An Effective Estimation of Distribution Algorithm for Solving Uniform Parallel Machine Scheduling Problem with Precedence Constraints

Chu-ge Wu, Ling Wang, Xiao-long Zheng
Department of Automation, Tsinghua University
Beijing, China
E-mail: wucg15@mails.tsinghua.edu.cn; wangling@tsinghua.edu.cn; zhengxl11@mails.tsinghua.edu.cn

***Abstract*—In this paper, an effective estimation of distributed algorithm (eEDA) is proposed to solve the uniform parallel machine scheduling problem with precedence constraints (prec-UFPMSP). In the eEDA, the permutation-based encoding scheme is adopted and the earliest finish time (EFT) method is used to decode the solutions to the detail schedules. A new effective probability model is designed to describe the relative positions of the jobs. Based on such a model, an incremental learning based updating method is developed and a sampling mechanism is proposed to generate feasible solutions with good diversity. In addition, the Taguchi method of design-of-experiment (DOE) method is used to investigate the effect of key parameters on the performance of the eEDA. Finally, numerical tests are carried out to demonstrate the superiority of the probability model, and the comparative results show that the eEDA outperforms the existing algorithm for most cases.**

Keywords—precedence constraint scheduling, uniform parallel machine, EDA, relative position probability model

I. INTRODUCTION

The parallel machine scheduling problem (PMSP) is one of the typical scheduling problems, which has been proved to be NP-hard [1]. The precedence constraints widely exist in the real world manufacturing systems, such as allocation of workflow in cloud computing [2], wafer probing scheduling [3], and block erecting in shipbuilding [4]. Therefore, the precedence constraints between the jobs need to be considered for the PMSP in many real-world situations. Due to its practical significance, the PMSP with precedence constraints has attracted increasing attention. This paper deals with the uniform parallel machine scheduling problem with precedence constraints (prec-UFPMSP) with makespan criterion. The problem is more complex than PMSP, because it should determine the processing sequence of the jobs and the machine assignment together. In addition, it is a relaxation of the prevailing complex scheduling problems, such as the resource constrained project scheduling problem (RCPSP) [5]. The effective solution algorithms for the prec-UFPMSP can also be promoted to solve the RCPSP. Therefore, it is of practical and academic significance to design effective algorithms for the prec-UFPMSP.

This work is supported by the National Key Basic Research and Development Program of China (No. 2013CB329503) and the National Science Fund for Distinguished Young Scholars of China (No. 61525304).

The estimation of distribution algorithm (EDA) [6] is a population-based evolutionary algorithm, which has been one of the focus among the evolutionary computing techniques in solving the complex optimization problems during recent years. The EDA generates new solutions by sampling the probability model, which is used to describe the distribution of the promising solutions in the search space. The EDA has been shown to be effective in solving many academic and engineering optimization problems in a variety of fields [7]. Inspired by the successful applications of the EDA, especially in the field of production scheduling, the EDA will be adopted in this paper to solve the prec-UFPMSP. To well solve the problem, a novel probability model of EDA is designed for the prec-UFPMSP to describe the relative positions between the jobs; and a novel sampling mechanism is proposed to generate feasible solutions so as to handle the precedence constraints effectively. Numerical tests show that the best known solutions of 33 out of 48 benchmarking instances have been updated by the proposed algorithm.

The reminder of the paper is organized as follows: Section 2 reviews the related work on this problem. The description and a mathematical model of the prec-UFPMSP are provided in Section 3. Then, the eEDA is proposed in details in Section 4. In Section 5, the effect of parameter setting is investigated and numerical comparisons are provided. Finally, the paper is ended with some conclusions and future work in Section 7.

II. RELATED WORK

For the prec-PMSP, there exist the following studies about the mathematical models. Aho et al [8] studied the jobs with unit processing time on parallel machines with makespan criterion. Munier et al [9] presented the approximation bounds for the PMSP with the constrains, such as ordinary precedence constraints, release dates and delivery times. Chekuri et al [10] analyzed the minimization of total weighted completion times on a single machine. Liu and Yang [11] also presented a mathematical model for prec-UFPMSP.

