Computers & Industrial Engineering 62 (2012) 917–926

Contents lists available at SciVerse ScienceDirect

Computers & Industrial Engineering

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

A bi-population based estimation of distribution algorithm for the flexible job-shop scheduling problem ☆

Ling Wang *, Shengyao Wang, Ye Xu, Gang Zhou, Min Liu

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

ARTICLE INFO

Article history:
Received 15 June 2011
Received in revised form 12 December 2011
Accepted 13 December 2011
Available online 17 December 2011

Keywords:
Flexible job-shop scheduling problem
Estimation of distribution algorithm
Bi-population
Probability model
Critical path
Design of experiment

ABSTRACT

In this paper, an effective bi-population based estimation of distribution algorithm (BEDA) is proposed to solve the flexible job-shop scheduling problem (FJSP) with the criterion to minimize the maximum completion time (makespan). The BEDA stresses the balance between global exploration and local exploitation. In the framework of estimation of distribution algorithm, two sub-populations are used to adjust the machine assignment and operation sequence respectively with a splitting criterion and a combination criterion. At the initialization stage, multiple strategies are utilized in a combination way to generate the initial solutions. At the global exploration phase, a probability model is built with the superior population to generate the new individuals and a mechanism is proposed to update the probability model. At the local exploitation phase, different operators are well designed for the two sub-populations to generate neighbor individuals and a local search strategy based on critical path is proposed to enhance the exploitation ability. In addition, the influence of parameters is investigated based on Taguchi method of design of experiment, and a suitable parameter setting is determined. Finally, numerical simulation based on some widely used benchmark instances is carried out. The comparisons between BEDA and some existing algorithms as well as the single-population based EDA demonstrate the effectiveness of the proposed BEDA in solving the FJSP.

1. Introduction

The flexible job-shop scheduling problem (FJSP) is a generalization of the classical job-shop scheduling problem (JSP) for flexible manufacturing systems, which is of wide application background and is very close to the real manufacturing situation. In the FJSP, each machine may have the ability of performing more than one type of operations. The FJSP consists of two sub-problems: the routing sub-problem that assigns each operation to a machine among a set of capable machines, and the scheduling sub-problem that sequences the assigned operations on all the machines to obtain a feasible schedule to optimize a certain objective function. The FJSP is more difficult to solve than the classical JSP because it should determine the assignment of operations to machines and the sequence of all the operators. Therefore, it is considered as one of the most difficult problems in the field of combinatorial optimization (Lawler, Lenstra, Rinnooy Kan, & Shmoys, 1993). The study on the FJSP in theory, methodology and applications has significant importance in both academic field and application field.

The first work to address the FJSP was by Bruker and Schlie (1990), where a polynomial algorithm was proposed to solve the problem with two jobs and each operation has the same processing time on different machines. Later, Brandimarte (1993) proposed a hybrid tabu search (TS) algorithm with some existing dispatching rules to solve the FJSP. Dauzere-Peres and Paulli (1997) proposed a TS algorithm based on an integrated approach, which was improved by Mastrolilli and Gambardella (2000) with two developed neighborhood functions in terms of computation time and solution quality. Gao, Sun, and Gen (2008) proposed a genetic algorithm (GA) hybridizing with the variable neighborhood search, and Pezzella, Morganti, and Ciaschetti (2008) proposed a GA integrating different strategies. Recently, Yazdani, Amiri, and Zandieh (2010) developed a parallel variable neighborhood search (PVNS) algorithm based on six neighborhood structures, and Xing, Chen, Wang, Zhao, and Xiong (2010) proposed a knowledge-based ant colony optimization algorithm (KBACO).

As for the study on the multi-objective flexible job-shop scheduling problem (MFJSP), Kacem, Hammadi, and Borne (2002) proposed a localization approach to solve the assignment problem in the MFJSP. Xia and Wu (2005) proposed a hierarchical approach by using particle swarm optimization (PSO) to assign operations to machines and by using simulated annealing (SA) to schedule operations on each machine. Tay and Ho (2008) developed an integrated approach based on genetic programming (GP) for

☆ This manuscript was processed by Area Editor (Maged M. Dessouky).

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

0360-8352/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cie.2011.12.014

solving the MFJSP to minimize makespan, mean tardiness and mean flow times. Recently, Xing, Chen, and Yang (2009) introduced a local search algorithm, and Li, Pan, and Liang (2010) developed a TS algorithm with an effective neighborhood structure by combining two adaptive rules.

As a relatively new population-based optimization algorithm, estimation of distribution algorithm (EDA) has gained an increasing study and wide applications during recent years (Larranaga & Lozano, 2002). Population-based incremental learning (PBIL) (Baluja, 1994) is the earliest model of the EDA. According to the complexity of the model, the EDA can be classified as univariate model, bivariate model or multivariate model. The PBIL, univariate marginal distribution algorithm (UMDA) (MÄuhlenbein & Paass, 1996) and compact GA (CGA) (Harik, Lobo, & Goldberg, 1998) are univariate models, while mutual information maximization for input clustering (MIMIC) (De Bonet, Isbell, & Viola, 1997), combining optimizers with mutual information trees (COMIT) (Baluja & Davies, 1997) and bivariate marginal distribution algorithm (BMDA) (Pelikan & Mühlenbein, 1999) are bivariate models. The factorized distribution algorithms (FDA) (Mühlenbein & Mahnig, 1999), extended compact GA (ECGA) (Harik, 1999) and Bayesian optimization algorithm (BOA) (Pelikan, Goldberg, & Cantú-Paz, 1999) are multivariate models. For more details about the EDA, please refer (Larranaga & Lozano, 2002).

So far, the EDA has been applied to a variety of academic and application problems, such as feature selection, cancer classification, quadratic assignment problem, machinery structure design and nurse rostering (Jarboui, Eddaly, & Siarry, 2009; Wang & Fang, 2012; Zhou & Sun, 2007). However, to the best of our knowledge, there is no research work about the EDA for solving the FJSP. In this paper, we will propose a bi-population based EDA (BEDA) to solve the FJSP with the criterion to minimize the maximum completion time. When designing the algorithm, we stress the balance between global exploration and local exploitation. Multiple strategies are employed in a combination way to generate initial solutions. A probability model is built and an updating mechanism is proposed for generating new solutions at global exploration phase. Different searching operators are designed and the critical path based local search is applied at local exploitation phase. Especially, two sub-populations are used to adjust the machine assignment and operation sequence respectively with a splitting criterion and a combination criterion. Besides, we investigate the influence of parameter setting based on the design of experiment testing. Finally, we use two sets of benchmark instances to test the performances of the BEDA and to compare the BEDA with some existing methods and the single-population based EDA to solve the FJSP.

The remainder of the paper is organized as follows: In Section 2, the FJSP is formulated. In Section 3, the basic EDA is introduced briefly. Then, the framework of BEDA for solving the FJSP is proposed in Section 4. The influence of parameter setting is investigated based on design of experiment testing in Section 5, and computational results and comparisons are provided as well. Finally we end the paper with some conclusions in Section 6.

