International Journal of Production Research, 2013
Vol. 51, No. 12, 3778–3793, http://dx.doi.org/10.1080/00207543.2013.765077

An effective estimation of distribution algorithm for the flexible job-shop scheduling problem with fuzzy processing time

Shengyao Wang*, Ling Wang, Ye Xu and Min Liu

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

(*Received 14 April 2012; final version received 25 December 2012*)

Considering the fuzzy nature of the data in real-world scheduling, an effective estimation of distribution algorithm (EDA) is proposed to solve the flexible job-shop scheduling problem with fuzzy processing time. A probability model is presented to describe the probability distribution of the solution space. A mechanism is provided to update the probability model with the elite individuals. By sampling the probability model, new individuals can be generated among the search region with promising solutions. Moreover, a left-shift scheme is employed for improving schedule solution when idle time exists on the machine. In addition, some fuzzy number operations are used to calculate scheduling objective value. The influence of parameter setting is investigated based on the Taguchi method of design of experiment, and a suitable parameter setting is suggested. Numerical testing results and comparisons with some existing algorithms are provided, which demonstrate the effectiveness of the proposed EDA.

Keywords: flexible job-shop scheduling problem; fuzzy processing time; fuzzy completion time; estimation of distribution algorithm; probability model; design of experiment

Notation

n: the number of the jobs to be processed;
m: the number of the total machines;
$J = \{J_1, J_2, \ldots, J_n\}$: the set of n jobs to be processed;
$M = \{M_1, M_2, \ldots, M_m\}$: the set of m machines;
$O_{i,j}$: the jth operation of J_i;
n_i: the number of the operations of J_i;
$m_{i,j}$: the number of the capable machines for $O_{i,j}$;
T_O: the number of total operations of all the n jobs;
$t_{i,j,k} = (t^1_{i,j,k}, t^2_{i,j,k}, t^3_{i,j,k})$: the fuzzy processing time of $O_{i,j}$ on machine M_k;
$C_{i,j} = (C^1_{i,j}, C^2_{i,j}, C^3_{i,j})$: the fuzzy completion time of $O_{i,j}$;
$C_{\max}$: the maximum fuzzy completion time of a schedule;
$S_{i,j}$: the fuzzy starting time of $O_{i,j}$;
t^S_k: the beginning time of a time interval;
t^E_k: the ending time of a time interval;
P: the population size;
SP: the size of the superior population;
A_1: the operation probability matrix;
A_2: the machine probability matrix;
$p_{ij}(l)$: the element of A_1 at generation l;
$q_{ijk}(l)$: the element of A_2 at generation l;
α: the learning rate of A_1;
β: the learning rate of A_2;
η: the percentage of the superior population from the whole population.

*Corresponding author. Email: wangshengyao@tsinghua.org.cn

1. Introduction

The flexible job-shop scheduling problem (FJSP) is an extension of the classical job-shop scheduling problem (JSP) for the flexible manufacturing systems, which is very close to the real manufacturing situation. In the FJSP, some machines can perform 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. The FJSP is more difficult to solve than the classical JSP since it needs to determine the assignment of operations to the related machines.

The first work to address the FJSP was by Bruker and Schlie (1990), where a polynomial algorithm was presented for the problem with only two jobs and identical machines. Later, Brandimarte (1993) presented a hybrid tabu search (TS) algorithm with some existing dispatching rules. Dauzere-Peres and Paulli (1997) also presented a TS algorithm based on an integrated approach, which was improved by two developed neighbourhood functions (Mastrolilli and Gambardella 2000) in terms of computational time and solution quality. Gao, Sun, and Gen (2008) designed a genetic algorithm (GA) hybridising with variable neighbourhood search (VNS), and Pezzella, Morganti, and Ciaschetti (2008) developed a GA integrating different strategies. Besides, Amiri et al. (2010) developed a VNS algorithm based on six neighbourhood structures, Yazdani, Amiri, and Zandieh (2010) developed a parallel VNS algorithm, and Xing et al. (2010) developed a knowledge-based ant colony optimisation algorithm. Recently, a bi-population based estimation of distribution algorithm (BEDA) (Wang et al. 2012c) and a novel artificial bee colony (ABC) algorithm (Wang et al. 2012d) were presented to solve the FJSP. As for the multi-objective flexible job-shop scheduling (MFJSP), relevant algorithms include heuristic algorithm (Wang, Zhou, and Xi 2008), GA (Frutos, Olivera, and Tohme 2010; Wang et al. 2010), genetic programming (Tay and Ho 2008), TS algorithm (Li, Pan, and Liang 2010), particle swarm optimisation (PSO) (Xia and Wu 2005; Moslehi and Mahnam 2011), and ABC algorithm (Li, Pan, and Gao 2011; Wang et al. 2012a).

In the literature, it often assumes that the processing time and completion time are deterministic values. In real-world scheduling, however, the processing time of an operation cannot be known precisely and the completion time can be obtained ambiguously. Thus, it may be more appropriate to consider fuzzy processing time and fuzzy completion time (Sakawa and Kubota 2000). The fuzzy job-shop scheduling problem (fJSP) extends the JSP by considering the processing time or the due-date to be fuzzy variable. Relevant results include following. Sakawa and Mori (1999) presented a GA to maximise the minimum agreement index. Then Sakawa and Kubota (2000) improved the GA (Sakawa and Mori 1999) to maximise the minimum agreement index, to maximise the average agreement index, and to minimise the maximum fuzzy completion time. Lei (2008) designed a Pareto archive PSO for the multi-objective fJSP to obtain a set of Pareto optimal solutions. González-Rodríguez, Vela, and Puente (2010) presented a GA to search possible active schedules. Recently, Hu, Yin, and Li (2011) presented a modified differential evolution (DE) algorithm for the fJSP, and Lei (2010b, 2011) presented a random key GA and a swarm-based neighbourhood search with availability constraints.

Introducing fuzzy constraints to the FJSP makes the problem much closer to the real applications, such as the textile industry, automobile assembly and semiconductor manufacturing. The FJSP with fuzzy processing time is regarded as the combination of the FJSP and the fJSP, namely fuzzy flexible job-shop scheduling problem (fFJSP). Lei (2010a) presented an efficient decomposition-integration genetic algorithm (DIGA), in which the main population was decomposed as two sub-populations that evolved independently and the new main population was formed by sub-population integration. Recently, Lei (2012) developed an effective co-evolutionary genetic algorithm (CGA) to minimise the fuzzy makespan. The study on this topic is still in its infancy, and it is important to develop novel algorithms for the problem.

As a novel kind of evolutionary algorithm based on statistical learning, estimation of distribution algorithm (EDA) (Larranaga and Lozano 2002) has gained increasing studies and wide applications during recent years. Due to different kinds of the relationships among variables, the EDA has different models. Accordingly, it can be classified as univariate model, bivariate model or multivariate model. Univariate model assumes that the variables are independent with each other, e.g. the population-based incremental learning (Baluja 1994), the univariate marginal distribution algorithm (Mäuhlenbein and Paass 1996) and the compact GA (Harik, Lobo, and Goldberg 1998). Bivariate model assumes that each variable is associated with another one, e.g. the mutual information maximisation for input clustering (De Bonet, Isbell, and Viola 1997), the combining optimisers with mutual information trees (Baluja and Davies 1997) and the bivariate marginal distribution algorithm (Pelikan and Mühlenbein 1999). Multivariate model considers the relationship between all the variables, e.g. the factorised distribution algorithms (Mühlenbein and Mahnig 1999), the extended compact GA (Harik 1999) and the Bayesian optimisation algorithm (Pelikan, Goldberg, and Cantú-Paz 1999). For more details about the EDA, please refer Larranaga and Lozano (2002).

The EDA has been applied to a variety of academic and application problems, such as feature selection (Saeys et al. 2003), inexact graph matching (Cesar et al. 2005), software testing (Sagarna and Lozano 2005), flow-shop scheduling

(Jarboui, Eddaly, and Siarry 2009), resource-constrained project scheduling (Wang and Fang 2012), multi-dimensional knapsack problem (Wang, Wang, and Fang 2012b), and so on. However, to the best of our knowledge, there is no research work about the EDA for solving the fFJSP. Inspired by the successful applications of the EDA, we shall propose an effective EDA to enrich the tool-kit for solving the fFJSP. To be specific, we build a probability model to generate population and provide a mechanism to update the model, and we employ a left-shift scheme to improve solutions. The influence of parameter setting is investigated based on the design of experiment. Numerical testing results and comparisons are provided to show the effectiveness of the proposed EDA.

