Soft Comput
DOI 10.1007/s00500-013-1136-1

METHODOLOGIES AND APPLICATION

A block based estimation of distribution algorithm using bivariate model for scheduling problems

Pei-Chann Chang · Meng-Hui Chen

Abstract Recently, estimation of distribution algorithms (EDAs) have gradually attracted a lot of attention and have emerged as a prominent alternative to traditional evolutionary algorithms. In this paper, a block-based EDA using bivariate model is developed to solve combinatorial problems. Instead of generating a set of chromosomes, our approach generates a set of promising blocks using bivariate model and these blocks are reserved in an archive for future use. These blocks will be updated every other k generation. Then, two rules, i.e., AC1 and AC2, are developed to generate a new chromosome by combining the set of selected blocks and rest of genes. This block based approach is very efficient and effective when compared with the traditional EDAs. According to the experimental results, the block based EDA outperforms EDA, GA, ACO and other evolutionary approaches in solving benchmark permutation problems. The block based approach is a new concept and has a very promising result for other applications.

Keywords Combinatorial problems · Estimation of distribution algorithms · Bivariate probabilistic model · Artificial chromosomes

1 Introduction

Recently, the estimation of distribution algorithms (EDAs) has been one of the major evolutionary computing paradigms applied in solving the combinatorial optimization problems (Ceberio et al. 2012). EDAs provide another way to evaluate solutions instead of using the crossover and mutation. EDAs build a probabilistic model based on the statistical information extracted from the selected solutions and generate new solutions by sampling from the probabilistic model (Larrañaga and Lozano 2002). These new solutions are sampled from the model to replace the old population fully or in part. EDAs are good at solving hard problems when we do not have prior knowledge of the problems characteristics.

In order to obtain the better performance, some researchers have attempted to combine local heuristics with evolutionary algorithm. Chang et al. (2008a) propose a genetic algorithm by injecting artificial chromosomes (ACs) into new population to solve the single machine scheduling problems. A roulette wheel selection method is applied to generate an artificial chromosome by assigning genes onto each position according to the probability model. By injecting these artificial chromosomes, the genetic algorithm will speed up the convergence of the evolutionary processes. Other heuristics combining with different nature inspired approaches are also developed to solve permutation flow-shop scheduling problems (PFSP) such as (Ahmadizar 2012; Costa et al. 2012; Pen and Ruiz 2012; Dong et al. 2013; Tasgetiren et al. 2011).

In this paper, a block-based EDA using bivariate model is developed to solve combinatorial problems. Instead of generating a set of chromosomes, our approach generates a set of promising blocks using bivariate model and these blocks are reserved in an archive for future use. These blocks will be updated every other k generation. Then, a new solution is generated by combining the set of selected blocks and rest of genes. The block based approach is very efficient and effective when compared with the traditional EDAs.

Communicated by Y. Jin.

P.-C. Chang (✉) · M.-H. Chen
Department of Information Management, Yuan Ze University, Chung-Li, Taoyuan 32026, Taiwan, ROC
e-mail: iepchang@saturn.yzu.edu.tw

Published online: 06 October 2013

2 Literature review

2.1 Flow-shop scheduling problems

Permutation flow-shop scheduling problems are different kinds of the combinatorial problems. To find the solution of an optimization problem is a common challenge in which an algorithm may be trapped in the local optima of the objective function when the complexity is high, and there are several local optima in solution space.

According to Baker (1974) and Chang et al. (2008b) flow-shop scheduling problem is a scheduling problem of combinatorial issues. Garey and Johnson (1979) also categorized these kinds of problems as NP-hard (non-deterministic polynomial-time hard). NP-hard in computational complexity theory, is a class of problems that are "at least as hard as the hardest problems in NP", meaning that the cost will be enormous to reach the best permutation of the PFSP problem. Therefore, most of the research works emerged to develop effective heuristics and meta-heuristics.

Flow-shop scheduling can be defined as follows: if p (i, j) is the processing time for Job i on Machine j, and a job permutation is represented by $\{\pi_1, \pi_2, \ldots, \pi_n\}$, when there are n jobs and m machines, the completion times C (π_i, j) is calculated as the follows:

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

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

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

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

The makespan is defined as:

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

There are some important additional conditions to this problem:

- Operations are independent and available for processing at time zero.
- Each machine i processes at most a job j at a time.
- M machines are continuously available.
- All jobs j can be processed only on one machine i at a time.
- Setup and removal times are independent from process sequence and are included in the processing times.

There are many other criteria that can be considered for the purpose of optimization. We refer the reader to Bagchi (1999) for a detailed discussion of scheduling using GA. For details of the flow-shop as well as other scheduling and sequencing problems, we refer the reader to Baker (1975). As proposed by Reeves (1995), a more general flow-shop scheduling problem can be defined by allowing the permutation of jobs to be different on each machine. However, the more general flow-shop scheduling problem tends to show only small improvement in solution quality over the PFSP, while it substantially increases the complexity of the problem. The size of the solution space increases from n! to (n!)m. Other objective functions for PFSP have also received a lot of attention, such as the mean flow-time (the time a job spends in the process), or minimizing the mean tardiness (assuming some deadline for each job).

Some of the PFSP problems in manufacturing industries like their jobs may have non-identical release dates, and there may be sequence-dependent setup times, and limited buffer storage between machines and so on. The problem in real world will make the problem more complex. However, genetic algorithm provides a more realistic view to solve the problem. Since it can generate alternatives of sequences to the decision maker and a more applicable sequence can be decided to solve the current problem with satisfactory results.

2.2 Estimation of distribution algorithms

In EDAs, the problem specific interactions among the variables of individuals are taken into consideration. The two main steps of EDAs are to estimate the probability distribution of selected solutions and to generate new chromosomes by sampling this probability distribution. In Evolutionary Computations the interactions are kept implicitly in mind whereas in EDAs the interrelations are expressed explicitly through the joint probability distribution associated with the individual variable selected at each generation. The probability distribution is calculated from a database of selected individuals from the previous generation. An offspring is obtained from the sampling of this probability distribution. Neither crossovers nor mutations have been applied in EDAs. However, to derive the estimation of the joint probability distribution associated with the database containing the selected individuals is a challenge task. The flowchart of EDA is shown in Fig. 1.

Paul and Iba (2002) proposed three probability distribution models. In this paper, independent uni-variate marginal distribution is estimated from marginal frequencies as shown in the following:

$$pl(X_i) = \frac{\sum_{j=1}^{N} \delta_i(X_i = x_i | D_{i-1}^{Se})}{N} \tag{6}$$

$\delta_i(x_i = x_i | D_{i-1}^{Se}) = 1$ if in jth individual x_i has its ith value; otherwise it is zero.