2. Problem formulation

The flexible job-shop scheduling problem (FJSP) is commonly defined as follows. There are n jobs $J = \{J_1, J_2, \ldots, J_n\}$ to be processed on m machines $M = \{M_1, M_2, \ldots, M_m\}$. A job J_i is formed by a sequence of n_i operations $\{O_{i,1}, O_{i,2}, \ldots, O_{i,n_i}\}$ to be performed one after another according to a given sequence. The execution of $O_{i,j}$ requires one machine out of a set of $m_{i,j}$ given machines $M_{i,j} \subseteq M$. Preemption is not allowed, i.e., each operation must be completed without interruption once it starts. All jobs and machines are available at time 0. Setup times of machines and move times between operations are negligible. The processing time of $O_{i,j}$ performed on machine M_k is $t_{i,j,k} > 0$. Let $S_{i,j}$ and $C_{i,j}$ be the starting time and completion time of operation $O_{i,j}$. Let $S^m_{k,r}$ be the starting time of $O_{i,j}$ performed on machine k in priority r. Let q_k be the number of operations assigned to machine k. The FJSP is to determine both the assignment of machines and the sequence of operations on all the machines to minimize a certain scheduling objective function, e.g., the maximum complete time of all the jobs ($C_{\max}$).

Mathematically, the FJSP with makespan minimization can be formulated as follows (Li, Pan, Suganthan, & Chua, 2011):

$$\text{Minimize } C_{\max} \tag{1}$$

$$\text{Subject to: } S_{i,j} + \sum_{k \in M_{i,j}} (t_{i,j,k} \cdot x_{i,j,k}) \leqslant S_{i,j+1}, i = 1,2,\ldots,n;\ j = 1,2,\ldots,n_i - 1 \tag{2}$$

$$\sum_{k \in M_{i,j}} x_{i,j,k} = 1,\ i = 1,2,\ldots,n, j = 1,2,\ldots,n_i;\ k = 1,2,\ldots,m \tag{3}$$

$$\sum_{r} y_{i,j,k,r} = x_{i,j,k},\ i = 1,2,\ldots,n, j = 1,2,\ldots,n_i;\ k = 1,2,\ldots,m;\ r = 1,2,\ldots,q_k \tag{4}$$

$$S^m_{k,r} + t_{i,j,k} \cdot y_{i,j,k,r} \leqslant S^m_{k,(r+1)}, i = 1,2,\ldots,n; j = 1,2,\ldots,n_i;\ k = 1,2,\ldots,m;\ r = 1,2,\ldots,q_k - 1 \tag{5}$$

$$S^m_{k,r} + (1 - y_{i,j,k,r})L \geqslant S_{i,j}, i = 1,2,\ldots,n; j = 1,2,\ldots,n_i;\ k = 1,2,\ldots,m;\ r = 1,2,\ldots,q_k \tag{6}$$

$$S^m_{k,r} S_{i,j} + (1 - y_{i,j,k,r})L, i = 1,2,\ldots,n; j = 1,2,\ldots,n_i;\ k = 1,2,\ldots,m;\ r = 1,2,\ldots,q_k \tag{7}$$

$$x_{i,j,k} = \begin{cases} 1, & \text{if machine } k \text{ is selected for operation } O_{i,j} \\ 0, & \text{otherwise} \end{cases} \tag{8}$$

$$y_{i,j,k,r} = \begin{cases} 1, & \text{if operation } O_{i,j} \text{ is performed on machine } k \text{ in priority } r \\ 0, & \text{otherwise} \end{cases} \tag{9}$$

where Eq. (1) is the objective function; Eq. (2) ensures that the operations belonging to the same job satisfy the precedence constraints; Eq. (3) ensures that each operation is assigned to only one machine from its candidate machine set; Eq. (4) guarantees that each operation should be processed only at one priority on one machine; Eq. (5) ensures that each machine processes one operation at a time; Eqs. (6) and (7) guarantee that each operation should be started only after the previous operation of the same machine and the previous operation of the same job are both completed; and L is a number large enough.

3. Estimation of distribution algorithm

Estimation of distribution algorithms (EDA) is a relatively new paradigm in the field of evolutionary computation, which employs explicit probability distributions in optimization (Larranaga & Lozano, 2002). Compared with the genetic algorithm, the EDA reproduces new population implicitly instead of the crossover and mutation operators. In the EDA, a probability model of the most promising area is built by statistical information based on the searching experience, and then the probability model is used for sampling to generate the new individuals. Meanwhile, the probability model is updated in each generation with the potential individuals of the new population. In such an iterative way, the population evolves, and finally satisfactory solutions can be obtained.

The general framework of the EDA is illustrated in Fig. 1.

The critical step of the above procedure is to estimate the probability distribution. The EDA makes use of the probability model to describe the distribution of the solution space. The updating process reflects the evolutionary trend of the population. Due to the difference of problem types, a proper probability model and a suitable updating mechanism should be well developed to estimate the underlying probability distribution. Nevertheless, the EDA pays more attention to global exploration while its exploitation capability is relatively limited. So, an effective EDA should balance the exploration and the exploitation abilities.

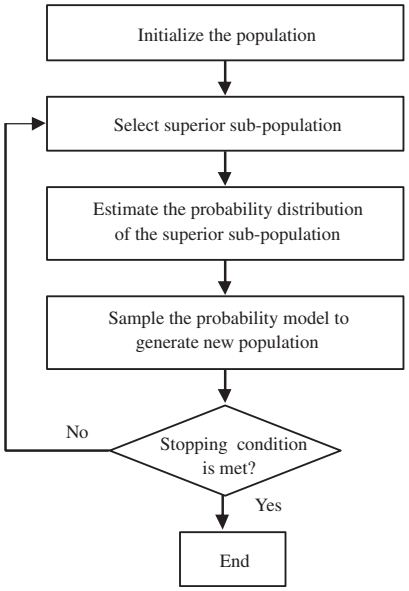

Fig. 1. The general framework of the EDA.

Table 1
Processing time table.

Operations	Machines			
	M_1	M_2	M_3	M_4
$O_{1,1}$	4	7	6	5
$O_{1,2}$	2	6	∞	5
$O_{2,1}$	4	5	7	∞
$O_{2,2}$	5	∞	6	3
$O_{2,3}$	∞	5	4	7
$O_{3,1}$	5	3	∞	6
$O_{3,2}$	∞	∞	4	∞
$O_{4,1}$	2	4	∞	5
$O_{4,2}$	∞	4	2	∞
$O_{4,3}$	5	4	6	3

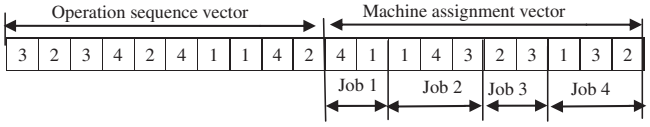

Fig. 2. Illustration of the representation of a feasible solution.

4. BEDA for FJSP

In this section, we will propose a bi-population based estimation of distribution algorithm (BEDA) to solve the FJSP. First, we will introduce the solution representation, population initialization, probability model and updating mechanism, splitting criterion and combination criterion, searching operators for sub-populations and local search strategy. Then, we will present the framework of the BEDA.

4.1. Solution representation

Every individual of the population is a solution of the FJSP, which is a combination of operation scheduling decision and machine assignment. Thus, a solution can be expressed by the processing sequence of operations on the machines and the assignment of operations on machines. In this paper, a solution is represented by two vectors (operation sequence vector and machine assignment vector), which correspond to the two sub-problems of the FJSP, respectively.

For the operation sequence vector, the number of elements equals to the total number of operations T_o. The job number denotes the operation of each job, and the kth occurrence of a job number refers to the kth operation in the sequence of this job. For the machine assignment vector, each element represents the corresponding selected machine for each operation. So the number of elements is also T_o. To explain the representation, we provide an example by considering a problem with 4 jobs and 4 machines as shown in Table 1. In Fig. 2, it illustrates the representation of a feasible solution.

