Expert Systems With Applications 200 (2022) 116910

Contents lists available at ScienceDirect

Expert Systems With Applications

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

Scheduling of energy-efficient distributed blocking flowshop using pareto-based estimation of distribution algorithm

Xiaohui Zhang [a], Xinhua Liu [b,c], Andrzej Cichon [d], Grzegorz Królczyk [e], Zhixiong Li [e,f,*]

[a] School of Electrical and Control Engineering, Xuzhou University of Technology, Xuzhou 221018, China
[b] School of Mechanical and Electrical Engineering, China University of Mining & Technology, Xuzhou 211006, China
[c] Technology and Innovation Research Center of JiangYan EDZ, Taizhou 225500, China
[d] Department of Electrical, Control and Computer Engineering, Opole University of Technology, Opole 45758, Poland
[e] Department of Manufacturing Engineering and Automation Products, Opole University of Technology, Opole 45758, Poland
[f] Yonsei Frontier Lab, Yonsei University, Seoul 03722, Republic of Korea

ARTICLE INFO

Keywords:
Estimation of distribution algorithm
Blocking constraint
Multi-objective optimization
Energy efficiency

ABSTRACT

This study investigates the impact of production scheduling decisions aims at improving productive and energy-efficient performances simultaneously in distributed blocking flowshops (EDBFSP). To reach a compromise between the conflicting objectives, a Pareto multi-objective optimization model based on the estimation of distribution algorithm (MOEDA) is proposed. Firstly, an initialization method based on the problem-specific characteristics is designed to create a promising population with quality and diversity; secondly, a probabilistic model based on a Bayesian network is constructed to predict position relationships between jobs. Two neighborhood operators with modified insertion technique are proposed to realize the adjustments of both job sequence and processing speed; thirdly, two operators are developed to execute multi-objective local searches on the elite solutions. Aiming at efficient utilization of the resulted blocking and idle time, an energy-saving method is designed for EDBFSP. In the experimental parts, to gain the best performance, the key parameters of MOEDA have been calibrated. The validation is conducted to assess the performances of the designed initialization method, neighborhood search, local search, and energy-saving strategies. The proposed MOEDA is also compared with mainstream metaheuristics for solving green scheduling problems. The experiment results show that the optimization and search ability of MOEDA have gained prominent advantages over other metaheuristics in both precision and distributivity.

1. Introduction

With the development of 'Kanban' and 'just-in-time' (JIT) concepts, manufacturing with zero or finite intermediate buffer has gained sustained attention. Such patterns are often mapped as the blocking flowshop manufacturing in academic researches, which has been extensively investigated over the past years. In a classical blocking flowshop, it is generally held that no buffers exist between two adjacent machines. The job that already finished one certain operation should be immediately transferred to the next operation/process unit for machining. If the next machine is tied up, this job has to be blocked on the previous machine and can be released until the next machine is available. The optimization of the scheduling problems in a blocking flowshop (BFSP) with makespan criterion is to eliminate or at least minimize the existed blocking intervals between operations. Heretofore, BFSP has been arisen in many industrial fields (Shao et al., 2018), such as the iron and steel industry, chemicals and pharmaceuticals, aeronautics parts fabrication, etc.

On the other hand, with the promotion of economic globalization and increased mergers between enterprises, the emergence of large-scale or concurrent production brings the pattern of distributed manufacturing essential. Distributed manufacturing decentralizes tasks into factories or workshops from different geographical locations. This pattern can help the manufacturers raise productivity, reduce cost, control risks, and adjust marketing policies more flexibly. Herein, we extend the aforementioned BFSP in a distributed environment as a distributed BFSP (DBFSP), which is demonstrated in Fig. 1. DBFSP considers f blocking factories, of which the machine configurations are the same. The jobs can be assigned to any factories and the processing of

* Corresponding author.
E-mail address: zhixiong.li@yonsie.ac.kr (Z. Li).

https://doi.org/10.1016/j.eswa.2022.116910
Received 29 July 2021; Received in revised form 23 January 2022; Accepted 12 March 2022
Available online 25 March 2022

each job follows the manufacturing procedure of a single blocking flowshop. To solve DBFSP, two decisions need to be made: the allocation of jobs to different factories, and the sequencing of jobs in the respective factory. Apparently, both decisions are highly associated with each other and could not be addressed without considering their serial characteristics.

Practical DBFSP often reveals that a single optimization goal is insufficient to figure a realistic scenario since the manufacturing system usually contains multiple conflicting objectives (Feng et al., 2020). The conflicting objectives concerned usually include the maximum completion time (makespan), total tardiness (TT), total flow time (TFT), energy consumption, carbon footprint, and so on. When optimizing multiple conflicting objectives, the improvement of one single objective may lead to the performance degradation of another or several other objectives. For instance, minimizing the makespan may require raising the processing speeds of jobs, but higher processing speed could form a stark contrast to the equilibrium of energy consumption or carbon emission, which may cause the over-run of energy or pollution costs. Hence, effective metaheuristics are needed to find desirable results in solution space and provide promising trade-offs for decision-makers.

Bearing the above observations, we have summarized the following motivations for this study:

Though different types of distributed flowshop scheduling problems have been studied, more practical objectives are still advised to consider. Recently, increasing environmental protection consciousness has resulted in the necessity of energy conservation measures in industrial fields. Making desirable decisions on schedules will be a green and effective way to improve energy efficiency. Therefore, the research on energy-conscious multi-objective DBFSP is of great significance to promote energy conservation for distributed manufacturing.

Due to the problem-specific characteristics of distributed flowshop with multiple optimization objectives, it is hard to work out schemes that can obtain global optimal values for each objective simultaneously. An effective and efficient algorithm is required to optimize the solutions deeply, as well as obtain the diversity to cover the partially quite distinctive features of other objectives.

Estimation of distribution algorithm (EDA) (Mühlenbein & Paass, 1996), as a kind of evolutionary optimization algorithm, is intended to avoid the damage of building blocks caused by crossover and mutation operation of genetic algorithm (GA). EDA describes the distribution of candidate solutions in their objective space through a probability model, which is constructed by the statistical method from a macro point of view. Then, EDA predicts better search space and generates new individuals by sampling the probabilistic model established from incumbent individuals. Compared to traditional GA, the main advantages of EDA are concluded as follows: (1) EDA uses joint probability distribution to express the linkage relationship between genes, which avoids the damage of building blocks caused by the crossover and mutation operation of GA; (2) EDA reduces the blindness and randomness of operator selection and parameter settings of GA, and also reduces the requirement of prior knowledge for the problem; (3) EDA employs the probability model to express the joint probability distribution among gene variables, and uses statistics to learn the parameters of the probability model, which has a higher theoretical basis and evolution orientation.

In this study, we propose a Pareto multi-objective estimation of distribution algorithm (MOEDA) to solve the energy-aware distributed blocking flowshop scheduling problem (EDBFSP). EDBFSP widely exists

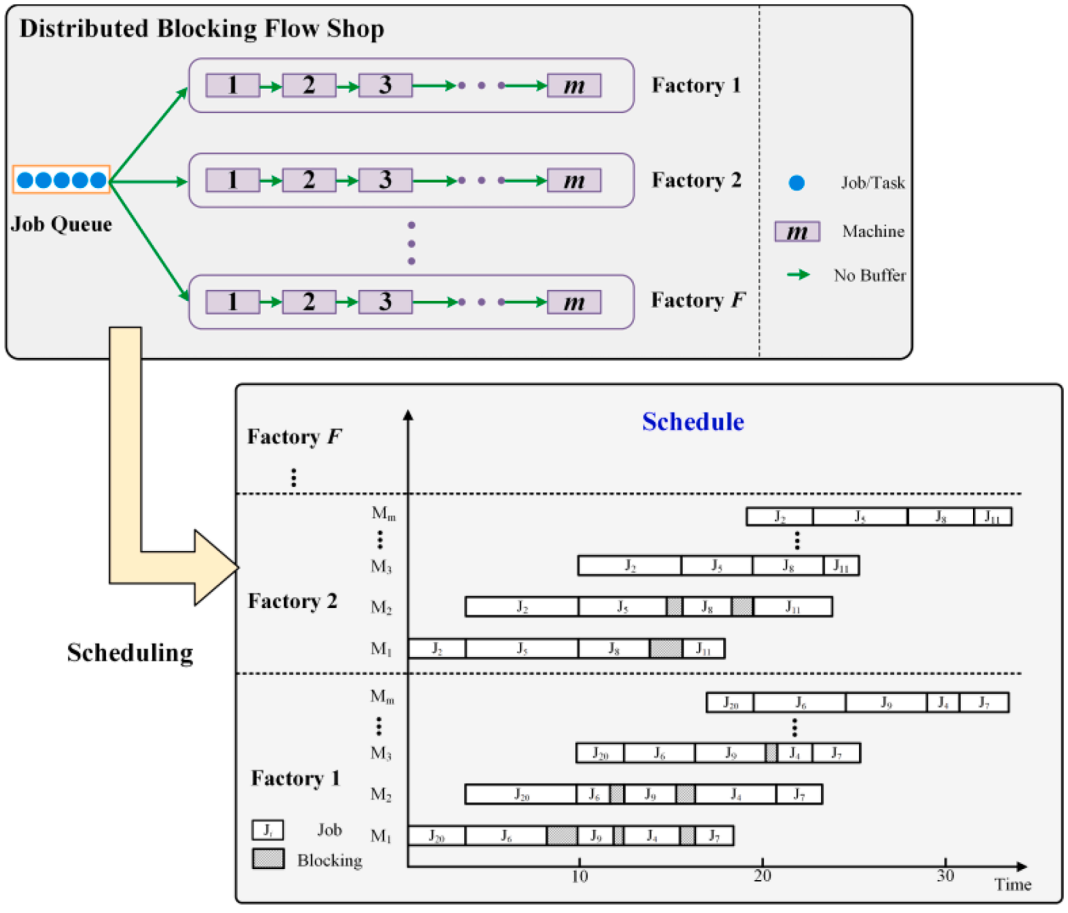

Fig. 1. An example of DBFSP with a Gantt chart.

in many industrial fields, especially the group such as pharmaceuticals and semiconductor enterprise, which usually have multiple factories all over the world. Limited to the process technology (i.e. cleanliness and temperature), the buffer zone is not provided between adjacent equipment. On the other hand, these groups should shoulder the social responsibility of energy conservation and emission reduction. Hence, EDBFSP has attracted more and more attention. The objectives of EDBFSP to be optimized are makespan and energy consumption. For each operation of all jobs, a speed scale vector was applied to adjust the processing speed, which is considered as the energy-related decision variable. The contributions to the MOEDA are introduced as follows: (1) A Bayesian network-based probabilistic model was constructed to predict the new individuals; (2) Two neighborhood operators were designed according to problem-specific characteristics of EDBFSP. A modified job insertion technique was proposed to realize the adaptive adjustment of speed level during the insertion procedure. (3) A Pareto multi-objective local search framework was designed to perturb the obtained elite solutions. At last, an energy-saving method based on non-critical operations was developed. We validated the effectiveness of MOEDA by conducting experimental simulations on benchmark instances along with a comparative study using the mainstream algorithms for energy-aware scheduling problems in distributed manufacturing systems.

The rest components of this paper are outlined as follows. Section 2 systematically reviews the relevant literature. Section 3 states the EDBFSP and gives the corresponding mathematical model. The detailed procedure of MOEDA is presented in Section 4. Section 5 demonstrates the numerical experiments and statistical analysis. Conclusions and future works are summarized in Section 6.

2. Related work

Recent publications related to this study are divided into three parts: distributed flowshop scheduling problem (DFSP), energy-aware flowshop scheduling problem, and the estimation of distribution algorithm. The relevant literature is summarized in the subsections.

2.1. Distributed flowshop scheduling problem (DFSP)

As aforementioned, DFSP is more sophisticated than a single flowshop scheduling problem since the assignment of jobs to factories should be considered in advance. This problem-specific characteristic also requires a reversal-change of optimization techniques such as neighborhood strategies and local search schemes. In 2010, DFSP with makespan criterion was studied by Naderi and Ruiz (2010) for the first time. The authors have analyzed different mathematical models of DFSP and proposed 12 heuristics combining with 2 job-to-factory assignment rules. Following this vein, various approaches were proposed to address DFSP. Recent optimization models are such as iterated greedy (IG) algorithm (Ruiz et al., 2019), benders decomposition algorithm (Hamzadayı, 2020), artificial bee colony (ABC) algorithm (Meng & Pan, 2021), and iterated local-search (ILS) algorithm (Jing et al., 2021).

As the research further developed, some variants have been studied by extending the permutation constraints of DFSP. For instance, DBFSP was raised by changing the permutation mode to “no buffer between adjacent operations” mode. To optimize the makespan criterion, Zhang et al. (2019) proposed a discrete fruit fly optimization algorithm (DFOA). Later on, Shao et al. (2020a) proposed a hybrid enhanced discrete fruit fly optimization algorithm (HEDFOA). Zhao et al. (2020) developed an ensemble discrete differential evolution (EDE) algorithm. Chen et al. (2020) designed an iterated greedy algorithm. Chen et al. (2021) also proposed an enhanced IG algorithm based on their previous work. Besides, Zhang et al. (2019) investigated DFSP with limited buffer constraint (DFSPLB) and proposed a dedicated DDE algorithm. Shao et al. (2020b) studied the distributed assembly blocking flowshop scheduling problem (DABFSP) and have proposed an ILS algorithm with a multi-type perturbation procedure containing four kinds of perturbed operators based on problem-specific knowledge.

The aforementioned literature, however, focused only on a single objective. Regarding the multi-objective DFSP, the literature is limited due to the complexity of the optimization models. Deng and Wang (2017) developed a competitive memetic algorithm (CMA) for a multi-objective DFSP with the same criteria. Wang and Wang (2018) proposed a knowledge-based cooperative algorithm (KCA) to address energy-efficient scheduling of DFSP with makespan and energy consumption as optimization objectives. Shao et al. (2019) proposed a hybrid EDA to optimize the DNWFSP considering total weighted tardiness (TWT) and makespan as performance criteria.

2.2. Energy-Conscious flowshop scheduling problem

Recently, the energy-conscious flowshop scheduling problem has gained extensive attention. A variety of machine-based policies and strategies have been investigated to improve energy efficiency. Mouzon et al. (2007) observed that the non-bottleneck machines would waste a huge amount of energy in idle time. They constructed a manufacturing framework with turn-on and turn-off strategies for non-bottleneck machines to minimize the energy cost and other objectives. Limited by the actual situation on-site, the turn-on and turn-off strategy may not applicable since machine status cannot be frequently switched during the manufacturing process. Wang et al. (2018) found that converting blocking time to idle time could save energy while keeping makespan unvaried. They integrated this strategy into a parallel variable neighborhood search algorithm (PNVS) to optimize the energy consumption and makespan for a BFSP. Similar to Wang et al. (2018), Jiang and Zhang (Jiang & Zhang, 2019) have investigated the property of non-processing energy in a hybrid flow-shop with buffer constraint and developed a multi-objective evolutionary algorithm based on decomposition (MOEA/D). Besides, Fazli and Wang (Fazli Khalaf & Wang, 2018) proposed a framework that decreased the energy demand through on-site renewables and energy storage strategy, which was proven to be effective to optimize the electricity cost for a two-stage stochastic flowshop. Recently, some researchers considered that machines can consume different energy due to different operation patterns. Thereby, a speed control mechanism for machine operation was proposed (Fang et al., 2011, 2013). When the mechanism requires a machine to processes at a higher speed, the energy consumption grows whereas the processing time reduces. Obviously, the setting of speed level will cause a conflict between the processing time and energy consumption. Based on this framework, some scheduling models have been further proposed and investigated (Lei et al., 2018). For an energy-aware flowshop scheduling problem, Ding et al. (2016) developed an extended NEH-Insertion procedure, which was able to realize the synchronous adjustment of processing speed when executing the job insertion procedure for finding a better position in solution space. Jiang and Wang (Jiang & Wang, 2019) proposed an improved multi-objective evolutionary algorithm (IMOEA) to optimize the flowshop with makespan and energy consumption criteria. Öztop et al. (2020) proposed a multi-objective variable block insertion heuristic to study the trade-off between TFT and energy consumption of a permutation flowshop. Yüksel et al. (2020) proposed a multi-objective discrete artificial bee colony algorithm to minimize total tardiness and energy consumption for a no-wait flowshop scheduling problem (NWPFSP). Besides, Schulz et al. (2019) proposed three energy-saving strategies, which not only consider the varied process speed of the machine but also the varied energy unit price and the peak-valley load as energy-related decision variables. The proposed strategies were integrated to optimize the multi-objective for a hybrid flowshop.

