.75 | 1 829 216.8 | 0.89 |

results, the VNIG obtains the best results for most instances. The performance of the ES algorithm is second only to VNIG. The IGR algorithm performs worse than VNIG and ES in solving the DBFSP with balanced energy cost. Overall, the VNIG substantially outperforms the compared algorithms for solving the DBFSP_SDST with balanced energy cost. It may be that the proposed variable local search can better explore unknown neighborhoods and prevent the solution from falling into a local optimum.

Figs. 5 and 6 report the minimum and maximum values of 90 instances yielded by all the compared algorithms. Because the energy combustion cost value is large, the gap between the results of different algorithms is small when the results are displayed in a figure. Therefore, to show the gap between algorithms more clearly, we normalize the results using Eq. (26). The purpose is to reduce the values in equal proportion and to clearly show the differences. The minimum and maximum values are shown in Figs. 5–6 when $t = 2$ and $t = 3$, respectively. From Figs. 5–6, we know that the maximum values among all the algorithms are obtained by DABC, so it can be inferred that this algorithm has the worst performance in solving the DBFSP_SDST with balanced energy cost. CRO shows large fluctuations on some small instances and gradually stabilizes as the size of the instances increases. Meanwhile, we can conclude that the performance of ES is better than that of IGR. As a whole, the values obtained by the VNIG proposed in this paper are excellent in most cases, and we can consider that the performance of VNIG in solving DBFSP_SDST is the best among the compared algorithms.

To have a clear identification of the experimental results, we give the ANOVA of all the algorithms. As shown in Fig. 7, the means plots and interactions plots with 95% LSD intervals represent the average level and overall performance of the algorithms. From Fig. 7, it can be seen that the VNIG proposed in this paper is better than the compared algorithms. The performance of the ES algorithm is worse than that of VNIG but better than that of IGR. In addition, the performance of CRO and IGR is better than that of DABC. We can assume that the algorithm with outstanding local search capability can show superior performance in solving DBFSP_SDST.

To better demonstrate the convergence of the proposed algorithm, we chose scales of $100 \times 10 \times 3$, $200 \times 10 \times 3$, $300 \times 10 \times 3$, and $500 \times 8 \times 5$ as examples and plotted the evolutionary curves. The algorithms are running at $t = 10$ when the instance size is small, and the large instance is run at $t = 20$. The result is shown in Fig. 8, where the $X$-axis represents the running time (unit: seconds) and the $Y$-axis represents the total energy consumption cost. All comparison algorithms are listed by different lines. According to the analysis above, we can conclude that the performance of DABC is poor. Therefore, to better show the evolutionary trend of other algorithms, we have removed DABC in Fig. 8. From the four instances, the convergence of the proposed algorithm VNIG is the most rapid among the four algorithms. The VNIG can obtain an excellent initial solution and obtain the best result among the compared algorithms. From Fig. 8, we observe that VNIG can always get excellent solutions in different sizes of examples, which further demonstrates the effectiveness of the VNIG algorithm in solving the DBFSP_SDST with balanced energy cost.

**Remarks.** As the above experimental results and analysis show, the proposed VNIG algorithm is an effective algorithm for solving the DBFSP_SDST with balanced energy cost. The reasons can be concluded as follows. (1) The MME_en strategy combining the MM algorithm and NEH2_en makes the algorithm obtain an excellent initial solution, and a high-quality initial solution is of great importance for the improvement of the solution. (2) Several local search strategies based on energy consumption cost can enhance the neighborhood search ability to further improve the quality of the solution. (3) The variable neighborhood search strategy based on the learning selection method is the key operation that enables increasing the diversity of VNIG by avoiding the solution from falling into local optima in the large-scale instance. Based on the above analyses, the effective neighborhood search operations and the variable neighborhood search strategy based on the learning selection method are reasons for the outstanding performance of VNIG.

