Knowledge-Based Systems 74 (2015) 167–175

Contents lists available at ScienceDirect

Knowledge-Based Systems

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

A bi-population EDA for solving the no-idle permutation flow-shop scheduling problem with the total tardiness criterion

Jing-nan Shen, Ling Wang *, Sheng-yao Wang

Tsinghua National Laboratory for Information Science and Technology (TNList), Department of Automation, Tsinghua University, Beijing 100084, China

ARTICLE INFO

Article history:
Received 20 December 2013
Received in revised form 10 September 2014
Accepted 13 November 2014
Available online 20 November 2014

Keywords:
No-idle permutation flow-shop scheduling problem
Total tardiness
Estimation of distribution algorithm
Bi-population
Probability model

ABSTRACT

In this paper, an effective bi-population estimation of distribution algorithm (BEDA) is presented to solve the no-idle permutation flow-shop scheduling problem (NIPFSP) with the total tardiness criterion. To enhance the search efficiency and maintain the diversity of the whole population, two sub-populations are used in the BEDA. The two sub-populations are generated by sampling the probability models that are updated differently for the global exploration and the local exploitation, respectively. Meanwhile, the two sub-populations collaborate with each other to share search information for adjusting the models. To well adjust the models for generating promising solutions, the global probability model is updated during the evolution with the superior population and the local probability model is updated with the best solution that has been explored. To further enhance exploitation in the promising region, the insertion operator is used iteratively as the local search procedure. To investigate the influence of parameter setting, numerical study based on the Taguchi method of design-of-experiment is carried out. The effectiveness of the bi-population strategy and local search procedure is shown by numerical comparisons, and the comparisons with the recently published algorithms by using the benchmarking instances also demonstrate the effectiveness of the proposed BEDA.

1. Introduction

As a well-known combinational optimization problem, the permutation flow-shop scheduling problem (PFSP) [1] has been studied by many researchers [2–7] due to its academic and practical significance. In some real production environments, *e.g.* foundry production [8] and fiber glass processing [9], the idle running of machines is not allowed for certain economic or technical reasons. Thus, the no-idle version of the PFSP (NIPFSP) comes into being, which has been proved to be NP-hard [10]. Note that the no-idle condition is different from the no-wait case. For the no-idle case, each machine has to process jobs without any interruption from the start of processing the first job to the completion of the last job. For the no-wait case, the processing of all operations of each job should be done without any interruption either on or between any two consecutive machines. Compared with the traditional PFSP, research work about the NIPFSP is relatively limited. Relevant results in this area include the following.

Adiri and Pohoryles [11] first developed a polynomial time algorithm for the two-machine NIPFSP with the total completion time criterion. Vachajitpan [12] proposed a linear programming model as well as a branch and bound algorithm for the NIPFSP with the makespan criterion. Kamburowski [13] provided a better insight into the NIPFSP by identifying a simple network representation of the makespan criterion. In [14], a heuristic was presented by modeling the NIPFSP as a traveling salesman problem (TSP). Kalczynski and Kamburowski [9] developed a constructive heuristic, which was more effective than the TSP-based approach [14] and the Nawaz–Enscore–Ham (NEH) heuristic [15]. In [16], Baraz and Mosheiov introduced an efficient $O(n^2)$ greedy algorithm consisting of a sequence construction and an improvement procedure. As for meta-heuristics, a discrete differential evolution (DDE) algorithm [17] and a discrete particle swarm optimization (DPSO) algorithm [18] were proposed for the NIPFSP with the makespan criterion, where a speed-up scheme for the insertion neighborhood was presented to reduce the computational complexity from $O(mn^3)$ to $O(mn^2)$. Later, Deng and Gu [19] proposed a hybrid DDE algorithm and Tasgetiren et al. [20] presented a variable iterated greedy algorithm with differential evolution for the NIPFSP with makespan criterion. Taking the due dates of jobs into account, it is of significance to real life situations by considering the minimization of the total tardiness [21,22] so as to satisfy the external due dates committed to the customers. Compared to the research work on makespan criterion, the related work about the total tardiness criterion is rather rare. In [21], some versions of differential evolution (DE) algorithms

* Corresponding author. Tel.: +86 10 62783125; fax: +86 10 62786911.
E-mail address: wangling@mail.tsinghua.edu.cn (L. Wang).

http://dx.doi.org/10.1016/j.knosys.2014.11.016
0950-7051/

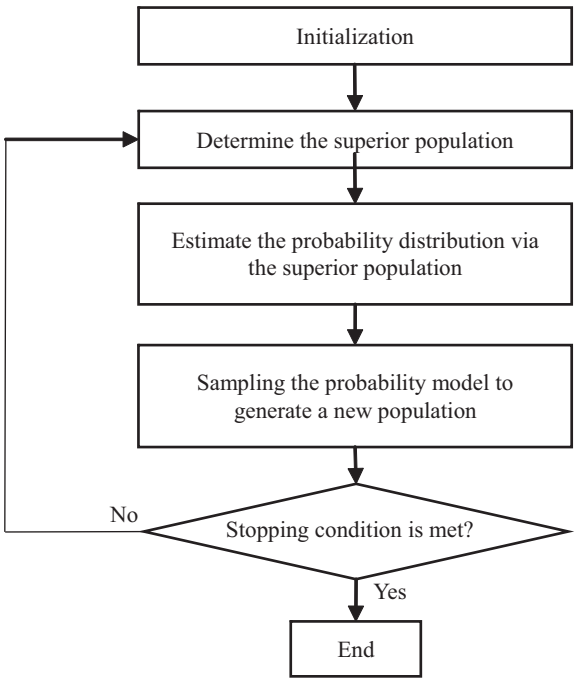

Fig. 1. The general framework of the EDA.

were developed; Very recently, a discrete artificial bee colony (DABC) algorithm [22] were developed. Clearly, it is very important to develop new effective and efficient solution approaches for solving the NIPFSP with total tardiness criterion.

As a population-based evolutionary algorithm, estimation of distribution algorithm (EDA) [23] has gained increasing attention during recent years. Instead of using crossover and mutation operators of genetic algorithm (GA), the EDA produces offspring implicitly by using a probability model according to the knowledge of the obtained solution space during the search process. The EDA is of good ability in global exploration and has been applied to solve some types of scheduling problems, including flow-shop scheduling [24], job-shop scheduling [25], nurse scheduling [26], and resource-constrained project scheduling [27]. However, to the best of our knowledge, there is no reported work about the EDA for solving the NIPFSP. Inspired by the success of the EDA in solving scheduling problems, we will develop an effective bi-population EDA (BEDA) in this paper to solve the NIPFSP with the total tardiness criterion. To solve the problem effectively, two sub-populations are generated by sampling different probability models, i.e., the global model updated with the superior population, and the local probability model updated with the best solution that has been explored. In such a way, the search efficiency can be enhanced and the diversity of the whole population can be maintained to some extent. In addition to the sampling process based on the probability models, the insertion operator is adopted to further improve the local exploitation ability. To reflect the influence of parameter setting, we carry out the study by using the Taguchi method of design-of-experiment (DOE) [28]. The effectiveness of the bi-population strategy and local search procedure is experimentally demonstrated, and the comparisons between the BEDA and the very recent DABC and GA [22] are also presented to show the effectiveness of the BEDA.