Different EDAs use different models for the estimation of probability distribution. Population based incremental learn-

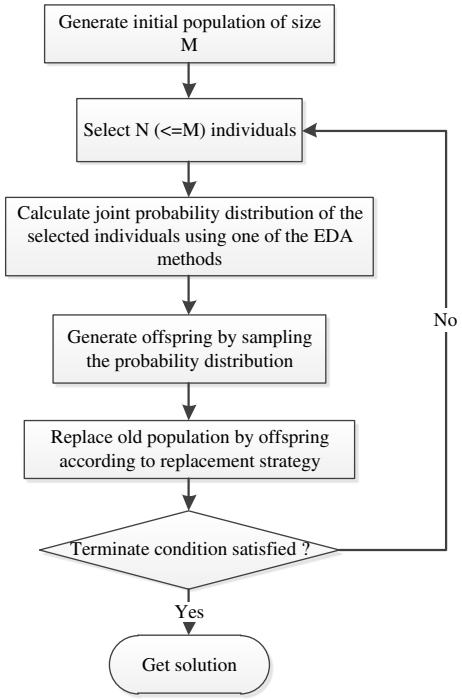

Fig. 1 EDA flowchart

ing (PBIL) proposed by Baluja (1994) and compact genetic algorithm (CGA) proposed by Harik et al. (1999) treat each variable of solutions as independent from one another. As a result, n-dimensional joint probability distribution factorizes as a product of n univariate and independent probability distributions. PBIL and CGA both use a probability vector while the update rules of probability vector of PBIL and CGA are different.

In order to estimate the relationship between variables, a bivariate dependencies model is built. Zhang (2004) proposed univariate marginal distribution algorithm (UMDA) and another factorized distribution algorithm (FDA) for discrete optimization problems. He introduced the heuristic functions and the limit models of these two algorithms and analyzed the stability of these limit models. Higher order statistics is used to improve the chance of finding the global optimal solution. For applications of UMDA with Laplace corrections in binary and permutation domains is proposed by Paul and Iba (2002).

Edge histogram sampling algorithm (EHBSA) is a local search mechanism proposed by Tsutsui (2002). He proposed probabilistic model-building genetic algorithms (PMBGAs) in permutation representation domain using edge histogram based sampling algorithms (EHBSAs). He also proposed two types of sampling mechanisms, EHBSA/WO without template and EHBSA/WT with template which execute based on different situations. Tsutsui et al. (2006) modify his previous research using the EHBSA and proposes another histogram based model which called the node histogram sampling algorithm (NHBSA). In this paper we employee EHBSA be one of local search mechanism.

In recent researches, some researchers hybridize EDA with other Meta-heuristic algorithms to form a powerful algorithm. Chen et al. (2012) proposed a new probability model by combining the uni-variate and bivariate probability models of EDA together. They define a new binary variable to indicate relationship between a pair of jobs. This variable collects the job which is immediately following another job in each chromosome. By combining the uni-variate probability model with bivariate probability model, a new probability model is formed which contain two different statistical information. The new probability model is used to generate new population. This hybrid heuristic algorithm is able to obtain good performance on flow-shop problem. Tzeng et al. (2012) proposed an algorithm hybridizing EDA with ant colony system (ACS) for the minimization of makespan in PFSP. They propose a new filter strategy and local search method to evaluate the solution and to use these solutions to generate new pheromone trails. A solution construction method of ACS is applied to generate members for the next iteration. In addition, they developed a new jump strategy to create more diversity for the solution to escape local optimal. Liu et al. (2011) proposed an algorithm to solve PFSP that combined particle swarm optimization (PSO) with EDA. The combination of the EDA procedure with PSO enables the sharing of information from the collective experience of the swarm and use a local optimal mechanism to increase the primitive intelligence for each particle.

In this paper, a bivariate probabilistic model is employed to generate a set of promising blocks instead of chromosomes. Then, these blocks will be recombined with rest of genes to form a legal chromosome. The new population generated from this block based approach has a very good fitness performance when compared with the traditional EDAs. In addition, a new local search mechanism is also proposed to speed up the convergence of the block based EDA.

3 A block based estimation of distribution algorithm

Basically, in BBEDA, we use a gene linkage probability matrix to identify a set of suitable blocks which is a small part of the chromosome to be recombined into a complete legal chromosome. In order to understand the behavior of block based evolution from the point of view of blocks creation and composition, there is a need for a very simple and direct representation of blocks. The block consists of a series of genes linked to each other continuously. To mine the blocks from the set of high fit chromosomes, two methods can be applied: static block size, on which equally sized blocks are created or dynamic block size where blocks are created with random sizes. In this research, we will focus on static block

size. In order to maintain diversity as well as to encourage the proper recombination, the size of block is very important while dealing with the problem with different number of jobs.

The first step is to collect the top M chromosomes in fitness performance generated from the initial solutions. Then, convert the statistical information from those M chromosomes into dominance matrix and dependency matrix Chen et al. (2012). A set of blocks then can be generated using this bivariate probabilistic model. These blocks will be ranked according to the sum of the probability in each position. A fixed number of blocks will be maintained in the archive.

In the second step, two different types of combinations mechanism of artificial chromosome are developed. Using these blocks, ACs are generated by combining blocks and the rest of genes together. In the third step, we modify the edge histogram based on the sampling algorithm with template (EHBSA) mechanism form EDA proposed by Tsutsui (2002). The local search strategy is named as mEHBSA and is executed in the later iterations. The mEHBSA can speed up the convergence of BBEDA.

For further explanation of the flow, the pseudo-code is given as follows.

1. Initial population();
2. Calculate fitness();
3. While stopping criterion is not satisfied.

   3.1 If refresh counter > threshold$_M$ then
   Refresh dominance and dependency matrix.
   Endif
   3.2 Update dominance and dependency matrix by best N % of population.
   3.3 If counter of generate AC > threshold$_A$ then
   Mining blocks and generate
   artificial chromosomes.
   Endif

4. Execute mEHBSA();
5. Calculate fitness and sorting population by Cmax.
6. Use tournament selection to choose chromosomes to next iteration.
7. Endwhile

The flowchart of BBEDA is represented in Fig. 2.

