An Effective DE-EDA for Permutation Flow-shop Scheduling Problem

Zuo-cheng Li
Institute of Industrial Engineering & Logistics Optimization
Northeastern University
Shenyang, China
zuocheng_li@163.com

Qingxin Guo
Institute of Industrial Engineering & Logistics Optimization
Northeastern University
Shenyang, China
guoqingxin@ise.neu.edu.cn

Lixin Tang
Institute of Industrial Engineering & Logistics Optimization
Northeastern University
Shenyang, China
lixintang@mail.neu.edu.cn

***Abstract*—Aiming at the permutation flow-shop scheduling problem (PFSSP) with makespan criterion, a combination algorithm based on differential evolution (DE) and estimation of distribution algorithm (EDA), namely DE-EDA, is proposed. Firstly, DE-EDA combines the probability-dependent macro information extracted by EDA and the individual-dependent micro information obtained by DE to execute the exploration, which is helpful in guiding the global search to explore promising solutions. Secondly, in order to make DE well suited to solve PFSSP, a convert rule named smallest-ranked-value (SRV) is designed to generate the discrete job permutations from the continuous values. Thirdly, a sequence-learning-based Bayes posterior probability is presented to estimate EDA's probability model and sample new solutions, so that the global information of promising search regions can be learned precisely. In addition, a simple but effective two-stage local search is embedded into DE-EDA to perform the exploitation, and thereafter numerous potential solution(s) with relative better fitness can be exploited in some narrow search regions. Finally, simulation experiments and comparisons based on 29 well-known benchmark instances demonstrate the effectiveness of the proposed DE-EDA.**

Keywords—differential evolution; estimation of distribution algorithm; probability model; permutation flow-shop scheduling problem; makespan

I. Introduction

With rapid and competitive global economy, production scheduling plays a crucial role in both industrial engineering and logistics optimization fields. This paper deals with the permutation flow-shop scheduling problem (PFSSP) with makespan criterion, which expresses a certain kind of flow shop with simplified technological constraints [1]. As a well-known problem in the area of scheduling, it has been proved that the PFSSP is NP-hard [2]. Owing to its widespread engineering backgrounds and theoretical values, PFSSP has become one of the hottest issues in the production scheduling fields.

Since Johnson's pioneering work [3], many methods have been developed for solving PFSSP. In all of these approaches, exact algorithms such as the mathematical programming [4], are only suitable for small-size cases. For this reason, constructive heuristic algorithms and evolutionary algorithms have been widely applied to solve PFSSP. In [5], a self-adaptive iterated local search algorithm was presented for solving PFSSP with total flow time criterion, which utilized a novel strategy to evaluate the neighborhoods around the local optimum and adjust the perturbation strength. Yahyaoui *et al.* [6] reported an iterated local search (ILS) with a certain kind of variable neighborhood descent (VND) hyper-heuristic, and the objectives of PFSSP to be optimized were the makespan and the total flow time. In [7], a differential evolution hybridized with a problem-dependent local search mechanism was proposed to deal with both single objective and multi-objective PFSSP. Xu *et al.* [8] aimed at the PFSSP under considering the total flow time objective, and an improved memetic algorithm based on dynamic neighborhood was developed. In their work, the best known Nawaz-Enscore-Ham (NEH) [9] heuristic was used to construct the dynamic neighborhood, and thereafter a diversity index strategy is developed to enhance the ability associated with overcoming the local optima. And meanwhile, the enhanced particle swarm optimization (PSO), such as PSO with variable neighborhood search (PSOVNS) [10], PSO-based memetic algorithm (PSOMA) [11], also performed well in obtaining PFSSP's approximately optimal solution. Recently, relevant attentions have been captured to extend the latest newborn evolutionary algorithm to solve PFSSP. Lin *et al.* [12] reported a hybrid evolutionary algorithm that is the hybrid backtracking search algorithm (HBSA), which was proposed for PFSSP to minimize the makespan.

As important branches of evolutionary algorithms, differential evolution (DE) [13, 14] and estimation of distribution algorithm (EDA) have been applied to solve a wide set of optimization problems successfully. The individual-dependent micro information of promising solutions can be obtained by DE's mutation and crossover operations, but the global information is somewhat insufficient to be considered. Fortunately, EDA provides a powerful tool to characterize the distribution of the promising search regions, and the probability-dependent macro information can be captured to guide the exploration. Thus, not only does the DE-EDA extract the macro information to estimate the probability distribution of the solution space, but it also considers the micro information to create new individuals. That is, combining the DE and EDA is helpful in balancing the micro and macro information of the search space. Based on such a reasonable idea, DE-and-EDA-based algorithms have been proposed to deal with some certain kinds of optimization problems. Sun *et al*. [15] firstly put forward a combination framework of DE and EDA, namely DE/EDA for addressing the continuous optimization problem. Zhou *et al*. [16] presented a combination of DE and EDA for continuous bi-objective optimization. Song and Tang [17] proposed a novel hybrid DE-EDA for dynamic optimization problem. Bai *et al*. [18] proposed a DE-EDA with feasibility rules for the nonlinear programming (NLP) and mixed integer nonlinear programming (MINLP) models in the area of engineering optimization. To our best knowledge, relevant works associated with applying DE-EDA for scheduling problem is quite scarce, and there is no published works on DE-EDA for PFSSP.

This paper proposes a DE-EDA for PFSSP with makespan criterion. To begin with, DE-EDA combines the probability-dependent macro information extracted by EDA and the individual-dependent micro information obtained by DE reasonably to execute the exploration. A smallest-ranked-value (SRV) is designed for converting the continuous values to job permutations to make DE suitable for discrete encoded PFSSP. Moreover, a special sequence-learning-based Bayes posterior probability is presented to serve as the probability model of DE-EDA, owing to which the global information of promising search regions can be learned precisely. In addition, a simple but effective two-stage local search is embedded into DE-EDA to perform the exploitation.