**Table 11**
Energy consumption costs of the compared algorithms when $t = 3$.

| Job | m ∗ f | DABC[2018] | | VNIG | | IGR[2020] | | ES[2022] | | CRO[2017] | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | **avg** | RPI | **avg** | RPI | **avg** | RPI | **avg** | RPI | **avg** | RPI |
| 100 | 5 ∗ 2 | 467 826.8 | 10.41 | **425 341.6** | **0.38** | 431 018 | 1.72 | 427 832.8 | 0.97 | 428 566.6 | 1.14 |
| | 8 ∗ 2 | 625 155.6 | 11.14 | **564 759.6** | **0.41** | 570 072 | 1.35 | 569 608.8 | 1.27 | 569 234.7 | 1.20 |
| | 10 ∗ 2 | 228 788.2 | 10.23 | **207 919.6** | **0.18** | 210 942.9 | 1.63 | 210 496.3 | 1.42 | 210 500.5 | 1.42 |
| | 5 ∗ 3 | 132 459.2 | 14.10 | **116 668.6** | **0.50** | 117 678.6 | 1.37 | 117 285.9 | 1.03 | 117 887 | 1.55 |
| | 8 ∗ 3 | 461 279 | 10.50 | **418 654.8** | **0.29** | 423 876.1 | 1.54 | 422 433.9 | 1.19 | 422 721.2 | 1.26 |
| | 10 ∗ 3 | 458 603.1 | 8.33 | **424 254.6** | **0.21** | 427 898.1 | 1.07 | 426 746.2 | 0.80 | 428 246.8 | 1.16 |
| | 5 ∗ 4 | 151 309.6 | 10.47 | **137 507.2** | **0.39** | 138 684.7 | 1.25 | 138 037.8 | 0.78 | 139 238.6 | 1.65 |
| | 8 ∗ 4 | 424 778.5 | 9.02 | **391 479** | **0.47** | 394 059.1 | 1.14 | 393 176 | 0.91 | 395 860.5 | 1.60 |
| | 10 ∗ 4 | 277 418.6 | 9.34 | **255 351.6** | **0.65** | 257 738.2 | 1.59 | 256 707.8 | 1.18 | 257 217 | 1.38 |
| | 5 ∗ 5 | 56 551.3 | 10.04 | **51 636.9** | **0.48** | 52 190.6 | 1.55 | 51 984.4 | 1.15 | 52 424.7 | 2.01 |
| | 8 ∗ 5 | 325 389.4 | 10.91 | **294 722.8** | **0.46** | 296 059.8 | 0.92 | 295 994 | 0.89 | 298 008.9 | 1.58 |
| | 10 ∗ 5 | 400 656.2 | 11.10 | **362 520.2** | **0.53** | 364 816.1 | 1.16 | 363 976.5 | 0.93 | 367 335.1 | 1.86 |
| | 5 ∗ 6 | 174 907 | 11.89 | **157 647.3** | **0.84** | 159 328.4 | 1.92 | 158 251.8 | 1.23 | 159 831 | 2.24 |
| | 8 ∗ 6 | 308 722.3 | 10.98 | **280 101.7** | **0.69** | 283 610.1 | 1.95 | 282 956.4 | 1.72 | 285 363.6 | 2.58 |
| | 10 ∗ 6 | 329 940 | 10.12 | **301 292.4** | **0.56** | 303 979.2 | 1.46 | 302 140.1 | 0.84 | 305 489.8 | 1.96 |
| | 5 ∗ 7 | 121 923.3 | 12.38 | **109 167.8** | **0.62** | 110 655.6 | 1.99 | 109 634.3 | 1.05 | 111 490.3 | 2.76 |
| | 8 ∗ 7 | 138 743.3 | 10.51 | **126 586.1** | **0.82** | 127 763.1 | 1.76 | 126 958.7 | 1.12 | 128 080.6 | 2.01 |
| | 10 ∗ 7 | 167 656.1 | 9.87 | **153 135.3** | **0.35** | 155 127.4 | 1.66 | 154 997.9 | 1.57 | 155 026 | 1.59 |
| | Mean | 941 746 | 9.29 | **863 973.6** | **0.27** | 868 704.8 | 0.82 | 866 442.4 | 0.55 | 867 898.8 | 0.72 |
| 200 | 5 ∗ 2 | 325 992.3 | 10.6 | **296 985.3** | **0.5** | 299 694.9 | 1.5 | 298 719.1 | 1.1 | 300 022.2 | 1.7 |
| | 8 ∗ 2 | 1 125 525 | 7.97 | **1 045 120** | **0.26** | 1 053 698 | 1.08 | 1 050 006 | 0.73 | 1 051 067 | 0.83 |
| | 10 ∗ 2 | 982 253.7 | 7.69 | **914 338.8** | **0.24** | 923 719.8 | 1.27 | 918 720.6 | 0.72 | 918 928.8 | 0.74 |
| | 5 ∗ 3 | 470 054.9 | 10.71 | **426 190.7** | **0.38** | 431 085 | 1.53 | 428 675.3 | 0.96 | 429 987.6 | 1.27 |
| | 8 ∗ 3 | 1 294 144 | 8.84 | **1 192 088** | **0.26** | 1 204 878 | 1.34 | 1 200 812 | 0.99 | 1 202 799 | 1.16 |
| | 10 ∗ 3 | 537 984.1 | 8.77 | **496 077.9** | **0.30** | 502 092.5 | 1.52 | 498 725.9 | 0.84 | 498 437.7 | 0.78 |
| | 5 ∗ 4 | 418 311.1 | 9.20 | **384 177** | **0.29** | 387 628.5 | 1.19 | 386 523.2 | 0.91 | 387 346.1 | 1.12 |
| | 8 ∗ 4 | 230 949 | 9.83 | **211 052.3** | **0.37** | 213 284.8 | 1.43 | 212 079.6 | 0.86 | 212 832 | 1.21 |
| | 10 ∗ 4 | 383 408.4 | 8.90 | **353 263.5** | **0.33** | 357 599.9 | 1.56 | 355 466.5 | 0.96 | 356 388.4 | 1.22 |
| | 5 ∗ 5 | 173 264.5 | 10.96 | 157 079.4 | 0.56 | 158 737.4 | 1.66 | **156 880.2** | **0.47** | 158 404.5 | 1.44 |
| | 8 ∗ 5 | 514 237.7 | 10.14 | **467 843.6** | **0.21** | 474 849 | 1.71 | 471 759 | 1.05 | 473 869.2 | 1.50 |
| | 10 ∗ 5 | 554 673.8 | 8.31 | **514 401.7** | **0.45** | 518 897.7 | 1.33 | 516 644.2 | 0.89 | 518 415 | 1.23 |
| | 5 ∗ 6 | 339 006.4 | 11.93 | **304 192.4** | **0.44** | 308 969.4 | 2.02 | 306 773.8 | 1.29 | 308 956 | 2.01 |
| | 8 ∗ 6 | 204 151.6 | 10.80 | **185 056.2** | **0.44** | 187 744.3 | 1.90 | 185 890.9 | 0.89 | 186 445.8 | 1.19 |