As for the solution algorithms, there exist the following typical work for the parallel machine scheduling problem with

978-1-5090-0623-6/16/$31.00

precedence constraints. In [9], Munier et al proposed heuristic algorithms for the prec-PMSP. In [12], a heuristic method was proposed to solve the prec-UFPMSP, and the local search methods and simulated annealing mechanism were also developed based on the heuristic method. But only the precedence constraint in chains (a job has at most one succeed and one predecessor) was considered. Liu and Yang [11] also developed some heuristic methods. Hassan et al [13] adopted the genetic algorithm (GA) to solve the prec-UFPMSP and realized a heuristic method for comparison. Under the background of the allocation of the workflow in cloud environment, Szabo et al [2] proposed an algorithm to solve the prec-PMSP. A problem specific operator was designed, and both single-objective and multi-objective situations were considered. Besides, setup times of jobs were added to generalize the problem to more complex ones. And the list scheduling methods were adopted in [5, 14].

Due to the complexity caused by the precedence constraints between the jobs, the generation way of the feasible solutions also caught much attention. Hassan et al [13] guaranteed the feasibility through a repairing mechanism. Omara et al [15] designed a specific crossover operator and a mutation operator to generate feasible solutions. Wang et al [16] proposed an EDA with a special probability model and a sampling mechanism to generate feasible solutions. Afzalirad and Rezaeian [17] considered the unrelated parallel machine problem with constraints, and they proposed a new corrective algorithm to obtain the feasible solutions at every stage of the algorithm.

III. PROBLEM FORMULATION

The prec-UFPMSP can be described as follows. There are n jobs to be processed on m machines with different speed ratios. Each job can be executed on any machine. The processing time of job j on machine i is calculated as $p_{ij}=p_j \times s_i$, where s_i denotes the speed ratio of machine i and p_j denotes the processing time of job j. The objective is to determine the job processing sequences on all the machines so as to minimize the makespan. In addition, the precedence relationship should be considered. If one job is precedent of the other, then the latter one cannot be processed until the former one finishes its processing. The problem can be represented as $Q|\text{prec}|C_{\max}$.

A. Notation

n: the number of jobs to be processed;

m: the number of machines;

C_j: $C_j \geq 0$, the completion time of job j;

p_{ji}: the processing time of job j on machine i;

x_{jir}: $\{0,1\}$. $x_{jir}=1$ illustrates that job j is the r-th job processed on machine i, otherwise, $x_{jir}=0$.

M: a very large constant.

$i \rightarrow j$: job i is precedent of job j.

B. Mathematical model [13]

$$\min \max_{j=1,2,...,n} C_j \tag{1}$$

Subject to:

$$\sum_{i=1}^{m}\sum_{r=1}^{n} x_{jir} = 1. \quad j = 1,2,...,n \tag{2}$$

$$\sum_{j=1}^{n} x_{jir} \leq 1. \quad i = 1,2,...,m; \forall r \tag{3}$$

$$\sum_{j_1=1}^{n} x_{j_1 ir+1} - \sum_{j_2=1}^{n} x_{j_2 ir} \leq 0. \quad i = 1,2,...,m; r = 1,2,...,n-1 \tag{4}$$

$$C_{j_1} - C_{j2} + M(2 - x_{j_1 ir+1} - x_{j_2 ir}) \geq p_{j_1 i}. \quad i = 1,2,...,n; r = 1,2,...,n-1; \forall j_1 \neq j_2 \tag{5}$$

$$C_{j_1} - C_{j_2} \geq \sum_{i=1}^{m}\sum_{r=1}^{n} p_{j_1 i} x_{j_1 ir}. \quad \forall j_1 \rightarrow j_2 \tag{6}$$

$$C_j \geq \sum_{i=1}^{n} p_{ji} x_{ji1}. \quad j = 1,2,...,n \tag{7}$$

where (1) is the objective function; and constraint (2) guarantees that each job must be processed only once; and (3) ensures each machine processes no more than one job at the same time; and constraint (4) means that the jobs on a certain machine are in order; and constraint (5) gives the limitation to the completion times of the adjacent jobs on the same machine; and (6) ensures the precedence constraints between jobs; and constraint (7) considers the completion time of the first job on each machine.

IV. THE PROPOSED EDA

In this section, the proposed algorithm will be introduced in details. Firstly, the encoding and decoding methods are proposed; secondly, the probability model and its initialization mechanism are presented. Then, the sampling method is introduced. Finally, the flowchart of the eEDA is provided.

A. Encoding and decoding method

In the eEDA, a job processing order permutation π is used to represent a solution, where π_i represents the i-th job to be scheduled.