The operation sequence and machine assignment of the solution in Fig. 2 can be interpreted as follows: ($O_{3,1}$, M_2), ($O_{2,1}$, M_1), ($O_{3,2}$, M_3), ($O_{4,1}$, M_1), ($O_{2,2}$, M_4), ($O_{4,2}$, M_3), ($O_{1,1}$, M_4), ($O_{1,2}$, M_1), ($O_{4,3}$, M_2), ($O_{2,3}$, M_3). The Gantt chart of this solution is illustrated in Fig. 3.

4.2. Population initialization

At the beginning of the evolution, some different strategies are utilized in a hybrid way in this paper for generating the initial solutions to guarantee the initial population with certain quality and diversity.

To generate the initial machine assignments, the following two rules are applied. (1) Random rule. Randomly select a machine from the candidate machines set for each operation and then place it at the position in the machine assignment vector. (2) Global minimum processing time rule (Pezzella et al., 2008). In our algorithm, for the initial machine assignments 60% solutions in the population are generated by the global minimum processing time rule and the other 40% solutions are generated by the random rule.

Once the machines are assigned to all the operations, all the operations will be sequenced. A schedule is feasible only if all the precedence constraints among operations of the same job are not violated. We apply the following three rules to generate initial operation sequences. (1) Random rule. Randomly generate the sequence of the operations on each machine. (2) Most time remaining rule (Pezzella et al., 2008). Sequence the jobs in the order of non-increasing remaining time, that is, the job with the most remaining time will be selected first. (3) Most number of operations remaining rule (Pezzella et al., 2008). The job with most remaining operations unprocessed has a high priority to be selected. In our algorithm, for the initial operation sequences 20% solutions in the population are generated by random rule, 40% solutions are generated by most time remaining rule, and the other 40% solutions are generated by most number of operations remaining rule.

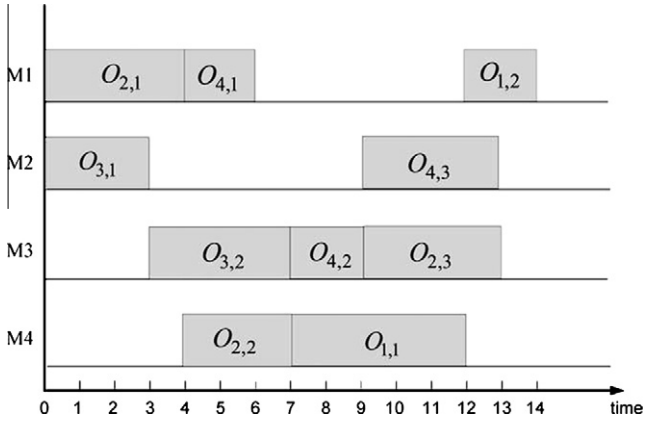

Fig. 3. Gantt chart of the solution shown in Fig. 2.

4.3. Probability model and updating mechanism

The BEDA stresses the balance between global exploration and local exploitation. At the exploration phase, the algorithm works like the basic EDA. Different from GA that produces offspring through crossover and mutation operators, EDA does it by sampling according to a probability model. So, the probability model has a great effect on the performances of EDA. In this paper, the probability model is designed as two probability matrixes, i.e., operation probability matrix and machine probability matrix.

The element $p_{ij}(l)$ of the operation probability matrix A_1 represents the probability that job j appears before or in position i of the operation sequence vector at generation l. The value of p_{ij} refers to the importance of a job when scheduling the operations on machines. For all i ($i = 1, 2, \ldots, T_o$) and j ($j = 1, 2, \ldots, n$), p_{ij} is initialized as $p_{ij}(0) = 1/n$, which ensures that the whole solution space can be sampled uniformly.

The element $q_{ijk}(l)$ of the machine probability matrix A_2 represents the probability that operation $O_{i,j}$ is processed on machine k at generation l. The value of q_{ijk} indicates the rationality of an operation processed on a certain machine. The probability matrix A_2 is initialized as follows:

$$q_{ijk}(0) = \begin{cases} \frac{1}{m_{i,j}}, & \text{if } O_{i,j} \text{ can be processed on machine } k \\ 0, & \text{else} \end{cases}, \quad \forall i,j,k \tag{10}$$

Via sampling according to the probability matrixes A_1 and A_2, new promising individuals may be generated. In particular, to generate a new solution the operation sequence vector should be generated first. For every position i, job j is selected with the probability of p_{ij}. If job j has already appeared n_j times in the operation sequence vector, it means the processing procedure of job j is completed. Then, the whole column $p_{1j}, p_{2j}, \ldots, p_{T_oj}$ of probability matrix A_1 will be set as zero. Similarly, the machine assignment vector is generated according to probability matrix A_2. In such a way, P individuals are generated.

Next, it determines the superior population that consists of the best SI solutions. And then, the probability matrixes A_1 and A_2 are updated according to the following equations:

$$p_{ij}(l+1) = (1-\alpha)p_{ij}(l) + \frac{\alpha}{i \times SI}\sum_{s=1}^{SI} I_{ij}^{s}, \quad \forall i,j \tag{11}$$

$$q_{ijk}(l+1) = (1-\beta)q_{ijk}(l) + \frac{\beta}{SI}\sum_{s=1}^{SI} \tilde{I}_{ijk}^{s}, \quad \forall i,j,k \tag{12}$$

where $\alpha, \beta \in (0, 1)$ are the learning rates of A_1 and A_2 respectively, I_{ij}^{s} and $\tilde{I}_{ij}^{s}$ are the following indicator functions of the sth individual in the superior population.

$$I_{ij} = \begin{cases} 1, & \text{if job } j \text{ appears before or in position } i \\ 0, & \text{else} \end{cases} \tag{13}$$

$$\tilde{I}_{ijk} = \begin{cases} 1, & \text{if operation } O_{i,j} \text{ is processed on machine } k \\ 0, & \text{else} \end{cases} \tag{14}$$

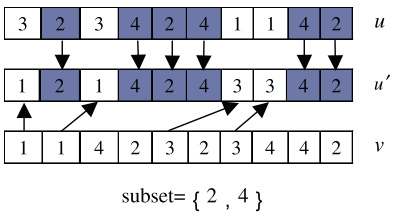

Fig. 4. Illustration of searching operator for SP2.

Note that, our probability is different from that used by Chen, Chen, Chang, Zhang, and Chen (2010) for different kinds of scheduling problems. To solve the FJSP, we design the probability model as two probability matrixes, i.e., operation probability matrix and machine probability matrix. To solve the single machine scheduling problems, Chen et al. (2010) used only one matrix as the probability model that represents the probability of all jobs at different positions. Moreover, the element $p_{ij}(l)$ of our designed operation probability matrix to represent the probability that job j appears before or in position i of the operation sequence vector at generation l. This mechanism takes into account the importance of a job when scheduling the operations on machines. Besides, our designed updating mechanism is similar to PBIL (Baluja, 1994) which also considers the previous probability information, while Chen's updating mechanism (Chen et al., 2010) is similar to UMDA (MÄuhlenbein & Paass, 1996) which considers the probability distribution of the current generation only.

