Expert Systems with Applications 37 (2010) 6441–6451

Contents lists available at ScienceDirect

Expert Systems with Applications

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

Guidelines for developing effective Estimation of Distribution Algorithms in solving single machine scheduling problems

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

[a] Department of Electronic Commerce Management, Nanhua University, No. 32, Chung Keng Li, Dalin, Chia-Yi 62248, Taiwan, ROC
[b] Institute of Manufacturing Engineering, National Cheng Kung University, Tainan 70101, Taiwan, ROC
[c] Department of Information Management, Yuan-Ze University, 135 Yuan-Dong Rd., Taoyuan 32026, Taiwan, ROC
[d] Department of Computing and Electronic Systems, University of Essex, Wivenhoe Park, Colchester CO4 3SQ, UK

A R T I C L E I N F O

Keywords:
Estimation of Distribution Algorithms
Intensification
Diversification
Single machine scheduling problems
Just-in-time

A B S T R A C T

The goal of this research is to deduce important guidelines for designing effective Estimation of Distribution Algorithms (EDAs). These guidelines will enhance the designed algorithms in balancing the intensification and diversification effects of EDAs. Most EDAs have the advantage of incorporating probabilistic models which can generate chromosomes with the non-disruption of salient genes. This advantage, however, may cause the problem of the premature convergence of EDAs resulted in the probabilistic models no longer generating diversified solutions. In addition, due to overfitting of the search space, probabilistic models cannot really represent the general information of the population. Therefore, this research will deduce important guidelines through the convergency speed analysis of EDAs under different computational times for designing effective EDA algorithms. The major idea is to increase the population diversity gradually by hybridizing EDAs with other meta-heuristics and replacing the procedures of sampling new solutions. According to that, this research further proposes an Adaptive EA/G to improve the performance of EA/G. The proposed algorithm solves the single machine scheduling problems with earliness/tardiness cost in a just-in-time scheduling environment. The experimental results indicated that the Adaptive EA/G outperforms ACGA and EA/G statistically significant in different stopping criteria. This paper, hence, is of importance in the field of EDAs as well as for the researchers in studying the scheduling problems.

© 2010 Elsevier Ltd. All rights reserved.

1. Introduction

Estimation of Distribution Algorithms (EDAs) is one of the most popular evolutionary algorithms in recent years (Aickelin, Burke, & Li, 2007; Baraglia, Hidalgo, Perego, Cnuce, & Cnr, 2001; Chang, Hsieh, Chen, Lin, & Huang, 2009b; Harik, Lobo, & Goldberg, 1999; Muhlenbein & Paaß, 1996; Zhang, Sun, & Tsang, 2005). EDAs explicitly learn and build a probabilistic model to capture the parental distribution, and then samples new solutions from the probabilistic model (Pelikan, Goldberg, & Lobo, 2002). 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 (Santana, Larrañaga, & Lozano, 2008). This is the most important characteristic to distinguish EDAs and Genetic Algorithms (GAs). As claimed by Zhang and Muhlenbein (2004), EDAs might be a promising method capable of capturing and manipulating the building blocks of chromosomes and, hence, efficiently solving hard optimization problems.

EDAs have been studied extensively. The concept originated from Ackley (1987) and Syswerda (1993) and was developed by Baluja (1995), Baluja and Davies (1998), and extended as cGA (Harik et al., 1999), UMDA (Muhlenbein & Paaß, 1996), GA-EDA (Peña et al., 2004), Guided Mutation (EA/G) (Zhang et al., 2005), Model-Based Evolutionary Algorithm (EA) (Zhou, Zhang, Jin, Tsang, & Okabe, 2005), Artificial Chromosomes with Genetic Algorithms (ACGA) (Chang et al., 2009b), Self-Guided GA (Chen, Chang, & Zhang, 2008b), and VNS·EDAs (Santana et al., 2008). These algorithms were classified as Estimation of Distribution Algorithms (EDA) or Evolutionary Algorithm with Probabilistic Model (EAPMs) by Zhang and Szeto (2005) and Pelikan et al. (2002). For a complete review of the algorithms discussed above, please refer to Larrañaga and Lozano (2002), Pelikan et al. (2002) and Lozano (2006).

Although the previous EDAs are promising to solve hard problems when we have no knowledge about the problems, researchers want to know how to design an effective EDAs or revise an existing EDAs. In addition, there are two cruxes of EDAs. First of all, EDAs may cause the problem of overfitting the search space and cannot represent the general information (Santana et al., 2008). Most

* Corresponding author. Tel.: +886 3 4352654; fax: +886 3 4559378.
E-mail addresses: shihhsin@mail.nhu.edu.tw (S.-H. Chen), cthunter@mail.wfc.edu.tw (M.-C. Chen), iepchang@saturn.yzu.edu.tw (P.-C. Chang), qzhang@essex.ac.uk (Q. Zhang), ymchen@mail.ncku.edu.tw (Y.-M. Chen).

0957-4174/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2010.02.073

importantly, due to the premature convergence of EDAs (Chang, Chen, & Fan, 2009a), the probabilistic models no longer generate diversified solutions resulted in poor performance.

This paper investigates the convergency speed of EA/G and ACGA. It is found the convergency speed of EA/G is rather fast; however, the probabilistic models no longer generates diversified individuals after certain generations which causes a problem of premature convergency. Although ACGA may not converge fast, it outperforms the EA/G significantly when sufficient computation time is allowed. After the analysis of convergency speed, we discovered interesting connections between intensification and diversification effects of EDAs and that alternatives with other algorithms. Several important guidelines are proposed and there are numerous possibilities for the researchers to develop effective searching algorithms. In order to show the usefulness of the guidelines, one guideline is implemented as an example of demonstration. The proposed algorithm is named Adaptive EA/G which extends the EA/G to solve the NP-Hard single machine scheduling problems with earliness/tardiness considerations in the just-in-time production environment.

The rest of the paper is organized as follows. Sections 2 and 3 are the brief explanations of EA/G and ACGA, respectively. In Section 4 is the review of the single machine scheduling problems that we study in this research. Section 5 illustrates the convergency speed of the two EDAs and some interesting guidelines are obtained through the convergency analysis. As a result, various guidelines are deducted to design effective searching algorithms and the application of these guidelines is explained in Section 6. In addtion, this paper proposes an Adaptive EA/G as an example to implement these guidelines. Section 7 is the experimental results whereas the Adaptive EA/G is evaluated by using the single machine scheduling problems. Finally, Section 8 gives conclusions of this paper.

2. Review of EA/G

One of the key issues in the design of evolutionary algorithms is how to generate effective offspring. The proximate optimality principle (Glover & Laguna, 1998), an underlying assumption in most (if not all) heuristics, assumes that good solutions have similar structure. This assumption is reasonable for most real-world problems, e.g., the percentage of common edges in any two locally optimal solutions of a traveling salesman problem obtained by the Lin–Kernighan method is about 85% on average (Lin & Kernighan, 1973). Based on this assumption, an ideal offspring generator should be able to produce a solution which is close to the best solutions found so far. Suppose the current population in an evolutionary algorithm with local search consists of the best locally optimal solutions found so far, a new solution generated by the conventional mutation is close (similar) to its parent, but may be far away from other better solutions since the mutation does not utilize any global information extracted from the current population.

EDAs extract global statistical information from the previous search and then represent it as a probability model, which characterizes the distribution of promising solutions in the search space. New solutions are generated by sampling from this model. However, the location information of the locally optimal solutions found so far has not been directly used in EDAs, there is no mechanism to directly control the similarity between new solutions and a given solution. The idea behind the proposed operator which we call guided mutation is to combine the global statistical information and location information of the solutions found so far to overcome the shortcoming of GAs and EDAs.

Several different probability models have been introduced in EDAs for modeling the distribution of promising solutions. The univariate marginal distribution (UMD) model is the simplest one and has been used in univariate marginal distribution algorithm (Muhlenbein, 1997), population-based incremental learning (Baluja, 1994), and compact GA (Harik et al., 1999). Let the search space be $\Omega = \{0, 1\}^n$, UMD model uses a probability vector $p = (p_1, \dots, p_n) \in [0, 1]^n$ to characterize the distribution of promising solutions in the search space, where p_i is the probability that the value of the ith position of a promising solution is one. The guided mutation operator uses a probability vector $p \in [0, 1]^n$ to guide to mutate an $x \in \{0, 1\}^n$ in the following way (see Fig. 1):

```
1: Guided mutation Operator $y = GM(p, x, \beta)$
2: Input: $p = (p_1, \dots, p_n) \in [0,1]^n, x = (x_1, \dots, x_n) \in \{0,1\}^n$ and $\beta \in [0,1]$.
3: Output: $y = (y_1, \dots, y_n) \in \{0,1\}^n$.
4: for $i := 1$ to $n$ do do
5:     Flip a coin with head probability $\beta$;
6:     If the head turns up, with probability $p_i$ set $y_i = 1$, otherwise set
         $y_i = 0$,
7:     Otherwise, $y_i := x_i$.
8: end for
```

Fig. 1. EA/G: MainProcedure().

