International Journal of Production Research
Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/tprs20

A Pareto-based estimation of distribution algorithm for the multi-objective flexible job-shop scheduling problem

Ling Wang [a] , Shengyao Wang [a] & Min Liu [a]

[a] Tsinghua National Laboratory for Information Science and Technology, Department of Automation , Tsinghua University , Beijing , 100084 , China

Published online: 04 Mar 2013.

To cite this article: Ling Wang , Shengyao Wang & Min Liu (2013) A Pareto-based estimation of distribution algorithm for the multi-objective flexible job-shop scheduling problem, International Journal of Production Research, 51:12, 3574-3592, DOI: 10.1080/00207543.2012.752588

To link to this article: http://dx.doi.org/10.1080/00207543.2012.752588

International Journal of Production Research, 2013
Vol. 51, No. 12, 3574–3592, http://dx.doi.org/10.1080/00207543.2012.752588

Taylor & Francis
Taylor & Francis Group

A Pareto-based estimation of distribution algorithm for the multi-objective flexible job-shop scheduling problem

Ling Wang, Shengyao Wang* and Min Liu

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

(*Received 23 February 2012; final version received 17 November 2012*)

To solve the multi-objective flexible job-shop problem (MFJSP), an effective Pareto-based estimation of distribution algorithm (P-EDA) is proposed. The fitness evaluation based on Pareto optimality is employed and a probability model is built with the Pareto superior individuals for estimating the probability distribution of the solution space. In addition, a mechanism to update the probability model is proposed, and the new individuals are generated by sampling the promising searching region based on the probability model. To avoid premature convergence and enhance local exploitation, the population is divided into two sub-populations at certain generations according to a splitting criterion, and different operators are designed for the two sub-populations to generate the promising neighbour individuals. Moreover, multiple strategies are utilised in a combination way to generate the initial solutions, and a local search strategy based on critical path is proposed to enhance the exploitation ability. Furthermore, the influence of parameters is investigated based on the Taguchi method of design of experiment, and a suitable parameter setting is suggested. Finally, numerical simulation based on some well-known benchmark instances and comparisons with some existing algorithms are carried out. The comparative results demonstrate the effectiveness of the proposed P-EDA in solving the MFJSP.

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

1. Introduction

The flexible job-shop scheduling problem (FJSP) is an extension of the classical job-shop scheduling problem (JSP) for flexible manufacturing systems, which allows each machine to have the ability to perform more than one type of operation. The FJSP has a wide application background and is very close to the real manufacturing situation. 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 with optimised objectives. The FJSP is more difficult to solve than the classical JSP due to the additional need to determine the assignment of operations to machines. Therefore, it is considered as one of the most difficult problems in combinatorial optimisation (Lawler et al. 1993) in both academic and application fields.

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. Dauzere-Peres and Paulli (1997) proposed a TS algorithm based on an integrated approach, which was improved with two developed neighbourhood functions by Mastrolilli and Gambardella (2000) in terms of computation time and solution quality. Gomes, Barbosa-Póvoab, and Novaisc (2005) presented a new integer linear programming (ILP) model to schedule the flexible job shop. Gao , Sun, and Gen (2008) proposed a genetic algorithm (GA) hybridising with the variable neighbourhood search, and Pezzella, Morganti, and Ciaschetti (2008) proposed a GA integrating different strategies. In addition, Amiri et al. (2010) developed a variable neighbourhood search (VNS) algorithm based on six neighbourhood structures. Yazdani, Amiri, and Zandieh (2010) proposed a parallel VNS algorithm and Xing et al. (2010) proposed a knowledge-based ant colony optimisation algorithm (KBACO). Recently, Wang et al. (2012b) proposed a bi-population-based estimation of distribution algorithm (BEDA) to solve the FJSP.

As for the methods to solve the multi-objective flexible job-shop scheduling (MFJSP), they can be roughly classified into two types: weighting approach and Pareto-based approach. The weighting approach usually solves the MFJSP by transforming the multi-objective problem to a single-objective problem through assigning a different weight for each

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

objective. Aiming at generating a set of Pareto optimal solutions, the Pareto-based approach solves the MFJSP based on the Pareto optimality concept, which helps decision makers select the favourite solution based on the trade-off between objective values. With respect to the weighting approach, Xia and Wu (2005) proposed a hierarchical approach by using particle swarm optimisation (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); Wang, Zhou, and Xi (2008) developed a filtered-beam-search-based heuristic algorithm (named as HFBS), which incorporated dispatching rules-based heuristics and intelligently explored the search space to avoid useless paths; Xing, Chen, and Yang (2009) introduced a local search algorithm; and Li, Pan, and Liang (2010) developed a TS algorithm with an effective neighbourhood structure by combining two adaptive rules. With respect to the Pareto-based approach, Kacem, Hammadi, and Borne (2002) proposed a localisation approach to solve the assignment problem in the MFJSP; Frutos, Olivera, and Tohme (2010) introduced a memetic algorithm based on the non-dominated sorting genetic algorithm II (NSGA-II); Wang et al. (2010) proposed a multi-objective genetic algorithm (MOGA) based on an immune and entropy principle to solve the MFJSP; Moslehi and Mahnam (2011) proposed a multi-objective particle swarm optimisation (MOPSO); Li, Pan, and Gao (2011) presented a Pareto-based discrete artificial bee colony (P-DABC) algorithm; Rabiee, Zandiehb and Ramezania (2012) concluded that the general non-dominated ranked genetic algorithm and Pareto archive evolutionary strategy performed better in comparison with NSGA-II and MOGA when solving the partial flexible job shop problem to minimise makespan and total operation costs; and recently Li, Pan, and Chen (2012) proposed a hybrid Pareto-based local search algorithm to solve the MFJSP.

As a novel kind of evolutionary algorithm based on statistical learning, the estimation of distribution algorithm (EDA) has been increasingly studied and widely applied in recent years (Larranaga and Lozano 2002). According to the complexity of the model, the EDA can be classified as a univariate model, bivariate model or multivariate model. The population-based incremental learning (PBIL) (Baluja 1994), univariate marginal distribution algorithm (UMDA) (Mühlenbein and Paass 1996) and compact GA (CGA) (Harik, Lobo, and Goldberg 1998) are univariate models, while mutual information maximisation for input clustering (MIMIC) (De Bonet, Isbell, and Viola 1997), combining optimisers with mutual information trees (COMIT) (Baluja and Davies 1997) and bivariate marginal distribution algorithm (BMDA) (Pelikan and Mühlenbein 1999) are bivariate models. The factorised distribution algorithms (FDA) (Mühlenbein and Mahnig 1999), extended compact GA (ECGA) (Harik 1999) and Bayesian optimisation algorithm (BOA) (Pelikan, Goldberg, and Cantú-Paz 1999) are multivariate models. For more details about the EDA, please refer to Larranaga and Lozano (2002).