As for the detailed procedure of each step, i.e., the generation of dominance and dependence matrix, the block mining process, and the modified EHBSA, they are explained as follows:

3.1 Dominance and dependency matrix

In this research, BBEDA applies two mechanisms to establish the probabilistic models sampling from the set of top

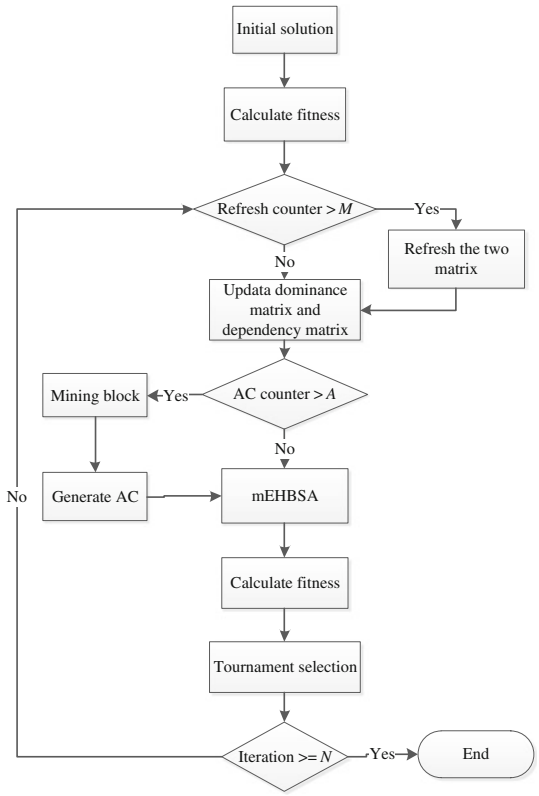

Fig. 2 The architecture of BBEDA

M chromosomes. One is dominance matrix, and another is dependency matrix. By combining these two matrices with different ration, the blocks and AC generated will be more diversified.

Dominance matrix emphasizes the relationship of jobs between positions. The top M solutions are selected and accumulated in the archive. Then, the statistical information, i.e., the frequencies of each job shown up in each position, is calculated in the dominance matrix in each generation. The composition of the matrix is updated by the accumulation of the probability.

To build up the dominance matrix, a set of m better chromosomes (C^1, C^2, ..., C^m) are selected from the current generation t. X_{ij}^k is a binary variable and can be treated like a gene within chromosome C^k (As Eq. 7).

$$X_{ij}^k = \begin{cases} 1 & if\ job\ i\ at\ position\ j \\ 0 & otherwise \end{cases},$$
$$i = 1 \cdots n;\ j = 1 \cdots n;\ k = 1 \cdots m \tag{7}$$

Then the $\emptyset_{ij}(t)$ in Eq. 8 which represents the number of times that job i at position j at the current generation t is found from summing up the statistic information from all m chromosomes to the X_{ij}^k. The total number of generations is G.

$$\emptyset_{ij}(t) = \sum_{k=1}^{m} X_{ij}^{k},$$
$$i = 1 \cdots n;\ j = 1 \cdots n;\ t = 1 \cdots G;\ k = 1 \cdots m \quad (8)$$

We transform the dominance matrix into dominance probability matrix, and each element of the dominance probability matrix is defined as follows:

$$P_{ij}(t) = \frac{\emptyset_{ij}(t)}{m}\ i = 1 \cdots n;\ j = 1 \cdots n;\ t = 1 \cdots G \quad (9)$$

A schematic diagram of the dominance matrix and dominance probability matrix is shown in Fig. 3.

The relationship of jobs between different jobs also has a significant influence for the new chromosomes to be generated, therefore, we established dependency matrix to store the information. In BBEDA, the dependency matrix is established in the following.

Each element in the dependency matrix and the dependency probability matrix is defined as in Eq. 10. To build up the dependency matrix, a set of m better chromosomes (C^1, C^2, ..., C^m) are selected from the current generation t. Y_{ij}^k is a binary variable and can be treated like a gene within chromosome C^k .

$$Y_{ij}^{k} = \begin{cases} 1 & if\ job\ i\ is\ next\ to\ job\ j \\ 0 & \text{otherwise} \end{cases},$$
$$i = 1 \cdots n;\ j = 1 \cdots n;\ k = 1 \cdots m \quad (10)$$

Then $\theta_{ij}(t)$ in Eq. 13 which represents the number of times that job i is next to job j at the current generation t is found from summing up the statistic information from all m chromosomes to the Y_{ij}^k. The total number of generations is G.

$$\theta_{ij}(t) = \sum_{k=1}^{m} Y_{ij}^{k},\ i = 1 \cdots n;$$
$$j = 1 \cdots n;\ t = 1 \cdots G;\ k = 1 \cdots m \quad (11)$$

We transform the dependency matrix into dependency probability matrix, and each element of the dependency probability matrix is defined as follows

$$P_{ij}(t) = \frac{\theta_{ij}(t)}{m}\ i = 1 \cdots n;\ j = 1 \cdots n;\ t = 1 \cdots G \quad (12)$$

A schematic diagram of the dependency matrix and dependency probability matrix is shown in Fig. 4.

In addition, the value of every element in dominance matrix and dependency matrix is set by 0.1 from the beginning (accumulated any solutions yet). The purpose is to increase the diversity of the new blocks and ACs generated. The blocks and ACs are generated by the roulette wheel selection method. Some position of the jobs are never accumulated which give them a better chance to be selected.

To sum up, a block is a more focused area within the chromosome to be looked into closely. Since, there are more chances that these blocks might be included in the final optimal solution. Of course, these blocks have to be evolved generation by generation. As a result, the new chromosome generated by recombining these blocks and the rest of genes will be also improved iteratively through the evolution procedure. In addition, by combining these two matrices, i.e., the dominance and dependency probability matrix, a set of more diversified blocks will be generated.

3.2 A block based approach

Combinatorial optimization problems (COPs) are intractable owing to the computational complexities. Different Metaheuristics have been proposed to solve COPs. Problem

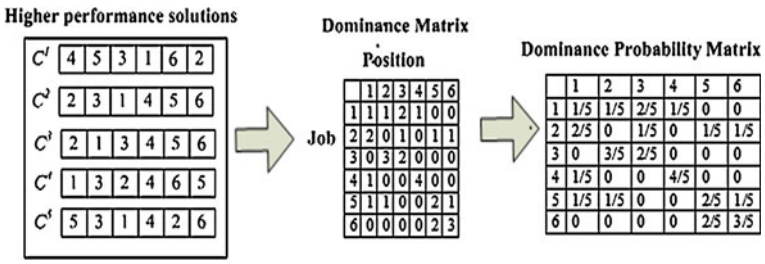

Fig. 3 Dominance matrix and dominance probability matrix

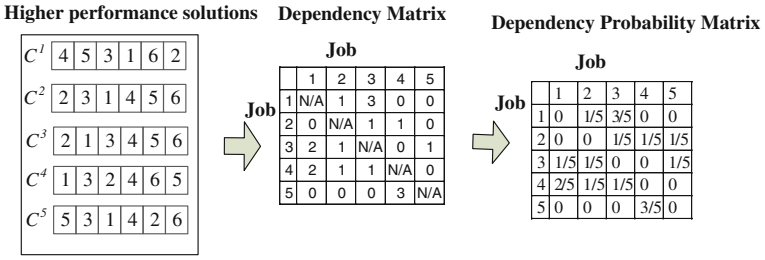

Fig. 4 Dependency matrix and dependency probability matrix

decomposition is one of the feasible approaches to reduce the problem complexities. Our attempt is to decrease the complexity of COPs by proposing blocks as virtual jobs and these virtual jobs certainly can reduce the problem dimensionality. Then, a better solution can be reached using the same amount of computation resources. For example, a single machine scheduling problem with ten jobs, it has 10! feasible solutions. As shown in Fig. 5, if we could find some information to combine a set of jobs to be grouped together. For example, job 1 with job 8, job 4 with job 7, and job 10 with job 5, and the total number of feasible solutions will be 7! only. If a longer block is defined, the total number of feasible solutions will be reduced even more abruptly. However, the quality of the blocks will be the key to the block based approach.