Remark 1. From the above guided mutation operator, y_i is directly copied from the parent x or randomly sampled from the probability vector p. The larger β is, the more elements of y are sampled from the probability vector p. In other words, β, similar to the mutation rate in conventional mutation, controls the similarity between offspring and the parent, while the parent can be chosen from the best solutions found so far.

Remark 2. In the correlated mutation (Eiben & Smith, 2003) for real vectors, the probability of generating an offspring in the steepest ascent direction is larger than in other directions. In the conventional mutation for binary strings, the probability of a vector y being generated from the parent vector x is entirely determined by the Hamming distance between x and y. The guided mutation operator can be regarded as a discrete counterpart of the correlated mutation. The probability vector p in the guided mutation can be learned and updated at each generation for modeling the distribution of promising solutions. Since some elements of the offspring y are sampled from the probability vector p, it can be expected that y should fall in or close to a promising area. Meanwhile, this sampling also provides diversity for the search afterwards.

3. Introduction of ACGA

An Artificial Chromosome Genetic Algorithm (ACGA) is able to capture the parental distribution by sampling new solutions (artificial chromosomes) from the probabilistic models periodically and the rest of the chromosomes are created by the genetic operators. Sampling new individuals periodically is the different characteristics from other EDAs because most EDAs generate new solutions entirely.

The primary procedure of ACGA is to collect gene information first and then use the gene information to generate artificial chromosomes. ACGA collects the chromosomes whose fitness is better than the average fitness of current population. Thus, the average fitness is calculated generation by generation for gene collection. After the gene collection, certain statistics of the gene distribution will be calculated. Then, ACs will be generated according to these statistics. A detailed procedure of the ACGA algorithm is depicted in Fig. 2.

Basically, there are two parameters to be decided in this algorithm, which are *startingGen* and *interval*. The first parameter *startingGen* is to determine the starting time of generating artificial

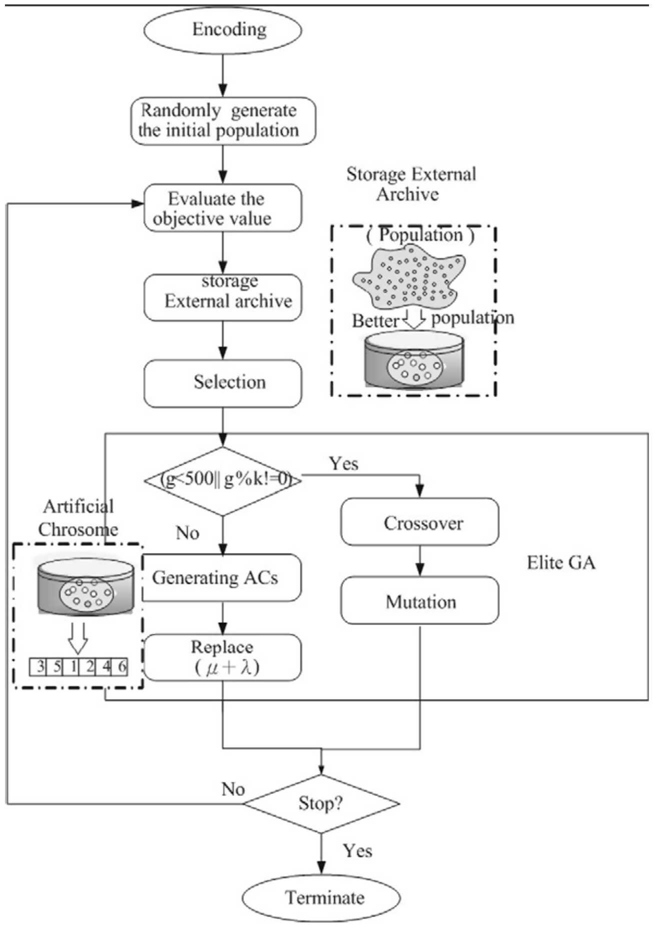

Fig. 2. The framework of the ACGA.

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. As a result, the probability model is applied after some generations. Later on, artificial chromosomes are not generated in each generation because it takes more computational time since the proportional selection takes $O(n^2)$ time complexity for each solution. Consequently, *interval* controls the time interval of artificial chromosomes generated. A set of experiments for parameter configuration has been set up by Design-of-Experiment (DOE). DOE will examine the significance of each factor. According to these preliminary results, both factors have no significant difference. Therefore, the *startingGen* and *interval* are set to 500 and 50 in later experiments, respectively.

The following Section 3.1 explains the proposed algorithm in detail. First, a step by step procedure is applied to explain how to establish a probabilistic model. Then in Section 3.2, an instance is applied to explain how to generate an offspring by the probabilistic model.

3.1. Establishing a probabilistic model

Suppose a population has M strings $X^1, X^2, \ldots, X^M$ at 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. (1).

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

The fitness of these M chromosomes is evaluated and the gene information is collected from N best chromosomes where $N \leqslant M$. The N chromosomes are set as $M/2$ in this research. The purpose of only selecting N best chromosomes from population is to prevent the quality of the probabilistic model from being downgraded by inferior chromosomes. Let $P_{ij}(t)$ be the probability of job i to show up at position j at current generation. Our probability model is similar to PBIL where the $P_{ij}(t)$ is updated as follows:

$$P_{ij}(t+1) = \frac{1}{N} \sum_{k=1}^{N} X_{ij}^k, \quad i = 1, \ldots, n, \ j = 1, \ldots, n \tag{2}$$

For the probabilistic matrix of all jobs at different positions, they are written as the Eq. (3).

$$P(t+1) = \begin{pmatrix} P_{11}(t+1) & \cdots & P_{1n}(t+1) \\ \vdots & \ddots & \vdots \\ P_{n1}(t+1) & \ldots & P_{nn}(t+1) \end{pmatrix} \tag{3}$$

3.2. Generating offsprings by the parental distribution

As soon as the probabilistic matrix P is built, jobs are assign onto each positions by proportional selection. Through this proportional selection, Zhang and Muhlenbein (2004) showed if the distribution of the new elements capture the parents well, global optimal will be obtained, and a factorized distribution algorithm converges globally under proportional selection. The assignment sequence for each position is assigned in random sequence. The assignment procedure is determined as follows:

S is a set of shuffled sequence which determines the sequence of each position is assigned a job, Ω is the set of un-arranged jobs, J is the set of arranged jobs. J is empty in the beginning, θ is a random probability is drawn from $U(0,1)$, i is a selected job by proportional selection and k the element index of the set S.

```
1: $S \leftarrow$ shuffled the job number $[1 \ldots n]$
2: $J \leftarrow \Phi$
3: **while** $k \neq \Phi$ **do**
4:     $\theta \leftarrow U(0,1)$
5:     Select a job $i$ satisfies $\theta \leqslant P_{ik}/\sum_{i\in\Omega} P(i,k)$
6:     $J(k) \leftarrow i$
7:     $\Omega \leftarrow \Omega \setminus i$
8:     $S \leftarrow S \setminus k$
9: **end while**
```

In Step 5, since the time-complexity of the proportional selection is $O(n^2)$, it spends more time than using crossover operator. As a result, it is the reason why this paper hybridizes the probabilistic model with genetic operators that can avoid the excessive computational efforts.

The focus of this research is to apply the EDAs to solve the NP-hard single-machine scheduling problem, particularly the minimization of the total weighted earliness and tardiness costs. We review this scheduling problem in Section 4.

4. Single machine scheduling problems

As a generalization of single-machine scheduling to minimize the weighted tardiness (Lenstra, Kan, & Brucker, 1975), the single-machine scheduling problem to minimize the total weighted earliness and tardiness costs is strongly NP-hard. The earlier works on this problem were due to Chang (1999), Chang and Lee (1992) and Wu et al. (1993). Belouadah, Posner, and Potts (1992) dealt with a similar problem with the objective of minimizing the total weighted completion time. The problem is the same as that discussed in Belouadah et al. (1992). Later on, Alves and Almeida (2007) developed various dominance rules to solve the problem. Valente and Alves (2005, 2007) presented branch-and-bound algorithms based on decomposition of the problem into the weighted earliness and the weighted tardiness sub-problems. Two lower bound procedures were used for each sub-problem. The lower bound for the original problem was the sum of the lower bounds for the two sub-problems. In Valente and Alves (2007), the authors analyzed the performance of various heuristic procedures, including dispatching rules, a greedy procedure, and a decision theory based search heuristic.