| | 10 ∗ 6 | 252 926.8 | 10.75 | **229 300.7** | **0.40** | 234 570 | 2.71 | 231 703.7 | 1.45 | 231 773.8 | 1.48 |
| | 5 ∗ 7 | 159 619.2 | 10.52 | **144 825.4** | **0.28** | 146 096.2 | 1.16 | 145 332 | 0.63 | 146 629 | 1.53 |
| | 8 ∗ 7 | 476 604.2 | 11.00 | **431 179.5** | **0.42** | 434 615.4 | 1.22 | 434 547.7 | 1.20 | 434 739.3 | 1.25 |
| | 10 ∗ 7 | 214 161.3 | 8.69 | **197 772.4** | **0.37** | 200 347.8 | 1.68 | 199 427.2 | 1.21 | 199 985.5 | 1.50 |
| | Mean | 490 075.0 | 9.7 | **450 221.2** | **0.3** | 455 224.3 | 1.5 | 452 998.1 | 1.0 | 453 941.5 | 1.3 |
| 300 | 5 ∗ 2 | 941 746 | 9.29 | 866 442.4 | 0.55 | 868 704.8 | 0.82 | **863 973.6** | **0.27** | 867 898.8 | 0.72 |
| | 8 ∗ 2 | 701 815.7 | 10.38 | **637 590.6** | **0.28** | 644 705.4 | 1.40 | 640 490.4 | 0.74 | 641 406.7 | 0.88 |
| | 10 ∗ 2 | 3 132 964 | 9.76 | **2 858 554** | **0.14** | 2 886 949 | 1.14 | 2 876 017 | 0.75 | 2 879 764 | 0.89 |
| | 5 ∗ 3 | 379 742.9 | 10.31 | **344 559.6** | **0.09** | 347 949.7 | 1.08 | 346 370.1 | 0.62 | 347 019.4 | 0.81 |
| | 8 ∗ 3 | 467 052.3 | 7.84 | **433 953.2** | **0.20** | 437 361 | 0.99 | 435 883.9 | 0.64 | 436 375 | 0.76 |
| | 10 ∗ 3 | 579 078.8 | 7.63 | **539 634.8** | **0.30** | 543 557.1 | 1.03 | 541 669.2 | 0.68 | 542 583.5 | 0.85 |
| | 5 ∗ 4 | 337 962.3 | 9.58 | **309 215** | **0.26** | 310 980.6 | 0.84 | 310 369.7 | 0.64 | 310 974.7 | 0.83 |
| | 8 ∗ 4 | 567 404 | 9.56 | **519 665** | **0.34** | 523 827.6 | 1.14 | 523 011.4 | 0.98 | 523 153.9 | 1.01 |
| | 10 ∗ 4 | 740 196.4 | 8.77 | **683 043.1** | **0.37** | 690 288.7 | 1.44 | 688 216.2 | 1.13 | 689 495.2 | 1.32 |
| | 5 ∗ 5 | 376 827.5 | 13.64 | **333 888.5** | **0.69** | 339 052.8 | 2.25 | 337 795 | 1.87 | 338 984.7 | 2.23 |
| | 8 ∗ 5 | 475 847.4 | 9.88 | **434 119.3** | **0.24** | 437 751.7 | 1.08 | 436 857.7 | 0.87 | 437 476.6 | 1.02 |
| | 10 ∗ 5 | 1 783 114 | 9.33 | **1 637 294** | **0.38** | 1 655 810 | 1.52 | 1 651 494 | 1.26 | 1 654 953 | 1.47 |
| | 5 ∗ 6 | 170 936.2 | 11.17 | **154 385.7** | **0.40** | 155 873.7 | 1.37 | 155 033.7 | 0.82 | 155 863.6 | 1.36 |
| | 8 ∗ 6 | 281 307.5 | 10.57 | **255 280.1** | **0.34** | 258 539.6 | 1.62 | 257 350.5 | 1.15 | 257 870.1 | 1.36 |
| | 10 ∗ 6 | 1 003 844 | 8.81 | **925 541.9** | **0.33** | 935 804.3 | 1.44 | 930 918.9 | 0.91 | 934 280.6 | 1.27 |
| | 5 ∗ 7 | 371 981.5 | 13.13 | **330 188.1** | **0.42** | 335 771.2 | 2.12 | 334 103.1 | 1.61 | 335 576 | 2.06 |
| | 8 ∗ 7 | 635 747.9 | 8.58 | **587 767.1** | **0.39** | 594 072.2 | 1.46 | 590 398.7 | 0.84 | 592 908.7 | 1.27 |
| | 10 ∗ 7 | 723 764.4 | 10.06 | **659 553.4** | **0.30** | 668 182.8 | 1.61 | 664 044 | 0.98 | 667 506.1 | 1.51 |
| | Mean | 759 518.5 | 9.9 | **694 900.4** | **0.3** | 701 954.6 | 1.4 | 699 248.1 | 0.9 | 700 782.8 | 1.2 |
| 400 | 5 ∗ 2 | 583 960.7 | 10.54 | **529 026.7** | **0.14** | 532 437.1 | 0.79 | 531 209.3 | 0.56 | 531 056.6 | 0.53 |
| | 8 ∗ 2 | 2 461 379 | 8.98 | **2 262 161** | **0.16** | 2 276 884 | 0.81 | 2 271 773 | 0.59 | 2 273 896 | 0.68 |
| | 10 ∗ 2 | 1 304 584 | 8.10 | **1 208 548** | **0.14** | 1 216 008 | 0.76 | 1 213 446 | 0.55 | 1 214 174 | 0.61 |
| | 5 ∗ 3 | 507 808.4 | 10.23 | **461 846** | **0.25** | 464 953.4 | 0.92 | 463 673 | 0.65 | 464 849 | 0.90 |
| | 8 ∗ 3 | 1 275 307 | 9.26 | **1 169 199** | **0.17** | 1 179 110 | 1.02 | 1 175 848 | 0.74 | 1 177 687 | 0.90 |
| | 10 ∗ 3 | 1 610 226 | 9.37 | **1 474 477** | **0.15** | 1 491 004 | 1.27 | 1 482 670 | 0.71 | 1 484 177 | 0.81 |
| | 5 ∗ 4 | 667 321.8 | 10.45 | **605 373.9** | **0.20** | 610 239.5 | 1.00 | 608 266.6 | 0.68 | 610 083.6 | 0.98 |
| | 8 ∗ 4 | 695 785.6 | 10.90 | **628 989.9** | **0.26** | 634 453.3 | 1.13 | 632 783.2 | 0.86 | 633 322.9 | 0.95 |
| | 10 ∗ 4 | 817 006.2 | 9.34 | **749 355.8** | **0.29** | 755 522.8 | 1.11 | 754 996.9 | 1.04 | 754 738.4 | 1.01 |
| | 5 ∗ 5 | 275 469.9 | 10.95 | **248 654.3** | **0.15** | 250 493.7 | 0.89 | 249 712.8 | 0.57 | 250 306.3 | 0.81 |
| | 8 ∗ 5 | 393 290.6 | 9.49 | **359 798.8** | **0.17** | 363 531 | 1.20 | 362 107.4 | 0.81 | 362 714.9 | 0.98 |

