Hybrid Estimation of Distribution Algorithm for Blocking Flow-Shop Scheduling Problem with Sequence-Dependent Setup Times

Zi-Qi Zhang[1,2], Bin Qian[1,2(✉)], Bo Liu[3], Rong Hu[1], and Chang-Sheng Zhang[1]

[1] School of Information Engineering and Automation, Kunming University of Science and Technology, Kunming 650500, China
bin.qian@vip.163.com

[2] School of Mechanical and Electrical Engineering, Kunming University of Science and Technology, Kunming 650500, China

[3] Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing 100190, China

Abstract. This paper presents an innovative hybrid estimation of distribution algorithm, named HEDA, for blocking flow-shop scheduling problem (BFSP) with sequence-dependent setup times (SDSTs) to minimize the makespan criterion, which has been proved to be typically NP-hard combinatorial optimization problem with strong engineering background. Firstly, several efficient heuristics are proposed according to the property of BFSP with SDSTs. Secondly, the genetic information of both the order of jobs and the promising blocks of jobs are concerned to generate the guided probabilistic model. Thirdly, after the HEDA-based global exploration, a reference sequence-based local search with path relinking technique is developed and incorporated into local exploitation to escape from local optima and improve the convergence property. Due to the reasonable balance between EDA-based global exploration and sequence dependent local exploitation as well as comprehensive utilization of the speedup evaluation method, the BFSP with SDSTs can be solved effectively and efficiently. Finally, computational results and comparisons with the existing state-of-the-art algorithms are carried out, which demonstrate the superiority of the proposed HEDA in terms of searching quality, robustness, and efficiency.

Keywords: Estimation of distribution algorithm
Blocking flow-shop scheduling problem · Sequence-dependent setup times
Path relinking

1 Introduction

Flow shop scheduling problems (FSSPs) have attracted much attention in the manufacturing systems of contemporary enterprises and have widely potential applications in the research field of both computer science and operation research. The blocking flow-shop scheduling problem (BFSP) is a special case of FSSPs, which considers that the buffer capacity between two machines may be absent. Many modern production and manufacturing systems can be modelled as the BFSP when no buffers exist between consecutive machines, e.g. serial manufacturing processes, chemical industry, iron and steel industry and robotic cell, etc. In BFSP, there are no infinite buffer capacity between consecutive machines. In other words, under the no buffers circumstances, the completed jobs on a machine have to be blocked on its machine until its next machine is available. In addition, each machine can handle no more than one job at a time and all jobs have to visit each machine exactly once. To the best of our knowledge, almost all of the works about BFSP usually focused on assuming the setup times on each machine as the part of processing times. However, the setup times are very important and need to be explicitly treated in some actual production process [1], which should be separated from the processing times and consider independently (e.g., cleaning, fixing or releasing parts to machines, changing tools, and adjustments to machines). Therefore, the main motivation of this paper is that we consider a kind of setup times in BFSP, i.e., sequence-dependent setup times (SDSTs), where SDSTs depend upon both current job being processed and the next job in the sequence.