Concerning the energy-aware scheduling problems in a distributed manufacturing environment, the literature is rather limited. Deng and Wang et al. (2016) investigated the DFSP with multi-objective developed a competitive memetic algorithm (CMA) to minimize the

makespan and carbon emission, which is directly derived from energy consumption. Jiang, Wang, and Lu (2017) employed MOEA/D with a critical-path-based energy-saving method. For DFSP with no-wait constraint, Wang et al. (2017) designed a cooperative algorithm (CA) to improve energy consumption and makespan. Fu et al. (2019) have studied DFSP with the stochastic processing time constraint and designed a brainstorm optimization algorithm (MOBO) to evaluate three objectives: total tardiness, makespan, and energy consumption. Zheng et al. (2020) combined EDA with IG to address the multi-objective fuzzy distributed hybrid flow shop scheduling problem, which considers the fuzzy total tardiness and robustness as the objectives.

2.3. Engineering application of EDA

EDA (Mühlenbein & Paass, 1996) is a new kind of intelligent algorithm based on statistical theory. Unlike other typical evolutionary algorithms that evolve through movement-based operation (e.g., crossover and mutation), EDA evolves through the probabilistic model. Since the probability distribution of good solutions can guide the search direction, the randomness of evolution could be effectively avoided. It implies that EDA can learn good information from the incumbent elite individuals. The flow diagram of basic EDA is presented in Fig. 2. Generally, the flow path of basic EDA is described as follows:

Set the parameter, initialize the population, and select elite individuals;
Build a probabilistic model through learning the statistical information of the elite individuals;
Sample the updated probabilistic model, and predict the new individuals.

EDA has been verified with good global search and optimization ability and applied to solve engineering problems such as traveling salesman problem (Song et al., 2017), communication detection problem (Zhou et al., 2018), contract distribution (Hu et al., 2018), etc. For scheduling problems, Tian et al. (Tian et al., 2019) designed a hybrid EDA in dealing with a scheduling problem with resource constraints and time uncertainty. Pérez-Rodríguez and Hernández-Aguirre (2018) embedded mallows distribution into EDA to construct better sequences for the flexible jobshop scheduling problem (FJSP). Zhou et al. (2018) proposed a reentrant hybrid flowshop model where inspection and repair operations are considered. A hybrid differential evolution algorithm with EDA using an ensemble model was proposed to solve the problem. For multi-objective scheduling problems, Wu and Wang (2018) adopted multi-model EDA to determine the task process permutation and voltage supply levels under a cloud computing system. Amiri and Behnamian (2020) applied EDA to solve the flowshop scheduling problem with makespan and energy consumption under uncertainty.

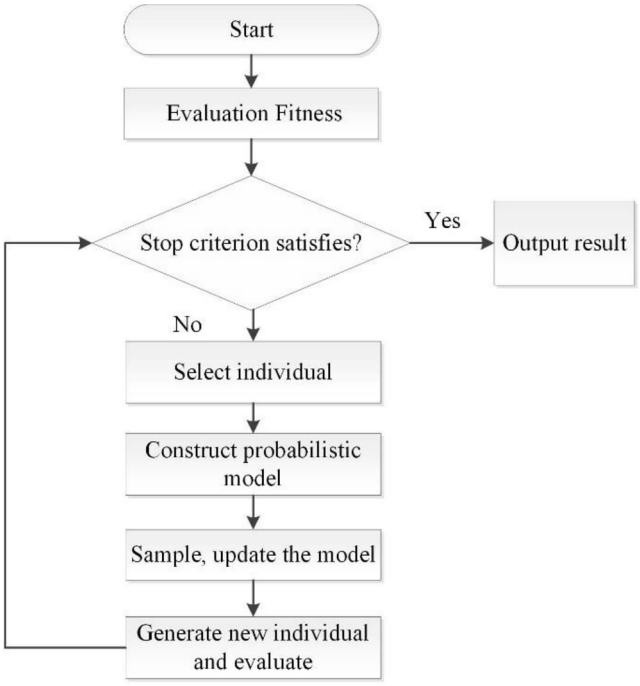

Fig. 2. Flow diagram of basic EDA.

2.4. Discussion

From the literature review, it can be concluded that some methodologies were proposed to address the distributed scheduling problems. Nevertheless, the vast majority of them still focused on single-objective optimization. Practical productions, in fact, often need to consider complex environments with multiple conflicting objectives. On the other side, with growing concerns over cost-down and environmental requirements, the energy-saving concept is receiving extensive attention. Therefore, it is strongly necessary to study energy-aware DBFSP from both theoretical and applicable perspectives.

3. Problem statement

For a better understanding, the general concept of multi-objective optimization is firstly introduced in this section; then, the mathematical model and properties of EDBFSP are described in the subsections.

3.1. Multi-objective optimization problem (MOP)

MOP depicts a procedure that can improve the performance of more than one objective, which aims at offering a set of criteria that enable decision-makers to improve a given activity compared to previous levels continuously. For solving MOP, the decomposition-based approach and the Pareto-dominance-based approach are considered as two popular approaches for approximating optimal solution sets (Wang et al., 2018). The former approach decomposes MOP into a set of sub-problems through linear or non-linear aggregation functions and assesses all sub-problems synchronously while the latter one addresses MOP with more than two objectives according to dominance relation between solutions. Since the Pareto-based approach is with the property of easy implementation and mostly solves problems with no more than three objectives, we adopt it to solve EDBFSP. In general, a MOP for minimization issue can be expressed in the following form (Shao et al., 2019):

$$\mathbf{min}\ f(\mathrm{X}) = (f_1(\mathrm{X}), f_2(\mathrm{X}), \ldots, f_r(\mathrm{X})) \tag{1}$$

$$\mathrm{X} = (x_1, x_2, \ldots, x_n) \in R^n \tag{2}$$

$$\mathbf{s.t.} \begin{cases} g_i(\mathrm{X}) \geqslant 0 & i = 1, \ldots, p \\ h_j(\mathrm{X}) \geqslant 0 & j = 1, \ldots, s \end{cases} \tag{3}$$

where $f_r(\mathrm{X})$ represents the r−th. sub-objective, X indicates the solution set in the decision variable space R^n. p and s denote the number of equality and inequality constraints, respectively. To apply the Pareto-based approach, the following concepts need to be clarified:

Pareto Dominance: There exist two feasible solutions $\times$ and $\mathbf{y}$, $\mathbf{x}$ dominates $\mathbf{y}$ (denote as $\mathbf{x} \prec \mathbf{y}$) if and only if the following conditions are satisfied:

$$\begin{cases} \forall a \in \{1, 2, \ldots, r\}, f_{\_a}(\mathbf{x}) \leqslant f_{\_a}(\mathbf{y}) \\ \exists b \in \{1, 2, \ldots, r\}, f_{\_b}(\mathbf{x}) \leqslant f_{\_b}(\mathbf{y}) \end{cases} \tag{4}$$

Pareto optimal/non-dominated solution: If the solution × is not dominated by all other solutions of a certain algorithm, it is defined as a non-dominated solution.

Pareto optimal solution set: The set constructed by all Pareto optimal/non-dominated solutions obtained from a certain algorithm.

Pareto front (PF): The PF is formed by the non-dominated solutions in Pareto optimal solution set. When solving practical problems, decision-makers can consider the equilibrium of multiple objectives that need to be optimized and select the final decision solution on the Pareto frontier. Thus, it can be concluded that the essence of Pareto multi-objective optimization is to obtain an optimal solution set whose distribution and convergence are as close as possible to the real **PF**.

3.2. Problem statement of EDBFSP

EDBFSP can be expressed in the following form. Assume a set of jobs $J = \{J_1, J_2, ..., J_n\}$ and a set of identical factories$F = \{F_1, F_2, ..., F_f\}$, each of which furnishes with a set of machines.$M = \{M_1, M_2, ..., M_m\}$.. The jobs can be assigned to any of the f factories for processing. Each machine can process at a set of speed scale$V = \{V_1, V_2, ..., V_{sm}\}$. In general, EDBFSP consists of three sub-problems: the assignment of jobs to factories, the sequencing of jobs in each factory, and the selection of speed scale for each operation. Assume $n_k (n_k \leqslant n)$ jobs are assigned to the factoryF_k, and the job sequence is denoted asπ_k, where $\pi_k(l)$ represents the $l-$th job inπ_k. Then, $O_{\pi_k(l),i}$ denotes the operation of job $\pi_k(l)$ on machine i. The processing speed of a certain operation $O_{\pi_k(l),i}$ is represented as$v_{\pi_k(l),i} \in V$. Also, the operation $O_{\pi_k(l),i}$ has a standard processing time$P_{\pi_k(l),i}$. If $O_{\pi_k(l),i}$ is processed at speed$v_{\pi_k(l),i}$, its actual processing time $p_{\pi_k(l),i}$ can be calculated as$p_{\pi_k(l),i} = P_{\pi_k(l),i}/v_{\pi_k(l),i}$. Assume $S_{\pi_k(l),0}$ denotes the start time of job $\pi_k(l)$ on the first machine in factoryF_k, and $d_{\pi_k(l),i}$ is defined as the departure time of operation$O_{\pi_k(l),i}$. Regarding the energy-related concept, a machine has 2 operation modes: standby and process. Under standby mode, the energy consumption per unit time is defined asSE, which is defined as a constant; while under process mode, the energy consumption per unit time is$PE_{\pi_k(l),i,v}$. If a job is blocked on a machine, this machine is switched to standby mode. Let EC be the total energy consumption, the mathematical model of EDBFSP can be expressed as follows.

Objectives:$min\{C_{max}, EC\}$

s.t.

$$X_{\pi_k(l),i} \in \{0,1\},\ l = 1,2,...n_k, i = 1,2,...,m, k = 1,2,...,f \tag{5}$$

$$Y_{\pi_k(l),i,v} \in \{0,1\},\ l = 1,2,...n_k, i = 1,2,...,m, k = 1,2,...,f \tag{6}$$

$$p_{\pi_k(l),i} = P_{\pi_k(l),i} \cdot \sum_{v=1}^{sm} \frac{Y_{\pi_k(l),i}}{v_{\pi_k(l),i}}, l=1,2,...n_k, i=1,2,...,m, k=1,2,...,f, v=1,2...,sm \tag{7}$$

$$D_{\pi_k(0),i} \geqslant 0,\ i = 1,2,...m,\ k = 1,2,...f \tag{8}$$

$$D_{\pi_k(l),i} \geqslant D_{\pi_k(l-1),i+1},\ l = 2,...n_k, i = 1,2,3,...,m-1, k = 1,2,...,f \tag{9}$$

$$D_{\pi_k(l),i} \geqslant D_{\pi_k(l),i-1} + \sum_{j=1}^{n} X_{\pi_k(l),i} \times p_{\pi_k(l),i}, l=2,...n_k, i=1,2,3,...,m-1, k=1,2,...,f \tag{10}$$

$$D_{\pi_k(l),0} \geqslant D_{\pi_k(l-1),1},\ l = 1,2,...n_k, k = 1,2,...f \tag{11}$$

$$C_{max} \geqslant D_{\pi_k(l),m},\ l = 2,...n_k, k = 1,2,...,f \tag{12}$$

$$\sum_{k=1}^{f} \sum_{l=1}^{n} X_{\pi_k(l),j} = 1,\ j = 1,2,...,n \tag{13}$$

$$\sum_{j=1}^{n} X_{\pi_k(l),j} \leqslant 1,\ j = 1,2,...,n, k = 1,2,..f \tag{14}$$

$$Z_{k,i}(t) \in \{0,1\},\ i = 1,2,...,m, k = 1,2,...,f \tag{15}$$

In the above equations, Eq. (5) sets up a binary decision variable$X_{\pi_k(l),i}$. When a job appears on the $l-th$ position ofπ_k,$X_{\pi_k(l),i} = 1$; otherwise,$X_{\pi_k(l),i} = 0$. Eq. (6) denotes a binary decision variable. When a job is processed with a speed$v_{\pi_k(l),i} \in V$,$Y_{\pi_k(l),i,v} = 1$; otherwise,$Y_{\pi_k(l),i,v} = 0$. Eq. (7) calculates the actual processing time of a job. Eq.8 defines the start time constraint of processing. Eq. (9) represents the time relationship of leaving machines between two adjacent jobs in π_k in factoryF_k. Eq. (10) defines the departure time relationship between two adjacent processes of job$\pi_k(l)$. Eq. (11) gives the relationship between the departure time of each job on the first machine and its precedence job on next machine. Eq. (12) denotes the process finish time of the factory. Eq. (13) ensures that one job can only appear in one position ofπ_k. Eq. (14) guarantees that each position in π_k can only be assigned with one job. Eq. (15) denotes a binary decision variable$Z_{k,i}(t)$. When a machine in factory F_k is processed at time t,$Z_{k,i}(t) = 1$; otherwise,$Z_{k,i}(t) = 0$.

The calculation of makespan for DBFSP refers to (Shao et al., 2020a; Zhang et al., 2019). As a result, the global makespan for EDBFSP is determined through the comparison between the makespans of all factories:

$$C_{max}(\Pi) = \max_{k=1}^{f} (C(\pi_k)) \tag{16}$$

The total energy consumption of a schedule can be derived as follows:

$$EC = \sum_{k=1}^{f} \int_{0}^{C(\pi_k)} (\sum_{v=1}^{sm} \sum_{i=1}^{m} PE_{\pi_k(l),i,v} \cdot Z_{k,i}(t) + \sum_{i=1}^{m} SE \cdot (1 - Z_{k,i}(t)))dt \tag{17}$$

For an EDBFSP with 6 jobs, 2 factories, and 3 machines, a diagrammatical real-time energy consumption curve is sketched in Fig. 3 as an example. In Fig. 3, C_1 and C_2 represent the makespan of each factory, respectively. The area occupied by the energy consumption curve signifies the energy consumption of each factory.

3.3. Property of EDBFSP

In this section, some problem-specific properties are explored and introduced, which help to obtain more non-dominated solutions during the search procedure.

Property. (*(Shao et al., 2019)*) Assumes that two schedules Π_1 and Π_2 have the same makespan, i.e.,$C_{max}(\Pi_1) = C_{max}(\Pi_2)$. The relationship fulfills: (1)$\forall i \in \{1,2,...,m\}, j \in \{1,2,...,n\}, v_{j,i}(\Pi_1) \leqslant v_{j,i}(\Pi_2)$; (2)$\exists i \in \{1,2,...,m\}, j \in \{1,2,...,n\}, v_{j,i}(\Pi_1) < v_{j,i}(\Pi_2)$. Then, the dominance relationship of energy consumption can be described as$E(\Pi_1) < E(\Pi_2)$.

4. MOEDA for EDBFSP

Although EDA has presented strong global search ability and shown excellent performances in various industrial applications, difficulties yet expose when employing it to address the complex and multi-dimensional scheduling problems. The reasons could be summarized as follows. Firstly, the random initialization strategy of basic EDA reduces the quality of solutions, which raises difficulties in seeking non-dominated solutions. Secondly, a simple probabilistic model could not promote the convergence of the population, especially when dealing with multi-objective problems. Thirdly, it lacks an effective local intensification strategy to implement sophisticated searches nearby the solutions with good quality.