4.4. Splitting and combination criteria

The simple EDA pays more attention to global exploration while its exploitation capability is limited. So, an effective EDA should balance the exploration and the exploitation abilities. Meanwhile, it often appears the stagnation in the EDA after running some generations, which is a common problem for the EDA that entirely samples new solutions by using the probability models (Chen et al., 2010). To achieve satisfactory searching quality, the algorithm should consider enhancing the exploitation capability and maintaining certain population diversity.

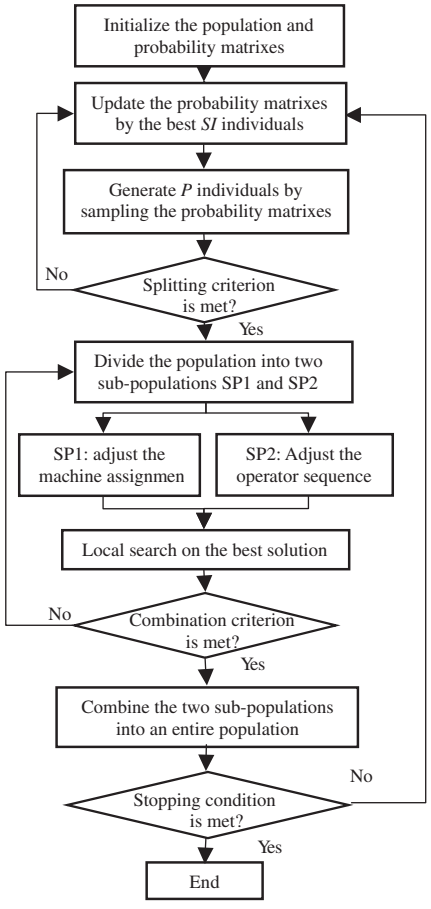

Fig. 5. The framework of the BEDA for the FJSP.

Since the FJSP consists of two sub-problems, i.e., operation routing and machine scheduling, it may optimize the objective function by adjusting the operation sequence and the machine assignment, respectively. In addition, it may be unnecessary to perturb both the operation sequence vector and machine assignment vector of a certain solution simultaneously in every generation, especially for local exploitation. Thus, we consider using two sub-populations to adjust the machine assignment and operation sequence respectively with a splitting criterion and a combination criterion.

In the BEDA, the population with P individuals may be divided into two sub-populations with a splitting criterion, and the two sub-populations may be recombined together as the entire population with a combination criterion. In particular, if the best solution found so far keeps fixed at consecutive generation *ter* (i.e., the splitting criterion), the population will be divided into two sub-populations with size $P/2$ to generate neighbor individuals with different methods. To guarantee the balance of the two sub-populations, before splitting it sorts all the individuals in an ascending order according to their makespan values. Then, the first individual is assigned to the first sub-population (SP1), the second one is assigned to the second sub-population (SP2), the third one is assigned to SP1, and so on, until all the individuals are assigned. After splitting, if the best solution found by SP1 and SP2 has no improvement after *ter* generations (i.e., the combination criterion), SP1 and SP2 are recombined together as an entire population for further evolution.

In the next sub-section, we will describe the design of searching operators for the two sub-populations after splitting.

4.5. Searching operators for sub-populations

In the exploitation phase of the BEDA, SP1 and SP2 have different function to exploit neighbor solutions. Since it should determine machine assignment and operation sequence, the two sub-populations use different operators to adjust the machine assignment and operation sequence, respectively.

4.5.1. Operator for SP1

The sub-population SP1 only adjusts the machine assignment. For the two individuals in SP1, the searching procedure is as follows:

Step 1: Randomly select several operations in all T_o operations of the n jobs.
Step 2: Exchange the machines assigned to the selected operators of the two individuals and obtain two new individuals.
Step 3: If the new individual is better than the old one, replace the old individual with the new one.

The above procedure is implemented between every two adjacent individuals in SP1, i.e. the first one with the second one, the third one with the fourth one, and so on.

4.5.2. Operator for SP2

The sub-population SP2 only adjusts the operation sequence. For the two individuals in SP2, the searching procedure is as follows:

Step 1: Randomly generate a subset of all jobs.
Step 2: Given individuals u and v, the new individual $u'(v')$ inherits the element of $u(v)$ at the position if the element belongs to the subset; otherwise, it inherits the elements of $v(u)$ that do not belong to the subset from left to right.
Step 3: Replace the old two individuals with the best two among u, v, u' and v'.

Table 2
Combinations of parameter values.

Parameters	Factor level			
	1	2	3	4
η	10	20	30	40
α	0.1	0.2	0.3	0.4
β	0.1	0.2	0.3	0.4
ter	10	30	50	70

An example is illustrated in Fig. 4 for the above procedure.

4.6. Local search based on critical path

It is widely accepted that a local search procedure is efficient in improving the solutions generated by the EDA. In this paper, a local search based on critical path is designed to enhance the local exploitation around the best solution obtained by the two sub-populations to solve the FJSP.

Denote $S^E_{i,j}$ as the earliest starting time of operation $O_{i,j}$ and $S^L_{i,j}$ as the latest starting time without delaying the makespan. Thus, the earliest completion time of operation $O_{i,j}$ is $C^E_{i,j} = S^E_{i,j} + t_{i,j,k}$, and the latest completion time is $C^L_{i,j} = S^L_{i,j} + t_{i,j,k}$, where $t_{i,j,k}$ is the processing time of $O_{i,j}$ on machine k. Let $PM^k_{i,j}$ be the operation processed on machine k right before the operation $O_{i,j}$ and $SM^k_{i,j}$ be the operation processed on machine k right after $O_{i,j}$. Let $PJ_{i,j} = O_{i,j-1}$ be the operation of job i that precedes $O_{i,j}$ and $SJ_{i,j} = O_{i,j+1}$ be the operation of job i that follows $O_{i,j}$.

Since the makespan is no shorter than any possible critical path, the makespan may be improved only by moving the critical operations. Let O_l ($l = 1, 2, \ldots, N_c$) be the critical operation to be moved, where N_c is the total number of critical operations of a solution. Moving O_l is to delete it from its current position and then insert it at another feasible position. Obviously, the makespan of the new solution is no larger than that of the old one. If O_l is assigned before $O_{i,j}$ on machine k, it can be started as early as $C^E(PM^k_{i,j})'$ and can be finished as late as $S^{L'}_{i,j}$ without delaying the required makespan. Besides, O_l cannot violate the precedence relations of the same job. Thus, the assignable idle time interval for O_l can be defined by $\max\{C^E(PM^k_{i,j})', C^E(PJ_l)'\} + t_{l,k} \leqslant \min\{S^{L'}_{i,j}, S^L(SJ_l)'\}$.

The above moving process is repeated until all the critical operations are moved. Let N_l be the number of positions to move O_l feasibly, then the total number of moving neighbors of a solution is

$$N_{total} = \sum_{l=1}^{N_c} N_l$$

Table 3
Orthogonal array and ARV values.