(*continued on next page*)

**Table 11** (*continued*).

| Job | m ∗ f | DABC[2018] | | VNIG | | IGR[2020] | | ES[2022] | | CRO[2017] | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | **avg** | RPI | **avg** | RPI | **avg** | RPI | **avg** | RPI | **avg** | RPI |
| | 10 ∗ 5 | 1 238 094 | 9.68 | **1 131 174** | **0.21** | 1 146 014 | 1.52 | 1 142 242 | 1.19 | 1 144 837 | 1.42 |
| | 5 ∗ 6 | 171 631.6 | 11.37 | **154 475.8** | **0.24** | 155 874.6 | 1.15 | 155 557 | 0.94 | 155 896 | 1.16 |
| | 8 ∗ 6 | 792 043 | 9.65 | **725 047.4** | **0.37** | 731 225.4 | 1.23 | 729 667.3 | 1.01 | 731 034.8 | 1.20 |
| | 10 ∗ 6 | 374 784.4 | 9.18 | **343 837.7** | **0.16** | 347 231.1 | 1.15 | 345 599 | 0.68 | 346 391.3 | 0.91 |
| | 5 ∗ 7 | 407 196.8 | 13.63 | **359 277.7** | **0.26** | 363 750.3 | 1.50 | 362 111.1 | 1.05 | 363 950 | 1.56 |
| | 8 ∗ 7 | 577 463.9 | 8.94 | **531 036.4** | **0.18** | 535 927.2 | 1.11 | 533 345.6 | 0.62 | 534 897.5 | 0.91 |
| | 10 ∗ 7 | 777 184.7 | 9.82 | **709 367.9** | **0.23** | 718 819.4 | 1.57 | 715 122.8 | 1.05 | 717 131.8 | 1.33 |
| | Mean | 829 474.3 | 10.0 | **758 424.9** | **0.2** | 765 193.3 | 1.1 | 762 785.1 | 0.8 | 763 952.5 | 1.0 |
| 500 | 5 ∗ 2 | 1 587 790 | 11.70 | **1 423 683** | **0.16** | 1 432 393 | 0.77 | 1 429 787 | 0.59 | 1 430 610 | 0.65 |
| | 8 ∗ 2 | 3 548 259 | 9.11 | **3 255 493** | **0.11** | 3 278 817 | 0.83 | 3 272 791 | 0.64 | 3 271 848 | 0.61 |
| | 10 ∗ 2 | 1 065 611 | 7.86 | **988 597.1** | **0.06** | 996 233.2 | 0.84 | 993 819.2 | 0.59 | 992 573.9 | 0.47 |
| | 5 ∗ 3 | 629 016.4 | 12.66 | **559 666.9** | **0.24** | 565 092.4 | 1.21 | 562 662.5 | 0.78 | 563 104.1 | 0.86 |
| | 8 ∗ 3 | 2 025 654 | 9.85 | **1 846 809** | **0.15** | 1 867 968 | 1.30 | 1 855 974 | 0.65 | 1 858 029 | 0.76 |
| | 10 ∗ 3 | 2 298 302 | 8.60 | **2 118 682** | **0.11** | 2 137 671 | 1.01 | 2 130 157 | 0.66 | 2 131 341 | 0.71 |
| | 5 ∗ 4 | 1 162 130 | 9.46 | **1 063 975** | **0.21** | 1 072 291 | 0.99 | 1 068 025 | 0.59 | 1 071 104 | 0.88 |
| | 8 ∗ 4 | 1 188 833 | 8.37 | **1 098 679** | **0.15** | 1 109 251 | 1.12 | 1 104 385 | 0.67 | 1 105 630 | 0.79 |
| | 10 ∗ 4 | 1 740 254 | 8.19 | **1 611 268** | **0.17** | 1 624 682 | 1.00 | 1 619 200 | 0.66 | 1 620 554 | 0.75 |
| | 5 ∗ 5 | 497 868.1 | 10.79 | **450 191.4** | **0.18** | 452 726.5 | 0.75 | 452 125.5 | 0.61 | 452 961.3 | 0.80 |
| | 8 ∗ 5 | 1 160 409 | 9.10 | **1 065 147** | **0.15** | 1 077 804 | 1.34 | 1 072 625 | 0.85 | 1 074 600 | 1.04 |
| | 10 ∗ 5 | 1 738 340 | 8.27 | **1 608 410** | **0.18** | 1 620 556 | 0.94 | 1 617 766 | 0.76 | 1 620 021 | 0.90 |
| | 5 ∗ 6 | 1 038 196 | 12.77 | **923 566.2** | **0.32** | 936 114 | 1.68 | 929 805.3 | 0.99 | 935 445 | 1.61 |
| | 8 ∗ 6 | 852 840.8 | 10.07 | **776 970.8** | **0.28** | 786 746.4 | 1.54 | 783 681.6 | 1.14 | 784 754.3 | 1.28 |
| | 10 ∗ 6 | 860 416.1 | 8.34 | **795 478.1** | **0.16** | 801 955.9 | 0.98 | 798 977 | 0.60 | 800 601.9 | 0.81 |
| | 5 ∗ 7 | 210 182.3 | 11.91 | **188 287.2** | **0.25** | 190 179.5 | 1.26 | 189 267.5 | 0.77 | 190 131.9 | 1.24 |
| | 8 ∗ 7 | 1 068 845 | 9.77 | **976 054.1** | **0.24** | 986 110.8 | 1.28 | 982 685.1 | 0.92 | 985 320.9 | 1.20 |
| | 10 ∗ 7 | 828 744.9 | 10.02 | **756 043.4** | **0.37** | 766 575.2 | 1.77 | 762 206.9 | 1.19 | 764 077.1 | 1.44 |
| | Mean | 1 305 649.5 | 9.8 | **1 194 833.4** | **0.2** | 1 205 731.5 | 1.1 | 1 201 441.1 | 0.8 | 1 202 928.2 | 0.9 |