The remainder of the paper is organised as follows: In Section 2, the fFJSP is described, and some operations on fuzzy processing time are introduced. In Section 3, the basic EDA is described briefly. Then, in Section 4 the EDA for the fFJSP is proposed in details. The influence of parameter setting is investigated and computational results and comparisons are provided in Section 5. Finally we end the paper with some conclusions and future work in Section 6.

2. Flexible job-shop scheduling problem with fuzzy processing time

2.1 *Problem description*

The FJSP with fuzzy processing time (fFJSP) is commonly defined as follows. There are a set of n jobs $J = \{J_1, J_2, \ldots, J_n\}$ to be processed on m machines $M = \{M_1, M_2, \ldots, M_m\}$. Each job J_i consists of a sequence of n_i operations $\{O_{i,j}, j = 1, 2, \ldots, n_i\}$. Each routing has to be determined to complete a job. The execution of $O_{i,j}$ requires one machine out of a set of $m_{i,j}$ capable machines out of M. The processing time of $O_{i,j}$ on machine M_k is represented as a triangular fuzzy number (TFN) $t_{i,j,k} = (t^1_{i,j,k}, t^2_{i,j,k}, t^3_{i,j,k})$, where $t^1_{i,j,k}$ is the best processing time, $t^2_{i,j,k}$ is the most probable processing time and $t^3_{i,j,k}$ is the worst processing time. Similarly, the fuzzy completion time of $O_{i,j}$ is represented as a TFN $C_{i,j} = (C^1_{i,j}, C^2_{i,j}, C^3_{i,j})$, where $C^1_{i,j}$ is the best completion time, $C^2_{i,j}$ is the most probable completion time and $C^3_{i,j}$ is the worst completion time.

Typically, it assumes that:

- All the jobs are independent and available to be processed at the initial time.
- One machine can process only one operation and one job can be processed on only one machine at a time.
- The releasing time of all machines is not considered or set as 0.
- The transfer time between different machines is included in the processing time and the buffer size is infinite.
- Once an operation is started, it cannot be interrupted (Lei 2010a; 2012).

The fFJSP is to determine both the assignment of machines and the sequence of operations on all the machines to minimise the maximum fuzzy completion time, $C_{\max} = \max_{i=1,2,\ldots n} C_{i,j}$.

.

2.2 *Operations on fuzzy processing time*

In fuzzy context, it requires to define certain fuzzy number operations (Bortolan and Degani 1985) to produce a schedule. These operations include addition operation, max operation of two fuzzy numbers and the ranking method of fuzzy numbers. Addition operation is used to calculate the fuzzy completion time of operation. Max operation is used to determine the fuzzy beginning time of operation and the ranking method is to compare the maximum fuzzy completion time.

For two TFNs $X = (x_1, x_2, x_3)$ and $Y = (y_1, y_2, y_3)$, their addition is shown by the following formula: $X + Y = (x_1 + y_1, x_2 + y_2, x_3 + y_3)$.

In this paper, the following criterion (Sakawa and Kubota 2000) is adopted to rank fuzzy numbers to build a feasible schedule.

Step 1: The greatest number $Z_1(X) = \dfrac{x_1 + 2x_2 + x_3}{4}$ will be chosen as the first criterion to rank two TFNs.
Step 2: If two TFNs have the same Z_1, $Z_2(X) = x_2$ is used as a second criterion.
Step 3: If Z_1 and Z_2 are identical, $Z_3(X) = x_3 - x_1$ will be chosen as a third criterion.

Compared with the criterion by Sakawa and Mori (1999), this criterion can perform better on fuzzy scheduling (Lei 2010b). Thus, in this paper, the max of two TFNs X and Y is approximated by the criterion (Lei 2010a, 2012). That is, if $X > Y$, then $X \vee Y = X$; else $X \vee Y = Y$.

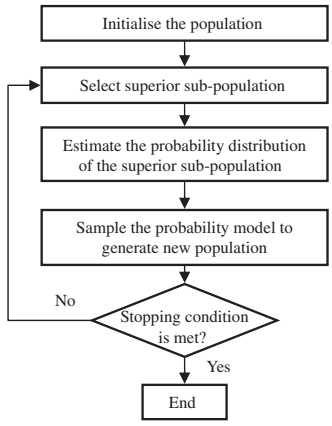

Figure 1. The general flowchart of the EDA.

Table 1. Processing time table.

	Machines		
Operations	M_1	M_2	M_3
$O_{1,1}$	(6,7,10)	(8,10,11)	(9,11,14)
$O_{1,2}$	(1,2,4)	(6,8,9)	∞
$O_{2,1}$	(6,7,9)	(5,6,8)	(7,10,12)
$O_{2,2}$	(7,9,11)	(5,9,12)	(4,7,9)
$O_{3,1}$	(18,21,24)	∞	(16,19,22)
$O_{3,2}$	∞	(10,14,17)	(7,10,11)
$O_{3,3}$	(7,10,13)	(4,6,9)	(4,5,7)

3. Estimation of distribution algorithm

Estimation of distribution algorithm (EDA) is a relatively new paradigm in the field of evolutionary algorithms, which employs explicit probability distributions in optimisation (Larranaga and Lozano 2002). Different from the GA that reproduces a new population with the crossover and mutation operators, the EDA does it implicitly. With the tool of statistical analysis, the EDA tries to estimate the underlying probability distribution of the potential individuals and builds a probability model of the most promising area by statistical information based on the search experience. The probability model is used for sampling to generate the new individuals and is updated in each generation with the elite individuals of the new population. In such an iterative way, the population evolves, and finally satisfactory solutions can be obtained.

The general flowchart of the EDA is illustrated in Figure 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 probability model built by the superior individuals helps the algorithm focus on the region with promising solutions. The updating process reflects the evolution trend of the population. Therefore, a proper probability model and a suitable updating mechanism are crucial to the EDA and should be well developed.

4. The EDA for fFJSP

4.1 *Solution representation*

Every individual of the population denotes a solution of the fFJSP, which is a combination of operation scheduling decision and machine assignment. Thus, a solution can be expressed by the processing sequence of operations and the assignment of operations on the machines. In this paper, a solution consists of two vectors corresponding to the two sub-problems of the fFJSP, i.e. operation sequence vector and machine assignment vector.

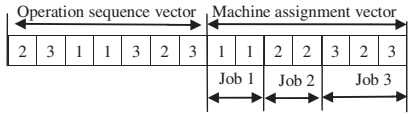

Figure 2. Illustration of the representation of a feasible solution.

For the operation sequence vector, the number of elements equals to the number of total 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 the elements is also T_O.

Given the operation sequence vector and machine assignment vector for a solution, the processing order on each machine is obtained. With the max operation on TFNs introduced in Section 2.2, the fuzzy starting time of $O_{i,j}$ can be calculated. With the addition operation on TFNs, the fuzzy completion time of $O_{i,j}$ can be calculated based on its fuzzy starting time and the fuzzy processing time. To further explain the representation, we provide an example by considering a problem with three jobs and three machines as shown in Table 1.

In Figure 2, it illustrates the representation of a feasible solution. The representation has the following operation sequence and machine assignment: ($O_{2,1}$, M_2), ($O_{3,1}$, M_3), ($O_{1,1}$, M_1), ($O_{1,2}$, M_1), ($O_{3,2}$, M_2), ($O_{2,2}$, M_2), ($O_{3,3}$, M_3). Thus, the processing orders on each machine are as follows: M_1: $O_{1,1} - O_{1,2}$; M_2: $O_{2,1} - O_{3,2} - O_{2,2}$; M_3: $O_{3,1} - O_{3,3}$. Then, the fuzzy completion time can be obtained.

The fuzzy Gantt chart (Lei 2010a, 2010b, 2012) of this solution is shown in Figure 3. The TFN under the line is the fuzzy beginning time of the operation and the TFN on the line is the fuzzy completion time of the operation.

