An estimation of distribution algorithm for the multi-objective flexible job-shop scheduling problem

Shengyao Wang, Ling Wang, Min Liu and Ye Xu
Tsinghua National Laboratory for Information Science and Technology (TNList),
Department of Automation, Tsinghua University
Beijing, China
wangshengyao@tsinghua.org.cn; wangling@tsinghua.edu.cn; lium@tsinghua.edu.cn; xuye05@mails.tsinghua.edu.cn

***Abstract*—In this paper, an effective estimation of distribution algorithm (EDA) is proposed to solve the multi-objective flexible job-shop scheduling problem (MFJSP) to minimize the maximum completion time, the total workload of machines and the workload of the critical machine simultaneously. Within the framework of EDA, the new individuals are generated by sampling a probability model, which is built and updated with the superior sub-population by a proposed mechanism. Moreover, the EDA utilizes multiple strategies in a combination way to generate the initial solutions, and used a local search strategy based on critical path to enhance the exploitation ability. Based on the Taguchi method of design-of-experiment, the influence of parameter setting is investigated and suitable parameters are suggested. Finally, numerical simulation based on some well-known benchmarks and comparisons with some existing algorithms are carried out. The results demonstrate the effectiveness of the proposed EDA to solve the MFJSP.**

Keywords—estimation of distribution algorithm; multi-objective flexible job-shop scheduling problem; probability model; critical path; design of experiment

I. INTRODUCTION

The flexible job-shop scheduling problem (FJSP) is a generalization of the classical job-shop scheduling problem (JSP) for flexible manufacturing systems. Close to the real manufacturing situation, the FJSP is of wide application background . In the FJSP, some machines may have the ability of performing more than one type of operations. So, the FJSP consists of two sub-problems: the routing sub-problem that assigns each job to the capable machines, and the scheduling sub-problem that sequences the assigned jobs on all the machines to obtain a feasible schedule to optimize the schedule objectives. Therefore, the FJSP is more difficult to solve than the classical JSP because it should determine the assignment of jobs to machines as well as the sequence of all the jobs. The FJSP has been considered one of the most difficult problems in combinatorial optimization [1]. Hence, the study of the FJSP in theory, methodology and applications is significant in both engineering field and academic field.

Bruker and Schlie proposed a polynomial algorithm to solve the FJSP with two jobs, where each operation had the same processing time on different machines [2]. Later, a hybrid tabu search (TS) algorithm with some existing dispatching rules was proposed [3]. Dauzere-Peres and Paulli proposed a TS algorithm based on an integrated approach [4]. Then Mastrolilli and Gambardella improved the TS algorithm in terms of computation time and solution quality with two neighborhood functions [5]. Saidi-mehrabad and Fattahi proposed a TS algorithm with two heuristics to solve the FJSP with sequence-dependent setups [6]. Chen et al. proposed a hierarchical algorithm hybridizing the machine selection module and the operation scheduling module for solving the FJSP with parallel machines and reentrant process [7]. As for genetic algorithm (GA), Gao et al. proposed a GA hybridizing with the variable neighborhood search to solve the FJSP [8]. Pezzella et al. proposed a GA integrating different strategies [9], and Defersha and Chen presented a parallel GA for the FJSP with sequence-dependent setups [10]. In addition, a parallel variable neighborhood search (PVNS) algorithm based on six neighborhood structures was developed [11] and a knowledge-based ant colony optimization algorithm (KBACO) was proposed in [12]. In [13], an efficient architecture was proposed for scheduling the FJSP with machine availability constraints. Recently, Li et al. developed a hybrid TS algorithm with an efficient neighborhood structure for the FJSP [14].

Compared to the single-objective FJSP, the study on multi-objective flexible job-shop scheduling (MFJSP) is relatively limited. To solve the assignment problem in the MFJSP, Kacem et al. proposed a localization approach [15]. Xia and Wu used particle swarm optimization (PSO) to assign jobs to machines and used simulated annealing (SA) to schedule jobs on each machine in a hierarchical approach[16]. Tay and Ho introduced an integrated approach based on genetic programming for solving the MFJSP to minimize makespan, mean tardiness and mean flow times [17]. Xing et al. developed a local search algorithm [18] and Li et al. introduced a TS algorithm combining two adaptive rules with an effective neighborhood structure [19].

As a population-based algorithm, estimation of distribution algorithm (EDA) [20] has gained increasing attention and wide applications during recent years. According to the complexity of the model, the EDA can be classified as univariate model, bivariate model or multivariate model. The population-based incremental learning [21], univariate marginal distribution algorithm [22] and compact GA [23] are univariate models, while mutual information maximization for input clustering [24], combining optimizers with mutual information trees [25] and bivariate marginal distribution algorithm [26] are bivariate

This research was financially supported by the National Key Basic Research and Development Program of China (No. 2013CB329503), National Science Foundation of China (No. 61174189, 61025018) and National Science and Technology Major Project of China (No. 2011ZX02504-008).

