Int. J. Production Economics 145 (2013) 387–396

Contents lists available at SciVerse ScienceDirect

Int. J. Production Economics

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

An effective estimation of distribution algorithm for solving the distributed permutation flow-shop scheduling problem

Sheng-yao Wang*, Ling Wang, Min Liu, Ye Xu

Department of Automation, Tsinghua National Laboratory for Information Science and Technology (TNList), Tsinghua University, Beijing 10084, China

ARTICLE INFO

Article history:
Received 13 June 2012
Accepted 3 May 2013
Available online 14 May 2013

Keywords:
Distributed permutation flow-shop scheduling
Estimation of distribution algorithm
Probability model
Design of experiment

ABSTRACT

In this paper, an effective estimation of distribution algorithm (EDA) is proposed to solve the distributed permutation flow-shop scheduling problem (DPFSP). First, the earliest completion factory rule is employed for the permutation based encoding to generate feasible schedules and calculate the schedule objective value. Then, a probability model is built for describing the probability distribution of the solution space, and a mechanism is provided to update the probability model with superior individuals. By sampling the probability model, new individuals can be generated among the promising search region. Moreover, to enhance the local exploitation, some local search operators are designed based on the problem characteristics and utilized for the promising individuals. In addition, the influence of parameter setting of the EDA is investigated based on the Taguchi method of design of experiments, and a suitable parameter setting is suggested. Finally, numerical simulations based on 420 small-sized instances and 720 large-sized instances are carried out. The comparative results with some existing algorithms demonstrate the effectiveness of the proposed EDA in solving the DPFSP. In addition, the new best-known solutions for 17 out of 420 small instances and 589 out of 720 large instances are found.

1. Introduction

The permutation flow-shop scheduling problem (PFSP) has been concentrated on by many researchers due to its wide applications in economics and industrial engineering (Hejazi and Saghafian, 2005). The PFSP has been proved to be NP-complete when the number of machines is more than three (Garey et al., 1976). After the pioneering work of Johnson (1954), much research work has been carried out on the PFSP (Cheng and Janiak, 2000; Suliman, 2000; Chung et al., 2002; Cheng and Kovalyov, 2003; Cheng et al., 2004, 2013; Ruiz and Maroto, 2005; Lin et al., 2008; Tseng and Lin, 2010a, 2010b; Sun et al., 2012; Shabtay et al., 2013; Wang et al., 2013c, 2013d). In most research of the PFSP, a common assumption is that there is only one production center or factory, which means that all jobs are assumed to be processed in the same factory. Nevertheless, with the development of the business concept, coproduction between companies is more and more common nowadays (Wang and Shen, 2007). Besides, multi-plant companies and supply chains are taking a more important role in practice (Moon et al., 2002). Therefore, the distributed manufacturing strategy comes into being, which enables companies to achieve higher product quality, lower production costs and lower management risks (Kahn et al., 2004).

Scheduling in distributed systems is more difficult than the classical shop scheduling, because it should determine the assignment of jobs to factories as well as the processing sequence in each factory. Obviously, both sub-problems are related to each other and cannot be solved sequentially if high performance is desired (Naderi and Ruiz, 2010). Compared to the classical shop scheduling, the literature on the distributed scheduling is relatively limited and the study on this topic is in its infancy. Jia et al. (2002, 2003) studied the distributed job shop problem under different criteria and employed a standard genetic algorithm (GA) to solve the problem. Later, Jia et al. (2007) refined the previous GA to solve the small-sized and medium-sized distributed scheduling problems. Chan et al. (2005, 2006) proposed an adaptive GA to solve the distributed job shops with makespan criterion for larger problems. De Giovanni and Pezzella (2010) proposed an improved GA to solve the distributed and flexible job-shop scheduling problem. As for the distributed permutation flow-shop scheduling problem (DPFSP), Naderi and Ruiz (2010) presented six mixed integer linear programming models and developed two factory assignment rules and 14 heuristics based on dispatching rules, effective constructive heuristics and variable neighborhood descent methods. Besides, to evaluate the proposed models and algorithms, the authors generated 420 small-sized instances and 720 large-sized instances which are available at http://soa.iti.es,

* Correspondence to: Room 427A, Main Building, Tsinghua University, Beijing 100084, China. Tel.: +86 10 62783125; fax: +86 10 62786911.
E-mail address: wangshengyao@tsinghua.org.cn (S.-y. Wang).

0925-5273/$ - see front matter
http://dx.doi.org/10.1016/j.ijpe.2013.05.004

along with the best known solution for each instance. Based on these instances, Gao et al. (2012) proposed a tabu search algorithm for solving the DPFSP and tested the performance of the proposed algorithm. However, the authors centred their study only on a part of the instances and no results were listed for direct comparisons.

Estimation of distribution algorithm (EDA) is a relatively novel population-based optimization algorithm, which has led to increasing studies and wide applications during recent years (Larranaga and Lozano, 2002). Considering different kinds of the relationships between variables, the EDA has different complexity of the model. Accordingly, the EDA can be classified as a univariate model, bivariate model or multivariate model. Univariate models assume that the variables are independent of each other, e.g., the population-based incremental learning (Baluja, 1994), the univariate marginal distribution algorithm (Mühlenbein and Paass, 1996) and the compact GA (Harik et al., 1998). Bivariate models assume that each variable is associated with another one, e.g., the mutual information maximization for input clustering (De Bonet et al., 1997), the combining optimizers with mutual information trees (Baluja and Davies, 1997) and the bivariate marginal distribution algorithm (Pelikan and Mühlenbein, 1999). Multivariate models consider the relationship between all the variables, e.g., the factorized distribution algorithms (Mühlenbein and Mahnig, 1999), the extended compact GA (Harik, 1999) and the Bayesian optimization algorithm (Pelikan et al., 1999). 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), single machine scheduling (Chen and Chen, 2013) flow-shop scheduling (Jarboui et al., 2009), resource-constrained project scheduling (Wang and Fang, 2012), multi-dimensional knapsack problem (Wang et al., 2012a), flexible job-shop scheduling (Wang et al., 2012b, 2013a, 2013b), and so on. However, to the best of our knowledge, there is no research work about the EDA for solving DPFSP. In this paper, we will propose an effective EDA to solve the DPFSP with the criterion to minimize the maximum completion time. Specifically, the earliest completion factory rule is employed for the permutation based encoding to generate feasible schedules and calculate the schedule objective value. Meanwhile, a probability model is built with the superior individuals for generating new individuals, and a mechanism is provided to update the probability model. Besides, some local search operators are designed based on the problem characteristics and utilized to enhance the exploitation capability. In addition, the influence of parameters is investigated based on Taguchi method of design of experiment, and a suitable parameter setting is suggested. Finally, we use the benchmark instances generated by Naderi and Ruiz (2010) to test the performances of the EDA and to compare it with some existing methods to solve the DPFSP.

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

2. Distributed permutation flow-shop scheduling problem

The distributed permutation flow-shop scheduling problem (Naderi and Ruiz, 2010) can be described as follows. There are n jobs $J=\{J_1, J_2, \ldots, J_n\}$ to be processed in F factories, where each factory contains the same set of m machines $M=\{M_1, M_2, \ldots, M_m\}$. A job J_i is formed by a sequence of m operations $\{O_{i,1}, O_{i,2}, \ldots, O_{i,m}\}$ to be performed one after another, where the execution of $O_{i,j}$ requires machine M_j and processing time $t_{i,j}>0$. When a job is assigned to a certain factory, it cannot be transferred to another factory and all its operations can only be processed in the factory. Besides, the following assumptions for the classical flow-shop scheduling are adopted. All jobs are independent and available for processing at time 0. Each machine can process only one job at a time and each job can be processed on only one machine at a time. Preemption is not allowed, i.e., each operation must be completed without interruption once it is started. Setup times of machines and move times between operations are negligible. The DPFSP is to determine both the assignment of jobs to the factories and the sequences of jobs in all the factories to minimize a certain scheduling objective function. In this paper, we consider the maximum completion time (makespan) as the criterion.

