Applied Soft Computing 17 (2014) 167–175

Contents lists available at ScienceDirect

Applied Soft Computing

Artificial chromosomes with genetic algorithm 2 (ACGA2) for single machine scheduling problems with sequence-dependent setup times

Shih-Hsin Chen [a,*], Min-Chih Chen [b], Yeong-Cheng Liou [a]

[a] Department of Information Management, Cheng Shiu University, No. 840, Chengcing Rd., Niaosong Dist., Kaohsiung City 83347, Taiwan, ROC
[b] Institute of Manufacturing Engineering, National Cheng Kung University, Tainan 70101, Taiwan, ROC

ARTICLE INFO

Article history:
Received 7 April 2011
Received in revised form
18 December 2013
Accepted 22 December 2013
Available online 18 January 2014

Keywords:
ACGA
Bi-variate EDAs
Scheduling problems
Sequence-dependent setup times
Common due-date
Variable neighborhood search

ABSTRACT

Artificial chromosomes with genetic algorithm (ACGA) is one of the latest versions of the estimation of distribution algorithms (EDAs). This algorithm has already been applied successfully to solve different kinds of scheduling problems. However, due to the fact that its probabilistic model does not consider variable interactions, ACGA may not perform well in some scheduling problems, particularly if sequence-dependent setup times are considered. This is due to the fact that the previous job will influence the processing time of the next job. Simply capturing ordinal information from the parental distribution is not sufficient for a probabilistic model. As a result, this paper proposes a bi-variate probabilistic model to add into the ACGA. This new algorithm is called the ACGA2 and is used to solve single machine scheduling problems with sequence-dependent setup times in a common due-date environment. A theoretical analysis is given in this paper. Some heuristics and local search algorithm variable neighborhood search (VNS) are also employed in the ACGA2. The results indicate that the average error ratio of this ACGA2 is half the error ratio of the ACGA. In addition, when ACGA2 is applied in combination with other heuristic methods and VNS, the hybrid algorithm achieves optimal solution quality in comparison with other algorithms in the literature. Thus, the proposed algorithms are effective for solving the scheduling problems.

1. Introduction

In our previous studies [1–3], the ACGA algorithm was applied in the solving of several scheduling problems. The main characteristic of ACGA is that it alternates the EDAs and genetic operators in each generation. Since other EDAs [4–8] do not use genetic operators, this is a distinguishing feature of ACGA. This approach is beneficial for EDAs to have a diversified population [9]; GA-EDA [10] used a similar framework.

ACGA utilizes a univariate probabilistic model which extracts parental distribution from previous searches when the EDAs operator is responsible for generating offspring. After extraction the univariate probabilistic model is used to sample new solutions called artificial chromosomes. These prior studies show that the ACGA algorithm is able to provide satisfactory results.

The univariate probabilistic model of ACGA assumes that there are no dependencies between/among variables. However, some studies have pointed out that when variable interactions exist, EDAs may employ bi-variate or even multi-variate probabilistic models [11,12,5,13]. This research studied single machine scheduling problems with sequence-dependent setup times in a common due date environment [14]. Because a prior job influences the processing time of the next job when we consider the setup times, there exists strong interaction between the jobs. If ACGA is used to solve this scheduling problem, a satisfactory result may not be achieved. Stem from Jarboui et al. [15], a new bi-variate probabilistic model in conjunction with the univariate probabilistic model was adopted into the proposed algorithm, named ACGA2. As a result, ACGA2 is able to capture a more accurate parental distribution from the two probabilistic models and thus produce better offspring.

To provide more comparable result, some heuristic approaches and a famous local search algorithm, i.e., variable neighborhood search (VNS), are both employed in the ACGA2. The resultant algorithms could obtain better solution quality when we compare it with others in literature.

The organization of this paper is as follows: Section 2 discusses the importance of the scheduling problems in question and provides a problem definition. The related works of EDAs and VNS are also discussed in this section. The details of the ACGA2 and the theoretical analysis of using the univariate and bi-variate probabilistic models are introduced in Section 3. In Section 4 we make extensive comparisons with other algorithms in the literature that are commonly used to solve the scheduling problems under discussion. Finally, we present our conclusions in Section 5.

* Corresponding author.
E-mail addresses: shchen@csu.edu.tw (S.-H. Chen), cthunter@wfu.edu.tw (M.-C. Chen), ycliou@csu.edu.tw (Y.-C. Liou).

1568-4946/$ – see front matter © 2014 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.asoc.2013.12.019

2. Background to the scheduling problems with setup costs

This research discusses single machine scheduling problems with sequence-dependent setup time in a common due date environment. The objective was to minimize the total earliness and tardiness cost. To explain the importance of these scheduling problems, Section 2.1 presents a literature survey and the problem statement. In Section 2.2, we further explain the model of the scheduling problem.

2.1. Review and problem statements

Single-machine scheduling problems are one of the well-studied problems by many researchers. The application of single machine scheduling with setups can be found in minimizing the cycle time for pick and place (PAP) operations in Printed Circuit Board manufacturing company [16]; in a steel wire factory in China [17] and a sequencing problem in the weaving industry [18]. The results developed in the literature not only provide the insights into the single machine problem but also for more complicated environment such as flow shop or job shop. For the detail review of the scheduling problems with setup costs, it is available in [19].

The problem considered in this paper is to schedule a set of n jobs $\{j_1, j_2, \ldots, j_n\}$ on a single machine that is capable of processing only one job at a time without preemption. As explained in [20,14], all jobs are available at time zero, and a job j requires a processing time P_j. Job j belongs to a group $g_j \in \{1, \ldots, q\}$ (with $q \leq n$). Setup or changeover times, which are given as two $q \times q$ matrices, are associated to these groups. This means that in a schedule where j_j is processed immediately after j_i where $i,j \in \{1,2,\ldots,n\}$, there must be a setup time of at least S_{ij} time units between the completion time of j_i, denoted by C_i, and the start time of j_j, which is $C_j - P_j$. During this setup period, no other task can be performed by the machine and we assume that the cost of the setup operation is $c(g_i; g_j) \geq 0$ and let it be equal to machine setup time S_{ij} which is included as sequence dependent.