978-1-4673-5909-2/13/$31.00 ©2013 IEEE

models. The factorized distribution algorithm [27], extended compact GA [28] and bayesian optimization algorithm [29] are multivariate models. For more details about the EDA, please refer to [20].

So far the EDAs have been applied to a variety of academic and engineering optimization problems, such as cancer classification, feature selection, machinery structure design, quadratic assignment problem, nurse rostering and so on [21]. However, to the best of our knowledge, there is no research work about the EDA for solving the MFJSP. In this paper, we will propose an effective EDA to solve the MFJSP with the criteria to minimize the maximum completion time, the total workload of machines and the workload of the critical machine simultaneously. A probability model is built with its updating mechanism proposed. Meanwhile, multiple strategies are utilized in a combination way when generating the initial solutions, and a critical-path-based local search strategy is developed to improve the convergence speed. In addition, the influence of parameter setting is investigated. Finally, to demonstrate the effectiveness of the EDA for solving the MFJSP, we test the performance of the EDA and compare it with some existing methods.

The remainder of the paper is organized as follows: In Section II, the problem formulation of the MFJSP is introduced. In Section III, the basic EDA is presented briefly. Then, the proposed EDA for solving the MFJSP is described in details in Section IV. The influence of parameter setting is investigated based on design-of-experiment testing, and then computational results and comparisons are provided in Section V. Finally we end the paper with some conclusions and future work in Section VI.

II. Problem Formulation

The FJSP 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 $M=\{M_1,M_2,\ldots,M_m\}$. 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 processed 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., once the operation starts, it must be completed without interruption. All jobs and machines are available at time 0. Setup times of machines are negligible. The processing time of $O_{i,j}$ on machine M_k is $t_{i,j,k}>0$. Let $C_{i,j}$ be the completion time of $O_{i,j}$. The FJSP is to determine the assignment of machines as well as the sequence of operations on all the machines to optimize certain scheduling objectives.

In this paper, we consider the MFJSP with the following three objectives to be minimized:

(1) The maximal completion time of machines, i.e. C_M;

(2) The total workload of machines, i.e. W_T, which is of interest in minimizing processing time of the machines to improve economic efficiency;

(3) The maximal machine workload, i.e. W_M, which considers the workload balance among all machines.

Mathematically, the MFJSP can be simply formulated as follows [19]:

$$\text{Minimize: } C_M=\max_{1\le i\le n}\{C_{i,n_i}\} \tag{1}$$

$$\text{Minimize: } W_T=\sum_{k=1}^{m}\sum_{i=1}^{n}\sum_{j=1}^{n_i}t_{i,j,k}x_{i,j,k} \tag{2}$$

$$\text{Minimize: } W_M=\max_{1\le k\le m}\{\sum_{i=1}^{n}\sum_{j=1}^{n_i}t_{i,j,k}x_{i,j,k}\} \tag{3}$$

$$\text{Subject to: } C_{i,j}-C_{i,j-1}\ge t_{i,j,k}x_{i,j,k},\ j=2,3,\cdots,n_i;\forall i,k \tag{4}$$

$$\sum\nolimits_{k\in M_{i,j}}x_{i,j,k}=1,\forall i,j \tag{5}$$

$$x_{i,j,k}=\begin{cases}1, \text{if } O_{i,j} \text{ is processed on machine } k\\ 0, \text{otherwise}\end{cases} \tag{6}$$

$$C_{i,j}\ge 0,\forall i,j \tag{7}$$

where, equation (4) ensures that the operations of the same job satisfy the precedence constraints, and equation (5) states that one machine must be selected from the set of available machines for each operation.

III. Estimation of Distribution Algorithm

As a new paradigm in the field of evolutionary computation, estimation of distribution algorithms (EDA) employs explicit probability distributions for optimization [20]. Compared with GAs, the EDAs reproduce new individuals implicitly instead of the crossover and mutation operators. Based on the searching experience, EDA builds a probability model of the most promising area by statistical information and then use the probability model for sampling to generate new individuals. Meanwhile, the parameters of the probability model are updated in each generation with the superior individuals of the new population. In such an iterative way, the population evolves and finally obtains satisfactory solutions.

The general framework of the EDA [22] can be illustrated in Fig. 1.