So far, the EDA-based algorithms have 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), the multi-dimensional knapsack problem (Wang, Wang, and Fang 2012a) and so on. However, to the best of our knowledge, there has been no research work about the EDA for solving the MFJSP. In this paper, we propose a Pareto-based EDA (P-EDA) to solve the MFJSP with the criteria to simultaneously minimise the maximum completion time, the total workload of machines and the workload of the critical machine. According to the fitness evaluation based on Pareto optimality, a probability model is built with the Pareto superior individuals for generating new solutions. Moreover, an updating mechanism for the probability model is proposed. To stress the balance of global exploration and local exploitation, the population is divided into two sub-populations at certain generations according to a splitting criterion, and different searching operators are designed for two sub-populations. In addition, multiple strategies are employed in a combined way to generate initial solutions, and a critical path-based local search is designed to enhance the exploitation. After investigating the influence of parameter settings based on the design of experiment testing, we carry out numerical simulation with two sets of benchmark instances for performance testing and comparison.

The remainder of the paper is organised as follows: in Section 2, the MFJSP is formulated. In Section 3, the basic EDA is introduced briefly and then the framework of the P-EDA for solving the MFJSP is proposed. The influence of parameter settings is investigated based on the design of experiment testing in Section 4, and computational results and comparisons are given as well. Finally we end the paper with conclusions in Section 5.

2. Multi-objective flexible job-shop scheduling problem

2.1 *Basic concepts of multi-objective optimisation*

Generally, a multi-objective optimisation problem can be described as follows:

$$\text{Minimise} \quad y = f(x) = (f_1(x), f_2(x), ..., f_q(x)) \tag{1}$$

where $x \in \Omega$ is the decision vector in space Ω, $y \in R^q$ is the objective vector with $q(q > 1)$ objectives.

The following concepts in the multi-objective optimisation are also adopted in this paper.

Pareto dominance: A solution a is said to dominate solution b if and only if $\forall i \in \{1,2,\ldots,q\}: f_i(a) \leqslant f_i(b)$ and $\exists i \in \{1,2,\ldots,q\}: f_i(a) < f_i(b)$.

Pareto optimality: A feasible solution a is optimal in the Pareto sense if it is not dominated by any other solution in the feasible space.

Archive set: An archive set AS is used to collect all the Pareto optimal solutions, which form the Pareto optimal frontier in the objective space. The archive set is iteratively updated during the searching process.

Non-dominated sorting: Non-dominated sorting is used to divide the solutions into several levels according to the dominance degree. The first frontier contains all the Pareto optimal solutions, and the second frontier contains all the solutions that are dominated by the individuals in the first frontier only, and so on. Every solution in each frontier is assigned a rank value. Solutions in the first frontier are given a rank value of 1, and the ones in the second frontier are assigned a rank value of 2, and so on. Please refer to Deb et al. (2000) for the detailed implementation.

Crowding distance: Crowding distance is a measure of how close a solution is to its neighbours with the same rank value. A set of solutions with larger crowding distance represents better diversity.

2.2 *Formulation of MFJSP*

The flexible job-shop scheduling problem (FJSP) is commonly described 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 $C_{i,j}$ be the completion time of operation $O_{i,j}$. The FJSP is to determine both the assignment of machines and the sequence of operations on all the machines to minimise a certain scheduling objective function.

As in the literature (Kacem, Hammadi, and Borne 2002; Xia and Wu 2005; Tay and Ho 2008; Xing, Chen, and Yang 2009; Li, Pan, and Liang 2010; Li, Pan, and Gao 2011; Frutos, Olivera, and Tohme 2010; Wang, Olivera, and Mahnam 2011), we consider the multi-objective FJSP with the following three objectives to be minimised: (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 assigning the machine with relatively small processing time to improve economic efficiency; and (3) the maximal machine workload, i.e. W_M, which considers the workload balance among all machines to prevent too much work been assigned to a single machine.

Mathematically, the MFJSP can be formulated as follows (Li, Pan, and Liang 2010):

$$\text{Minimise } C_M = \max_{1 \leqslant i \leqslant n} \{C_{i,n_i}\} \tag{2}$$

$$\text{Minimise } W_T = \sum_{k=1}^{m} \sum_{i=1}^{n} \sum_{j=1}^{n_i} t_{i,j,k} x_{i,j,k} \tag{3}$$

$$\text{Minimise } W_M = \max_{1 \leqslant k \leqslant m} \left\{ \sum_{i=1}^{n} \sum_{j=1}^{n_i} t_{i,j,k} x_{i,j,k} \right\} \tag{4}$$

$$\text{Subject to } C_{i,j} - C_{i,j-1} \geqslant t_{i,j,k} x_{i,j,k}, j = 2, 3, \cdots, n_i; \forall i, k \tag{5}$$

$$\sum_{k \in M_{i,j}} x_{i,j,k} = 1, \forall i, j \tag{6}$$

$$x_{i,j,k} = \begin{cases} 1, \text{if operation } O_{i,j} \text{ is processed on machine } k \\ 0, \text{otherwise} \end{cases} \tag{7}$$

$$C_{i,j} \geq 0, \forall i, j \tag{8}$$

where, Equation (5) ensures that the operations belonging to the same job satisfy the precedence constraints, and Equation (6) states that one machine must be selected from the set of available machines for each operation.

3. The P-EDA for MFJSP

In this section, we will introduce the basic EDA briefly and then propose a Pareto-based estimation of distribution algorithm with a special probability model and an updating mechanism based on the Pareto optimality to solve the MFJSP.

Compared to the BEDA for solving the FJSP (Wang et al. 2012b), the P-EDA proposed here for solving the multi-objective FJSP is different. First, the BEDA updates the probability model with some best solutions of the population. However, such a procedure cannot be implemented for solving the multi-objective optimisation problem since it is unable to determine the set of the best solutions. Thus, in the P-EDA the concept of Pareto superior is proposed for EDA to update the probability model with the Pareto superior population. Second, the splitting and combination criteria in the BEDA are the best solution found so far has no improvement after some generations, which cannot be implemented when solving the MFJSP. Thus, the P-EDA uses an archive set AS to reserve good solutions and uses different splitting and combination criteria related to the AS. Third, for the searching procedures of the two sub-populations and local search stage, the strategies to reserve solutions are also different in the P-EDA and the BEDA.