The remainder of this paper is organized as follows. In Section II, PFSSP with makespan criterion is briefly introduced and formulated. In Section III, DE-EDA is proposed and described in detail. The experiments and comparisons are provided in Section IV. Finally, we end this paper with some conclusions and future works in Section V.

II. Brife Introduction to PFSSP

In PFSSP with *n* jobs and *m* machines, each job should be processed in a given and defined sequence, every job should visit all machines only once and the job permutation is the same for each machine. Denote $\boldsymbol{\pi}=[\pi_1,\pi_2,...,\pi_n]$ the sequence or permutation of jobs to be processed, $p(\pi_i,k)$ the processing time of job π_i on machine k, $C(\pi_i,k)$ the completion time of job π_i on machine k. Without loss of generality, we assume that each job is processed according to the sequence $1,2,...,m$. Then, $C(\pi_i,k)$ can be calculated as follows [2]:

$$C(\pi_1,1)=p(\pi_1,1), \tag{1}$$

$$C(\pi_i,1)=C(\pi_{i-1},1)+p(\pi_i,1),\ i=2,...,n, \tag{2}$$

$$C(\pi_1,k)=C(\pi_1,k-1)+p(\pi_1,k),\ k=2,...,m, \tag{3}$$

$$C(\pi_i,k)=\max\{C(\pi_{i-1},k),C(\pi_i,k-1)\}+p(\pi_i,k),\quad i=2,...n,\ k=2,...,m. \tag{4}$$

Hence, the makespan can be defined as

$$C_{\max}(\pi)=C(\pi_n,m). \tag{5}$$

The PFSSP with the makespan criterion is to find a schedule in the set of all permutations $\boldsymbol{\Pi}$ as

$$\boldsymbol{\pi}^*=\arg\{C_{\max}(\boldsymbol{\pi})\}\to\min,\ \forall\boldsymbol{\pi}\in\boldsymbol{\Pi}. \tag{6}$$

III. DE-EDA for PFSSP

In this section, we will propose the procedure of DE-EDA after introducing the solution representation and SRV rule, DE with the first stage local search, EDA with sequence-learning mechanism, and the second stage local search.

A. Solution Representation and SRV Rule

Due to the distinct characters of the individuals in DE and EDA, a central premise should be considered that is to design personalized solution representations for them. In this paper, we utilize a job-based solution representation [1] for EDA, and a continuous representation to represent the individuals of DE. In order to make continuous individuals in DE suitable for discrete encoded PFSSP, we design a named SRV rule for converting the continuous values to job permutations. Let $\boldsymbol{X}_i=[x_{i,1},\cdots,x_{i,n}]$ denote the *i*th individual in DE's population, and $\boldsymbol{\pi}_i=[\pi_{i,1},\cdots,\pi_{i,n}]$ denote its corresponding job permutation. Then, the procedure of SRV is described as follows:

```
Step 1: Rank $\boldsymbol{X}_i$ by ascending order to obtain a new sequence
        $\tilde{\boldsymbol{X}}_i=[\tilde{x}_{i,1},...,\tilde{x}_{i,n}]$.
Step 2: Calculate the intermediate sequence $\boldsymbol{\psi}_i=[\psi_{i,1},...,\psi_{i,n}]$;
        For $l$ = 1 to $n$ do
          For $ll$ = 1 to $n$ do
            If $\boldsymbol{X}_{i,l}=\tilde{\boldsymbol{X}}_{i,ll}$ then
              $\psi_{il}=ll$;
              Break;
            End;
          End;
        End.
Step 3: Obtain final permutation $\boldsymbol{\pi}_i$;
        For $l$ =1 to $n$ do
          $\pi_{i,\psi_{i,l}}=l$;
        End.
```

Obviously, the conversion process of SRV is very simple to be addressed, which will be useful to convert the discrete job permutation to DE's continuous sequence.

B. DE with the First Stage Local Search (DE_FSLS)

In DE-EDA, the role of DE lies in obtaining the individual-dependent micro information to guide the global search to promising solution regions. In this paper, we adopt the mutation and crossover approaches presented in [7], and the so-called DE/rand-to-best/1/exp is used to perform the parallel search. And, in order to refine the solutions obtained by DE's mutation and crossover, an *Interchange*-based local search with first move strategy (i.e., the first stage local search) is embedded into DE. This means that the *Interchange*-based local search needs to be applied for each individual before executing DE's selection operation.

Denote $FirstStageLS(\boldsymbol{X}_i(gen))$ the procedure of first stage local search at generation *gen*, $Interchange(\boldsymbol{\pi}_i(gen),u,v)$ the interchange of the job at the *u*th dimension and the job at the *v*th dimension. Then, the procedure of $FirstStageLS(\boldsymbol{X}_i(gen))$ is given as follows:

```
Step 1: Set $l=0$.
Step 2: Convert individual $\boldsymbol{X}_i(gen)$ to a schedule $\boldsymbol{\pi}_i(gen)$ via the SRV rule.
Step 3: Set $\boldsymbol{\pi}_{i\_0}=\boldsymbol{\pi}_i(gen)$.
Step 4: Set $l=l+1$ and $\boldsymbol{\pi}_{i\_1}=\boldsymbol{\pi}_{i\_0}$; if $l>(n(n-1))/2$, then go to Step 7.
Step 5: Randomly select $u$ and $v$, $\boldsymbol{\pi}_{i\_1}=Interchange(\boldsymbol{\pi}_{i\_0},u,v)$, where $u\neq v$.
Step 6: If $f(\boldsymbol{\pi}_{i\_1})\geq f(\boldsymbol{\pi}_{i\_0})$, then go to Step 4.
Step 7: Output $\boldsymbol{\pi}_i(gen)=\boldsymbol{\pi}_{i\_1}$.
Step 8: Convert $\boldsymbol{\pi}_i(gen)$ to $\boldsymbol{X}_i(gen)$ according to the conversion process of SRV's.
```