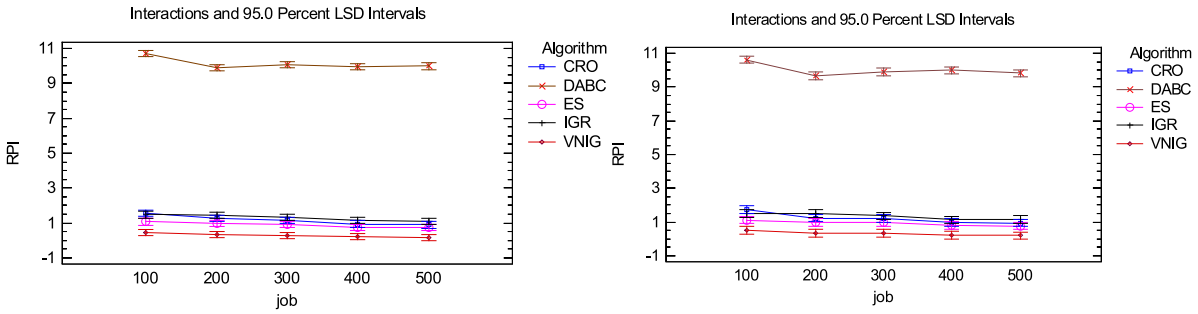



**Fig. 7.** Confidence intervals for the compared algorithms.

### 5.7. Friedman test

Friedman test is a nonparametric test for the presence of significant differences in multiple overall distributions [28]. This test first assumes that multiple paired samples from multiple overall distributions are not significantly different. The hypothesis will be accepted when the value of $p$ is not less than 0.05. If the hypothesis is rejected, the groups of samples are considered to be significantly different.

We analyze 90 examples with situations of $t = 2$ and $t = 3$. The results are shown in Tables 12–13. According to the Friedman test (confidence level $\alpha = 0.050$), the *p value* is 0.000, which indicates that the compared algorithms are significantly different. By observing the results, the rank values of our proposed algorithm VNIG are the smallest (1.00 and 1.01). VNIG also had the smallest mean values (0.285 and 0.311) and the smallest minimum values (0.08 and 0.06). Meanwhile, the VNIG obtains the best standard deviation values (0.1409 and 0.1569) and maximum values (0.79 and 0.84). From an overall perspective, we can deduce that the VNIG is the most stable algorithm with CPU = 2 and CPU = 3. And the VNIG demonstrates excellent performance in solving resource-balance DBFSP_SDST problems.

**Table 12**
Results achieved by Friedman test (confidence level $\alpha = 0.050$) when $t = 2$.

| Algorithms | Ranks | CN | Mean | Std. Deviation | Min | Max |
|---|---|---|---|---|---|---|
| VNIG | **1.00** | 90 | **0.285** | **0.1409** | **0.08** | **0.79** |
| IGR | 3.79 | 90 | 1.295 | 0.3357 | 0.71 | 2.40 |
| DABC | 5.00 | 90 | 10.135 | 1.1676 | 7.82 | 12.95 |
| CRO | 3.09 | 90 | 1.150 | 0.3889 | 0.61 | 2.37 |
| ES | 2.12 | 90 | 0.896 | 0.2327 | 0.35 | 1.66 |
| *p value* | 0.000 | | | | | |

**Table 13**
Results achieved by Friedman test (confidence level $\alpha = 0.050$) when $t = 3$.

| Algorithms | Ranks | CN | Mean | Std. Deviation | Min | Max |
|---|---|---|---|---|---|---|
| VNIG | **1.01** | 90 | **0.311** | **0.1569** | **0.06** | **0.84** |
| IGR | 3.81 | 90 | 1.325 | 0.3733 | 0.75 | 2.71 |
| DABC | 5.00 | 90 | 10.008 | 1.3894 | 7.63 | 14.10 |
| CRO | 3.12 | 90 | 1.700 | 1.4348 | 0.34 | 5.92 |
| ES | 2.07 | 90 | 0.915 | 0.2802 | 0.55 | 1.87 |
| *p value* | 0.000 | | | | | |