3.1 *The basic EDA*

The estimation of distribution algorithm is a relatively new paradigm in the field of evolutionary computation, which employs explicit probability distributions in optimisation (Larranaga and Lozano 2002). Different from GA that reproduces new populations with the crossover and mutation operators, the EDA does it implicitly. With the statistical analysis tool, 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 searching 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 framework 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 updating process reflects the evolutionary trend

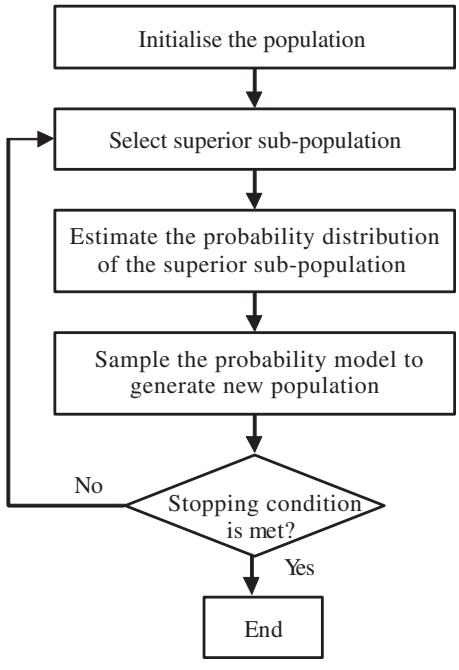

Figure 1. The general framework of the EDA.

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.

3.2 *Solution representation*

Every individual of the population is a solution of the MFJSP, 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 consists of two vectors corresponding to the two sub-problems of the MFJSP, i.e. operation sequence vector and machine assignment vector.

For the operation sequence vector, the number of elements equals 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 four jobs and four machines, as shown in Table 1. Figure 2 illustrates the representation of a feasible solution.

The operation sequence and machine assignment of the solution in Figure 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 Figure 3.

3.3 *Population initialisation*

To guarantee the initial population with a certain quality and diversity, some different strategies are utilised in a hybrid way to generate the initial solutions at the beginning of the evolution.

We apply the following two rules to generate the initial machine assignments. (1) Random rule. Randomly select a machine for each operation from the set of candidate machines and then place it at the position in the machine assignment vector. (2) Global minimum processing time rule (Pezzella, Morganti, and Ciaschetti 2008). In our algorithm, for the initial machine assignments, 40% of solutions are generated by the random rule, and the other 60% of solutions in the population are generated by the global minimum processing time 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. To generate initial operation sequences, the following three rules are applied. (1) Random rule. Randomly generate the sequence of the operations on each machine. (2) Most time remaining rule (Pezzella, Morganti, and Ciaschetti 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, Morganti, and Ciaschetti 2008). The job with most remaining operations unprocessed has a high priority to be selected. In our algorithm, for the initial operation sequences 20% of solutions in the population are generated by the random rule, 40% of solutions are generated by the most time remaining rule, and the other 40% of solutions are generated by the most number of operations remaining rule.

3.4 *Probability model and updating mechanism*

Compared with GA, the EDA produces offspring by sampling according to a probability model instead of crossover and mutation operators. So the probability model has a great effect on the performances of the EDA. In the P-EDA, the

Table 1. Processing time table.

	Machines			
Operations	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

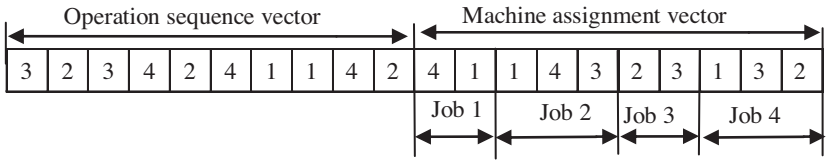

Figure 2. Illustration of the representation of a feasible solution.

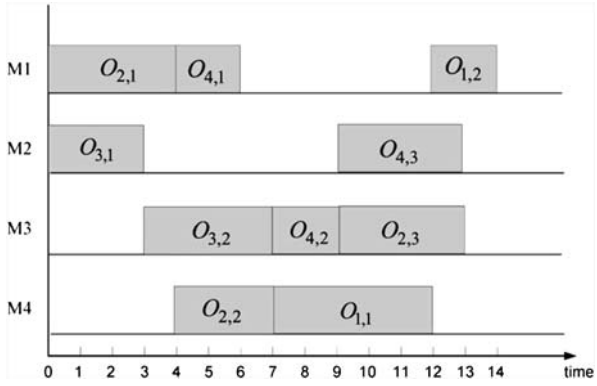

Figure 3. Gantt chart of the solution shown in Figure 2.

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 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 $(i = 1, 2, \cdots, T_o)$ 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 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:

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

Via sampling according to the probability matrixes A_1 and A_2, it generates new individuals. 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 $p_{1j}, p_{2j}, \cdots, p_{T_o j}$ of the probability matrix A_1 will be set as zero. Similarly, the machine assignment vector is generated according to the probability matrix A_2. In such a way, P individuals are generated.

Next, it determines the Pareto superior population that consists of Ps solutions. Firstly, the non-dominated sorting is applied to the population and the Pareto archive set AS is updated by using the solutions in the first Pareto level frontier. Then, the crowding distance of each individual is calculated in the following way: sequence the members with the same rank value in ascending order according to each objective value, and calculate the crowding distance of each individual as the sum of the distance between its right and left neighbours in the sequence. As for the first individual and the last individual, their crowding distances are defined as infinity. The pseudo code to calculate the crowding distance is illustrated in Figure 4.

To reduce the computational complexity, there is no use in sorting or calculating the other individuals in the population if the number of calculated individuals is larger than Ps. Based on the rank and crowding distance, we say solution a is better than b if $rank(a) < rank(b)$ or $(rank(a) = rank(b) \,\&\, d(a) > d(b))$. By this means, the best Ps solutions are determined as the Pareto superior population.

The probability matrixes A_1 and A_2 are then updated according to the following equations:

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

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

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

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

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

3.5 *Population-splitting mechanism*

The simple EDA pays more attention to global exploration while its exploitation capability is limited. Meanwhile, the stagnation often appears after some generations in the evolving process of EDA which is a common problem for the EDA that produces new solutions purely by using the probability models (Chen et al. 2010). To improve the searching capability of an EDA, it should balance the exploration and the exploitation abilities and maintain the population diversity as well.

As the MFJSP consists of two sub-problems (operation routing and machine scheduling), it is an alternative method to optimise the objective function by adjusting the operation sequence and machine assignment separately. From another point of view, it may be unnecessary to simultaneously change the operation sequence vector and machine assignment vector of a certain solution in each generation. In our P-EDA, the population with P individuals is divided into two sub-populations at certain generations according to a splitting criterion. In particular, if the archive set AS keeps fixed at consecutive *ter* generations of EDA-based exploration (i.e., the splitting criterion), the population will be divided into two sub-populations SP1 and SP2 with a size of P/2. To guarantee the balance of the two sub-populations, all the individuals are assigned to SP1 or SP2 randomly. Then, SP1 and SP2 use different operators (see Subsections 3.5.1 and 3.5.2) to adjust the machine assignment and operation sequence, respectively. Once AS keeps unchanged during the next

Procedure Calculate_crowding_distance
// Q is the set of the solutions in a certain rank;
// N = the size of Q;
For each solution X in Q
  $d(X) = 0$;
// $d(\ )$ denotes the crowding distance;
For each objective m
  Q' = sort Q with mth objective in ascending order;
// $Q'(i)$ is the ith solution in current solution set;
  $d(Q'(1))$ = infinite;
  $d(Q'(N))$ = infinite;
  For $i = 2$ to $N - 1$
    $d(i) = d(i) + Q'(i+1).\text{m} - Q'(i-1).\text{m}$;
// $Q'(i)$.m denotes the value of mth objective for $Q'(i)$.