In order to generate the feasible solutions in accordance with the precedence constraints, job i is placed before job j in the permutation if job i is the precedent of job j. These permutations that do not violate the precedence constraints are defined legal ones. The legal permutations consist of a dominant set of schedules, which contain at least one optimal schedule [14]. It guarantees that the encoding mechanism can cover the best solution. Therefore, the eEDA is designed to generate legal permutations. In addition, by considering the legal permutations only, it can reduce the search space and save computational cost.

The eEDA maps the solutions into the detail schedules by using the earliest finish time formulation (EFT). For each solution, the jobs in the permutation are assigned to the

machine based on the EFT until all the jobs are scheduled. To explain the decoding mechanism, a problem with 2 machines and 4 jobs is used as an example. Suppose s_1=2, s_2=3, and p_j=1 (j=1,2,3,4). The precedence constraints between jobs are 2→3, 2→4 and 3→4.

Without loss of generality, a legal processing permutation (1, 2, 3, 4) is considered. Let C_{ji} denote the completion time of job j on machine i. Job 1 is scheduled first according to the permutation. Since p_{11}=2×1=2 and p_{12}=3×1=3, thus C_{11}=2 and C_{12}=3. Because $C_{11} < C_{12}$, job 1 is scheduled on machine 1 according to the EFT. Then, job 2 is considered. There is no precedence constratint between job 1 and job 2, and C_{22}=p_{22}=3 < C_{21}=C_{11}+p_{21}=2+2=4, so job 2 is scheduled on machine 2. Since there exists precedence constraint between job 2 and job 3, job 3 cannot start until job 2 finishes its processing. The completion time of job 3 on different machines can be calculated as C_{31}=C_2+p_{31}=3+2=5 and C_{32}=C_2+p_{32}=3+3 =6. Job 3 is scheduled to machine 1, since $C_{31} < C_{32}$. And job 4 is assigned to machine 1 because $C_{41} < C_{42}$. Therefore, the Gantt chart of the corresponding schedule is illustrated in Fig. 1.

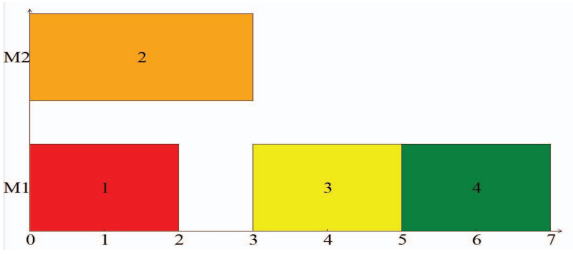

Fig. 1. Corresponding Gantt chart of the solution

B. Probability model

The probability model plays an important role in the EDA since it describes the distribution of the solutions in the search space and the solutions are generated by sampling the model. If the model can well reflect the characteristics of the problem, it may be easier to obtain better solutions.

In the prec-UFPMSP, the precedence constraints provide the relative position information between some job pairs. In many existing work, the absolute position model was adopted. Such kind of model only describes the probability that a job is placed at different positions, but it cannot represent the relative position relationship between jobs, which is related to the feasibility of solutions in the problem with precedence constraints. To overcome the shortcoming of the absolute position model, a relative position is designed in the eEDA. The model transfers the precedence constraint information into the probability value to describe the search space. Thus, a probability model P(g) considering the relative position is designed as follows.

$$P(g)=\begin{bmatrix} 0 & \rho_{12}(g) & \cdots & \rho_{1n}(g) \\ \rho_{21}(g) & 0 & \cdots & \rho_{2n}(g) \\ \vdots & \vdots & \ddots & \vdots \\ \rho_{n1}(g) & \rho_{n2}(g) & \cdots & 0 \end{bmatrix} \tag{8}$$

In (8), element $\rho_{ij}(g)$ represents the probability that job i is placed in front of job j in the permutation at the g-th generation. Clearly, $\forall i,j,g,\rho_{ij}(g)+\rho_{ji}(g)=1$. Note that, $\forall j,g,\rho_{jj}(g)=0$ means that each job is impossible to be placed in front of itself.

The initialization mechanism is designed to make use of the information of the problem. As mentioned before, if job i is the precedent of job j, job i must be placed in front of job j. Therefore, $\rho_{ij}(0)$ is set to be 1 and $\rho_{ji}(0)$ is 0. On the other hand, if the precedence relationship between the jobs is not given, $\rho_{ij}(0)$ and $\rho_{ji}(0)$ are both set to be 0.5. The probability values of ρ_{ij} are initialized as follows:

$$\rho_{ij}(0)=\begin{cases} 1, i\to j \\ 0, j\to i \\ 0.5, otherwise. \end{cases} \tag{9}$$

In the procedure of updating the model, all solutions are sorted in an ascending order with respect to the makespan, and the top *EPSize* solutions are selected to construct the elite population.