The remainder of the paper is organized as follows: In Section 2, the NIPFSP is described. In Section 3, the BEDA for the NIPFSP is presented in details. In Section 4, the influence of parameter setting is investigated, and the numerical results and comparisons are provided. Finally, we end the paper with some conclusions and future work in Section 5.

2. Problem formulation

2.1. Notation

n	the number of jobs to be processed
m	the number of machines
J	$\{J_1, J_2, \ldots, J_n\}$: the job set to be processed
M	$\{M_1, M_2, \ldots, M_m\}$: the machine set
$\{O_{j,1}, O_{j,2}, \ldots, O_{j,m}\}$	the operation sequence of job J_j
$t_{j,i}$	the processing time of job J_j on machine M_i
d_j	due date of job J_j
$\pi = [\pi(1), \pi(2), \ldots, \pi(n)]$	a sequence of all the job numbers
C_j	the completion time of job J_j
TTD_j	tardiness of job J_j
TTD	total tardiness of a schedule

2.2. The problem statement

The NIPFSP is described as follows: a job set $J = \{J_1, J_2, \ldots, J_n\}$ is processed on m machines $M = \{M_1, M_2, \ldots, M_m\}$. Each job consists of m operations and the ith operation can only be performed on machine M_i. Each operation should be completed without interruption once it is started. Suppose all the jobs are available at time zero. Jobs are processed on each machine with the same order. Each machine can process at most one job at a time and each job can be processed on only one machine at a time. No idle time between any operations on the machines is permitted.

For the NIPFSP with the total tardiness criterion, it needs to determine the optimal processing order of all the jobs at each machine to minimize the total tardiness. For a schedule $\pi = [\pi(1), \pi(2), \ldots, \pi(n)]$, let $\sigma_k = [\pi(1), \pi(2), \ldots, \pi(k)]$ denote a partial sequence of π, the total tardiness TTD can be calculated as follows [22]:

$$Dif(\sigma_1, i, i+1) = t_{\pi(1),i} \quad i = 1, \ldots, m-1 \tag{1}$$

$$Dif(\sigma_k, i, i+1) = \max\{Dif(\sigma_{k-1}, i, i+1) - t_{\pi(k),i}, 0\} + t_{\pi(k),i+1} \quad k = 2, \ldots, n; i = 1, \ldots, m-1 \tag{2}$$

$$C_{\pi(n)} = \sum_{i=1}^{m-1} Dif(\sigma_n, i, i+1) + \sum_{k=1}^{n} t_{\pi(k),1} \tag{3}$$

$$C_{\pi(k)} = C_{\pi(k+1)} - t_{\pi(k),m} \quad k = 1, \ldots, n-1 \tag{4}$$

$$TTD_{\pi(k)} = \max\{C_{\pi(k)} - d_{\pi(k)}, 0\} \quad k = 1, \ldots, n \tag{5}$$

$$TTD = \sum_{k=1}^{n} TTD_{\pi(k)} \tag{6}$$

where $Dif(\sigma_k, i, i+1)$ is the minimum difference between the completion time of the last job of σ_k on machine M_i and M_{i+1}.

3. BEDA for NIPFSP

In this section, a bi-population EDA (BEDA) will be presented to solve the NIPFSP with the total tardiness criterion. First, the basic EDA will be introduced. Then, the BEDA will be introduced in details, including initialization, probability model and updating mechanism, and local search. Finally, the flowchart of BEDA will be illustrated.

3.1. Basic EDA

Different from other well-known population-based meta-heuristics (*e.g.* GA), the EDA produces offspring in an implicit way by sampling the probability model, instead of using the crossover

```
Procedure LS(π)
π_R = A random sequence;
k = 1, i = 1;
While ( i ≤ n )
   k = k mod n;
   π_1 =Remove job π_R(k) from π;
   π_2 =Best permutation obtained by inserting job π_R(k) in
   all possible positions of π_1;
   If ( f(π_2) < f(π) )
      Let π = π_2 and i = 1;
   Else
      Let k = k + 1;
   Endif
Endwhile
Endprocedure
```

Fig. 2. The procedure of local search.

and mutation operators. At the beginning of each generation, the superior sub-population is determined and the obtained best solution is updated. Then, the probability model is updated based on the superior sub-population and the offspring is generated by sampling the probability model. The EDA repeats the above process until a stopping condition is met.

The general framework of the EDA is illustrated in Fig. 1 [25,29].

From the framework, it can be seen that the probability model plays an important role in the EDA. Therefore, it should be well designed according to the characteristics of the considered problem. However, the EDA is often good at exploration but weak at exploitation relatively. An effective EDA should well balance the exploration and exploitation.

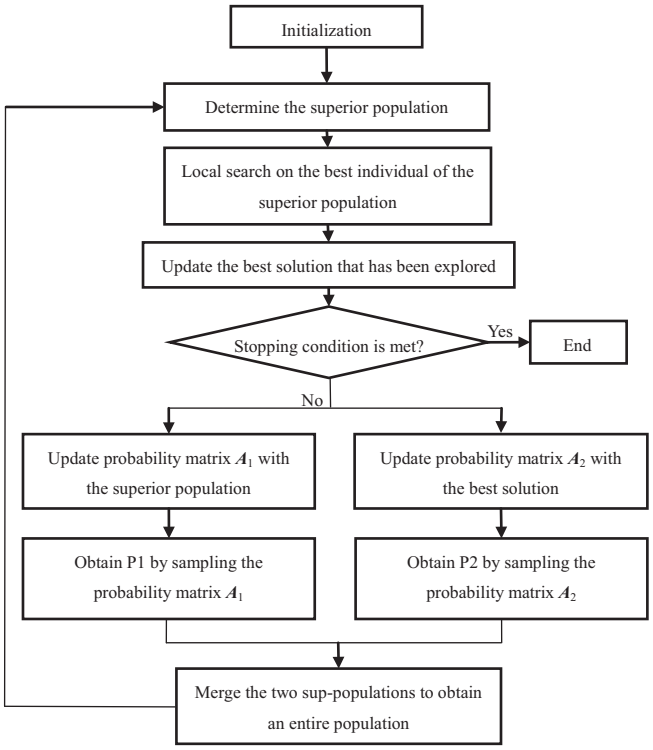

Fig. 3. The flowchart of the BEDA.

3.2. Initialization

In the initialization phase, a total of *P_Size* individuals will be generated. Each individual denotes a solution of the NIPFSP, which is represented by a sequence of all the job numbers, i.e., $\pi = [\pi(1), \pi(2), \ldots, \pi(n)]$. For example, a solution $\pi = [3,2,1,4]$ represents that job 3 is scheduled first, and then job 2 and job 1, and finally job 4 is the last one to be scheduled. To guarantee an initial population with certain quality and diversity, the population is initialized as follows: two solutions are constructed by the NEH heuristic [8] and the NEH_{EDD} heuristic [30] respectively, and the rest solutions are generated randomly.