According to previous researches, EDA use different kinds of probabilistic models to generate new solutions. Instead of generating chromosomes, we propose a mechanism which is called "block evolving" to decrease the complexity of COPs. We transform the dominance and dependency matrix into a probability matrix, and using the probability matrix to generate blocks. There are two advantages to using blocks instead of a chromosome. One is as mentioned above; blocks when compared with genes can decrease the complexity of COPs. The other is blocks are more effective than chromosomes in terms of convergence speed. To generate a chromosome, the probability model will be applied n times if the chromosome length is n. To generate a block instead, the probability model will be applied only m times if the block size is m. For example as shown in Fig. 6, an instance of probability matrix shows the probability of each job assigned in each position. A chromosome is generated with the sequence of (3, 2, 5, 4, 1) and the probability will be $0.2 \times 0.2 \times 0.1 \times 0.2 \times 0.1$. However, if we generate a block with the size of 2, and the block is (5, 4) since job 5 has the largest probability in position one and job 4 in position 2. The probability of generating block (5, 4) will be 0.6×0.6 which is much larger than the probability of generating chromosome (3, 2, 5, 4, 1). After generating a set of high quality blocks, we can recombine them with the rest of genes and to form a legal chromosome. The quality or probability of this chromosome is better or higher than that of the traditional EDA.

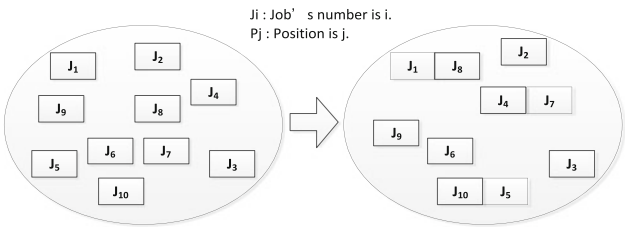

Fig. 5 Complexity reduction

Probability Matrix

	P_1	P_2	P_3	P_4	P_5
J_1	1/5	1/5	3/5	0	0
J_2	0	1/5	0	4/5	0
J_3	1/5	0	2/5	0	2/5
J_4	0	3/5	0	1/5	1/5
J_5	3/5	0	0	0	2/5

Fig. 6 An instance of a Probability matrix

But, relatively speaking, it is noteworthy that the block size cannot be too long either. For example, the probability of job i in correct position is x_i, the probability of jobs j in correct position is x_j and the probability of jobs k in correct position is x_k. A block performance for a pair of jobs i and j, is defined as follows:

$$H_{i,j} = \mathrm{p}(x_i, x_j) = p(x_i)p(x_j) \tag{13}$$

Then another block for a three of jobs i, j and k, is defined as follows:

$$H_{i,j,k} = \mathrm{p}(x_i, x_j) = p(x_i)p(x_j)p(x_k) \tag{14}$$

Since $\mathrm{p}(x_i)$, $\mathrm{p}(x_j)$ and $\mathrm{p}(x_k)$ are all smaller than or equal to one, the probability of $B_{i,j,k}$ is smaller than or equal to $B_{i,j}$.

A block with size L can be generated according to the following procedures:

$U_J = \{J_1, J_2, \ldots, J_n\}$: A set of unselected jobs and $U_G = \{G_1, G_2, \ldots, G_n\}$: A set of unselected positions.
$S_j = \emptyset$: Scheduled job.
L: block size.
B: block archive.
Step1. According to the dominance and dependency probability matrix, randomly select a G_j from U_G and using RWS to select J_i from U_j for G_j. $U_G = U_G / G_j$, $U_j = U_j / J_i$, $S_j = S_j \cup J_i$.
Step2. $j = j + 1$. Then using RWS to select J_i from U_j for G_j. $U_G = U_G / G_j$, $U_j = U_j / J_i$, $S_j = S_j \cup J_i$.
Step3. Repeat step2 until $j = L - 1$. A block is generated and reserve the block in B with the best fitness.

A schematic diagram of the block mining procedure is shown in Fig. 7.

After block mining, two different rules using block are designed to generate ACs. These two rules are called AC1 and AC2. Some symbols and the procedure of AC1 and AC2 are as follows:

J_i: Job number is i.
P_j: position is j.
U: Unscheduled jobs, U = {1, 2, …, n }.
S: Scheduled jobs, $S = \{\phi\}$.
B: Block archive, $B = \{b_1, b_2, \ldots\}$.
AC1:
Step1. Select a J_i by using RWS and put it in homologous position, then $U = U/\{i\}$, $S = S + \{i\}$. If i is a starting job of a b_m, select the block and $B = B/\{b_m\}$

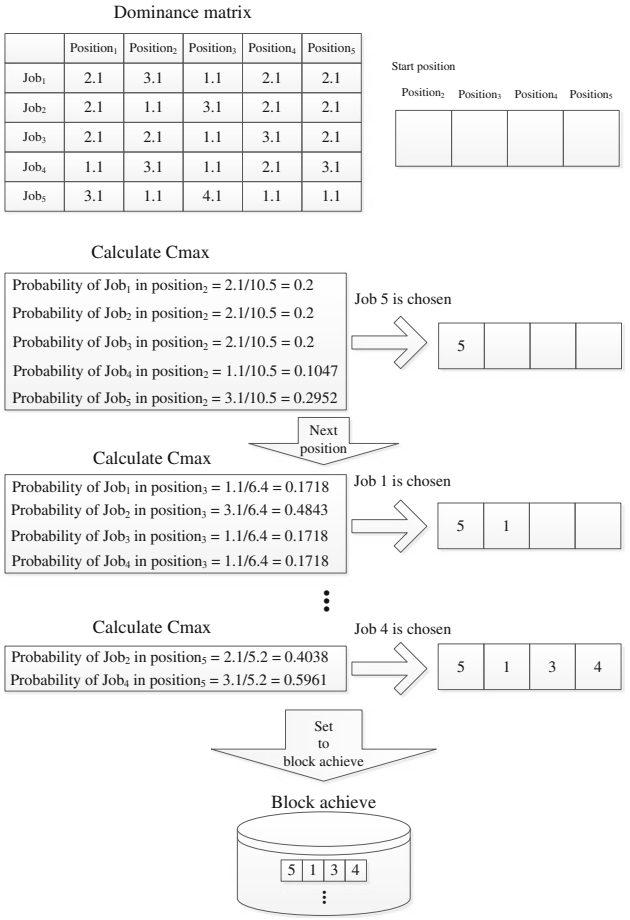

Fig. 7 Block mining procedure

Step2. If U = ϕ, stop. Otherwise, go to step1.