To break through the above shortcomings, we designed MOEDA that

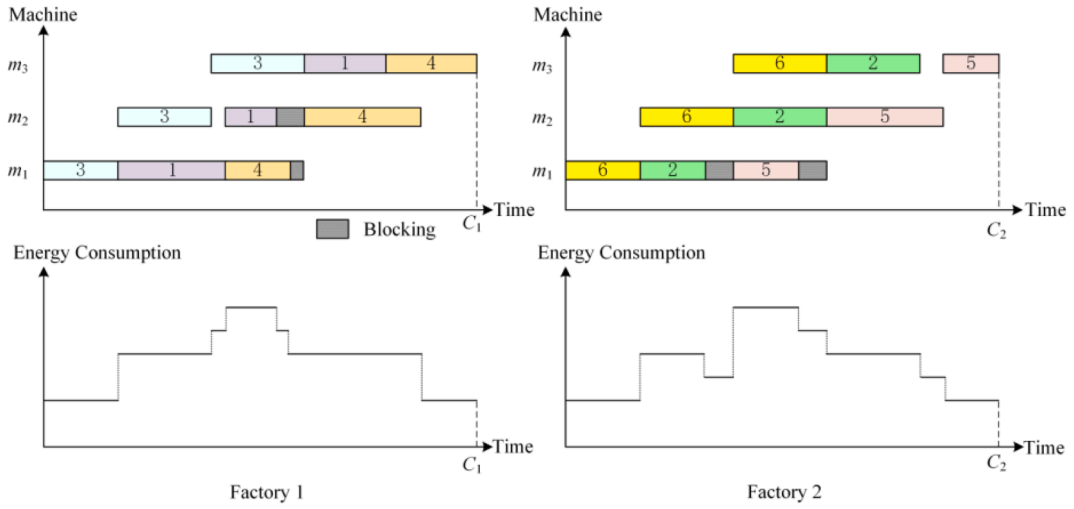

Fig. 3. Demonstration of real-time energy consumption in a distributed blocking flowshop.

inherits and extends the searching idea of basic EDA to optimize EDBFSP. The flowchart of the proposed MOEDA is illustrated in Fig. 4. MOEDA contains 4 key components: initialization, neighborhood search, local search, and energy saving. In the neighborhood search phase, a Bayesian network-based probabilistic model was developed to inherit information from the incumbent non-dominant solutions, with the view of exploring better solution spaces. An effective insertion method was proposed to adjust the speed level of each operation during the job sequencing procedure. To enhance the exploitation ability, a Pareto multi-objective local search framework was designed. Finally, a non-critical operation-based energy-saving method was presented. Since MOEDA equilibrates the exploration and exploitation, satisfactory performances were expected to reach when adopting it in addressing EDBFSP.

4.1. Encoding and decoding scheme

A complete solution of EDBFSP contains two parts: the job sequence and the speed level matrix for each operation of the jobs. The solution Π is encoded as a sequence-speed list:$\Pi = (\boldsymbol{\pi}, \boldsymbol{V})$. In$\Pi$, $\boldsymbol{\pi} = \{\pi_1, \pi_2, ..., \pi_f\}$ denotes the job sequence which is composed of all partial sequences π_k for each factory. This presentation can be easily decoded since π_k decides not only the jobs assigned to factoryF_k, but also the operation sequence of the jobs in this factory. On the other side, $\boldsymbol{V}$ is a $n \times m$ matrix that gives the speed levels for each operation of all jobs. For a better understanding, an example of the encoding and decoding scheme is given in Example 1.

Example 1. A feasible solution Π of EDBFSP with job number$n = 6$, machine number$m = 3$, and factory number $f = 2$ is assumed. The job sequence part is$\boldsymbol{\pi} = \{\pi_1, \pi_2\}$, in which $\pi_1 = \{J_1, J_2, J_5\}$ and$\pi_2 = \{J_4, J_3, J_6\}$. When decoding$\Pi$, job$J_1$,$J_2$, and J_5 are allocated to factory F_1 and are processed in the order of$J_1 \rightarrow J_2 \rightarrow J_5$. Similarly, the decoding result of π_2 for another factory F_2 is$J_4 \rightarrow J_3 \rightarrow J_6$. The speed matrix for each operation is shown as Eq. (18), where the row number and column number represent the machine number of each factory and the total number of jobs, respectively.

$$\boldsymbol{V} = \begin{bmatrix} 1 & 2 & 1 & 1 & 3 & 3 \\ 2 & 1 & 3 & 1 & 1 & 1 \\ 2 & 3 & 2 & 3 & 2 & 3 \end{bmatrix}^{\mathrm{T}} \tag{18}$$

4.2. Initialization of population

Good seeds can lead to a rapid convergence of the algorithm when a candidate makes one of the objectives minimal in an initial population for solving MOP (Gong et al., 2018). In this study, MOEDA aims to start from solutions with good makespan, and then stepwise guide the search to the energy-related objective. The graphical evolution procedure of the initial solution is demonstrated in Fig. 5.

Since $\boldsymbol{\pi}$ and $\boldsymbol{V}$ in a feasible solution $\Pi = (\boldsymbol{\pi}, \boldsymbol{V})$ have different dimensions, they can only be initialized separately. In this study, $\boldsymbol{V}$ is randomly initialized, with the view of keeping the diversity and distributivity. To obtain $\boldsymbol{\pi}$ with good makespan, we have modified the ECF (earliest completion factory) initialization rule (Naderi & Ruiz, 2010). ECF firstly extended the insertion idea of the NEH heuristic (Nawaz et al., 1983) into the distributed flowshop scheduling. The ECF heuristic can be presented in following three steps:

Step 1: Calculating the total processing time $T_j = \sum_{i=1}^{m} P_{j,i}$ of each job, arrange all jobs in descending order according toT_j.
Step 2: Orderly assign each job to the factory which can finish processing of this job at the earliest time, namely, the minimum C_{max} that considers this job as the last one.
Step 3: Repeat Step 1 and Step 2, until all jobs are assigned.

ECF was originally developed for traditional DFSP with permutation constraints. It considers that a job with a greater T_j ought to be allocated preferentially than those with smallerT_j. Nevertheless, for blocking constraint, allocating a job with higher T_j in the forepart of a schedule could result in a greater block for the subsequent jobs. Consequently, the greater block generates a larger makespan. Thus, the original ECF is not suitable for DBFSP. Pan et al. (2011) investigated such characteristics and proposed NEH-P concept that arranges jobs with shorter T_j preferentially. NEH-P concept has been verified to be more efficient and suitable than the NEH method for blocking mode through extensive experiments (Miyata & Nagano, 2019). In this study, we employ the concept of NEH-P and modify the first step of ECF to arrange all jobs in ascending order according toT_j. The rest steps of ECF remain. We define the initialization heuristic is defined as ECFPV (earliest completion factory NEH-P with velocity variable), which is sketched in **Pseudocode**

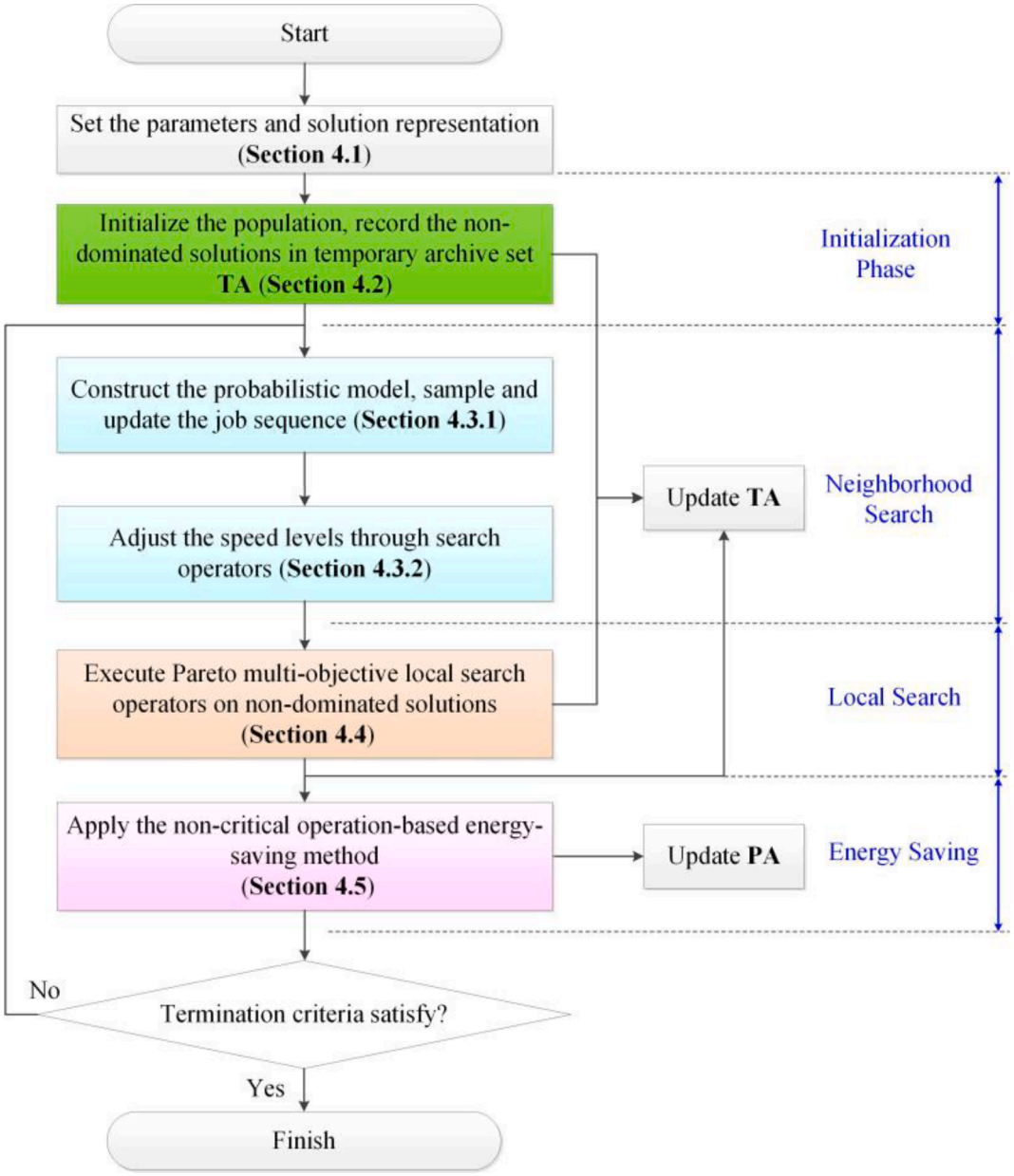

Fig. 4. Flowchart of the proposed MOEDA.

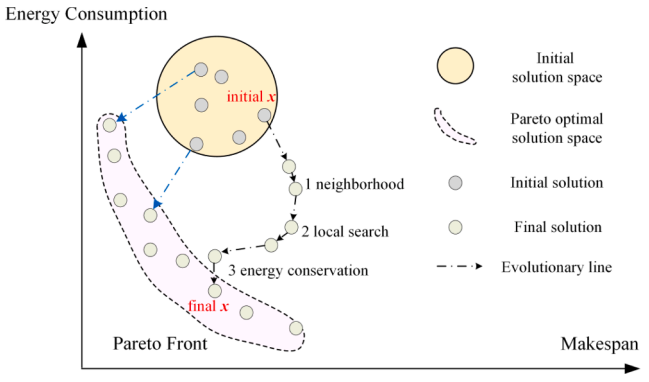

Fig. 5. Evolutionary stages of initial solutions of MOEDA.

1.

Pseudocode 1 ECFPV Initialization

Input: population size *Ps*, factory number *f*, job number *n*, temporary archive **TA**
Output: population ***POP***, **TA**
01: **For** $i = 1$ to *Ps*
02: Randomly generate the speed level matrix $\boldsymbol{V}_i$
03: Calculate the total processing time $\boldsymbol{T}_j$ according to $\boldsymbol{V}_i$ and standard processing time
04: Sort the jobs in ascending order according to the value of $\boldsymbol{T}_j$
05: Generate $\boldsymbol{\pi}_i$ using Step 2 of ECF heuristic
06: let $\Pi_i = (\boldsymbol{\pi}_i, \boldsymbol{V}_i)$
07: **End For**
08: Evaluate the population ***POP***, obtain the non-dominated solutions
09: Store the found non-dominated solutions in **TA**

After initialization, the Pareto dominance relationship introduced in Section 3.1 is applied to select the non-dominated individuals. It is worth mentioning that a temporary archive **TA** is set to store the obtained non-dominated solutions in each generation. A Pareto archive **PA**

is established to record the non-dominated solutions found along with the whole evolution. The solutions in **TA** need to be compared to those in **PA** from each generation, and **PA** is stepwise updated to approximate the Pareto front (**PF**).

4.3. Neighborhood search

After initialization, a neighborhood search framework is designed to help MOEDA search in a promising solution space. On one hand, a probabilistic model based on the Bayesian network is employed to predict the job sequence$\boldsymbol{\pi}$. On the other hand, the speed matrix $\boldsymbol{V}$ is updated based on specifically designed speed adjustment operators.

4.3.1. Bayesian network-based probabilistic model

Since EDBFSP is a discrete multi-variable problem, it is necessary to figure out the dependence between different variables. Bayesian network (Qian et al., 2017) is a typical probabilistic model that represents the relationship between multiple random variables, it is a more ideal choice to employ the Bayesian network to construct the probabilistic model for MOEDA. The prototype diagram of the Bayesian network is shown in Fig. 6. In a Bayesian network, each node represents a probabilistic variable, and the directed arc between two nodes represents their dependent relationship. In Fig. 6, if the parent nodes of the child node X_3 are X_1 and X_2, the conditional occurrence probability of variable X_3 can be expressed as:

$$P(X_3 = x_3|X_1 = x_1; X_2 = x_2; \ldots; X_7 = x_7) = P(X_3 = x_3|X_1 = x_1, X_2 = x_2) \tag{19}$$

In general, there are two methods to count the nodes in Bayesian networks (Yanai & Iba, 2013). One is to average the occurrence number of the same node, and the other is to calculate the weight value for the product of the occurrence number and its fitness value. In this study, we employed the first method: assume the sampling size is Ns, the probability variable is defined as$P(X_i)$. The set of nodes on which X_i is associated is denoted asC_i. The probability of the corresponding variable in C_i of solution s can be obtained as follows:

$$P(X_i = x|C_i = c) = \frac{\sum_{s=1}^{Ns} \delta(s, X_i = x|C_i = c)}{Ns} \tag{20}$$

where δ is a binary decision variable:

$$\delta(s, X_i = x|C_i = c) = \begin{cases} 1 & \text{if there exists } X_i = x \text{ and } C_i = c \\ 0 & \text{else} \end{cases} \tag{21}$$

Bayesian network only selects high-quality solutions from the current population to build a probabilistic model. The purpose is to complete the inheritance and learning of high-quality information. The probability model needs to be updated and sampled to produce new individuals. The update mechanisms used in this study is the well-known machine learning concept-Heb rule (Shao et al., 2019):

$$P^{'} = (1 - \alpha_u)P(X_i = x|C_i = c) + \alpha_u P_{default}(X_i = x|C_i = c) \tag{22}$$

In Eq. (22), $\alpha_u \in (0, 1)$ represents the update rate or learning rate of the probabilistic model. The update mechanism based on Heb rule can overlook the quantitative changes in the probability distribution and keep the probabilities of nodes positive, to avoid the appearance of illegal solutions. In Eq. (22), the occurrence probability of nodes $P_{default}$ can be initialized normally:

$$P_{default} = \frac{1}{\text{number of nodes}} \tag{23}$$

In job sequence π for EDBFSP, $P_{default}(X_i = x|C_i = c)$ denotes the probability that a job has appeared at one node (position). From Eq. (20) to (23), it can be concluded that the update procedure of the probabilistic model is essentially a process of reinforcement learning. After iterative updating, Bayesian network will fully learn the information of high-quality solutions. The distribution status of solutions will be described more accurately, which is helpful for the convergence of the algorithm.