The probability values of the model are updated using the population based incremental learning method (PBIL) [18] as follows:

$$p_{ij}(g+1)=(1-\alpha)p_{ij}(g)+\alpha\frac{1}{EPSize}\sum_{k=1}^{EPSize}I_{ij}^{k}(g) \tag{10}$$

where $\alpha\in(0,1)$ represents the learning rate and $I_{ij}^{k}(g)$ is the following indicator function corresponding to the k-th solution of the elite population.

$$I_{ij}^{k}(g)=\begin{cases} 1, \text{if job } i \text{ is before job } j \text{ in the } k\text{ - th individual} \\ 0, \text{otherwise} \end{cases} \tag{11}$$

C. Sampling method

The population is generated by sampling the probability model in the eEDA and the elements in the permutation are sampled independently.

For each element π_i (i=1…n) in the permutation, the Pseudo code of the sampling procedure is shown in Fig 2.

```
sample_array[n]=0;
sum=0;
for j = 0; j ≤ n; j++ do
    sample_array[j] = ∏_{k∈C} P[k][j];
    sum += sample_array[j];
for j = 0; j ≤ n; j++ do
    sample_array[j] = sample_array[j]/sum;
π_i = RouletteWheelMethod(sample_array);
return π_i;
```

Fig. 2. Pseudo code of sampling an element in the permutation

In the procedure, a temporary probability array *sample_array* is built to sample the job number, and C is the set of the jobs that are not scheduled till now, and the function RouletteWheelMethod (*sample_array*) illustrates that π_i is generated by the roulette wheel method based on *sample_array*. The sampling method guarantees that the generated permutations are legal. Besides, due to the designed sampling method, all the solutions generated by sampling $P(0)$ do not violate the precedence constraints. According to (10), the fixed 0 and 1 in the probability model will not be altered, which guarantees the solutions sampled based on the probability model are legal during the whole procedure of evolution.

D. Flowchart

With the above design, the flowchart of the proposed eEDA is illustrated in Fig. 3.

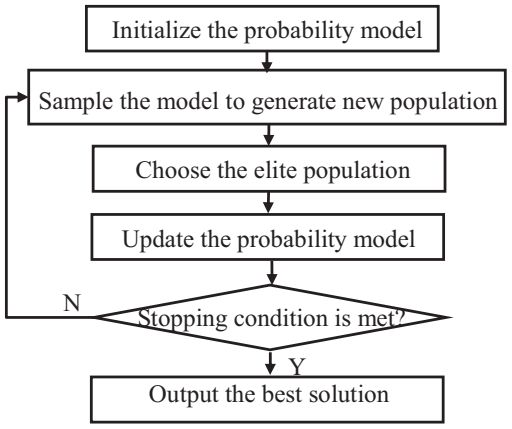

Fig. 3. Flowchart of the eEDA

In the initialization phase, the probability model is initialized based on the precedence constraints. Then, the eEDA uses the EDA-based mechanism to sample the relative position probability model to construct the population. After that, it sorts the population and determines the best *EPSize* individuals as the elite population. Then, it updates the probability values according to the elite population. In such a way, the algorithm runs until the stopping condition is met.

V. Numerical Results and Comparisons

To test the performances of the eEDA, numerical tests are carried out using a set of benchmarking instances [13]. The testing set consists of 48=4×4×3 instances, where n = {20, 50, 100, 200} and m = {2, 5, 10, 20}. For each combination of the number of machine and job, 3 different instances are generated, in which the densities of precedence constraints are high, medium and low, respectively. The eEDA is implemented in C language and run on a personal computer with a 2.83 GHz processor and 4GB RAM.

A. Parameter setting

The eEDA has three key parameters: *PSize* (size of the population), and η (percentage of elite solutions, *EPSize* = *PSize*×η%), and α (learning rate). To investigate the effect of these parameters, the Taguchi method of design-of-experiment method (DOE) [19] is employed, where a moderate scaled instance (m=5, n=100 and the density is high) is used. The stopping criterion is set the same one as [13], i.e. 10 seconds.

