Computers & Industrial Engineering 62 (2012) 536–545

Contents lists available at SciVerse ScienceDirect

Computers & Industrial Engineering

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

Extended artificial chromosomes genetic algorithm for permutation flowshop scheduling problems ☆

Yuh-Min Chen [a], Min-Chih Chen [a], Pei-Chann Chang [b,*], Shih-Hsin Chen [c]

[a] Institute of Manufacturing Engineering, National Cheng Kung University, Tainan 70101, Taiwan, ROC
[b] Department of Information Management, Yuan-Ze University, 135 Yuan-Tung Rd., Taoyuan 32026, Taiwan, ROC
[c] Department of Electronic Commerce Management, Nanhua University, No. 55, Sec. 1, Nanhua Rd., Zhongkeng, Dalin Township, Chiayi County 62248, Taiwan, ROC

A R T I C L E I N F O

Article history:
Received 12 March 2011
Received in revised form 29 October 2011
Accepted 1 November 2011
Available online 18 November 2011

Keywords:
Evolutionary algorithm with probabilistic models
Scheduling problems
Estimation of distribution algorithms

A B S T R A C T

In our previous researches, we proposed the artificial chromosomes with genetic algorithm (ACGA) which combines the concept of the Estimation of Distribution Algorithms (EDAs) with genetic algorithms (GAs). The probabilistic model used in the ACGA is the univariate probabilistic model. We showed that ACGA is effective in solving the scheduling problems. In this paper, a new probabilistic model is proposed to capture the variable linkages together with the univariate probabilistic model where most EDAs could use only one statistic information. This proposed algorithm is named extended artificial chromosomes with genetic algorithm (eACGA). We investigate the usefulness of the probabilistic models and to compare eACGA with several famous permutation-oriented EDAs on the benchmark instances of the permutation flowshop scheduling problems (PFSPs). eACGA yields better solution quality for makespan criterion when we use the average error ratio metric as their performance measures. In addition, eACGA is further integrated with well-known heuristic algorithms, such as NEH and variable neighborhood search (VNS) and it is denoted as $eACGA_{hybrid}$ to solve the considered problems. No matter the solution quality and the computation efficiency, the experimental results indicate that $eACGA_{hybrid}$ outperforms other known algorithms in literature. As a result, the proposed algorithms are very competitive in solving the PFSPs.

1. Introduction

In recent years, the Estimation of Distribution Algorithms (EDAs) have been one of the major evolutionary computing paradigms applied in solving combinatorial optimization problems (Ceberio, Irurozki, & Lozano, 2012; Larrañaga & Lozano, 2002). EDAs offer promising solutions that explicitly build a probabilistic model based on the information extracted from previous searches. Instead of using the crossover and mutation operators, EDAs generate new solutions by sampling from the probabilistic model (Larrañaga & Lozano, 2002). Through the probabilistic models characterizing the solution space, EDAs are good at solving hard problems when we do not have prior knowledge of the problems.

In order to improve the performance of EDAs, some recent attempts have been made to combine EDAs with the crossover and mutation operators (Chang, Chen, & Fan, 2008; Pena et al., 2004). The main characteristic of these algorithms is that EDAs alternate with GAs in the evolutionary progress. They showed that the synergy of EDAs with genetic operators produces better solution quality. Chen, Chen, Chang, Zhang, and Chen (2010) demonstrated why the hybrid framework is superior than we use EDAs or GAs alone. The reason is that although EDAs improve the solution quality efficiently in first few runs, the loss of diversity grows very fast as more iterations are run (Branke, Lode, & Shapiro, 2007; Chen, Chang, Cheng, & Zhang, 2012; Shapiro, 2006). Because GAs explore the solution space by using the crossover and mutation genetic operators, they supply broader diversified chromosomes into the population than EDAs. By combining these two approaches, EDAs and GAs compensate for each other's disadvantages. In the research of Chang et al. (2008), artificial chromosome with genetic algorithms (ACGA) makes a good example to demonstrate the combination of EDAs with genetic operators yields better solution quality.

ACGA applied an univariate probabilistic model in the algorithm while bivariate or multivariate probabilistic models was not considered. Because the use of high order interactions in probabilistic model may generate more sophisticated results among the variable interactions (Bengoetxea, Larrañaga, Bloch, Perchant, & Boeres, 2002; Bosman & Thierens, 2001; Pelikan, Tsutsui, & Kalapala, 2007), ACGA may not capture the problem structure well when there exists variable interactions. Given that the scheduling

☆ This manuscript was processed by Area Editor T.C. Edwin Cheng.
* Corresponding author. Tel.: +886 3 4638800.
E-mail addresses: solo535353@gmail.com (Y.-M. Chen), iepchang@saturn.yzu.edu.tw (P.-C. Chang), wordstar.chen@gmail.com (S.-H. Chen).

problems have the variable interactions in nature, this paper attempts to integrate higher order probabilistic models with ACGA to enhance the solution quality.

We extend our previous study in ACGA and propose an extended artificial chromosome with genetic algorithms (eACGA) to deal with the intractable combinatorial optimization problems by using both the univariate and the bi-variate statistic information. To be more specifically, the two statistic models are based on a frequency information of a variable and the linkage between two variables, respectively. It is the major difference between eACGA with previous EDAs because most EDAs could use only one statistic model. Moreover, in order to provide a comparable result, NEH, a well-known heuristic approach for flowshop scheduling problems, and a famous local search algorithm, i.e., variable neighborhood search (VNS), are both employed in the eACGA. The resultant algorithm is called $eACGA_{hybrid}$. To evaluate the performance of the the proposed algorithms, they are compared with several famous permutation-oriented EDAs and some EAs in literature.

Contributions of this work: The probabilistic model used in EDAs directly impacts its performance (Lozano, Larranaga, Inza, & Bengoetxea, 2006). The importance of this research is to propose new probabilistic models which consider both univariate and bivariate statistic information. This characteristic is distinguished from previous EDAs because most EDAs employed just one statistic information. These probabilistic models could represent better individual information for EDAs. There are two researches based on similar idea (Jarboui, Eddaly, & Siarry, 2009; Pan & Ruiz, 2012), however there are some major differences among these three approaches. The probabilistic model and learning strategy of eACGA are quite different from theirs. Finally, according to the latest review of EDAs in permutation-based combinatorial problems (Ceberio et al., 2012), EDAs have not been extensively developed in this direction. This work is thus of importance in the area of EDAs when it comes to the permutation problems.

The rest of the paper is outlined as follows: Section 2 is the problem definition of the PFSPs for the makespan criterion, Section 3 describes a detailed explanation of the eACGA, and Section 4 presents experimental results on the performance of the proposed algorithms in treating the permutation flowshop scheduling problem to minimize the makespan. Finally, Section 5 draws a conclusion of this research.