The critical step of the above procedure is to estimate the probability distribution. The probability model is use to describe the distribution of the solution space. The evolutionary trend of the population is reflected by the updating process. According to the different problem types as well as problem properties, we should developed a proper probability model and its updating mechanism to estimate the underlying probability distribution. Nevertheless, the exploitation capability of EDA is limited because it pays more attention to global exploration. So, balancing the exploration and the exploitation abilities is crucial to an effective EDA.

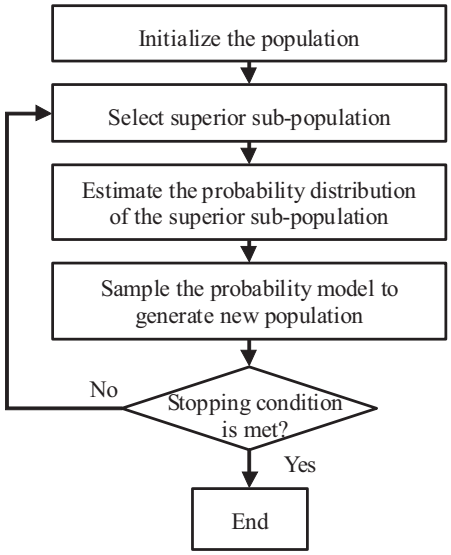

Figure 1. The general framework of the EDA.

IV. EDA FOR MFJSP

In this section, we will propose an effective EDA for solving the MFJSP. First, we will introduce the solution representation, objectives handling, population initialization, probability model, updating mechanism and local search strategy. Then, we will present the flowchart of proposed EDA.

A. Solution Representation

Each solution of the MFJSP is an individual of the population, which is a combination of operation squence and machine assignment. So, a solution can be expressed by the processing sequence of operations and the assignment of operations on the machines. Corresponding to the two sub-problems of the MFJSP, it consists of two vectors named operation sequence vector and machine assignment vector, respectively.

In the operation sequence vector, the number of genes equals to the total number of all the operations T_O. The operation of a job is represented by the corresponding job number, and the k-th occurrence of a job number in the vector refers to the k-th operation in the processing sequence of the job. In the machine assignment vector, each number represents the corresponding machine selected for processing each operation. So the number of genes is also T_O. Next, we provide an example by considering a problem with 4 jobs / 4 machines to explain the representation. Table I lists the processing data. For a feasible solution, the representation is illustrated in Fig. 2.

For the solution in Fig.2, the operation sequence and machine assignment 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). Fig. 3 shows the Gantt chart of this solution.

TABLE I. 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

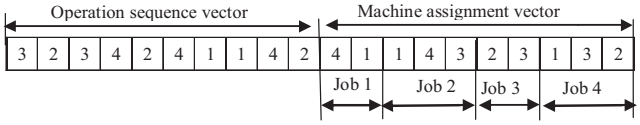

Figure 2. Illustration of the representation of a feasible solution.

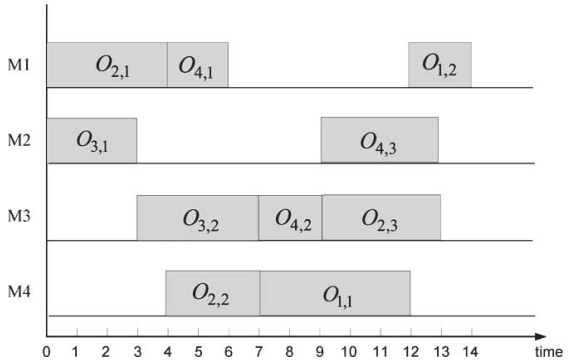

Figure 3. Gantt chart of the solution shown in Fig. 2.

B. Objectives Handling

The methods to solve the multi-objective optimization problems can be generally classified into three types: (1) assign different weight value for each objective and transfer the multi-objective problem to a single-objective problem; (2) use the non-Pareto approaches that utilize operators for processing the different objectives in a separated way; (3) based on the Pareto optimality concept and use the Pareto approaches that solve the multi-objective optimization.

For solving the MFJSP, many researchers applied the first type [16-19], i.e., the total objective function is the weighted sum of the three objective values:

$$F = w_1 \cdot C_M + w_2 \cdot W_T + w_3 \cdot W_M \quad (8)$$

where w_1, w_2 and w_3 represent the weight coefficients for the three objective values, respectively. According to [19], w_1, w_2 and w_3 are set to 0.5, 0.2 and 0.3 for Kacem instances [15], and they are set to 0.8, 0.05 and 0.15 for BRdata instances [3]. In this paper, these weights are also used in the simulation tests.

C. Population Initialization

When it generates the initial population, some different strategies are utilized in a hybrid way to guarantee certain quality and diversity.