For a better understanding, Fig. 7 presents a Bayesian network with 3 jobs and 10 non-dominant solutions. Each node $N_{l,j}$ represents that job j appears at the l-th position of the sequenceπ. The nodes in each row of the network represent the jobs that appear in one position ofπ, while the nodes in each column represent all positions of π that one certain job appears. The occurrence numbers of job combinations in all non-dominant solutions are expressed by the number of probabilities between two nodes connected by directed arcs. Therefore, the sum of all probability degrees is the number of non-dominant solutions. The nodes surrounded by dashed lines represent nonexistent nodes. In Fig. 7, $N_{2,3}$ denotes the situation that job 3 never appears at the second position in π of the 10 selected non-dominant solutions.

In the view of EDBFSP, we use a two-dimensional matrix $M\_js(g)$ to represent the probabilistic model between genes and the evolutionary direction of the population:

$$M\_js(g) = \begin{pmatrix} l_{1,1}(g) & \ldots & l_{n,1}(g) \\ \vdots & \ddots & \vdots \\ l_{1,n}(g) & \ldots & l_{n,n}(g) \end{pmatrix} \tag{24}$$

In Eq. (24), $l_{l,j}(g)$ denotes the probability that job $j(j = 1, 2, 3..., n)$ appears at the l-th position ofπ, which can be calculated with Eq. (20). When initializing$M_js(g)$, to ensure the uniform sampling space in the initial stage of the algorithm, MOEDA sets the initial distribution of the probabilistic model to uniform distribution according to Eq. (23), i.e.$l_{i,j}(g) = 1/n$, where n is the total number of jobs.

After initializing the probabilistic model, MOEDA employs the Heb's rule with the learning rate α_u to select high-quality individuals and then updates the probabilistic model. Then, the tournament selection strategy (Miller & Goldberg, 1995) is adopted to sample the new model, and further generate new individuals. The deatiled procedure is as follows. Firstly, Bayesian network updates all normalized probability values for the $l-th$ position in the probabilistic model (normally start from the first position). Then, adopt the tournament selection strategy to select the job with good probability values for the $l-th$ position. Later on, the same sampling procedure is executed to selected the job at the $l+1-th$ position based on the job at $l-th$ position. The above steps repeat until the job at the last position of π is selected. Then, Bayesian network completes the generation of one new individual.

Since π and V are separately encoded due to different dimensions, it is impossible to build a Bayesian network-based probabilistic model to predict π and V at the same time. In this study, we construct a Bayesian network to updateπ. After the new sequence π is generated, the corre-

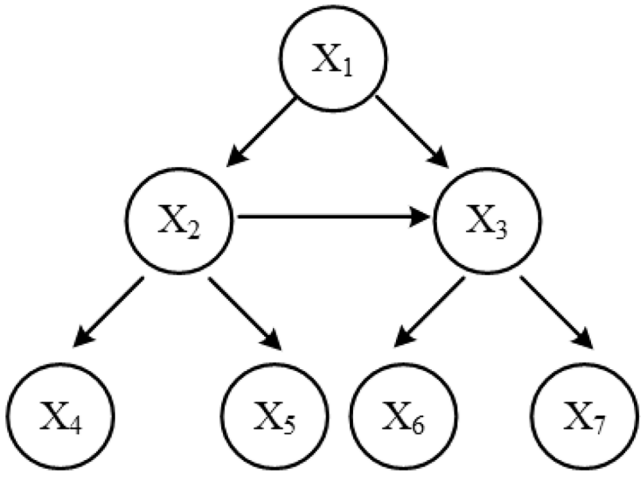

Fig. 6. Example of Bayesian network.

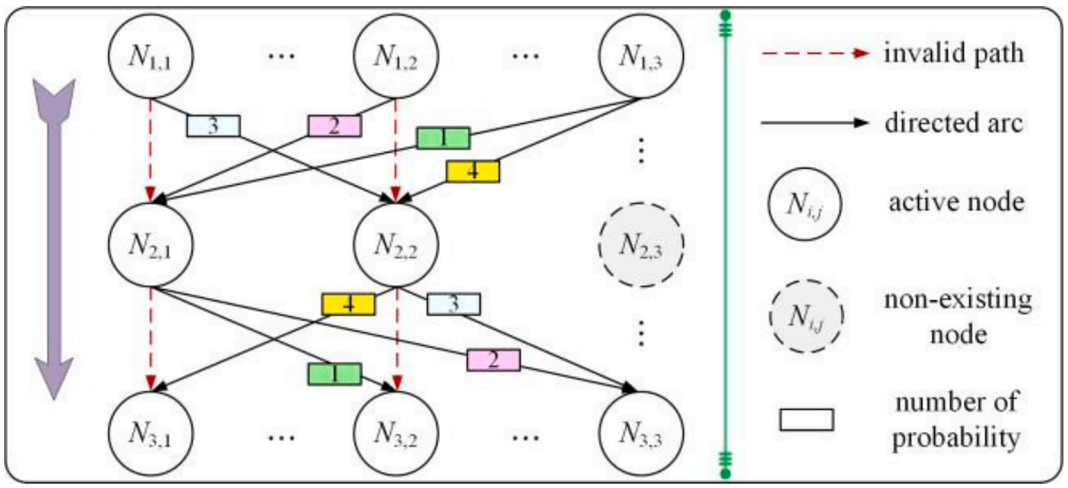

Fig. 7. An exemplary diagram of the Bayesian network for EDBFSP.

sponding speed level matrix V still initialized randomly. Moreover, to ensure the diversity and distribution of the population, 70% of the offspring are created by sampling the probabilistic model, and the remaining are generated using the proposed initialization strategy proposed in Section 4.2.

4.3.2. Update mechanism for process speed

After predicting the new job sequenceπ, the corresponding speed level matrix V is to be adjusted. Based on the properties analyzed in Section 3.3, two speed adjustment operators are designed for MOEDA, to further improve the individual quality of the population.

ED operator: According to **Property**, randomly select one operation, and reduce its speed level to the allowable minimum. The purpose of designing this operator is to obtain the longest processing time so that the energy consumption of this operation can be reduced without affecting the original makespan.

EC operator: Randomly select one job from the factory with maximum energy consumption, insert it into the factory with minimum energy consumption. The aim of designing **EC** operator is to adjust both operation speed level and job sequence. Since the EDBFSP model takes discrete speed levels, it is of great significance to determine the minimum speed level that an operation could allow when executing the insertion procedure. Ding et al. (2016) have proposed a job insertion technique (denote as **ES**), which realized the synchronous optimization of speed and job sequence for permutation flowshop scheduling problems. In this study, we extended **ES** for EDBFSP. Assume job q will be inserted in position h of the job sequence, its speed vector (speed levels for all operations) after inserting is defined asV_q^h. The overall insertion procedure is described in **Pseudocode 2** and sketched in Fig. 8. In **Pseudocode 2**, Line 01 calculates the corresponding variables after insertion, where $D_{h,i}$ defines the earliest departure time of job q on machine i, $Q_{h,i}$ is the tail that is the duration between the end of all operations and the latest starting time of job q, and $F_{h,i}$ denotes the earliest completion time on machine i after insertion. The detailed deduction procedure refers to speed-up method used for DBFSP (Ying & Lin, 2017; Zhang et al., 2019). Line 02 calculates the makespan with maximum allowable speed levelV_q^{sm}. Δ_m in Line 03 denotes the maximum processing time increment of job q on the last machine. Line 04 determines the minimum speed level u in relation toΔ_m. Since EDBFSP takes discrete speed levels, the real processing time increment on machine m might not be as exact asΔ_m. Hence, the start time of job q on machine m can be postponed with maximal Gap_m time units as derived in Line 07 through setting$i = m-1$. Then, Δ_{m-1} for job q on machine $m-1$ is determined. In EDBFSP, Δ_{m-1} is restricted by two constraints. The first one is that the processing time increment should not result in any postponement (blocking) of job q on its succeeding machine m, which is presented in Line 08. The second one is that the processing time increment ought not to affect the start time of its sequential operations on the same machine$m-1$, as stated in Line 09. Then, the insertion procedure for job q on the current machine finishes. The insertion procedure continues until the speed levels of job q on all

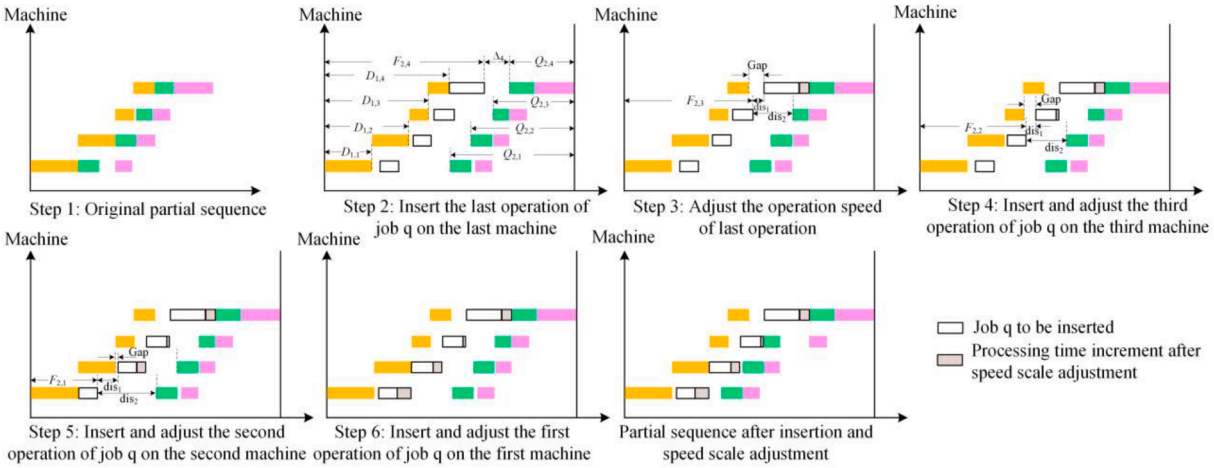

Fig. 8. Illustration of job insertion procedure with speed adjustment.

machines have been identified. After insertion, all the operations are unable to degrade the processing speed levels unless changing the job sequence (makespan) or the processing speeds of other jobs. Hence, it can be concluded that **ES** is a greedy-based search procedure essentially.

Pseudocode 2 ES insertion procedure

Input: job q to be inserted, maximum speed vector V_q^{sm}, makespan $C_{max}(\pi_q^h, V_q^{sm})$
Output: speed vector V_q^h after adjustment
01: calculate $D_{h,i}$, $Q_{h,i}$ and $F_{h,i}$
02: determine the $C_{max}(\pi_q^h, V_q^{sm})$ # insert q with V_q^{sm}
03: $\Delta_m = C_{max}(\Pi_q^h, V_q^{sm}) - F_{h,m} - Q_{h,m}$
04: Δ_m determines minimum speed level $v_{min} \in V$ of job q on the last machine, $p_{m,q,v_{min}} - p_{m,q,v^{sm}(m,q)} \leqslant \Delta_m$
05: set $v^h(q,m) = v_{min}$ # determine the speed level for job q on the last machine
06: **For** $i = m-1$ to 1
07: $Gap_{i+1} = \Delta_{i+1} - (p_{i+1,q,V^h(i+1,q)} - p_{i+1,q,V^{sm}(i+1,q)})$
08: $dis_1 = max\{D_{h-1,i+1} - F_{h,i}, 0\} + Gap_{i+1}$ # blocking constraint
09: $dis_2 = C_{max}(\Pi^h, V_{sm}) - F_{h,i} - Q_{h,i}$
10: $\Delta_i = min\{dis_1, dis_2\}$
11: Δ_i determines minimum speed level $v_{min} \in V$ on the other machines, $p_{m,q,v^h(q,i)} - p_{m,q,v^{sm}(q,i)} \leqslant \Delta_m$
12: set $V^h(q,i) = v_{min}$
13: **End For**

In brief, to ensure the diversity and distribution of solutions, the proposed speed adjustment operators are randomly applied to each individual of the population. The new individual is compared with the incumbent individual, and the non-dominated one will be kept. After neighborhood search, the non-dominated solutions found are recorded in **TA**. The overall neighborhood search procedure is sketched as **Pseudocode 3**.

Pseudocode 3 Neighborhood search

Input: population size *Ps*, temporary archive **TA**, job number n, factory number f, machine number m
Output: **TA**
01: **While** termination criterion not satisfies **do**
02: **For** $i = 1$ to Ps
03: # adjust the sequence π
04: select the non-dominated solutions from **TA** to construct a probabilistic model
05: update the probabilistic model $M\_js(g)$ by employing the Heb rule
06: generate new sequence $\pi_i(i = 1,\ldots,0.7*Ps)$ through sampling $M\_js(g+1)$
07: generate new sequence $\pi_i(i = 0.7*Ps+1,\ldots,Ps)$ through ECFPV heuristic
08: # adjust the speed level matrix V
09: implement the speed adjustment operators (ED or EC) randomly on each individual
10: update V_i
11: create a new schedule solution $\Pi_i = (\pi_i, V_i)$
12: **End For**
13: Evaluate the offspring based on the Pareto concept, store the non-dominated solutions in **TA**
14: **End While**

4.4. Pareto multi-objective local search

For an intelligent algorithm, the exploitation and exploration capacity should be well balanced. Although the neighborhood search can explore better solution space, it cannot fully contribute to the convergence of the whole population. In other words, MOEDA still needs a technique that helps avoid search stagnation and escape from the local optimum. In this study, two local search operators are designed and implemented on a randomly selected non-dominated solution from **TA**. The local search operators are expected to optimize the job sequence and speed synchronously.

Pareto local search operator 1 (Insertion): This operator refers to the factory with maximum makespan (denote as critical factory *fc*). Remove a job in *fc* and insert it in all possible positions of all other factories. When the makespan can not be improved anymore, the insertion step restarts by choosing another job in *fc*.

Pareto local search operator 2 (Swap): Each job in *fc* is removed sequentially and swapped with all jobs of all the other factories. The swap procedure terminates until all jobs in *fc* have been visited.

To implement the above-designed operators, a local search framework is presented. A solution is labeled if it is fully explored by both two local search operators. All solutions generated in the local search phase are stored and evaluated. The non-dominated solutions are added in **TA**. The Pareto multi-objective local search procedure is presented as **Pseudocode 4**.

Pseudocode 4 Pareto multi-objective local search framework

Input: temporary archive **TA**, *Nset*1, *Nset*2
Output: TA
01: **While** Π in **TA** is not selected
02: *Nset*1 = Pareto local search insert (Π)
03: *Nset*2 = Pareto local search swap (Π)
04: **mark** Π as visited
05: **For** $\Pi_{new} \in \{Nset_1, Nset_2, \Pi\}$ **do**
06: if Π_{new} is not dominated by any solutions of **TA**
07: put Π_{new} in **TA**, delete the dominated solutions
08: **End For**
09: **End While**

4.5. Energy-saving method

When investigating the inherent characteristics of blocking flowshop, we found that the blocking time of a job could be converted to processing time by reducing its operation speed without affecting the makespan (**Property**). Inspired by this property, a non-critical-path-based energy-saving method is presented to optimize the energy consumption of all non-dominated solutions stored in **TA**. The critical path in flowshop refers to a sequential job sequence from the starting time to the completion time without intervals between any two adjacent jobs (Jiang & Wang, 2019). The operations apart from the critical path are defined as non-critical operations. The proposed energy-saving method is implemented on the non-critical operations of the solutions from **TA**. The energy-saving method is demonstrated in Fig. 9.