Figure 4. Procedure of calculating the crowding distance.

consecutive *ter* generations of local exploitation (i.e. the combination criterion), SP1 and SP2 will be recombined together as an entire population for further evolution.

3.5.1. *Operator for SP1*

The sub-population SP1 only adjusts the machine assignment. For two certain individuals in SP1, the searching procedure is designed 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: Reserve the two better solutions in the sense of Pareto optimality among all the new and old individuals. If the new one and the old one are not dominated by each other, it reserves 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.

3.5.2. *Operator for SP2*

The sub-population SP2 only adjusts the operation sequence. For two certain 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: Reserve the two better solutions in the sense of Pareto optimality among all the new and old individuals. If the new one and the old one are not dominated by each other, it reserves the new one.

An example of Steps 1 and 2 above is illustrated in Figure 5.

3.6 *Local search based on critical path*

It is widely accepted that a local search procedure is efficient in improving the solutions generated by evolutionary algorithms. In the P-EDA, a local search based on critical path is designed to enhance the local exploitation around the solutions in the first Pareto level frontier.

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 ($l = 1, 2, ..., N_c$) 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

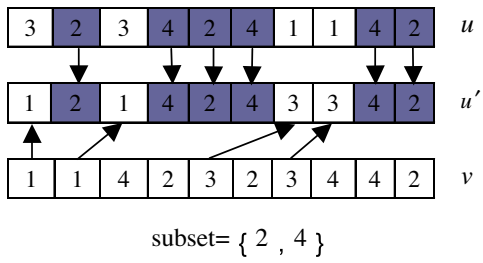

Figure 5. Illustration of searching operator for SP2.

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 $\max\{C^E(PM^k_{i,j})', C^E(PJ_l)'\} + t_{l,k} <= \min\{S^{L'}_{i,j}, S^L(SJ_l)'\}$.

The above moving process is repeated until all critical operations are moved. Let N_l be the number of positions to move O_l feasibly, then the total number of moving neighbours of a solution is $N_{total} = \sum_{l=1}^{N_c} N_l$. Moving critical operations will obtain new solutions. In our algorithm, the new solution will replace the old one if it is not dominated by the old one. In addition, if the two solutions are not dominated by each other and have different objectives, then it updates AS with the new solution when possible.

3.7 *Procedure of the P-EDA*

With the above design, the procedure of the P-EDA for solving the MFJSP is illustrated in Figure 6.

It can be seen that, at each generation, the P-EDA mainly includes two phases, the global exploration phase and the local exploitation phase. At the global exploration phase, the EDA-based searching mechanism is performed. That is, a probability model is built with the Pareto superior individuals of the entire population to generate the new individuals. At the local exploitation phase, local search operators are performed. That is, two sub-populations apply different operators to generate neighbour individuals and the critical path-based local search is for further exploitation. The splitting criterion and the combination criterion are used for switching different phases. The algorithm stops when the maximum number of generations *Gen* is reached. Since the global exploration and the local exploitation are balanced and especially the local exploitation is enhanced, the P-EDA may be of more powerful capability for solving the MFJSP.

4. Computational results and comparisons

To test the performance of the proposed P-EDA, numerical simulations are carried out by using two sets of widely used benchmarks, including five Kacem instances (Kacem, Hammadi, and Borne 2002) and 10 BRdata instances (Brandimarte 1993). The algorithm is coded in C and run on a Thinkpad T420 with a 2.3-GHz processor and 2 GB RAM.

4.1 *Parameter setting*

It is generally accepted that the difficulty in solving the MFJSP is closely associated with 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 as $P_S = \eta\% \cdot P$. What's more, the P-EDA 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 Mk04. Combinations of different values of these parameters are listed in Table 2.

For each parameter combination, the P-EDA is run 20 times independently. The non-dominated solutions among all the solutions obtained by each run are collected as the reference set RS^*. Clearly, the more solutions in RS^* obtained by the combination of parameters, the better the combination is. Thus, the average response variable (ARV) value is the average percentage of non-dominated solutions for each combination in RS^*. 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 treatments is 16, the number of parameters is four and the number of factor levels is four. 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 Figure 7. 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.

From Table 4 it can be seen that the parameter *ter* is the most significant parameter among the four parameters. That is, the parameter *ter* is crucial to the P-EDA for balancing the exploration and exploitation. A smaller value of *ter* makes the algorithm switch between splitting and recombination frequently so as to cause insufficient exploration and exploitation. On the contrary, a large value of *ter* cannot help the population maintain diversity. The significant rank of the learning rate β of A_2 is the second. A large value of β could lead to premature convergence. Furthermore, the learning rate α of A_1 ranks third. A small value of α could lead to slow convergence while a