To calculate the total tardiness as the schedule objective value, the speed-up method for insertion neighborhood [21,22] is employed. The idea of this method comes from the speed-up scheme proposed for the PFSP with makespan criterion. By means of recording some useful information about the partial sequences in advance, the method can reduce the computational complexity in evaluating the whole insertion neighborhood of a permutation from $O(mn^3)$ to $O(mn^2)$. For more details about the speed-up method, please refer to the literature [22].

3.3. Probability model and updating mechanism

Usually, all individuals in the EDA are generated by sampling a single probability matrix. Differently, the BEDA contains two sub-populations (denoted as *P*1 and *P*2), which are generated by sampling a global probability matrix $\boldsymbol{A}_1$ and a local probability matrix $\boldsymbol{A}_2$, respectively. The element $p_{ij}^k(l)$ of probability matrix $\boldsymbol{A}_k$ ($k = 1,2$) represents the probability that job J_j appears before or in position i of the sequence at the l-th generation. To ensure that the whole solution space can be sampled uniformly, p_{ij}^k is initialized as $p_{ij}^k = 1/n$ for all i and j. To generate individuals according to probability matrix $\boldsymbol{A}_k$, the following steps are implemented.

- Step 1. Set $i = 1$.
- Step 2. Select a job for position i. Job J_j is selected with probability p_{ij}^k.
- Step 3. If job J_j is selected, set the jth column of matrix $\boldsymbol{A}_k$ to 0 and normalize the rows to ensure that the sum of each row is 1.
- Step 4. Set $i = i + 1$ and go to step 2 until all the jobs have been selected.

With the above steps, all individuals of the two sub-populations can be generated. The population sizes of *P*1 and *P*2 (denoted as *P*1_*Size* and *P*2_*Size*, respectively) are set as Eqs. (7) and (8), respectively.

$$P1\_Size = (1 - \gamma\%) \times P\_Size \tag{7}$$

$$P2\_Size = \gamma\% \times P\_Size \tag{8}$$

Table 1
Combinations of parameter values.

Parameters	Factor level			
	1	2	3	4
P_Size	0.5*n*	*n*	1.5*n*	2*n*
η	10	20	30	40
γ	5	10	15	20
$\alpha(0)$	0.1	0.3	0.5	0.7
$\beta(0)$	0.1	0.3	0.5	0.7

where $\gamma \in (0, 100)$ is a parameter to control the rate of the two sub-populations. Obviously, when $\gamma = 0$ or 100, the whole population is generated by sampling only one matrix and the BEDA is converted to a single-population based EDA. To share information, the two sub-populations collaborate with each other. To be specific, sub-populations P1 and P2 will be merged as a whole one with *P_Size* individuals. Then, the best *SP_Size* individuals are selected to construct a superior population for adjusting the probability matrixes, where $SP\_Size = \eta\% \times P\_Size$.

In the BEDA, probability matrixes $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$ are updated in different ways as Eqs. (9) and (10), respectively. In specific, the global probability matrix $\boldsymbol{A}_1$ is updated according to the superior population (stress global exploration), while the local probability matrix $\boldsymbol{A}_2$ is updated according to the best solution that has been explored (stress local exploitation). Thus, γ should be set carefully to balance the exploration and exploitation abilities.

$$p_{ij}^1(l+1) = (1-\alpha(l))p_{ij}^1(l) + \frac{\alpha(l)}{i \times SP\_Size}\sum_{k=1}^{SP\_Size} I_{ij}^k \tag{9}$$

$$p_{ij}^2(l+1) = (1-\beta(l))p_{ij}^1(l) + \frac{\beta(l)}{i} I_{ij}^{best} \tag{10}$$

where I_{ij}^k is the indicator function of the kth individual in the superior population and I_{ij}^{best} is the indicator function of the best individual:

$$I_{ij}^k = \begin{cases} 1 & \textit{if job } j \textit{ appears before or in position } i \\ 0 & \textit{else} \end{cases} \tag{11}$$

$$I_{ij}^{best} = \begin{cases} 1 & \textit{if job } j \textit{ appears before or in position } i \\ 0 & \textit{else} \end{cases} \tag{12}$$

In addition, $\alpha(l)$ and $\beta(l)$ are the learning rates of the probability matrixes $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$ at the lth generation, respectively. To speed up convergence, the adaptive learning rates are adopted as follows:

$$\alpha(l) = \max\{\alpha(0) \times \exp(-0.01 \times l), 0.01\} \tag{13}$$

$$\beta(l) = \max\{\beta(0) \times \exp(-0.01 \times l), 0.01\} \tag{14}$$

where $\alpha(0)$ and $\beta(0)$ are the initial learning rates. In such a way, learning rates may be relatively large at the earlier period of the search process to accelerate the speed of convergence. As the population evolves, the learning rates decrease exponentially so as to enhance the exploitation ability. To ensure certain learning efficiency, it sets the learning rates no less than 0.01 as the above equations.

Table 2
Orthogonal array and ARV for instance Ta076.

Experiment number	Factor					ARV
	P_Size	η	γ	$\alpha(0)$	$\beta(0)$	
1	1	1	1	1	1	291425.4
2	1	2	2	2	2	283892.1
3	1	3	3	3	3	283453.8
4	1	4	4	4	4	285084.6
5	2	1	2	3	4	284041.6
6	2	2	1	4	3	281891.5
7	2	3	4	1	2	284156.3
8	2	4	3	2	1	290967.4
9	3	1	3	4	2	284293.8
10	3	2	4	3	1	287881.0
11	3	3	1	2	4	284002.5
12	3	4	2	1	3	284901.2
13	4	1	4	2	3	285128.4
14	4	2	3	1	4	284486.5
15	4	3	2	4	1	288139.9
16	4	4	1	3	2	283575.1

Table 3
Response value and significance rank for instance Ta076.

Level	*P_Size*	η	γ	$\alpha(0)$	$\beta(0)$
1	285,964	286,222	285,224	286,242	289,603
2	285,264	284,538	285,244	285,998	283,979
3	285,270	284,938	285,800	284,738	283,844
4	285,332	286,132	285,563	284,852	284,404
Delta	700	1685	577	1504	5760
Rank	4	2	5	3	1

3.4. Local search

To further enhance the exploitation ability, the insertion operator [22] is used iteratively for local search. Let π denote the current solution. It tries to insert each job of π into all possible positions to obtain a new schedule with the smallest total tardiness. All the jobs of π are handled in a random generated sequence, denoted as π_R. The procedure of the local search procedure is shown in Fig. 2.

Note that, the above local search is performed only on the best solution of the superior population, which is also the best one of the whole population. By iteratively performing insertion, the neighborhood of the best solution can be deeply exploited. Meanwhile, by using the speed-up method for insertion neighborhood [22], it may not cost too much time in local search. After the local search, the best solution that has been explored should be updated if a better one is found by the local search procedure.

3.5. Flowchart of BEDA

With the above design, the flowchart of the BEDA is illustrated in Fig. 3.

