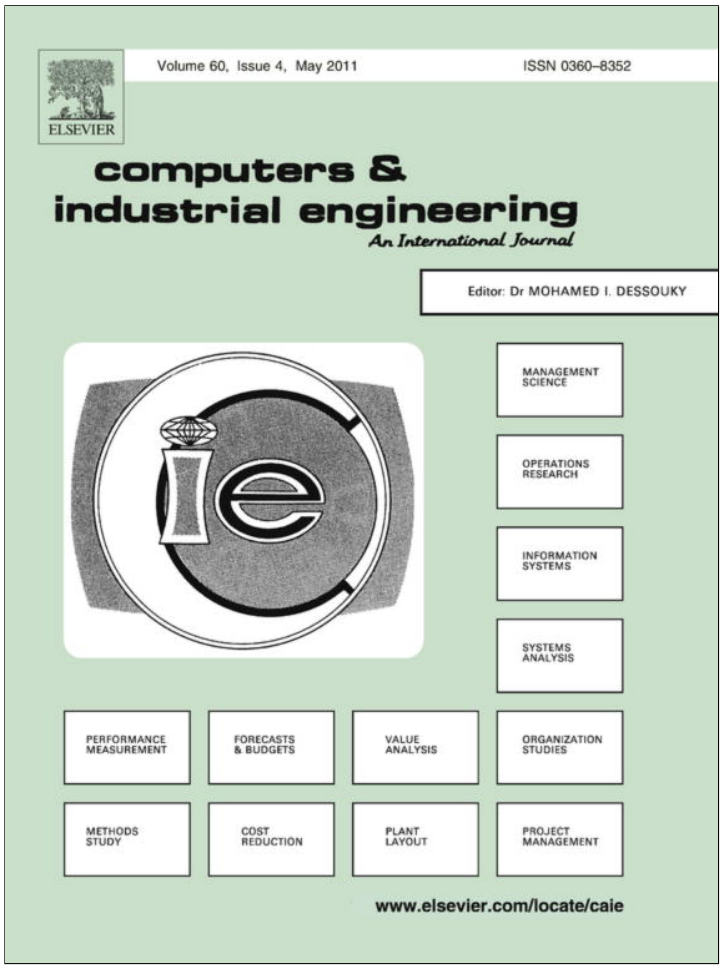

Computers & Industrial Engineering 60 (2011) 706–718

Contents lists available at ScienceDirect

Computers & Industrial Engineering

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

Estimation of distribution algorithm for permutation flow shops with total flowtime minimization ☆

Yi Zhang [a,b], Xiaoping Li [a,b,*]

[a] School of Computer Science & Engineering, Southeast University, 211189 Nanjing, PR China
[b] Key Laboratory of Computer Network and Information Integration (Southeast University), Ministry of Education, 211189 Nanjing, PR China

ARTICLE INFO

Article history:
Received 16 February 2010
Received in revised form 30 November 2010
Accepted 12 January 2011
Available online 18 January 2011

Keywords:
Estimation of distribution algorithm
Permutation flow shops
Total flowtime
Meta-heuristic

ABSTRACT

In this paper, an Estimation of Distribution Algorithm (EDA) is proposed for permutation flow shops to minimize total flowtime. Longest Common Subsequence (LCS) is incorporated into the probability distribution model to mine good "genes". Different from common EDAs, each offspring individual is produced from a seed, which is selected from the population by the roulette method. The LCS between the seed individual and the best solution found so far is regarded as good "genes", which are inherited by offspring with a probability less than 100% to guarantee the population diversity. An effective Variable Neighborhood Search (VNS) is integrated into the proposed EDA to further improve the performance. Experimental results show that the inheritance of good "genes" obtained by LCS can improve the performance of the proposed EDA. The proposed hybrid EDA outperforms other existing algorithms for the considered problem in the literature. Furthermore, the proposed hybrid EDA improved 42 out of 90 current best solutions for Taillard benchmark instances.

1. Introduction

The permutation flow shops (also called permutation flow shop scheduling problem, PFSP for short) has attracted the attentions of many researchers, since Johnson (1954) first introduced it. The target is to discover a permutation of jobs, which are processed over machines with the same machine sequence, in order to optimize one or more performance measures. It is well known that total flowtime minimization leads to stable or uniform utilization of resources, rapid turn-around of jobs and minimization of in-process inventory (French, 1982; Rajendran, 1994).

The PFSP can be denoted as $F|prmu|\sum C_j$ which was proved to be NP-hard by Garey, Johnson, and Sethi (1976). Naturally, a practical method is to find a solution as good as possible in acceptable computation time by heuristics rather than to discover an optimal solution. Heuristics can be divided into two classes, constructive and composite. Famous constructive heuristics are BES(LR) (Liu & Reeves, 2001), WY (Woo & Yim, 1998), RZ (Rajendran & Ziegler, 1997) and FL (Framinan & Leisten, 2003), in which RZ is always used as an insertion-based local search method. IH1–IH7 (Allahverdi & Aldowaisan, 2002), IH7-FL (Framinan & Leisten, 2003), FLR1 and FLR2 (Framinan, Leisten, & Ruiz, 2005), ICH1–ICH3 (Li, Wang, & Wu, 2009) are all good composites. As well, meta-heuristics are applied for the considered problem. Vempati, Chen, and Bulllngton (1993) first used a Genetic Algorithm (GA) for the considered problem. Zhang, Li, and Wang (2009) proposed a hybrid GA (HGA), in which artificial chromosome is constructed to mine good "genes" implicated in the population. Experimental results showed that this method helps HGA to obtain high performance but much computation time spent. Tseng and Lin (2009) presented a hybrid GA (named HGAYT in this paper), in which two local search methods response for a small and a large neighborhood, respectively. Their experimental results showed that HGAYT is very effective, and discovered many best solutions for the Taillard benchmark instances (1993). Rajendran and Ziegler (2004, 2005) presented four Ant-colony Algorithms (ACOs), including the famous PACO and M-MMAS. As well as based on pheromone trail intensities, offspring individuals are produced from a seed, i.e. the best solution found so far. This method benefits the four ACOs to be effective. Tasgetiren, Liang, Sevkli, and Gencyilmaz (2007) Jarboui, Ibrahim, Siarry, and Rebai (2008) investigated a Particle Swarm Optimization algorithm (PSO), called PSOvns and HCPSO respectively, which found many best solutions for the first 90 Taillard benchmark instances. Dong, Huang, and Chen (2009) constructed an Iterated Local Search algorithm (ILS). ILS starts with an initial solution, and then the local search is conducted until no improvement obtained. A perturbation operator is executed to make it jump out of the local optimal. Such a procedure is repeated until the termination criterion is satisfied. ILS is simple and effective, with the performance mostly depending on the effectiveness of the local search. Pan, Tasgetiren, and Liang (2008) proposed a hybrid Discrete Differential Evolutionary algorithm (DDERLS), which is integrated with an effective local search method. Jarboui, Eddaly, and Siarry (2009)

☆ This manuscript was processed by Area Editor T.C. Edwin Cheng.

* Corresponding author. Tel.: +86 25 52090916.
E-mail address: xpli@seu.edu.cn (X. Li).

0360-8352/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cie.2011.01.005

proposed an EDA (named JEDA for short in this paper), in which the probability distribution model was constructed based on the relative orders of jobs and similar blocks of jobs. As well, JHEDA was constructed by combining a Variable Neighborhood Search (JVNS for short) with JEDA. As reported in the literature, relationships among the above meta-heuristics are summarized as follows: (1) PSOvns outperforms PACO and M-MMAS; (2) HGA, ILS, HCPSO, HGAYT, DDERLS and JHEDA are all better than PSOvns; (3) DDERLS and JHEDA are better than HCPSO; (4) JHEDA outperforms HGA.

In fact, EDAs were proposed based on an estimation of distribution recently. EDA is population-based, similar to GA but without crossover and mutation operators. Instead, EDA explicitly extracts the global statistical information from the elite set selected from the parent population. A probability distribution model is constructed to sample new solutions and replace partial or all of the parent population. The well-known EDAs are population-based incremental learning (PBIL) (Baluja, 1994), univariate marginal distribution algorithm (UMDA) (Mühlenbein, 1998), mutual information maximization for input clustering (MIMIC) (De Bonet & Viola, 1997), combining optimizers with mutual information trees (COMIT) (Baluja & Davies, 1998), Bayesian optimization algorithm (BOA) (Pelikan, Goldberg, & Cantu, 1999), Bayesian evolutionary algorithm (BEA) (Zhang, 1999), iterated density estimation evolutionary algorithm (IDEA) (Bosman & Thierens, 2001), and global search based on reinforcement learning agents (GSBRL) (Miagkikh & Punch, 1999). Considering the location information of solutions found so far, Zhang, Sun, and Tsang (2005) proposed an EDA called EA/G, in which a new operator (Guided Mutation, GM for short) was constructed to generate offspring individuals. In GM, all elements of the best solution found so far can be inherited by the offspring individuals probabilistically. By this method, EA/G outperforms the famous algorithm MIMIC. Theoretically, the convergence of PBIL, UMDA and FDA was proved by Hohfeld and Rudolph (1997), Mühlenbein (1998), Zhang and Mühlenbein (2004), respectively. As well, EDAs have been applied in many areas, such as pattern recognition (Inza, Larranaga, & Etxeberria, 2000; Cesar, Bengoetxea, Bloch, & Larrangaga, 2005), software testing (Sagarna & Lozano, 2005), cancer classification (Blanco, Larranaga, & Inza, 2004), and operational research (Zhang, Sun, Tsang, & Ford, 2004).

In this paper, we propose an EDA (PEDA for simplicity) for the considered problem. In PEDA, the Longest Common Subsequence (LCS), which discovers the common information between two given sequences, is incorporated into the probability distribution model in order to mine good "genes". This idea is inspired by the fact that good "genes" mining can help GAs to obtain high performance, such as GAs given by Ruiz, Maroto, and Alcaraz (2006) and Zhang et al. (2009). Different from common EDAs, each offspring individual in PEDA is produced from a seed, selected from the current population by the roulette method. The LCS between the seed individual and the best solution found so far is considered as good "genes". However, we found that good "genes" inherited completely could deteriorate the population diversity after testing the GAs given by Ruiz et al. (2006) and Zhang et al. (2009). Though the former is efficient for PFSP with makespan minimization originally, we transformed it for the considered problem in the testing. Therefore, good "genes" should be inherited by offspring with a probability λ ($\lambda \in (0,1)$) in order to guarantee the population diversity. In addition, a new VNS is presented based on the JVNS (Jarboui et al., 2009) and employed as the local search to further improve the performance. Experimental results show that the LCS helps PEDA to outperform JEDA. The hybrid PEDA (PHEDA for short) is better than all of the existing algorithms for the considered problem in the literature, and can provide some new upper bounds of the benchmark instances given by Taillard (1993).