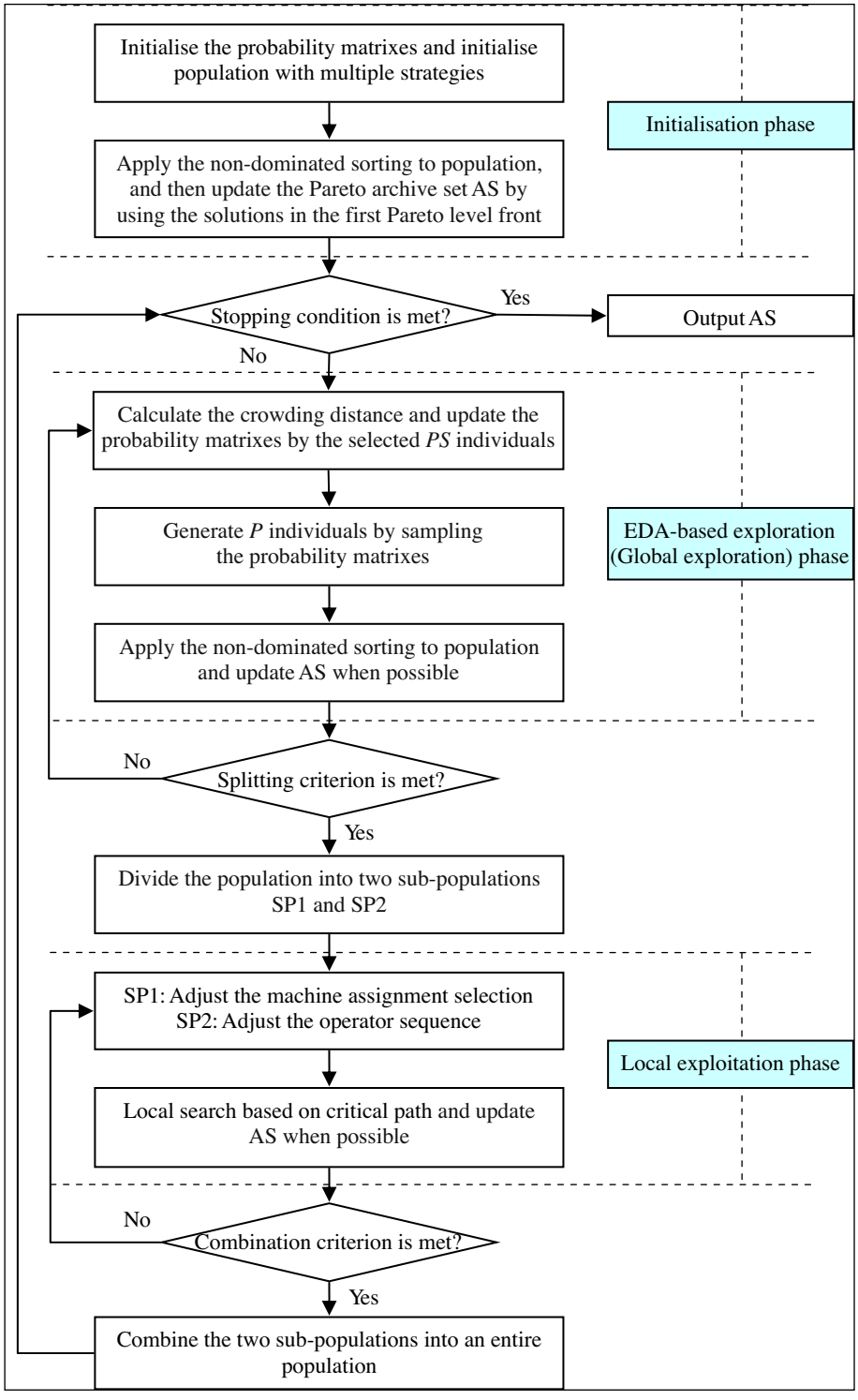

Figure 6. The framework of the P-EDA for the MFJSP.

large value could lead to premature convergence. Although the number of selected individual *Ps* to update the probability model has the slightest influence, an appropriate value still can help the algorithm build an accurate model. According to the above analysis, a good choice of parameter combination is suggested as $\eta = 20$, $\alpha = 0.3$, $\beta = 0.1$ and $ter = 30$.

Table 2. Combinations of parameter values.

	Factor level			
Parameters	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

Table 3. Orthogonal array and ARV values.

	Factor				
Experiment number	μ	α	β	*ter*	ARV (%)
1	1	1	1	1	29.17
2	1	2	2	2	41.67
3	1	3	3	3	33.33
4	1	4	4	4	29.17
5	2	1	2	3	37.5
6	2	2	1	4	37.5
7	2	3	4	1	33.33
8	2	4	3	2	37.5
9	3	1	3	4	33.33
10	3	2	4	3	29.17
11	3	3	1	2	45.83
12	3	4	2	1	29.17
13	4	1	4	2	29.17
14	4	2	3	1	33.33
15	4	3	2	4	37.5
16	4	4	1	3	33.33

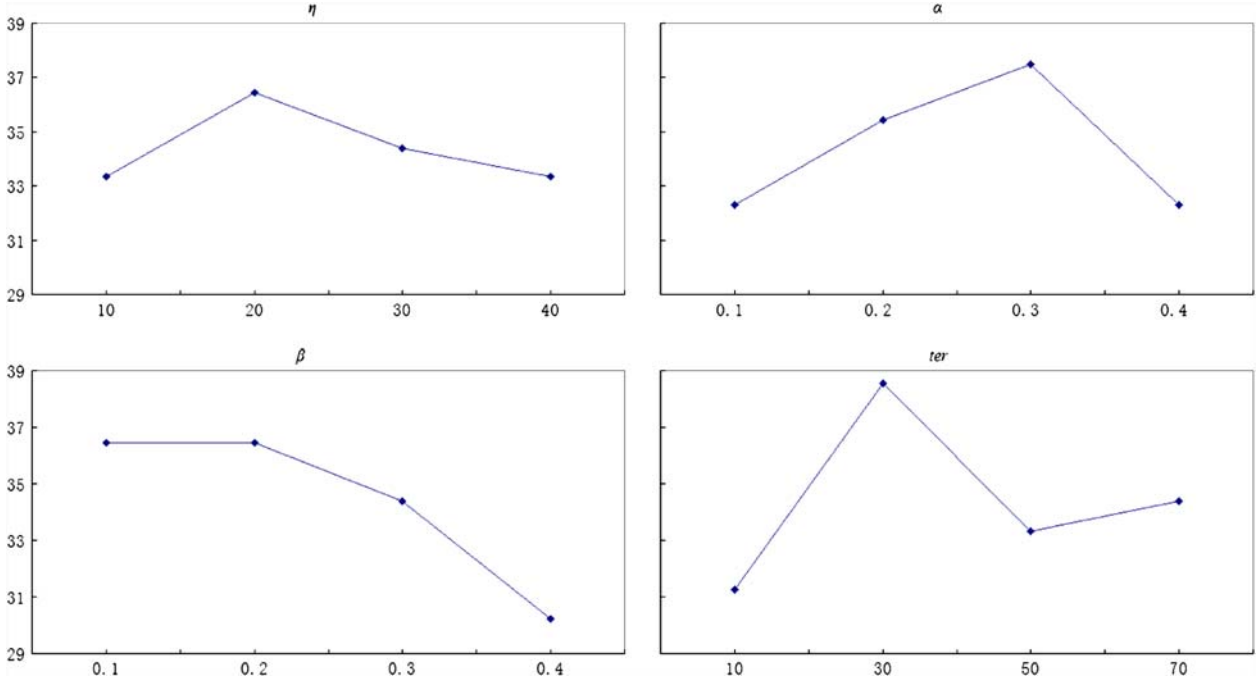

Figure 7. Factor level trend of the P-EDA.

Table 4. Response value.

Level	η	α	β	*ter*
1	33.335	32.2925	36.4575	31.25
2	36.4575	35.4175	36.46	38.5425
3	34.375	37.4975	34.3725	33.3325
4	33.3325	32.2925	30.21	34.375
Delta	3.125	5.205	6.25	7.2925
Rank	4	3	2	1

Table 5. Results of the five Kacem instances.