4.2 *Left-shift scheme*

Due to the precedence constraint among operations of the same job, idle time may occur between the consecutive operations on the machine. For the schedule generated in Section 4.1, idle time interval exists between $O_{2,1} - O_{3,2}$ as shown in Figure 3. Considering the criterion of makespan, the left-shift scheme is employed to improve the schedule by shifting the operations to the left as compact as possible. Given a time interval $[t_k^S, t_k^E]$ beginning from t_k^S and ending at t_k^E on the machine M_k to allocate operation $O_{i,j}$, $O_{i,j}$ can be started only after its immediate job predecessor $O_{i,j-1}$ is completed. So, the starting time of $O_{i,j}$ can be described as follow. If the operation has no predecessor, $C_{i,j-1}$ is replaced with (0, 0, 0).

$$S_{i,j} = \max\{t_k^S, C_{i,j-1}\} \tag{1}$$

When operation $O_{i,j}$ is assigned on machine M_k, it checks the idle time intervals between operations that have already been scheduled on machine M_k from left to right to find the earliest available time $S_{i,j}$. If there is enough time span from $S_{i,j}$ until t_k^E to complete $O_{i,j}$, i.e.

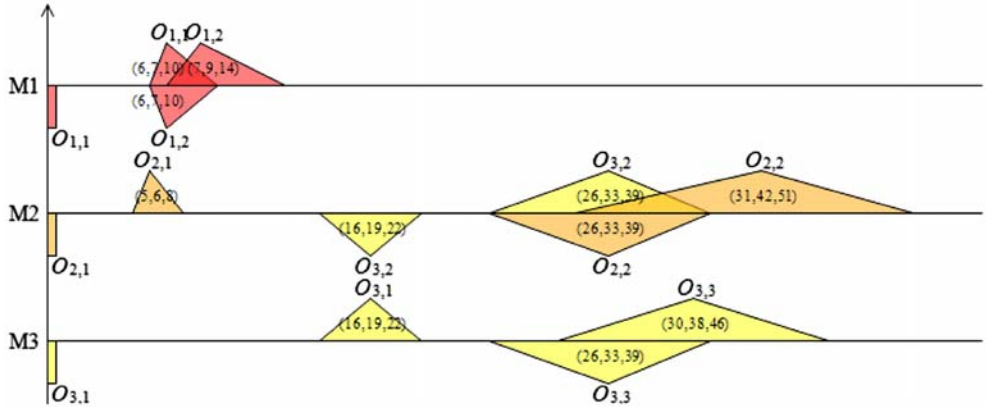

Figure 3. Fuzzy Gantt chart of the solution shown in Figure 2.

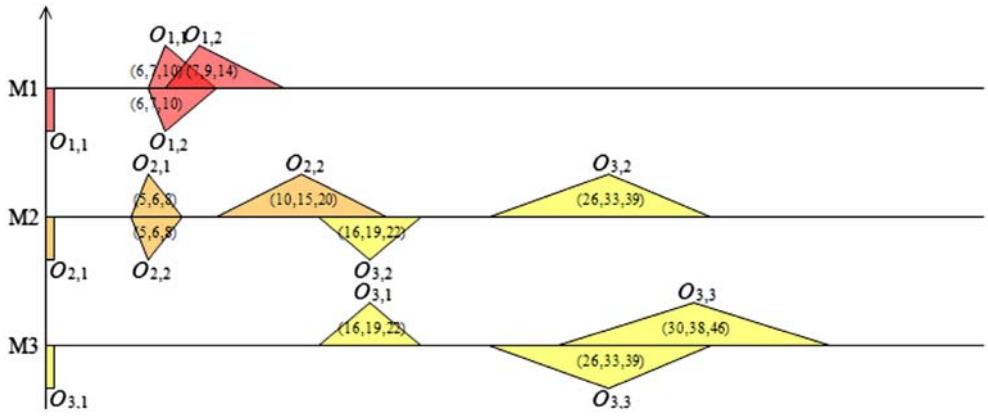

Figure 4. Fuzzy Gantt chart after left shifting to the schedule in Figure 2.

$$\max\{t_k^S, C_{i,j-1}\} + t_{i,j,k} \leq t_k^E \tag{2}$$

Then we say that time interval $[t_k^S, t_k^E]$ is available for $O_{i,j}$. That is, $O_{i,j}$ can be left shifted. For each operation following the operation sequence vector of the representation, the above left-shift scheme is used to allocate it on its assigned machine from left to right. In this way, a representation is decoded to a detailed schedule.

For the example shown in Table 1 and Figure 3, according to the operation sequence vector and the machine assignment vector in Figure 2, operation $O_{2,2}$ is assigned on machine M_2 behind operation $O_{3,2}$. It checks the idle time intervals between operations that have already been scheduled on the machine from left to right. And then it gets $S_{i,j} = (5, 6, 8)$ and $t_k^E = (16, 19, 22)$. According to the ranking criteria of fuzzy numbers, Equation (2) is satisfied as $t_{i,j,k} = (5, 9, 12)$. Thus, $O_{2,2}$ can be left shifted. The fuzzy Gantt chart after left shifting is shown in Figure 4.

4.3 *Probability model and updating mechanism*

The EDA produces new population by sampling a probability model. 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 operation probability matrix A_1 represents the probability that job J_j appears before or in position i of the operation sequence vector at generation l. The value of p_{ij} implies the importance of a job when scheduling the operations on machines. For every i ($i = 1, 2, \cdots, T_o$) and j ($j = 1, 2, \cdots, n$), p_{ij} is initialised as $p_{ij}(0) = 1/n$, which ensures that the whole solution space can be sampled uniformly.

The element $q_{ijk}(l)$ of machine probability matrix A_2 represents the probability that operation $O_{i,j}$ is processed on machine M_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 initialised as follows:

$$q_{ijk}(0) = \begin{cases} \dfrac{1}{m_{i,j}}, & \text{if } O_{i,j} \text{ can be processed on machine } k \\ 0, & \text{else} \end{cases}, \forall i, j, k \tag{3}$$

via sampling the probability matrixes A_1 and A_2, it generates new individuals. In particular, to generate a new solution it first generates the operation sequence vector from the first position to the last by sampling the probability matrix A_1. For every position i of the operation sequence vector, job J_j is selected with a probability of p_{ij}. If job J_j has already appeared n_j times, it means the processing procedure of job J_j is completed. Then, the whole column $p_{1j}, p_{2j}, \cdots, p_{T_oj}$ of the probability matrix A_1 will be set as zero and all the elements of A_1 will be normalised to maintain that each row sums up to 1. Similarly, the machine assignment vector is generated by sampling the probability matrix A_2. For every operation $O_{i,j}$, machine M_k is selected with a probability of q_{ijk}. In such a way, a population with P individuals are generated.

Next, it determines the superior sub-population with the best SP solutions of the population according to the ranking criteria of fuzzy numbers. And then, the probability matrices A_1 and A_2 are updated according to the following equations:

$$p_{ij}(l+1) = (1-\alpha)p_{ij}(l) + \frac{\alpha}{i \times SP}\sum_{s=1}^{SP} I_{ij}^{s}, \forall i,j \tag{4}$$

$$q_{ijk}(l+1) = (1-\beta)q_{ijk}(l) + \frac{\beta}{SP}\sum_{s=1}^{SP} \tilde{I}_{ijk}^{s}, \forall i,j,k \tag{5}$$

where I_{ij}^{s}, $\tilde{I}_{ijk}^{s}$ are the following indicator functions of the sth individual in the superior sub-population.

$$I_{ij}^{s} = \begin{cases} 1, \text{ if job } J_j \text{ appears before or in position } i \\ 0, \text{else} \end{cases} \tag{6}$$

$$\tilde{I}_{ijk}^{s} = \begin{cases} 1, \text{ if operation } O_{i,j} \text{ is processed on machine } M_k \\ 0, \text{ else} \end{cases} \tag{7}$$

The above updating process can be regarded as a kind of increased learning, where the second terms on the right hand side of Equations (4) and (5) represent learning information from the superior sub-population.

4.4 *Procedure of the EDA*

With the above design, the flowchart of the EDA for solving the fFJSP is illustrated in Figure 5.

In the flowchart, it generates P individuals by sampling the probability matrices after the matrices are initialised. Then, the best SP individuals are selected to update the probability model. If a stopping condition is met, it will stop the algorithm and output the optimal solution; otherwise, it will generate P new individuals for the next generation of evolution. Since the probability model is updated with elite individuals, it is helpful for the algorithm to track the search region with promising solutions. With left-shift scheme, solutions can be improved as possible. So, it is expected to solve the fFJSP effectively.