Let $\lambda^k=[\lambda^k(1), \lambda^k(2), \cdots, \lambda^k(n_k)]$ be the sequence of the jobs in factory k, where n_k is the total number of the jobs assigned to factory k. $C_{i,j}$ is denoted as the completion time of $O_{i,j}$. For a schedule Λ of the DPFSP, i.e., a set of sequences $\{\lambda^1, \lambda^2, \cdots, \lambda^F\}$, we can calculate the makespan $C_{\max}$ as follows:

$$C_{\lambda^k(1),1}=t_{\lambda^k(1),1}, \quad k=1,2,\cdots,F \tag{1}$$

$$C_{\lambda^k(i),1}=C_{\lambda^k(i-1),1}+t_{\lambda^k(i),1}, \quad k=1,2,\cdots,F; \quad i=2,3,\cdots,n_k \tag{2}$$

$$C_{\lambda^k(1),j}=C_{\lambda^k(1),j-1}+t_{\lambda^k(1),j}, \quad k=1,2,\cdots,F; \quad j=2,3,\cdots,m \tag{3}$$

$$C_{\lambda^k(i),j}=\max\{C_{\lambda^k(i-1),j}, C_{\lambda^k(i),j-1}\}+t_{\lambda^k(i),j},$$
$$k=1,2,\cdots,F; \quad i=2,3,\cdots,n_k; \quad j=2,3,\cdots,m \tag{4}$$

$$C_{\max}=\max C_{n_k,m}, \quad k=1,2,\cdots,F \tag{5}$$

The objective of solving the DPFSP is to find a schedule with the minimum makespan.

3. Estimation of distribution algorithm

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

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

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

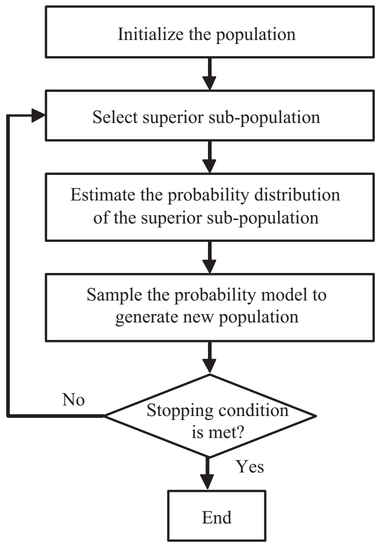

Fig. 1. General framework of the EDA.

Procedure ECF rule.
For $k=1$ to F
  $\lambda^k(1)=\pi(k)$;
  $n_k=1$;
For $k=F+1$ to n
  Find the factory f that can process job $\pi(k)$ with the earliest completion time;
  $n_f=n_f+1$;
  $\lambda^f(n_f)=\pi(k)$;

Fig. 2. Pseudo code of the ECF rule.

4. EDA for DPFSP

In this section, an estimation of distribution algorithm (EDA) is presented to solve the DPFSP. The framework of the EDA is illustrated, following the encoding and decoding schemes, probability model and its updating mechanism, and local search scheme.

4.1. Encoding and decoding schemes

Every individual of the population denotes a solution of the DPFSP, which is represented by a sequence of all the job numbers as Eq. (6) to determine the schedule order of all the jobs. For example, a solution $\pi=[2,3,1,4]$ implies that job 2 is scheduled first, and next are job 3 and job 1, in sequence. Job 4 is the last job to be scheduled.

$$\pi=[\pi(1),\pi(2),\cdots,\pi(n)] \tag{6}$$

To decode a sequence is to arrange the factories for all the jobs and determine the processing order in each factory so as to generate a feasible schedule. Considering the characteristics of the DPFSP, we employ an effective decoding rule called earliest completion factory (ECF) rule. For each job in the sequence to be scheduled, the ECF rule assigns it to the factory that can complete the job with the earliest completion time. The factory assignment is the same method as that implemented by Naderi and Ruiz (2010). Furthermore, the jobs in the same factory are processed in the order as they appear in the sequence π. The pseudo code of the ECF rule is illustrated in Fig. 2.

Obviously, the ECF rule aims at obtaining a schedule with small makespan and balancing the workload of factories. With the ECF rule, an individual of the EDA can be decoded to a feasible schedule Λ. Then the makespan of the schedule can be calculated as mentioned in Section 2.

4.2. Probability model and updating mechanism

Different from the GA that produces offspring through crossover and mutation operators, the EDA does it by sampling according to a probability model. So, the probability model has a great effect on the performance of the EDA. In this paper, the probability model is designed as a probability matrix P.

The element $p_{ij}(l)$ of the probability matrix P represents the probability that job j appears before or in position i of the solution sequence at generation l. The value of p_{ij} refers to the importance of a job when deciding the scheduling order. For all values of i and j, p_{ij} is initialized to $p_{ij}(0)=1/n$, which ensures that the whole solution space can be sampled uniformly.

In each generation of the EDA, the new individuals are generated via sampling the solution space according to the probability matrix P. For every position i, job j is selected with a probability p_{ij}. If job j has already appeared, it means that job j has been scheduled. Then, the whole jth column of probability matrix P will be set as zero and all the elements of P will be normalized to maintain that each row sums up to 1. In such a way, an individual is constructed until all the jobs appear in the sequence, and then its makespan can be calculated. In the EDA, a population with *P_Size* individuals are generated.

Next, it determines the superior sub-population that consists of the best *SP_Size* solutions, where $SP\_Size=\eta\%\times P\_Size$. And then the probability matrix P is updated according to the following equation:

$$p_{ij}(l+1)=(1-\alpha)p_{ij}(l)+\frac{\alpha}{i\times SP\_Size}\sum_{k=1}^{SP\_Size}I_{ij}^k,\forall i,j \tag{7}$$

where $\alpha\in(0,1)$ is the learning rate of P, and I_{ij}^k is the following indicator function of the kth individual in the superior sub-population.

$$I_{ij}^k=\begin{cases}1, & \text{if job } j \text{ appears before or in position } i\\ 0, & \text{else}\end{cases} \tag{8}$$

The updating process can be regarded as a kind of increased learning, where the second term on the right hand side of the equation represents learning information from the superior sub-population. Note that $\sum_{k=1}^{SP\_Size}I_{ij}^k$ indicates that the total appearance number of all the jobs before or in position i is $i\times SP\_Size$.

4.3. Local search scheme

It is widely accepted that a local search procedure is efficient in improving the solutions generated by the EDA (Wang et al., 2012a). In this paper, some local search operators are designed based on the problem characteristics to enhance the local exploitation around the best solution found by the EDA. Since the makespan of a solution can be reduced by improving the schedule in the factory with the latest completion time, we design the local search operators as follows.

Job-swap: In the factory with the latest completion time, randomly select two different jobs from the processing sequence and then swap them.

Job-insert: In the factory with the latest completion time, randomly choose two different jobs from the sequence and then insert the back one before the front one.

Job-inverse: In the factory with the latest completion time, invert the subsequence between two different random positions of a job sequence.

Factory-swap: Randomly select two different jobs, one from the factory with the latest completion time and the other from another randomly selected factory; then swap the factories assigned to them.

In each step of the local search, these operators are performed sequentially in the above order (one time for one operator) to generate another solution, and then the new solution replaces the old one if it has a smaller makespan. The above procedure is applied 200 times on the best individual of the current population in every generation.

4.4. Procedure of the EDA

With the above design, the procedure of the EDA for solving the DPFSP is illustrated in Fig. 3.

It can be seen that the EDA contains two main phases in every generation. At the global exploration phase, a probability model is built with the superior individuals of the entire population to generate the new individuals. At the local exploitation phase, the best solution adopts multiple local search operators based on the problem characteristics for further exploitation. The algorithm stops when the maximum number of generations *Gen* is reached.

5. Computational results and comparisons

To test the performance of the EDA, numerical tests are carried out with two sets of benchmarks (Naderi and Ruiz, 2010), which are available at http://soa.iti.es. The first set consists of 420 small-sized instances, where $n=\{4, 6, 8, 10, 12, 14, 16\}$, $m=\{2, 3, 4, 5\}$ and $F=\{2, 3, 4\}$. The second set consists of 720 large-sized instances, which is extended from the benchmark of Taillard (1993). The combinations of $n \times m$ are $\{20, 50, 100\} \times 5$, $\{20, 50, 100, 200\} \times 10$, and $\{20, 50, 100, 200, 500\} \times 20$ and the number of factories F is from $\{2, 3, 4, 5, 6, 7\}$.

