Available online at www.sciencedirect.com
ScienceDirect
Applied Soft Computing 8 (2008) 767–777
Applied Soft Computing
www.elsevier.com/locate/asoc

Mining gene structures to inject artificial chromosomes for genetic algorithm in single machine scheduling problems

Pei-Chann Chang [a,b,*], Shih-Shin Chen [b], Chin-Yuan Fan [b]

[a] Department of Information Management, Yuan Ze University, Taoyuan 32026, Taiwan, ROC
[b] Department of Industrial Engineering and Management, Yuan Ze University, Taoyuan 32026, Taiwan, ROC

Received 18 November 2006; received in revised form 5 June 2007; accepted 10 June 2007
Available online 14 June 2007

Abstract

In this paper, a genetic algorithm with injecting artificial chromosomes is developed to solve the single machine scheduling problems. Artificial chromosomes are generated according to a probability matrix which is transformed from the dominance matrix by mining the gene structure of an elite base. A roulette wheel selection method is applied to generate an artificial chromosome by assigning genes onto each position according to the probability matrix. The higher the probability is, the more possible that the job will show up in that particular position. By injecting these artificial chromosomes, the genetic algorithm will speed up the convergence of the evolutionary processes. Intensive experimental results indicate that proposed algorithm is very encouraging and it can improve the solution quality significantly.

Keywords: Genetic algorithm; Single machine scheduling; Total deviation; Dominance properties

1. Introduction

More and more sophisticated evolutionary algorithms (EAs) have been proposed and developed to solve combinatorial problems in recent years. Some of them were quite successful; however, it is not always clear why and how an EA works. In this research, we take a close look at the evolutionary process for a single machine scheduling problem and come out with the new idea of generating artificial chromosomes to further improve the solution quality of the genetic algorithm.

An artificial chromosome is generated by a fitness-based gene mutation matrix to be injected into the evolutionary procedure. From the point of view of searching each gene allocation distribution, a simple gene mutation matrix (a probability matrix) is developed, which directly extracts the gene information from current chromosomes. In addition, the proposed approach is also applied to illustrate how insights gained which can be further converted into our understanding of EA's behaviors and guide us in developing new and better techniques. The proposed algorithm will be tested on a single machine scheduling problem with the objectives to minimize the total deviations. In addition, the proposed algorithm will be compared with a set of dominance properties developed by us in earlier studies to evaluate the effectiveness of the new algorithm.

The rest of the paper is organized as follows: Section 2 evaluates the literature in the field of genetic algorithm and single machine scheduling with the objective of minimizing total deviations. Section 3 describes the approach of mining gene structure information from the elite base, the transformation of a probabilistic matrix from dominance matrix and the generation procedure of the artificial chromosomes. Design of experiment for the parameter setup of the genetic algorithm with injecting AC is conducted and the intensive experiments are tested against other approaches in Section 4. Finally, Section 5 gives the conclusion and future development of the research.

2. Literature review

There are two categories of literature to be reviewed in this research: one is the genetic algorithm and the other is the single machine scheduling problem.

* Corresponding author at: Department of Information Management, Yuan Ze University, Taoyuan 32026, Taiwan, ROC. Tel.: +886 936101320; fax: +886 34638884.
E-mail address: iepchang@saturn.yzu.edu.tw (P.-C. Chang).

1568-4946/$ – see front matter
doi:10.1016/j.asoc.2007.06.005

2.1. Review of the genetic algortihm

Genetic algorithms (GAs) are powerful search techniques that are used successfully to solve problems in many different disciplines. The genetic algorithm relies on genetic operators for selection, crossover, mutation, and replacement. The selection operators use the fitness values to select a portion of the population to be parents for the next generation. Parents are combined using the crossover and mutation operators to produce offspring. This process combines the fittest chromosomes and passes superior genes to the next generation, thus providing new points in the solution space. The replacement operators ensure that the 'least fit' or weakest chromosomes of the population are displaced by more fit chromosomes. Application of GA in various scheduling and other application problems can be referred in refs. [3,7,11–14], however, as observed by most researchers the GA will be trapped into local optimality in the earlier stages and cannot be converged into global optimal in most of the cases. The problems with the steady states GAs having premature convergence led to the desire to further improve the convergence of the algorithm. Especially, for most combinatorial problems such as Traveling Salesman Problems (TSP), machine scheduling problems, and vehicles routing problems are very difficult to solve and even for moderate cases the GA will be converged prematurely.

Apart from our previous works which extract gene information directly, many researchers have presented new genetic domain-dependent operators, for instance, see refs. [15,25]. Nevertheless, no new biologically inspired genetic operators have been widely adopted since the advent of GAs. Mitchell and Forrest [26] point out the importance of studying new genetic operators. Mitchell and Forrest [26] and Mitchell [27] state that it would be interesting to analyze if any of these biological mechanisms, incorporated in a GA, could lead to any significant advantages. Banzhaf et al. [5] share the same opinion and they highlight the significance of implementing evolutionary approaches using mechanisms such as conjugation, transduction or transposition.