2. Problem statements

The PFSPs to minimize the makespan can be defined as follows:

Suppose there are n jobs and m machines. Let $p(i,j)$, $1 \leqslant i \leqslant n$, $1 \leqslant j \leqslant m$, be the processing time of job i on machine j and $\pi = (\pi_1, \ldots, \pi_n)$ be a job permutation (i.e., processing order of the jobs). Then the completion times $C(\pi_i, j)$ are calculated as follows:

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

$$C(\pi_i, 1) = C(\pi_{i-1}, 1) + p(\pi_i, 1) \quad \text{for } i = 2, \ldots, n \tag{2}$$

$$C(\pi_1, j) = C(\pi_1, j-1) + p(\pi_1, j) \quad \text{for } j = 2, \ldots, m \tag{3}$$

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

The makespan is

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

The objective is to find a permutation π^* that minimizes $C_{\max}(\pi)$. The compared algorithms are used to solve the PFSPs for the makespan criterion. We collect the average error ratio (ER) because it is often used to evaluate the performance of algorithms applied to deal with the PFSPs. The error ratio of a solution X_i generated by an algorithm is calculated as follows:

$$ER_i = \frac{C_{max}(X_i) - U_i}{U_i}, \tag{6}$$

Where U_i is the makespan value of the best known or optimal solution for PFSPs. When different researches applied the U_i, they should employ the same best found objective value. Thus it provides a baseline for comparisons.

3. Methodology

The univariate probabilistic model used by ACGA assumed that there are no dependencies between/among variables. However, some researches pointed out when variable interactions exist, EDAs may employ the bivariate or even the multivariate probabilistic models (Bengoetxea et al., 2002; Bosman & Thierens, 2001; Pelikan et al., 2007). Because the scheduling problems have the variable interactions in nature, the motivation is to let the proposed algorithm integrate the bi-variate probabilistic model to enhance the solution quality.

In order to capture the information of variable interactions, we have taken the bivariate probability model into consideration. That is, eACGA extends from the ACGA, which collects not only the order information of the job in the sequence but also the variable interactions of the jobs. As a result, eACGA could extract the parental information by using the univariate probability model and bivariate probability model simultaneously.

In addition to the bivariate probabilistic model, some heuristic approaches are also quite helpful when dealing with the scheduling or the sequencing problems. First of all, NEH, a well-known heuristic in flowshop scheduling problems with the minimization of makespan, is utilized to generate a single chromosome in the population initialization stage.

Secondly, a neighborhood search method is applied in eACGA to further improve the solution quality. Local search provides better exploiting information for realistic problems because it generally can be computed very fast in solving large problems (Nareyek, 2001), The variable neighborhood search (VNS) is very useful as mentioned by Mladenovic and Hansen (1997), Hansen and Mladenovi (2001) and Jarboui et al. (2008), therefore it is employed to further improve the solution quality of eACGA. The new algorithm, i.e., eACGA combined with NEH and VNS, is named $eACGA_{hybrid}$ in short.

The following Section 3.1 explains the proposed algorithm in detail. Because the univariate and bivariate are the cores of EDAs, they are further explained in Section 3.2. After that, Section 3.3 demonstrates a 5-job instance to explain how to generate an offspring by the probabilistic model. Finally, we illustrate the procedures of the VNS in the last subsection.

3.1. The procedures of eACGA

The major procedures of eACGA include initializing population, selecting of better chromosomes, and deciding whether EDAs or genetic operators are run. In addition, VNS is an optional method which may improve a new solution in the search. After that, a replacement strategy is used and the stopping criterion is tested. The framework of eACGA is depicted in Fig. 1.

We describe the methods in the following steps.

Step 1: Initialization
We initialize the population consisting of a number of chromosomes. A chromosome represents a processing sequence for the scheduling problem. Each chromosome is generated randomly. In order to improve results, some researches employ heuristic algorithms to generate better initial solutions. For example, NEH is considered to

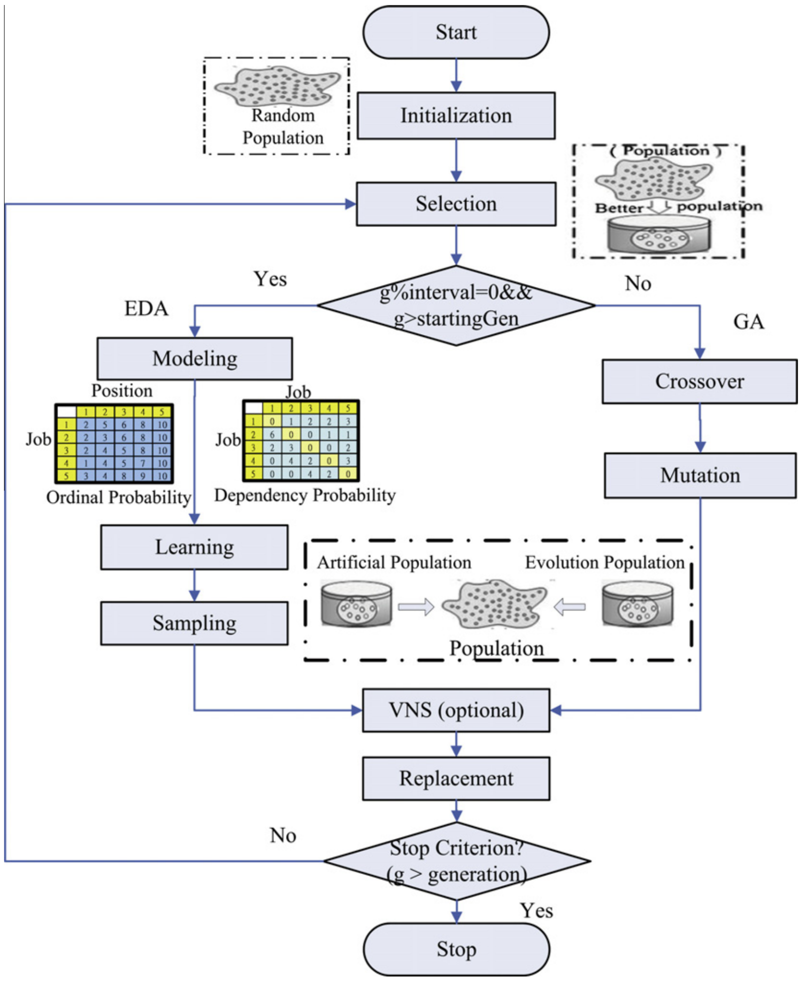


Fig. 1. The framework of eACGA.

generate a single chromosome in eACGA. On the other hand, we also have to initialize the probabilistic models used in eACGA. Regardless whether the univariate or the bivariate probabilistic model is used, each $P_{i[i]}(t)$ is initialized to be $\frac{1}{n}$, where n is the number of jobs and $P_{i[i]}(t)$ is the probability of job i in position $[i]$ in a promising solution.

Step 2: Selection
Evolutionary algorithms attempt to select more feasible solutions that corresponds to their objective values. The selection operator chooses better chromosomes to be survived. For the purpose of simplicity, the binary tournament operator is employed, which selects the better chromosomes with lower objective values in this minimization problem.

Step 3: Decision
Two parameters control whether the EDAs or GAs are run, *startingGen* and *interval*. The first parameter *startingGen* is to determine the starting time of generating artificial chromosomes. The main reason is that the probabilistic model should be only applied to generate better chromosomes when the searching process achieves a more stable state. The other important parameter *interval* sets the period during which the artificial chromosomes are generated. As a result, the algorithm alternates EDAs and genetic operators in the whole evolutionary progress. When EDAs are executed, they go through the process of constructing the probabilistic models, learning of parental distribution, and then sampling new offsprings from probabilistic models. On the other hand, the algorithm produces offsprings by using the elite GAs.