Apart from the sequence-dependent setup times, the objective is to complete all the jobs as close as possible to a large, common due date d. To accomplish this objective, the summation of earliness and tardiness is minimized. The earliness of job j is denoted as $E_j = \max(0, d - C_j)$ and its tardiness as $T_j = \max(C_j - d, 0)$, where C_j is the completion time of job j. Earliness and tardiness penalties for job j are weighted equally. The objective function is given by:

$$\min Z = \sum_{j=1}^{n} (E_j + T_j) = \sum_{j=1}^{n} |d - C_j| \tag{1}$$

The inclusion of both earliness and tardiness costs in the objective function is compatible with the philosophy of just-in-time production, which emphasizes producing goods only when they are needed. The early cost may represent the cost of completing a product early, the deterioration cost for a perishable goods or a holding (stock) cost for finished goods. The tardy cost can represent rush shipping costs, lost sales and loss of goodwill. Some specific examples of production settings with these characteristics are provided by Ow and Morton [21]. The set of jobs is assumed to be ready for processing at the beginning which is a characteristic of the deterministic problem. The set of jobs is assumed to be ready for processing at the beginning which is a characteristic of the deterministic problem. As a generalization of weighted tardiness scheduling, the problem is strongly NP-hard in [22]. Consequently, the early/tardy problem is also a strong NP-hard problem. It is the reason why this work attempts to use eACGA to conquer this NP-hard problem in a reasonable time.

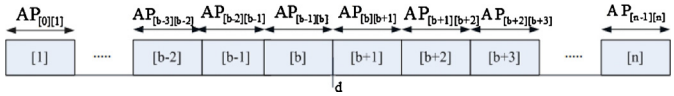

Fig. 1. The total earliness and total tardiness for a pre-assigned due-date d.

2.2. Formulations of the scheduling model

An example of the application of the common due date model would be an assembly system in which the components of the product should be ready at the same time, or to a shop where several jobs constitute a single customer's order [23]. Kanet [24] shows that the optimal sequence is when the bth job is completed at the due-date. The value of b is given by:

$$b = \begin{cases} n/2 & \text{if } n \text{ is even,} \\ (n+1)/2 & \text{if } n \text{ is odd,} \end{cases} \tag{2}$$

The common due-date (k^*) is the sum of processing times of jobs in the first b positions in the sequence; i.e.,

$$k* = C_b \tag{3}$$

As soon as the common due date is assigned, see Fig. 1, jobs can be classified into two groups that are early and tardy which are at position from 1 to b and b+1 to n respectively. The following notations are employed in the latter formulations.

j: job in position j.
A: the job set of tardy jobs.
B: the job set of early jobs.
$S_{[j][j+1]}$: Setup time of a job in position [j+1] follows a job in position [j].
$AP_{[j][j+1]}$: Adjusted processing time for the job in position j followed by the job in position [j+1].
b: the median position.
$AP_{[j][j+1]}$ is actually the processing time of job $j+1$ with setup time. Thus, the original form of $AP_{[j][j+1]}$ is written as:

$$AP_{[j][j+1]} = S_{[j][j+1]} + P_{j+1} \tag{4}$$

Our objective is to minimize the total earliness/tardiness cost. The formulation is given below.

$$\textit{Minimize} \quad f(x) = \sum_{j=1}^{n} (E_j + T_j) = TT + TE \tag{5}$$

where TT is total tardiness for a job sequence; TE is total earliness for a job sequence; and TT and TE can be transformed into the following equations based on the pre-defined adjusted processing time.

$$TT = \sum_{j=b}^{n-1} (n-j) AP_{[j][j+1]} \tag{6}$$

$$TE = \sum_{j=1}^{b} (j-1) AP_{[j-1][j]} \tag{7}$$

2.3. Related works of EDAs, VNS, and the combination of EDAs and VNS

In recent years, EDAs is one of the most popular evolutionary algorithms [25,26,15,7,27]. EDAs explicitly learn and build a probabilistic model to capture the parental distribution, and then samples new solutions from the probabilistic model [28]. Sampling from probabilistic models avoids the disruption of partial dominant solutions represented by the model, contrary to what

usually takes places when applying genetic operators, such as crossover and mutation operator [8]. This is the most important characteristic to distinguish EDAs and Genetic Algorithms (GAs). As claimed by Zhang and Muhlenbein [29], EDAs might be a promising method capable of capturing and manipulating the building blocks of chromosomes and, hence, efficiently solving hard optimization problems.

VNS has undergone a variety of systematic changes to its neighborhood structure during a local search [30,31]. Because VNS is able to solve the combinatorial optimization problems effectively, this meta-heuristic method has been used extensively. Take the scheduling problems for example, Tasgetiren et al. [32] employed the VNS to further improve the performance of Particle Swarm Optimization approach greatly when they solve the PFSPs. In the research of Sevkli and Aydin [33], since the combination of shaking and local search method could influence the performance, they tested various configurations when they attempted to deal with job shop scheduling problems. They indicated that the best shaking scheme may involve three consecutive steps: exchange, insert, and exchange. After the shaking procedure is done, the local search stage may apply two kinds of local heuristics, including the insert and exchange local search. The combination of shaking and local search scheme yields the lowest relative error ratio and the greatest optimum value for hit ratio.

When it comes to the combination of EDAs and VNS, Santana et al. [8] could be the first paper to evaluate the synergy of the two optimization algorithms. They defined three possible ways to combine these two methods in solving the protein side chain placement problem. After that, Jarboui et al. [15] proposed the EDAs with VNS, named (EDA-VNS) to solve the PFSPs with minimization of total flowtime. Their probabilistic model considered the order of the job in the sequence and the building blocks of the jobs. In addition, VNS is an improvement procedure as the EDA is run. Jarboui et al. [15] found EDA-VNS was effective in small benchmarks; however, when it comes to larger size problems, VNS is better than EDA-VNS in terms of the objective values and the computational time. It could be worthwhile to examine the reason why EDA-VNS does not outperform the VNS in large size benchmarks.

3. Methodology

In order to capture the information of variable interactions, the bi-variate probability model was taken into consideration as well. That is, ACGA2 extends from the ACGA, which not only collects the order information of the job in the sequence but also the variable interaction of the jobs. As a result, ACGA2 extracts the parental information by using both the univariate probability model and a bi-variate probability model.

The following Section 3.1 explains the proposed algorithm in detail. Because univariate and bi-variate models are the core of EDAs, they are further explained in Section 3.2. Section 3.3 discusses the reason of using the dependent probabilistic model in the ACGA2. To further improve the solution quality, we introduce the well-known local search operator VNS into the ACGA2. The detail procedures of VNS is shown in Section 3.4.