For the initial machine assignments, it applies the following two rules.

(1) Random rule. For each operation, select a machine from the candidate machines set randomly and then place it at the position in the machine assignment vector.

(2) Global minimum processing time rule [9]. In our EDA, the initial machine assignments of 40% solutions in the population are generated by the random rule and 40% solutions by the global minimum processing time rule.

Once all the operations are assigned to the machines, they will be sequenced. A feasible schedule is the one with all the precedence constraints among operations of the same job not violated. Three rules are applied to generate the initial operation sequences.

(1) Random rule. Generate the sequence of the operations on each machine randomly.

(2) Most time remaining rule [9]. According to the remaining time of the jobs, Sequence them in the non-increasing order. That is, process the job with the most remaining time first.

(3) Most number of operations remaining rule [9]. Process the job with most remaining operations unprocessed first.

In our EDA, the initial operation sequence of 20% solutions in the population are generated by random rule, 40% solution by most time remaining rule and 40% solutions by most number of operations remaining rule.

D. Probability Model and Updating Mechanism

In this paper, the probability model is designed as two probability matrixes, i.e. the operation probability matrix A_1 and the machine probability matrix A_2. The element $p_{ij}(l)$ of 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. To ensure that the whole solution space can be sampled uniformly, p_{ij} is initialized as $p_{ij}(0) = 1/n$ for all i ($i = 1,2,\cdots,T_o$) and j ($j = 1,2,\cdots,n$).

The element $q_{ijk}(l)$ of 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 appropriateness of a machine processing a certain operation. The probability matrix A_2 is initialized as follows:

$$q_{ijk} = \begin{cases} \dfrac{1}{m_{i,j}}, \text{if } O_{i,j} \text{ can be processed on machine } M_k \\ 0, \text{else} \end{cases}, \forall i, j, k \quad (9)$$

In each generation, the EDA generates the new individuals by sampling according to the probability matrixes A_1 and A_2. In order to generate a new solution, the operation sequence vector should be generated first. For every position i, job J_j is selected with the probability of p_{ij}. If job J_j has already appeared n_j times in the operation sequence vector, it means the processing procedure of job J_j is completed. So, the whole column $p_{1j}, p_{2j}, \cdots, p_{T_o j}$ 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 and then their makespan values are calculated.

Next, it determines the superior sub-population that consists of the best SP solutions, and updates the probability matrixes A_1 and A_2 according to the following equations:

$$p_{ij}(l+1) = (1-\alpha)\, p_{ij}(l) + \frac{\alpha}{i \times SP} \sum_{s=1}^{SP} I_{ij}^{s}, \forall i, j \quad (10)$$

$$q_{ijk}(l+1) = (1-\beta) q_{ijk}(l) + \frac{\beta}{SP} \sum_{s=1}^{SP} \tilde{I}_{ijk}^{s}, \forall i, j, k \quad (11)$$

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

$$I_{ij} = \begin{cases} 1, \text{if job } J_j \text{ appears before or in position } i \\ 0, \text{else} \end{cases} \quad (12)$$

$$\tilde{I}_{ijk} = \begin{cases} 1, \text{if operation } O_{i,j} \text{ is processed on machine } M_k \\ 0, \text{else} \end{cases} \quad (13)$$

E. Local Search Based on Critical Path

To improve the solutions generated by the EDA, a local search procedure is very efficient. For solving the MFJSP, a critical-path-based local search is designed to enhance the local exploitation around the best solution of each generation in this paper.

Denote $S_{i,j}^{E}$ as the earliest starting time of operation $O_{i,j}$ and $S_{i,j}^{L}$ as the latest starting time without delaying the makespan. Thus, the earliest completion time of operation $O_{i,j}$

is $C_{i,j}^E = S_{i,j}^E + t_{i,j,k}$, and the latest completion time is $C_{i,j}^L = S_{i,j}^L + t_{i,j,k}$, where $t_{i,j,k}$ is the processing time of $O_{i,j}$ on machine M_k. Denote $PM_{i,j}^k$ as the operation processed on machine M_k right before the operation $O_{i,j}$ and $SM_{i,j}^k$ as the operation processed on machine M_k right after $O_{i,j}$. Denote $PJ_{i,j} = O_{i,j-1}$ as the operation of job J_i preceding $O_{i,j}$ and $SJ_{i,j} = O_{i,j+1}$ as the operation of job J_i following $O_{i,j}$.