Denote $\boldsymbol{pop}(gen)$ the DE's population with size *PS* at generation gen, $\boldsymbol{X}^{lbest}(gen)$ the local best individual of current population $\boldsymbol{pop}(gen)$, and $\boldsymbol{T}_i(gen)=[t_{i,1},...,t_{i,n}]$ the *i*th trial individual at generation gen. Then, the procedure of DE_FSLS is given as follows:

```
Step 1: Set $i=1$.
Step 2: Execute DE's mutation and crossover and create the ith trial individual $\boldsymbol{T}_i(gen)$.
Step 3: Perform DE's Selection;
    Step 3.1: Set $\boldsymbol{T}_{i\_0}=FirstStageLS(\boldsymbol{T}_i(gen))$;
    Step 3.2: If $f(\boldsymbol{T}_{i\_0})<f(\boldsymbol{T}_i(gen))$, then $\boldsymbol{T}_i(gen)=\boldsymbol{T}_{i\_0}$;
    Step 3.3: If $f(\boldsymbol{T}_i(gen))<f(\boldsymbol{X}_i(gen-1))$, then $\boldsymbol{X}_i(gen)=\boldsymbol{T}_i(gen)$; else $\boldsymbol{X}_i(gen)=\boldsymbol{X}_i(gen-1)$;
    Step 3.4: If $f(\boldsymbol{X}_i(gen))<f(\boldsymbol{X}^{lbest}(gen))$, then set $\boldsymbol{X}^{lbest}(gen)=\boldsymbol{X}_i(gen)$.
Step 4: Set $i=i+1$; if $i\leq PS$, then go to Step 2.
Step 5: Output $\boldsymbol{X}^{lbest}(gen)$, $\boldsymbol{pop}(gen)$, and their makespan.
```

Obviously, the initial population $\boldsymbol{pop}(0)$ is randomly generated from a uniform distribution $U[LB,UB]$, where *LB* and *UB* denotes the lower bound and upper bound of the DE's individuals respectively.

C. EDA with Sequence-learning Mechanism (EDA_SLM)

As mentioned, DE-EDA extracts the probability-dependent macro information according to the unique operations of EDA. In this subsection, we will propose the EDA_SLM in detail.

1) Probability Model, Initialization, and Updating Method

The probability model is a key element of EDA for executing global exploration. In this paper, the probability matrix is designed as [19]

$$\boldsymbol{P}_{matrix}(gen)=\begin{pmatrix} p_{11}(gen) & \dots & p_{1n}(gen) \\ \vdots & \ddots & \vdots \\ p_{n1}(gen) & \cdots & p_{nn}(gen) \end{pmatrix}_{n\times n}, \tag{7}$$

where $\sum_{w=1}^{n}p_{wj}(gen)=1$, and $p_{wj}(gen)$ is the probability of job w appearing in the *j*th position of $\boldsymbol{\pi}$ at generation *gen*. It is important to note that the probability matrix can be initialized (when *gen*=0) or restarted (when *gen*>0) by setting $p_{wj}(gen)=1/n$.

Denote $\boldsymbol{\pi}^{lbest}(gen)=[\pi_i^{lbest},\pi_2^{lbest},...,\pi_n^{lbest}]$ the local best individual (permutation) at generation *gen*, *LR* the learning rate. The matrix $\boldsymbol{P}_{matrix}(gen+1)$ is updated according to $\boldsymbol{\pi}^{lbest}(gen)$ by the following two steps:

```
Step 1: Set $x=\pi_j^{lbest}(gen)$ and $p_{xj}(gen+1)=p_{xj}(gen)+LR$ for $j=1,...,n$.
Step 2: Set $p_{wj}(gen+1)=p_{wj}(gen+1)/\sum_{y=1}^{n}p_{yj}(gen+1)$ for $w=1,...,n$, and $j=1,...,n$.
```

2) Sequence-learning Mechanism

Unlike other types of combinatorial optimization problems (such as assignment problems [20]), the objective values of PFSSP inherently depend on the sequence and dependency relationships among these jobs. Thus, to define and extract the interactions or dependencies of jobs is helpful in obtaining more valuable and precise global information of promising solutions. In this paper, the sequence-learning matrix is defined to record the sequence and dependency relationships.

$$\boldsymbol{T}_{matrix}(gen)=\begin{pmatrix} t_{11}(gen) & \dots & t_{1n}(gen) \\ \vdots & \ddots & \vdots \\ t_{n1}(gen) & \cdots & t_{nn}(gen) \end{pmatrix}_{n\times n}, \tag{8}$$

where the element $t_{wj}(gen)$ denotes the accumulation times of job *w* right before job *j*. Obviously, $\boldsymbol{T}_{matrix}(gen)$ can be initialized or restarted by setting $t_{wj}(gen=0)=0$ or $t_{wj}(gen>0)=0$. And, the $\boldsymbol{T}_{matrix}(gen)$ is updated according to $\boldsymbol{\pi}^{lbest}(gen)$ as follows:

```
Step 1: Set $j=1$.
Step 2: Set $x_1=\pi_j^{lbest}(gen)$, $x_2=\pi_{j+1}^{lbest}(gen)$.
Step 3: Set $t_{x_1x_2}(gen)=t_{x_1x_2}(gen)+1$ and $j=j+1$; if $j<n$, then go to Step 2.
Setp 4: Output $T_{matrix}(gen)$.
```

Then, we can obtain the two-dimensional joint distribution law of job *w* right before job *j* as

$$\boldsymbol{Q}_{matrix}(gen)=\begin{pmatrix} q_{11}(gen) & \dots & q_{1n}(gen) \\ \vdots & \ddots & \vdots \\ q_{n1}(gen) & \cdots & q_{nn}(gen) \end{pmatrix}_{n\times n}, \tag{9}$$

where

$$q_{wj}(gen)=t_{wj}(gen)/\sum_{x=1}^{n}\sum_{y=1}^{n}(t_{xy}(gen)/gen). \quad (10)$$

3) New Population Generation Method

In DE-EDA, we utilize a sequence-learning-based Bayes posterior probability to estimate the probability and then sample new solutions. Based on Bayes posterior probability, each random variable's probability value can be seen as the condition probability of its previous variable or variables. Denote r_{wj} the condition distribution law (CDL) of the *j*th dimension random variable, which satisfying the condition that the $(j-1)$th position of $\boldsymbol{\pi}$ has been filled by new generated job π_{j-1}. Then, r_{wj} can be calculated as follow:

$$r_{wj}(gen)=\frac{p_{wj}(gen)\cdot q_{w\pi_{j-1}}(gen)}{p_{\pi_{j-1}j}(gen)}, w=1,...,n, j=2,...,n, \quad (11)$$

where $r_{w1}=p_{w1}$, for $w=1,...,n$.