Experiment number	Factor				ARV
	η	α	β	*ter*	
1	1	1	1	1	64.5
2	1	2	2	2	64.2
3	1	3	3	3	64.65
4	1	4	4	4	64.85
5	2	1	2	3	64.95
6	2	2	1	4	64.9
7	2	3	4	1	65.75
8	2	4	3	2	65.45
9	3	1	3	4	66.45
10	3	2	4	3	66.05
11	3	3	1	2	63.9
12	3	4	2	1	64.65
13	4	1	4	2	66.25
14	4	2	3	1	65.5
15	4	3	2	4	64.25
16	4	4	1	3	64.55

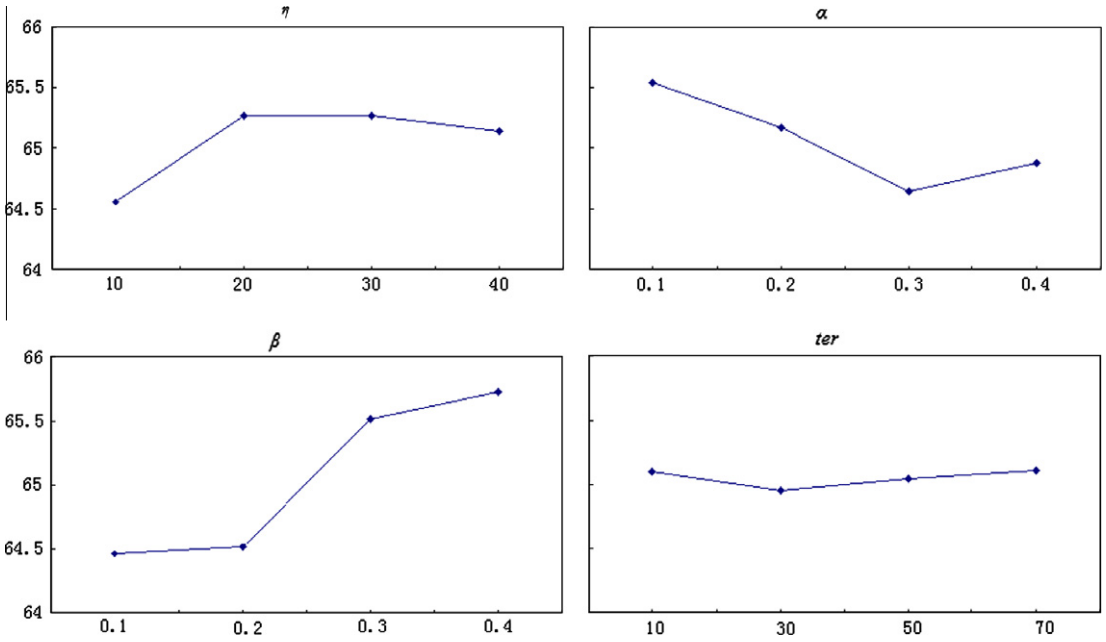

Fig. 6. Factor level trend of the BEDA.

Table 4
Response value.

Level	η	α	β	ter
1	64.55	65.5375	64.4625	65.1
2	65.2625	65.1625	64.5125	64.95
3	65.2625	64.6375	65.5125	65.05
4	65.1375	64.875	65.725	65.1125
Delta	0.7125	0.9	1.2625	0.1625
Rank	3	2	1	4

Table 5
Results of the five Kacem instances.

Instance	$n \times m$	PVNS	KBACO	TSPCB		BEDA			
				Best	AVG	Best	AVG	SD	TAVG(s)
Case 1	4×5	N/A	N/A	**11**	11.00	**11**	11.00	0.00	0.01
Case 2	8×8	**14**	**14**	**14**	14.20	**14**	14.00	0.00	0.23
Case 3	10×7	N/A	N/A	**11**	11.00	**11**	11.00	0.00	0.30
Case 4	10×10	**7**	**7**	**7**	7.10	**7**	7.00	0.00	0.42
Case 5	15×10	**11**	12	**11**	11.70	**11**	11.00	0.00	14.88

Note: the bold values mean the best results.

Moving critical operations will obtain new solutions. For the FJSP, there may be more than one critical path for a schedule. To improve a solution, all its critical paths should be changed. So, a solution with fewer critical paths is more likely to be improved efficiently. Thus, in our algorithm the new solution will replace the old one if one of the following conditions is satisfied: (1) The new solution has a smaller makespan than the old solution; (2) The new solution with the same makespan as the old one has fewer critical paths. Moreover, to take advantage of the move of critical operations, we propose the following local search procedure based on moving operations for the best individual of the population in each generation.

Step 1: Generate s' by moving all critical operations of solution s, then let $s' = s$.
Step 2: If s' replaces s successfully, then go back to Step 1; else, stop.

4.7. Procedure of the BEDA

With the above design, the procedure of the BEDA to solve the FJSP is illustrated in Fig. 5.

It can be seen that the BEDA contains two main phases in every generation. At the global exploration phase, a probability model is built with the superior individuals of the entire population to generate the new individuals like the classic EDA. At the local exploitation phase, two sub-populations apply different operators to generate neighbor individuals and the best solution uses the critical path based local search for further exploitation. The splitting criterion and the combination criterion are used for switching between two different phases. The algorithm stops when the maximum number of generations *Gen* is reached.

4.8. Computation complexity analysis

To update the probability matrix A_1 and matrix A_2, sort P individuals in an ascending order is with the computational complexity $O(P\log P)$ by using the quick sorting method. Then, update all the $T_o \times n$ elements of A_1 by the operator sequence vectors of the best SI individuals is with the complexity $O(T_o \times SI + T_o \times n)$. Similarly, the expected computational complexity to update A_2 is $O(T_o \times SI + T_o \times m)$. So, the computational complexity for updating the probability modal is $O[T_o(2SI + m + n) + P\log P]$.

To obtain a new individual by sampling the probability model, every gene is generated with the roulette strategy according to the matrix A_1 and A_2. An operation sequence vector is constructed with the complexity $O(T_o \times n)$, while a machine assignment vector is constructed with the complexity $O(T_o \times m)$. So, P individuals can be generated with the complexity $O[PT_o(m + n)]$.

In addition, each kind of operator for adjusting the individuals of two sub-populations can be done with the complexity $O(T_o)$. Considering the size of sub-population, the total complexity of the adjustment is $O(PT_o)$.

Totally, the complexity of the BEDA is not large. It can be seen from the numerical simulation in the next section that the BEDA is very efficient in solving the FJSP.

Table 6
Results of 10 BRdata instances.

Instance	$n \times m$	PVNS			KBACO			TSPCB			BEDA			
		Best	AVG	SD	Best	AVG	SD	Best	AVG	SD	Best	AVG	SD	TAVG(s)
Mk1	10 × 6	40	40.0	N/A	**39**	39.8	0.43	40	40.30	N/A	40	41.02	0.83	1.09
Mk2	10 × 6	**26**	26.4	N/A	29	29.1	0.28	**26**	26.50	N/A	**26**	27.25	0.67	2.16
Mk3	15 × 8	**204**	204.0	N/A	**204**	204	0.00	**204**	204.0	N/A	**204**	204.00	0	2.18
Mk4	15 × 8	**60**	60.6	N/A	65	66.1	1.06	62	64.88	N/A	**60**	63.69	1.99	9.02
Mk5	15 × 4	173	173.0	N/A	173	173.8	1.08	**172**	172.9	N/A	**172**	173.38	0.56	7.10
Mk6	10 × 15	**60**	61.0	N/A	67	69.1	1.03	65	67.38	N/A	**60**	62.83	1.06	30.21
Mk7	20 × 5	141	141.2	N/A	144	145.4	1.42	140	142.21	N/A	**139**	141.55	1.07	17.07
Mk8	20 × 10	**523**	523	N/A	**523**	523	0	**523**	523.00	N/A	**523**	523.00	0	4.30
Mk9	20 × 10	**307**	308.8	N/A	311	312.2	1.81	310	311.29	N/A	**307**	310.35	0.96	91.99
Mk10	20 × 15	208	209.4	N/A	229	233.7	3.27	214	219.15	N/A	**206**	211.92	2.59	190.11

Note: the bold values mean the best results.