The earliness/tardiness scheduling problem with equal release dates and no idle time has been considered by several authors. Both exact and heuristic approaches have been proposed to solve the problem. Among the exact algorithm approaches, branch-and-bound algorithms were presented by Abdul-Razaq and Potts (1988), Li (1997), and Liaw (1999). The lower bounding procedure of Abdul-Razaq and Potts (1988) is based on the sub-gradient optimization approach and the dynamic programming state-space relaxation technique, whereas Li (1997) and Liaw (1999) used Lagrangian relaxation and the multiplier adjustment method. Among these heuristics, Ow and Morton (1989) developed several dispatching rules and a filtered beam search procedure. In Valente and Alves (2005), the authors presented an additional dispatching rule and a greedy procedure. They also considered the use of dominance rules to further improve the schedule obtained by the heuristics. A neighborhood search algorithm was presented by Li (1997).

Chang, Chen, and Fan (2009c) developed a new algorithm, termed as the Electromagnetism-like algorithm, to deal with the single-machine scheduling problem with the consideration of earliness/tardiness penalties. The Electromagnetism-like algorithm was originally proposed by Birbil and Fang (2003), which is able to solve continuous problems. But the Electromagnetism-like algorithm needs to incorporate the random key method to solve discrete problems such as scheduling problems. Therefore, the Electromagnetism-like algorithm diversifies the inferior solutions, whereas the genetic algorithm operator, i.e., the crossover operator, recombines better solutions. Experimental results show that hybrid algorithms are far superior to using the Electromagnetism-like algorithm alone.

Some research has developed dominance properties (DPs) for this category of problems (Liaw, 1999; Luo & Chu, 2006; Luo, Chu, & Wang, 2006; Jouglet, Savourey, Carlier, & Baptiste, 2008; Sourd & Kedad-Sidhoum, 2003, 2007). DPs are employed in branch-and-bound algorithms to enhance the fathoming procedure to be combined with other meta-heuristics, such as integrating DPs with GA to solve the scheduling problem with earliness/tardiness penalties (Chang, Chen, & Mani, 2009d).

5. Convergency progress analysis of EDAs

In this section, ACGA and EA/G are analyzed by running the instances of single machine scheduling problems with the minimization of earliness/tardiness cost. When we observe the convergency progress of the two EDAs, simple Genetic Algorithm (SGA) with elitism is also adopted into the comparisons. The stopping criteria are based on the number of examined solutions, which are 50,000, 75,000, 100,000, and 125,000 solutions. The four examined solutions stand for the different implementation environments which allow lower, medium, high, and higher level of computational time. When taking a close look of the convergency behavior, we attempt to discover whether there is any difference among the three algorithms when the stopping criteria are different. The analysis is done by Design-of-Experiment to distinguish the difference of these algorithms. And the parameter settings are the same, such as the population size is 100, the crossover rate is 0.9, and mutation rate is 0.5 across all experiments. And when we use 50,000 solutions, it means that the algorithms stop at generation 500.

There are numerous data sets published in the literature (Sourd & Kedad-Sidhoum, 2003) for the single-machine scheduling problems, including 20, 30, 40, 50, 60, and 90 jobs. Each data set of 20 jobs up to 50 jobs contains 49 instances (problems) whereas there are nine instances in the data set of 60 jobs and 90 jobs. We carried out our experiments on these total 214 instances. Each algorithm will replicate every instance 30 times. The following subsections are the empirical results of solving the single-machine scheduling problems.

5.1. Empirical results of different stopping criteria

In Tables 1–4, they illustrate the minimum, average, and maximum objective values for the SGA, ACGA, and EA/G, respectively. It is clearly that ACGA and EA/G outperform the SGA. In order to test the significance between ACGA and EA/G, Analysis of Variance (ANOVA) is used. When the *P*-Value is less than 0.05, it means there is a significance of the factor. The detailed information of the ANOVA analysis is in Chen and Chen (2009).

In Chen and Chen (2009), the factor Method is very significant in the all ANOVA tables, Duncan Grouping test is used to further

Table 1
Selected results of these algorithms employ 50,000 examined solutions.

Instance	SGA			ACGA			EA/G		
	Min	Avg.	Max	Min	Avg.	Max	Min	Avg.	Max
sks222a	5286	5357.2	5504	5286	5289.3	5298	5286	5290.8	5298
sks255a	2372	2413.8	2508	2372	2382.7	2388	2372	2380.0	2388
sks288a	3421	3471.2	3576	3421	3421.0	3421	3421	3421.0	3421
sks322a	11,574	11,874.7	12,760	11,568	11,578.9	11,622	11,568	11,574.6	11,622
sks355a	6090	6430.1	6930	6056	6056.9	6058	6056	6065.7	6212
sks388a	11,317	11,345.1	11,517	11,317	11,317.0	11,317	11,317	11,318.5	11,340
sks422a	25,769	26,177.6	26,971	25,656	25,666.6	25,704	25,656	25,661.8	25,697
sks455a	6797	7409.1	8415	6405	6443.5	6545	6405	6435.2	6667
sks488a	16,910	17,600.0	18,431	16,862	16,862.9	16,888	16,862	16,862.0	16,862
sks522a	29,564	30,388.7	31,799	29,309	29,327.0	29,398	29,309	29,343.7	29,398
sks555a	10,338	11,903.9	13,510	10,187	10,233.9	10,456	10,187	10,208.6	10,264
sks588a	25,469	26,143.1	26,931	24,844	24,846.5	24,861	24,844	24,846.4	24,861
sks622a	44,150	45,269.9	46,818	43,048	43,098.0	43,369	43,048	43,107.1	43,286
sks655a	17,565	19,996.8	22,313	16,158	16,224.8	16,716	16,158	16,196.4	16,640
sks688a	34,886	36,366.7	38,108	33,551	33,638.5	33,797	33,551	33,600.3	33,686
sks922a	92,619	95,766.7	99,835	88,853	89,085.8	89,549	88,841	88,870.8	89,082
sks955a	36,733	41,872.8	47,525	30,606	30,828.9	31,235	30,582	30,648.2	30,804
sks988a	89,034	92,361.9	97,193	82,099	82,279.9	82,531	81,984	81,985.3	81,990

Table 2
Selected results of these algorithms employ 75,000 examined solutions.

Instance	SGA			ACGA			EA/G		
	Min	Avg.	Max	Min	Avg.	Max	Min	Avg.	Max
sks222a	5286	5356.9	5604	5286	5289.2	5298	5286	5289.2	5298
sks255a	2372	2428.6	2712	2372	2381.1	2388	2372	2382.9	2388
sks288a	3421	3472.0	3648	3421	3421.0	3421	3421	3421.0	3421
sks322a	11,568	11,880.5	12,358	11,568	11,574.6	11,622	11,568	11,570.6	11,622
sks355a	6056	6430.3	7061	6056	6057.0	6058	6056	6072.1	6242
sks388a	11,317	11,334.4	11,534	11,317	11,317.0	11,317	11,317	11,320.1	11,340
sks422a	25,755	26,245.1	27,044	25,656	25,660.5	25,704	25,656	25,663.9	25,697
sks455a	6613	7202.4	7916	6405	6428.0	6545	6405	6428.4	6545
sks488a	17,013	17,490.0	18,128	16,862	16,865.5	16,888	16,862	16,862.9	16,888
sks522a	29,588	30,294.6	31,365	29,309	29,318.3	29,396	29,309	29,320.8	29,398
sks555a	10,625	11,933.9	13,957	10,187	10,217.6	10,368	10,187	10,210.5	10,267
sks588a	24,992	25,863.0	26,348	24,844	24,844.6	24,861	24,844	24,846.4	24,861
sks622a	43,543	44,858.5	46,690	43,048	43,089.8	43,369	43,048	43,095.7	43,286
sks655a	17,645	19,304.3	21,366	16,158	16,175.1	16,570	16,158	16,241.2	16,640
sks688a	34,872	35,966.8	37,579	33,551	33,612.8	33,665	33,551	33,590.1	33,686
sks922a	92,013	94,714.1	98,407	88,842	88,940.7	89,631	88,842	88,884.5	89,078
sks955a	34,538	40,738.2	48,650	30,582	30,710.3	31,435	30,582	30,649.4	30,769
sks988a	87,099	91,698.7	97,224	81,984	82,037.5	82,198	81,984	81,989.6	82,112

Table 3
Selected results of these algorithms employ 100,000 examined solutions.