Based on above analysis, the new population of EDA can be generated as follows:

```
Step 1: Set $i=1$.
Step 2: Generate a new individual $\boldsymbol{\pi}_i(gen+1)$;
    For j=1 to n do
        For w=1 to n do
            Calculate $r_{wj}(gen)$;
        End;
        For w=1 to n do
            $r_{wj}(gen)=r_{wj}(gen)/\sum_{y=1}^{n}r_{yj}(gen)$;
        End;
        Randomly create a probability $r\sim[0,1)$;
        If $r\sim[0,r_{1j}(gen))$ then
            Set $CJ=1$;
        End;
        For w=1 to n do
            If $r\sim[\sum_{y=1}^{w}r_{yj}(gen),\sum_{y=1}^{w+1}r_{yj}(gen))$ then
                Set $CJ=w+1$;
            End;
        End;
        For l=j+2 to n do
            $p_{CJl}$=0;
        End;
        Set $\pi_{ij}(gen+1)=CJ$;
    End.
Step 3: Set $i=i+1$.
Step 4: If $i\le PS$, then go to Step 2.
```

4) Restarting Strategy

Let $\boldsymbol{RST}=\{1/sp,...,t/sp,...,sp-1/sp\}$ denote the restarting index and *genMax* denote the maximum generation. That is, if $gen=trunc((t/sp)\cdot genMax)$, then we restart $P_{matrix}(gen)$ and $T_{matrix}(gen)$ by utilizing the restarting method mentioned in subsection III-C-1 and III-C-2. Denote *TC* the training constant, and thereafter the procedure of restarting strategy is presented as follows:

```
Step 1: Restart $\boldsymbol{P}_{matrix}(gen)$ and $\boldsymbol{T}_{matrix}(gen)$ respectively.
Step 2: Training $P_{matrix}(gen)$;
    For i=1 to TC do
        Update $\boldsymbol{P}_{matrix}(gen)$ according to $\boldsymbol{\pi}^{lbest}(gen)$;
    End.
Step 3: Output $\boldsymbol{P}_{matrix}(gen)$ and $\boldsymbol{T}_{matrix}(gen)$.
```

D. The Second Stage Local Search

As mentioned in [7], an *Insert*-based local search (i.e., the second stage local search) is designed and introduced into DE-EDA to perform a wider and deeper local exploitation. Denote $Insert(\pi_i(gen,u,v))$ the function associated with inserting the job at the *v*th dimension to the *u*th dimension, where $u<v$. Denote $\boldsymbol{X}_{random}(gen)$ a randomly selected individual from current DE's population. Then, the procedure of *Insert*-based local search is given as follows:

```
Step 1: Set l=0.
Step 2: Set $\boldsymbol{\pi}_{i\_0}=\boldsymbol{\pi}^{lbest}(gen)$.
Step 3: Set l=l+1 and $\boldsymbol{\pi}_{i\_1}=\boldsymbol{\pi}_{i\_0}$; if $l>10\cdot n$, then go to Step 6.
Step 4: Randomly select u and v, where $u<v$;
        $\boldsymbol{\pi}_{i\_1}=Insert(\pi_{i\_0},u,v)$.
Step 5: If $f(\boldsymbol{\pi}_{i\_1})\ge f(\boldsymbol{\pi}_{i\_0})$, then go to Step 3.
Step 6: Output $\boldsymbol{\pi}^{lbest}(gen)=\boldsymbol{\pi}_{i\_1}$.
Step 7: Randomly select $\boldsymbol{X}_{random}(gen)$.
Step 8: Convert $\boldsymbol{\pi}^{lbest}(gen)$ to $\boldsymbol{X}^{lbest}(gen)$ according
        to $\boldsymbol{X}_{random}(gen)$ by using SRV's conversion process.
```

E. Procedure of DE-EDA

Denote $\boldsymbol{\pi}^{gbest}$ the global best individual, ***pop'***(*gen*) the EDA's population, Then, we present the procedure of DE-EDA as follows:

```
Step 1: Initialization;
    Step 1.1: Set $gen=0$;
    Step 1.2: Randomly generate DE's initial population
              $\boldsymbol{pop}(0)$ and obtain the schedules by using
              SRV in III-A; and calculate their makespan;
    Step 1.3: Obtain $\boldsymbol{\pi}^{lbest}(0)$ and $\boldsymbol{X}^{lbest}(0)$, and set $\boldsymbol{\pi}^{gbest}$
              $=\boldsymbol{\pi}^{lbest}(0)$;
    Step 1.4: Initialize $\boldsymbol{P}_{matrix}(0)$ and $\boldsymbol{T}_{matrix}(0)$ by using
              the initialization method in subsection III-C.
Step 2: Set $gen=gen+1$.
Step 3: Determine whether or not to execute the restarting
        strategy via the restarting strategy in III-C.
```

Step 4: Perform EDA;

Step 4.1: Update the probability matrix by using the update methods in III-C;

Step 4.2: Generate new population $\boldsymbol{pop'}(gen)$ by using the new population generation methods in III-C, and calculate their objective values;

Step 4.3: Update $\boldsymbol{\pi}^{lbest}(gen)$;

Step 4.4: Update $\boldsymbol{X}^{lbest}(gen)$ according to $\boldsymbol{\pi}^{lbest}(gen)$ and a randomly selected $\boldsymbol{X}_{random}(gen-1)$ by using SRV's conversion process.