The rest of this paper is organized as follows: Section 2 gives a brief description of PFSP with total flowtime minimization. PEDA along with VNS are introduced in Section 3. A full performance evaluation and comparison is shown in Section 4, followed by conclusions in Section 5.

2. Description of $F|prmu|\sum C_j$

PFSP is an important combinational optimization problem with characteristics described below:

- n jobs are processed on m machines;
- each job has m operations processed on m machines $M_1,\ldots,M_m$ sequentially with the same order;
- each operation has predetermined processing time;
- each machine can process one operation exclusively at a time;
- preemption is not allowed;

Let $\Omega = \{J_1,\ldots,J_n\}$ be the job set and π be a permutation of n jobs, which can be denoted as $(\pi_{[1]},\ldots,\pi_{[n]})$ in which $\pi_{[i]} \in \Omega$ is the ith ($i = 1,\ldots,n$) job in π. To represent the start of a schedule, a dummy job $\pi_{[0]}$ is introduced and added to the first slot in π with zero processing times, i.e. π can also be represented as $(\pi_{[0]},\pi_{[1]},\ldots,\pi_{[n]})$. Let $C_{i,\pi_{[k]}}$ be the completion time of job $\pi_{[k]}$ on machine i, and $C_{i,\pi_{[0]}} = 0$. $t_{i,j}$ denotes the processing time of job j ($j = 0,1,\ldots,n$) on machine i ($i = 1,2,\ldots,m$). Then $C_{i,\pi_{[k]}}$ can be calculated as follows:

$$C_{i,\pi_{[k]}} = \begin{cases} C_{1,\pi_{[k-1]}} + t_{1,\pi_{[k]}}, & i = 1, \quad k = 1,\cdots,n \\ max\{C_{i-1,\pi_{[k]}}, C_{i,\pi_{[k-1]}}\} + t_{i,\pi_{[k]}}, & i = 2,\cdots,m, \quad k = 1,\cdots,n \end{cases} \tag{1}$$

The total flowtime $F(\pi)$ can be calculated by $F(\pi) = \sum_{k=1}^{n} C_{m,\pi_{[k]}}$.

3. Proposed estimation of distribution algorithm

Generally, EDAs contain the following steps.

1. **Initialization**: initial the population and parameters.
2. **Probability model construction**: select part of the population to generate the elite set, from which the probability model is constructed.
3. **Offspring production**: sample from the probability model to produce offspring.
4. **Local search (optional)**: act the local search on offspring individuals.
5. **Replacement**: replace part/all of the individuals in the population by the offspring.
6. **Termination criterion check**: if the termination criterion is not met, go to step 2.

Based on the above framework, the proposed hybrid EDA is described in details below.

3.1. Representation, population initialization and selection

Permutations of jobs are simply used to encode individuals, which is frequently adopted in the literature. In order to guarantee the diversity of the initial population (denoted as P_c), all the individuals are generated randomly. Such a method is also adopted by Tseng and Lin (2009), and Jarboui et al. (2009). The best $\alpha \times 100\%$ ($\alpha \in (0,1]$) of the population are selected to construct the elite set (represented as Π_e), i.e. $\lfloor \alpha |P_c| \rfloor = |\Pi_e|$.

3.2. Probability model

3.2.1. Global statistical information extraction

The probability distribution model is built based on the global statistic information which is extracted from the elite set. In PEDA,

the "voting" procedure, used by Zhang et al. (2009) & Chang et al. (2007) as well, is adopted to extract the global statistical information. As details about the "voting" procedure were described by Zhang et al. (2009) & Chang et al. (2007), respectively, we just summarize some key steps here. For $\pi \in \Pi_e$, the number of each job at each position is counted and recorded in a $(n+1)\times(n+1)$ "dominance matrix" M, in which $M_{[i][0]}=0$ $(i=0,1,\ldots,n)$ and $M_{[0][j]}=0$ $(j=1,2,\ldots,n)$, and $M_{[i][j]}$ denotes the number of job i positioning at slot j in Π_e. Additionally, each $\pi \in \Pi_e$ is given a weight according to its objective function (i.e. total flowtime) so that better individuals contribute more greatly to the "dominance matrix". Let π_w $(\pi_w \in \Pi_e)$ be the individual with the worst objective function (i.e., largest total flowtime) in the current elite set, π^* the best solution found so far, the weight of any $\pi \in \Pi_e$ can be calculated as follows:

$$w_\pi = \frac{F(\pi_w)-F(\pi)}{F(\pi^*)} \tag{2}$$

According to Eq. (2), it is obvious that individuals with lower total flowtime are given larger weights, and the "voting" procedure with the complexity $O(n|\Pi_e|)$ is described below:

1. *VOTING*(Π_e) :
2. For each $\pi \in \Pi_e$
3. For each position j in π
4. If ($\pi_{[j]}$ is job i) (3)
5. $M_{[i][j]} = M_{[i][j]} + w_\pi$
6. Return M.
7. End *VOTING*

3.2.2. Longest Common Subsequence (LCS)

Define $\pi \subseteq \pi_A$ to represent that π is a subsequence of π_A. The common subsequence can be defined below:

Definition 1 (*Cormen, Leiserson, Rivest, & Stein, 2006*). For two given sequence π_A and π_B, π is a sequence, if $\pi \subseteq \pi_A$ and $\pi \subseteq \pi_B$, π is a common sequence between π_A and π_B.

Definition 2 (*Cormen et al., 2006*). For two given sequence π_A and π_B, Π_{AB} is the set of all common sequence of π_A and π_B, the longest one in the Π_{AB} is the longest common subsequence.

Let $\pi_{A,[i]}$ denote the ith element in the π_A, and π_A^i be the sequence including the first i number of elements, i.e. $\pi_A^i = (\pi_{A,[1]}, \pi_{A,[2]}, \ldots, \pi_{A,[i]})$, the following property describes the LCS structure (Cormen et al., 2006).

Property 1 (Cormen et al., 2006). *For two given sequence* $\pi_A = (\pi_{A,[1]}, \pi_{A,[2]}, \ldots, \pi_{A,[p]})$ *and* $\pi_B = (\pi_{B,[1]}, \pi_{B,[2]}, \ldots, \pi_{B,[q]})$, $\pi_C = (\pi_{C,[1]}, \pi_{C,[2]}, \ldots, \pi_{C,[k]})$ *is the LCS between* π_A *and* π_B,

(1) If $\pi_{A,[p]} = \pi_{B,[q]}$, then $\pi_{A,[p]} = \pi_{B,[q]} = \pi_{C,[k]}$ and π_C^{k-1} is the LCS between π_A^{p-1} and π_B^{q-1}.
(2) If $\pi_{A,[p]} \neq \pi_{B,[q]}$ and $\pi_{A,[p]} \neq \pi_{C,[k]}$, then π_C is the LCS between π_A^{p-1} and π_B.
(3) If $\pi_{A,[p]} \neq \pi_{B,[q]}$ and $\pi_{B,[q]} \neq \pi_{C,[k]}$, then π_C is the LCS between π_A and π_B^{q-1}.

The proof was given by Cormen et al. (2006). Property 1 shows that how to generate the LCS between two given sequence π_A and π_B depends on three sub-problems: (1) generating the LCS between π_A^{p-1} and π_B^{q-1}, (2) generating the LCS between π_A^{p-1} and π_B, and (3) generating the LCS between π_A and π_B^{q-1}. Furthermore, the recursion procedure implies that sub-problems (2) and (3) depend on the sub-problems of sub-problem (1). This means the LCS generating procedure has the overlapped sub-problem characteristic. So, Dynamic Programming (DP) can be used to generate the LCS between two given sequences.

Define a $(p+1)\times(q+1)$ matrix c, in which $c_{[i][j]}$ represents the length of the LCS between π_A^i and π_B^j. And, $c_{[p][q]}$ is the length of the LCS between π_A and π_B. If $i=0$ or $j=0$, the length of the LCS is 0. Based on Property 1, the length of the LCS can be calculated by the following recursive formulation: (seen in Cormen et al. (2006))

$$c_{[i][j]} = \begin{cases} 0 & \text{if } i=0 \text{ OR } j=0 \\ c_{[i-1][j-1]}+1 & \text{if } \pi_{A,[i]} = \pi_{B,[j]} \text{ AND } i,j>0 \\ \max\{c_{[i-1][j]}, c_{[i][j-1]}\} & \text{if } \pi_{A,[i]} \neq \pi_{B,[j]} \text{ AND } i,j>0 \end{cases} \tag{4}$$

Based on Formulation (4), the matrix c can be obtained by the following procedure (Cormen et al., 2006).

1. *LENGTH_LCS* (π_A,π_B)
2. $p \leftarrow |\pi_A|, q \leftarrow |\pi_B|$.
3. $c_{[i][0]} \leftarrow 0 (i=1,2,\cdots,p)$, $c_{[0][j]} \leftarrow 0 (j=0,1,\cdots,q)$.
4. For $i \leftarrow 1$ to p
5. For $j \leftarrow 1$ to q
6. If $(\pi_{A,[i]} = \pi_{B,[j]})$ then $c_{[i][j]} \leftarrow c_{[i-1][j-1]}+1$, $b_{[i][j]} \leftarrow$ "$\nwarrow$".
7. Else if $(c_{[i-1][j]} \geqslant c_{[i][j-1]})$ then $c_{[i][j]} \leftarrow c_{[i-1][j]}$, $b_{[i][j]} \leftarrow$ "$\uparrow$".
8. Else $c_{[i][j]} \leftarrow c_{[i][j-1]}$, $b_{[i][j]} \leftarrow$ "$\leftarrow$".
9. Return c and b.
10. END *LENGTH_LCS*

It is obvious that the complexity of this procedure is $O(pq)$. Besides the matrix c, a $(p+1)\times(q+1)$ matrix b is also constructed, which records the "path" to generate the LCS. The generation of LCS starts from $b_{[p][q]}$ and follows the arrow symbols (i.e. "$\nwarrow$", "$\uparrow$" or "$\leftarrow$") $b_{[i][j]}$ records. $b_{[i][j]}$ = "$\nwarrow$" implying $\pi_{A,[i]} = \pi_{B,[j]}$ is one element of the LCS. The following recursive procedure (Cormen et al., 2006) is adopted to generate the LCS when matrixes b and c have already been obtained by the *LENGTH_LCS* procedure.