Considering the scale of the benchmarking instances, four factor levels are employed and the different values of these parameters are listed in Table I. The orthogonal array $L_{16}(4^3)$ is chosen accordingly. For each parameter combination, the eEDA is run 20 times independently and the average value of 20 runs is used as the response variable (RV) value. The orthogonal array and RV values are listed in Table II.

TABLE I. Factor Levels of Parameters

Parameters	Factor level			
	1	*2*	*3*	*4*
PSize	100	150	200	250
η(%)	1	2	3	4
α	0.05	0.10	0.15	0.20

TABLE II. The RV Value

Experiment Number	Factor			RV
	Psize	η(%)	α	
1	1	1	1	6590.55
2	1	2	2	6590.65
3	1	3	3	6590.95
4	1	4	4	6591.05
5	2	1	2	6590.65
6	2	2	1	6590.60
7	2	3	4	6590.75
8	2	4	3	6590.90
9	3	1	3	6591.00
10	3	2	4	6590.80
11	3	3	1	6590.95
12	3	4	2	6590.90
13	4	1	4	6590.80
14	4	2	3	6590.85
15	4	3	2	6590.80
16	4	4	1	6591.10

According to the results, the influence trend of each parameter is illustrated in Fig. 4. The significance rank of each parameter is analyzed and the results are listed in Table III.

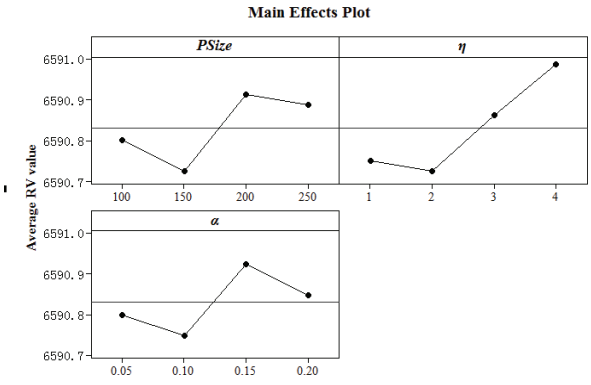

Fig. 4. The influence trend of each parameter

From Fig. 4 and Table III, it can be seen that all the parameter values should be set neither too small nor too large. The percentage of elite population is of the most significant impact. A small value of η may result in that useful information obtained from the elite population is not enough to update the probability model, while a big value of η may bring the worse solutions into the elite population and decrease the accuracy of the model. The significance of *PSize* ranks the second. The *Psize* affects the balance of search breadth and depth. Thus, a moderate value is preferred. Although α has the slightest influence, a small value of α may slow the convergence of the probability model, while a large value may lead to premature convergence.

TABLE III. RESPONSE TABLE

Level	*PSize*	η(%)	α
1	6591	6591	6591
2	6591	6591	6591
3	6591	6591	6591
4	6591	6591	6591
Delta	0	0	0
Rank	2	1	3

For further investigation, non-parametric tests are carried out. According to the results of DOE, *PSize* = 150, η = 2, and α = 0.1 are recommended for parameter setting. In the non-parametric tests, we alter one parameter value and keep the other two parameters fixed as the recommended values to investigate the effect. For each parameter combination, the eEDA is run 20 times independently and the average makespan value of 20 runs is used as RV value. Then, the Kruskal-Wallis test (with 95% confidence) is carried out based on the RV values and the results are shown in Table IV.

TABLE IV. KRUSKAL-WALLIS TESTS FOR THE PARAMETERS

	1	2	3	4	*p*-value
PSize	100	150	200	250	0.175
η	1	2	3	4	0.854
α	0.05	0.10	0.15	0.20	0.323

It can be seen from Table IV that the difference between the eEDAs with variable parameter settings is not significant. In other word, the eEDA is not sensitive to the parameter values. Due to the robustness of the eEDA, the choice of parameter combination does not affect its performance too much. To obtain better results, the recommended values based on the results of DOE are adopted.

B. Superority of the probability model

To testify the superiority of the relative position probability model, the eEDA is compared with the EDA adopting the absolute position model. Considering that there is no EDA applied to solve the prec-UFPMSP while the problem can be regarded as a special case of the RCPSP, we re-implement the pareto-archived EDA (PAEDA) [16] for comparisons, which is proposed for the RCPSP by using the absolute position model. For a fair comparison, the two algorithms use the same parameter values as aforementioned values but are different in terms of the probability model and the sampling method.