In Fig. 9(b), the red dotted line with an arrow depicts the critical path for a factory with 4 jobs processing on 4 machines. Firstly, the energy-saving method identifies the non-critical operations; then, the idle time $I_{\pi_k(j),i}$ and blocking time $B_{\pi_k(j),i}$ are calculated using the following equations:

$$I_{\pi_k(j),i} = max\{(D_{\pi_k(j+1),i} - p_{\pi_k(j+1),i}) - D_{\pi_k(j),i}, 0\}, \quad j = 1,\ldots,n, i = 1,\ldots,m, k = 1,\ldots,f \tag{25}$$

$$B_{\pi_k(j),i} = max\{D_{\pi_k(j-1),i+1} - (D_{\pi_k(j),i-1} + p_{\pi(j),i}), 0\}, \quad j = 1,\ldots,n, i = 1,\ldots,m, k = 1,\ldots,f \tag{26}$$

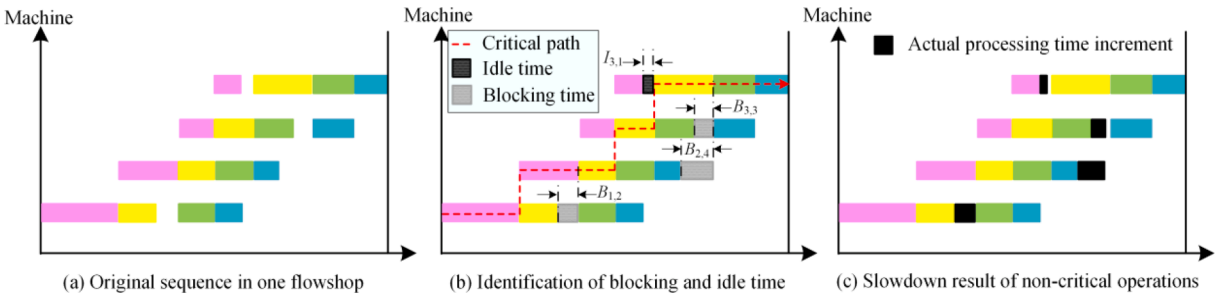

Fig. 9. Demonstration of critical path-based energy saving procedure.

The maximum allowable extension of processing time $max(\Delta T_I(u))$ and $max(\Delta T_B(v))$ should comply with the following principle:

$$max(\Delta T_I(u)) \leqslant I_{\pi_k(j),i} \tag{27}$$

$$max(\Delta T_B(v)) \leqslant B_{\pi_k(j),i} \tag{28}$$

By slowing down the operation speed to an appropriate level $v \in V$, the processing time is increased, and the energy consumption is therefore reduced. If there is no appropriate speed for the current operation, the original speed level is kept. Finally, the solutions after adjustment are evaluated and the non-dominated solutions are stored in **PA**. The energy-saving method is sketched in **Pseudocode 5**.

Pseudocode 5 Energy-saving method

Input: solutions in **TA**, **PA**
Output: **PA**
01: **For** each solution in **TA** **do**
02: identify the non-critical operations
03: calculate the idle time $I_{\pi_k(j),i}$ and blocking time $B_{\pi_k(j),i}$
04: calculate the maximum allowable extension time $max(\Delta T_I(u))$ and $max(\Delta T_B(v))$
05: identify the maximum allowable speed v
06: update the speed level matrix and corresponding processing time
07: **End For**
08: Evaluate the new solutions and store the non-dominated solutions in **PA**

4.6. Overview of MOEDA framework

Based on the previous descriptions, this study proposed a Pareto-based MOEDA to optimize EDBFSP. MOEDA mainly includes the following 4 parts: population initialization, neighborhood search, Pareto multi-objective local search, and energy conservation. First of all, the ECFPV heuristic aimed to generate initial solutions with good makespans. Secondly, the Bayesian network-based probabilistic model was constructed to learn the information from the non-dominated solutions. To reduce energy consumption, two speed adjustment operators were designed, and an improved insertion technique was embedded to realize the adaptive adjustment of the speed level during the job insertion procedure. Subsequently, a Pareto multi-objective local search framework that contains two search operators was proposed. Besides, an energy-saving method was performed to further decrease energy consumptions. The overall framework of MOEDA is depicted in **Pseudocode 6**. MOEDA is expected to balance the exploration and exploitation during the search procedure.

Pseudocode 6 Procedure of MOEDA

Input: population size Ps, learning rate α_u, temporary archive **TA**, Pareto archive **PA**
Output: Pareto archive **PA**
01: set **TA** and **PA** to be empty
02: generate the population with ECFPV and random heuristics, evaluate each individual
03: store the non-dominated solutions in **TA**
04: initialize the probabilistic model $M\_js(g)$
05: **While** termination criterion not meet **do**

(*continued on next column*)

(*continued*)

Pseudocode 6 Procedure of MOEDA

06: extract the non-dominated solutions in **TA** to construct the probabilistic model $M\_js(g)$
07: sample $M\_js(g)$ and generate a new sequence π, random initialize speed level matrix $\boldsymbol{V}$
08: implement the search operators on each individual, update the speed level matrix $\boldsymbol{V}$
09: generate new individual by integrating $\boldsymbol{\pi}$ and $\boldsymbol{V}$
10: evaluate the offspring and update **TA**
11: perform Pareto multi-objective local search on the non-dominated solutions from **TA**
12: store the non-dominated solutions in **TA**
13: implement the energy-saving method on the non-dominated solutions in **TA**
14: evaluate the non-dominated solutions, set **PA** $=$ **PA** $\cup$ **TA**
15: **End While**
16: construct **PF**

5. Experimental validation

In this paragraph, the system performance of MOEDA is assessed in different dimensions. Firstly, the experimental environment and the performance indicators are introduced. Secondly, the key parameters of MOEDA are calibrated. Afterward, the effectiveness of the proposed algorithmic components and optimization strategies are validated. Finally, MOEDA is compared with other well-known metaheuristics developed for energy-aware DFSP.

5.1. Experiment setting

The algorithm is coded in Python and loaded on a PC with an Intel(R) Core(TM) i7-8700 CPU @ 3.2 GHz/16G RAM under Windows 10. Since there was no published benchmark for EDBFSP, we generate test instances based on (Deng & Wang, 2017; Deng & Wang et al., 2016; Ding et al., 2016; Wang et al., 2018). The parameter scale is defined as: $n = \{20,30,50,80,100,200\}$, $m = \{5,10,20\}$ and $f = \{2,3,4,5\}$. There are in total 48 combinations, each of which contains 10 different instances. As a result, the test benchmark comprises 480 instances. The processing time for each operation is randomly generated and fixed, which is set uniformly distributed between 1 and 99. Five operation processing speed levels are pre-defined as $V = \{1,1.3,1.55,1.75,2.1\}$. According to literature (Deng & Wang et al., 2016), the energy consumption indicators are defined as $PE_{\pi_k(l),i,v} = 4 * v^2$ and $SE = 1.5$, respectively. As a matter of experience, the experiment results can approximate the real objective values when the replication number of the simulation is set infinitely. However, the simulation is limited by the computational capacity. Thus, in the experiment, the replication number is set as 10.

5.2. Evaluation indicators

Strictly speaking, to evaluate the performance of a multi-objective solver comprehensively and scrupulously, the evaluation index should cover at least two aspects: unary indicator and binary indicator, which describe the convergence and distribution (diversity) of the obtained

solutions (Wang et al., 2018). It also implies that "good" non-dominated solutions found by one particular algorithm should access the Pareto front as closely as possible while covering the Pareto front as widely as possible. In this study, we adopt three metrics for the experiment:

Unary Hypervolume metric (I_H): I_H scales the normalized amount of space that is formed by the non-dominated/Pareto-optimal solutions. Meanwhile, the covered normalized space considers reference points (optimum and anti-optimum) as its boundary for each objective.

The I_H of a bi-objective is formulated as follows:

$$I_H(S) = VOL\left(\bigcup_{x\in P}\left|f_1'(y), z_1\right| \times \left|f_2'(y), z_2\right|\right) \tag{29}$$

where $VOL(\cdot)$ represents the Lebesgue measurement, f_1' and f_2' denote the normalized objective values, $\boldsymbol{S}$ is the non-dominated solution set used for constructing the solution space. $z^* = (z_1, z_2)$ denotes the referent points set for demarcating the optimum and anti-optimum boundaries. The accuracy of calculating the I_H relies highly on the selection of reference points. To create a fair comparison, the reference points are normally set as $z_1 > 1$ and $z_2 > 1$(Ciavotta et al., 2013; Shao et al., 2019). Like (Shao et al., 2019), we set $(z_1, z_2) = (1.2, 1.2)$ to guarantees a relative largerI_H, which further ensures better coverage and convergence of the Pareto front.

Binary coverage metric (I_C) (Deng & Wang et al., 2016): I_C evaluates the quality of solutions by measuring the dominance relationship between the solution sets $\boldsymbol{P}_1$ and $\boldsymbol{P}_2$ that obtained via two comparing algorithms. The index $C(\boldsymbol{P}_1, \boldsymbol{P}_2)$ denotes the ratio of the solutions in $\boldsymbol{P}_2$ that are dominated by the solutions in$\boldsymbol{P}_1$:

$$C(\boldsymbol{P}_1, \boldsymbol{P}_2) = |\{\boldsymbol{b} \in \boldsymbol{P}_2 | \exists \boldsymbol{a} \in \boldsymbol{P}_1, \boldsymbol{a} \succ \boldsymbol{b} or \boldsymbol{a} \succ \boldsymbol{b}\}/|\boldsymbol{P}_2| \tag{30}$$

where $|\boldsymbol{P}_2|$ represents the solution number in$\boldsymbol{P}_2$, $\boldsymbol{a}$ and $\boldsymbol{b}$ represent two comparing solutions. Apparently, The larger $C(\boldsymbol{P}_1, \boldsymbol{P}_2)$ is, the better solutions in P_1 than solutions inP_2.

The non-dominance ratio (I_R) (Ishibuchi et al., 2017): I_R estimates the rate of the non-dominated solutions obtained by a certain algorithm to the non-dominated solutions provided by all solution sets:

$$RS = |S\{\boldsymbol{x} \in S | \exists \boldsymbol{y} \in FS, \boldsymbol{x} = \boldsymbol{y}\}|/|FS| \tag{31}$$

where $\boldsymbol{S}$ is the solution set considered, FS denotes the total solution sets obtained by integrating all single solution sets. $\boldsymbol{x}$ and $\boldsymbol{y}$ represent two compared solutions. It can be found that the greater the value of RS is, the better the performance of the specified solution set (algorithm) has. With the above-mentioned indicators, the experiments are implemented from the following perspectives:

1) Calibration of the key parameters;
2) Effectiveness validation of algorithmic components and optimization strategies;
3) Comparison with other metaheuristics designed for multi-objective scheduling problems.

5.3. Parameter calibration

MOEDA contains two key parameters: population size Ps and learning rateα_u. We employed the well-known Design of Experiment (DOE) to calibrate the key parameters. The candidate values for the parameters are set as follows:$Ps = \{20, 40, 60, 80, 100\}$, and$\alpha_u = \{0.1, 0.2, 0.3, 0.4\}$. 60 instances are randomly generated for calibration, with $90*n*m*f$ milliseconds as the standard termination criterion. The two key parameters are denoted as the control factors and I_H is defined as the response variable. The Analysis of Variance (ANOVA) is applied to analyze the calibration results. Table 1 shows the statistical results of ANOVA. In Table 1, F-ratio represents the influence of control factors, p-value represents whether the control factors have significant influences on the response value. The confidence interval is set at 95%.

Table 1
ANOVA Analysis results of parameter combinations of MOEDA.

Source	Sum sq.	d.f.	Mean Sq.	F-ratio	p-value
Ps	2.56	4	0.64	4.79	0.0071
α_u	4.89	3	1.63	8.86	0.0008
$Ps*\alpha_u$	7.40	12	0.62	3.55	0.8839

As can be seen from Table 1, the p-values of Ps and α_u are 0.0071 and 0.0008, which are smaller than 0.05. It means that both Ps and α_u have a great influence on the performance of MOEDA. The F-ratio of α_u is greater than that of Ps, which indicates that α_u has a greater impact on algorithm performance. The main effect plot is shown in Fig. 10. As can be seen, with the increment of Ps, the performance becomes better. This is because the increment of Ps can enrich the diversity of the population, which improves the probability of obtaining optimal solutions. However, an overlarge Ps may waste computing resources, which leads to the compression of local search time, thereby, the search efficiency of the algorithm is affected. It can be observed that, when$\alpha_u = 0.2$, the algorithm performs its best. Based on the above analysis, the key parameters of MOEDA are set as $Ps = 60$ and$\alpha_u = 0.2$.

5.4. Effectiveness of the proposed optimization strategies

In this section, the effectiveness of the proposed algorithmic components (initialization strategy, neighborhood structure, and local search scheme) of MOEDA are validated. We generated 4 variant algorithms, i.e., MOEDA_NI, MOEDA_NN, MOEDA_NL and MOEDA_NES from the original MOEDA. Each variant only varied one component comparing to the proposed MOEDA. The acronym MOEDA_NI denotes MOEDA without initialization strategy, while MOEDA_NN, MOEDA_NL, and MOEDA_NES represent MOEDA without neighborhood structure, local search scheme, and energy conversation strategy, respectively. In the experiment, 60 instances for factory number $f = 3$ were chosen as the test instances, I_H and I_R were chosen as the performance indicators. All variants adopted the same elapsed CPU time as the termination criterion. Table 1 demonstrates the statistical results of the two performance indicators. Table 2 presents the average values of results across 60 test instances with 10 replicative runnings per instance, and the best results of each combination are emphasized in bold.

As seen from Table 2, MOEDA has obtained larger I_H than MOEDA_NI, MOEDA_NN, MOEDA_NL, and MOEDA_NES, which implies the proposed optimization strategies have made significant contributions to MOEDA. To be specific, MOEDA_NN obtained the worst computational results, which indicates that the neighborhood search strategy influences the performance of MOEDA more. The neighborhood search plays the role in generating new populations and record non-dominated solutions. The speed-up mechanism has reduced the evaluation time of the generated neighboring solutions, and thereby strengthens the search efficiency effectively. The exploration of the entire algorithm would enormously reduce when the neighborhood search is excluded. MOEDA_NL obtained the second-worst results, which reveal that the local search scheme has also made a large contribution in boosting the performance of MOEDA. The local search scheme aims to help MOEDA escape from the local optimum. When the neighbor structures of the non-dominated solutions have not been renewed for a long time, meanwhile local search operators are not implemented, the evolution procedure of the population could stepwise stagnate and the algorithm would trap into local optimum. Thus, the exploitation of MOEDA can be enhanced with the proposed local search scheme. MOEDA_NI with a random initialization approach is also inferior to MOEDA, which indicates that the initialization scheme affects MOEDA partially. The ECFPV heuristic aims to start from the best region for makespan, and

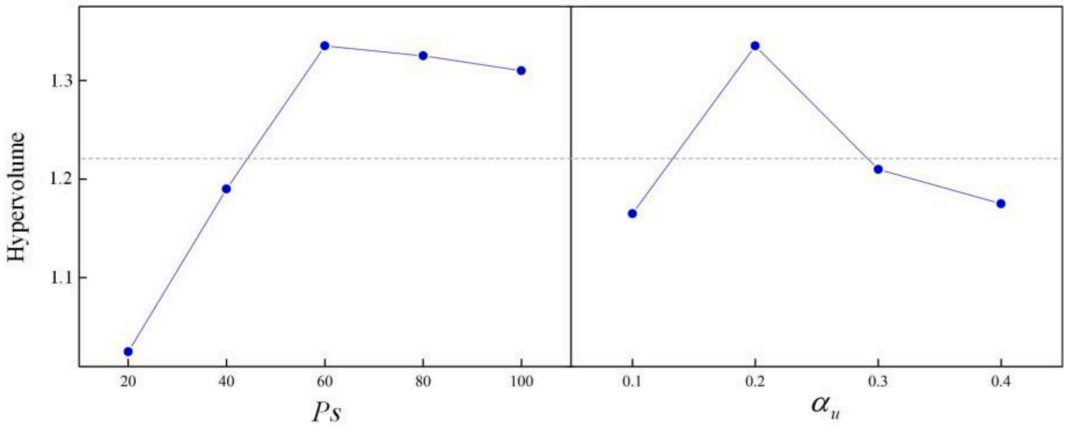

Fig. 10. Main effects plot of parameters for MOEDA.

Table 2
Comparisons between MOEDA and its variant algorithms.