AC2:

Step1. Put all blocks in the homologous position, and B = ϕ.

Step2. Select a J_i by using RWS and put it in homologous position, then U = U/{i}, S = S + {i}.

Step3. If U = ϕ, stop. Otherwise, go to step2.

The reasons for using AC1 and AC2 to generate ACs are for increasing the diversity and convergence speed simultaneously. AC1 only applying part of the blocks will have more diversity than AC2. AC2 using all blocks will be focused only for convergence speed.

As shown in Fig. 8, there are two different situations when applying AC1 to generate new chromosomes. One is that when jobs of blocks are not selected in the homologous positions, then the blocks will not be selected afterwards in this generation. The other is when the job is a starting job in a block and in the homologous position. The block will be placed into the homologous position of the chromosome.

Otherwise, as shown in Fig. 9, AC2 puts all blocks into homologous positions in the beginning, and then different

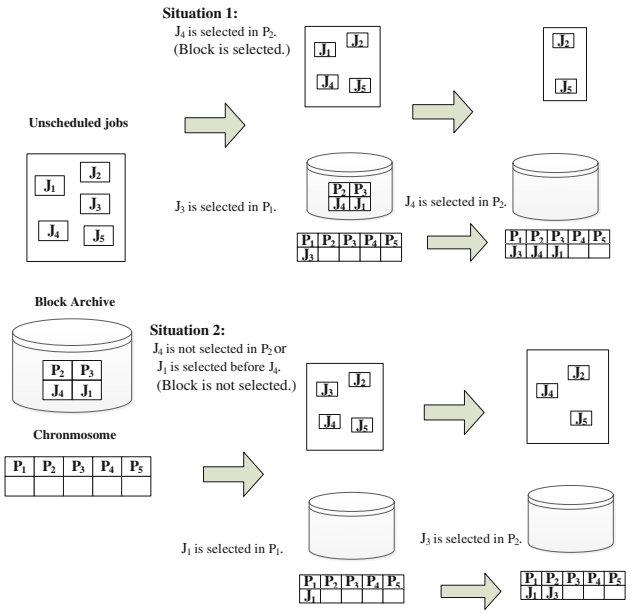

Fig. 8 AC1 mechanism

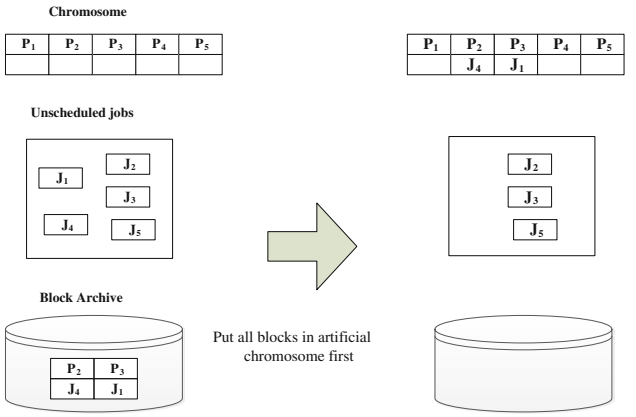

Fig. 9 AC2 mechanism

jobs are selected into each empty position from unscheduled jobs by roulette wheel selection (RWS) method.

AC1 and AC2 both use roulette wheel selection to select jobs by these probabilities. Each job's probability is based on the hybrid probability that merges dominance and dependency probability matrix. The combination probability (CPjob n) is defined as follows:

$$CP_{jobn} = (W_{dom} * P_{jobn}^{dom}) + (W_{dep} + P_{jobn}^{dep}) \quad (15)$$

The value of weight of dominance and dependency probability matrix (W_{dep}) are defined by user. In this research, the value of W_{dep} is defined between 0.3 and 0.7. W_{dep} increases as generation goes on. Number of W_{dep} is one minus W_{dep}.

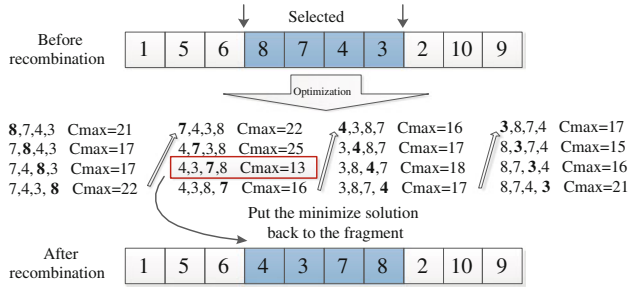

Fig. 10 A schematic diagram of the mEHBSA procedure

3.3 Modified EHBSA

EHBSA is an important heuristic in improving the quality of chromosomes generated by EDA. EHBSA/WT uses a template in sampling a new string from the selected chromosome. When n cut points are obtained for the template, the template should be divided into n segments. Then, we choose one segment randomly and sample nodes for the segment. Nodes in other n-1 segments remain unchanged. This sampling method is denoted by EHBSA/WT/n.

In this paper, we propose a modified rule instead of the original EHBSA/WT. The detail of mEHBSA is described in the followings:

Step1. Randomly select a segment from the chromosome.
Step2. Starting from the first gene of the segment, switch each gene with the next gene in order until it reaches the last position.
Step3. Calculate the fitness of the new chromosome after each exchange.
Step4. Then, the procedure repeats again based on the new chromosome generated in the last round of switch in step 3.
Step3. Select the chromosome with the best fitness among these new generated chromosomes as a new AC.

A schematic diagram of the mEHBSA procedure is shown in Fig. 10.

4 Experimental results

In this section the experimental results of BBEDA are presented and compared to the performance of other algorithms. The test instances for comparisons are adopted from Reeves[1] and Taillard[2] in OR-Library to validate the performance of BBEDA. Each instance is executed for thirty runs. The job number is n and the machine number is m. The population size is set to 100, and the termination criteria is set to when the total number of chromosomes generated is equal to job number * machine number * 50 in Reeves instance. The termination criteria for other instances are attributed to PSO-Lian et al. (2006). The EDA to be compared with in this research uses univariate probability model and the sampling method is applying the solutions with top 20 % performance.

The error rate of each algorithm is defined as follows:

$$Error\ rate(\%) = \frac{observed\ value - optimal\ (opt.)\ value}{optimal\ value} \times 100\,\% \quad (16)$$

Table 1 shows the performance comparison on Taillard's instances. The comparison standard is based on Chang et al. (2010). From the result, the average error rate of BBEDA is 0.62 % which outperforms the other algorithms.

Table 2 shows ANOVA result of BBEDA when compared with AC2GA, PSO-Lian et al. (2006) and HGIA individually. The factor algorithms have a P-values are 0.002. Therefore, all these algorithms which listed in Table 1 performed very differently.