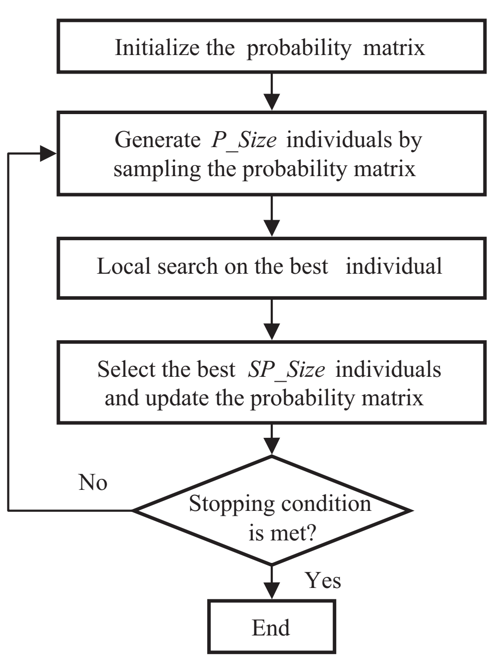

Fig. 3. Framework of the EDA for the DPFSP.

The EDA is coded in C language and run on a 3.2 GHz Intel Core i5 processor. To evaluate the performance of the EDA, same as in literature (Naderi and Ruiz, 2010), we evaluate the experimental results by relative percentage deviation (RPD) as follows:

$$RPD = \frac{alg-opt}{opt} \times 100 \tag{9}$$

where *opt* is the makespan of best-known solutions from http://soa.iti.es and *alg* corresponds to the makespan of the solution obtained by a certain algorithm. If the obtained RPD is less than 0, it implies that a new best solution is found.

5.1. Parameters setting

The proposed EDA contains several key parameters: *P_Size* (the population size), α (the learning rate of P), η (the parameter associated with the superior sub-population), and *Gen* (the maximum number of generations). To investigate the influence of these parameters on the performance of the EDA, we implement the Taguchi method of design of experiment (DOE) (Montgomery, 2005) by using a moderate-sized instance (Naderi and Ruiz, 2010), i.e., I_4_16_5_4, where 4_16_5 denotes the size ($F=4$, $n=16$ and $m=5$) of the instance and the last 4 denotes that it is the fourth instance of this size. Combinations of different values of these parameters are listed in Table 1.

For each parameter combination, the EDA is run 10 times independently and the average makespan value obtained by the EDA is calculated as the average response variable (ARV) value. 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 4,

Table 1
Combinations of parameter values.

Parameters	Factor level			
	1	2	3	4
P_Size	50	100	150	200
η	10	20	30	40
α	0.1	0.2	0.3	0.4
Gen	500	1000	1500	2000

Table 2
Orthogonal array and ARV values.

Experiment Number	Factor				ARV
	P_Size	η	α	*Gen*	
1	1	1	1	1	443.4
2	1	2	2	2	445.1
3	1	3	3	3	447.5
4	1	4	4	4	446.3
5	2	1	2	3	443.1
6	2	2	1	4	444.5
7	2	3	4	1	446.6
8	2	4	3	2	446.2
9	3	1	3	4	444.5
10	3	2	4	3	445.6
11	3	3	1	2	443.2
12	3	4	2	1	444.8
13	4	1	4	2	444.7
14	4	2	3	1	445.8
15	4	3	2	4	443.9
16	4	4	1	3	443.6

and the number of factor levels is 4. The orthogonal array and the obtained ARV values are listed in Table 2.

According to the orthogonal table, we illustrate the trend of each factor level in Fig. 4. Then, we figure out the response value of each parameter to analyze its significance rank. The results are listed in Table 3.

From Table 3 it can be seen that the learning rate α of P is the most significant one among the four parameters. That is, the learning rate of the matrix for machine assignment is crucial to the EDA. A large value of α could lead to premature convergence. In addition, η ranks the second, which implies that the number of the superior sub-population to update the probability model is also important. A small value of η can help the algorithm build an accurate model. Besides, the significant rank of the population size is the third. A large value of *P_Size* makes the algorithm sample the solution space sufficiently. However, a large population size will cause a large amount of computational budget. It can be seen from Fig. 4 that it makes no improvement when the size is too large. Similar conclusion can be drawn for the maximum number of the generations. According to the above analysis, a good choice of parameter combination is suggested as $P\_Size=150$, $\eta=10$, $\alpha=0.1$ and $Gen=1000$.

5.2. Results and comparison for small-sized instances

Considering the 420 small-sized instances, we compare the EDA with several heuristic algorithms (Naderi and Ruiz, 2010). For each instance, we run the EDA 10 times independently and obtain the best makespan and the RPD. Table 4 summarizes the results grouped by each combination of n and F (20 data per average) as Naderi and Ruiz (2010), where the results of the comparative algorithms are directly from literature.

From Table 4, it can be seen that the EDA is the best one among all the algorithms for solving the small-sized instances. The corresponding RPD values of the best solutions by the EDA are negative for the instances $\{2, 3, 4\} \times 16$ and 4×14, which implies that some of the best know solutions are updated by the EDA. In particular, our EDA obtains new best makespan values for 17 instances, which are listed in Table A1 in Appendix A. In addition, Fig. 5 illustrates the Gantt chart of the best solution obtained by the EDA for instance I_4_16_4_1. As for the other 403 small-sized instances, the best makespan values by the EDA are equal to the best known ones.

For the small-sized instances, the CPU times employed by both the EDA and the heuristic algorithms are extremely short. Similar to Naderi and Ruiz (2010), we will comment on the CPU times based on the large-sized instances in the next sub-section.

5.3. Results and comparison for large-sized instances

Next, we carry out tests with the large-sized instances. Table 5 presents the results of the experiments, averaged for each value of F (120 data per average).

Form Table 5, it can be seen that the EDA outperforms other algorithms in solving all the large-sized instances. On average, the EDA yields −1.63% RPD to the best known values. In particular, the EDA obtains new best makespan values for 589 out of 720 large-sized instances, which are listed in Tables B.1–B.6 in Appendix B (grouped by each value of F).

Besides, the CPU times employed by the EDA, VND(a), VND(b), NEH1 and NEH2 for the instances grouped by F are listed in Table 6.

From Table 6, it can be seen that all the heuristic algorithms spend the average CPU time below 0.15 s, while the EDA spends much more. The reason is that, the heuristic algorithm constructs a solution based on some heuristic rules while the EDA performs search procedure among the whole solution space. Fortunately, the average running time of the EDA is acceptable and does not

Table 3
Response value and rank of each parameter.