$n*m*f$	Indicators	Variant algorithms				
		MOEDA_NI	MOEDA_NN	MOEDA_NL	MOEDA_NES	MOEDA
20*5*3	I_H	1.29	1.27	1.27	1.36	**1.36**
	I_R	0.27	0.19	0.25	0.87	**0.87**
30*5*3	I_H	1.26	1.25	1.27	1.32	**1.35**
	I_R	0.23	0.17	0.24	0.71	**0.88**
30*10*3	I_H	1.25	1.22	1.24	1.34	**1.37**
	I_R	0.22	0.15	0.19	0.73	**0.88**
50*5*3	I_H	1.24	1.20	1.25	1.31	**1.34**
	I_R	0.16	0.16	0.17	0.75	**0.93**
50*10*3	I_H	1.25	1.19	1.23	1.28	**1.31**
	I_R	0.16	0.15	0.16	0.66	**0.96**
80*5*3	I_H	1.22	1.18	1.19	1.28	**1.32**
	I_R	0.13	0.00	0.10	0.75	**1.00**
80*10*3	I_H	1.22	1.15	1.19	1.25	**1.34**
	I_R	0.12	0.00	0.11	0.80	**1.00**
80*20*3	I_H	1.19	1.14	1.17	1.25	**1.32**
	I_R	0.10	0.00	0.08	0.64	**1.00**
100*10*3	I_H	1.18	1.12	1.15	1.26	**1.32**
	I_R	0.05	0.00	0.00	0.81	**1.00**
100*20*3	I_H	1.17	1.13	1.17	1.25	**1.34**
	I_R	0.00	0.00	0.00	0.83	**1.00**
200*10*3	I_H	1.15	1.14	1.18	1.21	**1.33**
	I_R	0.00	0.00	0.00	0.76	**1.00**
200*20*3	I_H	1.17	1.11	1.15	1.23	**1.31**
	I_R	0.00	0.00	0.00	0.89	**1.00**

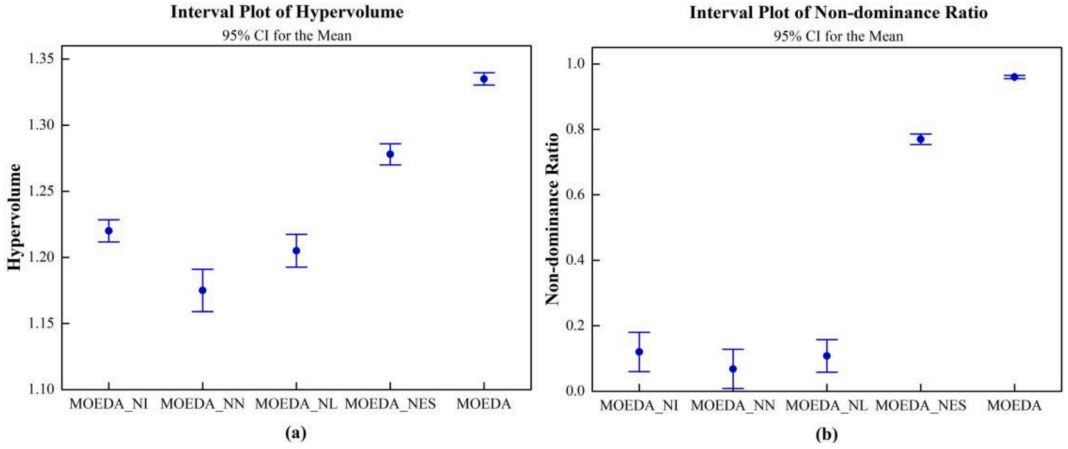

Fig. 11. Mean plots with 95% confidence intervals of I_H and I_R for all compared variants of MOEDA.

Table 3
Statistical results for binary indicator I_c of all compared algorithms.

f	C (MOEDA, MOEDA_NI)		C (MOEDA, MOEDA_NN)		C(MOEDA, MOEDA_NL)		C(MOEDA, MOEDA_NES)	
	(M, M_NI)	(M_NI, M)	(M, M_NN)	(M_NN, M)	(M, M_NL)	(M_NL, M)	(M, M_NES)	(M_NES, M)
2	0.78	0.19	0.88	0.10	0.80	0.16	0.59	0.38
3	0.75	0.23	0.89	0.09	0.82	0.12	0.62	0.41
4	0.80	0.16	0.86	0.07	0.82	0.11	0.53	0.36
5	0.77	0.20	0.84	0.05	0.79	0.17	0.47	0.29

stepwise guide the search to the energy-related objective. Thus, ECFPV can provide good start points for MOEDA. The MOEDA_NES performs worse than MOEDA, but better than other variant algorithms. Essentially, the energy-saving strategy is the perturbation of non-dominant solutions. The result has shown that it can influence the performance of MOEDA partially and increase the diversity of the solutions. Besides, the non-dominance ratio I_R of all the variants is inferior to MOEDA, which further reveals that the proposed optimization components are crucial for MOEDA in obtaining Pareto optimal solutions.

To further confirm the aforementioned conclusions, the mean and 95% confidence interval plot of the results from Table 2 are presented in Fig. 11. Note that the overlapping intervals imply no significant differences from statistical views. As shown in Fig. 11, the 95% confidence interval of MOEDA places over all variants of MOEDA without any overlapping, which reveals that there are significant differences between MOEDA and all variant algorithms. The effectiveness of MOEDA also confirms the necessity of each proposed improvement strategies. A lack of any components will degrade the overall performance of the algorithm.

Furthermore, the statistical results over indicator I_C between the original MOEDA and the variant algorithms are shown in Table 3. Due to space limitations, the MOEDA, MOEDA_NI, MOEDA_NN, MOEDA_NL and MOEDA_NES are represented by M, M_NI, M_NN, M_NL and M_NES, respectively. As can be seen in Table 3, all groups of comparisons share one feature that MOEDA is superior to its variant algorithms overI_C. Since I_C measures the quality of solutions, the results either imply that the solution set obtained by MOEDA approximates more closely to the Pareto optimal set. As a rule, to obtain good search results, the cooperation of all algorithmic components is indispensable.

5.5. Comparison with well-known approaches

Since there are no dedicated algorithms designed for solving EDBFSP, we compared MOEDA with the following metaheuristics: CMA (Deng & Wang, 2017), CA (Wang et al., 2017), KCA (Wang & Wang, 2018), and EDE (Zhao et al., 2020). CMA was proposed for optimizing the carbon footprint of DPFSP, CA was developed for energy-aware DFSP with no wait constraint (EDNWFSP), and KCA was designed for energy-conscious DPFSP. The abovementioned metaheuristics have demonstrated superiorities over the classical algorithm such as NSGA-II and SPEA-II. EDE was originally designed for DBFSP with makespan as single performance criterion. It was modified to optimize EDBFSP, that is, the Pareto concept was embedded in EDE. The speed-related variables were adopted with random strategy in different search stages, i.e. initialization, neighborhood search, and local search. Since all other compare metaheuristics have energy-saving steps, EDE was also equipped with this strategy. In the experiment, all metaheuristics were coded under the same compiling environment, involving the same programming language. We hew to obey the presentations of literature to implement the compared algorithms. Even if they were not originally proposed for EDBFSP, the core search ideas including algorithmic structure, solution presentation method, global and local search methods were preserved. Besides, the same data structure, fitness function, and library functions (e.g. insertion mechanism) were shared for all metaheuristics, to avoid the influences of different implementations on the performance of each metaheuristic. The parameters of the compared algorithms were fine-tuned for solving EDBFSP. The parameter settings of compared algorithms are listed in Table 4.

Table 4
Parameter determination of compared metaheuristics.

Algorithm	Parameter setting
CMA	Population size 50; local search control factor 400
CA	Population size 30; initialization rate with NEH 0.7; local search control factor 300
EDE	Population size 20; mutation factor 0.3; crossover rate 0.3; scale operator λ 0.5; ρ 0.5
KCA	Population size 10; proportion of EENEHFF2 initialization 0.5; local search control factor 80

The experiments were conducted based on the 480 instances generated in Section 5.1. To avoid the effect of randomness, each algorithm has run 10 replications for each instance independently. The termination criterion is set as $90 * n * m * f$ milliseconds CPU elapsed time. The statistical results in terms of three indicators for each combination are presented in Tables 5 to 8, which are grouped by different numbers of factories.

From Tables 5 to 8, it can be seen that in terms of all performance indicators, MOEDA outperforms its competitors on the instances. Whether on HypervolumeI_H, non-dominance ratioI_R , or coverage metricI_C, the results obtained by MOEDA are close to optimum values (1.44 forI_H, 1.0 forI_R, and 1.0 for I_C), which also implies that MOEDA achieved nearest to the reference Pareto front.

Although Tables 5-8 have validated the effectiveness of MOEDA, the results are still not convincing enough since the compared algorithms always have stochastic behaviors during their runtime. There is a need to take additional tests to identify whether the results between different approaches are statistically significant. For this purpose, two statistical tests including a parametric test and a non-parametric test were conducted, which are described as follows.

For the parametric test, two ANOVA tests were implemented, in which the performance indicators I_H and I_R were considered as the response variables. The results grouped by different factories are presented in the ANOVA plots with 95% confidence intervals in Fig. 12. As can be seen from Fig. 12, there are no overlapping intervals between MOEDA and other compared metaheuristics on each single test problem, which implies clearly that MOEDA has statistical differences with compared algorithms. In other words, MOEDA is statistically significantly better than other metaheuristics. Moreover, it can be observed that all the compared metaheuristics performed well on small-scale instances (e.g. $n = 20$), but degrade more or less when the problem scales become more complicated. To investigate such influence, the interaction plots between algorithms and the combination of $n * m$ are demonstrated in Fig. 13. As revealed in Fig. 13(a), the compared metaheuristics have shown strong competitiveness over small-scale instances, the obtained results of Hypervolume are very close to those of MOEDA. With the increment of the instance scales, the performances of all metaheuristics have declined. Especially for compared metaheuristics, the obtained results have shown that their performances sharply reduced over larger instances. Different from them, the performance of MOEDA is slightly affected. From Fig. 13(b), it can be seen that the non-dominance ratio obtained by MOEDA along each combination maintains an upward trend, whereas the values from the other metaheuristics

Table 5
The comparison result of different algorithms ($f = 2$).

$n*m*f$	CMA		CA		EDE		KCA		MOEDA		C(MOEDA, CMA)		C(MOEDA, CA)		C(MOEDA, EDE)		C(MOEDA, KCA)	
	I_H	I_R	I_H	I_R	I_H	I_R	I_H	I_R	I_H	I_R	MOEDA	CMA	MOEDA	CA	MOEDA	EDE	MOEDA	KCA
20*5*2	1.30	0.34	1.30	0.39	1.36	0.54	1.36	0.54	1.39	0.64	0.65	0.31	0.63	0.47	0.61	0.45	0.58	0.53
30*5*2	1.27	0.34	1.28	0.35	1.34	0.52	1.36	0.50	1.36	0.63	0.60	0.24	0.53	0.22	0.60	0.43	0.57	0.48
30*10*2	1.25	0.31	1.25	0.31	1.32	0.51	1.33	0.48	1.34	0.66	0.57	0.25	0.50	0.39	0.65	0.43	0.55	0.51
50*5*2	1.28	0.33	1.21	0.34	1.33	0.47	1.32	0.47	1.34	0.64	0.66	0.26	0.54	0.28	0.69	0.39	0.61	0.42
50*10*2	1.26	0.29	1.19	0.29	1.30	0.53	1.31	0.44	1.33	0.69	0.70	0.20	0.55	0.16	0.70	0.35	0.66	0.40
80*5*2	1.21	0.20	1.21	0.28	1.29	0.39	1.32	0.42	1.33	0.87	0.75	0.14	0.57	0.23	0.73	0.30	0.69	0.35
80*10*2	1.18	0.13	1.20	0.20	1.30	0.37	1.30	0.40	1.34	0.85	0.82	0.08	0.76	0.13	0.77	0.24	0.71	0.32
80*20*2	1.16	0.10	1.16	0.15	1.28	0.31	1.29	0.35	1.33	0.90	0.89	0.00	0.86	0.00	0.85	0.16	0.78	0.24
100*10*2	1.22	0.10	1.18	0.12	1.27	0.24	1.27	0.30	1.35	0.91	1.00	0.00	1.00	0.00	0.93	0.07	0.90	0.18
100*20*2	1.21	0.06	1.23	0.08	1.27	0.21	1.27	0.28	1.33	1.00	1.00	0.00	1.00	0.00	0.97	0.00	0.96	0.09
200*10*2	1.19	0.00	1.19	0.10	1.25	0.20	1.26	0.25	1.37	1.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00
200*20*2	1.15	0.00	1.16	0.00	1.25	0.15	1.27	0.20	1.32	1.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00

Table 6
Comparison result of different algorithms ($f = 3$).

$n*m*f$	CMA		CA		EDE		KCA		MOEDA		C(MOEDA, CMA)		C(MOEDA, CA)		C(MOEDA, EDE)		C(MOEDA, KCA)	
	I_H	I_R	I_H	I_R	I_H	I_R	I_H	I_R	I_H	I_R	MOEDA	CMA	MOEDA	CA	MOEDA	EDE	MOEDA	KCA
20*5*3	1.33	0.40	1.33	0.42	1.37	0.55	1.37	0.55	1.39	0.59	0.63	0.26	0.59	0.28	0.54	0.46	0.53	0.49
30*5*3	1.30	0.39	1.30	0.45	1.37	0.53	1.37	0.53	1.37	0.64	0.53	0.25	0.57	0.31	0.49	0.44	0.50	0.47
30*10*3	1.28	0.39	1.29	0.42	1.36	0.53	1.35	0.53	1.38	0.69	0.56	0.21	0.55	0.14	0.45	0.41	0.45	0.42
50*5*3	1.27	0.33	1.27	0.36	1.33	0.52	1.33	0.51	1.36	0.66	0.59	0.18	0.57	0.32	0.44	0.38	0.46	0.43
50*10*3	1.26	0.28	1.28	0.35	1.31	0.46	1.34	0.47	1.35	0.73	0.66	0.10	0.56	0.29	0.53	0.36	0.50	0.41
80*5*3	1.24	0.22	1.28	0.29	1.30	0.38	1.34	0.44	1.38	0.81	0.69	0.00	0.84	0.10	0.61	0.31	0.59	0.34
80*10*3	1.20	0.15	1.24	0.26	1.30	0.34	1.32	0.38	1.37	0.80	1.00	0.00	0.99	0.00	0.70	0.27	0.65	0.32
80*20*3	1.18	0.10	1.26	0.27	1.27	0.34	1.31	0.37	1.38	0.85	1.00	0.00	1.00	0.00	0.84	0.19	0.71	0.3
100*10*3	1.20	0.08	1.25	0.20	1.28	0.25	1.32	0.35	1.35	0.87	1.00	0.00	1.00	0.00	0.93	0.09	0.81	0.25
100*20*3	1.19	0.05	1.17	0.10	1.28	0.25	1.29	0.35	1.37	0.92	1.00	0.00	1.00	0.00	0.97	0.05	0.94	0.11
200*10*3	1.18	0.00	1.18	0.00	1.26	0.23	1.30	0.27	1.34	0.97	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00
200*20*3	1.19	0.00	1.22	0.00	1.26	0.17	1.28	0.23	1.35	1.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00

Table 7
Comparison result of different algorithms ($f = 4$).