3.1. ACGA2 procedures

The major procedures of ACGA2 comprised population initialization, selection of better chromosomes, and then decision-making on whether EDAs or genetic operators should be ran. After that, a replacement strategy was adopted and the stopping criterion was tested. The framework of the ACGA2 is depicted in Fig. 2.

We describe our methods in the following steps.

Population: A set of solutions
Generations: The maximum number of generations
$OP(t)$: Univariate Probabilistic model
$DP(t)$: Bi-variate Probabilistic model
t : Generation index
k : The execution of the EDA interval
startingGen : The generation at which the EDA will start to run

1: Initialize *Population*
2: $t \leftarrow 0$
3: Initialize $OP(t)$ and $DP(t)$
4: **while** $t <$ *Generations* **do**
5: Selection / Elitism(*Population*)
6: **if** $g \% k == 0$ *and* $t >$ *startingGen* **then**
7: $OP(t+1) \leftarrow$BuildUnivariateProbabilityModel(*Selected Chromosomes*)
8: $DP(t+1) \leftarrow$BuildBiVariateProbabilityModel(*Selected Chromosomes*)
9: Learning
10: Sampling new solutions into Population
11: **else**
12: Crossover()
13: Mutation()
14: **end if**
15: EvaluateFitness (*Population*)
16: Replacement()
17: $t \leftarrow t+1$
18: **end while**

Fig. 2. Algorithm 1: main procedure of ACGA2.

Step 1: initialization

We initialized a population consisting of a number of chromosomes in Line 1. A chromosome represents a processing sequence for the scheduling problem. Each chromosome was generated randomly. To improve results some studies have used heuristics to generate better initial solutions at this step. For example, dominance properties (DP) were adopted by [34] to generate a single chromosome in ACGA2. Shortest Adjusted Processing Time (SAPT) is the other heuristic that gives good initial solutions. The two heuristics were employed to work with ACGA2 and they are named $ACGA2_{DP}$ and $ACGA2_{SAPT}$, respectively.

Additionally we also had to initialize the probabilistic models used in ACGA2 in Line 3. Regardless of whether for univariate or bi-variate probabilistic models, each $P_{ij}(t)$ was initialized to be $1/n$, where n was the number of jobs and $P_{ij}(t)$ was the probability of job i in position j delivering a promising solution.

Step 2: selection and elitism strategy (Line 5)

Evolutionary algorithms attempt to select fitter solutions corresponding to their objective values. The selection operator selects the better chromosomes for survival. For ease of purpose, the binary tournament operator was employed, which selects better chromosomes with lower objective values in this minimization problem. In order to preserve elites in the population, a proportion of better chromosomes are stored in an external archive and to be copied into the mating pool during the selection stage.

Step 3: decision (Line 6)

There are two parameters that control whether to run EDAs or GAs, which are *startingGen* and *interval*. The first parameter *startingGen* is used to determine the starting time of the generation of artificial chromosomes. The main reason is that the probabilistic model should be only applied to generate better chromosomes when the searching process reaches a more stable state.

The other important parameter *interval* sets the period of artificial chromosomes generated. $g \,\%\, k$ represents that g mod k. When the resultant value is 0, it means the current generation reaches the k interval. As a result, the algorithm alternates EDAs and genetic operators throughout the entire evolutionary progress.

Step 4: variations

We executed the EDAs to construct the probabilistic models, learn parental distribution, and sample new offspring from probabilistic models. These are described in Step 4.1.1 to Step 4.1.3. Meanwhile genetic operators contained the crossover and mutation operators which comprise Step 4.2.1 and Step 4.2.2, respectively.

Step 4.1: EDAs
Step 4.1.1: modeling (Lines 7 and 8)

The univariate probabilistic model and the bi-variate probabilistic models were built while we ran the EDAs. The former represents all jobs at different positions referring to the frequency count of a job at those positions. The bi-variate probabilistic model is similar to a container of interaction counters which blocks out similar jobs in the sequences. Detail step by step details will be presented in Section 3.2.

Step 4.1.2: learning (Line 9)

As in PBIL [11], we updated the two probabilistic models in an incremental learning way. In addition, the learning rate determined the importance of the current and historical probability information. The probability learning models of the univariate and bi-variate probabilistic models are shown in Eqs. (15) and (16), respectively.

Step 4.1.3: sampling (Line 10)

Now that the two probabilistic models have been established, the actual procedure implemented in the optimization algorithm needs to be specified. The goal was to devise a strategy to form a offspring population which reflect the two probabilistic models. For each position in the sequence of a new individual, first we selected a job randomly in the first position, then, according to the multiplication of the two probabilistic models, we filled out the other sequences of the new individual with proportional selection.

Step 4.2.1: crossover (Line 12)

This study applies the two-point central crossover operator [35] to mate two chromosomes which are randomly selected. Crossover rate (P_c) decides whether the chromosome is mated with others.

Step 4.2.2: mutation (Line 13)

A chromosome is decided to be mutated 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: replacement (Line 16)

In order to improve the population quality and keep the population diversity, an individual replaces with the worst one in the parent population when the offspring is better. Furthermore, the offspring must be different from any one of the parent population.

3.2. Establishing probabilistic models

We will explain how to establish the univariate probabilistic model first and then the bi-variate probabilistic model. Suppose a population has M chromosomes $X^1, X^2, \ldots, X^M$ at the current generation t, which is denoted as Population(t). Then, X_{ij}^k is a binary variable in chromosome k, which is shown in Eq. (8).

$$X_{ij}^k = \begin{cases} 1 & \text{if job } i \text{ is assigned to position } j \\ 0 & \text{Otherwise} \end{cases} \tag{8}$$

where $i = 1, \ldots, n$; $j = 1, \ldots, n$.