Step 5: Perform DE_FSLS mentioned in III-B, and update $\boldsymbol{\pi}^{lbest}(gen)$ and $\boldsymbol{X}^{lbest}(gen)$.

Step 6: Perform the *Insert*-based local search by using the method in subsection III-D, and update $\boldsymbol{\pi}^{lbest}(gen)$ and $\boldsymbol{X}^{lbest}(gen)$.

Step 7: Update $\boldsymbol{\pi}^{gbest}$.

Step 8: If $gen < genMax$, then go to Step2.

Step 9: Output $\boldsymbol{\pi}^{gbest}$.

IV. Simulation Experiments and Comparisons

A. Experimental Setup

In this paper, 29 well-known benchmark instances are used to test the proposed DE-DEA. Among them, the first eight instances, namely Car1, Car2 through to Car8 designed by Carlier [21], and the other 21 instances are named Rec01, Rec03 through to Rec41 proposed by Reeves [22]. So far, these instances have been utilized as benchmarks for a wide set of algorithms by many researchers.

The proposed DE-EDA is coded with Visual studio 2008, C++ language, and executed on PC Intel Core-i5 3.30GHz processor with 4 GB memory. For each instance, DE-EDA is run 20 times independently, and the maximum generation *genMax* is set to 1000 serving as the terminal condition. In order to carry out relatively fair comparisons, three widespread evaluation metrics are referenced [12], as follows:

$$BRE = \frac{\min(\text{solutions}) - C^*}{C^*} \times 100\%, \tag{12}$$

$$ARE = \frac{\text{average}(\text{solutions}) - C^*}{C^*} \times 100\%, \tag{13}$$

$$WRE = \frac{\max(\text{solutions}) - C^*}{C^*} \times 100\%, \tag{14}$$

where C^* denotes the best known solution, and *BRE*, *ARE*, and *WRE* denotes the best relative error, average relative error, and worst relative error to C^* respectively.

B. Parameters Analysis and Setting

In DE-EDA, there exist eight parameters, i.e., the scaling factor *F*, the crossover parameter *CR*, the learning rate *LR*, population size *PS*, the training constant *TC*, the restarting constant *sp*, and the lower (upper) bound *LB* (*UB*). Obviously, *sp* is closely related to *genMax*, and *LB* (*UB*) can be set to well-known values in previous woks. Hence, we set *sp*=5, *LB*=0, and *UB*=4 without any discussion. For the first five key parameters (i.e., *F*, *CR*, *LR*, *PS*, and *TC*), we carry out the design-of-experiment (DOE) by using all the test instances mentioned in subsection IV-A. Different combinations of these parameters are listed in Table I.

TABLE I. Different Combinations of Parameters

Parameters	**Factor levels**			
	1	**2**	**3**	**4**
F	0.3	0.5	0.7	0.9
CR	0.05	0.1	0.15	0.2
LR	0.01	0.02	0.03	0.04
PS	30	50	70	100
TC	20	50	80	100

In the DOE, we choose the orthogonal array. That is, the total number of experiments is 16, the number of parameters is 5, and the number of factor levels is 4. The evaluation criteria is the $\overline{ARE}$ (i.e., the average value of *ARE*). The orthogonal array and the obtained are listed in Table II.

TABLE II. Orthogonal Array and the Obtained $\overline{ARE}$

No.	**Parameter combinations**					$\overline{ARE}$
	F	***CR***	***LR***	***PS***	***TC***	
1	1	1	1	1	1	1.083
2	1	2	2	2	2	1.129
3	1	3	3	3	3	1.196
4	1	4	4	4	4	1.243
5	2	1	2	3	4	1.146
6	2	2	1	4	3	1.158
7	2	3	4	1	2	1.370
8	2	4	3	2	1	1.220
9	3	1	3	4	2	1.219
10	3	2	4	3	1	1.306
11	3	3	1	2	4	1.223
12	3	4	2	1	3	1.328
13	4	1	4	2	3	1.391
14	4	2	3	1	4	1.447
15	4	3	2	4	1	1.284
16	4	4	1	3	2	1.325

From Table II, we can obtain the response values (*RV*) of each parameter in Table III, and the trend of each factor level can be illustrated by Fig. 1.

TABLE III. The *RV* of Each Parameter

Parameters	***RV***				
	F	***CR***	***LR***	***PS***	***TC***
1	1.163	1.210	1.197	1.307	1.223
2	1.224	1.260	1.222	1.241	1.261
3	1.269	1.268	1.271	1.243	1.268
4	1.362	1.279	1.328	1.226	1.265
range	0.199	0.069	0.131	0.081	0.045
grade	1	4	2	3	5