Table 3 shows the performance comparison on Reeves's instances. The comparison standard is based on Chang et al. (2011). The average error rate is 0.60 % which outperforms the other three algorithms. It can be concluded that BBEDA performs the effective searching ability on different problems and complexities.

Table 4 shows the performance comparison on Taillard's instances however, the comparison standard is based on Chen and Chen (2013). Again, the average error rate of BBEDA is 1.42 % which outperforms the other algorithms.

Table 5 shows ANOVA result of BBEDA when compared with algorithms listed in Table 4 individually. The factor algorithms have a P-values are 0.000. Therefore, all these algorithms which listed in Table 4 performed very differently.

Tables 6 and 7 show the result for testing BBEDA integrated with AC1or AC2 on Taillard's instances without local search strategy. The performance of BBEDA combined with AC1 has the best performance with the average error rate of 3.71 %. It shows that block mechanism and different recombination strategy can be helpful in increasing the convergence ability.

The three methods do not include any local search strategy. The result shows a significant difference in average value of error ratio and minimum value of error rate between BBEDA-AC1, BBEDA-AC2 and the original EDA. The result also proves that block based approach and different recombination strategies have effectively reduced the problem size and achieved good convergence ability.

Tables 8 and 9 show the performance comparisons for EDA, BBEDA+AC1 and BBEDA+AC2 with local search on Taillard's instances.

[1] http://people.brunel.ac.uk/~mastjjb/jeb/orlib/files/flowshop1.txt.

[2] http://mistic.heig-vd.ch/taillard/problemes.dir/ordonnancement.dir/ordonnancement.html.

Table 1 Performance comparison on Taillard's instances

Ins.	opt	BBEDA		HGIA		PSO-Lian		AC2GA	
		Min	Error_rate (%)	Min	Error_rate (%)	Min	Error_rate (%)	Min	Error_rate (%)
ta005	1,235	1,235	0.00	1,235	0.00	1,235	0.73	1,235	0.00
ta010	1,108	1,108	0.00	1,108	0.00	1,108	0.00	1,108	0.00
ta020	1,591	1,595	0.25	1,598	0.44	1,617	0.57	1,617	1.63
ta030	2,178	2,179	0.05	2,186	0.37	2,196	0.73	2,196	0.83
ta050	3,065	3,091	0.85	3,111	1.50	3,171	3.52	3,171	3.46
ta060	3,696	3,836	3.79	3,823	3.44	3,910	5.98	3,910	5.79
ta070	5,322	5,322	0.00	5,328	0.11	5,324	0.38	5,324	0.04
ta080	5,845	5,845	0.00	5,848	0.05	5,893	0.99	5,893	0.82
Avg.			0.62		0.74		1.61		1.57

Table 2 ANOVA results on the error rate of the final solutions obtained by the algorithms listed in Table 1 on the 8 instances

Source	DF	SS	MS	F	P
Algorithms	3	6.745×10^{-4}	2.248×10^{-4}	6.87	0.002
Instance	7	7.4926×10^{-3}	1.0704×10^{-3}	32.73	0.000
Error	21	6.868×10^{-4}	3.27×10^{-5}		
Total	31	8.8539×10^{-3}			

As shown in Figs. 11 and 12, these three algorithms, i.e., EDA, EDA-AC1, and EDA-AC2, are not applied with local search rules when solving Ta030 and Ta060 problems. The convergence speed of AC1 and AC2 are both better than EDA only. The reason is because blocks can effectively decompose the original problem sizes and these blocks are also very effective when recombined with the rest of genes. And as shown in Figs. 13 and 14, these three algorithms are also used with local search strategy, i.e., mEHBSA. The result also shows that the convergence speed of AC1 and AC2 are still

Table 3 Performance comparison on Reeves's instances

Ins.	n, m	opt	BBEDA		SGA		AC2GA		HGIA	
			Min	Error_rate (%)	Min	Error_rate (%)	Min	Error_rate (%)	Min	Error_rate (%)
Rec01	20, 5	1,247	1,247	0.00	1,249	0.16	1,249	0.16	1,247	0.00
Rec03	20, 5	1,109	1,109	0.00	1,111	0.18	1,109	0.00	1,109	0.00
Rec05	20, 5	1,242	1,242	0.00	1,245	0.24	1,245	0.24	1,245	0.24
Rec07	20, 10	1,566	1,566	0.00	1,583	1.09	1,566	0.00	1,566	0.00
Rec09	20, 10	1,537	1,537	0.00	1,565	1.82	1,537	0.00	1,537	0.00
Rec11	20, 10	1,431	1,431	0.00	1,456	1.75	1,431	0.00	1,431	0.00
Rec13	20, 15	1,930	1,930	0.00	1,970	2.07	1,930	0.00	1,932	0.10
Rec15	20, 15	1,950	1,950	0.00	1,990	2.05	1,951	0.05	1,951	0.05
Rec17	20, 15	1,902	1,902	0.00	1,960	3.05	1,902	0.00	1,924	1.16
Rec19	30, 10	2,093	2,099	0.29	2,162	3.30	2,099	0.29	2,103	0.48
Rec21	30, 10	2,017	2,036	0.94	2,064	2.33	2,021	0.20	2,046	1.44
Rec23	30, 10	2,011	2,020	0.45	2,075	3.18	2,021	0.50	2,020	0.45
Rec25	30, 15	2,513	2,530	0.68	2,623	4.38	2,515	0.08	2,530	0.68
Rec27	30, 15	2,373	2,379	0.25	2,461	3.71	2,387	0.59	2,394	0.88
Rec29	30, 15	2,287	2,292	0.22	2,392	4.59	2,289	0.09	2,301	0.61
Rec31	50, 10	3,045	3,056	0.36	3,207	5.32	3,101	1.84	3,077	1.05
Rec33	50, 10	3,114	3,114	0.00	3,162	1.54	3,131	0.55	3,115	0.03
Rec35	50, 10	3,277	3,277	0.00	3,280	0.09	3,277	0.00	3,277	0.00
Rec37	70, 20	4,951	5,111	3.23	5,251	6.06	5,190	4.83	5,084	2.69
Rec39	70, 20	5,087	5,189	2.01	5,301	4.21	5,285	3.89	5,150	1.24
Rec41	70, 20	4,960	5,115	4.26	5,263	6.11	5,164	4.11	5,097	2.76
Avg.				0.60		2.73		0.83		0.66

Table 4 Performance comparison on Taillard's instances based on Chen and Chen (2013)