If job i exists at position j, the number of occurrence of X_{ij} is incremented by 1. There are m chromosomes and the order information of job i on position j (f_{ij}) will be calculated as follows:

$$f_{ij}(t) = \sum_{k=1}^{m} X_{ij}^k, \qquad i = 1, \ldots, n, \quad j = 1, \ldots, n \tag{9}$$

The univariate probability model presents the occurrence possibility of these jobs in the sequence at different positions. In order to combine the univariate probabilistic model with the bi-variate probabilistic model, the univariate probabilistic model Op_{ij} will be the total number of times of appearance of the job i before or in the position j at current generation t. Thus, the ordinal probability is to accumulate the distribution Eq. (9) in position j, which is as follows:

$$Op_{ij}(t) = \sum_{l=1}^{j} \left(\sum_{k=1}^{m} X_{ij}^k \right)^l, \qquad i = 1, \ldots, n, \quad j = 1, \ldots, n \tag{10}$$

The ordinal probabilistic matrix of all jobs at different positions is written as Eq. (11).

$$Op(t) = \begin{pmatrix} Op_{11}(t) & \cdots & Op_{1n}(t) \\ \vdots & \ddots & \vdots \\ Op_{n1}(t) & \ldots & Op_{nn}(t) \end{pmatrix} \tag{11}$$

Moreover, how to establish a dependency probabilistic model is explained as follows. A dependence ((v_{ij})) means a job i influences another job j. Suppose a population has M strings $v^1, v^2, \ldots, v^M$ at current generation t. Then, v_{ij}^k is a binary variable in chromosome k, which is shown in Eq. (12).

$$v_{ij}^k = \begin{cases} 1 & \text{if job } j \text{ connect to job } i \\ 0 & \text{Otherwise} \end{cases} \tag{12}$$

where $i = 1, \ldots, n$; $j = 1, \ldots, n$.

The interaction information is collected from N best chromosomes where only paired interactions between the jobs are taken into account. Let $Dp_{ij}(t)$ (dependency probability) be the number of times of appearance of job i after the job j at current generation t. $Dp_{ij}(t)$ is updated as follows:

$$Dp_{ij}(t) = \sum_{k=1}^{m} v_{ji}^k, \qquad i = 1, \ldots, n, \quad j = 1, \ldots, n \tag{13}$$

The dependency probabilistic matrix of paired interaction of all jobs are written as Eq. (14).

$$Dp(t) = \begin{pmatrix} 0 & Dp_{12}(t) & \cdots & Dp_{1n}(t) \\ Dp_{21}(t) & 0 & \cdots & Dp_{2n}(t) \\ \vdots & \vdots & \ddots & \vdots \\ Dp_{n1}(t) & Dp_{n2}(t) & \cdots & 0 \end{pmatrix} \quad (14)$$

Furthermore, the above two probabilities with learning can continue to modify the search space to improve the performance of the algorithm. The equation is presented as Eqs. (15) and (16). In this research, the ordinal learning rate ($\lambda_{OP} \in (0, 1)$) and dependent learning rate ($\lambda_{DP} \in (0, 1)$) of the two parameters were decided by the design-of-experiment (DOE), which we discuss in the next section.

$$Op_{ij}(t) = Op_{ij}(t) \times (1.0 - \lambda_{Op}) + Op_{ij}(t-1) \times \lambda_{Op} \quad (15)$$

$$Dp_{ij}(t) = Dp_{ij}(t) \times (1.0 - \lambda_{Dp}) + Dp_{ij}(t-1) \times \lambda_{Dp} \quad (16)$$

As soon as the ordinal *Op* and dependent probability *Dp* are built, jobs are assigned onto each position. As far as diversification is concerned, there are three methods for creating diversified artificial chromosomes. The first is for random selection at the first position in the sequence. The second is for proportional selection to be used to mitigate the probability of job i being assigned to a position j. The third is zero value transformation which is used to transform 0 into $1/n$ in dependent probability when jobs do not have any interactions. The assignment procedure is determined as follows:

S: A set of shuffled sequence which determines the sequence of each position is assigned a job.
Ω: The set of un-arranged jobs.
J: The set of arranged 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 $\theta \le OP_{ik} \times DP_{ij(k-1)} / \sum OP(i, k) \times DP(i, J(k-1))$, where $i \in \Omega$
6: $J(k) \leftarrow i$
7: $\Omega \leftarrow \Omega \setminus i$
8: $S \leftarrow S \setminus k$
9: **end while**

3.3. Theoretical analysis

The inclusion of the dependency probabilistic (Eq. (14)) is quite necessary to extract the interaction among the pair-wised jobs. The major reason is that the adjusted processing time $AP_{[j][j+1]}$ in Eq. (4) is not equal to P_{j+1} because the setup time $S_{[j][j+1]}$ is not zero in the studied problem. It is apparently that there exists interactions between the jobs. Simply capturing ordinal information from the parental distribution is not sufficient for the studied problem.

In addition, when ACGA2 selects an appropriate job j after the job i, the statistic information in the two probabilistic models are used. In particular, the bi-variate model may forbid a job k which has a higher setup cost when job k is scheduled after job i. As a result, the solution space would be reduced due to the bi-variate probability model is employed in the ACGA2. As a result, ACGA2 would perform better than the ACGA since ACGA ignores all the variable interactions.

On the other hand, Zhang and Mühlenbein [29] proved that if the distribution of the new solutions matches that of the parent set, the algorithm will converge to the global optimal when we apply the proportional selection, truncation selection, and the tournament selection model. Even though both the ACGA and ACGA2 utilize the binary tournament, the models used by ACGA2 would provide more and accurate parental distribution during the generations owing to the interaction information is preserved. To understand the distribution generated by ACGA and ACGA2, we employ the entropy proposed by Shannon [36]. Suppose the information gains of ACGA and ACGA2 at generation t are $I(Op_{ij}^t)$ in Eq. (17) and $I(Op_{ij}^t * Dp_{ij}^t)$ in Eq. (18), respectively.

$$I(Op_{ij}^t) = -\sum_{i=1}^{n} \frac{\log_b P(Op_{ij}^t)}{n} \quad (17)$$

$$I(Op_{ij}^t * Dp_{ij}^t) = -\sum_{i=1}^{n} \frac{\log_b P(Op_{ij}^t * Dp_{ij}^t)}{n} = -\sum_{i=1}^{n} \frac{[\log_b P(Op_{ij}^t) + \log_b P(Dp_{ij}^t)]}{n} \quad (18)$$

where b is the base of the logarithm.

Because the probability values of $P(Op_{ij}^t)$ and $P(Dp_{ij}^t)$ are less than 1, the log value of $\log_b P(Op_{ij}^t)$ and $\log_b P(Dp_{ij}^t)$ are definitely negative. Consequently, $I(Op_{ij}^t)$ is less than $I(Op_{ij}^t * Dp_{ij}^t)$ while the later one has more information of $\log_b P(Dp_{ij}^t)$. The sampled distribution yielded by eACGA may match the better parental distribution than the one of ACGA.