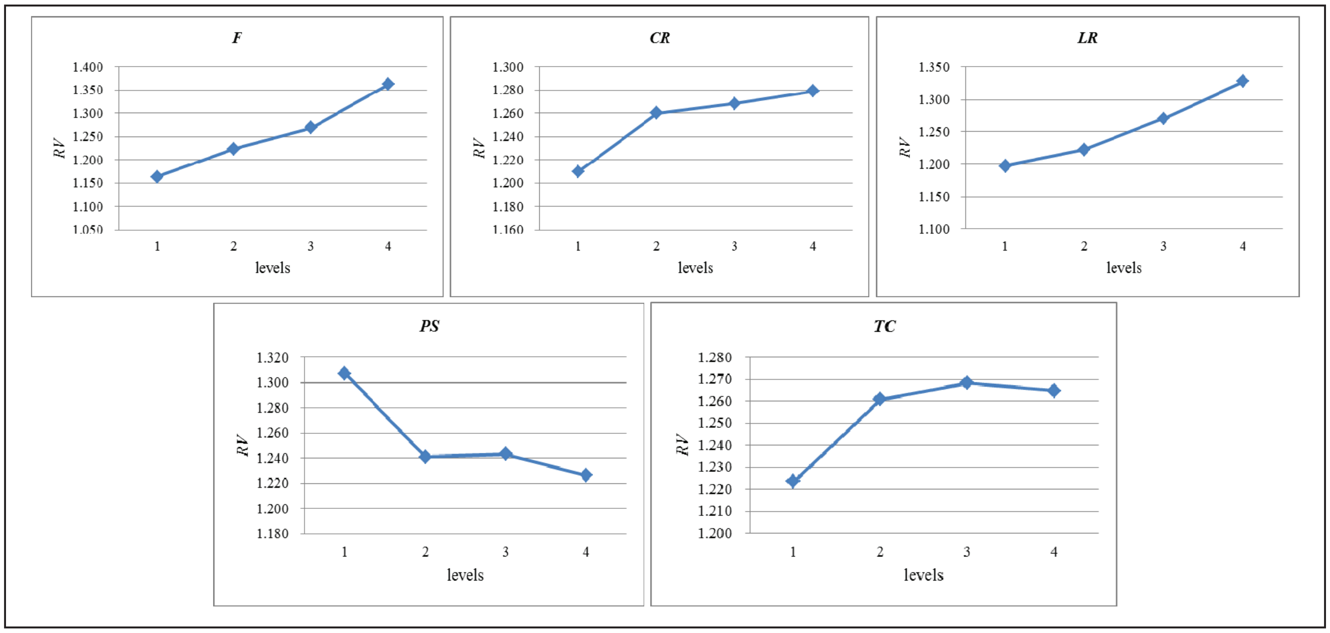

Fig.1 The trend of each factor level

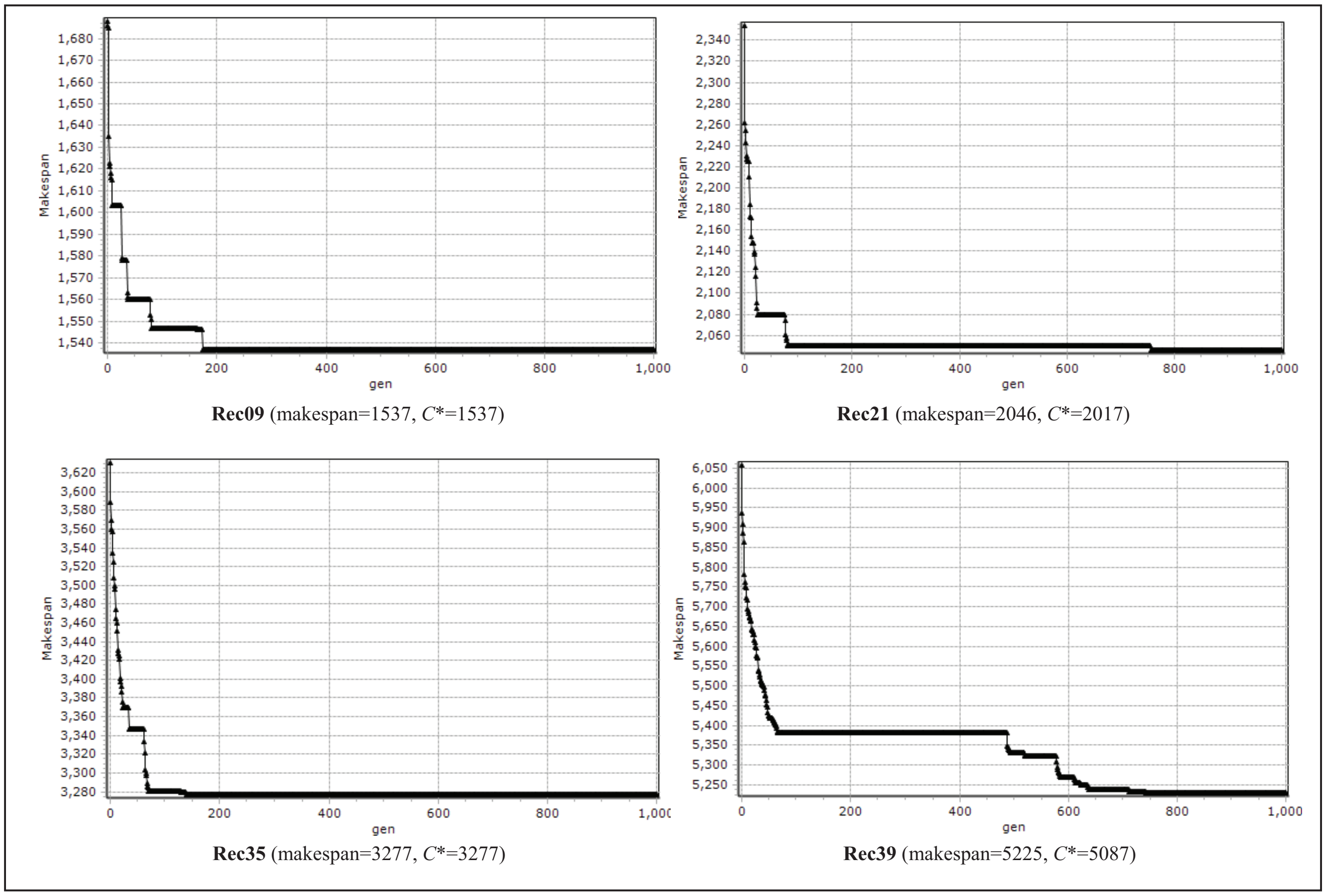

Fig. 2. The trend of makespan during the iteration (The horizontal and vertical axis represent the generation and makespan respectively)

Based on the above analysis, the conclusion can be drawn that DE-EDA is less sensitive to its parameters. And, the parameters of DE-EDA are set as follows: F=0.3, CR=0.05, LR=0.01, PS=100, and TC=20.