4.5 *Computational complexity analysis*

For each generation of the EDA, its computational complexity can be roughly analysed as follows.

For the updating process, first it is with the computational complexity $O(P\log P)$ by using the quick sorting method to select the best SP individuals from population; then, it is with the complexity $O(T_o \times SP + T_o \times n)$ to update all the $T_o \times n$ elements of A_1 by the operator sequence vectors and with the computational complexity $O(T_o \times SP + T_o \times m)$ to update A_2 by the machine assignment vectors. Thus, the computational complexity for updating process is $O[T_o(SP + m + n) + P\log P]$.

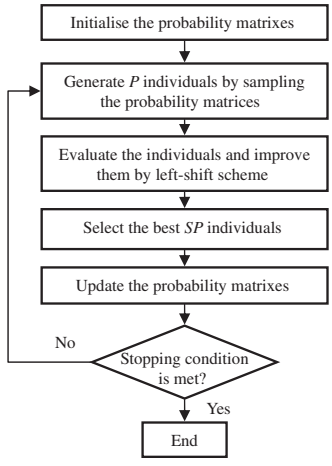

Figure 5. The flowchart of the EDA for the fFJSP.

For the sampling process, every gene is generated with the roulette strategy by sampling A_1 and A_2 to obtain a new individual. To generate an operation sequence vector, it is with the complexity $O(T_o \times n)$. To generate a machine assignment vector, it is with the complexity $O(T_o \times m)$. Thus, the computational complexity for generating P individuals is $O[PT_o(m + n)]$.

Table 2. Combinations of parameter values.

	Factor level			
Parameters	1	2	3	4
P	50.0	100.0	150.0	200.0
η	10.0	20.0	30.0	40.0
α	0.1	0.2	0.3	0.4
β	0.1	0.2	0.3	0.4

Table 3. Orthogonal array and ARV values.

	Factor				
Experiment number	P	η	α	β	ARV
1	1	1	1	1	32.7125
2	1	2	2	2	32
3	1	3	3	3	31.975
4	1	4	4	4	34.2125
5	2	1	2	3	33.1875
6	2	2	1	4	32.9875
7	2	3	4	1	30.9625
8	2	4	3	2	30.85
9	3	1	3	4	31.525
10	3	2	4	3	31.5375
11	3	3	1	2	32.05
12	3	4	2	1	31.55
13	4	1	4	2	31.775
14	4	2	3	1	30.625
15	4	3	2	4	32.55
16	4	4	1	3	33.1375

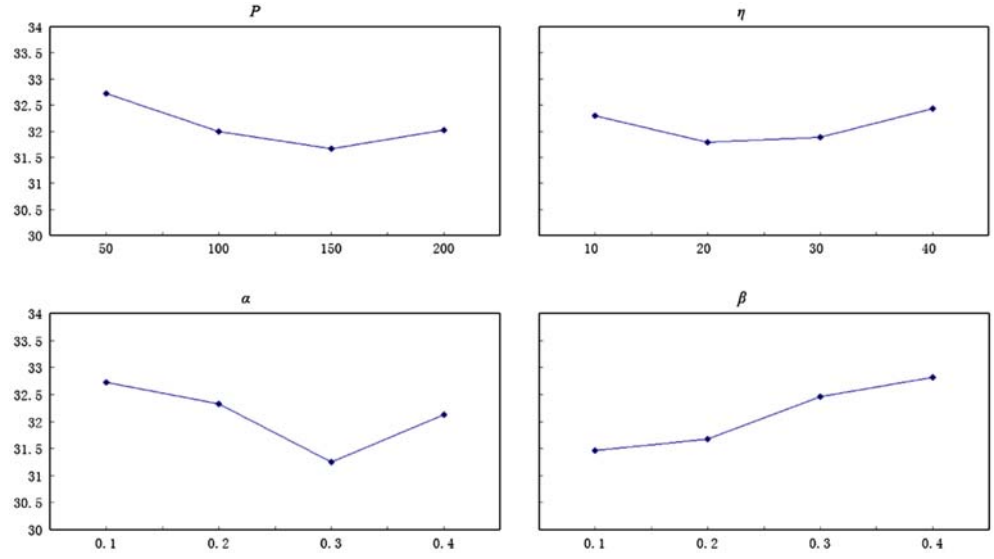

Figure 6. Factor level trend of the EDA.

It can be seen that the complexity of the proposed EDA is not large. In the next section we will further verify its efficiency by numerical test.

5. Computational results and comparisons

To test the performance of the proposed EDA, numerical simulations are carried out based on the instances from Lei (2010a, 2012). We code the algorithm in C++ and run it on Thinkpad T420 with a 2.3 GHz processor/2 GB RAM. For each instance, the algorithm is run 20 times independently with the same number of evaluations as Lei (2012), i.e. 150,000.

5.1 *Parameters setting*

In our EDA, it contains four key parameters: P (population size), η (percentage of superior sub-population from population, $\eta = SP/P \times 100$), α (learning rate of A_1), and β (learning rate of A_2). To investigate the influence of these

Table 4. Response value and rank of each parameter.

Level	P	η	α	β
1	32.725	32.3	32.721875	31.4625
2	31.996875	31.7875	32.321875	31.66875
3	31.665625	31.884375	31.24375	32.459375
4	32.021875	32.4375	32.121875	32.81875
Delta	1.059375	0.65	1.478125	1.35625
Rank	3	4	1	2

Table 5. Results of the five instances.

Instance	Algorithm	Average value	Best value	Worst value
Case 1	EDA	(20.3, 30.5, 41.6)	(20, 28, 40)	(22, 32, 43)
	CGA	(23.1, 33.1, 43.4)	(21, 29, 41)	(25, 37, 47)
	DIGA	(22.5, 32.7, 43.3)	(21, 31, 40)	(25, 36, 48)
	PEGA	(25.0, 35.1, 47.2)	(23, 31, 42)	(29, 40, 50)
	PSO+SA	(26.2, 36.9, 47.7)	(25, 32, 40)	(27, 41, 54)
Case 2	EDA	(33.7, 46.9, 57.9)	(32, 46, 57)	(34, 48, 58)
	CGA	(35.0, 47.1, 60.6)	(32, 47, 57)	(38, 49, 64)
	DIGA	(33.4, 47.5, 62.1)	(31, 47, 59)	(38, 50, 66)
	PEGA	(36.9, 51.0, 65.9)	(34, 45, 60)	(38, 55, 72)
	PSO+SA	(36.7, 51.2, 65.2)	(34, 45, 60)	(39, 54, 74)
Case 3	EDA	(32.8, 47.2, 62.9)	(31, 46, 60)	(34, 49, 66)
	CGA	(36.4, 50.8, 66.0)	(34, 47, 63)	(38, 53, 71)
	DIGA	(36.1, 51.5, 67.5)	(36, 47, 64)	(40, 55, 73)
	PEGA	(40.6, 56.4, 73.3)	(38, 51, 66)	(40, 59, 77)
	PSO+SA	(38.6, 54.4, 70.0)	(36, 51, 65)	(40, 57, 75)
Case 4	EDA	(24.8, 37.2, 51.9)	(21, 36, 50)	(24, 39, 57)
	CGA	(27.4, 40.4, 55.0)	(26, 37, 51)	(29, 42, 59)
	DIGA	(29.6, 42.4, 56.9)	(28, 40, 56)	(30, 46, 63)
	PEGA	(34.3, 48.8, 65.7)	(34, 46, 63)	(35, 50, 68)
	PSO+SA	(33.6, 47.9, 64.5)	(32, 45, 62)	(34, 49, 68)
Case 5	EDA	(38.6, 56.9, 78.3)	(36, 55, 73)	(40, 60, 81)
	CGA	(47.0, 65.4, 86.0)	(42, 62, 82)	(49, 70, 91)
	DIGA	(45.8, 66.3, 88.7)	(42, 63, 84)	(49, 71, 92)
	PEGA	(50.3, 74.0, 96.5)	(48, 68, 94)	(50, 74, 100)
	PSO+SA	(51.2, 74.6, 97.6)	(48, 72, 93)	(52, 73, 101)

parameters on the performance of the EDA, we implement the Taguchi method of design of experiment (DOE) (Montgomery 2005) by using Instance 1. Combinations of different values of these parameters are listed in Table 2.