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 [37,32]. In this paper, we have included the version of Jarboui et al. [37] in our research. Meanwhile, the combination of the ACGA2 and VNS is named $ACGA2_{VNS}$ 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 $ACGA2_{VNS}$. 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. The perturbation scheme is to produce neighborhood from current best solution. The new solution is represented as x_1. After that, the swap local search is used first to modify the solution of x_1 and then insertion local search is employed later. The two local search strategies were introduced by Jarboui et al. [37].

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.
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 ← insertionLocalSearch(x_2)
6:        if F(x'_2) < F(x_best) then
7:          x_best ← x'_2
8:          k ← 1
9:        else
10:         k ← 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 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 [33], 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 $ACGA2_{VNS}$.

4. Experimental results

The basic testing instances are as designed by Rabadi et al. [38] and the job size of each instance includes 10, 15, 20 and 25. The properties of the processing time range contain low, median and high values, which are based on Uniform(10, 60), Uniform(10, 110) and Uniform(10, 160), respectively. Moreover, these instances might not be sufficient to demonstrate the complexity of the problem. We apply similar concept to generate large size of problems, which include 50 and 100 jobs. The distribution of these instances is also based on the processing time range that includes low, median and high. Each combination has 15 similar instances, the total number of instances is 270 ($6 \times 3 \times 15$). Each instance is replicated 30 times for our proposed algorithm and the compared algorithms. In order to configure the parameter settings, we utilized the DOE to select the best setting of ACGA2 parameters. Tables 1 and 2 show the parameter settings of the ACGA2 and VNS experiment, respectively. We coded the algorithm in Java and ran it on a Windows 2003 server (Intel Xeon 3.2 GHz).

Table 1
*e*ACGA parameters setting.

Factor	Default
Crossover rate	0.5
Mutation rate	0.3
Starting generation	0
Interval	2
Ordinal probability learning rate	0.1
Dependent probability learning rate	0.5
Population size	100
Generations	1000

Table 2
VNS parameters setting.

Job size	*Penh*	k_{max}
10	0.1	20
15	0.1	20
20	0.1	50
25	0.1	50
50	0.1	100
100	0.1	100

In order to evaluate the performance of ACGA2, comparison was carried out with some algorithms in the literature. We divided these algorithms into two groups; stand-alone algorithms and hybrid algorithms. To test the efficiency of the proposed ACGA2 algorithm against stand-alone algorithms, we compared the SGA [39], ACGA [40] and SAPT [41]. We also selected some hybrid algorithms, such as SGA_{DP} [34], $ACGA_{DP}$ [40] and SA_{SAPT} [41]. In addition, when the ACGA2 was combined with two heuristic algorithms DPs [34] and SAPT [41] because they generate good initial solutions, we named them $ACGA2_{DP}$ and $ACGA2_{SAPT}$, respectively. A brief summary of these algorithms is as follows:

- SGA [39]: This is a standard genetic algorithm that follows an elitist strategy. The genetic operators are binary tournament selection, two-point central operator and swap mutation operator. During the selection stage, 10% of elites in the population are reserved for the next generation. We use the data from [34] for comparison.
- ACGA [40]: We have already mentioned that ACGA does not consider the dependencies between/among variables. When using ACGA to solve the benchmark problems of this study, we were able to distinguish performance differences when we employed the bi-variate probabilistic model together with the univariate probabilistic model.
- SAPT [41]: Shortest Adjusted Processing Time (SAPT) is based on the concept that jobs to be scheduled with shorter adjusted processing times should be closer to the median position in an optimal job sequence. In the study mentioned SAPT was also combined with the simulated annealing (SA) algorithm, called SAPT-SA. Since the presented data of SAPT-SA [41] are not complete and the termination criterion was different from our experiments, we only used the SAPT data from [41] for comparison.
- SGA_{DP} [34]: Dominance properties (DPs) were developed by swapping neighboring jobs. DPs are very efficient when combined with GAs. We take the results from [34] for comparison.
- $ACGA_{DP}$ [34]: According to past research on ACGA [40] and DPs [34], we combined both algorithms to solve the single machine scheduling problems with setup costs. DPs were utilized to generate a set of good initial solutions. ACGA takes advantage of these good initial solutions and then continues the evolutionary progress. According to our evaluation $ACGA_{DP}$ should perform better than ACGA.
- $ACGA2_{SAPT}$: SAPT is used to generate good initial solutions and ACGA2 improves the solution quality given by SAPT.
- $ACGA2_{SAPT+VNS}$: It is similar to $ACGA2_{SAPT}$; however, VNS is used to further improve the solution quality in the search.

To compare the performance of these algorithms, we employed the average relative error ratio and each algorithm evaluates 100,000 solutions. In the literature, the average relative error ratio is often used to evaluate the performance of algorithms, whereby the error ratio (ER_i) of a solution (X_i) generated by an algorithm is calculated as follows:

$$ER_i = \frac{Obj_{avg}(X_i) - Opt_i}{Opt_i} \times 100, \tag{19}$$

where Opt_i is the objective value of the best known or optimal solution which is available by Sourd [14] who applied Branch-and-Bound algorithm to derive the solution. For the larger size problems (i.e., 50 and 100 jobs), we aggregated the experiment results of the compared algorithms among the 30 replications while each algorithm applied more than 10 times of regular computation time. The obtained current best objective values is shown in Table 3. Finally, the $Obj_{avg}(X_i)$ is the X_i average objective value where we replicate each algorithm for 30 times.

Table 3
The current best objective value of larger size problems.

Size	Instance	Low	Med	High	Size	Instance	Low	Med	High
50	1	7165	7743	8564	100	1	26041	28350	29410
	2	6711	7694	8224		2	26214	28656	30939
	3	6965	7582	8875		3	26011	27759	29683
	4	6709	7788	8480		4	26143	28069	30424
	5	7029	7824	8948		5	26099	27810	29697
	6	6734	7657	8577		6	26316	28392	29204
	7	7094	8122	8789		7	26240	28116	30431
	8	7102	7638	8297		8	26367	28455	29322
	9	7157	7371	9565		9	26138	27609	29930
	10	7195	7599	8692		10	26090	28146	30080
	11	6930	7592	8734		11	26280	28195	29491
	12	6901	7614	8366		12	25922	27878	29687
	13	7131	7702	8506		13	26445	28503	29552
	14	7007	7606	8337		14	26292	27698	29760
	15	6908	7729	8615		15	26400	28131	29809

Table 4
Evaluation of algorithms in objective value and CPU times.