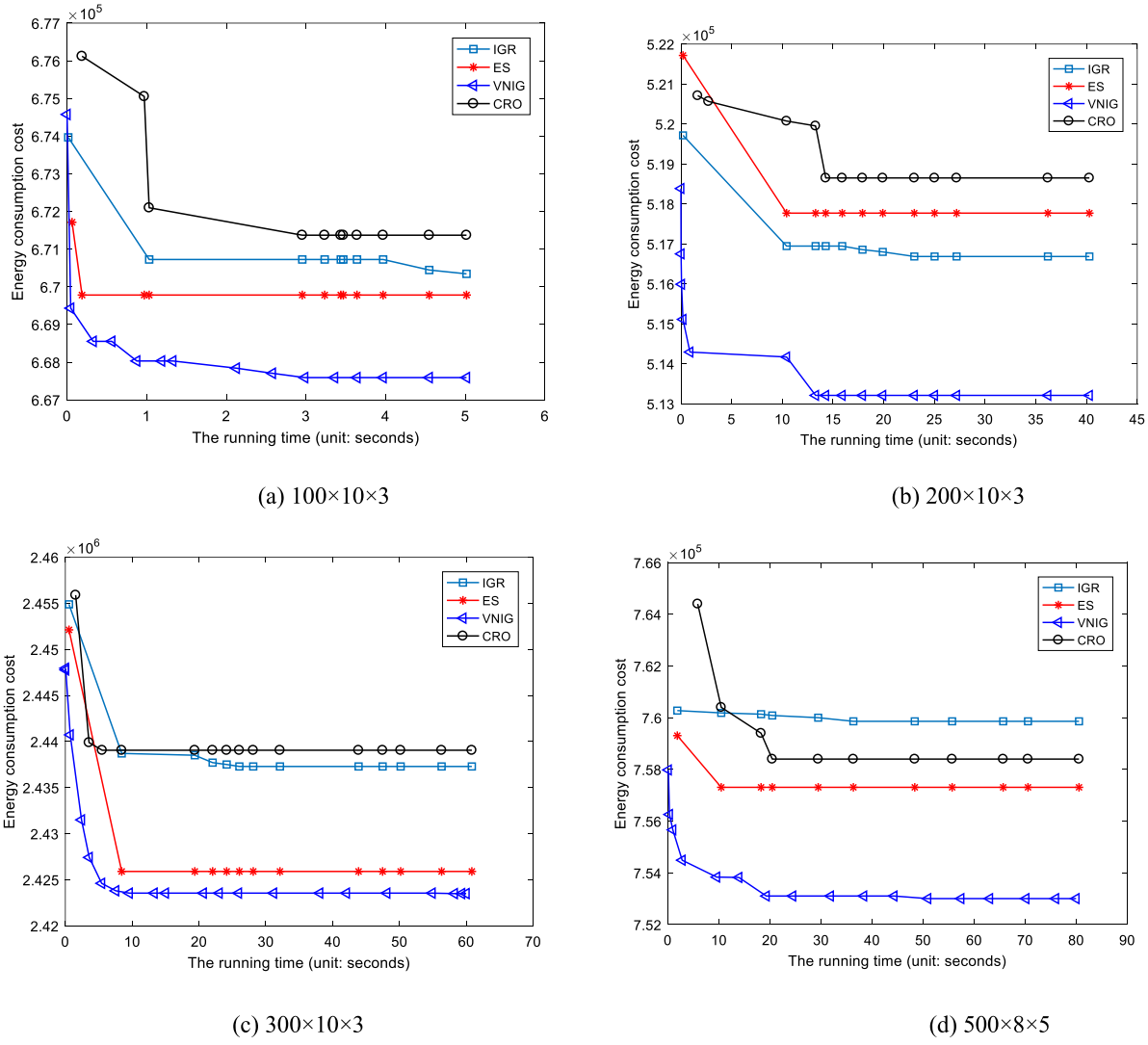



**Fig. 8.** Evolutionary Curve for the compared algorithms.

## 6. Conclusions

Most existing algorithms in the literature address the discrete flowshop scheduling problem without considering the blocking constraint. Thus, in this study, we formulate the mathematic model of DBFSP_SDST with balanced energy cost and propose a local search with the learning_based variable neighborhood search strategy that seeds the initial solution using MME_en based on the MM strategy and NEH2_en. In the improved variable neighborhood search strategy (IVNS), three different local search methods are used to perturb the job sequence. In the experimental part, the effectiveness of the proposed strategies is verified. The performance of the proposed algorithm is empirically evaluated on 90 instances of DBFSP_SDST. The proposed MME_en and NEH2_en are compared, and the effectiveness of the learning-based variable neighborhood strategy is verified. In addition, comparison results between the proposed VNIG algorithm and the existing four compared algorithms are reported. Through the above simulation experiments, the proposed VNIG shows superior performance compared with state-of-the-art algorithms.

There are several opportunities for future research on DBFSP_SDST. First, the three local search strategies are randomly selected in this study. It might be desirable to develop a self-adaptive mechanism to select the local search from them to improve the exploration capability of the algorithm. Second, some local search strategies can be developed to further reduce the computational complexity of the algorithm. Third, uncertainties related to machine breakdowns, wrong operations, and changes in due date should also be considered when tackling DBFSP_SDST. Last, to remedy the current unsatisfactory situation regarding the experiment replication and comparison, the termination criterion of an algorithm should be appropriately designed, such as replacing the maximal elapsed CPU time with the number of fitness evaluations, to fulfill the comparisons among heuristics algorithms for the flowshop scheduling problem.

## CRediT authorship contribution statement

**Xue Han:** Conception or design of the work, Acquisition, Analysis, Interpretation of data, Writing – original draft, Writing – review & editing. **Yuyan Han:** Conception or design of the work, Acquisition, Analysis, Interpretation of data, Writing – original draft, Writing – review & editing. **Biao Zhang:** Writing – review & editing, Interpretation of data. **Haoxiang Qin:** Writing – review & editing, Interpretation of data. **Junqing Li:** Writing – review & editing, Interpretation of data. **Yiping Liu:** Writing – review & editing, Interpretation of data. **Dunwei Gong:** Writing – review & editing, Interpretation of data.