1. *FLCS_OUTPUT* $(b,\pi_A,i\ j,\pi_C)$ /*$i = 0,1,\cdots,p$ and $j = 0,1,\cdots,q$*/
2. If ($i=0$ or $j=0$) then return.
3. If ($b_{[i][j]}$ = "$\nwarrow$") then *FLCS_OUTPUT* $(b,\pi_A, i-1, j-1, \pi_C)$, add $\pi_{A,[i]}$ into π_C.
4. If ($b_{[i][j]}$ = "$\uparrow$") then *FLCS_OUTPUT* $(b, \pi_A, i-1, j, \pi_C)$.
5. If ($b_{[i][j]}$ = "$\leftarrow$") then *FLCS_OUTPUT*$(b, \pi_A, i, j-1, \pi_C)$.
6. END *FLCS_OUTPUT*

In the procedure, π_C is an input sequence with the value of "Null". When the procedure finishes, π_C is the LCS. Since either i or j decreases in each recursion, the complexity of *FLCS_OUTPUT* is $O(p+q)$. The following complete procedure generates the LCS between two given sequences π_A and π_B.

1. *FLCS* (π_A, π_B)
2. Use *LENGTH_LCS* (π_A, π_B) to generate the two matrixes b and c.
3. $\pi_C \leftarrow$ Null.
4. Perform *FLCS_OUTPUT* (b, π_A, p, q, π_C) to construct π_C.
5. Return π_C.
6. END *FLCS*

It is obvious that the complexity of *FLCS* is $O(pq)$. For the considered problem, each sequence is a job permutation with the length n. Therefore, the complexity of *FLCS* becomes $O(n^2)$. An example is used to show the details of *FLCS*. In this example, $\pi_A = (1,2,3,4,5,6,7)$ and $\pi_B = (2,5,3,7,4,6,1)$. Fig. 1 shows the two matrixes b and c obtained by the procedure *LENGTH_LCS*.

In Fig. 1, the cell at the position (i,j) (i.e. the ith row and the jth column) records the $c_{[i][j]}$ and the arrow symbol in $b_{[i][j]}$. The length

π_A \ π_B	**2**	5	**3**	7	**4**	**6**	1
π_A	0	0	0	0	0	0	0	0
1	0	↑ 0	↑ 0	↑ 0	↑ 0	↑ 0	↖ 1
2	0	↖ 1	← 1	← 1	← 1	← 1	↖ 1
3	0	↑ 1	↑ 1	↖ 2	← 2	← 2	← 2	← 2
4	0	↑ 1	↑ 1	↑ 2	↑ 2	↖ 3	← 3	← 3
5	0	↑ 1	↖ 2	↑ 2	↑ 2	↑ 3	↑ 3	↑ 3
6	0	↑ 1	↑ 2	↑ 2	↑ 2	↑ 3	↖ 4	← 4
7	0	↑ 1	↑ 2	↑ 2	↖ 3	↑ 3	↑ 4	↑ 4

Fig. 1. The matrixes b and c obtained by *LENGTH_LCS*.

of the LCS between π_A and π_B is 4, which is recorded in the cell at (7,7). The generation of LCS starts from the cell at (7,7) as well, and follows the arrow symbol recorded in cells until reaching the cell at (1,0). As a result, a path (highlighted in Fig. 1) is constructed. In this path, the cells containing the arrow symbol "↖" correspond to the elements of LCS. So, the LCS (2,3,4,6) can be obtained, which is shown in bold in Fig. 1.

3.3. Offspring production

Based on the probability distribution model, each offspring individual in PEDA is produced from a seed. The $\beta \times 100\%$ ($\beta \in (0,1]$) of the current population are selected by the roulette method to construct a seed set $\Pi_s(|\Pi_s| = \lfloor \beta |P_c| \rfloor)$, without a replication. The LCS between each $\pi_k \in \Pi_s (k = 1,2,\ldots,|\Pi_s|)$ and the best solution found so far π^* is regarded as good "genes", and denoted as π_k^L. By experiments, we find that good "genes" completely inherited by offspring always deteriorates the population diversity, while testing both the GAs given by Ruiz et al. (2006) & Zhang et al. (2009). The former, originally for the PFSP with makespan minimization, was transformed to the considered problem in our testing. The replacement of the GA (Ruiz et al., 2006) is also used in the proposal. Results showed that all the individuals in the final population were similar. And, when the second condition (see Section 3.4) was removed from the replacement, more than 90% individuals in the final population are identical to the final solution. For the latter, over 95% individuals in the final population are identical to the final solution. These facts indicate that completely inheriting good "genes" by offspring deteriorates the population diversity, and results in premature and poor performances. To guarantee the population diversity, good "genes" (i.e. π_k^L $(k = 1,2,\ldots,|\Pi_s|)$) are inherited by offspring with a probability λ ($\lambda \in (0,1)$) in this paper. Let $\pi_{k,[j]}$ be the jth job in π_k. For any $\pi_k \in \Pi_s$ ($k = 1,2,\ldots,|\Pi_s|$), if $\pi_{k,[j]} \in \pi_k^L$ ($j = 1,2,\ldots,n$), job $\pi_{k,[j]}$ is copied to the position j in the offspring individual π_o with a probability λ. Then, each unassigned position j in π_o is set as the job i, which is selected from the unscheduled job set Ω_u with the probability $\frac{M_{[i][j]}}{\sum_{i\in\Omega_u} M_{[i][j]}}$. The offspring production procedure can be described as follows:

1. *PRODUCTION* (π_k,π_k^L)
2. $\pi_o \leftarrow$ Null. /*offspring initialization*/
3. For $j \leftarrow 1$ to n
4. Generate a uniform distributed random number $\mu(\mu \in [0,1])$.
5. If ($\pi_{k,[j]} \in \pi_k^L$ AND $\mu \leqslant \lambda$)
6. Insert $\pi_{k,[j]}$ to the position j ofπ_o.
7. $\Omega_u \leftarrow \Omega_u - \{\pi_{k,[j]}\}$.
8. For each unassigned position j in π_o
9. Select job i from Ω_u with probability $\frac{M_{[i][j]}}{\sum_{i\in\Omega_u} M_{[i][j]}}$.
10. Insert job i to the position j of π_o.
11. $\Omega_u \leftarrow \Omega_u - \{i\}$.
12. Return π_o.
13. End *PRODUCTION*

Since the "dominance matrix" M and the LCS $\pi_k^L (k = 1,2,\ldots,|\Pi_s|)$ have already been generated prior to *PRODUCTION*, the complexity of *PRODUCTION* is $O(n^2)$.

3.4. Replacement

In the proposed PEDA, the replacement procedure given by Ruiz et al. (2006) is adopted, and it was also used by Jarboui et al. (2009). An offspring individual replaces with the worst one in the current population as long as both of the following conditions being satisfied: (1) it is better than the worst one, and (2) no identical one already existing in the current population. It is obvious that this replacement not only improves the population quality but also keeps the population diversity.

3.5. Local search

To further improve the performance, a Variable Neighborhood Search algorithm (VNS) is employed as the local search in PHEDA. VNS is acted on offspring individuals with a probability *Penh* (*Penh* $\in (0,1]$) to balance the global search and the local search. Two effective neighborhood local search methods, *insert_local_search* and *swap_local_search* introduced by Jarboui et al. (2009), are adopted in VNS. However, the perturbation operator in JVNS generally conducted on the best solution found so far traps JVNS

into the local optimal. Therefore, an acceptance criterion is added to the proposed VNS. As well, a perturbation operator is incorporated by repeating d rounds the following procedure: randomly removing a job from the current permutation and reinserting it at a random position. This procedure is quite different from the perturbation operator in the IG given by Ruiz & Stützle (2007, 2008), in which d jobs are removed randomly to generate a list, and all the jobs in this list are reinserted sequentially by the NEH way (Nawaz, Enscore, & Ham, 1983).

In the proposed VNS, *insert_local_search* and *swap_local_search* are iterated on the current sequence π_c until no improvement could be made. If the obtained solution π_r is better than the best so far π_r^* in VNS (maybe π_r^* is different from π^*), π_r is absolutely accepted. Set $\pi_r^* \leftarrow \pi_r$, and conduct the perturbation operator on π_r. The resulted sequence updates π_c. Otherwise when $F(\pi_r) \geqslant F(\pi_r^*)$, π_r is accepted only with probability γ ($\gamma \in (0,1]$). The above procedure is repeated until the termination criterion is satisfied. Similar to that of Jarboui et al. (2009), the termination criterion is set as 50 consecutive iterations without improvement. The perturbation operator and VNS are given below:

1. *PERTURBATION* (π)
2. $\pi_c \leftarrow \pi$.
3. For $i \leftarrow 1$ to d
4. A job is randomly removed from π_c and reinserted into π_c.
5. Return π_c.
6. End *PERTURBATION*

1. *VNS* (π)
2. $\pi_r \leftarrow \pi, \pi_c \leftarrow \pi, \pi_r^* \leftarrow \pi$.
3. While (termination is not satisfied)
4. While (improvement obtained)
5. $\pi' \leftarrow$ *insert_local_search*(π_c).
6. $\pi_c \leftarrow$ *swap_local_search*(π').
7. If ($F(\pi_c) < F(\pi_r)$) then $\pi_r \leftarrow \pi_c$.
8. If ($F(\pi_r) < F(\pi_r^*)$) then
9. $\pi_r^* \leftarrow \pi_r$.
10. $\pi_r \leftarrow$ *PERTURBATION*(π_r).
11. $\pi_c \leftarrow \pi_r$.
12. Else
13. Generate a uniform distributed random number $\mu(\mu \in [0,1])$.
14. If ($\mu \leqslant \gamma$) then
15. $\pi_r \leftarrow$ *PERTURBATION*(π_r).
16. Else
17. $\pi_r \leftarrow$ *PERTURBATION*(π_r^*).
18. $\pi_c \leftarrow \pi_r$.
19. Return π_r^*.
20. End *VNS*

3.6. Description of the proposed EDA

From the above analysis, the proposed EDA can be formally described as follows.

1. *EDA*:
2. Initial parameters α, β, γ, λ, d and *Penh*.
3. Generate population P_c and select the best one as π^*.
4. While (termination is not satisfied)
5. Select the best $\lfloor \alpha |P_c| \rfloor$ individuals from P_c and construct Π_e.
6. $M \leftarrow$ *VOTING*(Π_e).
7. Select $\lfloor \beta |P_c| \rfloor$ individuals from P_c to generate Π_s by the roulette method.
8. For each π_k in Π_s /*$k = 1,2,\cdots,|\Pi_s|$*/
9. $\pi_k^L \leftarrow$ *FLCS*(π_k, π^*).
10. $\pi_o \leftarrow$ *PRODUCTION*(π_k, π_k^L).
11. VNS is performed on π_o with probability *Penh*.
12. Apply the replacement to update P_c.
13. Select the best one in P_c as π^*.
14. Return π^*.
15. End *EDA*

4. Computational results

In the section, parameters of PHEDA are determined by experiments. Performance is evaluated by comparing it with other existing good algorithms for the considered problem.