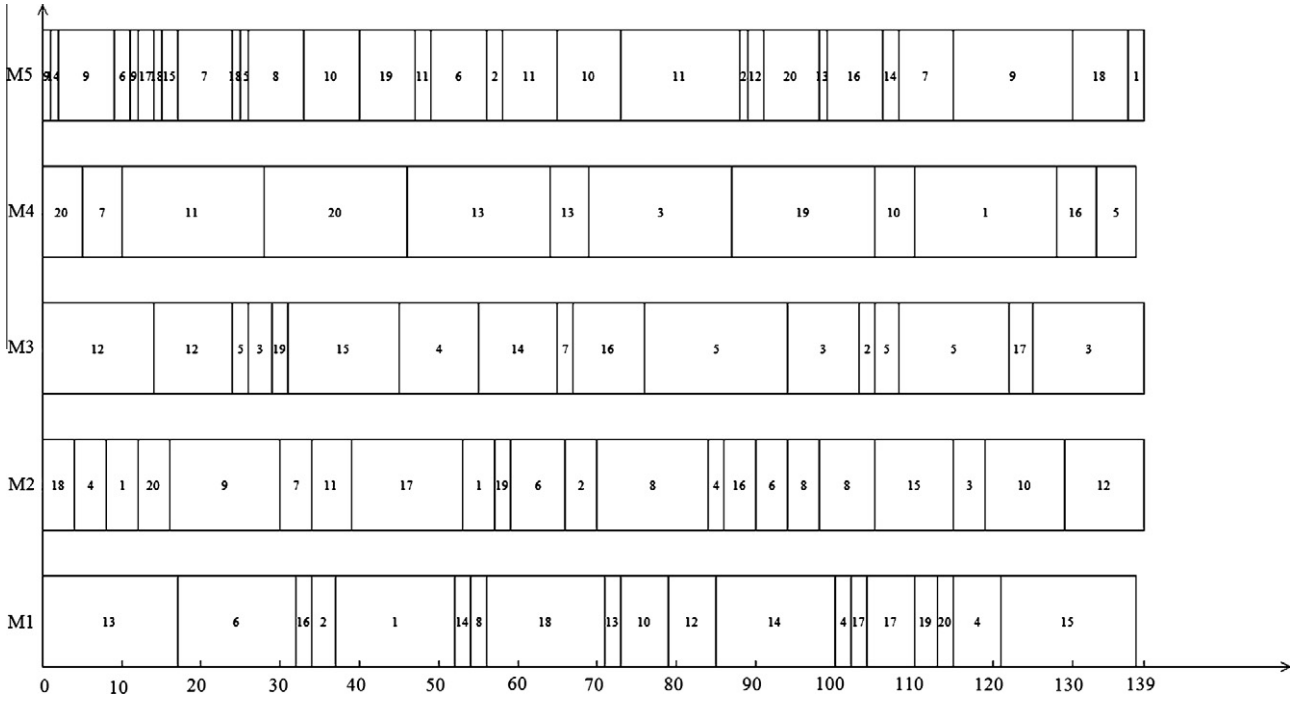

Fig. 7. The best solution of Mk7 obtained by BEDA (makespan = 139).

5. Computational results and comparisons

To test the performance of the proposed BEDA, numerical simulations are carried out with two sets of widely used benchmarks including five Kacem instances (Kacem et al., 2002) and 10 BRdata instances (Brandimarte, 1993). The algorithm is coded in C++ and run on a 3.2 GHz Intel Core i5 processor.

5.1. Parameters setting

It is generally accepted that the difficulty to solve the FJSP is closely associated with the problem size. Therefore, for each instance, the maximum number of generations is set as $Gen = 10 \cdot n \times m$ and the population size is set as $P = n \times m$. The number of selected individuals to update the probability model is set to $SI = \eta\% \cdot P$. Besides, the BEDA contains several other key parameters: the learning rate of A_1, i.e., α, the learning rate of A_2, i.e., β, and the parameter *ter* in the splitting and combination criteria. To investigate the influence of these parameters on the performance of the algorithm, we implement the Taguchi method of design of experiment (DOE) (Montgomery, 2005) by using a moderate-scale instance Mk4. Combinations of different values of these parameters are listed in Table 2.

For each parameter combination, the BEDA is run 20 times independently and the average response variable (ARV) value is the average of makespan value obtained by the BEDA. According to the number of parameters and the number of factor levels, we choose the orthogonal array $L_{16}(4^4)$. That is, the total number of treatment is 16, the number of parameters is 4, and the number of factor levels is 4. The orthogonal array and the obtained ARV values are listed in Table 3.

According to the orthogonal table, we illustrate the trend of each factor level in Fig. 6. Then, we figure out the response value of each parameter to analyze the significance rank of each parameter. The results are listed in Table 4.

From Table 4 it can be seen that the learning rate β of A_2 is the most significant parameter among the four parameters. That is, the learning rate of the matrix for machine assignment is crucial to the BEDA. Large value of β could lead to premature convergence. In addition, the significant rank of the learning rate α of A_1 is the second. Small value of α could lead to slow convergence while large value could lead to premature convergence. Besides, the number

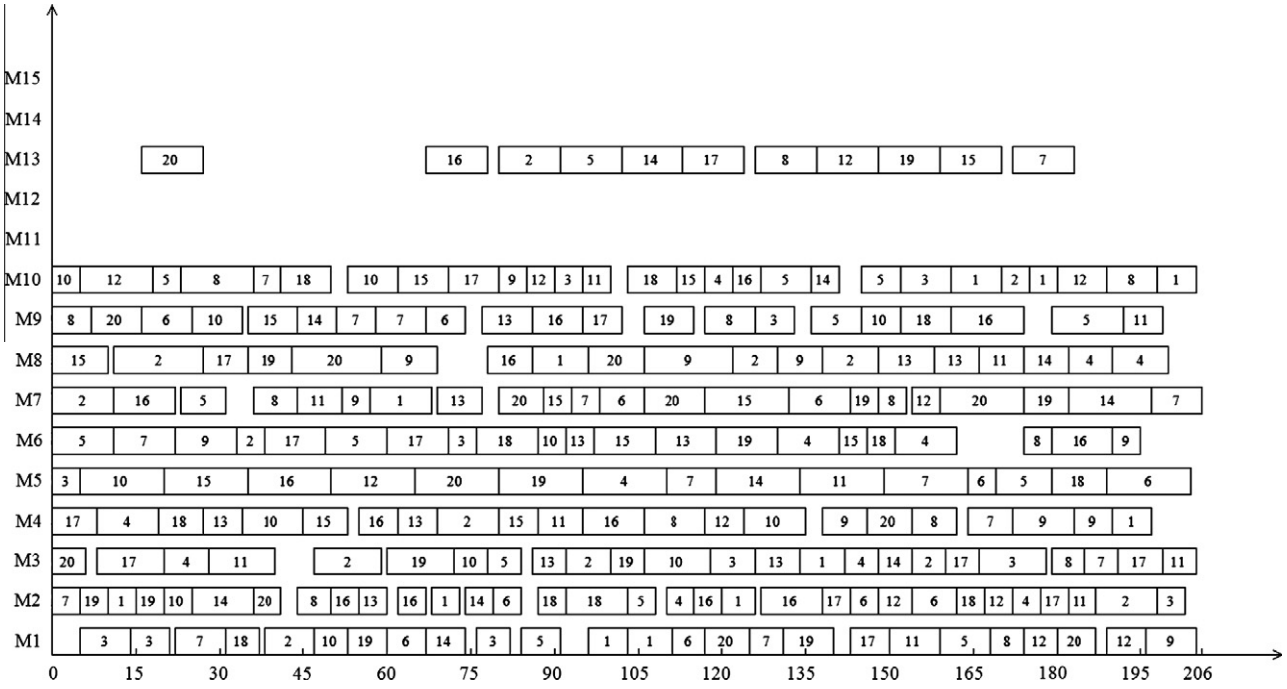

Fig. 8. The best solution of Mk10 obtained by BEDA (makespan = 206).