Instance	SGA			ACGA			EA/G		
	Min	Avg.	Max	Min	Avg.	Max	Min	Avg.	Max
sks222a	5286	5352.3	5603	5286	5288.9	5298	5286	5289.9	5298
sks255a	2372	2459.3	2936	2372	2380.0	2388	2372	2380.4	2388
sks288a	3421	3480.1	3684	3421	3421.0	3421	3421	3421.0	3421
sks322a	11,568	11,857.1	12,211	11,568	11,577.1	11,622	11,568	11,575.5	11,622
sks355a	6100	6335.4	7083	6056	6065.4	6193	6056	6062.1	6212
sks388a	11,317	11,323.9	11,413	11,317	11,317.0	11,317	11,317	11,320.1	11,340
sks422a	25,662	26,169.9	27,138	25,656	25,659.2	25,704	25,656	25,661.1	25,712
sks455a	6575	7298.5	9472	6405	6426.7	6666	6405	6424.6	6545
sks488a	17,126	17,528.5	18,059	16,862	16,862.9	16,888	16,862	16,862.9	16,888
sks522a	29,477	30,232.9	31,574	29,309	29,312.2	29,396	29,309	29,325.5	29,398
sks555a	10,667	11,910.3	15,024	10,187	10,215.8	10,299	10,187	10,224.1	10,299
sks588a	25,004	25,836.3	26,580	24,844	24,844.9	24,861	24,844	24,849.3	24,870
sks622a	43,401	44,786.5	45,863	43,048	43,119.9	43,479	43,048	43,103.2	43,273
sks655a	17,728	19,389.5	22,617	16,158	16,218.0	16,635	16,158	16,222.4	16,617
sks688a	34,517	35,775.6	37,418	33,551	33,638.6	33,665	33,551	33,596.6	33,686
sks922a	92,425	94,684.9	99,061	88,841	88,894.2	89,067	88,841	88,875.5	89,188
sks955a	35,558	39,495.4	43,256	30,582	30,682.8	31,312	30,590	30,643.2	30,768
sks988a	86,422	90,895.5	96,954	81,984	82,001.5	82,053	81,984	81985.2	81,989

Table 4
Selected results of these algorithms employ 125,000 examined solutions.

Instance	SGA			ACGA			EA/G		
	Min	Avg.	Max	Min	Avg.	Max	Min	Avg.	Max
sks222a	5286	5352.3	5603	5286	5288.9	5298	5286	5289.9	5298
sks255a	2372	2459.3	2936	2372	2380.0	2388	2372	2380.4	2388
sks288a	3421	3480.1	3684	3421	3421.0	3421	3421	3421.0	3421
sks322a	11,568	11,857.1	12,211	11,568	11,577.1	11,622	11,568	11,575.5	11,622
sks355a	6100	6335.4	7083	6056	6065.4	6193	6056	6062.1	6212
sks388a	11,317	11,323.9	11,413	11,317	11,317.0	11,317	11,317	11,320.1	11,340
sks422a	25,662	26,169.9	27,138	25,656	25,659.2	25,704	25,656	25,661.1	25,712
sks455a	6575	7298.5	9472	6405	6426.7	6666	6405	6424.6	6545
sks488a	17,126	17,528.5	18,059	16,862	16,862.9	16,888	16,862	16,862.9	16,888
sks522a	29,477	30,232.9	31,574	29,309	29,312.2	29,396	29,309	29,325.5	29,398
sks555a	10,667	11,910.3	15,024	10,187	10,215.8	10,299	10,187	10,224.1	10,299
sks588a	25,004	25,836.3	26,580	24,844	24,844.9	24,861	24,844	24,849.3	24,870
sks622a	43,401	44,786.5	45,863	43,048	43,119.9	43,479	43,048	43,103.2	43,273
sks655a	17,728	19,389.5	22,617	16,158	16,218.0	16,635	16,158	16,222.4	16,617
sks688a	34,517	35,775.6	37,418	33,551	33,638.6	33,665	33,551	33,596.6	33,686
sks922a	92,425	94,684.9	99,061	88,841	88,894.2	89,067	88,841	88,875.5	89,188
sks955a	35,558	39,495.4	43,256	30,582	30,682.8	31,312	30,590	30,643.2	30,768
sks988a	86,422	90,895.5	96,954	81,984	82,001.5	82,053	81,984	81,985.2	81,989

distinguish the performance of the two algorithms. In Duncan Group test, when the algorithms share the same alphabet, it means they are in the same group so that there is no difference between/among these algorithms. On the other hand, as soon as they are not in the same group (or to share the same alphabet), there is significant difference between/among them.

The Duncan grouping results show that when EA/G outperforms the ACGA under the stopping criterion of using 50,000 and 75,000 solutions. There is no difference between EA/G and ACGA when they both apply 100,000 solutions. Finally, ACGA outperforms the EA/G statistically significant in the case of employing 125,000 solutions. A performance transition is occurred at the stopping criterion of using 100,000 solutions.

In order to show the results clearly, we demonstrate the interaction plots of the algorithms together with the different examined solutions in Fig. 3. Through the interaction plots, it shows that the EA/G indeed outperforms the ACGA very much. However, the performance of EA/G does not be improved with the number of examined solutions increased after the level of 75,000. ACGA, nonetheless, is improved generation by generation and this algorithm is superior to EA/G when we apply longer computational time.

This phenomenon can be explained by Fig. 4 which utilizes the instance sks952a. EA/G converges faster than ACGA and SGA. After the generation 150, EA/G is converged and the performance is not improved. As a result, it could be a problem of premature convergency belonged to EA/G. So we may increase the diversity of the generated solutions for the EA/G or the EDAs completely sample new individuals from probabilistic models.

To conclude the comparison results, these experiments reveal interesting points when the three algorithms are ran under the various stopping criteria. EA/G outperforms the ACGA statistically significant when it stops under lower computational time whereas ACGA performs well when we apply higher level of stopping criterion. It shows EA/G might converge faster than ACGA; however, when we concern on the solution quality and we are able to employ higher level computing time. As a result, this paper continually discusses this phenomenon and then conduct some important guidelines in Section 6.

6. Guidelines of designing effective EDAs

There are some interesting results obtained in the Section 5. EA/G is better than ACGA when the computational time is limited. On the other hand, ACGA outperforms the EA/G when it comes to longer stopping criterion. The reason is that sampling new solutions

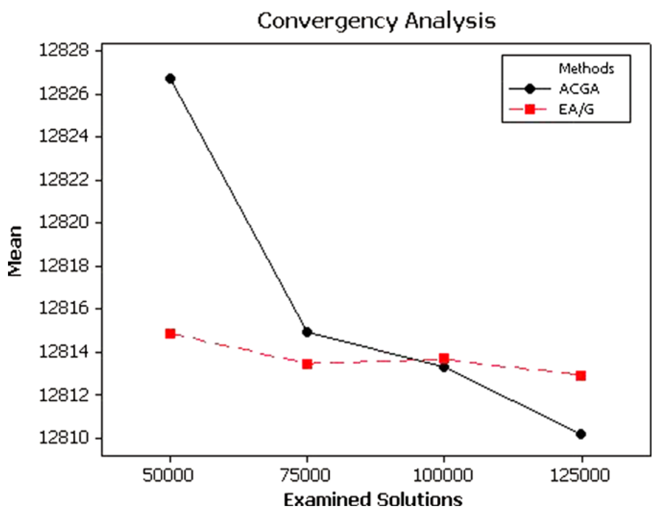

Fig. 3. ACGA and EA/G evaluate different examined solutions.

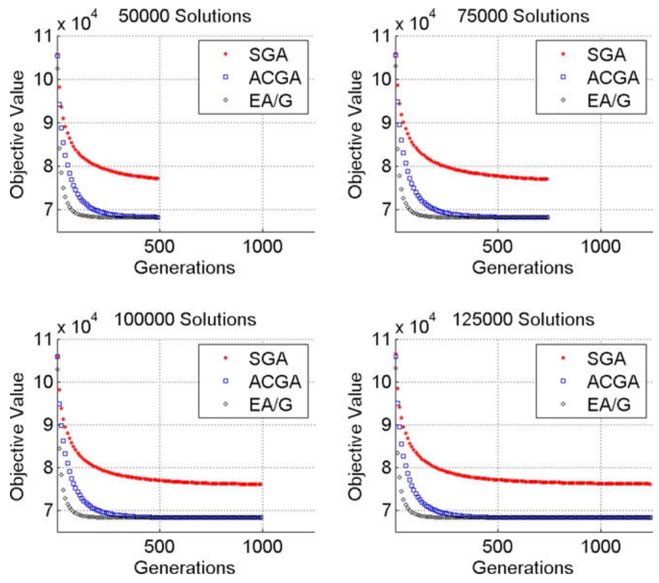

Fig. 4. Convergency analysis of the algorithms in different stopping criteria (instance-sks952a).

from probabilistic models avoids the disruption of good solution structures represented by the model. During the early evolutionary process, there are some salient genes that are available. Because EA/G completely samples new solutions from probabilistic models, it is helpful for the convergency speed in the early stage. Through the convergency plots in the previous section, however, EA/G seems no longer producing better solutions after 200 generations in most instances. It means that the stagnation of EA/G after some generations are executed.