From the flowchart, it can be seen that two sub-populations work together in the algorithm. Since the sub-population *P*1 is generated by sampling the global probability matrix $\boldsymbol{A}_1$, which is updated according to the superior population, global exploration is stressed. Since the sub-population *P*2 is generated by sampling the local probability matrix $\boldsymbol{A}_2$, which is updated according to the best solution that has been explored, local exploitation is stressed. Merging the two sub-populations, it can share their information to determine the superior population of the whole population, which is helpful to adjust the two probability matrixes more reasonably. Via the local search on the best solution, deep local exploitation can be further enhanced. By balancing the global exploration and the local exploitation, the BEDA is expected to solve the NIPFSP effectively.

4. Numerical testing results and comparisons

To investigate the performance of the BEDA, numerical tests are carried out with the benchmarking set by Taillard [31]. The benchmarking set consists of 120 instances, ranging from 20 jobs/5 machines to 500 jobs/20 machines. The total work (TWK) rule [32] is used to determine the due dates. To be specific, the due date of job J_j is calculated by $d_j = \tau \times \sum_{i=1}^{m} p_{ij}$, where τ is the tightness factor and $\sum_{i=1}^{m} p_{ij}$ is the total processing time of job J_j on all machines. Same as the literature [22], τ is set as 1, 2 and 3 to make the job due date loose, medium and tight, respectively.

The proposed BEDA is coded in C++ language and run on a 3.30 GHz processor with 8G RAM in Visual Studio 2012. Same as the literature [22], the following relative percent deviation (Δ) from the *NEH* solution is used to evaluate the performance of an algorithm.

$$\Delta = \frac{ALG - NEH}{NEH} \times 100 \tag{15}$$

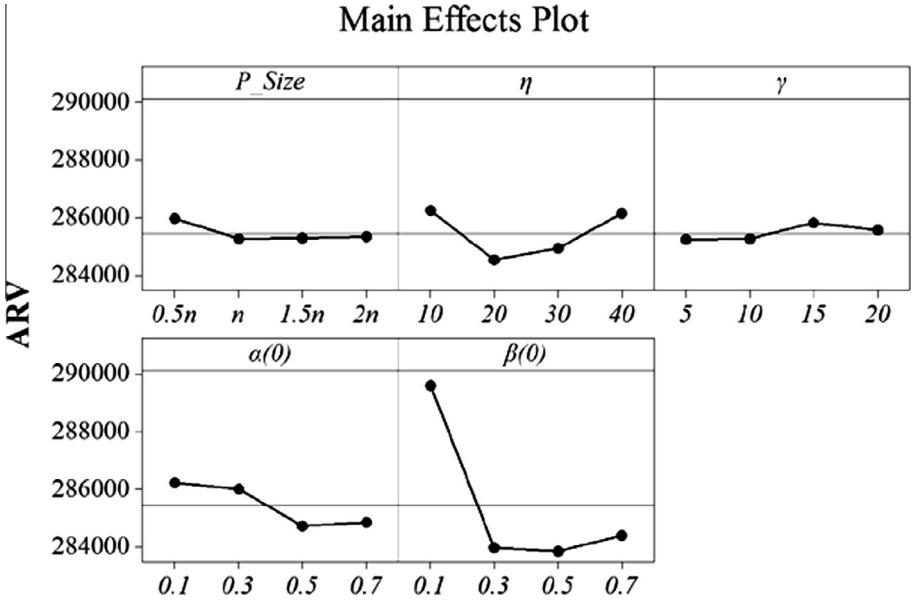

Fig. 4. Factor level trend.

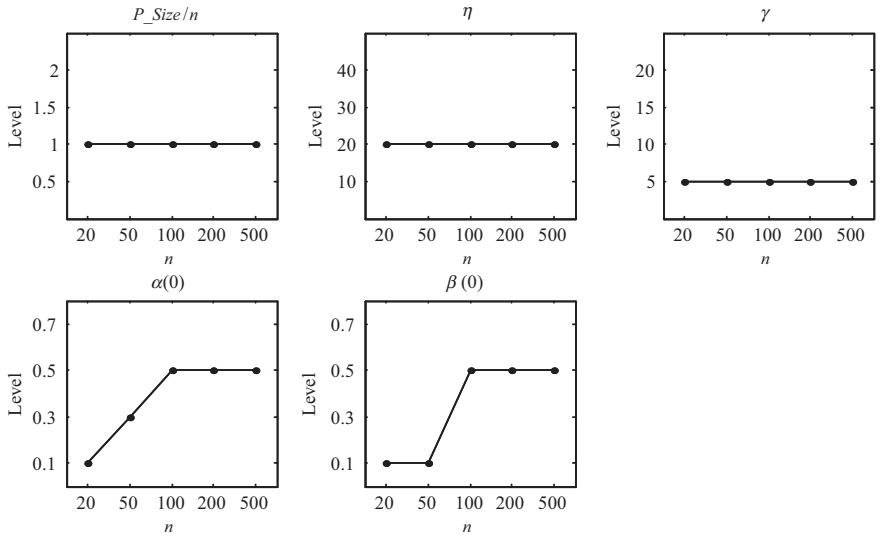

Fig. 5. Suggested parameter values for problems with different number of jobs (n).

where *ALG* denotes the objective value of the solution obtained by an algorithm, and *NEH* denotes the objective value of the *NEH* solution modified for the NIPFSP [22]. If Δ is less than 0, a better solution is obtained than *NEH*. The smaller the value is, the better the solution is. As for the stopping criterion, the maximum computational time is set as $T_{max} = 100 \times n$ milliseconds, which is only 1/1000 of that used in literature [22].

4.1. Parameters setting

The BEDA contains five key parameters: the population size (*P_Size*), the proportion of selected individuals to update the probability model $\boldsymbol{A}_1$ (η), the proportion of sub-population $P2$ (γ) and the initial learning rates $\alpha(0)$ and $\beta(0)$. To study the influence of these parameters on the performance of the BEDA, we implement the Taguchi method of design-of-experiment (DOE) [28] by using a set of instances with different problem scales. Clearly, the size of search space of the NIPFSP directly depends on the number of jobs. Thus, for each number of jobs n ($n = 20,50,100,200,500$), we randomly choose an instance with the medium due date case. That is, a total of 5 instances with different scales will be tested using the DOE to investigate the influence of the parameters. Combinations of the tested values of those parameters are listed in Table 1.

For each combination of the parameters, the BEDA is run 10 times independently with the stopping criterion mentioned above. The average value of the obtained total tardiness is regarded as the average response value (ARV). Due to the limitation of the pages, we only provide the DOE results with a moderate-scale instance Ta076 ($n = 100$, $m = 10$) as an example. The resulted ARV values are listed in Table 2 and the significance rank of each parameter is listed in Table 3. According to Table 3, the trend of each factor level is illustrated in Fig. 4.