		PSO + SA		TS			MOGA				P-DABC			P-EDA			
n × m	Obj.	S1	S2	S1	S2	S3	S1	S2	S3	S4	S1	S2	S3	S1	S2	S3	S4
Case 1 (4×5)	C_M			12	11		11	11	12		11	12	13	11	11	12	13
	W_T			32	32		32	34	32		32	32	33	32	34	32	33
	W_M			8	10		10	9	8		10	8	7	10	9	8	7
Case 2 (8×8)	C_M	15	16	14	15		15	15	16		14	15	16	14	15	16	16
	W_T	75	73	77	75		81	75	73		77	75	73	77	75	73	77
	W_M	12	13	12	12		11	12	13		12	12	13	12	12	13	11
Case 3 (10×7)	C_M			11	11						12	11	12	11	11	12	
	W_T			62	61						61	63	60	61	62	60	
	W_M			10	11						11	11	12	11	10	12	
Case 4 (10×10)	C_M	7		7	7	8	8	7	8	7	8	7	8	7	7	8	8
	W_T	44		43	42	42	42	42	41	45	41	43	42	42	43	41	42
	W_M	6		5	6	5	5	6	7	5	7	5	5	6	5	7	5
Case 5 (15×10)	C_M	12		11	11		11	12	11		12	11		11	11		
	W_T	91		91	93		91	95	98		91	93		91	93		
	W_M	11		11	10		11	10	10		11	11		11	10		

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

Instance	$n \times m$	TS[a]	MOGA[b]	P-DABC[c]	P-EDA[d]
Case 1	4 × 5	0.15	5.8	0.05	0.01
Case 2	8 × 8	3.08	9.5	2.36	0.89
Case 3	10 × 7	2.58	n/a	4.25	1.09
Case 4	10 × 10	3.12	14.2	13.56	3.03
Case 5	15 × 10	25.13	87.5	50.89	14.79

[a]Pentium IV 1.7-GHz CPU.
[b]2-GHz CPU.
[c]Pentium IV 1.8-GHz CPU.
[d]Core i5 2.3-GHz CPU.

Table 7. Results of instance MK01.

MOGA				P-EDA			
No.	C_M	W_T	W_M	No.	C_M	W_T	W_M
1	42	158	39	1	40	165	37
2	44	154	40	2	**40**	**167**	**36**
3	43	155	40	3	41	160	38
4	40	169	36	4	41	163	37
				5	42	157	40
				6	42	158	39
				7	42	165	36
				8	43	155	40
				9	44	154	40
				10	46	153	46
				11	47	153	42

Table 8. Results of instance MK02.

MOGA				P-EDA			
No.	C_M	W_T	W_M	No.	C_M	W_T	W_M
1	26	151	26	1	26	151	26
2	27	146	27	2	**27**	**145**	**27**
3	29	143	29	3	28	144	28
4	31	141	31	4	29	143	29
5	33	140	33	5	30	142	30
6	29	145	27	6	31	141	31
				7	31	150	26
				8	33	140	33

Table 9. Results of instance MK03.

MOGA				P-EDA			
No.	C_M	W_T	W_M	No.	C_M	W_T	W_M
1	204	855	199	1	204	850	204
2	204	871	144	2	**210**	**848**	**210**
3	204	882	135	3	213	844	213
4	204	884	133	4	221	842	221
5	213	850	199	5	222	838	222
6	214	849	210	6	231	834	231
7	221	847	199	7	240	832	240
8	222	848	199	8	249	830	249
9	231	848	188	9	266	828	258
10	230	848	177				

4.2 *Results of Kacem instances*

By using the five Kacem instances with a scale ranging from four jobs, five machines to 15 jobs, 10 machines, we compare the P-EDA with the existing weighted approaches PSO + SA (Xia and Wu 2005), TS (Li, Pan, and Liang 2010) and Pareto-based approaches MOGA (Wang et al. 2010) and PDABC (Li, Pan, and Gao 2011). The results are listed in

Table 10. Results of instance MK04.

MOGA				P-EDA			
No.	C_M	W_T	W_M	No.	C_M	W_T	W_M
1	66	345	63	1	**60**	**382**	**60**
2	65	362	63	2	**61**	**366**	**61**
3	63	371	61	3	62	379	60
4	62	373	61	4	**63**	**362**	**62**
5	61	382	60	5	**64**	**355**	**62**
6	60	390	59	6	64	365	61
7	73	350	55	7	**65**	**348**	**63**
8	74	349	54	8	67	344	66
9	74	348	55	9	69	343	67
10	90	331	76	10	72	340	72
				11	72	355	62
				12	78	337	78
				13	84	334	84
				14	98	330	98
				15	106	329	106
				16	114	328	114

Table 11. Results of instance MK05.

MOGA				P-EDA			
No.	C_M	W_T	W_M	No.	C_M	W_T	W_M
1	173	683	173	1	173	683	173
2	175	682	175	2	175	682	175
3	183	677	183	3	178	680	178
4	185	676	185	4	179	679	179
5	179	679	179	5	183	677	183
				6	185	676	185
				7	190	687	172
				8	191	675	191
				9	197	674	197
				10	203	673	203
				11	209	672	209

Table 5, where the results of the comparative algorithms are directly from the literature and all the solutions dominated by others are marked with underlines.

From Table 5, it can be seen that the P-EDA is more effective in solving the five Kacem instances. For example, for Case 1 and Case 2, the P-EDA obtains more non-dominated solutions than the other algorithms. When solving Case 3, the P-EDA obtains three non-dominated solutions while the TS algorithm only obtains two. The P-DABC also obtains three,

Table 12. Results of instance MK06.

MOGA				P-EDA							
No.	C_M	W_T	W_M	No.	C_M	W_T	W_M	No.	C_M	W_T	W_M
1	62	424	55	1	63	423	59	32	77	396	57
2	65	417	54	2	64	411	62	33	78	359	73
3	60	441	58	3	65	405	62	34	78	368	65
4	62	440	60	4	66	396	62	35	79	359	72
5	76	362	60	5	69	376	63	36	79	391	58
6	76	356	74	6	71	371	66	37	80	400	56
7	78	361	60	7	71	390	62	38	81	358	73
8	73	360	72	8	72	368	70	39	81	388	59
9	72	361	72	9	72	370	68	40	82	354	77
10	100	330	90	10	72	374	65	41	82	357	73
				11	72	384	61	42	82	399	56
				12	73	365	68	43	83	395	57
				13	73	368	66	44	84	352	79
				14	73	371	64	45	84	355	75
				15	73	379	60	46	84	360	70
				16	73	398	59	47	84	394	57
				17	74	362	71	48	84	397	56
				18	74	366	67	49	87	351	82
				19	74	373	63	50	87	353	78
				20	74	375	61	51	88	351	79
				21	74	396	59	52	88	354	76
				22	75	363	69	53	88	358	72
				23	75	372	63	54	89	352	78
				24	75	391	59	55	89	353	77
				25	75	392	58	56	90	349	82
				26	76	362	70	57	90	350	80
				27	76	373	62	58	90	461	55
				28	76	389	59	59	92	346	83
				29	76	417	57	60	96	348	81
				30	76	425	56	61	100	348	88
				31	77	360	71				

Table 13. Results of instance MK07.