For the computational complexity of the BFSP, Hall and Sriskandarajah proved that the BFSP with more than two machines was NP-hard in the strong sense [2]. That is to say, the BFSP with exponential computational complexity, which becomes more complicated and is difficult to solve completely with the increase of problem size. Because of the BFSP is proven to be NP-hard, let alone with SDSTs, many state-of-the-art approaches which include constructive heuristics and metaheuristics evolutionary algorithms (EAs) have been developed recent years to address BFSP with different objectives including makespan, total flow time and total tardiness, etc. Constructive heuristics utilize some rules to determine the relative priority of each job to construct a specific scheduling permutation, which usually consume less time but obtain unsatisfied performance. However, metaheuristics start from previous generated solutions and iteratively improve performance with domain knowledge through different evolutionary strategies, which usually obtain fairly satisfactory solutions but the processes are always time-consuming. For BFSP with makespan, Wang et al. [3] proposed a hybrid discrete differential evolution (HDDE), where several crossover and mutation operators were employed. Meanwhile, a speedup method was also proposed to evaluate the insertion neighborhood, which largely reduced the complexity of HDDE. Pan and Wang [4] introduced the concept of weight value into PF and presented two simple constructive heuristics, i.e., profile fitting (wPF) and the Pan-Wang (PW) heuristic. Then, based on wPF and PW three improved constructive heuristics, i.e., PF-NEH, wPF-NEH and PW-NEH were proposed. Wang et al. [5] developed a three-phase algorithm (TPA) in which a priority rule based on the average value and standard deviation of the processing time, a variant of the NEH-insert procedure and a modified simulated annealing algorithm were utilized to three phases, respectively. Ding et al. [6] investigated some new blocking properties of the BFSP and proposed an iterated greedy algorithm (B-IG) based on these priorities. Han et al. [7] presented a modified fruit fly optimization (MFFO) algorithm in which three key operators, i.e., a problem-specific heuristic, neighborhood strategy and a speedup insert neighborhood based local search were employed. Tasgetiren et al. [8] proposed two iterated greedy algorithms, i.e., IG_{IJ} and IG_{RLS}, which combined a constructive heuristic and two well-known speedup methods for both insertion and swap neighborhood structures.

For BFSP with total flow time, Wang et al. [9] developed three harmony search algorithms, i.e., hybrid harmony search (hHS) algorithm, hybrid global best harmony search (hgHS) algorithm and hybrid modified global best harmony search (hmgHS) algorithm. Fernandez-Viagas et al. [10] conducted a comprehensive evaluation of the available heuristics for the BFSP and proposed an efficient constructive heuristic based on the beam search. For BFSP with total tardiness, Ronconi and Henriques [11] proposed a constructive heuristic (FPDNEH) and a GRASP-based heuristic. Nagano et al. [12] presented an evolutionary clustering search (ECS) algorithm combined with a variable neighborhood search (VNS), which NEH-based procedure to generate an initial solution, the genetic algorithm to generate solutions and VNS to improve the solutions. For the parallel blocking flow shop scheduling problem, Ribas et al. [13] proposed some constructive and improvement heuristics to address parallel or distributed blocking flow-shop problem recently, which included an iterated local search (ILS) and an iterated greedy algorithm (IGA), both of which are combined with a variable neighborhood search (VNS). From the previous literature reviews, it can be seen that researches on this concerned problem with SDSTs are considerably scarce. Therefore, it is meaningful to develop and propose more effective and efficient approaches for addressing variety of blocking flow-shop scheduling problems.

As a statistical learning based stochastic optimization, Estimation of Distribution Algorithm (EDA) [14] has attracted increasing attention in the field of evolutionary computation during recent years. Unlike GA, which apply mutation and crossover operators, EDA generates offspring through building and sampling explicit probabilistic models from superior candidate solutions. Owing to its outstanding global exploration and inherent parallelism, EDA has already been extensively applied to deal with production scheduling problems. Ceberio et al. [15] provided an overview of EDA in combinatorial optimization problems, which is of great importance of EDA for researchers. Pan and Ruiz [16] proposed a probabilistic model by taking into account of both the order of the jobs in the sequence and the promising similar blocks of jobs and employed the efficient NEH initialization and local search to enhance the performance of EDA. Recently, Wang et al. [17] presented an EDA with critical path based local search scheme to solve the flexible job-shop scheduling problem. Moreover, Wang et al. [18] developed an effective estimation of distribution algorithm (EEDA) to solve the distributed permutation flow-shop scheduling problem (DPFSP). From the mechanism of EDA, it can be seen that the tradeoff between the proper probability model with updating mechanism and local search are crucial to develop effective meta-heuristics, which should achieve a better balance between global exploration and local exploitation. Consequently, to the best of our knowledge, there is no published work on EDA for the BFSP with SDSTs. In this work, EDA is employed to guide the promising search direction and find superior solutions, and the local intensification methods are designed to emphasize the exploitation from those promising regions.

The remainder of this work are organized as follows. Section 2 briefly introduces the BFSP with SDSTs. Section 3 elaborates on the proposed HEDA, such as HEDA global exploration, and reference sequence based local exploitation, respectively. Computational comparisons are given and analyzed in Sect. 4. Finally, Sect. 5 gives some concluding remarks and suggestions of future research.