## 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

The authors do not have permission to share data.

## Acknowledgments

This work was jointly supported by the National Natural Science Foundation of China under grant numbers 61803192, 62106073, 61973203, 61966012, 61773246, and 71533001. We are grateful for Guangyue Youth Scholar Innovation Talent Program support received from Liaocheng University, the Youth Innovation Talent Introduction and Education Program support received from Shandong Province Colleges and Universities, 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. Xue Han has approved the final version to be published and agrees to be accountable for all aspects of the work in ensuring that questions related to the accuracy or integrity of any part of the work are appropriately investigated and resolved.

## Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.asoc.2022.109502.

## References

[1] J. Liang, Y. Wang, Z.H. Zhang, et al., Energy efficient production planning and scheduling problem with processing technology selection, Comput. Ind. Eng. 132 (2019) 260–270.
[2] R. Ruiz, B. Naderi, The distributed permutation flowshop scheduling problem, Comput. Oper. Res. 37 (4) (2010) 754–768.
[3] Y. Han, J. Li, H. Sang, et al., Discrete evolutionary multiobjective optimization for energy-efficient blocking flow shop scheduling with setup time, Appl. Soft Comput. 93 (2020) 106343.
[4] S. Parthasarathy, C. Rajendran, An experimental evaluation of heuristics for scheduling in a real-life flowshop with sequence-dependent setup times of jobs, Int. J. Prod. Econ. 49 (3) (1997) 255–263.
[5] H. Guo, H. Sang, B. Zhang, L. Meng, L. Liu, An effective metaheuristic with a differential flight strategy for the distributed permutation flowshop scheduling problem with sequence-dependent setup times, Knowl.-Based Syst. 242 (1) (2022) 108328.
[6] Korhan Karabulut, Hande Öztop, Damla Kizilay, M. Fatih Tasgetiren, Levent Kandiller, An evolution strategy approach for the distributed permutation flowshop scheduling problem with sequence-dependent setup times, Comput. Oper. Res. 142 (2) (2022) 105733.
[7] C.Y. Cheng, P. Pourhejazy, K.C. Ying, S.Y. Huang, New benchmark algorithm for minimizing total completion time in blocking flowshops with sequence-dependent setup times, Appl. Soft Comput. 104 (2021) 107229.
[8] M.A. Hakim Newton, V. Riahi, K. Su, A Sattar, Scheduling blocking flowshops with setup times via constraint guided and accelerated local search, Comput. Oper. Res. 109 (2019) 64–76.
[9] J. Dong, C.M. Ye, Green scheduling of distributed two-stage reentrant hybrid flow shop considering distributed energy resources and energy storage system, Comput. Ind. Eng. 169 (2022) 108146.
[10] Z.Q. Zhang, B. Qian, R. Hu, et al., A matrix-cube-based estimation of distribution algorithm for the distributed assembly permutation flow-shop scheduling problem, Swarm Evol. Comput. 60 (2021) 100785.
[11] O.A. Arık, Artificial bee colony algorithm including some components of iterated greedy algorithm for permutation flow shop scheduling problems, Neural Comput. Appl. 33 (2021) 3469–3486.
[12] C. Lu, Q. Liu, B. Zhang, L.J. Yin, A Pareto-based hybrid iterated greedy algorithm for energy-efficient scheduling of distributed hybrid flowshop, Expert Syst. Appl. 204 (2022) 0957–4174.
[13] 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.
[14] S. Hatami, R. Ruiz, C. Andres-Romano, Heuristics and metaheuristics for the distributed assembly permutation flowshop scheduling problem with sequence dependent setup times, Int. J. Prod. Econ. 169 (2015) 76–88.
[15] B. Naderi, R. Ruiz, A scatter search algorithm for the distributed permutation flowshop scheduling problem, European J. Oper. Res. 239 (2) (2014) 323–334.
[16] H. Bargaoui, O. Belkahla Driss, Khaléd Ghédira, A novel chemical reaction optimization for the distributed permutation flowshop scheduling problem with makespan criterion, Comput. Ind. Eng. 111 (2017) 239–250.
[17] V. Fernandez-Viagas, J.M. Framinan, A bounded-search iterated greedy algorithm for the distributed permutation flowshop scheduling problem, Int. J. Prod. Res. 53 (4) (2015) 1111–1123.
[18] R. Ruiz, Q.K. Pan, B. Naderi, Iterated Greedy methods for the distributed permutation flowshop scheduling problem, Omega 83 (2019) 213–222.
[19] T. Meng, Q.K. Pan, L. Wang, A distributed permutation flowshop scheduling problem with the customer order constraint, Knowl.-Based Syst. 184 (2019) 104894.1-104894.17.
[20] V. Fernandez-Viagas, P. Perez-Gonzalez, J.M. Framinan, The distributed permutation flow shop to minimize the total flowtime, Comput. Ind. Eng. 118 (2018) 464–477.
[21] Q.K. Pan, L. Gao, L. Wang, et al., Effective heuristics and metaheuristics to minimize total flowtime for the distributed permutation flowshop problem, Expert Syst. Appl. 124 (2019) 309–324.
[22] Z.Q. Zhang, B. Qian, R. Hu, et al., A matrix-cube-based estimation of distribution algorithm for the distributed assembly permutation flow-shop scheduling problem, Swarm Evol. Comput. 60 (2021) 100785.
[23] S. Parthasarathy, C. Rajendran, An experimental evaluation of heuristics for scheduling in a real-life flowshop with sequence-dependent setup times of jobs, Int. J. Prod. Econ. 49 (3) (1997) 255–263.
[24] Mirabi Mohammad, Ant colony optimization technique for the sequence-dependent flowshop scheduling problem, Int. J. Adv. Manuf. Technol. 55 (1–4) (2010) 317–326.
[25] R. Vanchipura, R. Sridharan, A.S. Babu, Improvement of constructive heuristics using variable neighbourhood descent for scheduling a flow shop with sequence dependent setup time, J. Manuf. Syst. 33 (1) (2014) 65–75.
[26] M.S. Nagano, H.H. Miyata, D.C. Araújo, A constructive heuristic for total flowtime minimization in a no-wait flowshop with sequence-dependent setup times, J. Manuf. Syst. 36 (2015) 224–230.
[27] A. Sioud, C. Gagne, Enhanced migrating birds optimization algorithm for the permutation flow shop problem with sequence dependent setup times, European J. Oper. Res. 264 (1) (2018) 66–73.
[28] 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. (2020) 100742.
[29] J.P. Huang, Q.K. Pan, Z.H. Miao, et al., Effective constructive heuristics and discrete bee colony optimization for distributed flowshop with setup times, Eng. Appl. Artif. Intell. 97 (2021) 104016.
[30] G. Zhang, K. Xing, F. Cao, Discrete differential evolution algorithm for distributed blocking flowshop scheduling with makespan criterion, Eng. Appl. Artif. Intell. 76 (2018) 96–107.
[31] Z. Shao, D. Pi, W. Shao, Hybrid enhanced discrete fruit fly optimization algorithm for scheduling blocking flow-shop in distributed environment, Expert Syst. Appl. 145 (2019) 113147.
[32] F. Zhao, L. Zhao, L. Wang, et al., An ensemble discrete differential evolution for the distributed blocking flowshop scheduling with minimizing Makespan criterion, Expert Syst. Appl. 160 (2020) 113678.
[33] S. Chen, Q.K. Pan, X.L. Hu, et al., An Iterated Greedy Algorithm for Distributed Blocking Flowshop Problems with Makespan Minimization, IEEE, 2020, pp. 1536–1541.
[34] J.J. Wang, L. Wang, A knowledge-based cooperative algorithm for energy-efficiency scheduling of distributed flow-shop, IEEE Trans. Syst. Man Cybern. Syst. (2018) 1–15.
[35] F.L. Rossi, M.S. Nagano, Heuristics and iterated greedy algorithms for the distributed mixed no-idle flowshop with sequence-dependent setup times, Comput. Ind. Eng. 157 (2021) 107337.
[36] J. Pan, W. Zou, J. Duan, A discrete artificial bee colony for distributed permutation flowshop scheduling problem with total flow time minimization, in: 2018 37th Chinese Control Conference, CCC, Wuhan, 2018, pp. 8379–8383.
[37] M. Mashaei, B. Lennartson, Energy reduction in a pallet-constrained flow shop through on-off control of idle machines, IEEE Trans. Autom. Sci. Eng. 10 (1) (2013) 45–56.