Because the makespan value is no shorter than the flow time of any possible critical path, it may be improved only by moving the critical operations. Denote O_l ($l = 1,2,...,N_c$) as the critical operation to be moved, where N_c is the total number of critical operations. Moving O_l is to delete it from its current position and then insert it at another feasible position. Obviously, the makespan value of the new solution is no larger than the old one. If O_l is assigned before $O_{i,j}$ on machine M_k, it can be started as early as $C^E(PM_{i,j}^k)'$ and can be finished as late as $S_{i,j}^{L}{}'$ 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 $\max\{C^E(PM_{i,j}^k)', C^E(PJ_l)'\} + t_{l,k} <= \min\{S_{i,j}^{L}{}', S^L(SJ_l)'\}$.

The above moving process is repeated until all critical operations are moved. Denote N_l as the number of positions to move O_l feasibly, then the total number of moving neighbors of a solution is $N_{total} = \sum_{l=1}^{N_c} N_l$.

Moving critical operations will obtain new solutions. For the MFJSP, a number of solutions with different W_T or W_M may have the same value of C_M . 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 C_M than the old solution;

(2) the new solution with the same C_M as the old one has smaller W_M ;

(3) the new solution with the same C_M and W_M as the old one has smaller W_T .

Moreover, based on moving operations for the best individual of the population in each generation, the following local search procedure is used.

Step 1: Generate solution *s'* by moving all the critical operations of solution *s*, then let *s'* = *s*.

Step 2: If the total objective value of *s'* is smaller than *s*, then go back to Step 1; else, stop.

F. Procedure of the EDA

With the above design, the flowchart of the EDA for solving the MFJSP is illustrated in Fig. 4.

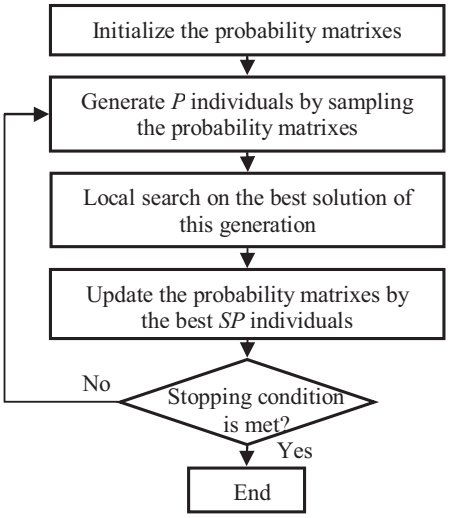

Figure 4. The framework of the EDA for the MFJSP.

In summary, the promising area of the solution space may be found by using the EDA-based estimating and sampling in the initial stage of evolution. Then, to obtain better solutions, the local search strategy is performed in the promising region. The algorithm combines the benefits of EDA and local search to balance global exploration and local exploitation. It stops when the maximum number of generations *Gen* is reached.

G. Computational complexity analysis

For each generation of the EDA, its computational complexity can be roughly analyzed as follow.

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 $O[T_o(SP+m+n) + P\log P]$.

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

It can be seen that the complexity of the proposed EDA is not large.

V. Computational Results and Comparisons

To test the performance of the proposed EDA, numerical simulations are carried out with two well-studied benchmark sets including five Kacem instances [15] and ten BRdata

instances [3]. Same as literature, for each instance we run the algorithm 20 times independently. The algorithm is coded in C++ and run on a 3.2GHz Intel Core i5 processor.

A. Parameters Setting

The difficulty to solve the MFJSP is closely associated with the problem size. Thus, for each instance, the maximum number of generations is set to $Gen = 10 \cdot n \times m$. Besides, the population size is set to $P = \mu \cdot n \times m$ and the number of the superior individuals selected to update the probability model is set to $SP = \eta\% \cdot P$. Except for μ and η, the proposed EDA contains several other key parameters: the learning speed of A_1 (α), and the learning speed of A_2 (β). By using the instance Mk07, we implement the Taguchi method of design-of-experiment (DOE) [30] to investigate the influence of these parameters on the performance of the EDA. Different combinations of these parameter values are listed in Table II.

TABLE II. COMBINATIONS OF PARAMETER VALUES

Parameters	Factor level			
	1	*2*	*3*	*4*
μ	0.5	1	1.5	2
η	10	20	30	40
α	0.1	0.2	0.3	0.4
β	0.1	0.2	0.3	0.4

For each parameter combination, the EDA is run 20 times independently and the response variable (RV) value is the best total objective function value obtained by the EDA in 20 times. 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 RV values are listed in Table III.

According to the orthogonal table, we illustrate the trend of each factor level in Fig. 5. Then, we figure out the response value of each parameter to analyze the significance rank of each parameter. The results are listed in Table IV.