4.1. Parameter determination

Parameters α, β, γ, λ, d and *Penh* should be well determined. α exerts influence on the elite set, which provides information for the probability distribution model. A small α may lead insufficient information provided. On the other side, a large one indicates some poor individuals participate in the "voting" and send "bad messages" to the "dominance matrix". β determines the size of the seed set, each of which will be matched with the best solution found so far to mine good "genes". A small β leads to obtain only a few good "genes" while a large value implies that poor individuals could be involved in. γ is the acceptance probability. The larger value it is, the more poor individuals may be accepted with high probabilities. And, a low value decreases the effects of the acceptance criterion. λ is the probability of good "genes" inherited by offspring. A large value deteriorates the population diversity. On the contrary, good "genes" may be lost. d determines the perturbation strength. The larger it is, the more powerful the strength is. *Penh* is the probability of local search conducted on offspring individuals, which balances the global search and the local search. In this paper, the famous Design of Experiments (DoE) approach (Montgomery, 2000) is adopted to investigate the best parameter setting for the proposal. The value domains of these parameters are set as $\alpha \in \{0.1,0.3,0.5\}$,$\beta \in \{0.1,0.2,0.3\}$,$\gamma \in \{0.01,0.03,0.05\}$,$\lambda \in \{0,0.2,0.5,0.8,1.0\}$,$d \in \{1,2,3\}$ and $Penh \in \{0.01,0.015,0.03\}$, respectively. So, there are $5 \times 3^5 = 1215$ combinations totally, all of which are tested. The two specific situations with $\lambda = 0$ and $\lambda = 1$ are included with the purpose to show the impacts of λ. $\lambda = 0$ means no good "genes" inheritance, whereas $\lambda = 1$ denotes good "genes" are inherited by offspring completely, which is similar to the GAs given by Ruiz et al. (2006) & Zhang et al. (2009).

PHEDA is implemented in Java and tested on a PC with P4 2.93 GHz CPU & 512M Memory. The Taillard (1993) benchmark instances are considered, which contains 12 groups with size ($n \times m$) varying from 20×5 to 500×20. Each group has 10 instances, and there are 120 instances totally. In the literature, it can be investigated that many recent meta-heuristics (four ACOs given by Rajendran & Ziegler (2004, 2005), PSOvns introduced by Tasgetiren et al. (2007), HCPSO developed by Jarboui et al. (2008), DDERLS proposed by Pan et al. (2008), ILS constructed by Dong et al. (2009), JHEDA investigated by Jarboui et al. (2009) and HGAYT presented by Tseng & Lin (2009)) are only tested on the first 90 instances, so is the proposed PHEDA. ARPD is adopted to evaluate the performance.

$$ARPD = \sum_{i=1}^{R} \left(\frac{(S_i - S_{best}) \times 100}{S_{best}} \right) / R \tag{5}$$

In Eq. (5), for each instance, S_i denotes the solution generated by the compared algorithms, R is the number of replications and S_{best} represents the best one among the results obtained by all the compared algorithms during the R replications. Due to the same S_{best} used, ARPD reflects the performance of algorithms. The lower

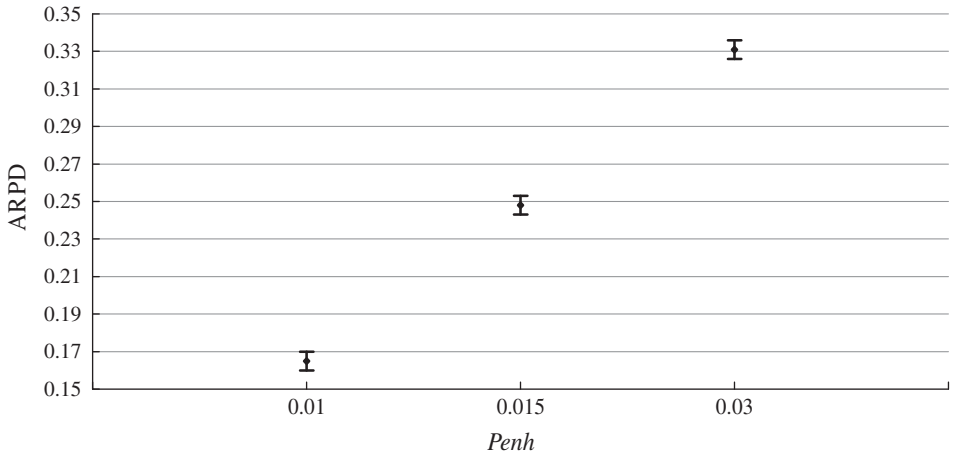

Fig. 2. Means plot for *Penh* factor, 95.0% LSD intervals.

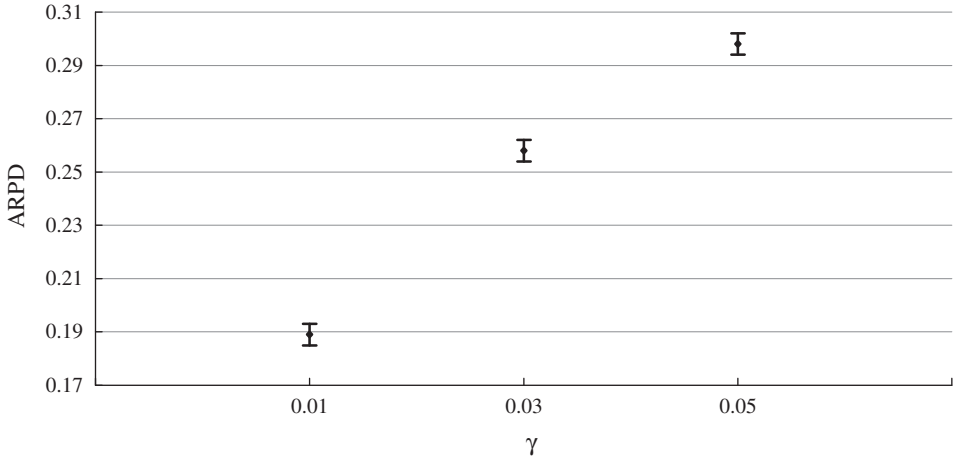

Fig. 3. Means plot for γ factor, 95.0% LSD intervals.

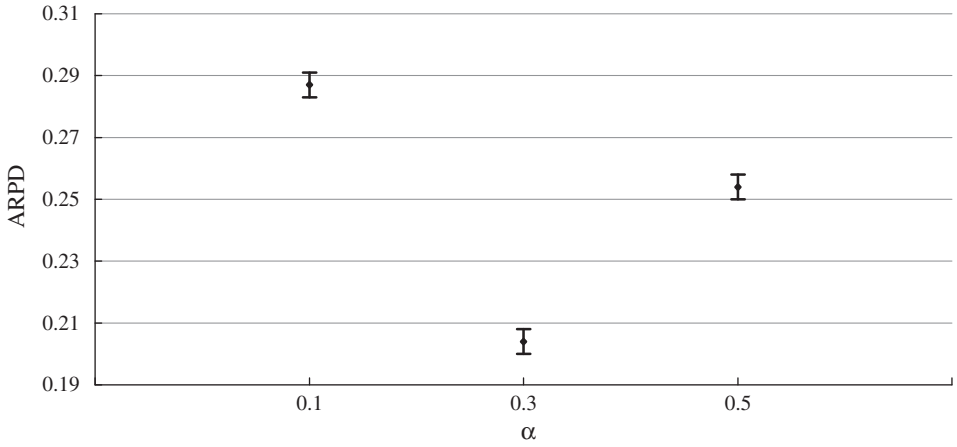

Fig. 4. Means plot for α factor, 95.0% LSD intervals.

ARPD is, the higher the performance is. In the experiment, $R = 10$ and the termination criterion is set as $(n \times m/2) \times 30$ ms maximum computation time. Setting the time limitation in this way allows the more computation time as the job number or the machine number increases. And, this method is also adopted by many researchers, such as Jarboui et al. (2009), Ruiz & Stützle (2007, 2008) & Ruiz et al. (2006). The experimental results are analyzed by the multi-factor analysis of variance (ANOVA) method. In the experiment, the three main hypotheses (normality, homoskedasticity and independence of the residuals) are checked and accepted. The *p*-values

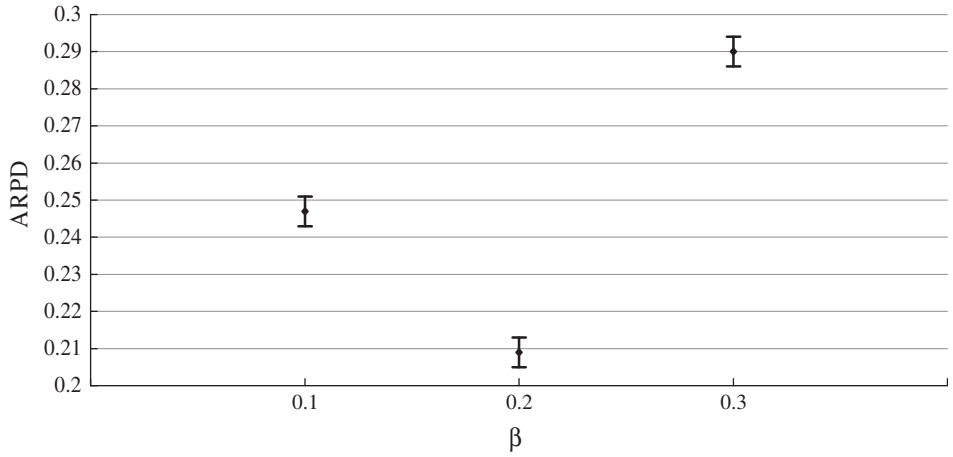

Fig. 5. Means plot for β factor, 95.0% LSD intervals.