Type	Size	Without hybridization				With hybridization					
		SGA	ACGA	SAPT	ACGA2	SGA_{DP}	$ACGA_{DP}$	$ACGA2_{DP}$	$ACGA2_{SAPT}$	$ACGA2_{VNS}$	$ACGA2_{SAPT+VNS}$
Average error ratio											
Low	10	0.86	0.63	2.55	0.27	0.31	0.28	0.18	0.11	0.00	0.00
	15	4.21	4.05	3.59	1.85	3.13	2.93	1.40	0.77	0.02	0.03
	20	9.58	8.83	3.13	3.97	6.85	6.98	3.66	1.80	0.16	0.14
	25	13.18	11.81	3.21	6.10	9.44	9.55	5.15	2.06	0.70	0.42
Med	10	1.89	1.29	2.80	0.18	0.93	0.94	0.26	0.36	0.00	0.00
	15	7.86	6.84	5.33	3.11	4.74	4.69	2.30	0.89	0.02	0.03
	20	13.93	12.53	3.93	5.74	10.06	9.86	4.90	2.25	0.22	0.28
	25	20.80	18.83	5.96	9.55	14.83	15.32	8.56	4.26	0.76	0.71
High	10	1.37	1.16	2.80	0.04	0.54	0.62	0.00	0.13	0.00	0.00
	15	9.70	9.35	8.22	3.42	6.38	5.66	2.44	0.69	0.01	0.01
	20	21.04	18.18	5.95	7.78	13.94	13.55	6.62	3.70	0.16	0.13
	25	27.88	24.74	6.83	13.09	20.22	19.78	10.88	4.84	1.32	1.02
	Avg	11.03	9.85	4.53	4.59	7.61	7.51	3.86	1.82	0.28	0.23
Low	50	23.49	21.98	5.29	15.10	17.79	17.78	12.73	2.32	1.45	0.98
	100	38.55	33.78	2.27	30.38	22.11	21.94	21.43	2.10	2.51	0.47
Med	50	23.49	21.98	5.29	15.10	17.79	17.78	12.73	2.32	2.83	0.98
	100	66.31	58.36	3.64	52.41	38.33	38.45	36.83	3.42	4.71	0.82
High	50	50.35	47.58	9.68	32.10	38.54	38.11	26.87	5.97	2.85	2.16
	100	91.46	80.92	4.82	72.30	53.58	53.29	51.84	4.41	6.49	1.08
	Avg	51.50	46.50	5.61	37.90	33.36	33.19	28.50	3.71	3.47	1.20
Average CPU time											
Low	10	0.26	0.39	0.01	0.70	0.39	0.57	0.98	0.77	1.21	1.23
	15	0.31	0.47	0.03	1.18	0.46	0.71	1.42	1.25	2.47	2.50
	20	0.37	0.56	0.07	1.85	0.54	0.90	2.06	1.96	4.98	5.10
	25	0.43	0.65	0.18	2.60	0.63	1.09	2.79	2.79	13.83	14.05
Med	10	0.26	0.39	0.02	0.72	0.38	0.57	0.98	0.77	1.21	1.23
	15	0.31	0.47	0.04	1.18	0.46	0.71	1.42	1.25	2.48	2.50
	20	0.37	0.56	0.06	1.86	0.55	0.89	2.06	1.96	4.96	5.09
	25	0.43	0.66	0.12	2.60	0.63	1.09	2.79	2.76	13.92	14.16
High	10	0.26	0.39	0.07	0.73	0.38	0.57	0.98	0.77	1.21	1.23
	15	0.31	0.46	0.11	1.18	0.46	0.72	1.42	1.25	2.46	2.49
	20	0.37	0.56	0.13	1.86	0.54	0.89	2.06	1.95	5.01	5.12
	25	0.43	0.65	0.18	2.60	0.63	1.09	2.79	2.75	13.96	14.11
	Avg	0.34	0.52	0.09	1.59	0.50	0.82	1.81	1.69	5.64	5.73
Low	50	0.87	1.29	0.48	7.63	1.20	2.33	7.50	10.35	328.76	329.31
	100	2.20	4.06	3.53	33.46	2.52	4.30	33.17	37.79	1307.82	1322.93
Med	50	0.86	1.28	0.42	7.63	1.20	2.35	7.49	9.13	328.00	329.43
	100	2.18	4.09	3.25	31.89	2.54	4.31	32.63	38.36	1315.21	1329.49
High	50	0.86	1.28	0.47	7.63	1.20	2.33	7.49	8.64	328.68	329.23
	100	2.21	4.12	2.25	31.72	2.53	4.33	32.16	37.51	1313.96	1337.19
	Avg	1.53	2.69	1.73	19.99	1.87	3.33	20.07	23.63	820.41	829.60

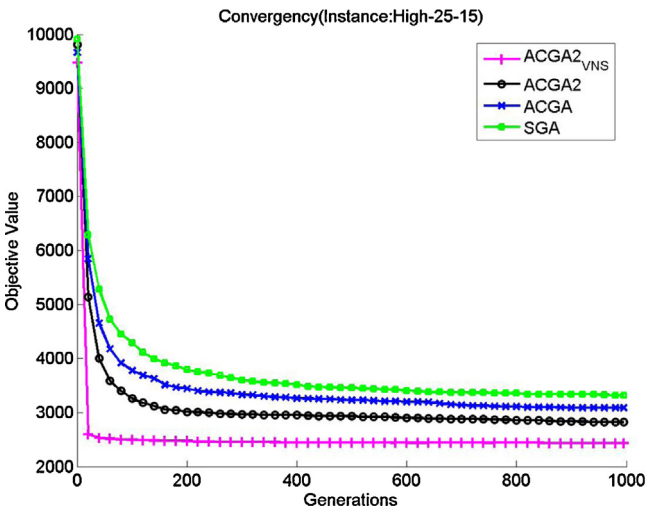

Fig. 3. The convergency (instance High 25-15).