To be specific, the job processing permutation is adopted by the PAEDA to encode the problem in [16], and the following absolute position probability model is used.

$$M\_act(g)=\begin{bmatrix} \alpha_{11}(g) & \cdots & \alpha_{1n}(g) \\ \vdots & \ddots & \vdots \\ \alpha_{n1}(g) & \cdots & \alpha_{nn}(g) \end{bmatrix} \tag{12}$$

where element $\alpha_{ji}(g)$ represents the probability that job j is set at the i-th place in the permutation at the g-th generation.

The probability model is initialized uniformly, that is $\alpha_{ij}(0)=1/n, \forall i, j$. To obtain feasible solutions, the probability vector of the PAEDA is defined as follows.

$$P_{\alpha_{ji}}=\alpha_{ji}\Big/\sum_{h=1}^{S_E}\alpha_{hi} \tag{13}$$

where $P_{\alpha_{ji}}$ represents the sampling probability that job j is chosen at the i-th position in the permutation, and S_E denotes the set of the active jobs at position i (the jobs whose preceding jobs have been scheduled). After job j is chosen, the j-th row of the probability model ($(\alpha_{j1}(g),\alpha_{j2}(g)\cdots\alpha_{jn}(g))$ in the *M_act*(g) is set as 0 to avoid generating the repeated job j in the solution.

Both algorithms are run 20 times independently. For further statistical analysis, the student's t-test is used to show the whether the proposed model is superior over that of the PAEDA significantly. The average results and the statistical test results are listed in Table V. If the *p*-value is less than 0.05, the difference between the algorithms are statistically significant at 95% confidence level, which is denoted 'Y'; otherwise, 'N'.

It can be seen from Table V that, for most low density cases, the eEDA outperforms the PAEDA significantly. For the medium density cases, the PAEDA outperforms the eEDA on n=20. The possible reason is the PAEDA is of better exploitation ability due to its probability model and sampling mechanism. Thus, the PAEDA performs better on the small scaled instances. For the high density cases, there are more precedence constraints between the jobs and the legal solutions are limited. Both the two algorithms can obtain the best solutions on n=20, and on n=50 the PAEDA still outperforms the eEDA. However, on the large scaled instances, the eEDA is much superior to the PAEDA. So, it can be concluded that the eEDA is able to locate good solutions quickly and has better exploration ability.

C. Comparison with existing algorithm

In [13], Hassan et al proposed a powerful GA for the prec-UFPMSP and provided the best results with the stopping criterion of 10 seconds time limits. Here, we also set the time limit with 10 seconds. In Table VI, the best and average results are listed by running the eEDA 20 times independently.

TABLE V. AVERAGE RESULTS OF DIFFERENT PROBABILITY MODEL AND SAMPLING MECHANISM

(*m*, *n*)	High				Medium				Low			
	PAEDA	eEDA	*p*-value	Sig	PAEDA	eEDA	*p*-value	Sig	PAEDA	eEDA	*p*-value	Sig
(2,20)	**798.00**	**798.00**	1.0000	N	**648.40**	648.55	1.0000	N	**648.00**	**648.00**	1.0000	N
(2,50)	**1953.00**	1962.05	1.0000	N	1688.70	**1688.00**	1.0000	N	1688.30	**1688.00**	1.0000	N
(2,100)	3690.55	**3662.75**	0.0000	Y	3256.40	**3245.75**	0.0000	Y	3248.00	**3243.20**	0.0000	Y
(2,200)	6908.15	**6747.70**	0.0000	Y	6892.25	**6745.40**	0.0000	Y	6809.75	**6729.80**	0.0000	Y
(5,20)	**1425.00**	**1425.00**	1.0000	N	**861.30**	878.60	1.0000	N	838.20	**837.10**	0.2330	N
(5,50)	**3356.65**	3362.60	0.9490	N	2385.95	**2330.05**	0.0000	Y	2026.00	**1997.85**	0.0000	Y
(5,100)	6606.15	**6590.65**	0.0000	Y	4482.75	**4397.15**	0.0000	Y	4084.95	**4013.00**	0.0000	Y
(5,200)	8929.35	**7945.40**	0.0000	Y	8669.70	**7817.85**	0.0000	Y	8176.00	**7588.65**	0.0000	Y
(10,20)	**798.00**	**798.00**	1.0000	N	446.55	**438.00**	1.0000	N	432.00	**432.00**	1.0000	N
(10,50)	1829.50	**1828.00**	1.0000	N	1258.80	**1251.00**	1.0000	N	1012.50	**985.00**	0.0000	Y
(10,100)	3554.25	**3551.70**	0.1270	N	2936.15	**2932.05**	0.2180	N	2034.80	**1961.25**	0.0000	Y
(10,200)	4324.80	**3773.25**	0.0000	Y	4373.25	**3858.80**	0.0000	Y	3614.20	**3206.90**	0.0000	Y
(20,20)	**798.00**	**798.00**	1.0000	N	**510.30**	510.50	0.7310	N	432.00	**432.00**	0.0000	Y
(20,50)	**1843.90**	1854.85	1.0000	N	1458.60	**1453.20**	0.0000	Y	1042.15	**1002.00**	0.0000	Y
(20,100)	3556.00	**3553.00**	0.0000	Y	**2947.65**	2947.95	0.5250	N	2080.85	**2017.90**	0.0000	Y
(20,200)	4484.15	**3899.30**	0.0000	Y	4467.3	**3885.45**	0.0000	Y	3410.25	**2881.40**	0.0000	Y