From Table 3, it can be seen that $\beta(0)$ is the most significant parameter for the moderate-scale instance, η ranks the second place, $\alpha(0)$ ranks the third place, *P_Size* ranks the fourth place, and γ ranks the last place. From Fig. 4, the influence of each parameter

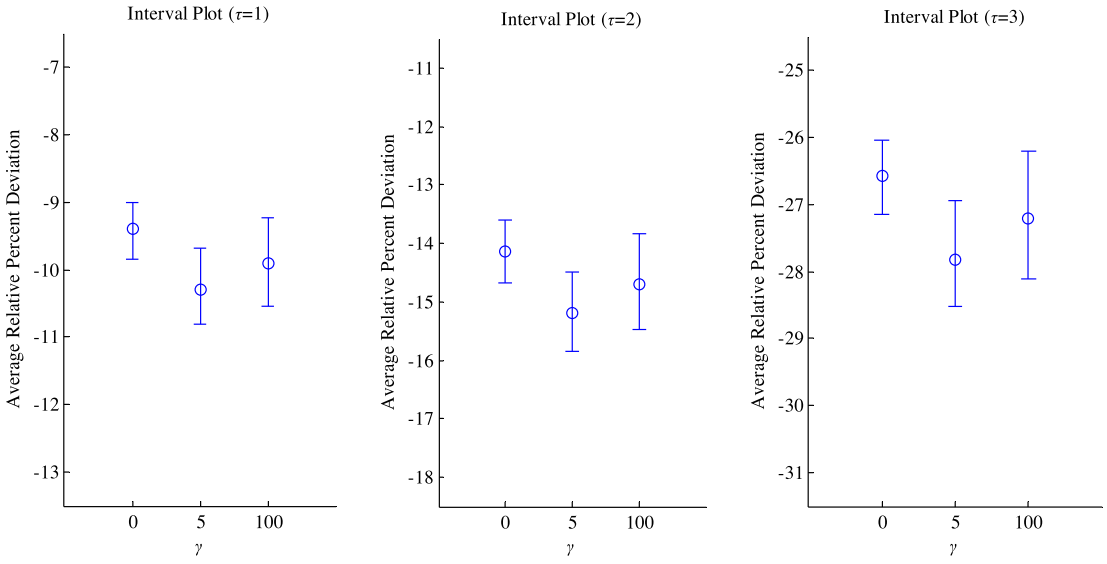

Fig. 6. Interval plot of average relative percent deviation ($\gamma = \{0, 5, 100\}$).

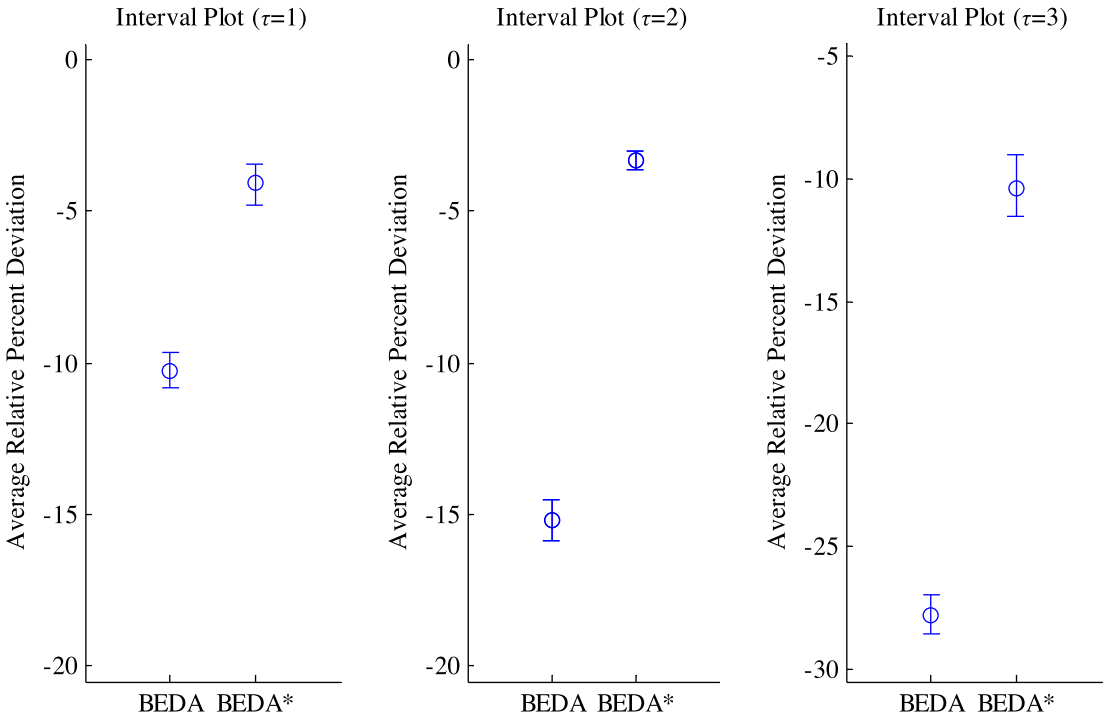

Fig. 7. Interval plot of average relative percent deviation (BEDA, BEDA*).

is clearly figured out. For the learning rate, usually a small value may lead to slow convergence, while a large value may result in premature convergence. Especially, in the BEDA the learning rate $\beta(0)$ should be determined carefully, since it is related to the best solution that has been explored. A too large rate may cause the local model over-dominated by the best solution, which will produce a sub-population without diversity. Since η is the proportion of population to determine the superior population, a small value will lead to few elite solutions involved in updating the model while a large value will cause the use of some bad solutions to update the model. As for the population size *P_Size*, with a fixed computational effort, a large value may lead to insufficient generations of search, while a small value may cause the sampling insufficient at each generation. As for γ, a larger value will slightly deteriorate the performance because it may lose the diversity of population by over using the local model.

Summarizing the results of all the five instances, we present the suggested values in Fig. 5 for solving the problems with different numbers of jobs. It can be seen that the suggested population size *P_Size* is linear to job number; and the best choices of η and γ are consistently suggested as 20 and 5, respectively. In addition, the suggested values of $\alpha(0)$ and $\beta(0)$ increase as n increases for the small-scale or moderate-scale problems, but they are both suggested as 0.5 for the large-scale problems.

To sum up, we suggest the parameter setting according to the above discussion for the problems with different job numbers as

Table 4
Computational results ($\tau = 1$).