Level	*P_Size*	η	α	*Gen*
1	445.575	443.925	443.675	445.15
2	445.1	445.25	444.225	444.8
3	444.525	445.3	446	444.95
4	444.5	445.225	445.8	444.8
Delta	1.075	1.375	2.325	0.35
Rank	3	2	1	4

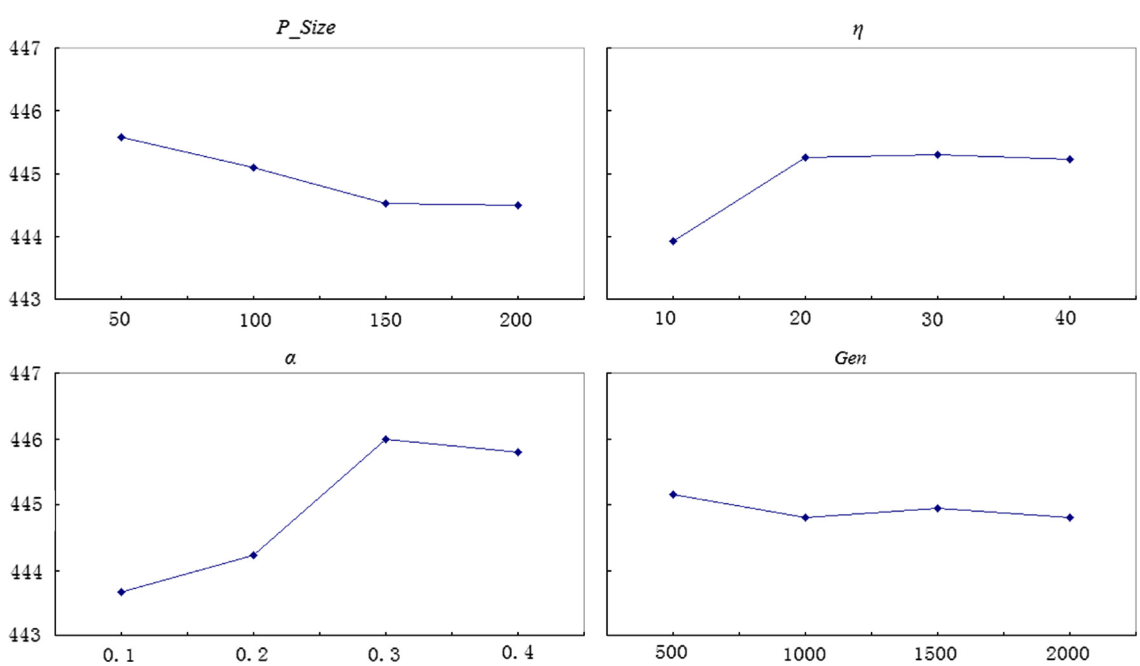

Fig. 4. Factor level trend of the EDA.

Table 4
RPD of the algorithms for the small-sized instances.

$F\times n$	SPT1	LPT1	Johnson1	CDS1	Palmer1	NEH1	VND(a)	SPT2	LPT2	Johnson2	CDS2	Palmer2	NEH2	VND(b)	EDA
2×4	12.95	19.31	6.13	2.69	7.75	2.61	**0.00**	10.02	17.80	3.76	0.72	5.22	0.15	0.15	**0.00**
2×6	13.30	29.53	10.34	7.56	7.57	4.42	1.26	10.38	28.37	8.08	4.60	4.83	1.44	1.44	**0.00**
2×8	17.31	35.43	9.75	12.33	9.97	4.43	2.10	16.16	33.09	7.98	10.33	8.64	2.53	2.52	**0.00**
2×10	21.03	36.93	8.38	9.72	10.18	4.28	3.22	17.49	37.12	7.52	8.38	9.25	3.27	2.89	**0.00**
2×12	20.40	39.36	10.14	11.81	10.77	6.95	3.38	17.23	37.42	8.29	10.49	10.32	4.13	3.90	**0.00**
2×14	17.73	40.00	10.27	6.59	10.57	6.05	2.70	17.34	38.36	9.21	5.24	9.24	3.16	2.82	**0.00**
2×16	17.11	42.42	12.33	10.53	10.97	6.66	2.92	16.95	41.63	10.99	8.26	9.00	3.84	3.30	**−0.04**
3×4	8.87	6.42	5.31	5.01	4.82	0.43	**0.00**	5.36	5.41	1.99	2.55	1.50	0.43	**0.00**	**0.00**
3×6	21.24	27.63	6.91	8.50	9.40	4.04	0.68	11.93	25.26	5.28	6.99	3.40	1.39	1.39	**0.00**
3×8	17.71	26.87	10.14	9.17	11.84	5.08	1.86	17.93	25.14	8.30	8.67	6.98	2.43	2.43	**0.00**
3×10	24.48	37.98	12.67	13.56	13.97	8.42	2.59	16.23	35.43	10.07	10.81	11.67	3.79	3.63	**0.00**
3×12	25.92	42.12	14.90	14.66	14.39	7.66	3.66	19.55	40.14	10.12	11.00	11.36	5.08	5.08	**0.00**
3×14	23.44	41.00	14.62	16.45	17.97	10.54	4.48	19.95	38.56	14.04	13.47	14.05	4.90	4.46	**0.00**
3×16	25.31	41.71	14.39	15.55	15.68	8.18	3.50	22.83	39.39	10.41	11.60	11.06	3.98	3.81	**−0.18**
4×4	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
4×6	13.34	14.52	7.65	4.99	6.02	3.10	**0.00**	11.31	12.52	5.62	3.39	2.98	0.47	**0.00**	**0.00**
4×8	15.21	16.86	9.20	9.23	10.17	4.19	0.77	13.89	16.44	5.56	8.61	7.04	0.77	0.77	**0.00**
4×10	22.65	34.47	13.19	13.90	16.21	7.07	1.57	16.84	30.64	8.93	9.30	7.23	2.30	2.22	**0.00**
4×12	25.51	38.72	13.61	18.13	15.49	8.58	4.23	20.54	34.06	8.68	14.49	12.39	4.97	4.68	**0.00**
4×14	26.61	42.89	13.50	15.47	16.53	10.94	4.25	22.31	39.79	8.51	11.23	12.25	4.54	4.46	**−0.02**
4×16	28.70	44.01	19.12	17.62	19.34	9.79	5.08	24.09	40.67	13.96	13.69	15.84	5.59	5.55	**−0.19**
Average	18.99	34.34	10.60	10.64	11.41	5.88	2.30	15.63	29.39	7.97	8.28	8.30	2.82	2.64	**−0.02**

Note: The bold values mean better results.

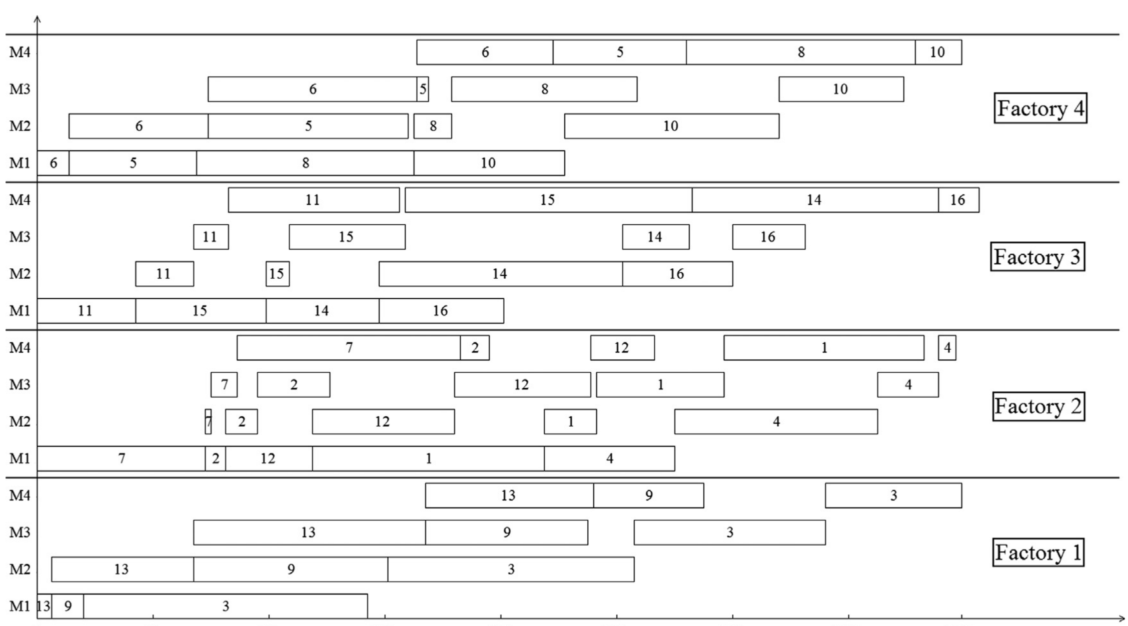

Fig. 5. Best solution of instance I_4_16_4_1 found by the EDA.

Table 5
RPD of the algorithms for the large-sized instances.