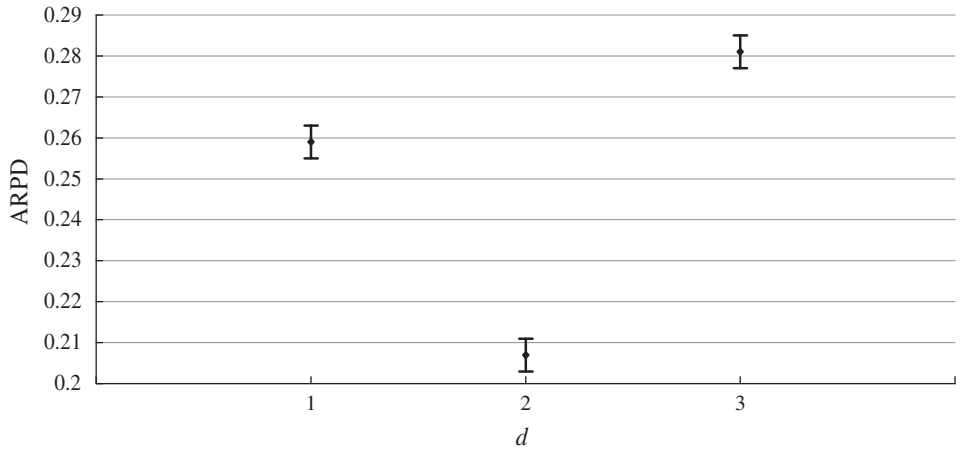

Fig. 6. Means plot for d factor, 95.0% LSD intervals.

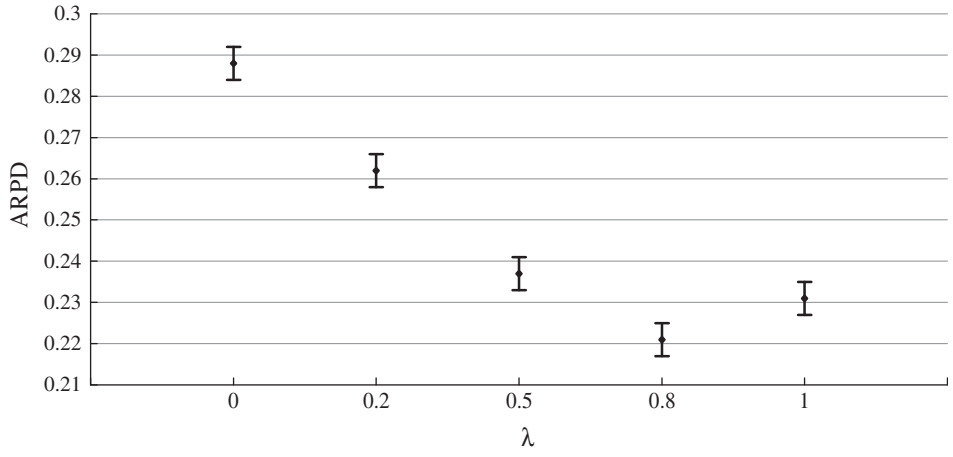

Fig. 7. Means plot for λ factor, 95.0% LSD intervals.

in the experiment are all close to zero, so analyzing the p-values is rarely useful. Instead, the F-ratio is focused on, which denotes the ratio between the variance explained by a factor and the unexplained variance. The greater the F-ratio is, the more effect the factor has on the response variable. Note that the interactions among more than two factors are not considered, since their F-ratios are quite small. The factor with the greatest F-ratio is first analyzed, followed by the second one, and so on. The greatest

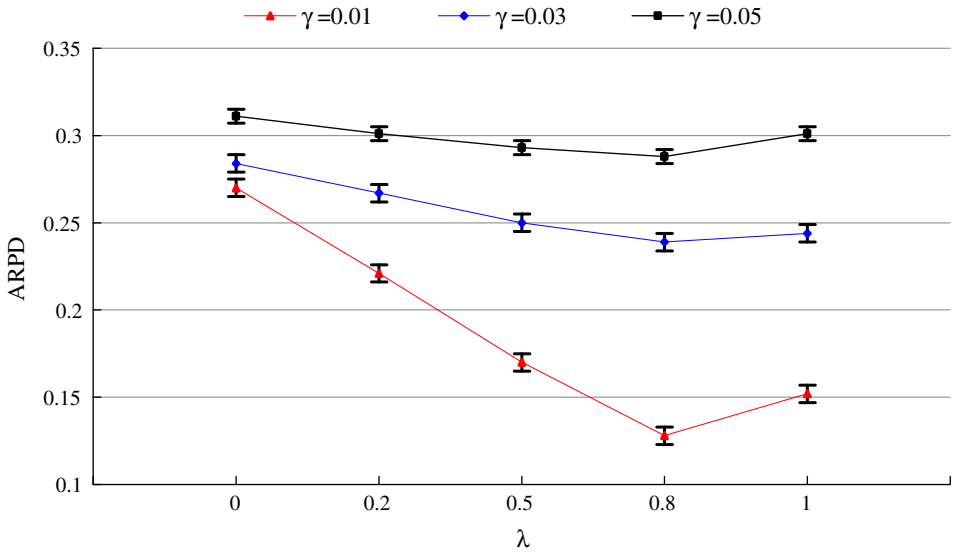

Fig. 8. Means plot for the interaction between λ and γ, 95.0% LSD intervals.

Table 1
Performance comparisons between PEDA and JEDA (no local search hybridized).

Problem	JEDA	PEDA
20 × 5	4.78	**3.65**
20 × 10	3.42	**2.98**
20 × 20	2.19	**1.82**
50 × 5	3.21	**2.94**
50 × 10	3.50	**2.99**
50 × 20	3.38	**2.21**
100 × 5	2.76	**2.18**
100 × 10	2.58	**1.61**
100 × 20	1.88	**1.10**
Average	3.08	**2.39**

Table 2
Performance comparisons on Taillard benchmark instances.

Problem	ILS	DDERLS	JHEDA	PHEDA
20 × 5	0.026	0.045	0.026	**0**
20 × 10	0.047	0.007	0.015	**0**
20 × 20	0.017	0.020	0.011	**0**
50 × 5	0.542	1.021	0.343	**0.201**
50 × 10	0.505	0.998	0.418	**0.352**
50 × 20	0.571	0.811	0.431	**0.380**
100 × 5	0.861	2.398	0.358	**0.280**
100 × 10	1.003	2.358	0.574	**0.439**
100 × 20	1.231	1.812	0.665	**0.291**
Average	0.534	1.052	0.316	**0.216**

F-ratio corresponds to the factor *Penh*, and the means plot with the Least Significant Difference (LSD) intervals (at the 95% confidence level) is given in Fig. 2.

Fig. 2 illustrates that PHEDA with *Penh* = 0.01 obtains the significantly best performance, while that with *Penh* = 0.03 yields the worst effectiveness.

The factor γ has the second greatest *F*-ratio, and the means plot with LSD intervals (at the 95% confidence level) is given in Fig. 3.

Fig. 3 shows that PHEDA with $\gamma = 0.01$ obtains the significantly best performance. PHEDA with a large γ value of 0.05 generates the worst effectiveness, and the reason lies in that many poor individuals are accepted.

Table 3
Performance comparisons between PHEDA and HGAYT.

Instance	PHEDA		HGAYT	
	ARPD	*t* (s)	ARPD	*t* (s)
50 × 5	0.196	15	**0.192**	16.55
50 × 10	**0.276**	30	0.329	40.33
50 × 20	**0.287**	70	0.318	82.23
100 × 5	**0.196**	80	0.274	94.17
100 × 10	**0.351**	200	0.458	251.62
100 × 20	**0.142**	500	0.419	588.86
Average	**0.241**	149.20	0.332	178.96

The three factors (α, β and d) have similar *F*-ratios, and the means plots with LSD intervals (at the 95% confidence level) are given in Figs. 4–6, respectively.

In Fig. 4, it can be seen that PHEDA with $\alpha = 0.3$ obtains the significantly best performance. $\alpha = 0.1$ yields the worst effectiveness and the reason is that only a few individuals participate in the "voting" procedure and the information is insufficient to construct the probability model. Fig. 5 shows that PHEDA with $\beta = 0.2$ obtains the significantly best performance. In Fig. 6, it can be seen that PHEDA with $d = 2$ generates the best effectiveness, significantly.

The last factor is λ, and the means plot with LSD intervals (at the 95% confidence level) is given in Fig. 7.

In Fig. 7, it can be seen that PHEDA with $\lambda = 0.8$ achieves the significantly best performance. PHEDA with $\lambda = 0$ generates the worst effectiveness, which implies that the inheritance of good "genes" improves the performance. The performance with $\lambda = 1$ is significantly worse than that with $\lambda = 0.8$, which indicates that completely inheriting the good "genes" deteriorates the performance.

Interactions between factors are considered as well. The interaction between λ and γ is focused on, and the means plot with LSD intervals (at the 95% confidence level) is given in Fig. 8. The other interactions between factors are not studied, since their *F*-ratios are very small. Fig. 8 illustrates that PHEDA obtains the best performance with the combination of $\lambda = 0.8$ and $\gamma = 0.01$, which conforms to the previous conclusion.

According to the above analysis, all the six parameters are selected as follows: $\alpha = 0.3$, $\beta = 0.2$, $\gamma = 0.01$, $\lambda = 0.8$, $d = 2$ and *Penh* = 0.01.

Table 4
Best known solutions for the first 90 Taillard benchmark instances.