Step 4: Variations
The whole procedure is described from Step 4.1.1 to Step 4.1.3. On the other hand, genetic operators contain the crossover and mutation operator which are described in Step 4.2.1 and Step 4.2.2, respectively.

Step 4.1: eACGA segment

Step 4.1.1: Modeling
The univariate probabilistic model and the bivariate probabilistic models are built while we run the EDAs. The former one represents the importance of the order of the jobs in the sequence. The bivariate probabilistic model illustrates the building blocks in the populations. More step by step details will be presented in Section 3.2.

Step 4.1.2: Learning
As in PBIL (Baluja & Davies, 1998), we update the two probabilistic models in an incremental learning method. Furthermore, the learning rate determines the importance of the current and historical probability information. The probability learning models of the univariate and bivariate probabilistic models are shown in Eqs. (11) and (12), respectively.

Step 4.1.3: Sampling
Now that the two probabilistic models have been established, the actual procedure implemented in the optimization algorithm needs to be specified. The goal is to devise a strategy to form the offspring populations reflecting the two probabilistic models. For each position in the sequence of a new individual, first we select a job randomly as the first position, then according to the multiplication of two probabilistic models, proportional selection fills out the other sequence of a new individual.

Step 4.2: Elite GA segment

Step 4.2.1: Crossover
This study applies the two-point central crossover operator Muruta and Ishibuchi (1994) to mate two randomly selected chromosomes. Crossover rate (P_c) decides whether the chromosome is mated with others.

Step 4.2.2: Mutation
Mutations occur inside the chromosome if a random probability value is lower than the mutation rate (P_m). The swap mutation operator is used in our experiments. When we decide to do the mutation, the genes of the two random positions are swapped.

Step 5: Variable neighborhood search (optional)
Fig. 1 also illustrates that the optional procedure of a VNS could be executed in eACGA. A probability variable *Penh* controls when the VNS starts to improve a new solution. More information for VNS is described in Section 3.4.

Step 6: Replacement
In order to improve the population quality and maintain the population diversity, the better offspring replaces the worst individual in the parent population. Furthermore, the offspring must be different from any one of the parent population.

3.2. Establishing probabilistic models

Because eACGA employs both univariate and bi-variate probabilistic models while most EDAs did not apply more than one model, we define them in this section. Moreover, we explain how to utilize the two statistics in eACGA.

To build the probabilistic models, a set of M better chromosomes X^1, X^2, ..., and X^M are selected at the current generation t. In principle, any selection method such as proportional selection and tournament selection can be used for this purpose. For the sake of simplicity, we adopt the 2-tournament selection in our method. Then, $X^k_{i[i]}$ is a binary variable in chromosome k (See Eq. (7)) which is used to define the univariate model later.

$$X^k_{i[i]} = \begin{cases} 1 & \text{if job } i \text{ before or at position}[i] \\ 0 & \text{Otherwise} \end{cases}, \quad i=1,\ldots,n;\ k=1,\ldots,M \tag{7}$$

where the domain of the position $[i]$ is also from $i = 1, \ldots, n$ and n is the number of jobs. After we sum up the the statistic information from all M chromosomes to the $X^k_{i[i]}$, we obtain the univariate model $\phi_{i[i]}(t)$ in Eq. (8) which represents the number of times that job i before or at position $[i]$ at the current generation t. This univariate model shows the importance of the jobs in the sequence and it was also used in Jarboui et al. (2009) and Pan and Ruiz (2012).

$$\phi_{i[i]}(t) = \sum_{k=1}^{M} X^k_{i[i]}, i = 1, \ldots, n \tag{8}$$

When it comes to the bi-variate probabilistic model, it is the same that we define a new binary variable $\upsilon^k_{i'i}(t)$ in Eq. (9). This variable indicates whether job i is immediately after the job i' in chromosome k.

$$\upsilon^k_{i'i}(t) = \begin{cases} 1 & \text{if job } i \text{ is next to the job } i' \\ 0 & \text{Otherwise} \end{cases}, \quad i=1,\ldots,n;\ k=1,\ldots,M \tag{9}$$

where $i \neq i'$. After we summarize the statistic information of $\upsilon^k_{i'i}(t)$ from the M chromosomes, the bi-variate statistic information $\psi_{i'l}(t)$ could be obtained in Eq. (10). $\psi_{i'i}(t)$ indicates the number of times that job i immediately after the job i'.

$$\psi_{i'i}(t) = \sum_{k=1}^{M} \upsilon^k_{i'i}, \quad i=1,\ldots,n,\ i \neq i',\ k=1,\ldots,M \tag{10}$$

$\phi_{i[i]}(t)$ and $\psi_{i'i}(t)$ model (in Eqs. (8) and (10)) are built so far. According to the Step 4.1.2 about the updating algorithm by PBIL, the two statistics with learning can continue to modify the search space and then improve the performance. Eqs. (11) and (12) set the learning approach. In this research, two learning rates, λ_ϕ and λ_ψ, are decided by Design-of-Experiment (DOE).

$$\phi_{i[i]}(t) = \phi_{i[i]}(t) \times (1.0 - \lambda_\phi) + \phi_{i[i]}(t-1) \times \lambda_\phi, \lambda_\phi \in (0,1) \tag{11}$$

$$\psi_{i'i}(t) = \psi_{i'i}(t) \times (1.0 - \lambda_\psi) + \psi_{i'i}(t-1) \times \lambda_\psi, \lambda_\psi \in (0,1) \tag{12}$$

After $\phi_{i[i]}(t)$ and $\psi_{i'i}(t)$ learn from previous search, we consider how to form the probabilistic models which utilize the both statistics. Let $P_{i[i]}(t)$ be the probability value of the job i at position $[i]$. This research likes to select a job i which has higher probability value than other jobs when the univariate and bi-variate statistic information are used. To do this, $\phi_{i[i]}(t)$ is multiplied by $\psi_{i'i}(t)$ which is proportioned to the summarized probability values of all unscheduled jobs that could be assigned at positions $[i]$. In addition, it is noticeable that when we select a job at the first position, $\psi_{i'i}(t)$ could be zero in the most cases while only few $\psi_{i'i}(t) > 0$. It causes only few jobs could be selected at the first position for producing offspring. The population diversity is decreased easily in this case (Pan & Ruiz, 2012). As a result, Pan and Ruiz (2012) use the univariate model to select a job at the first position.

It is reasonable to use the univariate model at $[i] = 0$; however, this research suggests that a random value with the uniform distribution is used. We investigate the performance difference of the two approaches in our experiment section. As a result, $P_{i[i]}(t)$ is defined as follows.

$$P_{i[i]}(t) = \begin{cases} U(0,1) & [i] = 1 \\ \phi_{i[i]}(t) \times \psi_{i'i}(t) / \sum_{l \in \Omega} (\phi_{l[i]}(t) \times \psi_{i'l}(t)) & [i] = 2,3,\ldots,n \end{cases} \tag{13}$$

where $i = 1, \ldots, n$ and Ω is the set of the unscheduled jobs. A proportional selection method is used to select a job from Ω and then set it to the position $[i]$. The following pseudo code presents the assignment procedures.