Instance (F)	SPT1	LPT1	Johnson1	CDS1	Palmer1	NEH1	VND(a)	SPT2	LPT2	Johnson2	CDS2	Palmer2	NEH2	VND(b)	EDA
2	18.71	33.70	12.68	9.29	10.44	2.92	0.10	17.71	32.36	11.58	8.52	9.34	1.21	0.32	**−1.55**
3	19.95	33.05	13.40	10.83	11.72	3.42	0.10	17.58	31.57	11.18	8.92	9.43	1.15	0.35	**−1.79**
4	20.07	33.30	13.48	11.19	11.95	4.21	0.06	17.23	30.42	10.67	8.54	8.90	1.11	0.46	**−1.76**
5	20.04	32.89	13.18	11.29	12.24	4.28	0.11	16.73	29.57	10.24	8.07	8.76	0.92	0.46	**−1.77**
6	20.32	32.58	13.57	11.50	12.70	4.73	0.11	16.07	28.55	9.94	7.83	8.45	0.95	0.51	**−1.52**
7	21.04	32.02	13.61	11.49	12.53	4.86	0.10	15.42	27.14	9.71	7.31	8.21	0.81	0.45	**−1.37**
Average	20.02	32.92	13.35	10.93	11.93	4.07	0.10	16.79	29.93	10.55	8.20	8.85	1.03	0.43	**−1.63**

Note: The bold values mean better results.

Table 6
CPU time spent by the algorithms (s).

Instance (F)	EDA	VND(a)	VND(b)	NEH1	NEH2
2	90.390	0.246	0.120	0.009	0.017
3	101.015	0.182	0.105	0.007	0.020
4	111.375	0.129	0.104	0.007	0.024
5	120.625	0.114	0.086	0.006	0.028
6	130.609	0.120	0.083	0.005	0.031
7	140.734	0.091	0.076	0.006	0.035
Average	115.791	0.147	0.096	0.007	0.026

Table A1
New best makespan obtained by the EDA for small-sized instances.

Instance	F	n	m	Best known	EDA	RPD
I_2_16_5_1	2	16	5	526	523	−0.57
I_2_16_5_3	2	16	5	652	650	−0.31
I_3_16_3_1	3	16	3	340	339	−0.29
I_3_16_3_3	3	16	3	350	349	−0.29
I_3_16_3_5	3	16	3	362	361	−0.28
I_3_16_4_2	3	16	4	458	457	−0.22
I_3_16_4_4	3	16	4	430	429	−0.23
I_3_16_4_5	3	16	4	419	414	−1.19
I_3_16_5_1	3	16	5	453	452	−0.22
I_3_16_5_3	3	16	5	476	475	−0.21
I_3_16_5_4	3	16	5	524	522	−0.38
I_4_14_5_4	4	14	5	425	423	−0.47
I_4_16_3_3	4	16	3	312	311	−0.32
I_4_16_4_1	4	16	4	323	319	−1.24
I_4_16_4_2	4	16	4	359	358	−0.28
I_4_16_5_3	4	16	5	365	363	−0.55
I_4_16_5_4	4	16	5	447	441	−1.34

Table B1
New best makespan obtained by the EDA for large-sized instances (F=2).

Instance	Best known	EDA	Instance	Best known	EDA	Instance	Best known	EDA
Ta001_2	770	751	Ta035_2	1508	1488	Ta069_2	2823	2797
Ta002_2	783	768	Ta036_2	1480	1477	Ta070_2	2730	2728
Ta003_2	676	645	Ta037_2	1482	1439	Ta071_2	3182	3133
Ta004_2	803	765	Ta038_2	1428	1412	Ta072_2	2969	2903
Ta005_2	751	731	Ta039_2	1389	1339	Ta073_2	3093	3035
Ta006_2	722	709	Ta040_2	1465	1444	Ta074_2	3235	3199
Ta007_2	731	708	Ta041_2	1823	1776	Ta075_2	3060	3006
Ta008_2	745	711	Ta042_2	1743	1722	Ta076_2	2902	2871
Ta009_2	742	720	Ta043_2	1731	1714	Ta077_2	3050	3025
Ta010_2	672	645	Ta044_2	1828	1776	Ta078_2	3124	3081
Ta011_2	1071	1050	Ta045_2	1811	1779	Ta079_2	3247	3205
Ta012_2	1145	1117	Ta046_2	1845	1767	Ta080_2	3235	3137
Ta013_2	1051	1001	Ta047_2	1902	1821	Ta081_2	3892	3833
Ta014_2	934	913	Ta048_2	1822	1787	Ta082_2	3842	3792
Ta015_2	1013	966	Ta049_2	1739	1710	Ta083_2	3866	3836
Ta016_2	959	928	Ta050_2	1854	1824	Ta084_2	3844	3800
Ta017_2	1029	991	Ta051_2	2585	2560	Ta086_2	3972	3899
Ta018_2	1083	1032	Ta052_2	2511	2467	Ta087_2	3887	3875
Ta019_2	1063	1028	Ta053_2	2513	2423	Ta088_2	4040	3967
Ta020_2	1117	1073	Ta054_2	2539	2485	Ta089_2	3929	3867
Ta021_2	1723	1686	Ta055_2	2470	2437	Ta090_2	3950	3899
Ta022_2	1589	1571	Ta056_2	2508	2453	Ta091_2	5721	5700
Ta023_2	1773	1736	Ta057_2	2496	2445	Ta092_2	5587	5553
Ta024_2	1676	1642	Ta058_2	2538	2479	Ta094_2	5711	5669
Ta025_2	1781	1697	Ta059_2	2551	2515	Ta097_2	5720	5708
Ta026_2	1691	1658	Ta060_2	2537	2494	Ta098_2	5659	5646
Ta027_2	1733	1684	Ta061_2	2846	2810	Ta099_2	5495	5470
Ta028_2	1661	1621	Ta062_2	2725	2705	Ta100_2	5663	5612
Ta029_2	1694	1668	Ta063_2	2665	2658	Ta102_2	6498	6490
Ta030_2	1645	1609	Ta064_2	2582	2571	Ta103_2	6554	6489
Ta031_2	1436	1407	Ta065_2	2685	2681	Ta104_2	6450	6445
Ta032_2	1508	1490	Ta066_2	2656	2626	Ta105_2	6404	6402
Ta033_2	1393	1368	Ta067_2	2719	2678	Ta106_2	6479	6411
Ta034_2	1484	1449	Ta068_2	2637	2614	Ta109_2	6503	6466

Table B2
New best makespan obtained by the EDA for large-sized instances (F=3).

Instance	Best known	EDA	Instance	Best known	EDA	Instance	Best known	EDA
Ta001_3	598	576	Ta036_3	1064	1045	Ta072_3	2157	2109
Ta002_3	594	582	Ta037_3	1072	1030	Ta073_3	2223	2190
Ta003_3	533	505	Ta038_3	1016	1003	Ta074_3	2327	2312
Ta004_3	630	602	Ta039_3	990	955	Ta075_3	2216	2178
Ta005_3	579	565	Ta040_3	1041	1020	Ta076_3	2111	2078
Ta006_3	574	554	Ta041_3	1417	1365	Ta077_3	2186	2170
Ta007_3	566	546	Ta042_3	1347	1320	Ta078_3	2283	2230
Ta008_3	590	560	Ta043_3	1335	1316	Ta079_3	2345	2302
Ta009_3	594	556	Ta044_3	1411	1368	Ta080_3	2315	2288
Ta010_3	531	501	Ta045_3	1406	1372	Ta081_3	2985	2943
Ta011_3	905	875	Ta046_3	1390	1355	Ta082_3	2938	2912
Ta012_3	961	927	Ta047_3	1440	1400	Ta083_3	2996	2948
Ta013_3	872	843	Ta048_3	1408	1373	Ta084_3	2954	2928
Ta014_3	796	760	Ta049_3	1350	1310	Ta085_3	3000	2969
Ta015_3	835	806	Ta050_3	1429	1402	Ta086_3	3030	3001
Ta016_3	783	772	Ta051_3	2121	2087	Ta087_3	2994	2973
Ta017_3	855	825	Ta052_3	2044	1999	Ta088_3	3064	3048
Ta018_3	894	856	Ta053_3	2009	1981	Ta089_3	3014	2972
Ta019_3	899	843	Ta054_3	2079	2027	Ta090_3	3041	2996
Ta020_3	946	893	Ta055_3	2027	1990	Ta091_3	3995	3950
Ta021_3	1509	1472	Ta056_3	2037	1987	Ta092_3	3913	3867
Ta022_3	1441	1387	Ta057_3	2030	1999	Ta093_3	4008	4004
Ta023_3	1576	1523	Ta058_3	2034	2021	Ta094_3	3941	3932
Ta024_3	1479	1446	Ta059_3	2081	2061	Ta096_3	3853	3817
Ta025_3	1512	1474	Ta060_3	2096	2030	Ta097_3	4007	3960
Ta026_3	1511	1462	Ta061_3	1972	1925	Ta099_3	3871	3820
Ta027_3	1505	1470	Ta062_3	1874	1854	Ta100_3	3918	3914
Ta028_3	1487	1422	Ta063_3	1850	1826	Ta101_3	4727	4690
Ta029_3	1509	1471	Ta064_3	1771	1760	Ta102_3	4767	4756
Ta030_3	1414	1407	Ta065_3	1863	1842	Ta103_3	4742	4733
Ta031_3	1007	979	Ta066_3	1800	1797	Ta104_3	4728	4722
Ta032_3	1082	1056	Ta067_3	1854	1836	Ta105_3	4695	4689
Ta033_3	1000	968	Ta069_3	1935	1917	Ta109_3	4767	4722
Ta034_3	1074	1030	Ta070_3	1874	1866			
Ta035_3	1077	1049	Ta071_3	2294	2267			