For each combination, the EDA is run 20 times independently. The response variable value (RV) is the defined as $(x_1 + 2x_2 + x_3)/4$ for a triangular fuzzy number $X = (x_1, x_2, x_3)$ of the makespan. The average response variable (ARV) value is the average of RV values obtained in 20 times. Clearly, the smaller the ARV is, the better the combination is. Considering four factor levels for each parameter, we get an orthogonal array $L_{16}(4^4)$, which is shown in Table 3 together with the obtained ARV values.

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

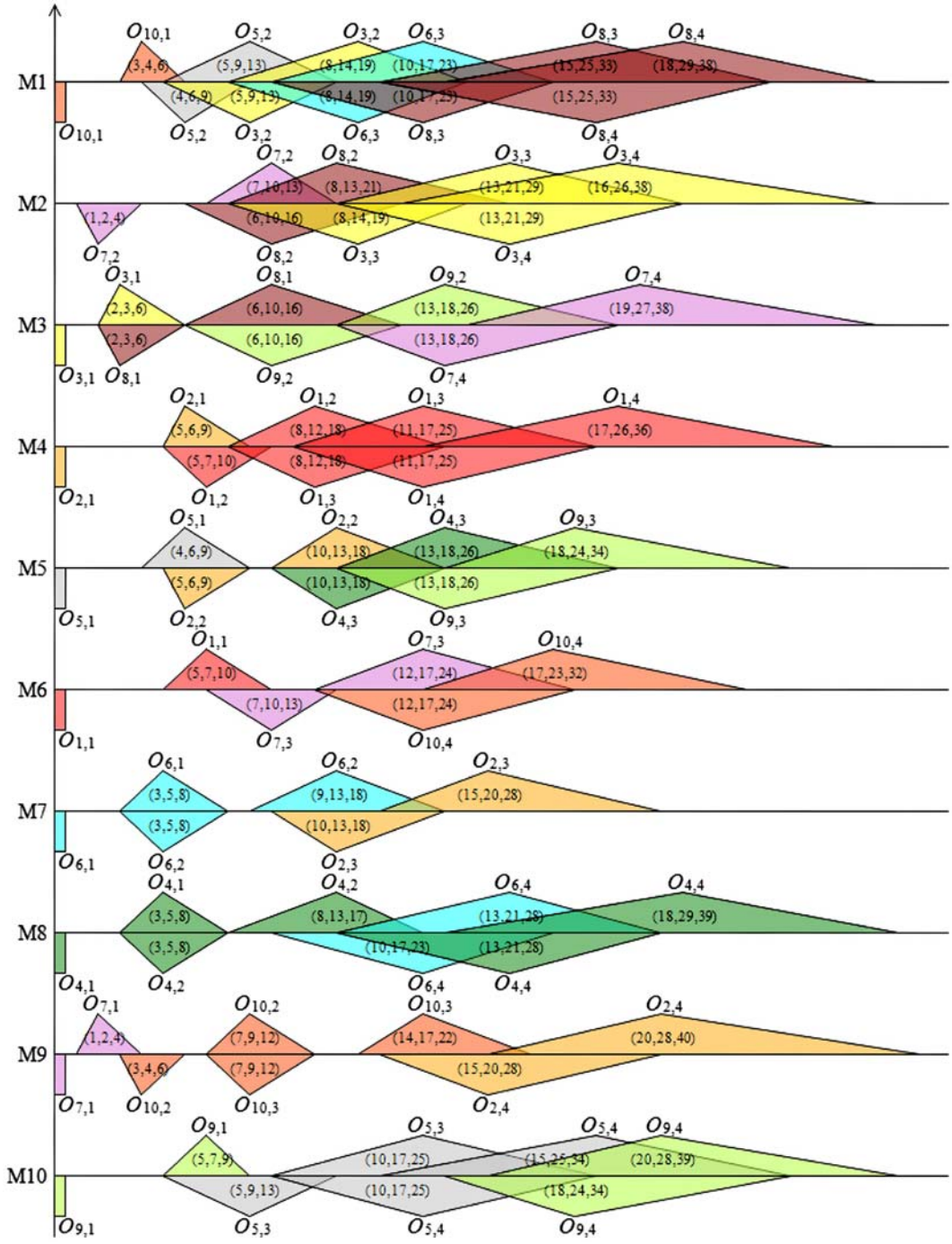

Figure 7. The best solution of Instance 1 obtained by the EDA.

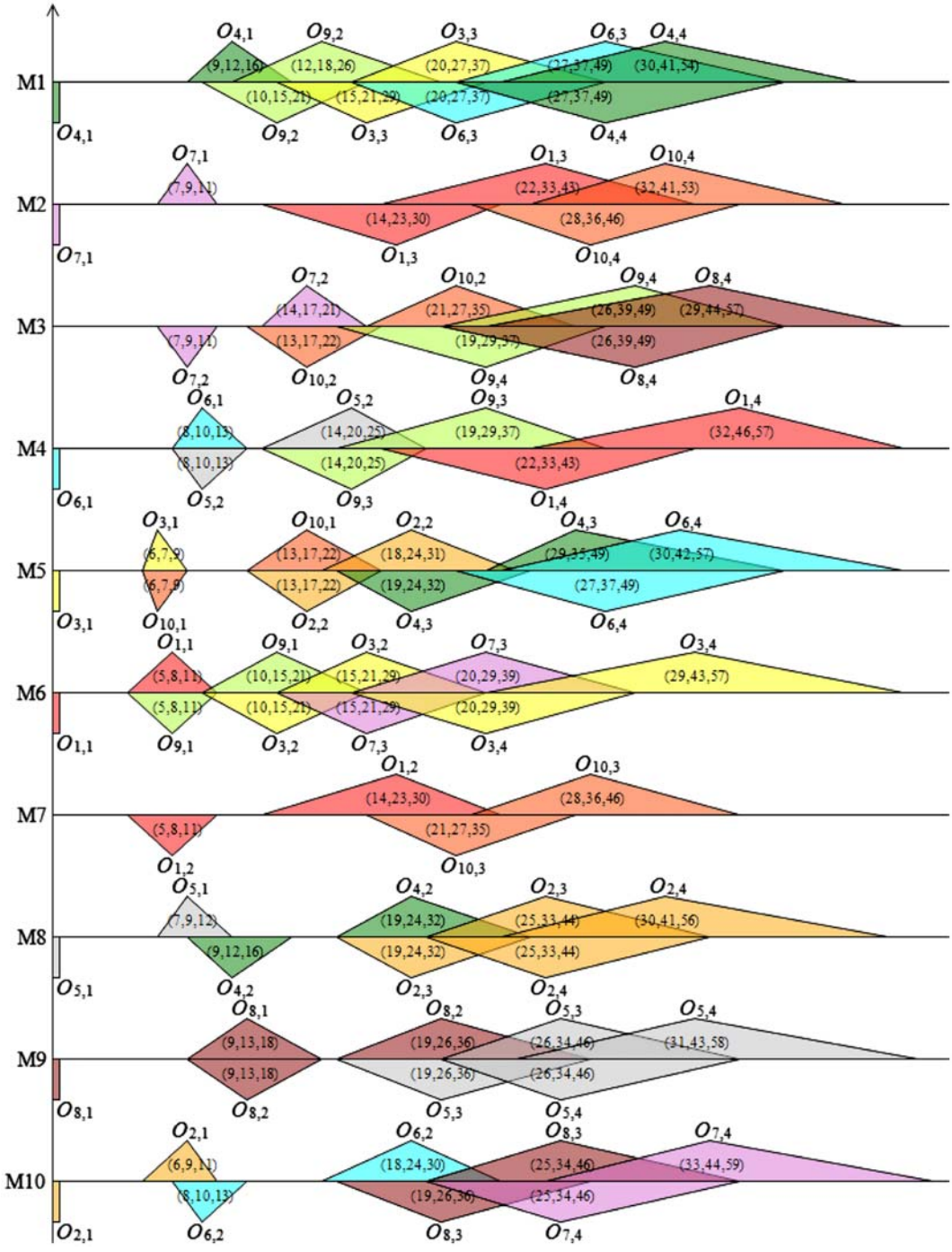

Figure 8. The best solution of Instance 2 obtained by the EDA.