B. Comparisons of DE-EDA, PSOVNS, PSOMA, HGA, and NEH

In this subsection, we compare DE-EDA with other three state-of-the-art PFSSP solvers, i.e. PSOVNS [10], PSOMA [11], and HGA [2]. Besides, we carry out a comparison with NEH (reported in [7]), which is one of the best deterministic heuristics for PFSSP known so far. For NEH, only the relative error (*RE*) values to *C** are given. We accept the results reported by relevant papers, and do not code and execute these algorithms. The comparison results are listed in Table IV. Also, in order to analyze the convergence feature of DE-EDA, we make an observation on the trend of makespan during its iteration. The trends of makespan for 4 instances (i.e., Rec09, Rec21, Rec35, and Rec39) are shown in Fig. 2.

From Fig.2, we can draw a conclusion that the proposed DE-EDA has a faster convergence associated with obtaining high-quality solution (see Rec09 and Rec35). Meanwhile, it has a potential ability to perform a wider and deeper search to avoid falling into local optimum at the last phase of iteration (see Rec21 andRec39).

Seeing from Table IV, it is shown that the solution quality of DE-EDA is acceptable, and it is quite competitive among the five compared algorithms. The values of NoB (number of best) obtained by DE-EDA are much better than PSOVNS, PSOMA, HGA, and NEH with respect to *ARE*, *WRE*, and *RE* respectively. Thus, DE-EDA is an effective algorithm for the permutation flow-shop scheduling problem.

TABLE IV. COMPARISONS OF DE-EDA, PSOVNS, PSOMA, HGA, AND NEH

Ins.	n, m	C*	PSOVNS			PSOMA			HGA			NEH	DE-EDA		
			BRE	ARE	WRE	BRE	ARE	WRE	BRE	ARE	WRE	RE	BRE	ARE	WRE
Car1	11, 5	7038	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
Car2	13, 4	7166	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	2.93	0.00	0.00	0.00
Car3	12, 5	7312	0.00	0.42	1.19	0.00	0.00	0.00	0.00	0.00	0.00	1.79	0.00	0.00	0.00
Car4	14, 4	8003	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.39	0.00	0.00	0.00
Car5	10, 6	7720	0.00	0.04	0.39	0.00	0.02	0.38	0.00	0.00	0.00	4.24	0.00	0.00	0.00
Car6	8, 9	8505	0.00	0.08	0.76	0.00	0.11	0.76	0.00	0.04	0.76	3.62	0.00	0.00	0.00
Car7	7, 7	6590	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	6.34	0.00	0.00	0.00
Car8	8, 8	8366	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.09	0.00	0.00	0.00
Rec01	20, 5	1247	0.16	0.17	0.32	0.00	0.14	0.16	0.00	0.14	0.16	8.42	0.00	0.02	0.16
Rec03	20, 5	1109	0.00	0.16	0.18	0.00	0.19	0.72	0.00	0.09	0.18	6.58	0.00	0.00	0.00
Rec05	20, 5	1242	0.24	0.25	0.42	0.24	0.25	0.40	0.00	0.29	1.13	4.83	0.00	0.23	0.24
Rec07	20, 10	1566	0.70	1.10	1.41	0.00	0.99	1.15	0.00	0.69	1.15	5.36	0.00	0.00	0.00
Rec09	20, 10	1537	0.00	0.65	1.37	0.00	0.62	1.69	0.00	0.64	2.41	6.77	0.00	0.01	0.26
Rec11	20, 10	1431	0.07	1.15	2.66	0.00	0.13	0.98	0.00	1.10	2.59	8.25	0.00	0.00	0.00
Rec13	20, 15	1930	1.04	1.79	2.64	0.26	0.89	1.50	0.36	1.68	3.06	7.62	0.00	0.58	1.04
Rec15	20, 15	1950	0.77	1.49	2.26	0.05	0.63	1.08	0.56	1.12	2.00	4.92	0.15	0.57	0.82
Rec17	20, 15	1902	1.00	2.45	3.37	0.00	1.33	2.16	0.95	2.32	3.73	7.47	0.37	1.08	2.10
Rec19	30, 10	2093	1.53	2.10	2.53	0.43	1.31	2.10	0.62	1.32	2.25	6.64	0.91	1.43	1.86
Rec21	30, 10	2017	1.49	1.67	2.03	1.44	1.60	1.64	1.44	1.57	1.64	4.56	1.14	1.55	1.64
Rec23	30, 10	2011	1.34	2.11	2.88	0.60	1.31	2.04	0.40	0.87	1.69	10.00	0.80	1.20	1.74
Rec25	30, 15	2513	2.39	3.17	3.78	0.84	2.09	3.23	1.27	2.54	3.98	6.96	1.23	2.25	3.22
Rec27	30, 15	2373	1.73	2.46	3.20	1.35	1.61	2.40	1.10	1.83	4.00	8.51	1.22	1.69	2.19
Rec29	30, 15	2287	1.97	3.11	4.07	1.44	1.89	2.49	1.40	2.70	4.20	5.42	1.62	2.32	2.89
Rec31	50, 10	3045	2.59	3.23	4.24	1.51	2.25	2.69	0.43	1.34	2.50	10.28	1.84	2.71	3.65
Rec33	50, 10	3114	0.84	1.01	1.48	0.00	0.65	0.83	0.00	0.78	0.83	4.75	0.35	0.77	0.87
Rec35	50, 10	3277	0.00	0.04	0.09	0.00	0.00	0.00	0.00	0.00	0.00	5.01	0.00	0.00	0.00
Rec37	75, 20	4951	4.38	4.95	5.74	2.10	3.54	4.04	3.75	4.90	6.18	9.14	3.98	5.09	6.14
Rec39	75, 20	5087	2.85	3.37	3.95	1.55	2.43	2.83	2.20	2.79	4.48	8.65	2.32	3.37	4.44
Rec41	75, 20	4960	4.17	4.87	5.59	2.64	3.68	4.05	3.64	4.92	5.91	10.69	3.75	5.19	6.29
NoB			11	5	5	22	15	12	20	10	12	1	17	19	22

V. Conclusions and Future Works

This paper proposes a DE-EDA for PFSSP with makespan criterion. In DE-EDA, the probability-dependent macro information and the individual-dependent micro information are combined reasonably to execute the exploration. To make DE suitable for discrete encoded PFSSP, a smallest-ranked-value is designed for converting the continuous values to job permutations. Moreover, a special sequence-learning-based Bayes posterior probability is presented to serve as the probability model of DE-EDA. In addition, a simple but effective two-stage local search is embedded into DE-EDA to improve the exploitation. Simulation experiments and comparisons based on 29 benchmark instances verify that the proposed DE-EDA is a competitive and effective algorithm for PFSSP. Our future work is to develop effective hybrid algorithm based on DE and EDA for other production scheduling problem.