increase greatly as the problem size increases. Meanwhile, the solutions with much better quality can be found by the EDA. Besides, the distributed scheduling in a real life scenario can be solved offline, so solution quality is more important than the efficiency of the algorithm.

So, it is concluded that the EDA is more effective than the existing methods in solving the DPFSP with the criterion to minimize the makespan, especially for the large-sized problems. The superiority of the EDA owes to the following aspects. (1) With the permutation based encoding and the ECF rule based decoding, it is helpful to obtain a schedule with small makespan. (2) With the well-designed probability model and the suitable updating mechanism, it is helpful to explore search procedure effectively, especially within the promising area of the solution space. (3) With multiple local search operators based on the problem characteristics, it is helpful to improve the good schedule by enhancing the exploitation capability.

6. Conclusions

In this paper, an effective estimation of distribution algorithm was proposed for solving the distributed permutation flow-shop scheduling problem with the criterion to minimize the makespan. To the best of our knowledge, this is the first reported work of the EDA for solving the DPFSP. To be specific, the earliest completion factory rule was employed for the permutation based encoding to generate feasible schedules. A probability model was designed to generate the new individuals and a mechanism was provided to update the probability model suitably. Some local search operators

Table B3
New best makespan obtained by the EDA for large-sized instances ($F=4$).

Instance	Best known	EDA	Instance	Best known	EDA	Instance	Best known	EDA
Ta001_4	528	492	Ta036_4	875	840	Ta071_4	1862	1847
Ta002_4	513	491	Ta037_4	841	832	Ta072_4	1722	1717
Ta003_4	465	442	Ta038_4	827	806	Ta073_4	1807	1777
Ta004_4	537	519	Ta039_4	792	768	Ta074_4	1911	1882
Ta005_4	510	488	Ta040_4	832	819	Ta075_4	1800	1771
Ta006_4	490	479	Ta041_4	1182	1149	Ta076_4	1716	1688
Ta007_4	497	471	Ta042_4	1127	1123	Ta077_4	1772	1757
Ta008_4	499	483	Ta043_4	1150	1127	Ta078_4	1820	1810
Ta009_4	500	475	Ta044_4	1190	1160	Ta079_4	1878	1856
Ta010_4	463	434	Ta045_4	1210	1164	Ta080_4	1885	1855
Ta011_4	809	783	Ta046_4	1178	1153	Ta082_4	2499	2465
Ta012_4	881	832	Ta047_4	1227	1188	Ta083_4	2524	2511
Ta013_4	792	756	Ta048_4	1215	1166	Ta084_4	2498	2482
Ta014_4	720	682	Ta049_4	1130	1111	Ta085_4	2523	2514
Ta015_4	754	720	Ta050_4	1230	1188	Ta086_4	2560	2545
Ta016_4	710	690	Ta051_4	1883	1850	Ta087_4	2545	2528
Ta017_4	789	744	Ta052_4	1785	1768	Ta088_4	2608	2590
Ta018_4	811	773	Ta053_4	1806	1754	Ta089_4	2561	2518
Ta019_4	784	760	Ta054_4	1831	1786	Ta090_4	2560	2537
Ta020_4	834	803	Ta055_4	1807	1766	Ta091_4	3098	3095
Ta021_4	1406	1367	Ta056_4	1779	1759	Ta092_4	3063	3043
Ta022_4	1321	1293	Ta057_4	1768	1753	Ta094_4	3064	3063
Ta023_4	1443	1405	Ta058_4	1819	1780	Ta096_4	3016	2992
Ta024_4	1399	1357	Ta059_4	1846	1820	Ta097_4	3140	3123
Ta025_4	1420	1368	Ta060_4	1844	1795	Ta098_4	3086	3076
Ta026_4	1391	1356	Ta061_4	1505	1494	Ta099_4	3005	3002
Ta027_4	1389	1362	Ta062_4	1449	1431	Ta101_4	3828	3813
Ta028_4	1348	1319	Ta063_4	1426	1407	Ta103_4	3867	3861
Ta029_4	1394	1363	Ta064_4	1373	1354	Ta104_4	3839	3827
Ta030_4	1345	1301	Ta065_4	1436	1426	Ta105_4	3830	3826
Ta031_4	794	782	Ta066_4	1422	1391	Ta106_4	3826	3813
Ta032_4	859	848	Ta067_4	1437	1419	Ta107_4	3938	3909
Ta033_4	807	778	Ta068_4	1402	1383	Ta108_4	3900	3887
Ta034_4	865	829	Ta069_4	1494	1485	Ta109_4	3873	3852
Ta035_4	848	831	Ta070_4	1452	1444	Ta110_4	3899	3886

Table B4
New best makespan obtained by the EDA for large-sized instances ($F=5$).

Instance	Best known	EDA	Instance	Best known	EDA	Instance	Best known	EDA
Ta001_5	469	442	Ta035_5	719	708	Ta069_5	1240	1228
Ta002_5	460	437	Ta036_5	741	723	Ta070_5	1202	1192
Ta003_5	432	393	Ta037_5	729	712	Ta071_5	1614	1586
Ta004_5	487	469	Ta038_5	704	689	Ta072_5	1493	1480
Ta005_5	459	434	Ta039_5	677	661	Ta073_5	1546	1531
Ta006_5	450	436	Ta040_5	716	696	Ta074_5	1634	1620
Ta007_5	445	435	Ta041_5	1048	1029	Ta075_5	1551	1522
Ta008_5	468	441	Ta042_5	1028	995	Ta076_5	1472	1449
Ta009_5	443	427	Ta043_5	1035	1012	Ta077_5	1530	1508
Ta010_5	401	391	Ta044_5	1060	1030	Ta078_5	1570	1549
Ta011_5	747	730	Ta045_5	1048	1035	Ta079_5	1615	1596
Ta012_5	808	770	Ta046_5	1048	1027	Ta080_5	1638	1594
Ta013_5	731	703	Ta047_5	1096	1056	Ta081_5	2242	2227
Ta014_5	657	634	Ta048_5	1061	1031	Ta082_5	2228	2204
Ta015_5	706	671	Ta049_5	1012	990	Ta083_5	2241	2231
Ta016_5	657	640	Ta050_5	1087	1058	Ta084_5	2224	2208
Ta017_5	732	693	Ta051_5	1730	1706	Ta085_5	2252	2239
Ta018_5	749	720	Ta052_5	1650	1626	Ta086_5	2278	2264
Ta019_5	737	712	Ta053_5	1647	1607	Ta087_5	2257	2249
Ta020_5	771	755	Ta054_5	1687	1642	Ta088_5	2316	2303
Ta021_5	1348	1303	Ta055_5	1636	1630	Ta089_5	2280	2249
Ta022_5	1261	1234	Ta056_5	1635	1619	Ta090_5	2267	2258
Ta023_5	1390	1347	Ta057_5	1638	1605	Ta092_5	2559	2543
Ta024_5	1321	1301	Ta058_5	1668	1626	Ta094_5	2553	2541
Ta025_5	1360	1306	Ta059_5	1690	1668	Ta095_5	2554	2538
Ta026_5	1339	1288	Ta060_5	1655	1647	Ta096_5	2509	2503
Ta027_5	1339	1298	Ta061_5	1256	1233	Ta097_5	2622	2604
Ta028_5	1299	1258	Ta062_5	1198	1183	Ta098_5	2582	2564
Ta029_5	1339	1301	Ta063_5	1176	1164	Ta099_5	2536	2509
Ta030_5	1289	1240	Ta064_5	1142	1123	Ta100_5	2570	2565
Ta031_5	680	664	Ta065_5	1189	1178	Ta102_5	3357	3337
Ta032_5	740	723	Ta066_5	1167	1142	Ta104_5	3347	3312
Ta033_5	691	667	Ta067_5	1192	1173	Ta106_5	3310	3290
Ta034_5	740	707	Ta068_5	1161	1142	Ta107_5	3384	3376