MOGA				P-EDA			
No.	C_M	W_T	W_M	No.	C_M	W_T	W_M
1	139	693	139	1	139	693	139
2	**140**	**686**	**138**	2	140	689	140
3	144	673	144	3	143	683	143
4	151	667	151	4	144	673	144
5	157	662	157	5	150	669	150
6	162	659	162	6	151	667	151
7	166	657	166	7	156	664	156
				8	157	662	157
				9	161	660	161
				10	162	659	162
				11	166	657	166
				12	166	686	142
				13	166	657	166
				14	175	655	175
				15	187	653	187
				16	202	651	202
				17	217	649	217

Table 14. Results of instance MK08.

MOGA				P-EDA			
No.	C_M	W_T	W_M	No.	C_M	W_T	W_M
1	**523**	**2524**	**515**	1	523	2524	523
2	523	2534	497	2	524	2519	524
3	524	2519	524	3	533	2514	533
4	578	2489	578	4	542	2509	542
5	587	2484	587	5	551	2504	551
				6	560	2499	560
				7	569	2494	569
				8	578	2489	578
				9	587	2484	587

but its solution (12, 61, 11) is dominated by (11, 61, 11), obtained by the P-EDA, and its solution (11, 63, 11) is dominated both by (11, 61, 11) and (11, 62, 10), obtained by the P-EDA. The comparative results are similar in Cases 4 and 5.

In addition, the average CPU time of 20 different runs is listed in Table 6. It can be seen that the P-EDA is the most efficient one among all the algorithms. For the large scale problem Case 5, it can be solved in an average of 14.79 s.

4.3 *Results of BRdata instances*

Next, we carry out tests with BRdata instances. The BRdata instances include 10 problems with scales from 10 jobs, 6 machines to 20 jobs, 15 machines. The comparative results to the MOGA (Wang et al. 2010) are listed in Tables 7–16, where better Pareto optimal solutions are in bold and the solutions dominated by the other algorithm are marked with underlines.

From Tables 7–16, it can be seen that the P-EDA outperforms the MOGA in solving BRdata instances. In six out of 10 instances, the P-EDA can obtain better Pareto optimal solutions that dominate solutions obtained by MOGA. Only in two out of 10 instances are some solutions obtained by the P-EDA dominated by the solutions obtained by the MOGA. As for the number of non-dominated solutions, the P-EDA outperforms MOGA in nine out of 10 instances. The only instance where MOGA obtains more non-dominated solutions than the P-EDA is MK03. Nevertheless, when solving MK03, the solution (214, 849, 210), obtained by MOGA, is dominated by (210, 848, 210), obtained by the P-EDA. In addition, for larger scale instances, i.e., MK09 and MK10, the P-EDA can find many more non-dominated solutions, a good number of which can dominate the solutions found by MOGA.

Table 15. Results of instance MK09.

MOGA				P-EDA							
No.	C_M	W_T	W_M	No.	C_M	W_T	W_M	No.	C_M	W_T	W_M
1	311	2290	299	1	**309**	**2301**	**299**	40	338	2241	326
2	310	3514	299	2	**311**	**2282**	**299**	41	338	2244	323
3	311	2287	301	3	312	2278	306	42	340	2240	327
4	314	2315	299	4	313	2269	304	43	341	2238	334
5	315	2283	299	5	313	2273	302	44	343	2238	332
6	332	2265	302	6	314	2265	312	45	343	2239	328
7	329	2266	301	7	316	2279	299	46	344	2237	334
8	328	2259	308	8	318	2263	310	47	346	2235	342
9	325	2275	299	9	319	2277	299	48	346	2236	334
				10	320	2255	315	49	346	2237	333
				11	321	2256	312	50	346	2273	299
				12	321	2260	310	51	347	2234	340
				13	321	2268	308	52	348	2233	340
				14	322	2266	307	53	349	2238	331
				15	323	2253	316	54	350	2235	339
				16	324	2252	320	55	351	2234	339
				17	324	2264	309	56	352	2235	334
				18	324	2268	304	57	353	2232	342
				19	325	2250	320	58	353	2236	333
				20	325	2254	315	59	353	2237	332
				21	325	2255	314	60	356	2231	340
				22	326	2248	320	61	356	2234	334
				23	326	2253	315	62	356	2235	333
				24	326	2264	308	63	357	2232	339
				25	326	2272	303	64	359	2230	342
				26	327	2252	316	65	359	2236	332
				27	327	2263	309	66	364	2228	346
				28	327	2272	302	67	366	2227	346
				29	328	2247	320	68	367	2229	342
				30	329	2266	304	69	367	2230	340
				31	330	2246	321	70	371	2226	358
				32	330	2265	307	71	376	2226	346
				33	330	2271	303	72	385	2225	374
				34	331	2244	326	73	391	2225	364
				35	332	2243	327	74	393	2224	374
				36	332	2274	299	75	396	2222	374
				37	333	2242	326	76	396	2224	358
				38	333	2245	322	77	399	2223	364
				39	335	2239	332				

Table 16. Results of instance MK10.