From Table 4 it can be seen that the learning rate α of A_1 is the most significant one among the four parameters. That is, the learning rate of the matrix for operator sequence is crucial to the EDA. A small value of α could lead to slow convergence while a large value could lead to premature convergence. The significance of the learning rate β of A_2 ranks the second. A large value of β could lead to premature convergence. Besides, the population size P ranks the third. A large population size ensures that the whole solution space can be sampled sufficiently. With a fixed maximum number of evaluations, however, a larger population size leads to fewer generations which may cause an insufficient evolution. As for η, it has the slightest influence while a moderate value is preferable. According to the above analysis, the parameter setting for the EDA is suggested as $P = 150$, $\eta = 20$, $\alpha = 0.3$ and $\beta = 0.1$, which will be used for the following tests.

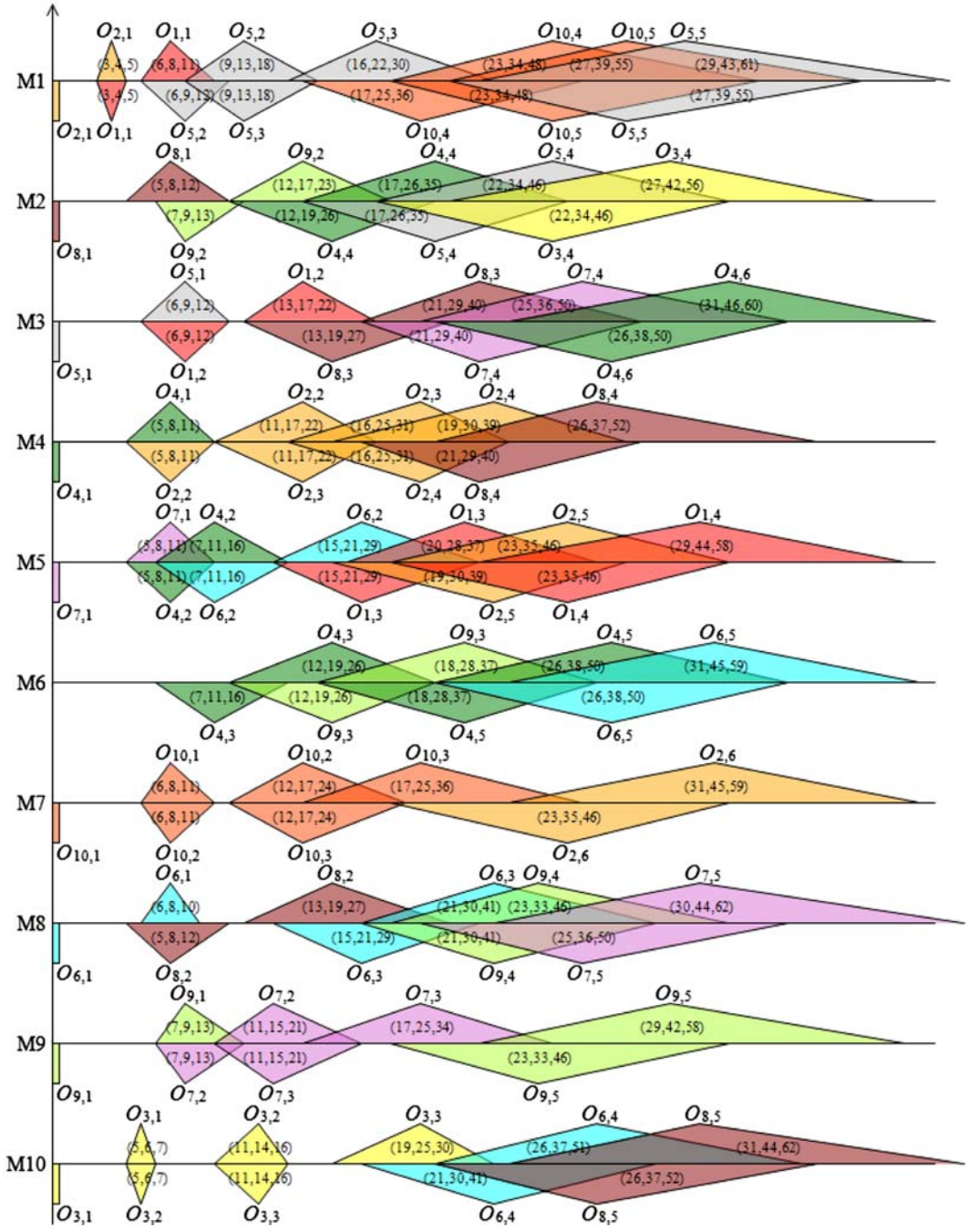

Figure 9. The best solution of Instance 3 obtained by the EDA.

5.2 *Testing results and comparisons*

With the five instances (Lei 2010a, 2012), we compare the EDA with four existing algorithms, including CGA (Lei 2012), DIGA (Lei 2010a), PEGA (Pezzella, Morganti, and Ciaschetti 2008), and PSO+SA (Xia and Wu 2005). The results are listed in Table 5, where the results of the existing algorithms are directly from literature (Lei 2012).

From Table 5, it can be seen that the EDA is the best one among all the algorithms in solving the five instances. Compared with other algorithms, the EDA can obtain the best values in terms of the average value, the best value and the worst value, which implies that the EDA is the most effective. For example, when solving Instance 4, the average value (24.8, 37.2, 51.9), the best value (21, 36, 50) and the worst value (24, 39, 57) obtained by EDA are better than

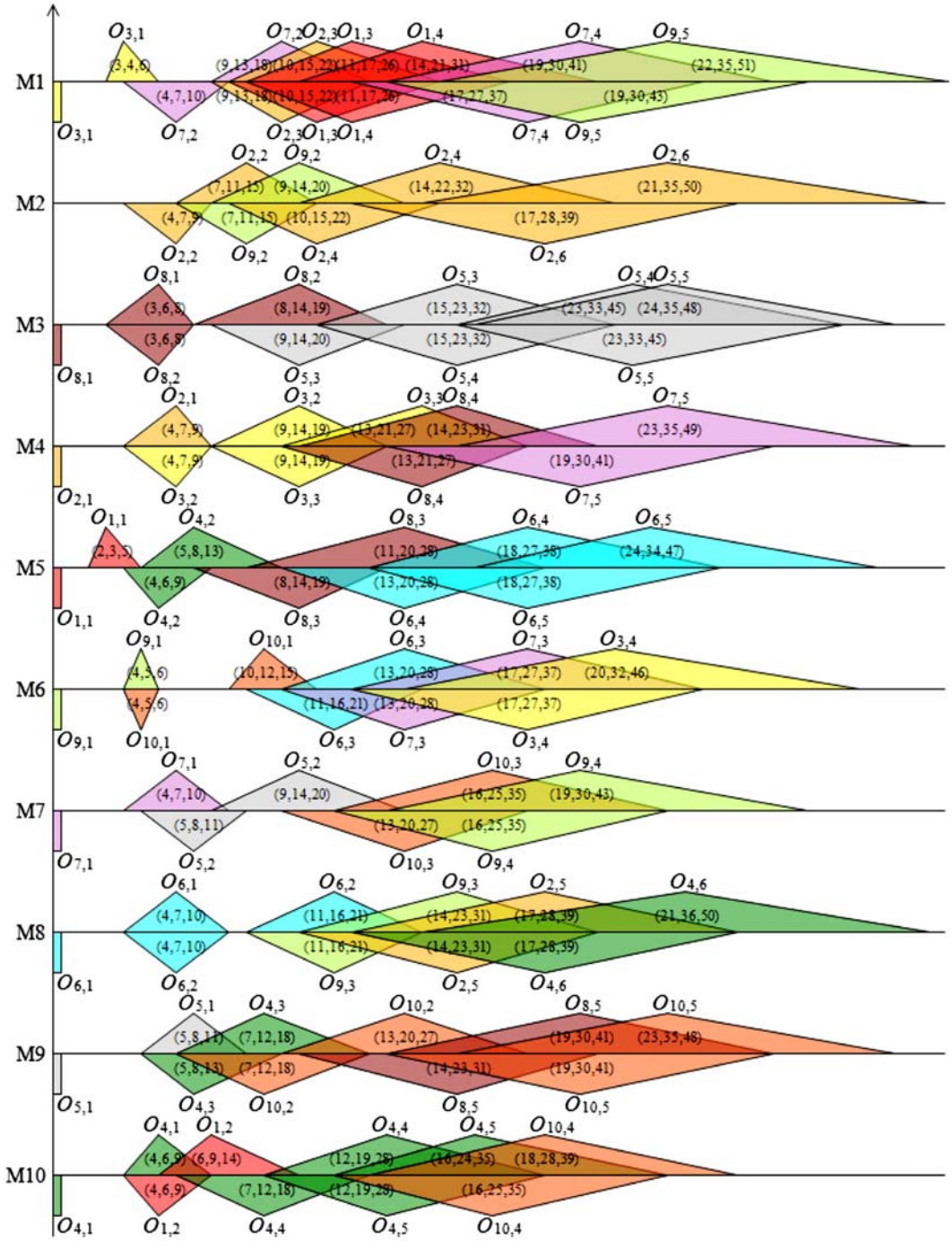

Figure 10. The best solution of Instance 4 obtained by the EDA.

those values obtained by the other four algorithms in the fuzzy sense. In Figures 7–11, the fuzzy Gantt charts of the best solutions obtained by the EDA for Instance 1–5 are illustrated.