TABLE VI. BEST AND AVERAGE RESULTS OF GA AND EDA

(*m*, *n*)	High			Medium			Low		
	GA	eEDA		GA	eEDA		GA	eEDA	
	Best	Best	Avg	Best	Best	Avg	Best	Best	Avg
(2,20)	**798**	**798**	**798.00**	**648**	**648**	648.55	**648**	**648**	**648.00**
(2,50)	1971	**1953**	1962.05	**1688**	**1688**	**1688.00**	**1688**	**1688**	**1688.00**
(2,100)	3781	**3658**	3662.75	3281	**3243**	3245.75	3251	**3243**	3243.20
(2,200)	6756	**6736**	6747.70	6751	**6736**	6745.40	6731	**6728**	6729.80
(5,20)	**1425**	**1425**	**1425.00**	**866**	873	878.60	860	**836**	837.10
(5,50)	**3351**	**3351**	3362.60	2485	**2297**	2330.05	2114	**1978**	1997.85
(5,100)	6780	**6590**	6590.65	4594	**4366**	4397.15	4207	**3981**	4013.00
(5,200)	8271	**7871**	7945.40	8125	**7772**	7817.85	7688	**7551**	7588.65
(10,20)	**798**	**798**	**798.00**	452	**438**	**438.00**	440	**432**	**432.00**
(10,50)	1835	**1828**	**1828.00**	1296	**1251**	**1251.00**	1010	**972**	985.00
(10,100)	3580	**3540**	3551.70	**2390**	2903	2932.05	2002	**1924**	1961.25
(10,200)	4030	**3721**	3773.25	3831	**3827**	3858.80	3341	**3185**	3206.90
(20,20)	**798**	**798**	**798.00**	**467**	509	510.50	462	**432**	**432.00**
(20,50)	1861	**1843**	1854.85	**1313**	1453	1453.20	1051	**1002**	**1002.00**
(20,100)	3579	**3553**	**3553.00**	**2455**	2930	2947.95	2137	**1995**	2017.90
(20,200)	4152	**3858**	3899.30	**3831**	3850	3885.45	3077	**2843**	2881.40

Note: The bold values mean the best results

From Table VI, it can be seen the eEDA outperforms the GA in most instances. On 87.5% of the instances the best results of the eEDA are not worse than that of the GA, and on 83.3% instances the average results of the eEDA are not worse than that of the GA. In most situations, especially on the large scaled instances, the eEDA performs much better than the GA. The reason mainly lies in that the eEDA has good exploration ability and on the large scale instance the eEDA can locate the good searching space quickly. Moreover, the probability model and its sampling mechanism are helpful to find the feasible solutions. However, on several large-scaled instances with medium density, GA has better performance. So, further research is still needed for the eEDA to enhance its search ability for most cases, such as the hybridization with local search.

VI. CONCLUSIONS

In this paper, an effective EDA is proposed to solve the prec-UFPMSP with the makespan criterion. Some best solutions of the benchmarking instances [13] are updated by the proposed eEDA. The main contribution of this work is to propose a simple but effective probability model and the sampling mechanism to deal with the precedence constraints between jobs. The probability model utilizes the characteristics of the problem in the initialization procedure. In addition, the sampling mechanism is able to produce the feasible solutions effectively. Due to the simplicity, the eEDA can be used to initialize the feasible population or contribute to a part of the solver for other complex problems such as the RCPSP.