Instance	Solutions	Algorithms
ta001	14033	PSOvns,HCPSO,HGA,JVNS,JHEDA,ILS,**PHEDA**,DDERLS,IGRIS,HGAYT
ta002	15151	MMMAS,PSOvns,HCPSO,HGA,JVNS,JHEDA,ILS,**PHEDA**,DDERLS,IGRIS,HGAYT
ta003	13301	PSOvns,HCPSO,HGA,JVNS,JHEDA,ILS,**PHEDA**,DDERLS,IGRIS,HGAYT
ta004	15447	PSOvns,HCPSO,HGA,JVNS,JHEDA,ILS,**PHEDA**,DDERLS,IGRIS,HGAYT
ta005	13529	MMMAS,PACO,PSOvns,HCPSO,HGA,JVNS,JHEDA,ILS,**PHEDA**,DDERLS,IGRIS,HGAYT
ta006	13123	PACO,PSOvns,HCPSO,HGA,JVNS,JHEDA,ILS,**PHEDA**,DDERLS,IGRIS,HGAYT
ta007	13548	PSOvns,HCPSO,HGA,JVNS,JHEDA,ILS, **PHEDA**,DDERLS,IGRIS,HGAYT
ta008	13948	PSOvns,HCPSO,HGA,JVNS,JHEDA,ILS,**PHEDA**,DDERLS,IGRIS,HGAYT
ta009	14295	PSOvns,HCPSO,HGA,JVNS,JHEDA,ILS,**PHEDA**,DDERLS,IGRIS,HGAYT
ta010	12943	PSOvns,HCPSO,HGA,JVNS,JHEDA,ILS,**PHEDA**,DDERLS,IGRIS,HGAYT
ta011	20911	PSOvns,HCPSO,HGA,JVNS,JHEDA,ILS,**PHEDA**,DDERLS,IGRIS,HGAYT
ta012	22440	MMMAS,PSOvns,HCPSO,HGA,JVNS,JHEDA,ILS,**PHEDA**,DDERLS,IGRIS,HGAYT
ta013	19833	MMMAS,PSOvns,HCPSO,HGA,JVNS,JHEDA,ILS,**PHEDA**,DDERLS,IGRIS,HGAYT
ta014	18710	PSOvns,HCPSO,JHEDA,ILS,**PHEDA**,DDERLS,IGRIS,HGAYT
ta015	18641	PSOvns,HCPSO,HGA,JVNS,JHEDA,ILS,**PHEDA**,DDERLS,IGRIS,HGAYT
ta016	19245	MMMAS,PACO,HCPSO,JHEDA,ILS,**PHEDA**,DDERLS,IGRIS,HGAYT
ta017	18363	PSOvns,HCPSO,HGA,JVNS,JHEDA,ILS,**PHEDA**,DDERLS,IGRIS,HGAYT
ta018	20241	MMMAS,PSOvns,HCPSO,HGA,JVNS,JHEDA,ILS,**PHEDA**,DDERLS,IGRIS,HGAYT
ta019	20330	MMMAS,PACO,PSOvns,HCPSO,HGA,JVNS,JHEDA,ILS,**PHEDA**,DDERLS,IGRIS,HGAYT
ta020	21320	MMMAS,PSOvns,HCPSO,HGA,JVNS,JHEDA,ILS,**PHEDA**,DDERLS,IGRIS,HGAYT
ta021	33623	MMMAS,HCPSO,HGA,JVNS,JHEDA,ILS,**PHEDA**,DDERLS,IGRIS,HGAYT
ta022	31587	HCPSO,HGA,JVNS,JHEDA,ILS,**PHEDA**,DDERLS,IGRIS,HGAYT
ta023	33920	MMMAS,HCPSO,HGA,JVNS,JHEDA,ILS,**PHEDA**,DDERLS,IGRIS,HGAYT
ta024	31661	HCPSO,HGA,JVNS,JHEDA,ILS,**PHEDA**,DDERLS,IGRIS,HGAYT
ta025	34557	HCPSO,HGA,JVNS,JHEDA,ILS,**PHEDA**,DDERLS,IGRIS,HGAYT
ta026	32564	HCPSO,HGA,JVNS,JHEDA,ILS,**PHEDA**,DDERLS,IGRIS,HGAYT
ta027	32922	PACO,HCPSO,HGA,JVNS,JHEDA,ILS,**PHEDA**,DDERLS,IGRIS,HGAYT
ta028	32412	HCPSO,HGA,JVNS,JHEDA,ILS,**PHEDA**,DDERLS,IGRIS,HGAYT
ta029	33600	HCPSO,HGA,JVNS,JHEDA,ILS,**PHEDA**,DDERLS,IGRIS,HGAYT
ta030	32262	HCPSO,HGA,JVNS,JHEDA,ILS,**PHEDA**,DDERLS,IGRIS,HGAYT

Instance	Solutions	Algorithms	Instance	Solutions	Algorithms
ta031	**64809**	**PHEDA**	ta061	**254031**	**PHEDA**
ta032	**68054**	**PHEDA**	ta062	**243017**	**PHEDA**
ta033	**63238**	**PHEDA**	ta063	**238421**	**PHEDA**
ta034	68281	HGAYT	ta064	**228139**	**PHEDA**
ta035	**69447**	**PHEDA**	ta065	241255	HGAYT
ta036	**66879**	**PHEDA**	ta066	**233001**	**PHEDA**
ta037	**66270**	**PHEDA**	ta067	**241201**	**PHEDA**
ta038	**64332**	**PHEDA**	ta068	**231841**	**PHEDA**
ta039	62981	JVNS	ta069	**249029**	**PHEDA**
ta040	**68770**	**PHEDA**	ta070	**243591**	**PHEDA**
ta041	87238	JHEDA	ta071	**300101**	**PHEDA**
ta042	83001	HGAYT	ta072	**275509**	**PHEDA**
ta043	**80101**	**PHEDA**	ta073	**288891**	**PHEDA**
ta044	86725	JVNS	ta074	**303410**	**PHEDA**
ta045	86508	ILS	ta075	**286396**	**PHEDA**
ta046	**86731**	**PHEDA**	ta076	**271801**	**PHEDA**
ta047	**88942**	**PHEDA**	ta077	**280832**	**PHEDA**
ta048	86860	HGAYT	ta078	**292954**	**PHEDA**
ta049	85688	JHEDA	ta079	**303712**	**PHEDA**
ta050	88149	JHEDA	ta080	**293210**	**PHEDA**
ta051	125831	JHEDA	ta081	**368491**	**PHEDA**
ta052	119247	JHEDA	ta082	**374333**	**PHEDA**
ta053	116696	JHEDA	ta083	**371970**	**PHEDA**
ta054	**120261**	**PHEDA**	ta084	**374339**	**PHEDA**
ta055	118457	JHEDA	ta085	**370910**	**PHEDA**
ta056	120820	JHEDA	ta086	**373717**	**PHEDA**
ta057	**123201**	**PHEDA**	ta087	**376122**	**PHEDA**
ta058	122770	HGAYT	ta088	**387002**	**PHEDA**
ta059	121872	HGAYT,JHEDA	ta089	**377221**	**PHEDA**
ta060	124354	HGAYT	ta090	**380587**	**PHEDA**

4.2. Performance of PEDA

PEDA is compared with JEDA (Jarboui et al., 2009), which is also implemented in Java. Parameters of JEDA are set as $\delta_1 = \delta_2 = 4/n$, $Q = O = 3$ and the population size is 60, which are identical to those used in the paper (Jarboui et al., 2009). And, parameters of PEDA are set as above with the population size being 60 as well for fair comparison. Both algorithms are tested on the same PC with P4 2.93 GHz CPU & 512M Memory. The first 90 Taillard benchmark instances are also tested, and the termination criteria for both the algorithms are identically set as $(n \times m/2) \times 30$ ms maximum computation time. $R = 100$ is consistent with that of Jarboui et al. (2009). Results are presented in Table 1.

Table 1 shows that PEDA outperforms JEDA for each group as well as the average on the nine groups, which implies that the incorporation of LCS for good "genes" mining leverages PEDA with high effectiveness. Good "genes" inherited by offspring individuals with the probability $\lambda = 0.8$ can maintain the population diversity.

Table 5
Performance comparisons between PEDA_GM and PEDA (no local search hybridized).

Problem	PEDA_GM	PEDA
20 × 5	5.08	**3.12**
20 × 10	3.96	**2.51**
20 × 20	2.93	**1.33**
50 × 5	4.26	**3.18**
50 × 10	4.14	**3.36**
50 × 20	3.63	**1.88**
100 × 5	3.21	**1.92**
100 × 10	2.96	**1.34**
100 × 20	2.10	**1.03**
Average	3.59	**2.19**

This is another important factor benefiting the high performance of PEDA.

4.3. Performance of PHEDA

In this subsection, the performance of PHEDA is evaluated by comparing it with other existing meta-heuristics for the considered problem in the literature, including ILS (Dong et al., 2009), DDERLS (Pan et al., 2008), HGAYT (Tseng & Lin, 2009), and JHEDA (Jarboui et al., 2009). As aforementioned in Section 1, all of these four algorithms outperform PSOvns (Tasgetiren et al., 2007). DDERLS and JHEDA are better than HCPSO (Jarboui et al., 2008), whereas it is claimed that JHEDA outperforms HGA (Zhang et al., 2009). Therefore, PSOvns, HCPSO and HGA are not compared in this paper.

Besides the proposed PHEDA, all the compared algorithms are implemented in Java with identical parameters to those in their papers. The population sizes of DDERLS, JHEDA and PHEDA are all set as 30 and the termination criteria for all of the algorithms are identically set as $(n \times m/2) \times 90$ ms maximum computation time. The first 90 Taillard benchmark instances are tested as well. $R = 5$, which is consistent with most of papers in the literature. Results are given in Table 2.

Table 2 illustrates that PHEDA obtains the best performance both on average and for each group. All the compared algorithms yield good solutions for the first three groups with small sizes, and PHEDA achieves the best performance for ARPD being 0. PHEDA also obtains the best effectiveness for the rest groups, especially for the largest size one (i.e. 100 × 20). Its ARPD (0.291) is much lower than those of JHEDA (0.665), ILS (1.231) and DDERLS (1.812). The performance of JHEDA is just inferior to that of PHEDA on average. ILS outperforms DDERLS, which obtains good solutions for the first three groups, whereas rather poor results for the rest ones with large sizes.

To compared with HGAYT, PHEDA is performed again for 10 replications, i.e. $R = 10$, which is consistent with Tseng & Lin (2009). Both algorithms obtain good results for the first three groups, so the rest six groups with large sizes ($\{50,100\} \times \{5,10,20\}$) are mainly tested. In ARPD, S_{best} represents the upper bounds given by Tseng & Lin (2009). Additionally, Tseng & Lin (2009) presented the average total flowtime ("Avg" for short) obtained during their experiments, which can be used directly to calculate the ARPD by the following equation:

$$ARPD = \frac{Avg - S_{best}}{S_{best}} \times 100\% \quad (6)$$

It is obvious that Eq. (6) is equivalent to Eq. (5), which is used for performance evaluation in this paper. Tseng and Lin used a PC with a CPU AMD K7 1.83 GHz & 512M Memory, which is reported to be comparable with a computer with CPU P4 2.60 GHz & 256M Memory (Tseng & Lin, 2009). So, it seems that our computer (CPU P4 2.93 GHz & 512M Memory) is a little better than the one used by Tseng and Lin, but it is well-known that the efficiency of Java is much lower than that of C++. Therefore, totally, our testing environment can be considered to be comparable with that of Tseng & Lin (2009). In fact, this comparison method was also adopted by Tseng & Lin (2009) to evaluate the efficiency of their hybrid GA. The termination criterion of PHEDA is set as maximum computation time with values summarized in Table 3.