2 Problem Statement

The BFSP with SDSTs, which denoted as $Fm|blocking, ST_{sd}|C_{max}$, can be described as follows. There are n jobs and m machines. Each of n jobs will be continuously processed without intermediated buffers. Since there are no intermediate buffers between consecutive machines, a job has to be blocked on current machine until the next machine has prepared for processing it. The sequence-dependent setup times are separable from the processing times and dependent upon both current job being processed and the adjacent job on each machine. Note that other common flow-shop assumptions are considered. Let $\boldsymbol{\pi} = [\pi_1, \pi_2, \ldots, \pi_n]$ denote a permutation sequence. Let $p_{\pi_i,l}$ denotes the processing time of job π_i on machine l, $d_{\pi_i,l}$ denotes the departure time of job π_i on machine l, and $d_{\pi_i,0}$ denotes the time job π_i starts its processing on the first machine. $s_{\pi_{i-1},l\pi_i,l}$ represents the setup times between two consecutive jobs π_{i-1} and π_i on machine l. Let $p_{\pi_0,l} = 0$, $l = 1, \ldots, m$ for initial job and $s_{\pi_0,\pi_i,l}$ represents the initial setup times of job π_i. Then $d_{\pi_i,l}$ can be calculated as follows:

$$d_{\pi_0,l} = 0,\ l = 1, \ldots, m \tag{1}$$

$$d\pi_{i,0} = d_{\pi_{i-1},1} + s_{\pi_{i-1},\pi_i,1},\ i = 1, 2, \ldots, n \tag{2}$$

$$d\pi_{i,l} = \max\{d_{\pi_{i-1},\pi_i,l+1} + s_{\pi_{i-1},\pi_i,l+1}, d_{\pi_i,l-1} + d_{\pi_i,l}\},\ i = 1, 2, \ldots, n;\ l = 1, 2, \ldots, m-1 \tag{3}$$

$$d_{\pi_i,m} = d_{\pi_i,m-1} + d_{\pi_i,m},\ i = 1, 2, \ldots, n \tag{4}$$

Therefore, the makespan of the $\boldsymbol{\pi} = [\pi_1, \pi_2, \ldots, \pi_n]$ can be given by $C_{max}(\boldsymbol{\pi}) = d_{\pi_n,m}$. The aim of this paper is to find a schedule $\boldsymbol{\pi}^*$ in the set of all schedules Π such that

$$C_{max}(\boldsymbol{\pi}^*) = \min_{\boldsymbol{\pi} \in \Pi} C_{max}(\boldsymbol{\pi}) \tag{5}$$

3 HEDA for BFSP with SDSTs

In this section, the HEDA for $Fm|blocking, ST_{sd}|C_{max}$ is proposed after explaining the solution representation and population initialization, probabilistic model, updating mechanism and new population generation and local intensification, i.e. a reference sequence based local search and a path relinking based local search, respectively.

3.1 Solution Representation and Population Initialization

In this work, we use the job permutation based representation, that is, each individual presents a feasible solution. Because of the properties of BFSP, the longer total processing time of job is, the higher probability of blocking occurs [6]. Therefore, an effective heuristic [4], i.e. PF-NEH, is employed to initialize and intensify the quality of the partial initial population. In addition, other initial solutions are randomly generated to maintain the diversity of the population. Note that the speedup method [3] is also utilized to evaluate the generated sequences and reduce the computational efforts.