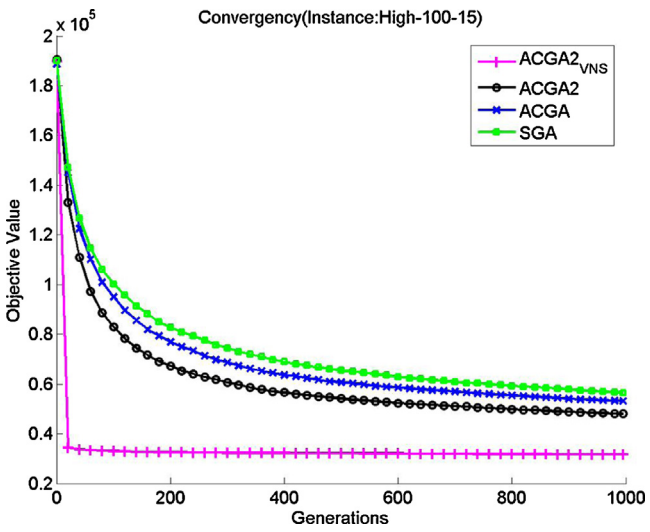

Fig. 4. The convergency (instance High 100-15)

Table 4 showed the statistics of the average *ER* values of all the algorithms on the test instances.

When we evaluate the algorithms without using other approach, ACGA2 not only outperforms SGA but also ACGA. In particular, the average error ratio of ACGA2 is half that of the ACGA. Thus it is clear that the adoption of a bi-variate probability model is beneficial for sequence-dependent scheduling problems. Note also, there is no difference between ACGA2 and SAPT in terms of average relative error ratio.

Apart from the stand-alone algorithms, two heuristics were further embedded in a meta-heuristic algorithm to improve the efficiency and effectiveness of the global search procedure. The results of the hybrid framework are also shown in Table 4. Both $ACGA2_{DP}$ and $ACGA2_{SAPT}$ performed better than ACGA2 and SAPT in terms of average relative error ratio. For example, $ACGA2_{SAPT}$ was 2.5 times lower than SAPT and the $ACGA2_{SAPT}$ algorithm was 4.2 times lower than SGA_{DP}. The empirical results show that domain-specific heuristics are useful for the scheduling problem under investigation and $ACGA2_{SAPT}$ achieves the best solution quality when it is compared with others in the literature.

We further compared the CPU time of these algorithms. The results are given in Table 4. SGA appears more efficient than ACGA2 and ACGA in terms of CPU time because it requires a linear time to create a new solution, whereas artificial chromosomes require $O(n^2)$ time. ACGA is faster than ACGA2 because we execute the EDAs more frequently in ACGA2 than in ACGA. In addition, due to the need to construct a bi-variate probabilistic model for ACGA2, more time is required. Comparing $ACGA2_{DP}$ and $ACGA2_{SAPT}$, $ACGA2_{SAPT}$ was superior to $ACGA2_{DP}$ in terms of CPU time. Although DPs and SAPT both use general pair-wise interchange (GPI) for sequencing, SAPT only uses the neighborhood generated by every possible pair-wise interchange, i.e. the neighborhood distance is 1. However, the neighborhood distance of DPs is 2 and 3 so DPs may need a longer computational time (Fig. 3).

5. Discussion and conclusions

This study proposed an ACGA2 which enables the ACGA algorithm to cope with interactions between variables. Both univariate and bi-variate probabilistic models are used in ACGA2 so that the ACGA2 can extract a more accurate problem structure from parental distribution. In addition, this paper gives a theoretical analysis of the two probabilistic models in this paper. ACGA2 and other algorithms were adopted to deal with the single machine scheduling problem with sequence-dependent setup times in a common due date environment. Due to the issue that the previous job impacts the processing time of the next job, there exists strong variable interactions in the problem. As a result, when ACGA2 is compared with ACGA, ACGA2 is superior to ACGA because the average error ratio of ACGA2 is just half the average error ratio of ACGA. It is thus apparent that the bi-variate probabilistic model together with the univariate probabilistic model could gain more information from previous searches for the conditions of sequence-setup times (Fig. 4).

Although the bi-variate probabilistic model is useful, some domain-specific heuristics could further improve the solution quality of ACGA2. When ACGA2 is run in combination with DP or SAPT, the two hybrid algorithms are further improved. In particular $ACGA2_{SAPT}$ is state-of-the-art when compared with others in the literature. In future studies, ACGA2 could be used to solve larger-size single machine sequence-dependent scheduling problems than the existing 25-job benchmark problem. Larger size problems could aid in distinguishing the performance of the different algorithms. In this way we could better understand the advantages or disadvantages of the proposed algorithm.

References

[1] P.C. Chang, S.H. Chen, C.Y. Fan, Genetic algorithm integrated with artificial chromosomes for multi-objective flowshop scheduling problems, Applied Mathematics and Computation 205 (2) (2008) 550–561.
[2] P.C. Chang, S.H. Chen, C.Y. Fan, Mining gene structures to inject artificial chromosomes for genetic algorithm in single machine scheduling problems, Applied Soft Computing Journal 8 (1) (2008) 767–777.
[3] P.C. Chang, J.C. Hsieh, S.H. Chen, J.L. Lin, W.H. Huang, Artificial chromosomes embedded in genetic algorithm for a chip resistor scheduling problem in minimizing the makespan, Expert Systems with Applications 36 (3 Part 2) (2009) 7135–7141.
[4] S. Baluja, An Empirical Comparison of Seven Iterative and Evolutionary Function Optimization Heuristics, Technical Report CMU-CS-95-193, Carnegie Mellon University, 1995.
[5] G.R. Harik, F.G. Lobo, D.E. Goldberg, The compact genetic algorithm, IEEE Transactions on Evolutionary Computation 3 (4) (1999) 287–297.
[6] H. Muhlenbein, G. Paaß, From recombination of genes to the estimation of distributions: I. Binary parameters, Lecture Notes in Computer Science 1141 (1996) 178–187.
[7] Q. Zhang, J. Sun, E. Tsang, An evolutionary algorithm with guided mutation for the maximum clique problem, IEEE Transactions on Evolutionary Computation 9 (2) (2005) 192–200.
[8] R. Santana, P. Larrañaga, J. Lozano, Combining variable neighborhood search and estimation of distribution algorithms in the protein side chain placement problem, Journal of Heuristics 14 (2008) 519–547.
[9] S.H. Chen, M.C. Chen, P.C. Chang, Q. Zhang, Y.M. Chen, Guidelines for developing effective Estimation of Distribution Algorithms in solving single machine scheduling problems, Expert Systems with Applications 37 (9) (2010) 6441–6451.