Contrary to EA/G, the convergency speed of ACGA is not fast in the early stage. ACGA, nevertheless, performs well in the case that the solution quality is more important so that we implement the algorithm with longer computational time. And the implementation time should be controlled in a reasonable time. It said that ACGA is a hybrid algorithm of probabilistic models and SGA. This hybrid algorithm samples new solutions from probabilistic models once in a while and other generations use standard genetic operators during the evolutionary process. The implication of this framework means that it may converge slower because the genetic operators may break good solution structure. The benefit of genetic operators, however, also brings better population diversity due to the disruptions by genetic operators. The intensification and diversification effect present the trade-off in the searching progress.

Through these statements, Meta-heuristics should have a technique to balance the intensification and diversification. For instance, Reactive Tabu Search (Battiti & Tecchiolli, 1994; Osman & Wassan, 2002) diversifies the solution by making a number of random moves when the search appears to be repeating an excessive number of solutions excessively often. With this assumption, ACGA has better chances to explore different solution space than EA/G under longer computational time. Due to these interesting finding, we deduct and clearly point out some guidelines for developing an effective searching algorithm in the following sections. In addition, this paper proposed an Adaptive EA/G in Section 6.5 which utilizes the first guideline to enhance the quality of the EA/G.

6.1. Increasing the population diversity gradually

The stagnation of EA/G apparently exists after some generations are ran. It could be a common problem for other EDAs which entirely sample news solutions from probabilistic models. For example, PBIL (Baluja & Davies, 1997), UMDA (Miettinen, 1999), cGA

(Hansen & Mladenović, 2001), BMDA (Okabe, Jin, Sendoff, & Olhofer, 2004), ECGA (Hansen, Mladenović, & Perez-Britos, 2001), BOA (Ow & Morton, 1989) are all in this class. As a result, these EDAs shall consider the population diversity when the population no longer generates diversified or better solutions.

In terms of revising these existing EDAs, a key approach is to control the probabilistic models after a number of generations are ran. Some diversity preservation techniques are proposed in Chang et al. (2009a), which stem from the concept of Ant-Colony Optimization (Corne et al., 1999; Gambardella, Taillard, & Dorigo, 1999). They test the diversity preservation techniques in solving the single-machine scheduling problems and flowshop scheduling problems. Their results indicate the best objective evaporation and Max–Min evaporation performs significantly in the single-machine scheduling and flowshop scheduling problems, respectively. It might be applicable for all EDAs.

Apart from the general methodology of controlling the probabilistic models, the other method is to modify each algorithm based on their own characteristics. Take EA/G for example, β is the parameter which determines a proportion of genes copied from a selected elite solution. β is set as a static parameter in the original version of EA/G. In order to increase the diversity, β can be decreased generation by generation so that the diversity of the population will be increased. In order to explain the ideas of these guidelines clearly, this paper chooses this guideline to change the β for designing an Adaptive EA/G in Section 6.5.

6.2. EDAs alternative with other meta-heuristics

Since EDAs may not generate diversify solutions well and cause the problem of premature convergency, incorporation of meta-heuristic, particularly the genetic algorithm, might be a good approach to raise the population diversity. There are manifold existing hybrid frameworks of combining the EDAs with genetic algorithms, such as GA-EDA, Model-Based EA, ACGA and Self-Guided GA. The main characteristics of these algorithm is to alternate EDAs together with genetic algorithms. GA-EDA and Model-Based EA apply the probabilistic models and genetic operators by turns. ACGA utilizes the probabilistic models occasionally. Self-Guided GA is different from all EDAs because this algorithm does not sample news solutions from the probabilistic models. The probabilistic models, instead, are treated as a fitness surrogate to evaluate the fitness of new solutions beforehand in each generation. Thus these hybrid algorithm could yield better solution quality when they employ longer computational efforts.

It is clearly that GA-EDA, ACGA, and Model-Based EA alternate the EDAs with GAs. If researchers want to further enhance the solution quality of these algorithms, they should note that the searching strategy of intensification by using the EDAs and diversification via GAs. Since EDAs may converge faster in the beginning and genetic operators might provide better population diversity, EDAs could be implemented frequently in the early stage while GAs dominate the most computational efforts in the later period. As a result, although GA-EDA and Model-Based EA applies the EDAs and GAs by turn in the original version, we can increase the frequency of the EDAs in the early stage and then decrease the usage of EDAs in the later stage. In addition, ACGA *injects* artificial chromosomes into the population periodically, the new version of ACGA could inject the artificial chromosome more frequently in the earlier stage and to enlarge the injection period while the algorithm is getting converged.

Except generating a population of chromosomes by probabilistic models or genetic operators in a single generation, a possible way is to sample a proportional solutions from probabilistic models and the rest of solutions are generated by using crossover and mutation operator. MGSPGA (Chang, Chen, & Liu, 2007) samples 20% solutions from a probability matrix and the rest of the chromosomes are generated by the genetic operators. Consequently, the goal of intensification and diversification are balanced in each generation. Because it is still unknown about which approach is better (samples new solutions entirely or partially), researchers should examine the two strategies in the near future.

Although most previous researchers employ genetic algorithms, the authors advocate that it is not limited to alternate EDAs with genetic algorithms. EDAs could work with other meta-heuristics. For example, an artificial immune system (AIS) might be able to generate diversified solutions (Pasti & de Castro, 2009). It is because an AIS employs many properties of natural immune systems, particularly the diversity (Hunt & Cooke, 1996). This natural system possesses a content addressable memory and the ability to forget little-used information (Hofmeyr & Forrest, 2000). Thus, AIS may provide diverse solutions that result in good solution quality when we alternate the EDAs with AIS.

Finally, there is a benefit when we do not sample all solutions from probabilistic models. When we solve sequential problems, the time-complexity of sampling procedures by applying the proportional selection is $O(n^2)$. It is particularly time-consuming when we solve larger size problems. As a result, we should use more efficient approach without using the proportional selection. The next section describes the methods to solve the crux of EDAs in solving the sequential problems by using the concept of Self-Guided GA.

6.3. Replacing the procedures of sampling new solutions

Owing to the proportional selection of EDAs is used in most sequencing problems, it causes higher computational cost, whose time-complexity is $O(n^2)$. It makes EDAs impractical in solving the larger size problems. As a result, it is the main motivation of Self-Guided GA (Chen et al., 2008b) which samples new solutions from probabilistic models. Instead, they use the probabilistic models as a fitness surrogate which determines the evolutionary progress. In addition, because the genetic operators are employed in the evolutionary process, Self-Guided GA is expected to produce better diversified population. This algorithm is ideal to be combined with other heuristic to further improve the solution quality. Detailed explanations are shown in the next section.

Apart from the replacing the proportional selection procedures, Sastry, Pelikan, and Goldberg (2004) also discuss the efficiency enhancement of EDAs. Efficiency-enhancement techniques speedup the search process of Estimation of Distribution Algorithms (EDAs). Principled approaches for designing an evaluation-relaxation, and a time continuation technique. So when we utilize these approaches, it is helpful to tackle with hard problems.

6.4. Incorporate EDAs with other heuristics

Although Zhang, Sun, and Tsang (2007) point out the positive effect of combining other heuristics and it may bring better solution quality, the meanings are two-fold in this paper. First of all, there are many possibilities to adopt some heuristics with EDAs and heuristic could enhance the convergency effect. Take the scheduling problems for instance, dominance properties are the mathematical derivations for the scheduling problems. Chen, Chang, Chen, and Chen (2008a) employ this technique to work with an estimation of distribution algorithm. As a result, the performance of this hybrid algorithm is better than any previous algorithms which work separately. When we solve the flowshop scheduling problems, there is a famous heuristic which is named NEH (Nawaz, Enscore, & Ham, 1983). This heuristic is used to generate a good initial solution and then the meta-heuristic is used to further improve the solution quality.