S: A set of shuffled sequence which determines the sequence of each job being assigned a position.
Ω: The set of unassigned jobs.
J: The set of assigned jobs. J is empty in the beginning.
θ: A random probability is drawn from U(0,1).

```
    $i$: A selected job by proportional selection
    $k$: The element index of the set $S$
1: $S \leftarrow$ shuffled the job number $[1 \ldots n]$
2: $J \leftarrow \Phi$
3: **while** $k \neq \Phi$ **do**
4:     $\theta \leftarrow U(0,1)$
5:     Select a job $i$ satisfies $\theta \leqslant P_{i[i]}(t)$, where $i \in \Omega$
6:     $J(k) \leftarrow i$
7:     $\Omega \leftarrow \Omega \backslash i$
8:     $S \leftarrow S \backslash k$
9: **end while**
```

To conclude the characteristics of eACGA with previous EDAs, eACGA employs both univariate and bi-variate statistics while most EDAs use only one probabilistic model. Secondly, the genetic operators alternative with the probabilistic models which may get rid of the diversity loss in EDAs. When eACGA is compared with the works of Jarboui et al. (2009) and Pan and Ruiz (2012), eACGA uses the learning algorithms to update the probabilistic models. In addition, the bi-variate model presented in eACGA avoids the disruption of the similar blocks of jobs within the individuals' sequences due to the bivariate model used by Jarboui et al. (2009) only considers the blocks in the same positions. When the blocks are disrupted, the bivariate model may not work well (Ruiz, Maroto, & Alcaraz, 2006). Apart from that, our approach of setting the first position in the sequence might be better than theirs owing to the diversity concern. The comparisons of the univariate model used by Pan and Ruiz (2012) and random value approach proposed in this paper are shown in Section 4.1.

3.3. Generating offsprings by the parental distribution

To demonstrate the procedures of the eACGA, we draw a 5-jobs instance as an example. Suppose there are ten promising solutions which are selected to build the probabilistic models, as shown in Fig. 2. We accumulate location and interaction information from these chromosomes to form the univariate and bivariate probabilistic models. The right-hand side of Fig. 2 shows that there are two job 1, two job 2, two job 3, one job 4, and three job 5 on position 1 in the location-based information matrix.

In the bivariate probabilistic model, there are six times that job 2 is next to job 1, two times that job 1 is in front of job 3, and 0 occurrence of the job 4 and job 5 is right after the job 1. The zero value in the bivariate matrix forbids the jobs to be put into together in any case. Since we should avoid this condition, the zero value is corrected to $\frac{1}{n}$ where n is the number of selected chromosomes. After we have the two probabilistic models, we are going to sample new solutions.

To generate a diversified sequence, the first position does not take the probabilistic models. We suppose job 3 is selected as the first job in the sequence. Then, when we assign a job at position 2, we should take the two probabilistic models into consideration. These values doing multiplication and normalization can produce probabilistic table of position 2 as shown in Fig. 3. The corresponding probability for job 1 is 0.2939; job 2 is 0.0009; job 3 is none because it has been selected, and so on. After the assignment probability is determined, the job will be selected by a roulette wheel selection method based on the probability of each job on this position.

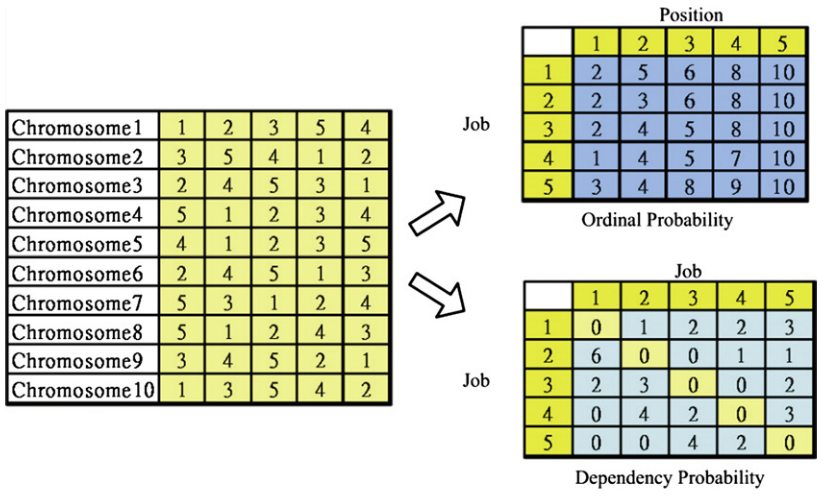
According to the probability selection method, we assume job 5 is selected, then job 5 is assigned to position 2. The position 3 is the next one to be assigned, an updated probabilistic matrix is shown in Fig. 3. Again, following the same procedure, the probability of each job in position 3 is calculated. Then, a roulette wheel selection method will select a job based on the probability of each job. Consequently, all jobs will be assigned a specific position. Finally, a new job sequence according to the ordinal and dependency matrix is generated.

3.4. VNS procedures

In order to improve the solution quality, VNS is hybridized into the searching process in this research since many researches have proved that VNS is effective in solving the PFSPs (Jarboui et al., 2009; Tasgetiren, Liang, Sevkli, & Gencyilmaz, 2007). In this paper, we have included the version of Jarboui et al. (2009) in our research. Meanwhile, the combination of the eACGA and VNS is named *eACGA*$_{hybrid}$ in this paper.

In the beginning of the VNS procedures, a VNS parameter *Penh* decides the probability to execute the VNS in the main procedure of *eACGA*$_{hybrid}$. We generate a random probability and to test whether it is less than or equal to the *Penh*. If the random value is less than or equal to the *Penh*, VNS procedure is thus executed and a current best solution x_{best} is selected to do the perturbation.

x_{best}: A selected elite chromosome.
x_1: We change the neighborhood structure of x_{best}.
x_2: A local optimal improved by a swap local search operator.



Chromosome1	1	2	3	5	4
Chromosome2	3	5	4	1	2
Chromosome3	2	4	5	3	1
Chromosome4	5	1	2	3	4
Chromosome5	4	1	2	3	5
Chromosome6	2	4	5	1	3
Chromosome7	5	3	1	2	4
Chromosome8	5	1	2	4	3
Chromosome9	3	4	5	2	1
Chromosome10	1	3	5	4	2

Position

Job	1	2	3	4	5
1	2	5	6	8	10
2	2	3	6	8	10
3	2	4	5	8	10
4	1	4	5	7	10
5	3	4	8	9	10

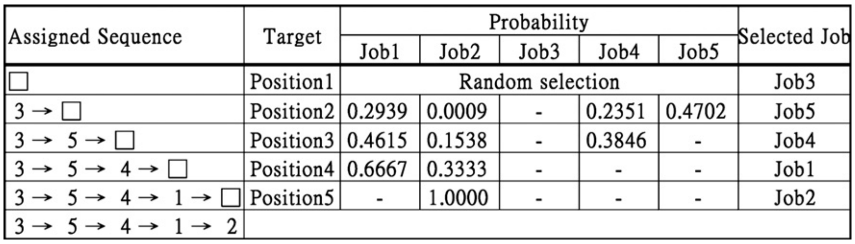
Ordinal Probability

Job

Job	1	2	3	4	5
1	0	1	2	2	3
2	6	0	0	1	1
3	2	3	0	0	2
4	0	4	2	0	3
5	0	0	4	2	0

Dependency Probability

Fig. 2. To collect gene and interaction information and to convert into location and interaction matrix.