Table 7
Comparisons of the bi-population method with single-population method.

Instance	$n \times m$	EDA			BEDA		
		Best	AVG	SD	Best	AVG	SD
Case 1	4 × 5	**11**	**11.00**	0	**11**	**11.00**	0.00
Case 2	8 × 8	**14**	14.34	0.51	**14**	**14.00**	0.00
Case 3	10 × 7	**11**	11.06	0.24	**11**	**11.00**	0.00
Case 4	10 × 10	**7**	7.16	0.37	**7**	**7.00**	0.00
Case 5	15 × 10	**11**	11.58	0.49	**11**	**11.00**	0.00
Mk1	10 × 6	**40**	41.18	0.74	**40**	**41.02**	0.83
Mk2	10 × 6	27	27.72	0.49	**26**	**27.25**	0.67
Mk3	15 × 8	**204**	**204.00**	0	**204**	**204.00**	0
Mk4	15 × 8	62	63.86	0.98	**60**	**63.69**	1.99
Mk5	15 × 4	173	173.52	0.57	**172**	**173.38**	0.56
Mk6	10 × 15	63	64.42	0.78	**60**	**62.83**	1.06
Mk7	20 × 5	141	142.00	1.06	**139**	**141.55**	1.07
Mk8	20 × 10	**523**	**523.00**	0	**523**	**523.00**	0
Mk9	20 × 10	309	310.86	0.45	**307**	**310.35**	0.96
Mk10	20 × 15	218	220.79	1.68	**206**	**211.92**	2.59

Note: the bold values mean the best results.

of selected individuals *SI* to update the probability model ranks the third. A small value can help the algorithm build an accurate model. Although the parameter *ter* in the splitting criterion and combination criterion has the slightest influence, but an appropriate value still can help the BEDA balance the exploration and exploitation well. Splitting and recombining the population too frequently may affect the smooth evolution. On the contrary, a large value of *ter* cannot help the population maintain diversity. According to the above analysis, a good choice of parameter combination is suggested as $P = n \times m$, $SI = 10\%P$, $\alpha = 0.3$, $\beta = 0.1$, $ter = 30$ and $Gen = 10 \cdot n \times m$.

5.2. Results of Kacem instances

Consider the first five Kacem instances with the scale ranging form 4 jobs 5 machines to 15 jobs 10 machines, we compare the BEDA with several algorithms including PVNS (Yazdani, Amiri, & Zandieh, 2010), KBACO (Xing et al., 2010) and TSPCB (Li et al., 2011). For each instance we run the BEDA 50 times independently. The results are listed in Table 5. We also give the average results (AVG), the standard derivation (SD), and average CPU time (TAVG) of the BEDA. The results of the comparative algorithms are directly from literatures where some AVG and SD values are unavailable.

From Table 5, it can be seen that the BEDA is the best one among the four algorithms for solving the five Kacem instances. Especially, the BEDA can consistently obtain the optimal values for all the five instances.

5.3. Results of BRdata instances

Next, we carry out tests with BRdata instances. The BRdata instances include 10 problems from 10 jobs 6 machines to 20 jobs 15 machines. For each instance we run the BEDA 50 times independently. Table 6 illustrates the comparison results with PVNS, KBACO and TSPCB for solving the BRdata instances.

Form Table 6, it can be seen that the BEDA outperforms other algorithms in solving almost all the instances in terms of the best results, average results and standard derivation. In particular, as for the best results, the BEDA outperforms PVNS in 4 out of 10 instances, outperforms KBACO in 7 out of 10 instances, and outperforms TSPCB in 5 out of 10 instances. So, it is concluded that the BEDA is more powerful in solving the FJSP. Moreover, the SD values of the BEDA are relatively small, from which it is concluded that BEDA is very robust. As for the efficiency, it can be seen that the average running time of the BEDA is acceptable, even for relatively larger-scale instances.

The Gantt charts for the best solutions of Mk7 and Mk10 obtained by the BEDA are illustrated in Figs. 7 and 8, respectively. Among all algorithms, only the BEDA can obtain these two best solutions.

5.4. Comparison between the bi-population method with single-population method

To demonstrate the effectiveness of the bi-population method, we compare the BEDA with the single-population based EDA by

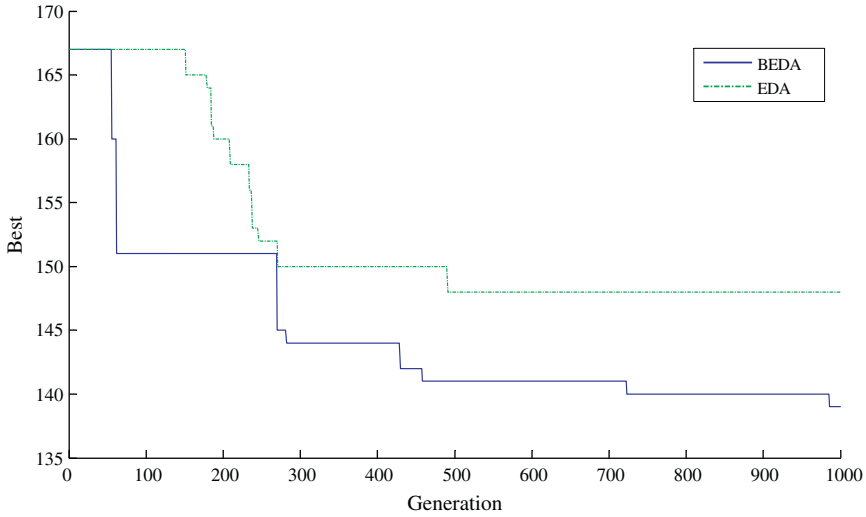

Fig. 9. Convergence curves in solving MK7 by BEDA and simple EDA.

using the five Kacem instances and the 10 BRdata instances. The single-population based EDA has the same probability model and updating mechanism as the BEDA, but it uses an entire population for evolution throughout the whole process. To be fair, the local search procedure is also used in the single-population based EDA, and all the parameters the same in the two algorithms. We also run the single-population based EDA 50 times independently. The comparative results are listed in Table 7.

From Table 7, it can be seen that the best makespan values obtained by the BEDA are better than those of the single-population based EDA in 7 out of all the 15 instances, while the results of the other 8 are the same. As for the average results in 50 times run, the BEDA are better than the single-population based EDA in 12 out of all the 15 instances, while the results of the other 3 are the same. So, the BEDA is more effective than the single-population based EDA in solving the FJSP, especially for the large-scaled problems.