From Table IV it can be seen that among the four parameters, the learning speed β of A_2 is the most significant parameter. It can be assumed that the matrix for machine assignment is crucial to the performance of the EDA. Large value of β could lead to premature convergence while small value could lead to slow convergence. Therefore, an appropriate value of β is very essential to the performance of the EDA. In addition, the significant rank of the learning speed α of A_1 is the second, with the setting principle similar to β. Besides, larger population size P within a certain range can help the EDA enhance the searching capability, but it is no use when P is large enough. Although the number of selected individual SP to update the probability model has the slightest influence to the EDA, a smaller value still can help to build a more accurate model.

TABLE III. ORTHOGONAL ARRAY AND BEST VALUES

Experiment Number	Factor				RV
	μ	η	α	β	
1	1	1	1	1	168.6
2	1	2	2	2	168.4
3	1	3	3	3	168.4
4	1	4	4	4	170.1
5	2	1	2	3	168.55
6	2	2	1	4	168.55
7	2	3	4	1	168.55
8	2	4	3	2	167.75
9	3	1	3	4	168.2
10	3	2	4	3	168.25
11	3	3	1	2	168.25
12	3	4	2	1	168.85
13	4	1	4	2	168.2
14	4	2	3	1	168.5
15	4	3	2	4	168.6
16	4	4	1	3	168.25

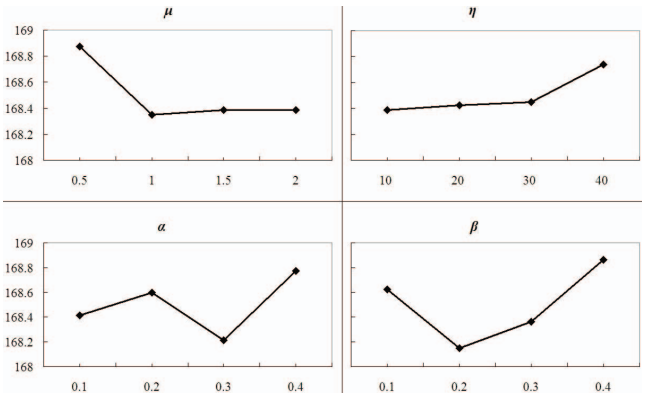

Figure 5. Factor level trend of the EDA.

TABLE IV. RESPONSE VALUE

Level	μ	η	α	β
1	168.875	168.3875	168.4125	168.625
2	168.35	168.425	168.6	168.15
3	168.3875	168.45	168.2125	168.3625
4	168.3875	168.7375	168.775	168.8625
Delta	0.525	0.35	0.5625	0.7125
Rank	3	4	2	1

According to the above analysis, a better choice of the parameter combination is suggested as $P=n\times m$, $SP=10\%P$, $\alpha=0.3$, $\beta=0.2$ and $Gen=10\cdot n\times m$.

B. Results of Kacem Instances

To compare the EDA with Xing's algorithm [18] and HTSA [19], the first five test samples (Kacem instances) with the dimensions ranging from 4 jobs 5 machines to 15 jobs 10 machines are run 20 times independently according to the literature. The results are listed in Table V, where $S_1 \sim S_4$ denote the obtained solutions.

From Table V, it can be seen the EDA is more effective than the other two algorithms. For solving the problem 8×8, the EDA obtains a solution that dominates one of Xing's algorithm. For solving 4×5 and 10×10 instances, the EDA obtains more optimal solutions. For solving other instances, the EDA obtains the same solutions. So, it can be concluded that the EDA is the best one among the three algorithms for solving the five Kacem instances.

C. Results of BRdata Instances

The BRdata instances include ten problems from 10 jobs 6 machines to 20 jobs 15 machines. Table VI illustrates the comparison results with Xing's algorithm [18] and HTSA [19] for solving the BRdata instances.

TABLE V. RESULTS OF THE FIVE KACEM INSTANCES

$n\times m$	Obj.	Xing		HTSA			EDA			
		S_1	S_2	S_1	S_2	S_3	S_1	S_2	S_3	S_4
4×5	C_M	12		11	12		11	11	12	
	W_T	32		32	32		34	32	32	
	W_M	8		10	8		9	10	8	
8×8	C_M	14	15	14	15		14	15		
	W_T	77	76	77	75		77	75		
	W_M	12	12	12	12		12	12		
10×7	C_M	11	11	11	11		11	11		
	W_T	61	62	61	62		61	62		
	W_M	11	10	11	10		11	10		
10×10	C_M	7	8	7	7	8	8	7	7	8
	W_T	42	42	43	42	42	41	43	42	42
	W_M	6	5	5	6	5	7	5	6	5
15×10	C_M	11	11	11	11		11	11		
	W_T	91	93	91	93		91	93		
	W_M	11	10	11	10		11	10		

Form Table VI, it can be seen that our EDA dominates Xing's algorithm in 6 out of 10 instances and the total objective value is better on 9 instances. For solving the instance Mk08, the EDA and Xing's algorithm can both obtain the same optimal solution. Besides, the EDA dominates HTSA on 2 instances and obtain the same optimal solution on other 4 instances. Similarly, the total objective values obtained by our EDA are better than HTSA on 6 out of 10 instances and the same on the other 4 instances. So, it can be concluded that the EDA is more effective in solving the larger scale MFJSP than Xing's algorithm and HTSA.