MOGA				P-EDA							
No.	C_M	W_T	W_M	No.	C_M	W_T	W_M	No.	C_M	W_T	W_M
1	224	1980	219	1	214	2043	212	41	**243**	**1877**	**215**
2	225	1976	211	2	219	1992	201	42	**243**	**1888**	**212**
3	233	1919	214	3	**222**	**1940**	**219**	43	**243**	**1894**	**211**
4	235	1895	225	4	223	1936	219	44	**244**	**1875**	**218**
5	235	1897	218	5	224	1933	219	45	**245**	**1867**	**225**
6	240	1905	215	6	**225**	**1924**	**209**	46	**245**	**1894**	**210**
7	240	1888	216	7	226	1923	212	47	**245**	**1903**	**209**
8	242	1913	214	8	227	1920	216	48	**246**	**1869**	**220**
9	246	1896	215	9	229	1907	225	49	**246**	**1893**	**210**
10	252	1884	224	10	**229**	**1916**	**212**	50	**247**	**1865**	**225**
11	256	1919	211	11	230	1904	225	51	**247**	**1891**	**211**
12	260	1869	244	12	230	1911	215	52	**248**	**1863**	**225**
13	266	1864	254	13	**230**	**1913**	**212**	53	**250**	**1868**	**224**
14	268	1858	264	14	231	1905	220	54	**251**	**1862**	**235**
15	276	1857	256	15	231	1906	216	55	**251**	**1866**	**224**
16	281	1854	268	16	**231**	**1911**	**212**	56	**251**	**1883**	**212**
17	217	2064	207	17	**231**	**1913**	**211**	57	**251**	**1887**	**211**
18	214	2082	204	18	**232**	**1909**	**211**	58	**253**	**1861**	**227**
				19	**233**	**1895**	**220**	59	**254**	**1917**	**205**
				20	**233**	**1904**	**210**	60	**255**	**1890**	**210**
				21	**233**	**1911**	**208**	61	**255**	**1909**	**208**
				22	**234**	**1888**	**215**	62	**256**	**1860**	**230**
				23	**234**	**1901**	**213**	63	**256**	**1889**	**210**
				24	**234**	**1903**	**211**	64	**257**	**1859**	**230**
				25	**236**	**1883**	**218**	65	**260**	**1857**	**230**
				26	**236**	**1990**	**213**	66	**260**	**1860**	**228**
				27	**236**	**1901**	**212**	67	**262**	**1856**	**236**
				28	**237**	**1887**	**215**	68	**262**	**1858**	**228**
				29	**238**	**1886**	**215**	69	263	1899	209
				30	**238**	**1895**	**210**	70	264	1907	208
				31	**239**	**1884**	**216**	71	265	1906	208
				32	**239**	**1885**	**215**	72	**266**	**1855**	**240**
				33	**240**	**1874**	**220**	73	**267**	**1854**	**240**
				34	**240**	**1880**	**216**	74	**267**	**1855**	**230**
				35	**240**	**1883**	**215**	75	268	1895	209
				36	**241**	**1871**	**220**	76	270	1911	207
				37	**241**	**1879**	**219**	77	276	1902	208
				38	**241**	**1888**	**214**	78	**277**	**1853**	**236**
				39	**241**	**1891**	**212**	79	278	1913	205
				40	**242**	**1877**	**216**	80	301	1851	250

All in all, from the above comparisons based on both Kacem instances and BRdata instances, it can be concluded that the P-EDA is more powerful than the existing weighted approaches and Pareto-based approaches in solving the MFJSP. The superiority of the P-EDA owes to the following aspects. (1) With the well-designed probability model and the suitable updating mechanism, it is helpful to explore the searching and obtain more good solutions effectively, especially within the promising area of the solution space. (2) With the critical path-based local search, it is helpful to enhance the exploitation and obtain a schedule with a small makespan. (3) With the investigation based on the design of experiment to determine a suitable parameter setting, it is helpful to obtain an optimal performance of the algorithm. With the above merits, the P-EDA is more effective than the existing algorithms in solving the MFJSP.

5. Conclusions

In this paper, an effective Pareto-based estimation of distribution algorithm was proposed to solve the multi-objective flexible job-shop scheduling problem with the criteria to minimise the makespan, the total workload of machines and the workload of the critical machine. We employed the Pareto-optimality-based fitness evaluation and designed a probability model with the Pareto superior population. We also proposed a mechanism to update the probability model for generating new individuals. Multiple strategies in a combined way were used to generate the initial solutions. To avoid premature convergence and enhance local exploitation, a splitting mechanism and two operators were designed to adjust the individuals of sub-populations in a separated way. In addition, 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 showed that the proposed P-EDA is more effective in solving the MFJSP than both the existing weighted approaches and Pareto-based approaches. Future work is to design EDA-based algorithms for the lot sizing scheduling problems.

Acknowledgments

This research was financially supported by the National Key Basic Research and Development Program of China (grant number 2013CB329503), the National Science Foundation of China (grant numbers 61174189, 61025018 and 60834004), the Doctoral Program Foundation of Institutions of Higher Education of China (grant number 20100002110014) and the National Science and Technology Major Project of China (grant number 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. 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, 1997, 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., 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 Recognit* 38: 2099–2113.

Chen, S.H., M.C. Chen, P.C. Chang, Q.F. Zhang, and Y.M. Chen. 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., 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.

Deb, K., S. Agrawal, A. Pratab, and T. Meyarivan. 2000. "A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II." *Lecture Notes in Computer Science* 1917: 849–858.

De Bonet, J.S., C.L. Isbell, Jr. and P. Viola. MIMIC: Finding optima by estimating probability densities. *Advances in Neural Information Processing Systems*, Cambridge: MIT Press, 1997, 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." *Computer & Operations Research* 35: 2892–2907.

Gomes, M.C., A.P. Barbosa-Póvoab, and A.Q. Novaisc. 2005. "Optimal scheduling for flexible job shop operation." *International Journal of Production Research* 43: 2323–2353.

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

Harik, G.R., F.G. Lobo, and D.E. Goldberg. The compact genetic algorithm. In: Proc of the IEEE Conf on Evolutionary Computation. Indianapolis, 1998, 523–528.

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

Kacem, I., S. Hammadi, and P. Borne. 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., and J.A. Lozano. 2002. *Estimation of distribution algorithms: A new tool for evolutionary computation*. Boston: Kluwer Press.

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

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." *International Journal of Advanced Manufacturing Technology* 55: 1159–1169.

Li, J.Q., Q.K. Pan, and J. Chen. 2012. "A hybrid Pareto-based local search algorithm for multi-objective flexible job shop scheduling problems." *International Journal of Production Research* 50: 1063–1078.

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.

Mastrolilli, M., and L.M. Gambardella. 2000. "Effective neighborhood functions for the flexible job shop problem." *J of Scheduling* 3: 3–20.

Mühlenbein, 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. Goldberg, D.E. and Cantú-Paz, E. 1999. BOA: The bayesian optimization algorithm. *Proc of the Genetic and Evolutionary Computation*. Orlando, 525–532.

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.

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

Rabieea, M., M. Zandiehb, and P. Ramezania. Bi-objective partial flexible job shop scheduling problem: NSGA-II, NRGA, MOGA and PAES approaches. *International Journal of Production Research*, 2012, DOI: 10.1080/ 00207543.2011.648280.

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.

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." *Computer & Operations Research* 39: 449–460.

Wang, L., S.Y. Wang, and C. Fang. 2012a. "A hybrid estimation of distribution algorithm for solving multidimensional knapsack problem." *Expert Systems with Applications* 39: 5593–5599.

Wang, L., S.Y. Wang, Y. Xu, G. Zhou, and M. Liu. 2012b. "A bi-population based estimation of distribution algorithm for the flexible job-shop scheduling problem." *Computers & Industrial Engineering* 62: 917–926.

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." *International Journal of Advanced Manufacturing Technology* 51: 757–767.

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.

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

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.