Table B5
New best makespan obtained by the EDA for large-sized instances ($F=6$).

Instance	Best known	EDA	Instance	Best known	EDA	Instance	Best known	EDA
Ta001_6	431	410	Ta033_6	630	595	Ta064_6	984	969
Ta002_6	433	404	Ta034_6	657	630	Ta066_6	997	988
Ta003_6	390	369	Ta035_6	656	629	Ta067_6	1015	1011
Ta004_6	460	432	Ta036_6	664	646	Ta068_6	993	985
Ta005_6	436	404	Ta037_6	663	630	Ta069_6	1075	1059
Ta006_6	431	402	Ta038_6	632	614	Ta070_6	1035	1026
Ta008_6	424	414	Ta039_6	607	586	Ta071_6	1432	1421
Ta009_6	413	396	Ta040_6	643	617	Ta072_6	1332	1323
Ta010_6	379	365	Ta041_6	959	941	Ta073_6	1378	1366
Ta011_6	710	695	Ta042_6	924	914	Ta074_6	1466	1446
Ta012_6	754	730	Ta043_6	949	928	Ta075_6	1372	1360
Ta013_6	703	673	Ta044_6	979	957	Ta077_6	1360	1350
Ta014_6	619	605	Ta045_6	967	950	Ta078_6	1400	1382
Ta015_6	670	652	Ta046_6	959	941	Ta079_6	1427	1416
Ta016_6	638	613	Ta047_6	1014	969	Ta080_6	1444	1422
Ta017_6	681	671	Ta048_6	972	943	Ta082_6	2030	2011
Ta018_6	720	692	Ta049_6	928	905	Ta083_6	2061	2047
Ta019_6	704	702	Ta050_6	1006	972	Ta084_6	2042	2019
Ta020_6	746	723	Ta051_6	1636	1599	Ta085_6	2068	2054
Ta021_6	1296	1264	Ta052_6	1559	1524	Ta088_6	2121	2109
Ta022_6	1213	1191	Ta053_6	1526	1512	Ta089_6	2085	2063
Ta023_6	1328	1320	Ta054_6	1580	1546	Ta090_6	2087	2071
Ta024_6	1283	1267	Ta055_6	1550	1528	Ta091_6	2244	2237
Ta025_6	1313	1281	Ta056_6	1556	1515	Ta095_6	2214	2208
Ta026_6	1272	1256	Ta057_6	1530	1509	Ta097_6	2267	2259
Ta027_6	1294	1251	Ta058_6	1549	1527	Ta098_6	2236	2227
Ta028_6	1252	1240	Ta059_6	1598	1569	Ta099_6	2188	2183
Ta029_6	1289	1257	Ta060_6	1575	1545	Ta100_6	2228	2220
Ta030_6	1235	1205	Ta061_6	1070	1060	Ta105_6	2958	2949
Ta031_6	596	592	Ta062_6	1035	1022			
Ta032_6	666	644	Ta063_6	1019	1003			

Table B6
New best makespan obtained by the EDA for large-sized instances ($F=7$).

Instance	Best known	EDA	Instance	Best known	EDA	Instance	Best known	EDA
Ta001_7	415	386	Ta036_7	610	587	Ta065_7	922	910
Ta002_7	403	382	Ta037_7	584	576	Ta066_7	892	878
Ta003_7	375	360	Ta038_7	576	561	Ta067_7	907	898
Ta004_7	432	413	Ta039_7	556	534	Ta068_7	881	876
Ta005_7	410	385	Ta040_7	588	560	Ta069_7	953	933
Ta006_7	396	383	Ta041_7	881	879	Ta070_7	926	916
Ta008_7	402	386	Ta042_7	869	848	Ta071_7	1306	1298
Ta009_7	401	377	Ta043_7	893	866	Ta072_7	1213	1211
Ta010_7	368	347	Ta044_7	905	890	Ta073_7	1245	1243
Ta011_7	699	670	Ta045_7	908	890	Ta074_7	1343	1322
Ta012_7	724	706	Ta046_7	895	880	Ta075_7	1256	1244
Ta013_7	688	650	Ta047_7	944	907	Ta076_7	1192	1190
Ta014_7	601	586	Ta048_7	900	883	Ta077_7	1253	1231
Ta015_7	637	628	Ta049_7	869	848	Ta078_7	1272	1261
Ta016_7	615	591	Ta050_7	943	910	Ta079_7	1303	1299
Ta017_7	674	671	Ta051_7	1553	1523	Ta080_7	1319	1302
Ta018_7	695	692	Ta052_7	1476	1449	Ta081_7	1904	1901
Ta020_7	715	707	Ta053_7	1470	1438	Ta082_7	1896	1884
Ta021_7	1263	1243	Ta054_7	1514	1469	Ta083_7	1919	1907
Ta024_7	1247	1241	Ta055_7	1479	1463	Ta085_7	1933	1912
Ta025_7	1271	1253	Ta056_7	1469	1447	Ta086_7	1941	1938
Ta027_7	1237	1232	Ta057_7	1465	1438	Ta088_7	2007	1972
Ta029_7	1245	1240	Ta058_7	1481	1455	Ta089_7	1947	1933
Ta030_7	1201	1168	Ta059_7	1514	1502	Ta091_7	2004	1997
Ta031_7	565	539	Ta060_7	1509	1477	Ta096_7	1954	1948
Ta032_7	610	589	Ta061_7	956	944	Ta097_7	2028	2024
Ta033_7	562	539	Ta062_7	913	905	Ta098_7	1989	1983
Ta034_7	604	573	Ta063_7	909	891	Ta102_7	2733	2732
Ta035_7	586	570	Ta064_7	869	860			

were also designed based on the problem characteristics to enhance the exploitation. The influence of parameter setting was investigated by using DOE test. Extensive testing results and comparisons demonstrated the effectiveness of the proposed EDA in solving the DPFSP. The new best-known solutions for 17 out of 420 small-sized instances and 589 out of 720 large-sized instances were presented as well. The future work is to design EDA-based algorithms for distributed job-shop scheduling problem and multi-objective distributed schedule problem.

Acknowledgments

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

Appendix A

See Table A1.

Appendix B

See Tables B1–B6.

References

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

Baluja, S., Davies, S., 1997. Using optimal dependency-trees for combinatorial optimization: learning the structure of the search space. In: Proceedings of the 14th International Conference on Machine Learning, San Francisco, 30–38.

Cesar, R.M., Bengoetxea, E., Bloch, I., Larranaga, P., 2005. Inexact graph matching for model-based recognition: evaluation and comparison of optimization algorithms. Pattern Recognition 38, 2099–2113.