Secondly, because the heuristic method is used to generate good solutions, the researchers should pay more attentions to the diversity issue of EDAs. The reason is before EDAs carry out the exploration, there is a good basis provided by heuristics. It is apparently that the designing strategy should consider how to let the EDAs yield better population diversity. A heuristic method, consequently, may work well with ACGA because a heuristic method is responsible for the convergency and ACGA produces diversified population. In addition, according to the guideline one, EA/G may apply lower β value which reduces the number of copied genes from a selected elite individual when combining with a heuristic. In general, EDAs work together with a heuristic is a good approach because we may obtain better solution quality in a shorter time and we should avoid probabilistic models no longer generate diversified individuals.

6.5. An example of 1st guideline: adaptive EA/G

The parameter β is an important parameter which controls the proportional of copied genes from an elite solution. In our previous research (Chen et al., 2008b), this parameter is statistically significant in solving the single machine and flowshop scheduling problems. The reason is if β is higher, it means EA/G converges faster in the beginning. In addition, we discuss that the EA/G no longer improves the solution quality after 100 or 200 generations in most instances in guideline 1. The convergency progress apparently reveals the stagnation of this algorithm and the main reason might be the population diversity decreased. As a result, this paper proposes an Adaptive EA/G by adjusting the parameter β generation by generation.

We define the total number of generation is as *MaxGen* and *g* is the current generation. In Eq. (4), it shows that the new setting of β as β'. Thus in the very beginning, the β' is high and the β is decreased gradually with the evolutionary progress. Through this simple approach, the β' is decreased and the number of copied elite genes are decreased so that the population diversity is increased.

$$\beta' = \beta * (MaxGen - g)/(MaxGen) \tag{4}$$

Fig. 5 demonstrates the main procedures of the Adaptive EA/G. The differences are in Steps 1, 2, and 6 which calculate the β' first and then let the EA/G to apply this new β' value. In order to validate the performance of the Adaptive EA/G, it is thus compared with other algorithms and the experimental result is shown in the Section 7 in solving the single-machine scheduling problems under different stopping criteria. The following is the pseudo code of the proposed Adaptive EA/G.