[38] O. Masmoudi, A. Yalaoui, Y. Ouazene, et al., Lot-sizing in flow-shop with energy consideration for sustainable manufacturing systems, IFAC Papersonline 48 (3) (2015) 727–732.
[39] Tang. D., Min. D., M.A. Salido, et al., Energy-efficient dynamic scheduling for a flexible flow shop using an improved particle swarm optimization, Comput. Ind. 81 (2016) 82–95.
[40] O. Masmoudi, A. Yalaoui, Y. Ouazene, et al., Solving a capacitated flow-shop problem with minimizing total energy costs, Int. J. Adv. Manuf. Technol. 90 (2017) 2655–2667.
[41] G. Wang, X. Li, L. Gao, et al., An effective multiobjective whale swarm algorithm for energy-efficient scheduling of distributed welding flow shop, Ann. Oper. Res. 310 (2022) 223–255.
[42] X.L. Ding, J. Zhu, C. Liu, Lagrangian relaxation algorithms for hybrid flow-shop scheduling problems with energy saving, Adv. Mater. Res. 997 (2014) 821–826.
[43] G. Wang, X. Li, L. Gao, et al., A multi-objective whale swarm algorithm for energy-efficiency distributed permutation flow shop scheduling problem with sequence dependent setup times, IFAC-PapersOnLine 52 (13) (2019) 235–240.
[44] Chen. J.F.A., Wang. L.A., Peng. Z.P.B., A collaborative optimization algorithm for energy-efficiency multiobjective distributed no-idle flow-shop scheduling, Swarm Evol. Comput. 50 (4) (2019) 100557.
[45] H.B. Song, J. Lin, A genetic programming hyperheuristic for the distributed assembly permutation flow-shop scheduling problem with sequence dependent setup times, Swarm Evol. Comput. 60 (2021) 100807.
[46] Y.Z. Li, Q.K. Pan, K.Z. Gao, et al., A green scheduling algorithm for the distributed flowshop problem, Appl. Soft Comput. 109 (2021) 107526.
[47] X. Han, Y. Han, Q.-d. Chen, J.-q. Li, H.-y. Sang, Y.-p. Liu, Q.-k. Pan, Y. Nojima, Distributed flow shop scheduling with sequence-dependent setup times using an improved iterated greedy algorithm, Complex Syst. Model. Simul. 1 (3) (2021) 198–217.
[48] H. Qin, Y. Han, B. Zhang, L. Meng, Y. Liu, Q. Pan, D. Gong, An improved iterated greedy algorithm for the energy-efficient blocking hybrid flow shop scheduling problem, Swarm Evol. Comput. 69 (2021) 100992.
[49] K. Karabulut, D. Kizilay, M.F. Tasgetiren, L. Gao, Levent Kandiller, An evolution strategy approach for the distributed blocking flowshop scheduling problem, Comput. Ind. Eng. 163 (2022) 107832.