3.2 Probabilistic Model

Probabilistic model is utilized to investigate the distribution information of the superb solutions, which guides the promising evolutionary directions. Therefore, the model has a key influence on the performances of HEDA. Obviously, an appropriate model can greatly enhance the efficiency and effectiveness of the presented algorithms. In this study, the probabilistic model of HEDA is denoted by $\mathbf{P}(\mathbf{gen})$ as follows:

$$\mathbf{P}(\mathbf{gen}) = \begin{bmatrix} \mathbf{P}_1(\mathbf{gen}) \\ \vdots \\ \mathbf{P}_\mathbf{n}(\mathbf{gen}) \end{bmatrix}_{n\times 1} = \begin{bmatrix} P_{1,1}(gen) & \cdots & P_{1,n}(gen) \\ \vdots & \ddots & \vdots \\ P_{n,1}(gen) & \cdots & P_{n,n}(gen) \end{bmatrix}_{n\times n} \tag{6}$$

where $\mathbf{P_i}(\mathbf{gen}) = [P_{i,1}(gen), P_{i,2}(gen), \cdots, P_{i,n}(gen)]$ denote the ith vector of $\mathbf{P}(\mathbf{gen})$ in gen generation. Especially, $P_{i,j}(gen)$ is the probability of job j appearing in the ith position of $\boldsymbol{\pi}$ at gen. It's clear that the probability matrix is a random matrix, where each $P_{i,j}(gen)$ is a probability value for a certain event. In addition, for a certain position, the total probability of all jobs in this position of a solution sequence inevitable to 1, i.e., $\sum_{j=1}^{n} P_{i,j}(gen) = 1,\ \ gen \geq 0$. Note that the values in $\mathbf{P}(\mathbf{gen})$ reflect the priority of promising jobs in candidate sequences. Therefore, the elements of probability matrix affect the selection of the superior individuals. The higher value of $P_{i,j}(gen)$ $(i,j = 1,2,\cdots,n)$ is, the higher possibility for job j is selected in position i.

According to the above definitions, HEDA generates offspring by sampling from the probability matrix proposed in Eq. (6), which means each individual is generated based on $\mathbf{P}(\mathbf{gen})$ by roulette sampling. Therefore, the values in $\mathbf{P}(\mathbf{gen})$ determine the composition of the generated population. Note that we set $P_{i,j}(0) = 1/(n \times n)$ $(i,j = 1,2,\cdots,n)$ when $gen = 0$, which can guarantee the diversity of initial population, and accumulate more quality information appropriately at the initial phase.

3.3 Updating Mechanism

Probability model guides the searching direction and an effective update mechanism has a great influence on the searching performance. To make the $\mathbf{P}(\mathbf{gen})$ accurately estimate and effectively update the probability distribution of the superior population, and guide HEDA to promising search directions, two matrixes are proposed in this section, which record the total information of the order of jobs and the similar blocks, respectively. Therefore, the two matrixes $\boldsymbol{\eta}(\mathbf{gen})$ and $\boldsymbol{\xi}(\mathbf{gen})$ are defined as follows:

$$\boldsymbol{\eta}(\mathbf{gen}) = \begin{bmatrix} \boldsymbol{\eta}_1(\mathbf{gen}) \\ \vdots \\ \boldsymbol{\eta}_\mathbf{n}(\mathbf{gen}) \end{bmatrix}_{n\times 1} = \begin{bmatrix} \eta_{1,1}(gen) & \cdots & \eta_{1,n}(gen) \\ \vdots & \ddots & \vdots \\ \eta_{n,1}(gen) & \cdots & \eta_{n,n}(gen) \end{bmatrix}_{n\times n} \tag{7}$$

$$\boldsymbol{\xi}(\mathbf{gen}) = \begin{bmatrix} \boldsymbol{\xi}_1(\mathbf{gen}) \\ \vdots \\ \boldsymbol{\xi}_\mathbf{n}(\mathbf{gen}) \end{bmatrix}_{n\times 1} = \begin{bmatrix} \xi_{1,1}(gen) & \cdots & \xi_{1,n}(gen) \\ \vdots & \ddots & \vdots \\ \xi_{n,1}(gen) & \cdots & \xi_{n,n}(gen) \end{bmatrix}_{n\times n} \tag{8}$$

where each $\eta_{i,j}(gen)$ in $\boldsymbol{\eta}(\mathbf{gen})$ records the information of the number of times that job j appears before or in position i in the set of superior individuals. Additionally, $\boldsymbol{\eta}(\mathbf{gen})$ can efficiently reduce the rate of convergence and effectively prevent premature convergence. The ordinal matrix $\boldsymbol{\eta}(\mathbf{gen})$ can be calculated as follows:

$$\eta_{i,j}(gen) = \sum\nolimits_{s=1}^{Sbest} X_{i,j}^{s}(gen),\ i,j = 1,2,\ldots,n \tag{9}$$

where $X_{i,j}^{s}$ is a binary variable and *Sbest* represents the number of elite solution. In order to make the HEDA not trapped into local optimum and efficiently increase the diversity of population, the value of $X_{i,j}^{s}$ is set as 1 from i to n when job j appears in position i, otherwise 0. Then the indicative function is expressed as follows:

$$X_{i,j}^{s}(gen) = \begin{cases} 1, & \text{if job } j \text{ appears before or in position } i \\ 0, & \text{else} \end{cases} \tag{10}$$

Using the ordinal matrix can save the historical information of selected elite solutions. Note that the normalization should be performed to $\boldsymbol{\eta}(\mathbf{gen})$ after it is updated.

Inspired by the schema theorem and the hypothesis of building blocks, the dependency matrix, i.e., $\boldsymbol{\xi}(\mathbf{gen})$, is proposed to record the number of times that job j' appears immediately after job j when job j is in the previous position of job j' placed. Then each element $\xi_{j',j}(gen)$ in $\boldsymbol{\xi}(\mathbf{gen})$ can be calculated as follows:

$$\xi_{j',j}(gen) = \sum\nolimits_{s=1}^{Sbest} Y_{j',j}^{s}(gen),\ j',j = 1,2,\ldots,n \tag{11}$$

$$Y_{j',j}^{s}(gen) = \begin{cases} 1, & \text{if job } j \text{ appears immediately after job } i \\ 0, & \text{else} \end{cases} \tag{12}$$

It is note that the normalization is also required for the dependency matrix $\boldsymbol{\xi}(\mathbf{gen})$ after updating, which can slow down the convergence and adjust the searching space. Then, $\boldsymbol{\eta}(\mathbf{gen})$ and $\boldsymbol{\xi}(\mathbf{gen})$ which present the importance of the job order and the similar blocks can be obtained by Eqs. (7)–(12). Therefore, the probabilistic model can be updated by utilizing the information of $\boldsymbol{\eta}(\mathbf{gen})$ and $\boldsymbol{\xi}(\mathbf{gen})$, which can effectively balance the historical and current information of superior population. The incremental learning probabilistic model based on Heb-rule can be established as Eq. (13).

$$P_{i,j}(gen+1) = \begin{cases} rP_{i,j}(gen) + (1-r)\eta_{i,j}(gen+1),\ i = 1 \\ rP_{i,j}(gen) + \frac{(1-r)}{\mu}\left[\delta_1\eta_{i,j}(gen+1) + \delta_2\xi_{[i-1],j'}(gen+1)\right],\ i = 2,3,\ldots,n \end{cases} \tag{13}$$

Note that $[i-1]$ presents the selected job placed in position $i-1$ and r $(0 < r \leq 1)$ is learning rate. Especially, if $r = 0$, the $\mathbf{P}(\mathbf{gen})$ can update directly without inertia. Note that $\mu = \sum_{j=1}^{n} \left(\delta_1 \eta_{ij}(gen+1) + \delta_2 \xi_{i-1,j}(gen+1)\right)$ $(i = 2, 3, \ldots, n)$, where δ_1 and δ_2 are two user-defined parameters, which indicate the importance of the two matrix $\boldsymbol{\eta}(\mathbf{gen})$ and $\boldsymbol{\xi}(\mathbf{gen})$ and the diversity of the population in each iteration. It has obvious that the updating mechanism is more advantageous to enhance the promising genetic information, which efficiently improve the guidance of the global exploration.