VI. CONCLUSIONS

In this paper, an effective estimation of distribution algorithm was proposed for solving the multi-objective flexible job-shop scheduling problem with the criteria to minimize the maximum completion time, the total workload of machines and the workload of the critical machine simultaneously. We designed a probability model to generate the new individuals via sampling based on it. We also designed an updating mechanism for the probability model with the superior sub-population. To generate the initial solutions, multiple strategies are used in a combination way. In addition, to enhance the exploitation ability, a critical-path-based local search strategy was employed. Moreover, by using DOE-based testing, the influence of parameter setting was investigated. Simulation tests and comparisons to several existing algorithms demonstrated the effectiveness of the proposed EDA in solving the MFJSP. The future work is to design adaptive EDAs for scheduling problems with uncertainty.

REFERENCES

[1] E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys, "Sequencing and scheduling: Algorithms and complexity," in: Logistics of production and inventory, S. C. Graves Ed. Amsterdam: Elsevier, 1993, pp. 445-522.

[2] P. Bruker, and R. Schlie, "Job-shop scheduling with multi-purpose machines," Computing, vol. 45, no. 4, pp. 369-375, 1990.

[3] P, Brandimarte, "Routing and scheduling in a flexible job shop by tabu search," Annals of Operations Research, vol. 41, no. 3, pp. 157-183, 1993.

[4] S. Dauzere-Peres, and J. Paulli, "An integrated approach for modeling and solving the general multiprocessor job-shop scheduling problem using tabu search," Annals of Operations Research, vol. 70, no. 0, pp. 281-306, 1997.

TABLE VI. RESULTS OF TEN BRDATA INSTANCES

Case	$n\times m$	Xing				HTSA				EDA			
		F	C_M	W_T	W_M	F	C_M	W_T	W_M	F	C_M	W_T	W_M
Mk01	10×6	48.00	42	162	42	**45.75**	40	167	36	**45.75**	40	167	36
Mk02	10×6	34.35	28	155	28	**32.25**	26	151	26	**32.25**	26	151	26
Mk03	15×8	236.4	204	852	204	236.4	204	852	204	**236.3**	204	850	204
Mk04	15×8	82.05	68	352	67	76.25	61	366	61	**76.10**	60	382	60
Mk05	15×4	203.25	177	702	177	**197.75**	172	687	172	**197.75**	172	687	172
Mk06	10×15	91.60	75	431	67	81.2	65	398	62	**80.4**	63	423	59
Mk07	20×5	178.35	150	717	150	167.75	140	695	140	**167.25**	140	685	140
Mk08	20×10	**623.05**	523	2524	523	**623.05**	523	2524	523	**623.05**	523	2524	523
Mk09	20×10	412.35	311	2374	299	407.85	310	2294	301	**407.1**	309	2301	299
Mk10	20×15	314.20	227	1989	221	305.35	214	2053	210	**304.95**	219	1992	201

[5] M. Mastrolilli, and L. M. Gambardella, "Effective neighborhood functions for the flexible job shop problem," J of Scheduling, vol. 3, pp. 3-20, 2000.

[6] M. Saidi-mehrabad, and P. Fattahi, "Flexible job shop scheduling with tabu search algorithms," International J of Advanced Manufacturing Technology, vol. 32, no. 5-6, pp. 563-570, 2007.

[7] J. C. Chen, K. H. Chen, J. J. Wu, and C. W. Chen, "A study of the flexible job shop scheduling problem with parallel machines and reentrant process," International J of Advanced Manufacturing Technology, vol. 39, no. 3-4, pp. 344-354, 2008.

[8] J. Gao, L. Y. Sun, and M. Gen, "A hybrid genetic and variable neighborhood descent algorithm for flexible job shop scheduling problems," Computers & Operations Research, vol. 35, no. 9, pp. 2892-2907, 2008.

[9] F. Pezzella, G. Morganti, and G. Ciaschetti, "A genetic algorithm for the flexible job-shop scheduling problem," Computers & Operations Research, vol. 35, no. 10, pp. 3202-3212, 2008.