Instance	EHBSA WT	EHBSA WO	NHBSA WT	NHBSA WO	PRE DA	JED A	SG GA	eSG GA	BBE DA
ta001	0.30	1.41	1.27	1.49	1.2	1.16	1.2	1.16	0.02
ta002	0.05	0.49	0.27	0.31	0.88	0.49	0.46	0.36	0.25
ta003	1.3	3.04	0.71	1.23	2.04	1.73	1.19	0.50	0.00
ta004	0.19	1.81	0.22	1.17	0.78	0.7	0.9	0.48	0.21
ta005	0.51	1.17	0.96	1.17	0.82	1.21	0.92	0.7	0.24
ta006	0.00	1.26	0.53	1.13	1.06	1.54	1.36	1	0.14
ta011	0.53	1.52	0.58	1.11	0.95	0.56	0.82	0.71	0.27
ta012	0.88	3.09	0.81	1.16	1.24	1.1	0.53	0.28	0.77
ta013	1.1	2.08	1.18	1.77	1.53	1.5	1.18	0.82	0.83
ta014	0.37	2.27	0.65	1.26	1.31	1.15	0.77	0.87	0.34
ta015	0.44	0.99	0.58	1.23	0.98	0.93	0.84	0.66	0.75
ta016	0.55	1.56	0.73	1.32	0.87	1.5	0.9	0.61	0.78
ta041	3.5	9.18	3.74	4.51	3.57	3.61	2.95	3	2.39
ta042	3.52	8.97	3.88	4.98	2.34	3.3	3.16	2.75	1.66
ta043	4.16	10.68	4.29	4.25	3.38	3.9	3.13	2.74	2.47
ta044	1.81	7.2	1.72	1.66	1.3	1.98	1.4	1.05	0.32
ta045	3.72	9.34	3.58	2.42	3.09	3.84	3.29	2.15	1.64
ta046	2.69	7.47	2.9	4.22	2.36	3	2.75	2.35	0.99
ta081	8.18	14.51	7.84	8.8	5.53	5.85	5.96	5.46	3.81
ta082	6.51	13.05	5.84	6.35	4.02	4.05	4.02	3.37	3.60
ta083	6.46	12.26	5.81	6.44	3.77	3.91	3.81	3.29	3.28
ta084	5.82	11.98	5.15	5.32	2.95	3.67	3.44	3.19	2.62
ta085	6.8	12.45	6.52	7.4	3.97	4.46	4.4	4.05	3.27
ta086	6.81	12.43	6.68	7.45	4.27	4.58	4.41	3.99	3.30
Avg.	2.76	6.26	2.77	3.26	2.26	2.49	2.24	1.90	1.42

Table 5 ANOVA results on the error rate of the final solutions obtained by the algorithms listed in Table 4 on the 24 Instances

Source	DF	SS	MS	F	P
Algorithms	8	3.7459×10^{-2}	4.6823×10^{-3}	30.71	0.000
Instance	23	9.8077×10^{-2}	4.2642×10^{-3}	27.96	0.000
Error	184	2.8058×10^{-2}	1.525×10^{-4}		
Total	215	1.63594×10^{-1}			

Table 6 Average error rate comparison on Taillard's instances with AC1 or AC2

Instance	n, m	EDA (%)	BBEDA-AC1 (%)	BBEDA-AC2 (%)
ta005	20, 5	4.62	1.18	0.90
ta010	20, 5	6.60	2.02	2.37
ta020	20, 10	6.74	2.30	3.28
ta030	20, 20	7.63	3.19	3.93
ta050	50, 10	10.84	5.24	5.25
ta060	50, 20	15.42	9.22	10.02
ta070	100, 5	3.76	1.29	0.97
ta080	100, 10	7.08	5.22	4.90
Avg.		7.84	3.71	3.95

Table 7 Min error rate comparison on Taillard's instances

Instance	n, m	EDA (%)	BBEDA-AC1 (%)	BBEDA-AC2 (%)
ta005	20, 5	3.24	0.73	0.49
ta010	20, 5	4.96	0.63	1.08
ta020	20, 10	5.47	1.51	1.70
ta030	20, 20	5.46	1.61	2.62
ta050	50, 10	10.21	3.88	3.46
ta060	50, 20	14.10	5.68	6.85
ta070	100, 5	3.31	0.45	0.38
ta080	100, 10	6.14	3.83	2.86
Avg.		6.61	2.29	2.43

Table 8 Performance comparison on Taillard's instances with local search

Instance	n, m	EDA+LS (%)	BBEDA-AC1+LS (%)	BBEDA-AC2+LS (%)
ta005	20, 5	0.75	0.55	0.58
ta010	20, 5	0.33	0.23	0.15
ta020	20, 10	1.76	2.06	1.45
ta030	20, 20	3.12	1.77	2.15
ta050	50, 10	2.59	2.70	2.40
ta060	50, 20	6.76	6.55	7.26
ta070	100, 5	0.15	0.18	0.18
ta080	100, 10	0.67	0.66	0.66
Avg.		2.02	1.84	1.85

Table 9 Min error rate comparison on Taillard's instance with local search

Instance	n, m	EDA+LS (%)	BBEDA-AC1+LS (%)	BBEDA-AC2+LS (%)
ta005	20, 5	0.00	0.00	0.00
ta010	20, 5	0.00	0.00	0.00
ta020	20, 10	1.13	1.19	0.88
ta030	20, 20	1.70	0.41	0.83
ta050	50, 10	1.57	1.89	1.73
ta060	50, 20	4.98	4.46	5.87
ta070	100, 5	0.09	0.11	0.00
ta080	100, 10	0.21	0.14	0.53
Avg.		1.21	1.03	1.23

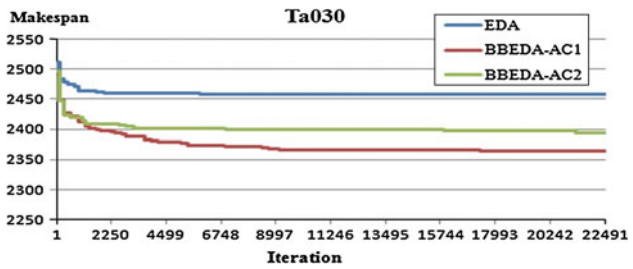

Fig. 11 Ta030 convergence graph

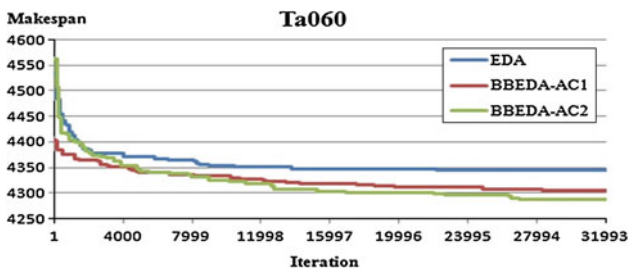

Fig. 12 Ta060 convergence graph

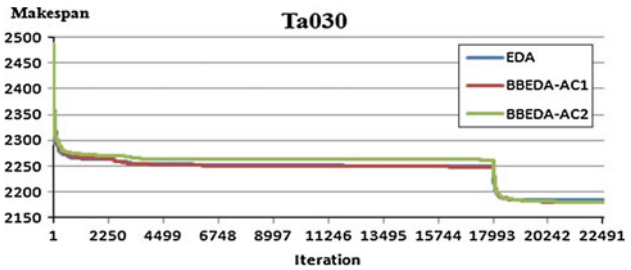

Fig. 13 Ta030 convergence graph with local search strategy

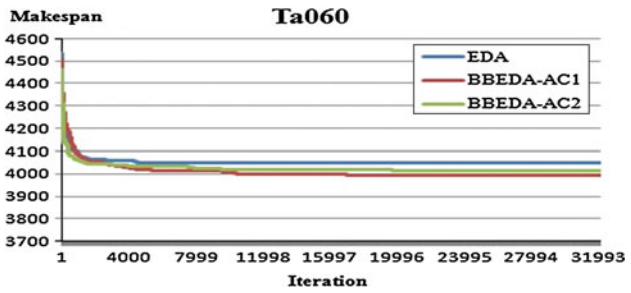

Fig. 14 Ta060 convergence graph with local search strategy

better than EDA only. As for the abrupt drop in the convergence diagram, that is because of the application of mEHBSA afterwards.