Assigned Sequence	Target	Probability					Selected Job
		Job1	Job2	Job3	Job4	Job5	
□	Position1	Random selection					Job3
3 → □	Position2	0.2939	0.0009	-	0.2351	0.4702	Job5
3 → 5 → □	Position3	0.4615	0.1538	-	0.3846	-	Job4
3 → 5 → 4 → □	Position4	0.6667	0.3333	-	-	-	Job1
3 → 5 → 4 → 1 → □	Position5	-	1.0000	-	-	-	Job2
3 → 5 → 4 → 1 → 2							

Fig. 3. Example of an artificial chromosome.

x'_2: A local optimal generated by an insertion local search operator.
$F(x_{best})$: The objective value of the solution x_{best}.
k: The current number of VNS iteration
k_{max}: The maximum number of VNS iteration

```
1: $k \leftarrow 1$
2: while $k < k_{max}$ do
3:    $x_1 \leftarrow$ generate a neighborhood solution of $x_{best}$
4:    $x_2 \leftarrow$ swapLocalSearch ($x_1$)
5:    $x'_2 \leftarrow$ insertionLocalSearch ($x_2$)
6:    if $F(x'_2) < F(x_{best})$ then
7:        $x_{best} \leftarrow x'_2$
8:        $k \leftarrow 1$
9:    else
10:       $k \leftarrow k + 1$
11:   end if
12: end while
```

The number of k_{max} is set as the stopping criterion of the VNS. Because the number of k_{max} is dependent on the benchmark instance, we determine this parameter by design-of-experiment. In the insert loop of VNS, a new solution is created by the shaking procedure. The neighborhood structure comprises steps of exchange, insert, and exchange to variate a current best solution. By using this method suggested by Sevkli and Aydin (2009), we thus create and further improve the new solution x_1 by a swap local search. The resultant solution of swap local search is x_2 that might be improved by a swap local search. Then, an insertion local search acts on the solution x_2 so that it generates a new solution x'_2. The final step is to compare the fitness of the solution x'_2 with the current best solution x_{best}. If the new solution x'_2 is better than x_{best}, it replaces the x_{best} and k is reset to one. Otherwise, k is increased by one. Through the systematic exploration and exploitation, VNS could improve the performance of the proposed $eACGA_{hybrid}$.

4. Computational results

In order to evaluate the performance of the eACGA and eACGA-$_{Hybrid}$, we conducted extensive experiments to test them on permutation flowshop scheduling problems in minimizing makespan. They were implemented by Java 1.6 SDK on Windows 2003 server with Intel Xeon 3.2 GHz CPU. We selected the probabilistic models used in the proposed algorithm by Design-of-Experiments (DOE) in Section 4.1. Then, eACGA was compared with others in literature. The comparisons are classified into two categories. The first category is the EDAs approaches where Ceberio et al. (2012) examined numerous famous permutation-oriented EDAs to test the flowshop scheduling problems in Section 4.2. The other type is EAs and hybrid algorithms in Section 4.3. Both experiments used the well-known flowshop instances given by Taillard (1993). These experiment results are shown in the following sections.

Table 1
Parameter settings of eACGA.

Method	Settings
eACGA	Starting generation = 0.5 × (total generations) Interval = 0.02 × (total generations) Crossover rate = 0.9 Mutation rate = 0.4 ordinal probability learning rate = 0.7 dependent probability learning rate = 0.1 Population size = 400
Common settings	Elitism rate = 10%

Table 2
ANOVA table of the model selection.

Source	DF	SS	Mean square	*F* value	*P* value
Instance	109	76,106,350,523	698,223,399	980,292	<.0001
methods	1	54,662	54,662	76.75	<.0001
Instance ∗ method	109	246,287	2260	3.17	<.0001
Error	6380	4,544,222	712		
Corrected total	6599	76,111,195,694			

4.1. DOE on the selection of the probabilistic models

Section 3.2 discussed two possible alternatives which could be used in eACGA. The first approach is to generate the first job by an univariate probabilistic model and the second one is to generate the probability value by random. The two approaches in eACGA were test on the 110 Taillard flowshop instances, including the 20 jobs with 5 machines to 200 jobs with 20 machines. In addition, they took the same parameter settings (See Table 1) and use a stopping criterion set by Tasgetiren et al. (2007) was to examine $500 \times 2 \times n$ solutions where n is the number of jobs. The ANOVA results was shown in Table 2.

In this ANOVA table, the source indicates factors and combinations of factors. In our case, *Instance* and *Method* are factors. *DF* represents the degree of freedom and *SS* is the sum of squares. The mean square is equal to *SS* divided by *DF*. If the *Pr*-value of a factor (source) is less than 0.05, it means that there is a significant difference in this factor (Montgomery, 2008). Due to *Pr*-value of the methods is less than 0.0001, there exists a significant difference between the two methods. We further conducted the Duncan grouping test to differentiate the performance of these algorithms (See Table 3).

In Table 3, Mean is the average value and *N* is the number of the observations. If two algorithms share the same alphabet (i.e., they are in the same group), there is no significant difference between them. Otherwise they are significantly different (Montgomery, 2008). It is an evident from the Duncan grouping test that the two approaches were significantly different in their performance and the generating probability value by random performed

Table 3
Duncan grouping test at model selection (Univariate: Univariate model is used. Random: Generating probability value by random).

Duncan grouping	Mean	*N*	Methods
A	4997.4633	3300	Univariate
B	4991.7076	3300	Random

Table 4
The parameter settings for the permutation-oriented EDAs and eACGA.

Algorithm	Parameter setting value
Common setting	Population size = $10 \times n$ Stopping criterion = $100 \times n$(maximum number of generations)

significantly better than the univariate model is used when we like to set the first job in the sequence.

These results revealed the univariate model might be influenced by some salient genes which dominated the first position when the population was converged during the evolutionary progress (Chang et al., 2008). The probabilistic models could not sample diversified offspring and EDAs could be trapped in local optimal. As a result, the random value is suggested when we design an EDAs which considers the univariate and bi-variate probabilistic models when we select a job at the first position. eACGA and eACGA$_{Hybrid}$ applied this approach in the following experiments.

4.2. Experiment results of permutation-oriented EDAs

Many well-known EDAs selected in Ceberio et al. (2012) and they were applied for comparison. These algorithms were UMDA (Larrañaga, Etxeberria, Lozano, & Peña, 2000), MIMIC and EBNA$_{BIC}$ in Bengoetxea et al. (2002), Tree (Pelikan et al., 2007), UMDA$_c$ and EGNA$_{ee}$ in Larrañaga and Lozano (2002), IDEA–ICE (Bosman & Thierens, 2001), EHBSA$_{WT}$ (Tsutsui, 2009), EHBSA$_{WO}$ (Tsutsui & Miki, 2002), NHBSA$_{WT}$ (Tsutsui, 2006), NHBSA$_{WO}$ (Tsutsui, 2006), and REDA$_{UMDA}$ and REDA$_{MIMIC}$ in Romero and Larrañaga (2009).

The four instance sets of the flowshop benchmarks were collected, such as the tai20 × 5, tai20 × 10, tai50 × 10, tai50 × 20, and tai100 × 20. In addition, the first six instances from each set were used. So the amount of the 24 instances were employed to test the EDAs. To make a fair comparison, all the EDAs (include the proposed eACGA) used the same population size and generations. The following Table 4 is the parameter settings of these EDAs. eACGA remain applied the same settings (except the population size is changed to $10 \times n$) in Table 1.