As for the future work, it is interesting to design a self-adaptive probability model according to the characteristic of the problem or to develop an adaptive way to utilize the merits of different models. In addition, the proposed could be generalized to the with more complexity like the distributed production systems.

REFERENCES

[1] T.C.E. Cheng, C.C.S. Sin, "A state-of-the-art review of parallel-machine scheduling research," European Journal of Operational Research, vol. 47(3), pp. 271-292, 1990.

[2] C. Szabo, T. Kroeger, "Evolving multi-objective strategies for task allocation of scientific workflows on public clouds," Evolutionary Computation (CEC), IEEE Congress on Brisbane, pp. 1-8, 2012.

[3] W.L. Pearn, S.H. Chung, M.H. Yang, "Minimizing the total machine workload for the wafer probing scheduling problem," IIE transactions, vol. 34(2), pp. 211-220, 2002.

[4] J. Bao, X. Hu, Y. Jin, "A genetic algorithm for minimizing makespan of block erection in shipbuilding," Journal of Manufacturing Technology Management, vol. 20(4), pp. 500-512, 2009.

[5] B. Gacias, C. Artigues, P. Lopez, "Parallel machine scheduling with precedence constraints and setup times," Computers & Operations Research, vol. 37(12), pp. 2141-2151, 2010.

[6] P. Larranaga, J.A. Lozano, Estimation of DistributionAlgorithms: A New Tool for Evolutionary Computation. Springer Science & Business Media, 2002.

[7] J. Ceberio, E. Irurozki, A. Mendiburu, et al, "A review on estimation of distribution algorithms in permutation-based combinatorial optimization problems," Progress in Artificial Intelligence, vol. 1(1), pp. 103-117, 2012.

[8] I. Aho, E. Mäkinen, "On a parallel machine scheduling problem with precedence constraints," Journal of Scheduling, vol. 9(5), pp. 493-495, 2006.

[9] A. Munier, M. Queyranne, A.S. Schulz, "Approximation bounds for a general class of precedence constrained parallel machine scheduling problems", Siam Journal on Computing, vol. 35(5), pp. 1241-1253, 2000.

[10] C. Chekuri, R. Motwani, "Precedence constrained scheduling to minimize sum of weighted completion times on a single machine," Discrete Applied Mathematics, vol. 98(1), pp. 29-38, 1999.

[11] C. Liu, S. Yang, "A heuristic serial schedule algorithm for unrelated parallel machine scheduling with precedence constraints," Journal of Software, vol. 6(6), pp. 1146-1153, 2011.

[12] J. Herrmann, J.M. Proth, N. Sauer, "Heuristics for unrelated machine scheduling with precedence constraints," European Journal of Operational Research, vol. 102(3), pp. 528-537, 1997.

[13] M.A. Hassan Abdel-Jabbar, I. Kacem, S. Martin, "Unrelated parallel machines with precedence constraints: application to cloud computing," Cloud Networking(CloudNet), IEEE 3rd International Conference on Luxembourg, pp. 438-442, 2014.

[14] J. Hurink, S. Knust, "List scheduling in a parallel machine environment with precedence constraints and setup times," Operations Research Letters, vol. 29(5), pp. 231-239, 2001.

[15] F.A. Omara, M.M. Arafa, "Genetic algorithms for task scheduling problem," Journal of Parallel and Distributed Computing, vol. 70(1), pp. 13-22, 2010.

[16] L. Wang, C. Fang, C.D. Mu, et al, "A Pareto-archived estimation-of-distribution algorithm for multiobjective resource-constrained project scheduling problem," IEEE Transactions on Engineering Management, vol. 60(3), pp. 617-626, 2013.

[17] M. Afzalirad, J. Rezaeian, "Design of high-performing hybrid meta-heuristics for unrelated parallel machine scheduling with machine eligibility and precedence constraints," Engineering Optimization, vol. 1, pp. 1-20, 2015.

[18] S. Baluja, "Population-based Incremental learning: a method for integrating genetic search based function optimization and competitive learning," Carnegie-Mellon Univ (Pittsburgh), Dept of Computer Science, pp. 1-20, 1994.

[19] D.C. Montgomery, Design and Analysis of Experiments. New York: Wiley, 1984.