3.4 New Population Generation and Local Intensification

Sampling to generate new population means that all jobs are selected dependent on the probabilistic model $\mathbf{P}(\mathbf{gen})$. The sampling procedure is described as follows:

Step 1: Set control parameter $p = 1$ and randomly generate a probability value r where $r \in \left[0, \sum_{l=1}^{n} P_{il}(gen)\right)$.
Step 2: Get a candidate job j_c by using roulette selection scheme.
  Step 2.1: If $r \in [0, P_{i1}(gen))$ $(i \in \{1, \ldots, n\})$, then set $j_c = 1$, and go to Step 3.
  Step 2.2: If $r \in \left[\sum_{l=1}^{h_c-1} P_{il}(gen), \sum_{l=1}^{h_c} P_{il}(gen)\right)$ and $(i \in \{1, \ldots, n\}$, $h_c = 2, \ldots, n)$, then select the candidate job $j_c = h_c$, and go to Step 3.
Step 3: If the candidate job j_c do not repeat with the selected jobs, then return j_c.
Step 4: Put the selected j_c into the corresponding position and execute the sampling method independently and repeatedly until generating a feasible solution.
Step 5: Set $p = p + 1$. If $p \leq popsize$, go to Step 2. Otherwise output $\mathbf{pop}(\mathbf{gen})$.

To enhance the exploitation, two local search methods based on the speedup methods, i.e., a reference sequence based local search (RLS) [9] and a path relinking based local search, are employed and embedded in the EDA for stressing exploitation.

4 Simulation Results and Comparisons

4.1 Experimental Setup

Some randomly generated instances are adopted to test the performance of HEDA. The $n \times m$ instances are generated with {20, 30, 50, 70, 100, 200, 500} $\times$ {5, 10, 20}. Each group consists of 10 instances. Therefore, we have a total of 210 different instances. The processing time $p_{j_i,l}$ and the setup time $s_{j_{i-1},j_i,l}$ are generated from two different uniform distributions [1, 100] and [0, 100], respectively. To evaluate the performance of each algorithm, the average relative percentage deviation (ARPD) is computed to measure the quality of the solutions. The ARPD is calculated as follows:

$$ARPD = \frac{1}{R} \sum_{i=1}^{R} \frac{C_{max}^{i} - C_{max}^{best}}{C_{max}^{best}} \times 100 \qquad (14)$$

where C_{max}^{i} is the solution obtained by a specific algorithm in ith experiment and C_{max}^{best} is the best solutions obtained by all of the compared algorithms and R is the number of repetitions. Obviously, the less the value of ARPD is, the better performance of the algorithm is. Note that a full factorial design of experiment is carried out to determine the best parameters for the compared algorithms. The parameters for HEDA are set as follows: $popsize = 100$, $r = 0.75$, $Sbest = 10$, $\delta_1 = 0.7$ and $\delta_2 = 0.3$. All algorithms have been re-implemented in Embarcadero Studio XE-10. Computational experiments independently run 30 times on a server with 2.60 GHz Intel® Xeon® E5-4620 v2 processors (32 cores) and 64 GB RAM running Windows 10 Operating System. Therefore, the results are impartial and completely comparable.

4.2 Computational Results and Comparisons

To evaluate the effectiveness of the proposed HEDA, we have re-implemented the existing state-of-the-art algorithms, i.e. HDDE [3], TPA [5], B-IG [6], MFFO [7], $\mathrm{IG_{RLS}}$ [8], hmgHS [9], ECS [12], PEDA [16], and EEDA [18], for comprehensive comparisons as all details given by the original papers. Since no algorithms were proposed for the problem under consideration until now, we have adapted them by using the makespan criterion presented in Sect. 2. The statistical results of *ARPD* with the same maximum elapsed time limit of $t = 30 \times n \times (m/2)$ milliseconds as a termination criterion are reported in Table 1, where *Average* is a statistical average and the best, the second-best, and the third-best values in each cell are represented by using the bold, the bold with underlined, and the italic with underlined fonts, respectively.

It is clear through Table 1 that HEDA performs steadily and well in terms of the overall ARPD values, and that it outperforms the others in different scale problems. However, $\mathrm{IG_{RLS}}$ is a potential competitive algorithm better than HDDE and hmgHS.

Table 1. Computational results of compared algorithms and HEDA.