These 14 algorithms ran 10 times on these instances and then to collect the average error ratio (*ER*). The best known (or optimal solution) used in Eq. (6) for Taillard instances was the last version in year 2005. The complete results of all EDAs were listed in Table 5.

Through the 24 instances in Table 5, the lowest average error ratio in each instance was marked in boldface. The results presented that eACGA got 16 lowest average error ratio in the 24 instances. EHBSA$_{WT}$ performed well out of the 7 instances and NHBSA$_{WT}$ got only one instance that outperformed others. In addition, the total average error ratio of eACGA which was the lowest value among all the permutation-oriented EDAs. The results showed eACGA is quite competitive when it was compared with existing permutation-oriented EDAs.

It is interesting for eACGA to compare the performance with them. There are two major differences between eACGA and these EDAs. The first reason could be that pure EDAs may encounter the problem of diversity loss during the evolutionary progress (Branke et al., 2007; Chen et al., 2010; Shapiro, 2006). Diversity loss caused the probabilistic model no longer generating diversified offsprings. Due to eACGA gains the diversified population from the genetic operators, it enables the proposed algorithm to generate better solutions without staying at the local optimal. Secondly, eACGA considered the univariate together with bi-variate statistics while most EDAs may just utilize one probabilistic model. It is thus important to take the diversity into account and using not only one statistic when we design a new EDAs.

Because it is not sufficient that we just compared eACGA with some existing EDAs, the proposed algorithms (eACGA and *eACGA$_{Hybrid}$*) were further compared with some meta-heuristics in the next section.

Table 5
Average error ratio evaluation of the permutation-oriented EDAs and eACGA to test on Taillard instances.

	UMDA	MIMIC	EBNA$_{BIC}$	TREE	UMDA$_c$	EGNA$_{ee}$	IDEA–ICE	EHBSA$_{WT}$	EHBSA$_{WO}$	NHBSA$_{WT}$	NHBSA$_{WO}$	REDA$_{UMDA}$	REDA$_{MIMIC}$	eACGA
ta001	1.16	1.55	1.12	1.64	4.68	4.08	1.96	**0.30**	1.41	1.27	1.49	1.49	2.79	1.16
ta002	0.96	0.67	1.40	1.16	3.90	3.02	0.90	**0.05**	0.49	0.27	0.31	1.21	3.73	0.13
ta003	3.95	3.36	3.27	4.18	16.47	15.58	5.06	1.30	3.04	0.71	1.23	5.26	14.14	**0.57**
ta004	1.29	2.01	1.28	3.74	13.03	12.53	4.83	**0.19**	1.81	0.22	1.17	4.18	10.46	0.56
ta005	1.26	1.22	1.33	1.86	9.15	8.87	3.18	0.51	1.17	0.96	1.17	2.78	5.42	**0.40**
ta006	1.28	1.93	1.26	2.81	12.68	11.56	4.45	**0.00**	1.26	0.53	1.13	3.22	8.05	1.13
ta011	2.47	2.78	3.06	4.92	14.83	14.02	6.06	0.53	1.52	0.58	1.11	5.92	7.88	**0.46**
ta012	3.30	3.41	2.77	5.65	13.90	13.12	6.12	0.88	3.09	0.81	1.16	6.36	8.83	**0.50**
ta013	4.55	3.33	4.06	5.06	13.29	10.52	6.66	1.10	2.08	1.18	1.77	7.27	6.76	**0.90**
ta014	2.81	2.47	3.17	5.39	16.26	16.30	6.31	**0.37**	2.27	0.65	1.26	7.22	11.31	0.76
ta015	3.47	4.16	3.60	5.31	16.94	17.35	7.07	**0.44**	0.99	0.58	1.23	5.88	17.28	0.67
ta016	2.96	2.71	2.63	4.52	17.22	16.11	5.66	**0.55**	1.56	0.73	1.32	3.78	13.80	0.63
ta041	5.37	4.43	6.41	8.09	17.59	16.57	8.50	3.50	9.18	3.74	4.51	11.56	12.95	**2.97**
ta042	5.14	5.02	4.63	8.35	18.32	18.33	9.19	3.52	8.97	3.88	4.98	11.42	16.47	**2.95**
ta043	4.94	5.07	4.89	8.59	19.65	18.90	10.69	4.16	10.68	4.29	4.25	12.63	14.92	**3.18**
ta044	4.34	2.61	4.09	5.07	14.05	11.46	7.19	1.81	7.20	1.72	1.66	8.70	8.09	**1.13**
ta045	6.30	4.00	6.01	7.92	17.00	12.94	8.43	3.72	9.34	3.58	**2.42**	11.26	11.87	2.75
ta046	5.13	3.85	6.26	6.61	16.41	14.70	6.88	2.69	7.47	2.90	4.22	10.77	10.63	**2.36**
ta081	13.13	6.21	13.28	10.72	21.20	20.41	12.51	8.18	14.51	7.84	8.80	20.88	20.71	**5.00**
ta082	11.38	4.18	11.53	8.60	19.19	17.66	11.55	6.51	13.05	5.84	6.35	19.80	19.56	**3.18**
ta083	10.83	4.82	10.82	8.45	18.27	17.27	10.59	6.46	12.26	5.81	6.44	18.99	17.81	**3.25**
ta084	10.73	4.03	11.08	7.42	17.77	15.54	9.79	5.82	11.98	5.15	5.32	17.93	17.71	**2.85**
ta085	11.95	4.82	11.68	8.84	18.09	16.48	11.07	6.80	12.45	6.52	7.40	18.89	18.21	**4.04**
ta086	10.31	5.50	11.13	9.24	18.61	18.36	11.47	6.81	12.43	6.68	7.45	19.17	18.68	**3.82**
Avg.	5.38	3.51	5.45	6.01	15.35	14.24	7.34	2.76	6.26	2.77	3.26	9.86	12.42	**1.89**

Bold values represent the best performance of each instance among all algorithms.

Table 6
Parameter settings of the implemented algorithms for solving permutation optimization problems: SGA, ACGA, eACGA and $eACGA_{Hybrid}$.

Method	Settings
SGA	Crossover rate = 0.9 Mutation rate = 0.3 Population size = 500
ACGA	Starting generation = 0.7 × (total generations) Interval = 0.1 × (total generations) Crossover rate = 0.9 Mutation rate = 0.5 Population size = 500
eACGA and $eACGA_{Hybrid}$	Starting generation = 0.5 × (total generations) Interval = 0.02 × (total generations) Crossover rate = 0.9 Mutation rate = 0.4 Ordinal probability learning rate = 0.7 Dependent probability learning rate = 0.1 Population size = 400
Common settings	Elitism rate = 10%

4.3. Empirical results of evolutionary algorithms and hybrid algorithms

Some EAs and hybrid algorithms were tested on the 120 Taillard flowshop instances, including the 20 jobs with 5 machines to 500 jobs with 20 machines. Each instance was replicated 30 times on the compared algorithms. They were the SGA (Chang et al., 2008), ACGA (Chang et al., 2008), PSO_{SPV} (Tasgetiren et al., 2007), PSO_{VNS} (Tasgetiren et al., 2007), $H-CPSO$ (Jarboui et al., 2008), and DDE (Pan, Tasgetiren, & Liang, 2008). The brief introduction of them were discussed as follows:

- SGA: A standard genetic algorithm with elitism strategy. The genetic operators include binary tournament selection, two-point central operator and swap mutation operator. During the selection stage, 10% of elites in the population are reserved to the next generation.
- ACGA: This is our previously developed approach that alternates a probabilistic model with genetic operators which are also used in SGA. Under the hybrid framework, both global and location information are used. The above probabilistic model used in ACGA does not consider the dependencies between/among variables.
- PSO_{SPV} and PSO_{VNS}: Particle Swarm Optimization is designed to solve continuous problems while Tasgetiren et al. (2007) proposed the Smallest Position Value rule which enables the PSO to solve the PFSPs. In the same research, they combine particle swarm optimization and a very efficient local search called Variable Neighborhood Search to increase the solution quality. VNS indeed improved the performance of PSO_{SPV}. They used C programming language to code these two algorithms on an Intel Pentium IV 2.6 GHZ computer.
- $H-CPSO$: It combines particle swarm optimization and simulated annealing approach as the local search operator. They coded this algorithm in C++ and $H-CPSO$ was ran on a PC with Intel Pentium IV 3.2 GHZ.
- DDE: DDE utilizes a Discrete Differential Evolution, and a problem-specific NEH heuristic. This algorithm was coded in Visual C++ and tested on a computer with Intel Pentium IV 3.0 GHZ. We used the data from Pan et al. (2008) for comparison.

In order to evaluate the performance, a stopping criterion set by Tasgetiren et al. (2007) was to examine $500 \times 2 \times n$ solutions where n is the number of jobs. It ensures we made the fair-comparison among these algorithms. The parameter settings of SGA, ACGA, eACGA and $eACGA_{Hybrid}$ were given in Table 6.

When we calculated the average error ratio, the best known used in Eq. (7) for Taillard instances in April 2004. Table 7 showed the statistics of the average ER values of all the algorithms on all the 120 test instances. When eACGA was compared with other EAs without using local search, the results showed that eACGA yielded much lower average error ratio than those of the other four EAs. It indicated that the bivariate probability model in eACGA improves the performance greatly while ACGA only takes the univariate probabilistic model. The average error ratio of eACGA is also better than SGA, PSO_{SPV} and DDE which were recently proposed EAs in literature. In the group of using the local search, we found the average error ratio of $eACGA_{Hybrid}$ was 5.3 times lower than that of eACGA when VNS were applied across the 120 instances. The performance of the $eACGA_{Hybrid}$ algorithm was promising since it was very competitive against PSO_{VNS} and $H-CPSO$.

We further investigated the CPU time of these algorithms (See Table 8). The SGA used less CPU time than eACGA and ACGA because it requires a linear time to create new solution, whereas artificial chromosome requires $O(n^2)$ time. ACGA is faster than eACGA because the interval of artificial chromosome embedding in SGA is greater and eACGA needs a bi-variate probabilistic model. Additionally, the average CPU time of $eACGA_{Hybrid}$ was 4.45 times higher than eACGA in 120 instances. However, it is still worthwhile

Table 7
Average error ratio of EAs and hybrid algorithms tested on Taillard instances for makespan criterion. Since the experiments of PSO_{SPV}, PSO_{VNS} and $H-CPSO$ did not run the $tai500 \times 20$ instances, we employe notation "–" to present empty data.

$n \times m$	EAs without local search					EAs with local search		
	PSO_{SPV}	SGA	ACGA	DDE	eACGA	PSO_{VNS}	H-CPSO	$eACGA_{Hybrid}$
20 × 5	1.75	1.05	1.08	0.46	0.93	0.03	0.00	0.01
20 × 10	3.25	1.58	1.72	0.93	1.36	0.02	0.01	0.13
20 × 20	2.82	1.31	1.46	0.79	1.12	0.05	0.02	0.0
50 × 5	1.14	0.52	0.43	0.17	0.22	0.00	0.00	0.03
50 × 10	5.29	2.62	2.55	2.26	2.00	0.57	0.49	0.08
50 × 20	7.21	3.60	3.67	3.11	3.11	1.36	0.96	0.81
100 × 5	0.63	0.44	0.35	0.08	0.22	0.00	0.02	0.02
100 × 10	3.27	1.68	1.47	0.94	0.93	0.18	0.26	0.31
100 × 20	8.25	3.39	3.25	3.24	2.32	1.45	1.28	0.56
200 × 10	2.47	0.90	0.73	0.55	0.38	0.18	0.40	0.06
200 × 20	8.05	2.44	2.25	2.61	0.91	1.35	1.55	0.20
500 × 20	–	2.87	2.69	1.43	1.18	–	–	0.60
Mean1	–	1.87	1.80	1.38	1.22	–	–	0.23
Mean2	4.01	1.77	1.72	1.38	1.23	0.47	0.45	0.20

Table 8
Average CPU time of the compared algorithms. Because the experiments of PSO_{SPV}, PSO_{VNS} and $H-CPSO$ did not run the *tai*500 × 20 instances, we employ notation "–" to present empty data.

$n \times m$	EAs without local search					EAs with local search		
	PSO_{SPV}	SGA	ACGA	DDE	eACGA	PSO_{VNS}	H-CPSO	$eACGA_{Hybrid}$
20 × 5	0.2	0.4	0.4	0.0	0.4	13.5	0.9	4.8
20 × 10	0.2	0.4	0.4	0.1	0.4	26.3	7.7	10.5
20 × 20	0.3	0.4	0.5	0.1	0.4	69.3	18.8	20.1
50 × 5	1.4	1.9	2.1	0.1	2.3	2.8	31.9	12.2
50 × 10	1.6	2.1	2.3	0.4	2.5	79.8	78.0	30.2
50 × 20	2.1	2.3	2.5	1.0	2.7	168.1	145.6	70.5
100 × 5	10.4	6.8	7.0	0.3	8.8	52.6	22.5	27.7
100 × 10	10.9	7.3	7.4	0.8	9.3	211.0	229.7	55.6
100 × 20	12.8	8.1	8.2	3.6	10.2	310.8	372.0	141.6
200 × 10	81.8	26.6	27.9	3.0	41.8	191.3	315.1	101.2
200 × 20	89.8	29.4	30.8	14.6	45.6	438.7	480.3	187.3
500 × 20	–	222.3	243.4	69.8	595.7	–	–	2545.6
Mean1	–	25.7	27.7	7.8	60.0	–	–	267.3
Mean2	19.2	7.8	8.1	2.2	11.3	142.2	154.8	60.2

to incorporate VNS with eACGA because the average error ration of $eACGA_{Hybrid}$ to the eACGA was 5.3 times lower. Due to PSO_{SPV} has to sort the sequence of the real values which take $O(nlogn)$ time, this algorithm took more CPU time in the group of EAs without local search. In comparison of $eACGA_{Hybrid}$, PSO_{VNS} and $H-CPSO$ which are integrated VNS into EAs, $eACGA_{Hybrid}$ with less computational time than PSO_{VNS} and $H-CPSO$. Even their CPUs were slower than ours, $eACGA_{Hybrid}$ was more efficient than PSO_{VNS} and $H-CPSO$ after we converted these CPUs into the same computing capability. Finally, DDE used the less time even the largest instance was used. It showed this algorithms was competitive in the flowshop scheduling problems.