In Table 3, computation times of HGAYT are all summarized from the paper (Tseng & Lin, 2009). Table 3 illustrates that PHEDA obtains better performance in shorter computation time than HGAYT does on average. PHEDA generates the best solutions for all testing groups except the 50 × 5 group. Furthermore, the best solutions found by the proposed PHEDA are given and compared with the best solutions found so far which were reported by Tseng & Lin (2009), Dong et al. (2009), Pan et al. (2008), & Jarboui et al. (2009). The best solutions so far and the corresponding algorithms for the Taillard benchmark instances are shown in Table 4, which illustrates that PHEDA obtains 72 best solutions for the first 90 Taillard benchmark instances, and 42 are newly discovered. By comparing these with the upper bounds given by Tseng & Lin (2009), it is obvious that PHEDA outperforms HGAYT greatly.

4.4. Effectiveness of the proposed PRODUCTION operator

To evaluate the effectiveness of the proposed *PRODUCTION* operator, it is compared with the GM operator given by Zhang et al. (2005). The main ideas of GM lie in: (1) in each generation, offspring individuals are generated from the same seed, the best solution found so far, (2) elements of each offspring individual are either copied from the corresponding elements of the best solution found so far with probability θ, or sampled from the probability model with probability $1 - \theta$. Accordingly, a new GM operator (called *PRODUCTIOIN_GM*) is constructed for the problem considered in this paper, as GM given by Zhang et al. (2005) is used by an EDA for the maximum clique problem. Let π^* be the best solution found so far, Ω_u be the set of unassigned jobs, and π_o represent the offspring individual. For each position j ($j = 1,2,\ldots,n$) in π^*, insert $\pi^*_{[j]}$ to the position j of π_o with the probability θ. Then, each unassigned position j in π_o is set as the job i, which is selected from the unscheduled job set Ω_u with probability $\frac{M_{[i][j]}}{\sum_{i\in\Omega_u} M_{[i][j]}}$. The *PRODUCTIOIN_GM* is formally described as follows:

1. *PRODUCTIOIN_GM* (π^*) /*the seed is always the best solution found so far π^{**}/
2. $\pi_o \leftarrow$ Null. /*offspring initialization*/
3. For $j \leftarrow 1$ to n /*inherit elements of π^* by the offspring individual*/
4. Generate a uniform distributed random number $\mu(\mu \in [0,1])$.
5. If ($\mu \leqslant \theta$)
6. Insert $\pi^*_{[j]}$ to the position j ofπ_o.
7. $\Omega_u \leftarrow \Omega_u - \{\pi^*_{[j]}\}$.
8. For each unassigned position j in π_o /* sample from the probability model*/
9. Select job i from Ω_u with probability $\frac{M_{[i][j]}}{\sum_{i\in\Omega_u} M_{[i][j]}}$.
10. Insert job i to the position j of π_o.
11. $\Omega_u \leftarrow \Omega_u - \{i\}$.
12. Return π_o.
13. End *PRODUCTIOIN_GM*

By using *PRODUCTIOIN_GM* and keeping the other operators unchanged, a new EDA is constructed and named as PEDA_GM. And, the hybrid PEDA_GM is called PHEDA_GM for simplicity. It should be noted that, in PEDA_GM and PHEDA_GM, instead of constructing the seed set $\Pi_s(|\Pi_s| = \lfloor\beta|P_c|\rfloor)$, *PRODUCTIOIN_GM* is performed $\lfloor\beta|P_c|\rfloor$ rounds. As well, DoE and ANOVA are also adopted to deter-

mine the parameters of PEDA_GM and PHEDA_GM. $\theta \in \{0, 0.2, 0.5, 0.8, 1.0\}$ and other parameter domains are identical to those in Section 4.1. So, there are $5 \times 3^5 = 1215$ combinations totally, all of which are tested. In this experiment, $R = 5$ and the termination criterion is set as $(n \times m/2) \times 2$ ms maximum computation time. Similar to the above analysis, the best parameter setting is determined as follows: $\alpha = 0.3$, $\beta = 0.2$, $\gamma = 0.03$, $\theta = 0.5$, $d = 2$ and $Penh = 0.01$.

PEDA_GM is compared with PEDA. The population sizes of both algorithms are set as 30. The first 90 Taillard benchmark instances are also tested, and the termination criteria for both the algorithms are also set as $(n \times m/2) \times 30$ ms maximum computation time. $R = 5$, and the results are shown in Table 5.

Table 5 shows that PEDA outperforms PEDA_GM for each group as well as the average on the nine groups. The performance of PEDA is better than that of PEDA_GM for nearly 39% $((3.59 - 2.19)/3.59 \times 100\%)$, on average.

PHEDA_GM is compared with PHEDA. The population sizes of both algorithms are set as 30. The first 90 Taillard benchmark instances are also tested, and the termination criteria for both the algorithms are identically set as $(n \times m/2) \times 90$ ms maximum computation time. $R = 5$, and the results are shown in Table 6.

Table 6 illustrates that the effectiveness of PHEDA is much better than that of PHEDA_GM. According to Tables 5 and 6, it can be concluded that the proposed *PRODUCTION* is much more effective than *PRODUCTION_GM*, which adopts the main ideas of GM given by Zhang et al. (2005). In *PRODUCTION*, all the offspring individuals are generated from different seeds, which are selected from the current population by the roulette method. And, *PRODUCTION* only considers the LCS between the seed individual and the best solution found so far as good "genes", which are inherited by the offspring individuals probabilistically. This method not only inherits good "genes" by the offspring individuals but also keeps the diversity of population. On the other side, in *PRODUCTION_GM* and GM, for each generation, all the offspring individuals are generated from the same seed: the best solution found so far. And, *PRODUCTION_GM* and GM consider all the elements of the best solution found so far as good "genes", and inherit them by the offspring individuals probabilistically. So, the generated offspring individuals are quite similar, and the diversity of the population decreases very fast, which deteriorates the performance. To verify this conclusion, another experiment is conducted to explore the diversity of the population during the evolution. The genotype diversity defined by Zhu (2003) is adopted, which can be described below:

Table 6
Performance comparisons between PHEDA_GM and PHEDA (local search hybridized).

Problem	PHEDA_GM	PHEDA
20×5	0.035	**0**
20×10	0.063	**0**
20×20	0.079	**0**
50×5	0.428	**0.175**
50×10	0.461	**0.311**
50×20	0.692	**0.359**
100×5	0.529	**0.280**
100×10	0.634	**0.401**
100×20	0.786	**0.239**
Average	0.412	**0.196**

Definition 3. Zhu, 2003The genotype diversity of the population can be defined as

$$Diversity(P_c) = \frac{1}{|P_c|(|P_c|-1)\cdot n} \times \sum_{\pi_A, \pi_B \in P_c, \pi_A \neq \pi_B} Hamming(\pi_A, \pi_B) \quad (7)$$

where $Hamming(\pi_A, \pi_B) = \sum_{i=1}^{n} fn(\pi_{A,[i]}, \pi_{B,[i]})$ and $fn(\pi_{A,[i]}, \pi_{B,[i]}) = \begin{cases} 0, & \pi_{A,[i]} = \pi_{B,[i]} \\ 1, & otherwise \end{cases}$. It is obvious that the maximum of $Hamming(\pi_A, \pi_B)$ is n, so $Diversity(P_c)$ is restricted in $[0,1]$. The larger $Diversity(P_c)$ is, the higher the diversity of the population P_c is. PEDA, PEDA_GM, PHEDA and PHEDA_GM are performed on the first 90 Taillard benchmark instances, and each instance is tested five times. The termination criterion is set as 200 maximum generations, and $Diversity(P_c)$ of every 10 generations is calculated and recorded. The obtained $Diversity(P_c)$ are averaged over all the instances, and the experimental results are shown in Figs. 9 and 10.

Figs. 9 and 10 illustrate that the population diversities of both PEDA_GM and PHEDA_GM decrease very fast while those of PEDA and PHEDA decrease in a moderate way. This conforms to the above conclusions.

4.5. Effectiveness of the proposed PERTURBATION operator

To explore the effectiveness of *PERTURBATION*, an experiment is designed. By replacing *PERTURBATION* of proposed PHEDA with the perturbation operator of IG, a new algorithm (PHEDA_IG) is constructed and compared with PHEDA. Parameters of PHEDA_IG are determined by an experiment similar to that in Section 4.1. The

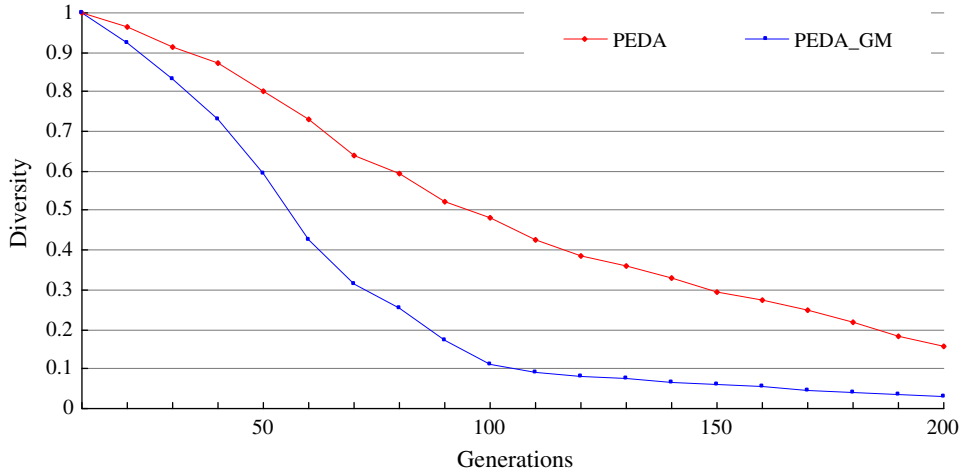

Fig. 9. Population diversity comparison between PEDA and PEDA_GM.

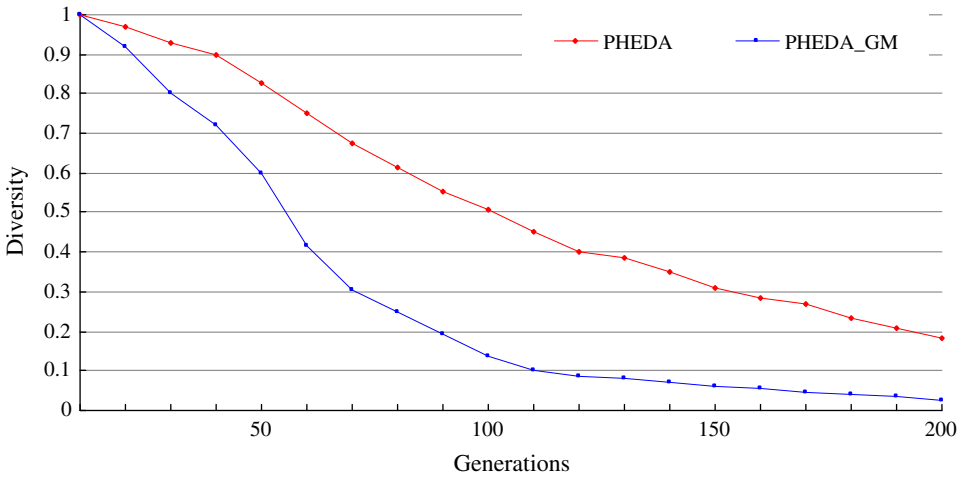

Fig. 10. Population diversity comparison between PHEDA and PHEDA_GM.