$n \times m$	GA				DABC				BEDA			
	Avg	Best	Worst	Std	Avg	Best	Worst	Std	Avg	Best	Worst	Std
20 × 5	−10.95	−11.52	−9.82	0.72	−11.29	−11.58	**−10.81**	0.36	**−11.30**	**−11.59**	**−10.81**	0.34
20 × 10	−12.51	−13.18	−11.60	0.66	−12.99	−13.26	−12.50	0.33	**−13.09**	**−13.30**	**−12.53**	0.34
20 × 20	−13.50	−13.95	−12.83	0.49	−13.93	−14.06	−13.71	0.17	**−13.95**	**−14.14**	**−13.77**	0.18
50 × 5	−10.06	−10.55	−9.39	0.46	−10.65	−11.09	−10.18	0.37	**−10.86**	**−11.34**	**−10.22**	0.45
50 × 10	−12.85	−13.68	−11.97	0.70	−13.54	−14.2	−12.97	0.51	**−13.90**	**−14.63**	**−12.99**	0.67
50 × 20	−12.76	−13.66	−11.79	0.77	−13.63	−14.22	−12.82	0.57	**−13.88**	**−14.79**	**−12.87**	0.77
100 × 5	−6.40	−7.15	−5.72	0.57	−6.80	−7.33	−6.22	0.44	**−7.05**	**−7.60**	**−6.53**	0.42
100 × 10	−8.12	−8.76	−7.40	0.55	−8.80	−9.38	−8.26	0.46	**−9.14**	**−9.65**	**−8.42**	0.51
100 × 20	−9.60	−10.43	−8.81	0.66	−10.33	−11.08	−9.67	0.56	**−11.23**	**−11.76**	**−10.58**	0.48
200 × 10	−5.53	−6.39	−4.80	0.63	−5.37	−5.72	−4.92	0.34	**−5.68**	**−6.28**	**−5.07**	0.47
200 × 20	−7.94	−8.66	−7.32	0.54	−8.14	−8.78	−7.63	0.47	**−8.95**	**−9.56**	**−8.38**	0.49
500 × 20	−3.15	−3.93	−2.52	0.58	−3.96	−4.50	−3.49	0.40	**−4.53**	**−4.96**	**−4.08**	0.37
Avg	−9.45	−10.16	−8.66	0.61	−9.95	−10.43	−9.43	0.42	**−10.30**	**−10.80**	**−9.69**	0.46

Table 5
Computational results ($\tau = 2$).

$n \times m$	GA				DABC				BEDA			
	Avg	Best	Worst	Std	Avg	Best	Worst	Std	Avg	Best	Worst	Std
20 × 5	−18.45	−18.88	−17.67	0.52	**−18.72**	**−19.10**	**−18.32**	0.35	**−18.72**	**−19.10**	−18.20	0.40
20 × 10	−28.87	−29.49	−27.80	0.71	−29.33	−29.62	**−28.96**	0.31	**−29.38**	**−29.68**	−28.78	0.41
20 × 20	−28.23	−29.04	−27.32	0.72	−29.10	−29.40	−28.37	0.44	**−29.12**	**−29.49**	**−28.59**	0.41
50 × 5	−11.62	−12.28	−10.76	0.63	−12.36	−12.80	−11.87	0.4	**−12.52**	**−12.99**	**−11.91**	0.45
50 × 10	−15.28	−17.06	−13.82	1.28	−16.57	−17.40	−15.77	0.65	**−16.95**	**−18.19**	**−15.65**	1.04
50 × 20	−17.64	−18.89	−16.37	1.05	−18.94	−19.93	−18.13	0.71	**−19.39**	**−20.40**	**−18.33**	0.83
100 × 5	−8.62	−9.36	−7.84	0.62	−9.14	−9.62	−8.52	0.47	**−9.31**	**−9.97**	**−8.67**	0.53
100 × 10	−10.73	−11.39	−10.03	0.57	−11.46	−11.98	−10.98	0.4	**−11.96**	**−12.71**	**−11.16**	0.62
100 × 20	−12.78	−13.76	−11.52	0.91	−13.58	−14.17	−12.83	0.55	**−14.91**	**−15.74**	**−14.01**	0.69
200 × 10	−5.99	−6.68	−5.27	0.57	−5.70	−6.18	−5.00	0.47	**−6.19**	**−6.68**	**−5.70**	0.39
200 × 20	−8.25	−9.01	−7.40	0.64	−8.36	−8.94	−7.92	0.42	**−9.44**	**−10.22**	**−8.65**	0.65
500 × 20	−3.31	−4.13	−2.50	0.69	−4.19	−4.81	−3.70	0.47	**−4.57**	**−4.88**	**−4.23**	0.27
Avg	−14.15	−15.00	−13.19	0.74	−14.79	−15.33	−14.20	0.47	**−15.20**	**−15.84**	**−14.49**	0.56

Table 6
Computational results ($\tau = 3$).

$n \times m$	GA				DABC				BEDA			
	Avg	Best	Worst	Std	Avg	Best	Worst	Std	Avg	Best	Worst	Std
20 × 5	−41.60	−42.28	−40.69	0.76	**−42.04**	**−42.28**	**−41.55**	0.32	−42.03	**−42.28**	**−41.55**	0.38
20 × 10	−68.94	−69.47	−68.08	0.58	−69.45	−69.54	−69.29	0.13	**−69.51**	**−69.59**	**−69.39**	0.10
20 × 20	−81.86	−82.61	−80.21	1.05	−82.21	−82.61	−81.81	0.41	**−82.29**	**−82.61**	**−81.88**	0.36
50 × 5	−14.90	−15.69	−14.05	0.66	−15.71	−16.28	−15.26	0.41	**−16.26**	**−16.85**	**−15.45**	0.55
50 × 10	−23.22	−25.33	−20.73	1.89	−24.92	−26.30	−23.28	1.24	**−25.90**	**−27.49**	**−23.63**	1.51
50 × 20	−30.26	−32.45	−28.12	1.75	−32.38	−34.05	−30.47	1.42	**−32.68**	**−34.33**	**−30.52**	1.52
100 × 5	−9.42	−10.31	−8.36	0.76	−10.12	−10.86	−9.39	0.61	**−10.52**	**−11.09**	**−9.92**	0.47
100 × 10	−11.90	−12.85	−10.94	0.77	−12.52	−13.24	−11.90	0.52	**−13.30**	**−13.87**	**−12.52**	0.52
100 × 20	−15.53	−16.57	−14.3	0.95	−16.73	−17.54	−15.88	0.68	**−18.22**	**−19.37**	**−17.23**	0.88
200 × 10	−6.67	−7.57	−5.81	0.73	−6.33	−7.16	−5.70	0.62	**−7.13**	**−7.69**	**−6.55**	0.46
200 × 20	−9.44	−10.23	−8.68	0.62	−9.83	−10.65	−9.14	0.58	**−10.81**	**−11.62**	**−10.01**	0.65
500 × 20	−3.87	−4.65	−3.15	0.60	−4.76	−5.29	−4.32	0.40	**−5.19**	**−5.61**	**−4.73**	0.35
Avg	−26.47	−27.5	−25.26	0.93	−27.25	−27.98	−26.50	0.61	**−27.82**	**−28.53**	**−26.95**	0.65

follows: $P\_Size = n$, $\eta = 20$, $\gamma = 5$, $\alpha(0) = \min\{0.005n, 0.5\}$, and $\beta(0) = \min\{0.005n, 0.5\}$. Such a parameter setting will be used for the BEDA in the following tests and comparisons.

4.2. Influence of bi-population strategy

Among the above five parameters, parameter γ determines the size of each sub-population. According to the above discussion, γ is fixed as 5. Obviously, when $\gamma = 0$, the size of sub-population $P2$ becomes 0, and the BEDA is converted into a single-population EDA with P1 only. When $\gamma = 100$, the size of sub-population $P1$ becomes 0, and the BEDA is converted into a single-population EDA with $P2$ only. Next, we compare the two single-population EDAs ($\gamma = 0, 100$) with the proposed BEDA ($\gamma = 5$). We run the BEDA 5 independent times for each instance considering different τ, and the interval plot of the average relative percent deviation for each choice of $\gamma = \{0, 5, 100\}$ is illustrated in Fig. 6.