Acknowledgements

This research is supported by the 111 Project (B16009), the National Natural Science Fund of China for Major International (Regional) Joint Research Project (Grant No. 71520107004), the Fund for the National Natural Science Foundation of China(Grant No. 61374203)

References

[1] L. Wang, "Shop scheduling with genetic algorithms," Tsinghua University Press & Springer, 2003,Beijing, China.

[2] L. Wang and D. Z. Zheng, "An effective hybrid heuristic for flow shop scheduling," International Journal of Advanced Manufacturing Technology, vol. 21, pp. 38-44, 2003.

[3] S. M Johnson, "Optimal two- and three-stage production schedules with setup times included," Naval Research Logistics Quarterly, vol. 1, pp. 61-68, 1954.

[4] B. J. Lageweg, J. K. Lenstra, and A. H. G. Rinnooy Kan, "A general bounding scheme for the permutation flow-shop problem," Opertions Research, vol. 26, pp. 53-67, 1978.

[5] X. Dong, M. Nowak, P. Chen, and Y. Lin, "A self-adaptive iterated local search algorithm on the permutation flow shop scheduling problem," IEEE International Conference Informatics in Control, Automation and Robotics (ICINCO), pp. 378-384, 2014.

[6] H. Yahyaoui, S. Krichen, B. Derbel, and E. G. Talbi, "A hybrid ILS-VND based hyper-heuristic for permutation flowshop scheduling problem," Procedia Computer Science, vol. 60, pp. 632-641, 2015.

[7] B. Qian, L. Wang, R. Hu, W. L. Wang, D. X. Huang, and X. Wang, "A hybrid differential evolution method for permutation flow-shop scheduling," International Journal of Advanced Manufacturing Technology, vol. 38, pp. 757-777, 2008.

[8] J. Xu, Y. Yin, T. C. E. Cheng, C. C. Wu, and S. Gu, "An improved memetic algorithm based on a dynamic neighbourhood for the permutation flowshop scheduling problem," International Journal of Production Research , vol. 52, pp. 1188-1199, 2014.

[9] M. Nawaz, E. Jr. Enscore, and I. Ham, "A heuristic algorithm for the m -machine, n -job flow-shop sequencing problem," Omega, vol. 11, pp. 91-95, 1983.

[10] M. F. Tasgetiren, Y. C. Liang, M. Sevkli, and G. Gencyilmaz, "A particle swarm optimization algorithm for makespan and total flowtime minimization in the permutation flowshop sequencing problem," European Journal of Operational Research, vol. 177, pp. 1930–1947, 2007.

[11] B. Liu, L. Wand, and Y. H. Jin, "An effective PSO-based memetic algorithm for flow shop scheduling," IEEE Transactions on Systems Man & Cybernetics, vol. 37, pp. 18-27, 2007.

[12] Q. Lin, L. Gao, X. Li, and C. Zhang, "A hybrid backtracking search algorithm for permutation flow-shop scheduling problem," Computers & Industrial Engineering, vol. 85, pp. 437-446, 2015.

[13] L. X. Tang, Y. Dong, and J. Liu, "Differential Evolution With an Individual-Dependent Mechanism," IEEE Transactions on Evolutionary Computation, vol. 19, pp. 560-574, 2015.

[14] L. X. Tang, Y. Zhao, and J. Liu, "An Improved Differential Evolution Algorithm for Practical Dynamic Scheduling in Steelmaking-Continuous Casting Production," IEEE Transactions on Evolutionary Computation, vol. 18, pp. 209-225, 2014.

[15] J. Y. Sun, Q. F. Zhang, and E. P. K. Tsang, "DE/EDA: A new evolutionary algorithm for global optimization," Information Sciences, vol. 169, pp. 249-262, 2005.

[16] A. Zhou, Q. F. Zhang, Y. C. Jin, and B. Sendhoff, "Combination of EDA and DE for continuous biobjective optimization," IEEE World Congress on Computational Intelligence, pp. 1447-1454, 2008.

[17] X. M. Song and L. X. Tang, "A novel hybrid differential evolution-estimation of distribution algorithm for dynamic optimization problem," IEEE Congress on Evolutionary Computation, pp. 1710-1717, 2013.

[18] L. Bai, J. Y. Wang, Y. H. Jiang, and D.X. Huang, "Improved hybrid differential evolution-estimation of distribution algorithm with feasibility rules for NLP/MINLP engineering optimization problems," Chinese Journal of Chemical Engineering, vol. 20, pp. 1074–1080, 2012.

[19] Z. C. Li, B. Qian, R. Hu, C. S. Zhang, and K. Li, "A self-adaptive hybrid population-based incremental learning algorithm for M-machine reentrant permutation flow-shop scheduling," Lecture Notes in Computer Science, vol. 7995, pp. 8–20, 2013.

[20] Q. F. Zhang, J. Y. Sun, E. P. K. Tsang, and J. Ford, "Estimation of distribution algorithm with 2-opt local search for the quadratic assignment problem," Towards a New Evolutionary Computation, pp. 281-292, 2006:.

[21] J. Carlier, "Ordonnancements a contraintes disjonctives," Opertions Research, vol. 12, pp. 333–351, 1978.

[22] C. R. Reeves, "A genetic algorithm for flowshop sequencing," Computers & Operations Research, vol. 22, pp. 5-13, 1995.