[10] J. Pena, V. Robles, P. Larranaga, V. Herves, F. Rosales, M. Perez, GA-EDA: hybrid evolutionary algorithm using genetic and estimation of distribution algorithms, in: B. Orchard, C. Yang, M. Ali (Eds.), Innovations in Applied Artificial Intelligence, Lecture Notes in Computer Science, vol. 3029, Springer, Berlin, Heidelberg, 2004, pp. 361–371.
[11] S. Baluja, S. Davies, Fast probabilistic modeling for combinatorial optimization, in: Proceedings of the Fifteenth National/Tenth Conference on Artificial Intelligence, John Wiley & Sons Inc., 1998, pp. 469–476.
[12] S. Baluja, S. Davies, Using Optimal Dependency-Trees for Combinational Optimization, in: Proceedings of the Fourteenth International Conference on Machine Learning Table of Contents, 1997, pp. 30–38.
[13] M. Pelikan, D.E. Goldberg, E. Cantu-Paz, BOA: the Bayesian optimization algorithm, in: Proceedings of the Genetic and Evolutionary Computation Conference GECCO-99, vol. 1, 1999, pp. 525–532.
[14] F. Sourd, Earliness-tardiness scheduling with setup considerations, Computers and Operations Research 32 (7) (2005) 1849–1865.
[15] B. Jarboui, M. Eddaly, P. Siarry, An estimation of distribution algorithm for minimizing the total flowtime in permutation flowshop scheduling problems, Computers & Operations Research 36 (9) (2009) 2638–2646.
[16] O. Kulak, I. Yilmaz, H. Günther, PCB assembly scheduling for collect-and-place machines using genetic algorithms, International Journal of Production Research 45 (17) (2007) 3949–3969, ISSN 0020-7543.
[17] F. Jin, J. Gupta, S. Song, C. Wu, Single machine scheduling with sequence-dependent family setups to minimize maximum lateness, Journal of the Operational Research Society 61 (7) (2009) 1181–1189, ISSN 0160-5682.
[18] M.H. Al-Haboubi, S.Z. Selim, A sequencing problem in the weaving industry, European Journal of Operational Research 66 (1) (1993) 65–71, ISSN 0377-2217.
[19] A. Allahverdi, C. Ng, T.C.E. Cheng, M.Y. Kovalyov, A survey of scheduling problems with setup times or costs, European Journal of Operational Research 187 (3) (2008) 985–1032, ISSN 0377-2217.
[20] K. Baker, G. Scudder, Sequencing with earliness and tardiness penalties: a review, Operations Research 38 (1) (1990) 22–36.
[21] P.S. Ow, T.E. Morton, The single machine early/tardy problem, Management Science 35 (2) (1989) 177–191.
[22] J. Lenstra, A. Kan, P. Brucker, Complexity of machine scheduling problems, Annals of Discrete Mathematics 1 (1977) 343–362.
[23] V. Gordon, J. Proth, C. Chu, A survey of the state-of-the-art of common due date assignment and scheduling research, European Journal of Operational Research 139 (1) (2002) 1–25.
[24] J. Kanet, Minimizing the average deviation of job completion times about a common due date, Naval Research Logistics Quarterly 28 (4) (1981) 643–651.
[25] U. Aickelin, E.K. Burke, J. Li, An estimation of distribution algorithm with intelligent local search for rule-based nurse rostering, Journal of the Operational Research Society 58 (12) (2007) 1574–1585.
[26] R. Baraglia, J. Hidalgo, R. Perego, A hybrid heuristic for the traveling salesman problem, IEEE Transactions on Evolutionary Computation 5 (6) (2001) 613–622.
[27] S.-H. Chen, M.-C. Chen, Addressing the advantages of using ensemble probabilistic models in Estimation of Distribution Algorithms for scheduling problems, International Journal of Production Economics 141 (1) (2013) 24–33.
[28] M. Pelikan, D.E. Goldberg, F.G. Lobo, A survey of optimization by building and using probabilistic models, Computational Optimization and Applications 21 (1) (2002) 5–20.
[29] Q. Zhang, H. Muhlenbein, On the convergence of a class of estimation of distribution algorithms, IEEE Transactions on Evolutionary Computation 8 (2) (2004) 127–136.
[30] N. Mladenovic, P. Hansen, Variable neighborhood search, Computers and Operations Research 24 (11) (1997) 1097–1100.
[31] P. Hansen, N. Mladenovi, Variable neighborhood search: principles and applications, European Journal of Operational Research 130 (3) (2001) 449–467.
[32] M.F. Tasgetiren, Y.C. Liang, M. Sevkli, G. Gencyilmaz, A particle swarm optimization algorithm for makespan and total flowtime minimization in the permutation flowshop sequencing problem, European Journal of Operational Research 177 (3) (2007) 1930–1947.
[33] M. Sevkli, M. Aydin, Variable neighbourhood search for job shop scheduling problems, Journal of Software 1 (2) (2009) 34, ISSN 1796-217X.
[34] P.C. Chang, T. Lie, J.Y. Liu, S. Chen, A genetic algorithm enhanced by dominance properties for single machine scheduling problems with setup cost, International Journal of Innovative Computing, Information and Control 7 (5) (2011) 1–21, ISSN 1349-4198.
[35] T. Murata, H. Ishibuchi, Performance evaluation of genetic algorithms for flowshop scheduling problems, in: Proceedings of the First IEEE World Congress on Computational Intelligence, IEEE, 1994, pp. 812–817.
[36] C.E. Shannon, A mathematical theory of communication, Bell System Technical Journal 27 (3) (1948) 379–423.
[37] B. Jarboui, M. Eddaly, P. Siarry, A. Rebaï, An estimation of distribution algorithm for minimizing the makespan in blocking flowshop scheduling problems, Computational Intelligence in Flow Shop and Job Shop Scheduling (2009) 151–167.
[38] G. Rabadi, M. Mollaghasemi, G. Anagnostopoulos, A branch-and-bound algorithm for the early/tardy machine scheduling problem with a common due-date and sequence-dependent setup time, Computers & Operations Research 31 (10) (2004) 1727–1751, ISSN 0305-0548.
[39] C.R. Reeves, A genetic algorithm for flowshop sequencing, Computers and Operations Research 22 (1) (1995) 5–13.
[40] P. Chang, S.H. Chen, C.Y. Fan, Generating artificial chromosomes with probability control in genetic algorithm for machine scheduling problems, Annals of Operations Research 180 (1) (2010) 197–211.
[41] G. Rabadi, G.C. Anagnostopoulos, M. Mollaghasemi, A heuristic algorithm for the just-in-time single machine scheduling problem with setups: a comparison with simulated annealing, International Journal of Advanced Manufacturing Technology 32 (3) (2007) 326–335, ISSN 0268-3768.