From Fig. 6, it can be seen that the performances are worse when $\gamma = 0$ and $\gamma = 100$ than that when $\gamma = 5$. So, it can be concluded that the BEDA outperforms the single-population EDA. That is, the bi-population strategy is effective to improve the performance of the EDA.

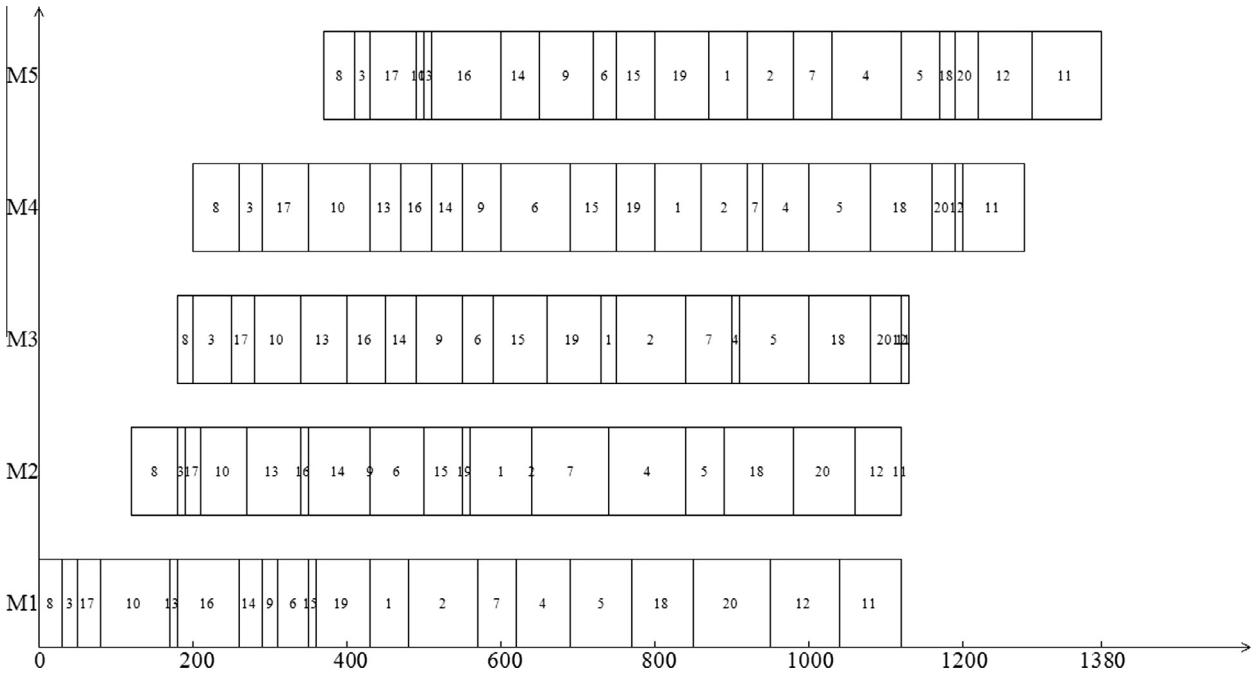

Fig. 8. Best solution of instance Ta001 obtained by BEDA.

Table 7
Results of statistical tests.

	BEDA (20 runs)		*p*-Value (vs GA)	*p*-Value (vs DABC)
	Avg	Std		
$\tau = 1$	−10.29	0.43	0.00075	0.063
$\tau = 2$	−15.22	0.48	0.00028	0.043
$\tau = 3$	−27.84	0.60	0.00022	0.031
Total	−17.78	0.50	0.00034	0.042

4.3. Influence of the local search procedure

To intensify the local exploitation, the insertion operator is performed iteratively on the best solution at each generation. In order to show the effectiveness of local search, we compare the BEDA to that without the local search procedure (denoted as BEDA*). In particular, the BEDA and BEDA* are run 5 independent times for all instances, respectively. The interval plot of the average relative percent deviation considering different τ is shown in Fig. 7.

From Fig. 7, it can be clearly seen that the BEDA outperforms the BEDA* for all the cases. So, the local search procedure is useful to improve the performances by enhancing the exploitation of the EDA.

4.4. Comparisons of BEDA, GA and DABC

To further evaluate the performances of the BEDA, we compare it with the existing GA and DABC recently published in [22]. Same as the literature [22], five independent runs are carried out for each instance, and the minimum (Best), average (Avg), maximum (Worst) and the standard deviation (Std) of the five runs are reported. For each case of due date tightness factor, the results are grouped by the combination of n and m. The comparative results are listed in Tables 4–6, where the results of the GA and DABC are directly from the literature [22] and the best results are highlighted in bold.

From Tables 4–6, it can be seen that the BEDA outperforms GA and DABC in solving the NIPFSP for all the tight, medium and loose cases. With less computational time, the BEDA can obtain better solutions in terms of minimum, average and maximum results. As for the standard deviations, the results of the three algorithms are almost at the same level. In Fig. 8, the Gantt chart of the best solution obtained by the BEDA in solving instance Ta001 is shown.

For further statistical analysis, we run the BEDA 20 times for each instance and use the student's *t*-test to show the statistical significance of the BEDA over the GA and DABC. The results of statistical tests are listed in Table 7.

When comparing the BEDA with GA, it can be seen from Table 7 that the *p*-values are $0.00 < 0.05$ for the tight, medium and loose cases. So, the differences between BEDA and GA are statistically significant at 95% confidence level for all the cases. When comparing the BEDA with DABC, it can be seen from Table 7 that the *p*-values are 0.04, 0.03, respectively, for the medium and loose cases, showing the differences between BEDA and DABC are significant at 95% confidence level; Although the *p*-value is 0.06 for the tight case, the difference between BEDA and DABC is still significant at 90% confidence level. Considering the average value for all cases together, the *p*-value on average is $0.04 < 0.05$, showing the BEDA is significantly better than DABC at 95% confidence level.

From the above tests and comparisons, it can be concluded that the BEDA is a more effective and efficient approach for solving the NIPFSP with the total tardiness criterion. The superiority of the BEDA mainly owes to the following two aspects. (1) Using the bi-population strategy, two sub-populations with specific probability models cooperate with each other for balancing global exploration and local exploitation. (2) Using the insertion based local search, the search region around the best solution of the population can be well exploited.

5. Conclusions

This was the first reported work of using the EDA to solve the NIPFSP with the total tardiness criterion. A bi-population EDA was developed by using two sub-populations that evolve and collaborate together. Two probability models were adjusted in different ways, stressing the balance of global exploration and local

exploitation. An insertion neighborhood based local search procedure was embedded to strengthen exploitation. The DOE based experimental investigation was carried out, showing the influence of parameter setting. The effectiveness of the bi-population strategy and local search procedure was demonstrated by numerical comparisons. The comparisons also showed that the BEDA outperformed the recently published algorithms in terms of solution quality and search efficiency. Further work could focus on the applications of the proposed algorithm to some real industrial problems. It is also interesting to develop the adaptive BEDA with new collaborating mechanisms between two sub-populations or knowledge based rules for adjusting models as well as to generalize the BEDA for solving the problems with other scheduling objectives and multiple objectives.