To conclude the experiments results in this Section, eACGA is satisfactory when it is compared with existing EDAs and other EAs according to these computational results. eACGA could be further improved when domain specific heuristic and local search operator were applied. Thus the proposed algorithms could be the state-of-art algorithms when it comes to the PFSPs for makespan criterion.

5. Conclusions and future researches

In this paper, an extended artificial chromosome genetic algorithm is proposed to deal with the NP-complete permutation flowshop scheduling problems in minimizing the makespan. Instead of using one statistic model in the most EDAs, we propose the probabilistic models applied in eACGA that extract the parental distribution of univariate and bi-variate information from the chromosomes generated in previous generations. eACGA then samples from the models to produce offsprings. The experimental results showed that eACGA performs better than numerous existing permutation-oriented EDAs and eACGA outperformed ACGA significantly. In addition, when eACGA combines with NEH heuristic and VNS, named $eACGA_{hybrid}$, it could further improve the solution quality near to optimality. eACGA performs the best when compared with other algorithms published in the literature. As a result, this algorithm is very promising and provides a new area for the researchers to explore.

In the future, since there are only a few permutation-oriented EDAs according to the latest review paper (Ceberio et al., in press), it remain needs more efforts in studying the EDAs to solve the permutation problems. eACGA or $eACGA_{hybrid}$ could be applied to deal with the scheduling problems, particularly in the setup consideration since the problem has strong variable interactions in nature. Furthermore, some efforts could be done in studying the probabilistic models which consider both univariate and bi-variate model. Finally, when research propose a new permutation-oriented EDAs, it is suggested to compare the new proposed algorithm with eACGA and that of Pan and Ruiz (2012). It could be very useful to identify the performance of their proposed algorithms.

References

Baluja, S., & Davies, S. (1998). Fast probabilistic modeling for combinatorial optimization. In *Proceedings of the fifteenth national/tenth conference on Artificial intelligence/Innovative applications of artificial intelligence table of contents* (pp. 469–476).

Bengoetxea, E., Larrañaga, P., Bloch, I., Perchant, A., & Boeres, C. (2002). Inexact graph matching by means of estimation of distribution algorithms. *Pattern Recognition, 35*(12), 2867–2880.

Bosman, P., & Thierens, D. (2001). Crossing the road to efficient idfas for permutation problems. In *A workshop within the 2001 genetic and evolutionary computation conference*. Citeseer.

Branke, J., Lode, C., & Shapiro, J. (2007). Addressing sampling errors and diversity loss in UMDA. In *Proceedings of the 9th annual conference on genetic and evolutionary computation* (pp. 515). ACM.

Ceberio, J. E., Irurozki, A. M., & Lozano, J. (2012). A review on estimation of distribution algorithms in permutation-based combinatorial optimization problems. *Progress in Artificial Intelligence*, in press. doi:10.1007/s13748-011-0005-3.

Chang, P. C., Chen, S. H., & Fan, C. Y. (2008). Mining gene structures to inject artificial chromosomes for genetic algorithm in single machine scheduling problems. *Applied Soft Computing Journal, 8*(1), 767–777.

Chen, S., Chen, M., Chang, P., Zhang, Q., & Chen, Y. (2010). Guidelines for developing effective estimation of distribution algorithms in solving single machine scheduling problems. *Expert Systems with Applications, 37*(9), 6441–6451.

Chen, S. H., Chang, P. C., Cheng, T. C. E., & Zhang, Q. (2012). A self-guided genetic algorithm for permutation flowshop scheduling problems. *Computers & Operations Research, 39*(7), 1450–1457.

Hansen, P., & Mladenovi, N. (2001). Variable neighborhood search: Principles and applications. *European Journal of Operational Research, 130*(3), 449–467.

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*(9), 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*(3), 526–538.

Larrañaga, P., Etxeberria, R., Lozano, J., & Peña, J. (2000). Optimization in continuous domains by learning and simulation of gaussian networks. In *A workshop within the 2000 genetic and evolutionary computation conference*. Citeseer.

Larrañaga, P., & Lozano, J. A. (2002). *Estimation of distribution algorithms: A new tool for evolutionary computation*. Kluwer Academic Publishers.

Lozano, J., Larranaga, P., Inza, I., & Bengoetxea, E. (2006). *Towards a new evolutionary computation: Advances in the estimation of distribution algorithms*. Springer.

Mladenovic, N., & Hansen, P. (1997). Variable neighborhood search. *Computers & Operations Research, 24*(11), 1097–1100.

Montgomery, D. (2008). *Design and analysis of experiments*. John Wiley & Sons Inc.

Muruta, T., & Ishibuchi, H., 1994. Performance evolution of genetic algorithms for flowshop scheduling problems. In *Proceedings of first IEEE international conference on evolutionary computation*.

Nareyek, A. (2001). *Constraint-based agents: An architecture for constraint-based modeling and local-search-based reasoning for planning and scheduling in open and dynamic worlds*. Springer.

Pan, Q., Tasgetiren, M., & Liang, Y. (2008). A discrete differential evolution algorithm for the permutation flowshop scheduling problem. *Computers & Industrial Engineering, 55*(4), 795–816.

Pan, Q. K., & Ruiz, R. (2012). An estimation of distribution algorithm for lot-streaming flow shop problems with setup times. *Omega, 40*(2), 166–180.

Pelikan, M., Tsutsui, S., & Kalapala, R. (2007). Dependency trees, permutations, and quadratic assignment problem. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation* (pp. 7–11). Citeseer.

Pena, J., Robles, V., Larranaga, P., Herves, V., Rosales, F., & Pérez, M. (2004). GA-EDA: Hybrid evolutionary algorithm using genetic and estimation of distribution algorithms. *Innovations in Applied Artificial Intelligence*, 361–371.

Romero, T., & Larrañaga, P. (2009). Triangulation of bayesian networks with recursive estimation of distribution algorithms. *International Journal of Approximate Reasoning, 50*(3), 472–484.

Ruiz, R., Maroto, C., & Alcaraz, J. (2006). Two new robust genetic algorithms for the flowshop scheduling problem. *Omega, 34*(5), 461–476.

Sevkli, M., & Aydin, M. (2009). Variable neighbourhood search for job shop scheduling problems. *Journal of Software, 1*(2), 34.

Shapiro, J. (2006). Diversity loss in general estimation of distribution algorithms. *Parallel Problem Solving from Nature-PPSN IX*, 92–101.

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

Tasgetiren, M., Liang, Y., 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*(3), 1930–1947.

Tsutsui, S. (2006). A comparative study of sampling methods in node histogram models with probabilistic model-building genetic algorithms. *IEEE international conference on systems, man and cybernetics, 2006. SMC'06* (Vol. 4, pp. 3132–3137). IEEE.

Tsutsui, S. (2009). Effect of using partial solutions in edge histogram sampling algorithms with different local searches. In *IEEE international conference on systems, man and cybernetics, 2009. SMC 2009* (pp. 2137–2142). IEEE.

Tsutsui, S., & Miki, M., 2002. Solving flow shop scheduling problems with probabilistic model-building genetic algorithms using edge histograms. In *Proc. of the 4th Asia–Pacific conference on simulated evolution and learning (SEAL02)*.