S: A set of shuffled sequence which determines the sequence of each position is assigned a job.
Ω: is 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 $[1 \dots n]$
2: $J \leftarrow \Phi$
3: Calculating $\beta' = \beta * (MaxGen - g)/(MaxGen)$
4: Guided mutation Operator $y = GM(p, x, \beta')$ $\beta' \in [0, 1]$.
5: Output: $y = (y_1, \dots, y_n) \in \{0, 1\}^n$.
6: while $k \neq \Phi$ do
7:   Flip a coin with head probability $\beta'$;
8:   if The head turns up then
9:     Select a job $i$ satisfies $\theta \leq P_{ik}/\sum_{i \in \Omega} P(i, k)$
10:    $J(k) \leftarrow i$
11:  else
12:    $y_i := x_i$.
13:  end if
14:  $\Omega \leftarrow \Omega \backslash i$
15:  $S \leftarrow S \backslash k$
16: end while
```

Fig. 5. Adaptive EA/G: MainProcedure().

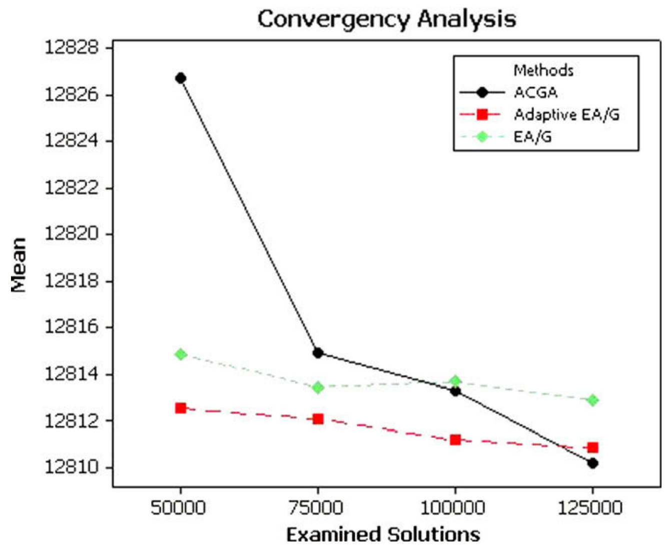

Fig. 6. Adaptive EA/G, ACGA, and EA/G evaluate different examined solutions.

7. Experiment results of adaptive EA/G

The proposed algorithm Adaptive EA/G is compared with the original EA/G, ACGA, and SGA. They are tested under various stopping criteria. These algorithms evaluate 50,000, 75,000, 100,000, and 125,000 solutions. Fig. 6 is the results of the three algorithms which evaluate different solutions. Adaptive EA/G is better than EA/G in different kinds of solution evaluations. EA/G is better than ACGA when they examine 50,000 and 75,000 solutions. It is a pivot point at 100,000 solutions because EA/G are not better than ACGA in average. While three of them use 125,000 solutions, ACGA outperforms the Adaptive EA/G and EA/G.

The comparison method still uses the ANOVA which is mentioned in Section 5 and the explanations of statistics terms are available in Montgomery (2001). The ANOVA Tables 5–8 which show the significance of these compared algorithms due to the *P*-

Table 5
ANOVA results at the stopping criterion of 50,000 examined solutions.

Source	DF	SS	Mean square	*F*-value	*P*-value
Instances	213	3.99E+12	18,732,022,984	9,920,062	<.0001
Methods	2	745,806.7783	372,903.3892	197.48	<.0001
Instances*method	426	19,754,600.38	46,372.30136	24.56	<.0001
Error	18,618	35,156,311.19	1888.296,873		
Corrected total	19,259	3.99E+12			

Table 6
ANOVA results at the stopping criterion of 75,000 examined solutions.

Source	DF	SS	Mean square	*F*-value	*P*-value
Instances	213	3.98E+12	18,707,615,459	1.17E+07	<.0001
Method	2	26,126.27695	13,063.13847	8.16	0.0003
Instances*method	426	2,060,246.612	4836.259652	3.02	<.0001
Error	18,618	29,802,388.15	1600.72984		
Corrected total	19,259	3.98E+12			

Table 7
ANOVA results at the stopping criterion of 100,000 examined solutions.

Source	DF	SS	Mean square	*F*-value	*P*-value
Instances	213	3.98E+12	18,702,943,555	1.26E+07	<.0001
Method	2	23,117.49668	11,558.74834	7.78	0.0004
Instances*method	426	1,015,995.681	2384.966388	1.61	<.0001
Error	18,618	27,664,794.92	1485.916582		
Corrected total	19,259	3.98E+12			

Table 8
ANOVA results at the stopping criterion of 125,000 examined solutions.

Source	DF	SS	Mean square	*F*-value	*P*-value
Instances	213	3.98E+12	18,698,272,722	1.43E+07	<.0001
Method	2	15,441.49675	7720.748,373	5.9	0.0027
Instances*method	426	1004598.492	2358.212,421	1.8	<.0001
Error	18,618	24,367,388.57	1308.808,066		
Corrected total	19,259	3.98E+12			

Table 9
Duncan grouping at the stopping criterion of 50,000 examined solutions.

Duncan grouping	Mean	*N*	Methods
A	12,826.724	6420	ACGA
B	12,814.85	6420	EA/G
C	12,812.509	6420	Adaptive EA/G

Table 10
Duncan grouping at the stopping criterion of 75,000 examined solutions.

Duncan grouping	Mean	*N*	Method
A	12,814.902	6420	ACGA
B	12,813.443	6420	EA/G
C	12,812.05	6420	Adaptive EA/G

Table 11
Duncan grouping at the stopping criterion of 100,000 examined solutions.

Duncan grouping	Mean	*N*	Method
A	12,813.66	6420	EA/G
A			
A	12,813.276	6420	ACGA
B	12,811.168	6420	Adaptive EA/G

Table 12
Duncan grouping at the stopping criterion of 125,000 examined solutions.

Duncan grouping	Mean	*N*	Method
A	12,812.898	6420	EA/G
B	12,810.827	6420	Adaptive EA/G
B			
B	12,810.149	6420	ACGA

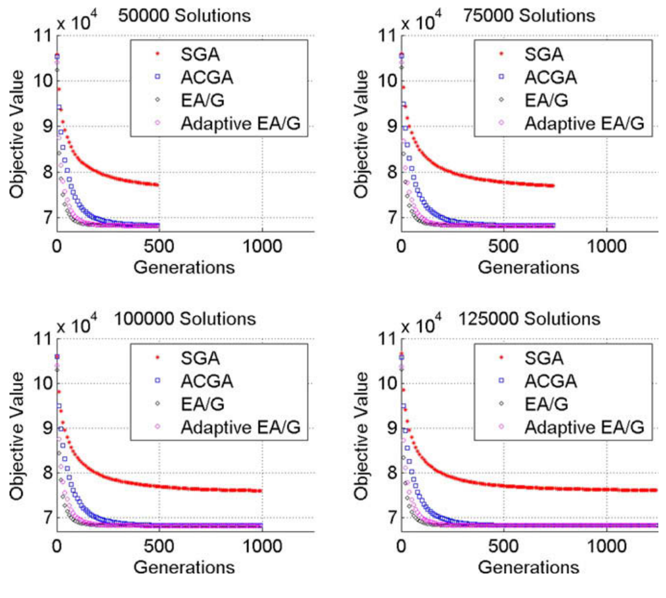

Fig. 7. Compare the convergency speed of Adaptive EA/G with other algorithms (instance-sks952a).

Value is less than 0.0001. Thus a Duncan grouping analysis is used to compare the significance of among the four algorithms.

In Duncan analysis, when the algorithms share the same alphabet, there is no difference between/among the algorithms (Montgomery, 2001). On the other hand, if the algorithms do not share the same alphabet, they are different to each other. Thus in Tables 9–12 present the results of the comparisons. Adaptive EA/G is consistently significant across the different stopping criteria compared to the original EA/G. In Fig. 7, Adaptive EA/G converges slightly slower than EA/G; however, it still converges faster than ACGA and SGA. In addition, because Adaptive EA/G maintains better pop-

Table 13
Selected results of Adaptive EA/G employ different examined solutions.

Instance	50,000			75,000			100,000			125,000		
	Min	Avg.	Max	Min	Avg.	Max	Min	Avg.	Max	Min	Avg.	Max
sks422a	25,656	25,662.8	25,712	25,656	25,659.7	25,697	25,656	25,665.1	25,697	25,656	25,669.4	25,793
sks455a	6405	6420.1	6545	6405	6425.4	6545	6405	6428.4	6545	6405	6415.7	6545
sks488a	16,862	16,865.5	16,888	16,862	16,862.9	16,888	16,862	16,863.7	16,888	16,862	16,863.7	16,888
sks522a	29,309	29,326.2	29,398	29,309	29,323.1	29,398	29,309	29,326.7	29,398	29,309	29,326.2	29,398
sks555a	10,187	10,213.9	10,256	10,187	10,224.2	10,299	10,187	10,219.4	10,301	10,187	10,223.3	10,351
sks588a	24,844	24,847.9	24,861	24,844	24,847.9	24,861	24,844	24,845.8	24,853	24,844	24,849.8	24,870
sks622a	43,048	43,100.5	43,371	43,048	43,078.7	43,380	43,048	43,081	43,273	43,048	43,078.6	43,244
sks655a	16,158	16,163.2	16,180	16,158	16,192.8	16,616	16,158	16,208.1	16,640	16,158	16192.8	16,640
sks688a	33,551	33,592	33,686	33,551	33,624.4	33,686	33,551	33,610	33,686	33,551	33,580.7	33,686
sks922a	88,842	88,872.1	88,994	88,841	88,878.5	89,100	88,842	88,871.6	89,005	88,841	88,858.7	88,956
sks955a	30,582	30,647	30,804	30,582	30,674.8	31,394	30,582	30,639.8	30,800	30,582	30,628.4	30,740
sks988a	81,984	81,986.4	81,991	81,984	81,985.3	81,989	81,984	81,985.2	81,989	81,984	81,985.3	81,989

ulation diversity, it finally outperforms the EA/G in the end of the generations.

When it comes to the comparison between Adaptive EA/G and ACGA, Adaptive EA/G is better than ACGA through 50,000 to 100,000 except in the stopping criterion of using 125,000 solutions. The reason of Adaptive EA/G does not outperform the ACGA is because the ACGA keeps better population diversity so that this algorithm has better chance to find out better solution in the longer computational time. As a result, when we allow to spend more computational time, we can increase the diversity of the Adaptive EA/G being higher. So the proposed algorithm Adaptive EA/G is robust when the algorithm is used in different circumstances.

Finally, Table 13 lists the basic statistic results of Adaptive EA/G under different solution evaluations from some selected instances. The complete experimental results and the technical report (Chen & Chen, 2009) are available on our website.[1]

8. Conclusions

This paper examines the convergency behavior of some EDAs and deduces some guidelines for the EDAs. The convergency analysis reveals the facts of intensification and diversification effect of EDAs. We compare EA/G with ACGA and SGA. EA/G completely samples new individuals from probabilistic models while the ACGA generates new solution by using the probabilistic models periodically. In the fact that the EDAs may not disrupt the good solution structure, the convergency speed is faster than that of using genetic operators. That is why EA/G converges faster than ACGA. However, due to the premature convergency of probabilistic models, EA/G no longer generates diversified solutions so that it is not able to get better solution quality. On the other hand, ACGA converges slower than EA/G but when we extend the computational time, the performance is improved. Thus it reveals interesting points of the convergency analysis.

The guidelines include: (1) increasing population diversity gradually, (2) hybridization of EDAs with other meta-heuristics, (3) replacing the procedures of sampling new solutions, and (4) incorporate EDAs with other heuristics. The first guideline is very important for all EDAs because they may encounter this problem of premature convergency. There are some solving approaches are given. In addition, to make the statement more clearly, we thus proposed an Adaptive EA/G which further improves the solution quality statistically significant. In guideline 2, the reason of incorporating EDAs with other meta-heuristics is to take the advantage of meta-heuristic which is able to provide better diversity. Then, because the sampling procedure is time-consuming and it makes EDAs non-practical in solving larger size problems, the paper suggests some guidelines to speedup the computational efficiency. Finally, although EDAs have been widely incorporated with other heuristics because it is very effective of this combination, we advocate we should pay more attentions to the solution diversity. The reason is that heuristic methods are capable of providing better intensification effects. As a result, the EDAs should increase the diversity of the solutions, such as using the guideline 1 which increases the diversity gradually or the guideline 2 to incorporate the meta-heuristics.

The proposed algorithm Adaptive EA/G is tested by single-machine scheduling scheduling problems. The experimental results indicate the Adaptive EA/G is statistically significant. As a result, it is as a good example of designing an effective algorithm by using these guidelines. For the future research, researchers may test the guidelines to solve various problems and then to validate these guidelines.

Acknowledgement

The authors appreciate the supports from the National Science Council, Taiwan, ROC with Grant contract No. NSC 98-2218-E-343-001-MY2.

References

Abdul-Razaq, T., & Potts, C. (1988). Dynamic programming state-space relaxation for single-machine scheduling. *The Journal of the Operational Research Society, 39*(2), 141–152.