Acknowledgments

The authors would like to thank the Editor-in-Chief and the anonymous reviewers for their constructive comments to improve the paper. This research is partially supported by the National Key Basic Research and Development Program of China (No. 2013CB329503), the National Science Foundation of China (No. 61174189), the Doctoral Program Foundation of Institutions of Higher Education of China (20130002110057).

References

[1] S.M. Johnson, Optimal two-and three-stage production schedules with setup times included, Naval Res. Logist. Quart. 1 (1) (1954) 61–68.
[2] J.M. Framinan, J.N.D. Gupta, R. Leisten, A review and classification of heuristics for permutation flow-shop scheduling with makespan objective, J. Oper. Res. Soc. 55 (12) (2004) 1243–1255.
[3] R. Ruiz, C. Maroto, A comprehensive review and evaluation of permutation flowshop heuristics, Eur. J. Oper. Res. 165 (2) (2005) 479–494.
[4] S.R. Hejazi, S. Saghafian, Flowshop-scheduling problems with makespan criterion: a review, Int. J. Prod. Res. 43 (14) (2005) 2895–2929.
[5] M.M. Yenisey, B. Yagmahan, Multi-objective permutation flow shop scheduling problem: literature review, classification and current trends, Omega (2013), http://dx.doi.org/10.1016/j.omega.2013.07.004.
[6] B. Naderi, R. Tavakkoli-Moghaddam, M. Khalili, Electromagnetism-like mechanism and simulated annealing algorithms for flowshop scheduling problems minimizing the total weighted tardiness and makespan, Knowl. – Based Syst. 23 (2) (2010) 77–85.
[7] W.C. Lee, Y.H. Chung, Permutation flowshop scheduling to minimize the total tardiness with learning effects, Int. J. Prod. Econ. 141 (1) (2013) 327–334.
[8] N.E.H. Saadani, A. Guinet, M. Moalla, Three stage no-idle flow-shops, Comput. Ind. Eng. 44 (3) (2003) 425–434.
[9] P.J. Kalczynski, J. Kamburowski, A heuristic for minimizing the makespan in no-idle permutation flow shops, Comput. Ind. Eng. 49 (1) (2005) 146–154.
[10] P. Baptiste, L.K. Hguny, A branch and bound algorithm for the F/no-idle/C_{max}, Proc. Int. Conf. Ind. Eng. Prod. Manage. (1997) 429–438.
[11] I. Adiri, D. Pohoryles, Flowshop/no-idle or no-wait scheduling to minimize the sum of completion times, Naval Res. Logist. Quart. 29 (3) (1982) 495–504.
[12] P. Vachajitpan, Job sequencing with continuous machine operation, Comput. Ind. Eng. 6 (3) (1982) 255–259.
[13] J. Kamburowski, More on three-machine no-idle flow shops, Comput. Ind. Eng. 46 (3) (2004) 461–466.
[14] N.E.H. Saadani, A. Guinet, M. Moalla, A travelling salesman approach to solve the F/no-idle/C_{max} problem, Eur. J. Oper. Res. 161 (1) (2005) 11–20.
[15] M. Nawaz, E.E. Enscore, I. Ham, A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem, Omega 11 (1) (1983) 91–95.
[16] D. Baraz, G. Mosheiov, A note on a greedy heuristic for flow-shop makespan minimization with no machine idle-time, Eur. J. Oper. Res. 184 (2) (2008) 810–813.
[17] Q.K. Pan, L. Wang, A novel differential evolution algorithm for no-idle permutation flow-shop scheduling problems, Eur. J. Ind. Eng. 2 (3) (2008) 279–297.
[18] Q.K. Pan, L. Wang, No-idle permutation flow shop scheduling based on a hybrid discrete particle swarm optimization algorithm, Int. J. Adv. Manuf. Technol. 39 (7–8) (2008) 796–807.
[19] G. Deng, X. Gu, A hybrid discrete differential evolution algorithm for the no-idle permutation flow shop scheduling problem with makespan criterion, Comput. Oper. Res. 39 (9) (2012) 2152–2160.
[20] M.F. Tasgetiren, Q.K. Pan, P.N. Suganthan, O. Buyukdagli, A variable iterated greedy algorithm with differential evolution for the no-idle permutation flowshop scheduling problem, Comput. Oper. Res. 40 (7) (2013) 1729–1743.
[21] M.F. Tasgetiren, Q.K. Pan, P.N. Suganthan, T.J. Chua, A differential evolution algorithm for the no-idle flowshop scheduling problem with total tardiness criterion, Int. J. Prod. Res. 49 (16) (2011) 5033–5050.
[22] M.F. Tasgetiren, Q.K. Pan, P.N. Suganthan, A. Oner, A discrete artificial bee colony algorithm for the no-idle permutation flowshop scheduling problem with the total tardiness criterion, Appl. Math. Model. 37 (10–11) (2013) 6758–6779.
[23] P. Larranaga, J. Lozano, Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation, Springer, Netherlands, 2002.
[24] B. Jarboui, M. Eddaly, P. Siarry, An estimation of distribution algorithm for minimizing the total flowtime in permutation flowshop scheduling problems, Comput. Oper. Res. 36 (9) (2009) 2638–2646.
[25] L. Wang, S.Y. Wang, Y. Xu, G. Zhou, M. Liu, A bi-population based estimation of distribution algorithm for the flexible job-shop scheduling problem, Comput. Ind. Eng. 62 (4) (2012) 917–926.
[26] U. Aickelin, J. Li, An estimation of distribution algorithm for nurse scheduling, Ann. Oper. Res. 155 (1) (2007) 289–309.
[27] L. Wang, C. Fang, An effective estimation of distribution algorithm for the multi-mode resource-constrained project scheduling problem, Comput. Oper. Res. 39 (2) (2012) 449–460.
[28] D.C. Montgomery, Design and Analysis of Experiments, John Wiley & Sons, Arizona, 2005.
[29] L. Wang, S.Y. Wang, Y. Xu, An effective hybrid EDA-based algorithm for solving multidimensional knapsack problem, Exp. Syst. Appl. 39 (5) (2012) 5593–5599.
[30] E. Vallada, R. Ruiz, G. Minella, Minimising total tardiness in the m-machine flowshop problem: a review and evaluation of heuristics and metaheuristics, Comput. Oper. Res. 35 (4) (2008) 1350–1373.
[31] E. Taillard, Benchmarks for basic scheduling problems, Eur. J. Oper. Res. 64 (2) (1993) 278–285.
[32] K.R. Baker, J.W.M. Bertrand, An investigation of due-date assignment rules with constrained tightness, J. Oper. Manage. 1 (3) (1981) 109–120.