In addition, the average CPU time of 20 different runs is listed in Table 6. It can be seen that the EDA is the most efficient one among all the algorithms. Even for Instance 5 with the largest scale, the EDA only spends an average of 9.83 s.

So, it is concluded that the EDA is an efficient and effective algorithm for solving the fFJSP. The superiority of the EDA owes to the following aspects.

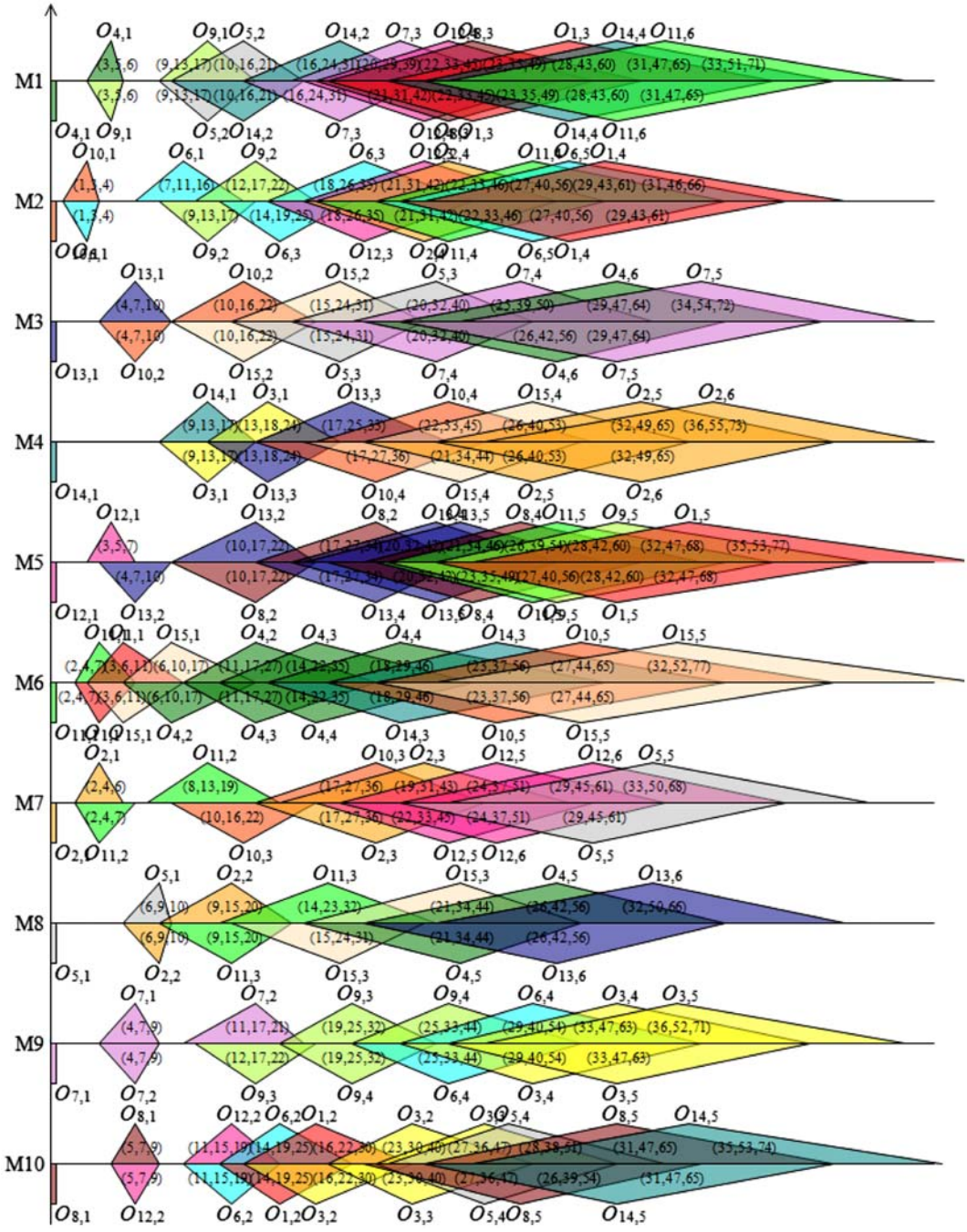

Figure 11. The best solution of Instance 5 obtained by the EDA.

Table 6. CPU time (s) of the five instances.

Algorithm	Instance 1	Instance 2	Instance 3	Instance 4	Instance 5
EDA[a]	3.65	3.63	4.86	4.56	9.83
CGA[b]	8.29	8.26	10.66	10.77	23.87
DIGA[b]	15.36	15.57	18.87	19.02	37.82
PEGA[b]	12.56	12.67	15.23	15.71	30.15
PSO+SA[b]	12.40	12.33	15.24	15.66	30.90

Notes: [a]Core i5 2.3 GHz CPU. [b]1.7 GHz CPU.

(1) With the well-designed probability model and the suitable updating mechanism, it is helpful to explore the search space effectively.
(2) With the left-shift scheme, it is helpful to refine a schedule with a smaller fuzzy makespan.
(3) With the investigation of parameter setting, suitable values are determined to achieve good performance.

6. Conclusions

In this paper, an effective estimation of distribution algorithm was proposed for solving the flexible job-shop scheduling problem with fuzzy processing time. A specific probability model was designed for the EDA to solve the problem. And a mechanism was provided to update the model reasonably. The model was used for generating promising solutions among the search space, and a left-shift scheme was used to improve the solutions. The influence of parameter setting was investigated and suitable setting was recommended. Numerical testing results demonstrated the effectiveness of the proposed EDA in comparing with the existing algorithms. The future work is to study the EDA for other kinds of fuzzy scheduling problems. We also will carry out some research of the EDA for multi-objective scheduling problems.

Acknowledgments

This research is partially supported by the National Key Basic Research and Development Program of China (2013CB329503), the National Science Foundation of China (61174189 and 61025018), the Doctoral Program Foundation of Institutions of Higher Education of China (20100002110014), and the National Science and Technology Major Project of China (No. 2011ZX02504-008).

References

Amiri, M., M. Zandieh, M. Yazdani, and A. Bagheri. 2010. "A Variable Neighbourhood Search Algorithm for the Flexible Job-Shop Scheduling Problem." *International Journal of Production Research* 48: 5671–5689.

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., and S. Davies. 1997. "Using Optimal Dependency-trees for Combinatorial Optimization: Learning the Structure of the Search Space." In: *Proceedings of the 14th International Conference on Machine Learning*. San Francisco, 30–38.

Bortolan, G., and R. Degani. 1985. "A Review of Some Methods for Ranking Fuzzy Subsets." *Fuzzy Sets and Systems* 15 (1): 1–19.

Brandimarte, P. 1993. "Routing and Scheduling in a Flexible Job Shop by Tabu Search." *Annals of Operations Research* 41: 157–183.

Bruker, P., and R. Schlie. 1990. "Job-Shop Scheduling with Multi-Purpose Machines." *Computing* 45: 369–375.

Cesar, R. M., E. Bengoetxea, I. Bloch, and P. Larranaga. 2005. "Inexact Graph Matching for Model-Based Recognition: Evaluation and Comparison of Optimization Algorithms." *Pattern Recognition* 38: 2099–2113.