In addition, Fig. 9 depicts the typical convergence curves of the best makespan values in the population obtained by the BEDA and the single-population based EDA when solving Mk7 instance. The two algorithms evolve with the same initial population. From Fig. 9, it can be seen that the bi-population based method is of faster convergence speed than the single-population based method. Besides, the BEDA is more powerful to avoid premature convergence than the single-population based method.

All in all, according to the above simulation tests and the comparisons it is shown that the proposed BEDA is effective, efficient and robust in solving the FJSP.

6. Conclusions

In this paper, an effective bi-population based estimation of distribution algorithm was proposed for solving the flexible job-shop scheduling problem with the criterion to minimize the makespan. We used two sub-populations to adjust the machine assignment and operation sequence respectively with a splitting criterion and a combination to balance the global exploration and local exploitation. For global exploration, we designed a probability model with the superior population to generate the new individuals via sampling based on the probability model. For local exploitation, we designed two operators to adjust the individuals of sub-populations in a separated way. We also designed an updating mechanism for the probability model and used multiple strategies in a combination way to generate the initial solutions. Besides, the critical path based local search was used to enhance the exploitation. The influence of parameter setting was investigated by using DOE based testing. Simulation tests and comparisons to several existing algorithms demonstrated the effectiveness of the proposed BEDA in solving the FJSP. The future work is to design EDA based algorithm for the multi-objective FJSP and the flexible flow shop scheduling problem.

Acknowledgments

This research is partially supported by National Science Foundation of China (61174189, 60834004 and 61025018), Doctoral Program Foundation of Institutions of Higher Education of China (20100002110014), the National Key Basic Research and Development Program of China (No. 2009CB320602) and National Science and Technology Major Project of China (No. 2011ZX02504-008).

References

Baluja, S. (1994). Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning. Technical report CMU-CS-94-163. Pittsburgh, PA: Carnegie Mellon University.

Baluja, S., & Davies, S. (1997). Using optimal dependency-trees for combinatorial optimization: Learning the structure of the search space. In *Proc. of the 14th int. conf. on machine learning, San Francisco* (pp. 30–38).

Brandimarte, P. (1993). Routing and scheduling in a flexible job shop by tabu search. *Annals of Operations Research, 41*, 157–183.

Bruker, P., & Schlie, R. (1990). Job-shop scheduling with multi-purpose machines. *Computing, 45*, 369–375.

Chen, S. H., Chen, M. C., Chang, P. C., Zhang, Q. F., & Chen, Y. M. (2010). Guidelines for developing effective estimation of distribution algorithms in solving single machine scheduling problems. *Expert Systems with Applications, 37*, 6441–6451.

Dauzere-Peres, S., & Paulli, J. (1997). An integrated approach for modeling and solving the general multiprocessor job-shop scheduling problem using tabu search. *Annals of Operations Research, 70*, 281–306.

De Bonet, J. S., Isbell, C. L., Jr., & Viola, P. (1997). *MIMIC: Finding optima by estimating probability densities. Advances in neural information processing systems.* Cambridge: MIT Press.

Gao, J., Sun, L. Y., & Gen, M. (2008). A hybrid genetic and variable neighborhood descent algorithm for flexible job shop scheduling problems. *Computer and Operations Research, 35*, 2892–2907.

Harik, G. (1999). Linkage learning via probabilistic modeling in the ECGA. Illigal report NO. 99010. Illinois Genetic Algorithms Laboratory. Illinois: University of Illinois at Urbana-Champaign.

Harik, G. R., Lobo, F. G., & Goldberg, D. E. (1998). The compact genetic algorithm. In *Proc. of the IEEE conf. on evolutionary computation, Indianapolis* (pp. 523–528).

Jarboui, B., Eddaly, M., & Siarry, P. (2009). An estimation of distribution algorithm for minimizing the total flowtime in permutation flowshop scheduling problems. *Computers and Operations Research, 36*, 2638–2646.

Kacem, I., Hammadi, S., & Borne, P. (2002). Pareto-optimality approach for flexible job-shop scheduling problems: hybridization of evolutionary algorithms and fuzzy logic. *Mathematics and Computers in Simulation, 60*, 245–276.

Larranaga, P., & Lozano, J. A. (2002). *Estimation of distribution algorithms: A new tool for evolutionary computation*. Netherlands: Springer.

Lawler, E. L., Lenstra, J. K., Rinnooy Kan, A. H. G., & Shmoys, D. B. (1993). Sequencing and scheduling: Algorithms and complexity. In *Logistics of production and inventory, Amsterdam* (pp. 445–522).

Li, J. Q., Pan, Q. K., & Liang, Y. C. (2010). An effective hybrid tabu search algorithm for multi-objective flexible job-shop scheduling problems. *Computers and Industrial Engineering, 59*, 647–662.

Li, J. Q., Pan, Q. K., Suganthan, P. N., & Chua, T. J. (2011). A hybrid tabu search algorithm with an efficient neighborhood structure for the flexible job shop scheduling problem. *International Journal of Advanced Manufacturing Technology, 52*, 683–697.

Mastrolilli, M., & Gambardella, L. M. (2000). Effective neighborhood functions for the flexible job shop problem. *Journal of Scheduling, 3*, 3–20.

MÄuhlenbein, H., & Paass, G. (1996). From recombination of genes to the estimation of distributions I: Binary parameters. *Lecture Notes in Computer Science, 1141*, 178–187.

Montgomery, D. C. (2005). *Design and analysis of experiments*. Arizona: John Wiley & Sons.

Mühlenbein, H., & Mahnig, T. (1999). Convergence theory and applications of the factorized distribution algorithm. *Journal of Computing and Information Technology, 7*, 19–32.

Pelikan, M., Goldberg, D. E., & Cantú-Paz, E. (1999). BOA: The bayesian optimization algorithm. In *Proc. of the genetic and evolutionary computation, San Francisco* (pp. 525–532).

Pelikan, M., & Mühlenbein, H. (1999). The bivariate marginal distribution algorithm. In J. M. Benítez (Ed.), *Advances in soft computing: engineering design and manufacturing*. London: Springer-Verlag.

Pezzella, F., Morganti, G., & Ciaschetti, G. (2008). A genetic algorithm for the flexible job-shop scheduling problem. *Computers and Operations Research, 35*, 3202–3212.

Tay, J. C., & Ho, N. B. (2008). Evolving dispatching rules using genetic programming for solving multi-objective flexible job-shop problems. *Computers and Industrial Engineering, 54*, 453–473.

Wang, L., & Fang, C. (2012). An effective estimation of distribution algorithm for the multi-mode resource-constrained project scheduling problem. *Computers and Operations Research, 39*, 449–460.

Xia, W. J., & Wu, Z. M. (2005). An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems. *Computers and Industrial Engineering, 48*, 409–425.

Xing, L. N., Chen, Y. W., Wang, P., Zhao, Q. S., & Xiong, J. (2010). A knowledge-based ant colony optimization for flexible job shop scheduling problems. *Applied Soft Computing, 10*, 888–896.

Xing, L. N., Chen, Y. W., & Yang, K. W. (2009). An efficient search method for multi-objective flexible job shop scheduling problems. *Journal of Intelligent Manufacturing, 20*, 283–293.

Yazdani, M., Amiri, M., & Zandieh, M. (2010). Flexible job-shop scheduling with parallel variable neighborhood search algorithm. *Expert Systems with Applications, 37*, 678–687.

Zhou, S. D., & Sun, Z. Q. (2007). A survey on estimation of distribution algorithms. *Acta Automatica Sinica, 33*, 113–124.