$n*m*f$	CMA		CA		EDE		KCA		MOEDA		C(MOEDA, CMA)		C(MOEDA, CA)		C(MOEDA, EDE)		C(MOEDA, KCA)	
	I_H	I_R	I_H	I_R	I_H	I_R	I_H	I_R	I_H	I_R	MOEDA	CMA	MOEDA	CA	MOEDA	EDE	MOEDA	KCA
20*5*4	1.32	0.42	1.34	0.45	1.37	0.56	1.38	0.56	1.39	0.65	0.56	0.33	0.42	0.37	0.57	0.42	0.55	0.49
30*5*4	1.32	0.39	1.34	0.43	1.36	0.52	1.38	0.53	1.38	0.69	0.51	0.38	0.49	0.32	0.52	0.40	0.50	0.43
30*10*4	1.31	0.37	1.32	0.48	1.34	0.47	1.37	0.51	1.38	0.68	0.59	0.17	0.54	0.35	0.46	0.35	0.48	0.40
50*5*4	1.30	0.36	1.31	0.39	1.33	0.48	1.37	0.45	1.39	0.72	0.47	0.24	0.56	0.31	0.58	0.32	0.52	0.47
50*10*4	1.27	0.27	1.25	0.37	1.34	0.41	1.36	0.43	1.39	0.80	0.56	0.26	0.57	0.29	0.63	0.26	0.59	0.38
80*5*4	1.26	0.25	1.27	0.30	1.35	0.44	1.36	0.44	1.38	0.80	0.65	0.21	0.62	0.23	0.70	0.21	0.67	0.34
80*10*4	1.18	0.18	1.26	0.29	1.30	0.38	1.33	0.46	1.35	0.82	0.72	0.12	0.64	0.17	0.74	0.22	0.74	0.27
80*20*4	1.22	0.11	1.24	0.22	1.31	0.34	1.33	0.40	1.35	0.87	0.90	0.00	0.89	0.18	0.82	0.15	0.81	0.24
100*10*4	1.25	0.09	1.25	0.25	1.30	0.26	1.34	0.38	1.36	0.90	1.00	0.00	1.00	0.00	0.83	0.11	0.79	0.19
100*20*4	1.18	0.05	1.22	0.20	1.27	0.21	1.30	0.32	1.37	0.94	1.00	0.00	1.00	0.00	0.91	0.05	0.93	0.09
200*10*4	1.20	0.05	1.24	0.21	1.25	0.21	1.28	0.27	1.34	0.97	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00
200*20*4	1.20	0.00	1.21	0.00	1.25	0.17	1.29	0.25	1.36	1.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00

Table 8
Comparison result of different algorithms ($f = 5$).

$n*m*f$	CA		CA		EDE		KCA		MOEDA		C(MOEDA, CMA)		C(MOEDA, CA)		C(MOEDA, EDE)		C(MOEDA, KCA)	
	I_H	I_R	I_H	I_R	I_H	I_R	I_H	I_R	I_H	I_R	MOEDA	CMA	MOEDA	CA	MOEDA	EDE	MOEDA	KCA
20*5*5	1.34	0.46	1.35	0.47	1.38	0.49	1.39	0.51	1.40	0.69	0.60	0.32	0.63	0.26	0.53	0.44	0.50	0.46
30*5*5	1.33	0.44	1.33	0.39	1.37	0.49	1.38	0.50	1.39	0.66	0.59	0.29	0.52	0.36	0.50	0.39	0.47	0.40
30*10*5	1.30	0.39	1.32	0.38	1.35	0.45	1.37	0.48	1.39	0.64	0.58	0.29	0.49	0.31	0.47	0.34	0.46	0.41
50*5*5	1.28	0.35	1.29	0.36	1.34	0.46	1.35	0.48	1.37	0.62	0.67	0.26	0.53	0.29	0.48	0.31	0.41	0.35
50*10*5	1.27	0.30	1.30	0.33	1.35	0.43	1.32	0.45	1.38	0.65	0.59	0.22	0.46	0.25	0.49	0.33	0.51	0.36
80*5*5	1.29	0.26	1.27	0.28	1.34	0.42	1.33	0.46	1.36	0.77	0.66	0.21	0.58	0.26	0.56	0.33	0.55	0.34
80*10*5	1.25	0.23	1.31	0.31	1.33	0.39	1.33	0.45	1.37	0.72	0.72	0.19	0.63	0.25	0.62	0.19	0.59	0.28
80*20*5	1.23	0.23	1.25	0.25	1.30	0.39	1.32	0.43	1.35	0.87	0.64	0.14	0.69	0.19	0.76	0.14	0.68	0.21
100*10*5	1.25	0.24	1.28	0.27	1.30	0.37	1.33	0.40	1.39	0.89	0.70	0.00	0.84	0.10	0.83	0.11	0.74	0.19
100*20*5	1.22	0.15	1.27	0.21	1.28	0.32	1.32	0.38	1.37	0.91	0.89	0.00	0.82	0.00	0.84	0.06	0.85	0.13
200*10*5	1.21	0.07	1.26	0.00	1.27	0.31	1.31	0.33	1.38	0.95	1.00	0.00	0.95	0.00	0.96	0.04	0.91	0.12
200*20*5	1.22	0.00	1.26	0.00	1.27	0.20	1.30	0.25	1.35	0.95	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00

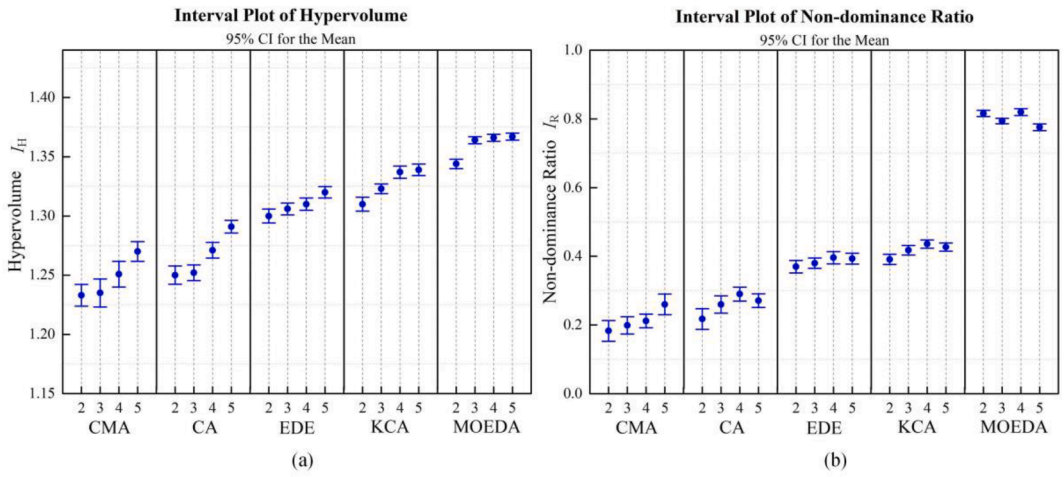

Fig. 12. Mean plots with 95% confidence intervals of I_H and I_R for all compared metaheuristics.

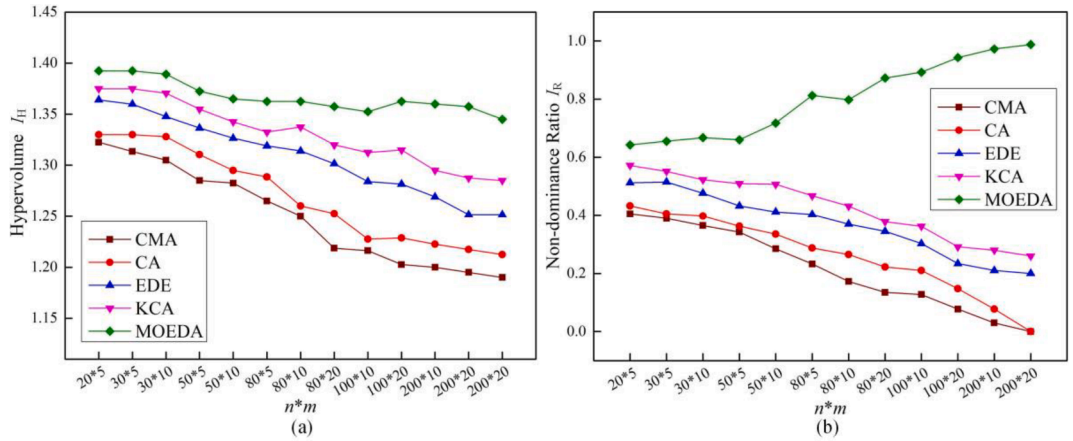

Fig. 13. Mean plots with 95% confidence intervals of all compared algorithms and combinations for indicators: (a) HypervolumeI_H; (b) Non-dominance ratioI_R.

Table 9
Result of Holm's test.

k	$\delta/(r+1-k)$	Hypervolume I_H			Non-dominance ratio I_R		
		Hypotheses	p-value	R/A	Hypotheses	p-value	R/A
1	0.0125	MOEDA = CMA	1.88E-04	R	MOEDA = CMA	3.49E-04	R
2	0.0167	MOEDA = CA	8.65E-03	R	MOEDA = CA	1.51E-04	R
3	0.025	MOEDA = EDE	5.53E-03	R	MOEDA = EDE	7.96E-03	R
4	0.05	MOEDA = KCA	3.94E-03	R	MOEDA = KCA	5.87E-03	R

Note: R/A denotes reject/accept the hypotheses.

degrade severely. It implies that MOEDA is more efficient in finding high-quality solutions that are proximal to the Pareto front. In the concrete, MOEDA has shown stability in different scales of test instances, which also reveals that its robustness is better than the compared algorithms.

2) For the non-parametric test, Holm's method is employed. Holm's test is a closed testing procedure designed for multiple hypotheses. In the test, a set $H = \{H_1, \ldots, H_r\}$ contains r hypotheses, the set $P = \{P_1, \ldots, P_r\}$ contains the p-values that represent the probabilities of monitoring the statistical results by accident. The p-values are sorted in ascending order as$p_{(1)} \leqslant p_{(2)} \leqslant \ldots \leqslant p_{(r)}$. Let δ be a significance threshold that determines to reject the null hypothesis, and k represents a minimum indicatrix that satisfies the inequality relations of$p_k \geqslant \delta/(r+1-k)$. As a result, null hypotheses from H_1 to H_{k-1} are rejected. In contrast, the hypotheses from H_k to H_r are accepted. In this study, the performance indicators I_H and I_R are defined as the response variables. MOEDA is considered as the reference algorithm, and the other two algorithms are assumed respectively equal to MOEDA. Hence, in total two hypotheses are yielded. To demonstrate the differences clearly, a 95% confidence level is also considered, which means$\delta = 0.05$. As shown in Table 9, all revenant p-values of hypotheses H_r on the performance indicators are close to zero, which are much smaller than the value ofδ. Hence, all hypotheses in this test are rejected. The test result reveals that MOEDA significantly outperforms the compared metaheuristics from the statistical view.

To visualize the algorithmic performances, the Pareto fronts obtained by the compared algorithms for certain instances (factory number $f = 3$) are sketched in Fig. 14. It is worth noting that the Pareto fronts discussed in this study are formed by those Pareto optimal solutions, which were selected in advance using the well-known fast non-dominated sorting method proposed by Deb et al. (Deb et al., 2002). As presented in Fig. 14, all Pareto fronts are relatively close and superposed on small instances (e.g., Instance 20_5_3_1, Instance 30_5_3_1, and Instance 30_10_3_1). With the increment of instance scales, the Pareto fronts of MOEDA are closer to the origin of coordinates, which indicates that MOEDA can find Pareto fronts that dominate those of other algorithms observably. It also reveals the strong exploration and

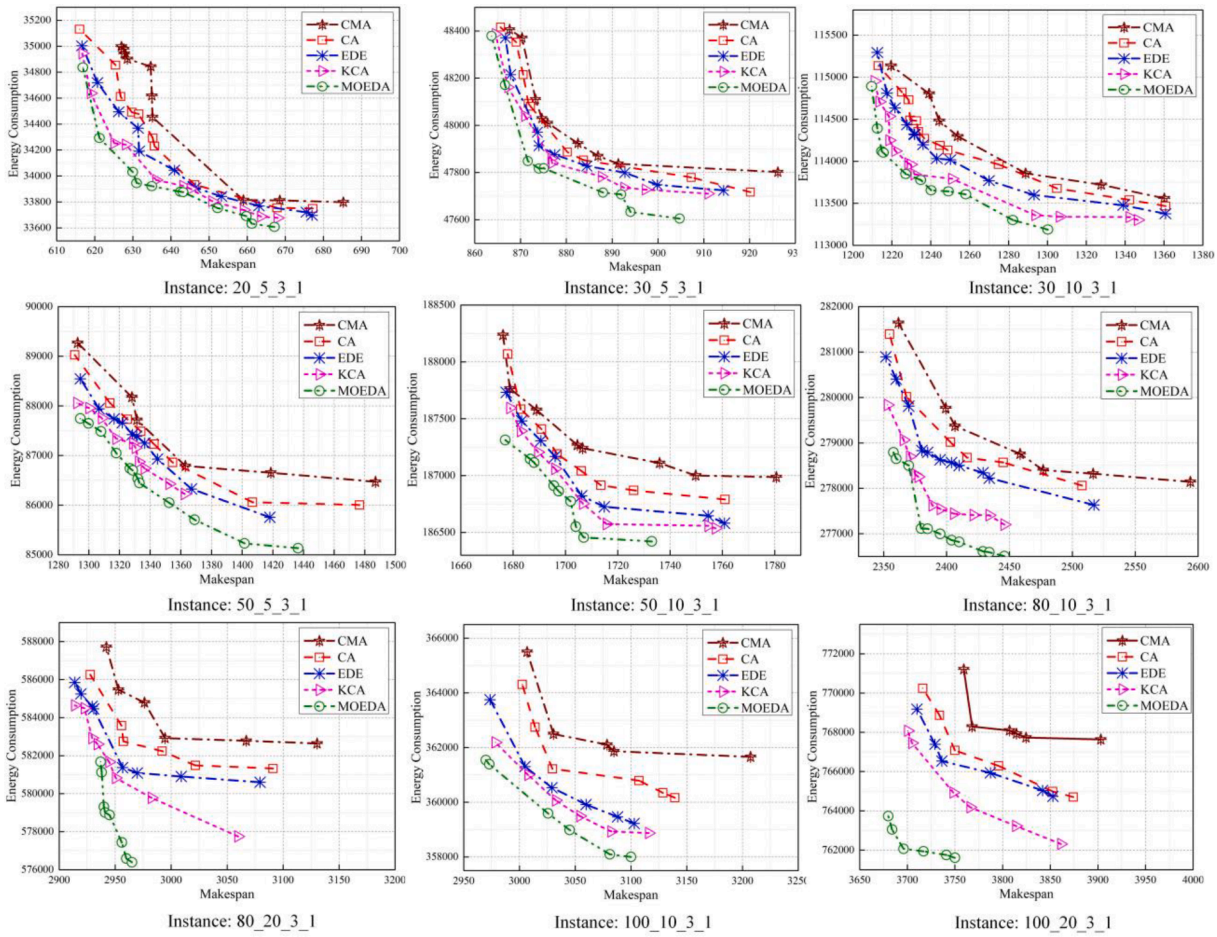

Fig. 14. The Pareto fronts obtained from selected test instances.

exploitation ability of MOEDA in both small and large instances.

To analyze the non-dominant solutions of **PF**, the test instance 50_5_3_1 is taken as an example. From the **PF** of MOEDA, it can be observed that the minimum mean makespan obtained by MOEDA is 1294.5, and the corresponding processing energy consumption of this solution is 87898.2. In the contrast, the minimum average energy consumption obtained by MOEDA is 85208.7, and the average mean makespan corresponding to this solution is 1436.5. When comparing the two solutions, the former one shortens the manufacturing time by about 9.87% but increases the total energy consumption by 3.16% correspondingly. This phenomenon also reflects the conflicted relationship between the two objectives. The decision-makers can make quantitative indicators according to the actual needs of manufacturing, to select the appropriate solution on the PF, and guide the scheduling task.

6. Conclusions and future works

This study investigated the EDBFSP that considers both productivity and energy consumption as optimization indicators. The adjustment of energy consumption is correlated to different operation speed levels of jobs under a pre-defined speed scaling framework. To solve EDBFSP, a multi-objective estimation of distribution algorithm (MOEDA) was proposed. First, a hybridized improved initialization strategy was designed to generate the population. Second, a Bayesian inference-based probabilistic model and specific speed adjustment operators were proposed to find better search spaces for both objectives. Afterward, a local search framework was proposed to perturb high-quality individuals, which aims at equilibrating the exploration and exploitation of the proposed algorithm. Lastly, an energy-saving method was implemented based on problem-specific characteristics. To validate its performance, the key parameter of MOEDA was calibrated. In the meantime, the performances of the proposed algorithmic components and optimization policies were verified in a simulative test. The proposed MOEDA was compared with two well-known metaheuristics on different benchmark instances. The analysis results demonstrated that MOEDA is significantly better than its competitors. The study also presents that energy consumption can be significantly reduced by employing energy-efficient machine behavior policies.