Chan, F.T.S., Chung, S.H., Chan, L.Y., Finke, G., Tiwari, M.K., 2006. Solving distributed FMS scheduling problems subject to maintenance: genetic algorithms approach. Robotics and Computer-Integrated Manufacturing 22, 493–504.

Chan, F.T.S., Chung, S.H., Chan, P.L.Y., 2005. An adaptive genetic algorithm with dominated genes for distributed scheduling problems. Expert Systems with Applications 29, 364–371.

Chen, S.H., Chen, M.C., 2013. Addressing the advantages of using ensemble probabilistic models in estimation of distribution algorithms for scheduling problems. International Journal of Production Economics 141, 24–33.

Cheng, T.C.E., Ding, Q., Lin, B.M.T., 2004. A concise survey of scheduling with time-dependent processing times. European Journal of Operational Research 152, 1–13.

Cheng, T.C.E., Janiak, A., 2000. A permutation flow-shop scheduling problem with convex models of operation processing times. Annals of Operations Research 96, 39–60.

Cheng, T.C.E., Kovalyov, M.Y., 2003. Scheduling a single server in a two-machine flow shop. Computing 70, 167–180.

Cheng, T.C.E., Wu, C.C., Chen, J.C., Wu, W.H., Cheng, S.R., 2013. Two-machine flowshop scheduling with a truncated learning function to minimize the makespan. International Journal of Production Economics 141, 79–86.

Chung, C.S., Flynn, J., Kirca, O., 2002. A branch and bound algorithm to minimize the total flow time for m-machine permutation flowshop problems. International Journal of Production Economics 79, 185–196.

De Bonet, J.S., Isbell Jr., C.L., Viola, P., 1997. MIMIC: finding optima by estimating probability densities, Advances in Neural Information Processing Systems. MIT Press, Cambridge pp. 424–430.

De Giovanni, L., Pezzella, F., 2010. An improved genetic algorithm for the distributed and flexible job-shop scheduling problem. European Journal of Operational Research 200, 395–408.

Gao, J., Chen, R., Deng, W., 2012. An efficient tabu search algorithm for the distributed permutation flowshop scheduling problem. International Journal of Production Research. http://dx.doi.org/10.1080/00207543.2011.644819.

Garey, M.R., Johnson, D.S., Sethi, R., 1976. The complexity of flowshop and jobshop scheduling. Mathematics of Operations Research 1, 117–129.

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

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

Hejazi, S.R., Saghafian, S., 2005. Flowshop-scheduling problems with makespan criterion: a review. International Journal of Production Research 43, 2895–2929.

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

Jia, H.Z., Fuh, J.Y.H., Nee, A.Y.C., Zhang, Y.F., 2002. Web-based multi-functional scheduling system for a distributed manufacturing environment. Concurrent Engineering-Research and Applications 10, 27–39.

Jia, H.Z., Fuh, J.Y.H., Nee, A.Y.C., Zhang, Y.F., 2007. Integration of genetic algorithm and Gantt chart for job shop scheduling in distributed manufacturing systems. Computers & Industrial Engineering 53, 313–320.

Jia, H.Z., Nee, A.Y.C., Fuh, J.Y.H., Zhang, Y.F., 2003. A modified genetic algorithm for distributed scheduling problems. Journal of Intelligent Manufacturing 14, 351–362.

Johnson, S.M., 1954. Optimal two- and three-stage production schedules with setuptimes included. Naval Research Logistics Quarterly 1, 61–68.

Kahn, K.B., Castellion, G.A., Griffin, A., 2004. The PDMA Handbook of New Product Development, 2nd ed.. Wiley, New York.

Larranaga, P., Lozano, J.A., 2002. Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation. Springer, Netherlands.

Lin, B.M.T., Lu, C.Y., Shyu, S.J., Tsai, C.Y., 2008. Development of new features of ant colony optimization for flowshop scheduling. International Journal of Production Economics 112, 742–755.

Mühlenbein, H., Paass, G., 1996. From recombination of genes to the estimation of distributions I: binary parameters. Lecture Notes in Computer Science 1141, 178–187.

Montgomery, D.C., 2005. Design and Analysis of Experiments. John Wiley & Sons, Arizona.

Moon, C., Kim, J., Hur, S., 2002. Integrated process planning and scheduling with minimizing total tardiness in multi-plants supply chain. Computers & Industrial Engineering 43, 331–349.

Mühlenbein, H., Mahnig, T., 1999. Convergence theory and applications of the factorized distribution algorithm. Jounal of Computing and Information Technology 7, 19–32.

Naderi, B., Ruiz, R., 2010. The distributed permutation flowshop scheduling problem. Computers & Operations Research 37, 754–768.

Pelikan, M., Goldberg, D.E., Cantú-Paz, E., 1999. BOA: the bayesian optimization algorithm. In: Proceedings of the Genetic and Evolutionary Computation, San Francisco, 525–532.

Pelikan, M., Mühlenbein, H., 1999. The bivariate marginal distribution algorithm. In: Benítez, J.M. (Ed.), Advances in Soft Computing: Engineering Design and Manufacturing. Springer-Verlag, London.

Ruiz, R., Maroto, C., 2005. A comprehensive review and evaluation of permutation flowshop heuristics. European Journal of Operational Research 165, 479–494.

Saeys, Y., Degroeve, S, Aeyels, D., Van de Peer, Y., Rouze, P., 2003. Fast feature selection using a simple estimation of distribution algorithm: a case study on splice site prediction. Bioinformatics 19, 179–188.

Sagarna, R., Lozano, J., 2005. On the performance of estimation of distribution algorithms applied to software testing. Applied Artificial Intelligence 19, 457–489.

Shabtay, D., Bensoussan, Y., Kaspi, M., 2013. A bicriteria approach to maximize the weighted number of just-in-time jobs and to minimize the total resource consumption cost in a two-machine flow-shop scheduling system. International Journal of Production Economics 136, 67–74.

Suliman, S.M.A., 2000. A two-phase heuristic approach to the permutation flow-shop scheduling problem. International Journal of Production Economics 64, 143–152.

Sun, L.H., Sun, L.Y., Wang, M.Z., Wang, J.B., 2012. Flow shop makespan minimization scheduling with deteriorating jobs under dominating machines. International Journal of Production Economics 138, 195–200.

Taillard, E., 1993. Benchmarks for basic scheduling problems. European Journal of Operational Research 64, 278–285.

Tseng, L.Y., Lin, Y.T., 2010a. A genetic local search algorithm for minimizing total flowtime in the permutation flowshop scheduling problem. International Journal of Production Economics 127, 121–128.

Tseng, L.Y., Lin, Y.T., 2010b. A hybrid genetic algorithm for no-wait flowshop scheduling problem. International Journal of Production Economics 128, 144–152.

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

Wang, L., Shen, W., 2007. Process Planning and Scheduling for Distributed Manufacturing. Springer, London.

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

Wang, L., Wang, S.Y., Liu, M., 2013a. A Pareto-based estimation of distribution algorithm for the multi-objective flexible job-shop scheduling problem.

International Journal of Production Research. http://dx.doi.org/10.1080/00207543.2012.752588.
Wang, L., Wang, S.Y., Xu, Y., Zhou, G., Liu, M., 2012b. A bi-population based estimation of distribution algorithm for the flexible job-shop scheduling problem. Computers & Industrial Engineering 62, 917–926.
Wang, S.Y., Wang, L., Liu, M., Xu, Y., 2013b. An effective estimation of distribution algorithm for the flexible job-shop scheduling problem with fuzzy processing time. International Journal of Production Research. http://dx.doi.org/10.1080/00207543.2013.765077.
Wang, S.Y., Wang, L., Liu, M., Xu, Y., 2013c. An enhanced estimation of distribution algorithm for solving hybrid flow-shop scheduling problem with identical parallel machines. The International Journal of Advanced Manufacturing Technology. http://dx.doi.org/10.1007/s00170-013-4819-y.
Wang, X.L., Xie, X.Z., Cheng, T.C.E., 2013d. A modified artificial bee colony algorithm for order acceptance in two-machine flow shops. International Journal of Production Economics 141, 14–23.