Table 7
Performance comparisons between PHEDA_IG and PHEDA (local search hybridized).

Problem	PHEDA_IG	PHEDA
20×5	**0**	**0**
20×10	**0**	**0**
20×20	0.017	**0**
50×5	0.243	**0.184**
50×10	0.459	**0.331**
50×20	0.612	**0.395**
100×5	0.309	**0.280**
100×10	0.513	**0.408**
100×20	0.392	**0.267**
Average	0.283	**0.207**

number of removed jobs in the perturbation operator is 3 (i.e. $d = 3$), and the other parameters are identical to those of PHEDA. The population sizes of both algorithms are set as 30. The first 90 Taillard benchmark instances are used as well, and the termination criteria for both the algorithms are set as $(n \times m/2) \times 90$ ms maximum computation time. $R = 5$. The results are shown in Table 7.

Table 7 shows that PHEDA outperforms PHEDA_IG for each group. Both algorithms generate good effectiveness for the two small size groups (20×5 and 20×10), whereas PHEDA obtains better solutions than PHEDA_IG for large size ones. This implies that the proposed *PERTURBATION* operator is more effective than that of IG, particularly for the large size instances.

5. Conclusions

In this paper, an Estimation of Distribution Algorithm (PEDA) was proposed for permutation flow shops with total flowtime minimization. To guarantee the diversity, all the individuals in the initial population were generated randomly. The Longest Common Subsequence (LCS) was incorporated into the probability distribution model to mine good "genes". Different from common EDAs, each offspring individual in PEDA was produced from a seed, selected from the current population by the roulette method. The LCS between the seed individual and the best solution found so far was considered as good "genes", and would be inherited by offspring with a probability, less than 100%, to guarantee the population diversity. To further improve the performance, an effective Variable Neighborhood Search (VNS) was incorporated into PEDA to form a hybrid EDA called PHEDA.

Experimental results showed that the proposed PEDA outperforms JEDA, and PHEDA is better than other existing algorithms for the considered problem. Moreover, PHEDA found 72 best solutions for the first 90 Taillard benchmark instances, in which 42 were newly discovered. Good "genes" mined by LCS can improve the performance of both PEDA and PHEDA.

Acknowledgments

This work is supported by the Scientific Research Foundation of Graduate School of Southeast University under Grant No. YBJJ0930, the Research and Innovation Program of Universities in Jiangsu Province under Grant No. CX09B_053Z, the National Natural Science Foundation of China (under Grant Nos. 60973073 and 60873236) and the National High Technology Research and Development Program of China (863 program) under Grant No. 2008AA04Z103.

References

Allahverdi, A., & Aldowaisan, T. (2002). New heuristics to minimize total completion time in m-machine flowshops. *International Journal of Production Economics., 77*, 71–83.

Baluja, S. (1994). Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning. Carnegie Mellon University, Technical Report.

Baluja, S., & Davies, S. (1998). Fast probability distribution modeling for combinatorial optimization. *Proceedings of the National Conference on Artificial Intelligence*, 469–476.

Blanco, R., Larranaga, P., & Inza, I. (2004). Gene selection for cancer classification using wrapper approaches. *International Journal of Pattern Recognition and Artificial Intelligence, 18*, 1373–1390.

Bosman, P. A. N., & Thierens, D. (2001). Advancing continuous IDEA's with mixture distributions and factorization selection metrics. *Proceedings, Genetic and Evolutionary Computation Conference*, 208–212.

Cesar, R. M., Bengoetxea, E., Bloch, I., & Larrangaga, P. (2005). Inexact graph matching for model-based recognition: Evaluation and comparison of optimization algorithms. *Pattern Recognition, 38*, 2099–2113.

Chang, P. C., Chen, S. H., & Liu, C. H. (2007). Sub-population genetic algorithm with mining gene structures for multi-objective flowshop scheduling problems. *Expert Systems with Applications, 33*, 762–771.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2006). *Introduction to algorithms* (2nd ed.). Cambridge, MA: MIT Press.

De Bonet, J. S. I., & Viola, P. (1997). MIMIC: Finding optima by estimating probability densities. *Advances in Neural Information Processing Systems, 9*, 424–431.

Dong, X., Huang, H., & Chen, P. (2009). An iterated local search algorithm for the permutation flowshop problem with total flowtime criterion. *Computers & Operations Research, 36*, 1664–1669.

Framinan, J. M., & Leisten, R. (2003). An efficient constructive heuristic for flowtime minimisation in permutation flowshops. *OMEGA, 31*, 311–317.

Framinan, J. M., Leisten, R., & Ruiz, R. (2005). Comparison of heuristics for flowtime minimisation in permutation flowshops. *Computers & Operations Research, 32*, 1237–1254.

French, S. (1982). *Sequencing and scheduling: an introduction to the mathematics of the job-shop*. Chichester: Ellis Horwood.

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

Hohfeld, M., & Rudolph, G. (1997). Towards a theory of population based incremental learning. *Proceedings of the International Conference on Evolutionary Computation*, 1–5.

Inza, P., Larranaga, P., & Etxeberria, B. S. (2000). Feature subset selection by Bayesian network-based optimization. *Artificial Intelligence, 123*, 157–184.

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

Jarboui, B., Ibrahim, S., Siarry, P., & Rebai, A. (2008). A combinatorial particle swarm optimisation for solving permutation flowshop problems. *Computers & Industrial Engineering, 54*, 526–538.

Johnson, S. M. (1954). Optimal two-and-three-state production schedules with setup times included. *Naval Research Logistics Quarterly, 1*, 61–68.

Li, X., Wang, Q., & Wu, C. (2009). Efficient composite heuristics for total flowtime minimization in permutation flowshops. *OMEGA, 37*, 155–164.

Liu, J., & Reeves, C. R. (2001). Constructive and composite heuristic solutions to the $P//\Sigma C_i$ scheduling problem. *European Journal of Operational Research, 132*, 439–452.

Miagkikh, V. V., & Punch, W. F. (1999). Global search in combinatorial optimization using reinforcement learning algorithms. *Proceedings of the International Conference on Evolutionary Computation*, 189–196.

Montgomery, D. C. (2000). *Design and analysis of experiments* (5th ed.). New York: Wiley.

Mühlenbein, H. (1998). The equation for response to selection and its use for prediction. *Evolutionary Computation, 5*, 303–346.

Nawaz, M., Enscore, E. E., & Ham, I. (1983). A heuristic algorithm for the *m*-machine *n*-job flow-shop sequencing problem. *OMEGA, 11*, 91–95.

Pan, Q. K., Tasgetiren, M. F., & Liang, Y. C. (2008). A discrete differential evolution algorithm for the permutation flowshop scheduling problem. *Computers & Industrial Engineering, 55*, 795–816.

Pelikan, M., Goldberg, D. E., & Cantu, E. (1999). BOA: The Bayesian optimization algorithm. *Proceedings, Genetic and Evolutionary Computation Conference*, 525–532.

Rajendran, C. (1994). A heuristic for scheduling in flowshop and flowline-based manufacturing cell with multi-criteria. *International Journal of Production Research, 32*, 2541–2558.

Rajendran, C., & Ziegler, H. (1997). An efficient heuristic for scheduling in a flowshop to minimize total weighted flowtime of jobs. *European Journal of Operational Research, 103*, 129–138.

Rajendran, C., & Ziegler, H. (2004). Ant-colony algorithms for permutation flowshop scheduling to minimize makespan_total flowtime of jobs. *European Journal of Operation Research, 115*, 426–438.

Rajendran, C., & Ziegler, H. (2005). Two ant-colony algorithms for minimizing total flowtime in permutation flowshops. *Computers & Industrial Engineering, 48*, 789–797.

Ruiz, R., Maroto, C., & Alcaraz, J. (2006). Two new robust genetic algorithms for the flowshop scheduling problem. *OMEGA, 34*, 461–476.

Ruiz, R., & Stützle, T. (2007). A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. *European Journal of Operational Research, 177*, 2033–2049.

Ruiz, R., & Stützle, T. (2008). An iterated greedy heuristic for the sequence dependent setup times flowshop problem with makespan and weighted tardiness objectives. *European Journal of Operational Research, 187*, 1143–1159.

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

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

Tasgetiren, M. F., Liang, Y. C., Sevkli, M., & Gencyilmaz, G. (2007). A particle swarm optimization algorithm for makespan and total flowtime minimization in the permutation flowshop sequencing problem. *European Journal of Operational Research, 177*, 1930–1947.

Tseng, L. Y., & Lin, Y. T. (2009). A hybrid genetic local search algorithm for the permutation flowshop scheduling problem. *European Journal of Operational Research, 198*, 84–92.

Vempati, V. S., Chen, C. L., & Bulllngton, S. F. (1993). An effective heuristic for flow shop problems with total flow time as criterion. *Computers & Industrial Engineering, 25*, 219–222.

Woo, D. S., & Yim, H. S. (1998). A heuristic algorithm for mean flowtime objective in flowshop scheduling. *Computers & Operations Research, 25*, 175–182.

Zhang, B. T. (1999). A Bayesian framework for evolutionary computation. *Proceedings of the International Conference on Evolutionary Computation*, 722–728.

Zhang, Q., & Mühlenbein, H. (2004). On the convergence of a class of estimation of distribution algorithms. *IEEE Transactions on Evolutionary Computation, 8*, 127–135.

Zhang, Q., Sun, J., & Tsang, E. (2005). An evolutionary algorithm with guided mutation for the maximum clique problem. *IEEE Transactions on Evolutionary Computation, 9*, 192–200.

Zhang, Q., Sun, J., Tsang, E. P. K., & Ford, J. (2004). Combination of guided local search and estimation of distribution algorithm for solving quadratic assignment problem. *Proceedings, Genetic and Evolutionary Computation Conference*, 42–48.

Zhang, Y., Li, X., & Wang, Q. (2009). Hybrid genetic algorithm for permutation flowshop scheduling problems with total flowtime minimization. *European Journal of Operational Research, 196*, 869–876.

Zhu, K. Q. (2003). Population diversity in genetic algorithm for vehicle routing problem with time windows. National University of Singapore, Technical Report.