Instance	HDDE	TPA	B-IG	MFFO	$\mathrm{IG_{RLS}}$	hmgHS	ECS	PEDA	EEDA	HEDA
20 × 5	0.016	0.137	0.018	*0.009*	**0.007**	0.064	0.562	0.078	0.027	**0.005**
20 × 10	*0.011*	0.034	0.012	*0.011*	**0.006**	0.036	0.457	0.034	0.012	**0.004**
20 × 20	**0.017**	0.047	0.025	*0.021*	**0.019**	0.083	0.239	0.076	0.023	**0.017**
30 × 5	**0.398**	0.784	*0.578*	0.512	*0.438*	0.815	1.283	0.645	0.536	**0.364**
30 × 10	**0.346**	0.732	0.507	0.493	*0.412*	0.726	1.184	0.535	0.482	**0.337**
30 × 20	**0.489**	0.836	0.612	0.587	*0.535*	0.923	1.326	0.624	0.546	**0.478**
50 × 5	1.924	1.536	*0.878*	1.382	**0.763**	1.185	2.651	1.436	1.147	**0.563**
50 × 10	1.967	1.464	*0.988*	1.402	**0.787**	1.231	2.649	1.312	1.347	**0.546**
50 × 20	1.783	1.215	*0.936*	1.117	**0.683**	1.189	2.276	1.183	1.382	**0.524**
100 × 5	2.163	1.687	*1.105*	1.782	**0.871**	1.624	2.853	1.382	1.515	**0.787**
100 × 10	2.231	1.602	*1.138*	1.927	**0.938**	1.633	2.589	1.236	1.674	**0.538**
100 × 20	2.562	1.275	*1.107*	1.776	**0.783**	1.687	2.611	1.312	1.647	**0.534**
200 × 10	2.216	1.131	**0.683**	1.132	1.028	1.732	1.521	0.979	*0.762*	**0.383**
200 × 20	1.874	1.137	**0.638**	1.213	1.126	1.734	1.527	0.837	*0.762*	**0.376**
500 × 10	1.457	1.038	**0.586**	1.178	*1.032*	1.637	1.426	0.787	*0.721*	**0.368**
500 × 20	0.957	0.794	**0.499**	0.793	*0.535*	0.937	0.871	0.668	0.639	**0.276**
Average	1.276	0.966	*0.644*	0.958	**0.623**	1.077	1.627	0.820	0.826	**0.381**

To be specific, for small-scaled instances (n = 20, 30), MFFO and HDDE obtain the best solutions, except for 20 × 5. For the medium-scaled instances (n = 50, 100), $\mathrm{IG_{RLS}}$ obtains the best performance. HEDA outperforms other algorithms at the considered margin for large-scaled instances (n = 200, 500). Meanwhile, some metaheuristics, i.e. HDDE, MFFO and hmgHS, are highly depend on the parameters for different scale instances. The superiority of HEDA owes to some aspects as follow: (1) The PF-NEH heuristic provides excellent initial solutions. (2) With a well-designed probability model and the suitable updating mechanism, it is helpful to explore the promising area in the solution space effectively. (3) With the path relinking based local search, it is helpful to further exploit these regions and enhance the exploitation capability. (4) With the suitable calibration of parameters, it is helpful to obtain satisfactory schedules. With the above merits, it is safely concluded that the HEDA is a new state-of-the-art algorithm for $Fm|blocking, ST_{sd}C_{max}$ with excellent quality and robustness.

5 Conclusions and Future Research

To the best of the current authors' knowledge, this is the first report on the application of the HEDA for blocking flow-shop scheduling problem (BFSP) with sequence-dependent setup times (SDSTs) to minimize the maximum completion time. The speedup method is designed to reduce the computing complexity successfully. Both the effectiveness of searching solutions and the efficiency of evaluating solutions were stressed, and the influence of parameter setting was investigated as well. Due to reasonably hybridize and balance the global exploration and local exploitation, HEDA's search behavior can be enriched and its search performance can be greatly enhanced, which outperforms other considered algorithms with the effectiveness and robustness for the BFSP with SDSTs. Our future work is to develop some effective metaheuristics to solve the flexible distributed energy efficient scheduling problems.