[10] F. M. Defersha, and M. Chen, "A parallel genetic algorithm for a flexible job-shop scheduling problem with sequence dependent setups," International J of Advanced Manufacturing Technology, vol. 49, no. 1-4, pp. 263-279, 2010.

[11] M. Yazdani, M. Amiri, and M. Zandieh, "Flexible job-shop scheduling with parallel variable neighborhood search algorithm," Expert Systems with Applications, vol. 37, no. 1, pp. 678-687, 2010.

[12] L. N. Xing, Y. W. Chen, P. Wang, Q. S. Zhao, and J. Xiong, "A knowledge-based ant colony optimization for flexible job shop scheduling problems," Applied Soft Computing, vol. 10, no. 3, pp. 888-896, 2010.

[13] E. Moradi, S. M. T. Fatemi Ghomi, and M. Zandieh, "An efficient architecture for scheduling flexible job-shop with machine availability constraints," International J of Advanced Manufacturing Technology, vol. 51, no. 1-4, pp. 325-339, 2010.

[14] J. Q. Li, Q. K. Pan, P. N. Suganthan, T. J. Chua, "A hybrid tabu search algorithm with an efficient neighborhood structure for the flexible job shop scheduling problem," International J of Advanced Manufacturing Technology, vol. 52, no. 5-8, pp. 683-697, 2011.

[15] I. Kacem, S. Hammadi, and P. Borne, "Pareto-optimality approach for flexible job-shop scheduling problems: hybridization of evolutionary algorithms and fuzzy logic," Mathematics and Computers in Simulation, vol. 60, no. 3-5, pp. 245-276, 2002.

[16] W. J. Xia, and Z. M. Wu, "An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems," Computers & Industrial Engineering, vol. 48, no. 2, pp. 409-425, 2005.

[17] J. C. Tay, and N. B. Ho, "Evolving dispatching rules using genetic programming for solving multi-objective flexible job-shop problems," Computers & Industrial Engineering, vol. 54, no. 3, pp. 453-473, 2008.

[18] L. N. Xing, Y. W. Chen, and K. W. Yang, "An effcient search method for multi-objective flexible job shop scheduling problems," J of Intelligent Manufacturing, vol. 20, no. 3, pp. 283-293, 2009.

[19] J. Q. Li, Q. K. Pan, and Y. C. Liang, "An effective hybrid tabu search algorithm for multi-objective flexible job-shop scheduling problems," Computers & Industrial Engineering, vol. 59, no. 4, pp. 647-662, 2010.

[20] P. Larranaga, and J. A. Lozano, Estimation of distribution algorithms: A new tool for evolutionary computation, Boston: Kluwer Academic Publishers, 2002.

[21] S. D. Zhou, and Z. Q. Sun, "A survey on estimation of distribution algorithms," Acta Automatica Sinica, vol. 33, no. 2, pp. 113-124, 2007.

[22] H. Mühlenbein, and G. Paass, "From recombination of genes to the estimation of distributions I: binary parameters," Lecture Notes in Computer Science, vol. 1141, pp. 178-187, 1996.

[23] G. R. Harik, F. G. Lobo, and D. E. Goldberg, "The compact genetic algorithm," IEEE Transaction on Evolutionary Computation, vol. 3, no. 4, pp. 287-297, 1999.

[24] J. S. De Bonet, C. L. Jr. Isbell, and P. Viola, "MIMIC: Finding optima by estimating probability densities," in: Advances in Neural Information Processing Systems. MIT Press, Cambridge, 1997, pp. 424-430.

[25] S. Baluja, and S. Davies, "Using optimal dependency-trees for combinatorial optimization: Learning the structure of the search space," Proc of the 14th Int Conf on Machine Learning. San Francisco, p. 30-38, 1997.

[26] M. Pelikan, and H. Mühlenbein, "The bivariate marginal distribution algorithm," in: Advances in Soft Computing - Engineering Design and Manufacturing, J. M. Benítez, Ed. London: Springer-Verlag, 1999, pp. 521-535.

[27] H. Mühlenbein, and T. Mahnig, "Convergence theory and applications of the factorized distribution algorithm," J of Computing and Information Technology, vol. 7, no. 1, pp. 19-32, 1999.

[28] G. Harik, "Linkage learning via probabilistic modeling in the ECGA," IlliGAL Report NO. 99010, Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign, Illinois, 1999.

[29] M. Pelikan, D. E. Goldberg, and E. Cantú-Paz, "BOA: The bayesian optimization algorithm," Proc of the Genetic and Evolutionary Computation. San Francisco, p. 525-532, 1999.

[30] D. C. Montgomery, Design and analysis of experiments, Arizona: John Wiley & Sons , 2005.