Ackley, D. H. (1987). *A connectionist machine for genetic hillclimbing*. Norwell, MA, USA: Kluwer Academic Publishers.

Aickelin, U., Burke, E. K., & Li, J. (2007). An Estimation of Distribution Algorithm with intelligent local search for rule-based nurse rostering. *Journal of the Operational Research Society, 58*(12), 1574–1585.

Alves, M. J., & Almeida, M. (2007). MOTGA: A multiobjective Tchebycheff based genetic algorithm for the multidimensional knapsack problem. *Computers and Operations Research, 34*(11), 3458–3470.

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. Pittsburgh, PA: Carnegie Mellon University.

Baluja, S. (1995). *An empirical comparison of seven iterative and evolutionary function optimization heuristics*. Technical Report No. CMU-CS-95-193. Carnegie Mellon University.

Baluja, S., & Davies, S. (1997). Using optimal dependency-trees for combinational optimization. In *Proceedings of the fourteenth international conference on machine learning (table of contents)* (pp. 30–38).

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

Baraglia, R., Hidalgo, J., Perego, R., Cnuce, I., & Cnr, P. (2001). A hybrid heuristic for the traveling salesman problem. *IEEE Transactions on Evolutionary Computation, 5*(6), 613–622.

Battiti, R., & Tecchiolli, G. (1994). The reactive tabu search. *ORSA Journal on Computing, 6*(2), 126–140.

[1] http://mail.nhu.edu.tw/shihhsin/publications/sourceCodes/convergencyAnalysis/.

Belouadah, H., Posner, M., & Potts, C. (1992). Scheduling with release dates on a single machine to minimize total weighted completion time. *Discrete Applied Mathematics, 36*(3), 213–231.

Birbil, Ş., & Fang, S. C. (2003). An electromagnetism-like mechanism for global optimization. *Journal of Global Optimization, 25*(3), 263–282.

Chang, P. C. (1999). A branch and bound approach for single machine scheduling with earliness and tardiness penalties. *Computers and Mathematics with Applications, 37*(10), 133–144.

Chang, P. C., Chen, S. H., & Fan, C. Y. (2009c). A hybrid electromagnetism-like algorithm for single machine scheduling problem. *Expert Systems with Applications, 36*(2, Part 1), 1259–1267.

Chang, P. C., Chen, S. H., & Liu, C. H. (2007). Sub-population genetic algorithm with mining gene structures for multiobjective flowshop scheduling problems. *Expert Systems with Applications, 33*(3), 762–771.

Chang, P. C., Chen, S. H., & Mani, V. (2009d). A hybrid genetic algorithm with dominance properties for single machine scheduling with dependent penalties. *Applied Mathematical Modeling, 33*(1), 579–596.

Chang, P., Chen, S. H., & Fan, C. Y. (2009a). Generating artificial chromosomes by mining gene structures with diversity preservation for scheduling problems. *Annals of Operations Research*.

Chang, P., Hsieh, J., Chen, S., Lin, J., & Huang, W. (2009b). Artificial chromosomes embedded in genetic algorithm for a chip resistor scheduling problem in minimizing the makespan. *Expert Systems with Applications, 36*(3, Part 2), 7135–7141.

Chang, P. C., & Lee, H. C. (1992). A greedy heuristic for bicriterion single machine scheduling problems. *Computers and Industrial Engineering, 22*(2), 121–131.

Chen, S. H., & Chen, M. C. (2009). *Convergency analysis of some Estimation of Distribution Algorithms in detail* (pp. 1–7). Technical Report (1).

Chen, S. H., Chang, P. C., & Zhang, Q. (2008b). Self-guided genetic algorithm. *Lecture Notes in Artificial Intelligence, 5227*, 292–299.

Chen, S. H., Chang, P. C., Chen, M. C., Chen, & Y. M. (2008a). Domininance properties for self-guided GA in solving the single machine scheduling problems. In *Proceedings of the CI-Sched 2009* (Vol. 5227, pp. 292–299).

Corne, D., Dorigo, M., Glover, F., Dasgupta, D., Moscato, P., Poli, R., et al. (1999). *New ideas in optimization*. UK Maidenhead, UK, England: McGraw-Hill Ltd.

Eiben, A., & Smith, J. (2003). *Introduction to evolutionary computing*. Springer.

Gambardella, L., Taillard, E., & Dorigo, M. (1999). Ant colonies for the QAP. *Journal of the Operational Research Society, 50*(2), 167–176.

Glover, F., & Laguna, M. (1998). *Tabu search*. Norwell, MA, USA: Kluwer Academic Publishers.

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

Hansen, P., Mladenović, N., & Perez-Britos, D. (2001). Variable neighborhood decomposition search. *Journal of Heuristics, 7*(4), 335–350.

Harik, G., Lobo, F., & Goldberg, D. (1999). The compact genetic algorithm. *IEEE Transactions on Evolutionary Computation, 3*(4), 287–297.

Hofmeyr, S., & Forrest, S. (2000). Architecture for an artificial immune system. *Evolutionary Computation, 8*(4), 443–473.

Hunt, J., & Cooke, D. (1996). Learning using an artificial immune system. *Journal of Network and Computer Applications, 19*(2), 189–212.

Jouglet, A., Savourey, D., Carlier, J., & Baptiste, P. (2008). Dominance-based heuristics for one-machine total cost scheduling problems. *European Journal of Operational Research, 184*(3), 879–899.

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

Lenstra, J., Kan, A., & Brucker, P. (1975). Complexity of machine scheduling problems. In *Proc. workshop studies in integer programming, Bonn*.

Li, G. (1997). Single machine earliness and tardiness scheduling. *European Journal of Operational Research, 96*(3), 546–558.

Liaw, C. F. (1999). A branch-and-bound algorithm for the single machine earliness and tardiness scheduling problem. *Computers and Operations Research, 26*(7), 679–693.

Lin, S., & Kernighan, B. (1973). An effective heuristic algorithm for the traveling-salesman problem. *Operations Research, 21*(2), 498–516.

Lozano, J. A. (2006). *Towards a new evolutionary computation: Advances in the Estimation of Distribution Algorithms*. Springer.

Luo, X., & Chu, F. (2006). A branch and bound algorithm of the single machine schedule with sequence dependent setup times for minimizing total tardiness. *Applied Mathematics and Computation, 183*(1), 575–588.

Luo, X., Chu, C., & Wang, C. (2006). Some dominance properties for single-machine tardiness problems with sequence-dependent setup. *International Journal of Production Research, 44*(17), 3367–3378.

Miettinen, K. (1999). *Nonlinear multiobjective optimization*. Springer.

Montgomery, D. C. (2001). Design and analysis of experiments.

Muhlenbein, H. (1997). The equation for response to selection and its use for prediction. *Evolutionary Computation, 5*(3), 303–346.

Muhlenbein, H., & Paaß, G. (1996). From recombination of genes to the estimation of distributions I. Binary parameters. *Lecture Notes in Computer Science, 1141*, 178–187.

Nawaz, M., Enscore, E., & Ham, I. (1983). A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. *OMEGA, 11*(1), 91–95.

Okabe, T., Jin, Y., Sendoff, B., & Olhofer, M. (2004). Voronoi-based Estimation of Distribution Algorithm for multi-objective optimization. In *Congress on evolutionary computation CEC2004* (Vol. 2).

Osman, I., & Wassan, N. (2002). A reactive tabu search meta-heuristic for the vehicle routing problem with back-hauls. *Journal of Scheduling, 5*(4), 263–285.

Ow, P. S., & Morton, T. E. (1989). The single machine early/tardy problem. *Management Science, 35*(2), 177–191.

Pasti, R., & de Castro, L. (2009). Bio-inspired and gradient-based algorithms to train MLPs: The influence of diversity. *Information Sciences*.

Pelikan, M., Goldberg, D. E., & Lobo, F. G. (2002). A survey of optimization by building and using probabilistic models. *Computational Optimization and Applications, 21*(1), 5–20.

Peña, J., Robles, V., Larrañaga, P., Herves, V., Rosales, F., & Perez, M. (2004). GA-EDA: Hybrid evolutionary algorithm using genetic and Estimation of Distribution Algorithms. In *Innovations in applied artificial intelligence: 17th international conference on industrial and engineering applications of artificial intelligence and expert systems. Proceedings of IEA/AIE 2004, Ottawa, Canada, May 17–20*.

Santana, R., Larrañaga, P., & Lozano, J. (2008). Combining variable neighborhood search and Estimation of Distribution Algorithms in the protein side chain placement problem. *Journal of Heuristics, 14*, 519–547.

Sastry, K., Pelikan, M., & Goldberg, D. E. (2004). Efficiency enhancement of genetic algorithms via building-block-wise fitness estimation. In *Congress on evolutionary computation, CEC2004* (Vol. 1).

Sourd, F., & Kedad-Sidhoum, S. (2003). The one-machine problem with earliness and tardiness penalties. *Journal of Scheduling, 6*(6), 533–549.

Sourd, F., & Kedad-Sidhoum, S. (2007). A faster branch-and-bound algorithm for the earliness-tardiness scheduling problem. *Journal of Scheduling*, 1–10.

Syswerda, G. (1993). Simulated crossover in genetic algorithms. *Foundations of Genetic Algorithms, 2*, 239–255.

Valente, J. M. S., & Alves, R. A. F. S. (2005). Improved heuristics for the early/tardy scheduling problem with no idle time. *Computers and Operations Research, 32*(3), 557–569.

Valente, J. M. S., & Alves, R. A. F. S. (2007). Heuristics for the early/tardy scheduling problem with release dates. *International Journal of Production Economics, 106*(1), 261–274.

Wu, S. D., Storer, R. H., & Chang, P. C. (1993). One-machine rescheduling heuristics with efficiency and stability as criteria. *Computers and Operations Research, 20*(1), 1–14.

Zhang, Q., & Muhlenbein, H. (2004). On the convergence of a class of Estimation of Distribution Algorithms. *IEEE Transactions on Evolutionary Computation, 8*(2), 127–136.

Zhang, Q., Sun, J., & Tsang, E. (2005). An evolutionary algorithm with guided mutation for the maximum clique problem. *IEEE Transactions on Evolutionary Computation, 9*(2), 192–200.

Zhang, Q., Sun, J., & Tsang, E. (2007). Combinations of Estimation of Distribution Algorithms and other techniques. *International Journal of Automation and Computing, 4*(3), 273–280.

Zhang, J., & Szeto, K. Y. (2005). Mutation matrix in evolutionary computation: An application to resource allocation problem. *Lecture Notes in Computer Science, 3612*, 112–119.

Zhou, A., Zhang, Q., Jin, Y., Tsang, E., & Okabe, T. (2005). A model-based evolutionary algorithm for bi-objective optimization. In: *Proceedings of the 2005 Congress on Evolutionary Computation CEC-2005* (pp. 2568–2575). Edinburgh: IEEE Press.