Acknowledgments. The authors are sincerely grateful to the anonymous referees. This research is partially supported by the National Science Foundation of China (51665025), and the Applied Basic Research Foundation of Yunnan Province (2015FB136).

References

1. Allahverdi, A.: The third comprehensive survey on scheduling problems with setup times/costs. Eur. J. Oper. Res. **246**, 345–378 (2015)
2. Hall, N.G., Sriskandarajah, C.: A survey of machine scheduling problems with blocking and no-wait in process. Oper. Res. **44**(3), 510–525 (1996)
3. Wang, L., Pan, Q.K., Suganthan, P.N., Wang, W.H., Wang, Y.M.: A novel hybrid discrete differential evolution algorithm for blocking flow shop scheduling problems. Comput. Oper. Res. **37**(3), 509–520 (2010)
4. Pan, Q.K., Wang, L.: Effective heuristics for the blocking flowshop scheduling problem with makespan minimization. Omega **40**(2), 218–229 (2012)

5. Wang, C., Song, S., Gupta, J.N.D., Wu, C.: A three-phase algorithm for flowshop scheduling with blocking to minimize makespan. Comput. Oper. Res. **39**(11), 2880–2887 (2012)
6. Ding, J.Y., Song, S., Gupta, J.N.D., Wang, C., Zhang, R., Wu, C.: New block properties for flowshop scheduling with blocking and their application in an iterated greedy algorithm. Int. J. Prod. Res. **54**(16), 1–14 (2016)
7. Han, Y., Gong, D., Li, J., Zhang, Y.: Solving the blocking flow shop scheduling problem with makespan using a modified fruit fly optimisation algorithm. Int. J. Prod. Res. **54**(22), 6782–6797 (2016)
8. Tasgetiren, M.F., Kizilay, D., Pan, Q.K., Suganthan, P.N.: Iterated greedy algorithms for the blocking flowshop scheduling problem with makespan criterion. Comput. Oper. Res. **77**(C), 111–126 (2017)
9. Wang, L., Pan, Q.K., Tasgetiren, M.F.: Minimizing the total flow time in a flow shop with blocking by using hybrid harmony search algorithms. Expert Syst. Appl. **37**(12), 7929–7936 (2010)
10. Fernandez-Viagas, V., Leisten, R., Framinan, J.M.: A computational evaluation of constructive and improvement heuristics for the blocking flow shop to minimise total flowtime. Expert Syst. Appl. **61**, 290–301 (2016)
11. Ronconi, D.P., Henriques, L.R.S.: Some heuristic algorithms for total tardiness minimization in a flowshop with blocking ☆. Omega **37**(2), 272–281 (2009)
12. Nagano, M.S., Komesu, A.S., Mi, H.H.: An evolutionary clustering search for the total tardiness blocking flow shop problem. J. Intell. Manuf. 1–15 (2017)
13. Ribas, I., Companys, R., Tort-Martorell, X.: Efficient heuristics for the parallel blocking flow shop scheduling problem. Expert Syst. Appl. **74**, 41–54 (2017)
14. Larranaga, P., Lozano, J.A.: Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation. Springer, Netherlands (2002). https://doi.org/10.1007/978-1-4615-1539-5
15. Ceberio, J., Irurozki, E., Mendiburu, A., Lozano, J.A.: A review on estimation of distribution algorithms in permutation-based combinatorial optimization problems. Prog. Artif. Intell. **1** (1), 103–117 (2012)
16. Pan, Q.K., Ruiz, R.: An estimation of distribution algorithm for lot-streaming flow shop problems with setup times. Omega Int. J. Manag. Sci. **40**(2), 166–180 (2012)
17. Wang, L., Wang, S.Y., Xu, Y., Zhou, G., Liu, M.: A bi-population based estimation of distribution algorithm for the flexible job-shop scheduling problem. Comput. Ind. Eng. **62**, 917–926 (2012)
18. Wang, S.Y., Wang, L., Liu, M., Xu, Y.: An effective estimation of distribution algorithm for solving the distributed permutation flow-shop scheduling problem. Int. J. Prod. Econ. **145**, 387–396 (2013)