5 Conclusion

In this study, we presented a block based EDA approach using bivariate model for solving the flow shop scheduling problems. The blocks are basically the subset of the building blocks of the chromosomes and two rules, i.e., AC1 and AC2 are developed to generate new chromosomes. The quality of the blocks mined from the previous population has a great impact on behavior of BBEDA. In addition, the blocks can be continuously updated based on the combination of dominance and dependency matrices. The length of the block is kept constant and is varied as per the size of the problem in this research. It is also ensured that the block size is not large. If the block lengths are too big, they may contain redundant information which will be carried over throughout the evolutionary process and the final result may not be good. If the block length is too small, the information contained within each block may be too little and the recombination process may not be able to come out with good quality of chromosomes.

The main contributions of this paper were to demonstrate that the BBEDA can be successfully extended to deal with "hard" optimization problems such as the Flow shop and the effectiveness and efficiency are largely improved when compared with previous researches in solving benchmark flow shop problems. Two effective rules, i.e., AC1 and AC2 are

also developed to generate new chromosomes in combining these effective blocks. In addition, a modified EHBSA is also proposed to further fine tune the chromosomes generated by BBEDA. The experimental results indicate BBEDA is very effective and efficient. Future works can be extended in applying this method on several other combinatorial problems.

References

Ahmadizar F (2012) A new ant colony algorithm for makespan minimization in permutation flow shops. Comput Ind Eng 63(2):355–361

Bagchi TP (1999) Multiobjective Scheduling by Genetic Algorithms. Kluwer, Boston

Baker KR (1974) Introduction to sequencing and scheduling. Wiley, New York

Baker KR (1975) A comparative study of flow-shop algorithms. Oper Res 23(1):62–73

Baluja S (1994) Population based incremental learning: a method for integrating genetic search based function optimization and competitive learning. Technical Report No. CMU-CS-94-163, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA

Ceberio J, Irurozki E, Mendiburu A, Lozano J (2012) A review on estimation of distribution algorithms in permutation-based combinatorial optimization problems. Prog Artif Intell 1(1):103–117

Chang PC, Chen SH, Fan CY (2008a) Mining gene structures to inject artificial chromosomes for genetic algorithm in single machine scheduling problems. Appl Soft Comput J 8(1):767–777

Chang PC, Chen SH, Fan CY, Chan CL (2008b) Genetic algorithm integrated with artificial chromosomes for multi-objective flow-shop scheduling problems. Appl Math Comput 205(2):550–561

Chang PC, Huang WH, Ting CJ (2010) Self-evolving Artificial Immune System via Developing T and B Cell for Permutation Flow-shop Scheduling Problems. Proceedings of World Academy of Science, Engineering and Technology 65:822–827

Chang PC, Huang WH, Ting CJ (2011) A hybrid genetic-immune algorithm with improved lifespan and elite antigen for flow-shop scheduling problems. Int J Prod Res 49(17):5207–5230

Chen SH, Chen MC (2013) Addressing the advantages of using ensemble probabilistic models in estimation of distribution algorithms for scheduling problems. Int J Prod Econ 141(1):24–33

Chen YM, Chen MC, Chang PC, Chen SH (2012) Extended artificial chromosome genetic algorithm for permutation flowshop scheduling problems. Comput Ind Eng 62(2):536–545

Costa WE, Goldbarg MC, Goldbarg EG (2012) New VNS heuristic for total owtime owshop scheduling problem. Expert Syst Appl 39(9):8149–8161

Dong X, Chen P, Huang HK, Nowak M (2013) A multi-restart iterated local search algorithm for the permutation flow-shop problem minimizing total flow time. Comput Oper Res 40(2):627–632

Garey MR, Johnson DS (1979) Computers and Intractibility: a Guide to the Theory of NP-Completeness. Freeman, San Francisco

Harik GR, Lobo FG, Goldberg DE (1999) The compact genetic algorithm. IEEE Trans Evolut Comput 3(4):523–528

Larrañaga PJ, Lozano A (2002) Estimation of distribution algorithms: A new tool for evolutionary computation. Kluwer Academic Publishers, Boston

Lian Z, Gu X, Jiao B (2006) A similar particle swarm optimization algorithm for permutation flow-shop scheduling to minimize makespan. Appl Math Comput 175(1):773–785

Liu HC, Gao L, Pan QK (2011) A hybrid particle swarm optimization with estimation of distribution algorithm for solving permutation flow-shop scheduling problem. Expert Syst Appl 38(4):4348–4360l

Paul TK, Iba H (2002) Linear and Combinatorial Optimizations by Estimation of Distribution Algorithms. 9th MPS Symposium on Evolutionary Computation, IPSJ Symposium, Japan, pp 99–106

Pen QK, Ruiz R (2012) Local search methods for the flow-shop scheduling problem with flowtime minimization. Eur J Oper Res 222(1):31–43

Reeves CR (1995) A genetic algorithm for flow-shop sequencing. Comput Oper Res 22(1):5–13

Tasgetiren MF, Pan QK, Suganthan PN, Chen AH (2011) A discrete artificial bee colony algorithm for the total flowtime minimization in permutation flow-shops. Inf Sci 181(1):3459–3475

Tsutsui S (2002) Probabilistic model-building genetic algorithms in permutation representation domain using edge histogram. Lect Notes Comput Sci 2439:224–233

Tsutsui S, Pelikan M, Goldberg DE (2006) Node Histogram vs. Edge Histogram: a Comparison of PMBGAs in Permutation Domains. Missouri Estimation of Distribution Algorithms Laboratory, MEDAL Report No. 2006009, July

Tzeng YR, Chen CL, Chen CL (2012) A hybrid EDA with ACS for solving permutation flow-shop scheduling. Int J Adv Manuf Technol 60:1139–1147

Zhang Q (2004) On Stability of fixed points of limit models of univariate marginal distribution algorithm and factorized distribution algorithm. IEEE Trans Evol Comput 8(1):80–93