Dauzère-Pérès, S., and J. Paulli. 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., C. L. Isbell Jr., and P. Viola. 1997. "MIMIC: Finding Optima by Estimating Probability Densities." *Advances in Neural Information Processing Systems*, Cambridge: MIT Press, 424–430.

Frutos, M., A. C. Olivera, and F. Tohme. 2010. "A Memetic Algorithm Based on a NSGAII Scheme for the Flexible Job-Shop Scheduling Problem." *Annals of Operations Research* 181: 745–765.

Gao, J., L. Y. Sun, and M. Gen. 2008. "A Hybrid Genetic and Variable Neighborhood Descent Algorithm for Flexible Job Shop Scheduling Problems." *Computers & Operations Research* 35: 2892–2907.

González-Rodríguez, I., C. R. Vela, and J. Puente. 2010. "A Genetic Solution Based on Lexicographical Goal Programming for a Multiobjective Job Shop with Uncertainty." *Journal of Intelligent Manufacturing* 21: 65–73.

Harik, G. R., F. G. Lobo, and D. E. Goldberg. 1998. "The Compact Genetic Algorithm." In: *Proceedings of the IEEE Conference on Evolutionary Computation.* Indianapolis, 523–528.

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.

Hu, Y. M., M. H. Yin, and X. T. Li. 2011. "A Novel Objective Function for Job-Shop Scheduling Problem with Fuzzy Processing Time and Fuzzy Due Date Using Differential Evolution Algorithm." *The International Journal of Advanced Manufacturing Technology* 56: 1125–1138.

Jarboui, B., M. Eddaly, and P. Siarry. 2009. "An Estimation of Distribution Algorithm for Minimizing the Total Flowtime in Permutation Flowshop Scheduling Problems." *Computers & Operations Research* 36: 2638–2646.

Larranaga, P., and J. A. Lozano. 2002. *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*. Boston, MA: Kluwer Press.

Lei, D. M. 2008. "Pareto Archive Particle Swarm Optimization for Multi-Objective Fuzzy Job Shop Scheduling Problems." *The International Journal of Advanced Manufacturing Technology* 37: 157–165.

Lei, D. M. 2010a. "A Genetic Algorithm for Flexible Job Shop Scheduling with Fuzzy Processing Time." *International Journal of Production Research* 48: 2995–3013.

Lei, D. M. 2010b. "Fuzzy Job Shop Scheduling Problem with Availability Constraints." *Computers & Industrial Engineering* 58: 610–617.

Lei, D. M. 2011. "Scheduling Fuzzy Job Shop with Preventive Maintenance through Swarm-Based Neighborhood Search." *The International Journal of Advanced Manufacturing Technology* 54: 1121–1128.

Lei, D. M. 2012 "Co-Evolutionary Genetic Algorithm for Fuzzy Flexible Job Shop Scheduling." *Applied Soft Computing* doi:10.1016/j.asoc.2012.03.025.

Li, J. Q., Q. K. Pan, and Y. C. Liang. 2010. "An Effective Hybrid Tabu Search Algorithm for Multi-Objective Flexible Job-Shop Scheduling Problems." *Computers & Industrial Engineering* 59: 647–662.

Li, J. Q., Q. K. Pan, and K. Z. Gao. 2011. "Pareto-Based Discrete Artificial Bee Colony Algorithm for Multi-Objective Flexible Job Shop Scheduling Problems." *The International Journal of Advanced Manufacturing Technology* 55: 1159–1169.

Mastrolilli, M., and L. M. Gambardella. 2000. "Effective Neighborhood Functions for the Flexible Job Shop Problem." *Journal of Scheduling* 3: 3–20.

MÄuhlenbein, H., and G. Paass. 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 and Sons.

Moslehi, G., and M. Mahnam. 2011. "A Pareto Approach to Multi-Objective Flexible Job-Shop Scheduling Problem Using Particle Swarm Optimization and Local Search." *International Journal of Production Economics* 129: 14–22.

Mühlenbein, H., and T. Mahnig. 1999. "Convergence Theory and Applications of the Factorized Distribution Algorithm." *Jounal of Computing and Information Technology* 7: 19–32.

Pelikan, M., and H. Mühlenbein. 1999. "The Bivariate Marginal Distribution Algorithm." In *Advances in Soft Computing: Engineering Design and Manufacturing*, edited by J. M. Benítez, 521–35. London: Springer-Verlag.

Pelikan, M., D. E. Goldberg, and E. Cantú-Paz. 1999. BOA: "The Bayesian Optimization Algorithm." In: *Proceedings of the Genetic and Evolutionary Computation*. Orlando, 525–532.

Pezzella, F., G. Morganti, and G. Ciaschetti. 2008. "A Genetic Algorithm for the Flexible Job-Shop Scheduling Problem." *Computers & Operations Research* 35: 3202–3212.

Saeys, Y., S. Degroeve, D. Van Aeyels, Y. de Peer, and P. Rouze. 2003. "Fast Feature Selection Using a Simple Estimation of Distribution Algorithm: A Case Study on Splice Site Prediction." *Bioinformatics* 19: 179–188.

Sagarna, R., and J. Lozano. 2005. "On the Performance of Estimation of Distribution Algorithms Applied to Software Testing." *Applied Artificial Intelligence* 19: 457–489.

Sakawa, M., and R. Kubota. 2000. "Fuzzy Programming for Multiobjective Job Shop Scheduling with Fuzzy Processing Time and Fuzzy Duedate through Genetic Algorithms." *European Journal of Operational Research* 120: 393–407.

Sakawa, M., and T. Mori. 1999. "An Efficient Genetic Algorithm for Job-Shop Scheduling Problems with Fuzzy Processing Time and Fuzzy Duedate." *Computers & Industrial Engineering* 36: 325–341.

Tay, J. C., and N. B. Ho. 2008. "Evolving Dispatching Rules Using Genetic Programming for Solving Multi-Objective Flexible Job-Shop Problems." *Computers & Industrial Engineering* 54: 453–473.

Wang, L., and C. Fang. 2012. "An Effective Estimation of Distribution Algorithm for the Multi-Mode Resource-Constrained Project Scheduling Problem." *Computers & Operations Research* 39: 449–460.

Wang, S. J., B. H. Zhou, and L. F. Xi. 2008. "A Filtered-Beam-Search-Based Heuristic Algorithm for Flexible Job-Shop Scheduling Problem." *International Journal of Production Research* 46: 3027–3058.

Wang, X. J., L. Gao, C. Y. Zhang, and X. Y. Shao. 2010. "A Multi-Objective Genetic Algorithm Based on Immune and Entropy Principle for Flexible Job-Shop Scheduling Problem." *The International Journal of Advanced Manufacturing Technology* 51: 757–767.

Wang, L., G. Zhou, Y. Xu, and M. Liu. 2012a. "An Enhanced Pareto-Based Artificial Bee Colony Algorithm for the Multi-Objective Flexible Job-Shop Scheduling." *International Journal of Advanced Manufacturing Technology* 60 (9–12): 1111–1123.

Wang, L., S. Y. Wang, and C. Fang. 2012b. "An Effective Hybrid EDA-Based Algorithm for Solving Multidimensional Knapsack Problem." *Expert Systems with Applications* 39 (5): 5593–5599.

Wang, L., S. Y. Wang, Y. Xu, G. Zhou, and M. Liu. 2012c. "A Bi-Population Based Estimation of Distribution Algorithm for the Flexible Job-Shop Scheduling Problem." *Computers & Industrial Engineering* 62 (4): 917–926.

Wang, L., G. Zhou, Y. Xu, S. Y. Wang, and M. Liu. 2012d. "An Effective Artificial Bee Colony Algorithm for the Flexible Job-Shop Scheduling Problem." *The International Journal of Advanced Manufacturing Technology* 60: 303–315.

Xia, W. J., and Z. M. Wu. 2005. "An Effective Hybrid Optimization Approach for Multi-Objective Flexible Job-Shop Scheduling Problems." *Computers & Industrial Engineering* 48: 409–425.

Xing, L. N., Y. W. Chen, P. Wang, Q. S. Zhao, and J. Xiong. 2010. "A Knowledge-Based Ant Colony Optimization for Flexible Job Shop Scheduling Problems." *Applied Soft Computing* 10: 888–896.

Yazdani, M., M. Amiri, and M. Zandieh. 2010. "Flexible Job-Shop Scheduling with Parallel Variable Neighborhood Search Algorithm." *Expert Systems with Applications* 37: 678–687.