Other than conclusions, there are some observations reported for future work:

The presented MOEDA could be methodologically improved in future work. The neighborhood search framework might be implemented in an integrated way. Also, the search for Pareto non-dominated solutions could be enhanced in regions with higher makespans, despite the practical application scenario. Moreover, the parameter calibration would be optimized by employing it for different problem scales.

The assumption for estimating the energy consumption applied in this research is insufficiently precise. Relying on the research characteristics at the present stage, some other aspects corresponding to manufacturing planning and scheduling might be surveyed. For instance, it is meaningful to embed the energy effects of switching machines on and off explicitly.

CRediT authorship contribution statement

Xiaohui Zhang: Formal analysis, Writing – original draft. **Xinhua**

Liu: Methodology, Software, Investigation, Funding acquisition. **Andrzej Cichon:** Resources, Software, Writing – review & editing. **Grzegorz Królczyk:** Supervision, Data curation, Writing – review & editing, Funding acquisition. **Zhixiong Li:** Conceptualization, Resources, Supervision, Writing – review & editing, Funding acquisition.

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.

Acknowledgments

The support of the Natural Science Foundation of Xuzhou, China (KC21070), National Natural Science Foundation of China (51975568& 61803192), National Natural Science Foundation of Jiangsu Province (BK20191341), Narodowego Centrum Nauki (No. 2020/37/K/ST8/02748), Industry-University-Research Collaboration Project of Jiangsu Province (BY2021559) and Priority Academic Program Development of Jiangsu Higher Education Institutions (PAPD) in carrying out this research is gratefully acknowledged.

References

Amiri, M. F., & Behnamian, J. (2020). Multi-objective green flowshop scheduling problem under uncertainty: Estimation of distribution algorithm. *Journal of Cleaner Production, 251*, Article 119734. https://doi.org/10.1016/j.jclepro.2019.119734

Chen, S., Pan, Q. K., Hu, X., & Tasgetiren, M. F. (2020). An Iterated Greedy Algorithm for Distributed Blocking Flowshop Problems with Makespan Minimization. *In 2020 39th Chinese Control Conference* (CCC) (pp. 1536-1541), Shenyang, China: IEEE. 10.23919/CCC50068.2020.9188884.

Chen, S., Pan, Q. K., & Gao, L. (2021). Production scheduling for blocking flowshop in distributed environment using effective heuristics and iterated greedy algorithm. *Robotics and Computer-Integrated Manufacturing, 71*, Article 102155. https://doi.org/10.1016/j.rcim.2021.102155

Ciavotta, M., Minella, G., & Ruiz, R. (2013). Multi-objective sequence dependent setup times permutation flowshop: A new algorithm and a comprehensive study. *European Journal of Operational Research, 227*(2), 301–313. https://doi.org/10.1016/j.ejor.2012.12.031

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation, 6*(2), 182–197. https://doi.org/10.1109/4235.996017

Deng, J., Wang, L., Wu, C., Wang, J. & Zheng, X. (2016). A competitive memetic algorithm for carbon-efficient scheduling of distributed flow-shop. *In International Conference on Intelligent Computing* (pp. 476-488). Lanzhou, China: Springer. doi: 10.1007/978-3-319-42291-6_48.

Deng, J., & Wang, L. (2017). A competitive memetic algorithm for multi-objective distributed permutation flow shop scheduling problem. *Swarm and Evolutionary Computation, 32*, 121–131. https://doi.org/10.1016/j.swevo.2016.06.002

Ding, J. Y., Song, S., & Wu, C. (2016). Carbon-efficient scheduling of flow shops by multi-objective optimization. *European Journal of Operational Research, 248*(3), 758–771. https://doi.org/10.1016/j.ejor.2015.05.019

Fang, K., Uhan, N. A., Zhao, F., & Sutherland, J. W. (2011). A new approach to scheduling in manufacturing for power consumption and carbon footprint reduction. *Journal of Manufacturing Systems, 30*(4), 234–240. https://doi.org/10.1016/j.jmsy.2011.08.004

Fang, K., Uhan, N. A., Zhao, F., & Sutherland, J. W. (2013). Flow shop scheduling with peak power consumption constraints. *Annals of Operations Research, 206*(1), 115–145. https://doi.org/10.1007/s10479-012-1294-z

Fazli Khalaf, A., & Wang, Y. (2018). Energy-cost-aware flow shop scheduling considering intermittent renewables, energy storage, and real-time electricity pricing. *International Journal of Energy Research, 42*(12), 3928–3942. https://doi.org/10.1002/er.41

Feng, Y., Hong, Z., Li, Z., Zheng, H., & Tan, J. (2020). Integrated intelligent green scheduling of sustainable flexible workshop with edge computing considering uncertain machine state. *Journal of Cleaner Production, 246*, Article 119070. https://doi.org/10.1016/j.jclepro.2019.119070

Fu, Y., Tian, G., Fathollahi-Fard, A. M., Ahmadi, A., & Zhang, C. (2019). Stochastic multi-objective modelling and optimization of an energy-conscious distributed permutation flow shop scheduling problem with the total tardiness constraint. *Journal of Cleaner Production, 226*, 515–525. https://doi.org/10.1016/j.jclepro.2019.04.046

Gong, D., Han, Y., & Sun, J. (2018). A novel hybrid multi-objective artificial bee colony algorithm for blocking lot-streaming flo w shop scheduling problems. *Knowledge-Based Systems, 148*, 115–130. https://doi.org/10.1016/j.knosys.2018.02.029

Hamzadayı, A. (2020). An effective benders decomposition algorithm for solving the distributed permutation flowshop scheduling problem. *Computers & Operations Research, 123*, Article 105006. https://doi.org/10.1016/j.knosys.2021.106881

Hu, L., Yang, X., & Fan, H. (2018). Optimization of Contract Distribution Based on Multi-objective Estimation of Distribution Algorithm. In *Proceedings of the 2018 International Conference on Computing and Artificial Intelligence* (pp. 9–12). Chengdu, China: ACM. https://doi.org/10.1145/-3194452.3194455.

Ishibuchi, H., Imada, R., Setoguchi, Y., & Nojima, Y. (2017). Reference point specification in hypervolume calculation for fair comparison and efficient search. In *Proceedings of the Genetic and Evolutionary Computation Conference* (pp. 585–592). New York, NY, USA: ACM. https://doi.org/10.1145/-3071178.3071264.

Jiang, E., Wang, L. & Lu, J. (2017). Modified multiobjective evolutionary algorithm based on decomposition for low-carbon scheduling of distributed permutation flow-shop. *In 2017 IEEE Symposium Series on Computational Intelligence* (SSCI) (pp. 1-7). Honolulu, HI, USA: IEEE. 10.1109/SSCI.2017.8280893.

Jiang, E. D., & Wang, L. (2019). An improved multi-objective evolutionary algorithm based on decomposition for energy-efficient permutation flow shop scheduling problem with sequence-dependent setup time. *International Journal of Production Research, 57*(6), 1756–1771. https://doi.org/10.1080/00207543.2018.1504251

Jiang, S. L., & Zhang, L. (2019). Energy-oriented scheduling for hybrid flow shop with limited buffers through efficient multi-objective optimization. *IEEE Access, 7*, 34477–34487. https://doi.org/10.1109/ACCESS.2019.2904848

Jing, X. L., Pan, Q. K., & Gao, L. (2021). Local search-based metaheuristics for the robust distributed permutation flowshop problem. *Applied Soft Computing, 105*, Article 107247. https://doi.org/10.1016/j.asoc.-2021.107247

Lei, D., Li, M., & Wang, L. (2018). A two-phase meta-heuristic for multiobjective flexible job shop scheduling problem with total energy consumption threshold. *IEEE Transactions on Cybernetics, 49*(3), 1097–1109. https://doi.org/10.1109/TCYB.2018.2796119

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

Miller, B. L., & Goldberg, D. E. (1995). Genetic algorithms, tournament selection, and the effects of noise. *Complex Systems, 9*(3), 193–212. https://doi.org/10.1162/evco.1996.4.2.113

Miyata, H. H., & Nagano, M. S. (2019). The blocking flow shop scheduling problem: A comprehensive and conceptual review. *Expert Systems with Applications, 137*, 130–156. https://doi.org/10.1016/- j.eswa.2019.06.069

Mouzon, G., Yildirim, M. B., & Twomey, J. (2007). Operational methods for minimization of energy consumption of manufacturing equipment. *International Journal of Production Research, 45*(18), 4247–4271. https://doi.org/10.1080/00207540701450013

Mühlenbein, H., & Paass, G. (1996). In *From recombination of genes to the estimation of distributions I. Binary parameters* (pp. 178–187). Berlin, Germany: Springer. https://doi.org/10.1007/3-540-61723-X_982.

Naderi, B., & Ruiz, R. (2010). The distributed permutation flowshop scheduling problem. *Computers & Operations Research, 37*, 754–768. https://doi.org/10.1016/j.cor.2009.06.019

Nawaz, M., Enscore, E. E., Jr, & Ham, I. (1983). A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. *Omega, 11*(1), 91–95. https://doi.org/10.1016/0305-0483(83)90088-9

Öztop, H., Tasgetiren, M. F., Eliiyi, D. T., Pan, Q. K., & Kandiller, L. (2020). An energy-efficient permutation flowshop scheduling problem. *Expert Systems with Applications, 150*, Article 113279. https://doi.org/10.1016/j.eswa.2020.113279

Pan, Q. K., Wang, L., Gao, L., & Li, W. D. (2011). An effective hybrid discrete differential evolution algorithm for the flow shop scheduling with intermediate buffers. *Information Sciences, 181*(3), 668–685. https://doi.org/10.1016/j.ins.2010.10.00945

Pérez-Rodríguez, R., & Hernández-Aguirre, A. (2018). A hybrid estimation of distribution algorithm for flexible job-shop scheduling problems with process plan flexibility. *Applied Intelligence, 48*, 3707–3734. https://doi.org/10.1007/s10489-018-1160-z

Qian, B., Li, Z. C., & Hu, R. (2017). A copula-based hybrid estimation of distribution algorithm for m-machine reentrant permutation flow-shop scheduling problem. *Applied Soft Computing, 61*, 921–934. https://doi.org/10.1016/j.asoc.2017.08.037

Ruiz, R., Pan, Q. K., & Naderi, B. (2019). Iterated Greedy methods for the distributed permutation flowshop scheduling problem. *Omega, 83*, 213–222. https://doi.org/10.1016/j.omega.2018.03.004

Schulz, S., Neufeld, J. S., & Buscher, U. (2019). A multi-objective iterated local search algorithm for comprehensive energy-aware hybrid flow shop scheduling. *Journal of Cleaner Production, 224*, 421–434. https://doi.org/10.1016/j.jclepro.2019.03.155

Shao, Z., Pi, D., & Shao, W. (2018). Estimation of distribution algorithm with path relinking for the blocking flow-shop scheduling problem. *Engineering Optimization, 50*(5), 894–916. https://doi.org/10.1080/0305215X.2017.1353090

Shao, W., Pi, D., & Shao, Z. (2019). A pareto-based estimation of distribution algorithm for solving multiobjective distributed no-wait flow-shop scheduling problem with sequence-dependent setup time. *IEEE Transactions on Automation Science and Engineering, 16*(3), 1344–1360. https://doi.org/10.1109/TASE.2018.2886303

Shao, Z., Pi, D., & Shao, W. (2020a). Hybrid enhanced discrete fruit fly optimization algorithm for scheduling blocking flow-shop in distributed environment. *Expert Systems with Applications, 145*, Article 113147. https://doi.org/10.1016/j.eswa.2019.113147

Shao, Z., Shao, W., & Pi, D. (2020b). Effective constructive heuristic and metaheuristic for the distributed assembly blocking flow-shop scheduling problem. *Applied Intelligence, 50*(12), 4647–4669. https://doi.org/10.1007/s10489-020-01809-x

Song, L., Liu, C., Zhu, J., & Shi, H. (2017). Solving traveling salesman problem with hybrid estimation of distribution algorithm. In *2017 IEEE 7th Annual International*

Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER) (pp. 886–891). Honolulu, HI, USA: IEEE. https://doi.org/10.1109/CYBER.2017.8446236.

Tian, J., Hao, X., & Gen, M. (2019). A hybrid multi-objective EDA for robust resource constraint project scheduling with uncertainty. *Computers & Industrial Engineering, 130*, 317–326. https://doi.org/10.1016/j.cie.2019.02.039

Wang, F., Deng, G., Jiang, T., & Zhang, S. (2018). Multi-objective parallel variable neighborhood search for energy consumption scheduling in blocking flow shops. *IEEE Access, 6*, 68686–68700. https://doi.org/10.1109/ACCESS.2018.2879600

Wang, J., Wang, L., Wu, C. & Shen, J. (2017). A Cooperative Algorithm for Energy-efficient Scheduling of Distributed No-wait Flowshop. *In 2017 IEEE Symposium Series on Computational Intelligence* (SSCI) (pp. 1-8). Honolulu, HI, USA: IEEE. 10.1109/SSCI.2017.8280956.

Wang, R., Lai, S., Wu, G., Xing, L., Wang, L., & Ishibuchi, H. (2018). Multi-clustering via evolutionary multi-objective optimization. *Information Sciences, 450*, 128–140. https://doi.org/10.1016/j.ins.2018.03.047

Wang, J. J., & Wang, L. (2018). A knowledge-based cooperative algorithm for energy-efficient scheduling of distributed flow-shop. *IEEE Transactions on Systems, Man, and Cybernetics: Systems, 50*(5), 1805–1819. https://doi.org/10.1109/TSMC.2017.2788879

Wu, C. G., & Wang, L. (2018). A multi-model estimation of distribution algorithm for energy efficient scheduling under cloud computing system. *Journal of Parallel and Distributed Computing, 117*, 63–72. https://doi.org/10.1016/j.jpdc.2018.02.009

Yanai, K., & Iba, H. (2013). Estimation of distribution programming based on Bayesian network. In *Congress on Evolutionary Computation* (pp. 1618–1625). Canberra, ACT, Australia: IEEE. https://doi.org/10.1109/CEC.2003.1299866.

Ying, K. C., & Lin, S. W. (2017). Minimizing makespan in distributed blocking flowshops using hybrid iterated greedy algorithms. *IEEE Access, 5*, 15694–15705. https://doi.org/10.1109/ACCESS.2017.-2732738

Yüksel, D., Taşgetiren, M. F., Kandiller, L., & Gao, L. (2020). An energy-efficient bi-objective no-wait permutation flowshop scheduling problem to minimize total tardiness and total energy consumption. *Computers & Industrial Engineering, 145*, Article 106431. https://doi.org/10.1016/j.cie.2020.106431

Zhang, X., Liu, X., Tang, S., Królczyk, G., & Li, Z. (2019). Solving scheduling problem in a distributed manufacturing system using a discrete fruit fly optimization algorithm. *Energies, 12*(17), 3260. https://doi.org/10.3390/en12173260

Zhao, F., Zhao, L., Wang, L., & Song, H. (2020). An ensemble discrete differential evolution for the distributed blocking flowshop scheduling with minimizing makespan criterion. *Expert Systems with Applications, 160*, Article 113678. https://doi.org/10.1016/j.eswa.2020.113678

Zheng, J., Wang, L., & Wang, J. J. (2020). A cooperative coevolution algorithm for multi-objective fuzzy distributed hybrid flow shop. *Knowledge-Based Systems, 194*. https://doi.org/10.1016/j.knosys.2020.105536

Zhou, B. H., Hu, L. M., & Shao, Z. Y. (2018). A hybrid differential evolution algorithm with estimation of distribution algorithm for reentrant hybrid flow shop scheduling problem. *Neural Computing and Applications, 30*(1), 193–209. https://doi.org/10.1007/s00521-016-2692-y