Jiao and Wang [18] introduce a novel genetic algorithm based on immunity (IGA), where a vaccine was regarded as a kind of knowledge related to the pending question and by vaccination the role of guidance function of the knowledge to evolutionary process. Yang et al. [40] present a novel algorithm called Ge–Ga, which combined a gene pool and a GA to direct the evolution of the whole population and efficiently improved the convergence speed and ability of finding the global optimal solution. Robles et al. [32] propose a hybrid algorithm which mixes the generic chromosomes and artificial chromosomes generated by estimation of distribution algorithm. Zhang and Szeto [41]] classify the above algorithm into EDA. For extensive review of evolutionary algorithm based on probability models, please refer to refs. [18,20,24]. Wang et al. [38] present a novel genetic algorithm (GTGA) with analogies to the concept and method of gene therapy theory. The core of GTGA lies on construction of a gene pool and a therapy operator. The method of creation and updating of the gene pool and construction of the therapy operator are very effective in restraining the premature convergence.

2.2. Review of the single machine scheduling problem

In this paper, a deterministic single machine scheduling problem without release date is investigated and the objective is to minimize the total sum of earliness and tardiness penalties. A detailed formulation of the problem is described as follows: A set of n independent jobs $\{J_1, J_2, \ldots, J_n\}$ has to be scheduled without preemptions on a single machine that can handle at most one job at a time. The machine is assumed to be continuously available from time zero onwards and unforced machine idle time is not allowed. Job J_j, $j = 1, 2, \ldots, n$ becomes available for processing at the beginning, requires a processing time p_j and should be completed on its due date d_j. For any given schedule, the earliness and tardiness of J_j can be respectively defined as $E_j = \max(0,\ d - C_j)$ and $T_j = \max(0,\ C_j - d)$, where C_j is the completion time of J_j. The objective is then to find a schedule that minimizes the sum of the earliness and tardiness penalties of all jobs $\sum_{j=1}^{n}(\alpha_j E_j + \beta_j T_j)$, where α_j and β_j are the earliness and tardiness penalties of job J_j. The inclusion of both earliness and tardiness costs in the objective function is compatible with the philosophy of just-in-time production, which emphasizes producing goods only when they are needed. The early cost may represent the cost of completing a product early, the deterioration cost for a perishable goods or a holding (stock) cost for finished goods. The tardy cost can represent rush shipping costs, lost sales and loss of goodwill. It is assumed that no unforced machine idle time is allowed, so the machine is only idle if no job is currently available for processing. This assumption reflects a production setting where the cost of machine idleness is higher than the early cost incurred by completing any job before its due date, or the capacity of the machine is limited when compared with its demand, so that the machine must indeed be kept running. Some specific examples of production settings with these characteristics are provided by refs. [4,30,31,34,35,39]. The set of jobs is assumed to be ready for processing at the beginning which is a characteristic of the deterministic problem. As a generalization of weighted tardiness scheduling, the problem is strongly NP-hard in ref. [21]. To the best of our knowledge, the earlier work in this problem is due to refs. [8–10,39]. Belouadah et al. [6] deal with the similar problem however with a different objective in minimizing the total weighted completion time and the problem is the same as discussed in ref. [17]. Kim and Shin [19] discuss some properties of the optimal solution, and these properties are used to develop both optimal and heuristic algorithms. Later on, Akturk and Ozdemir [2] develop various dominance rules to solve the problem. Valente and Alves [36] and Valente and Alves [37] present a branch-and-bound algorithm based on a decomposition of the problem into weighted earliness and weighted tardiness sub-problems. Two lower bound procedures are presented for each sub-problem, and the lower bound for the original problem is then simply the sum of the lower bounds for the two sub-problems. In ref. [36] they analyze the performance of various heuristic procedures,

including dispatch rules, a greedy procedure and a decision theory search heuristic.

The early/tardy problem with equal release dates and no idle time, however, has been considered by several authors, and both exact and heuristic approaches have been proposed. Among the exact approaches, branch-and-bound algorithms are presented by refs. [1,22,23]. The lower bounding procedure of ref. [1] is based on the sub-gradient optimization approach and the dynamic programming state–space relaxation technique, while refs. [22,23] use Lagrangean relaxation and the multiplier adjustment method. Among the heuristics, ref. [30] develop several dispatch rules and a filtered beam search procedure. In ref. [37], they present an additional dispatch rule and a greedy procedure, and also consider the use of dominance rules to further improve the schedule obtained by the heuristics. A neighborhood search algorithm is also presented by ref. [22].

3. Methodology

As surveyed in the literature, most approaches in solving the single machine scheduling problems are traditional optimization methods, such as branch-and-bound algorithm; dynamic programming; Lagrangean relaxation and Heuristics. Instead, the genetic algorithm is proposed in this research to solve the SME problems. However, to prevent the premature convergence, artificial chromosomes will be generated to speed up the convergence and jump out the local optimality to reach a near global optimal.

The first observation in this research is that during the evolving process of the GA, all the chromosomes will converge slowly into certain distribution after the final runs. If we take a close look at the distribution of each gene in each assigned position, we will find out that most the genes will be converged into certain locations which means the gene can be allocated to the position if there is a probabilistic matrix to guide the assignment of each gene to each position.

Artificial chromosomes are developed according to this observation and a dominance matrix will record this gene distribution information. The dominance matrix is transformed into a probability matrix to decide the next assignment of a gene to a position. Consequently, AC is integrated into the procedure of genetic algorithm and it attends to improve the performance of genetic algorithm. The primary procedure is to collect gene information first and to use the gene information to generate artificial chromosomes. Before collecting the gene information, AC collects the chromosomes whose fitness is better by comparing the fitness value of each chromosome with average fitness value of current population. Thus, the average fitness is calculated. A detailed procedure of the ACGA algorithm is depicted in Fig. 1.

A main procedure of the AC algorithm is listed as follows:

Main procedure
Population: The population used in the genetic algorithm.
Generations: The number of generations.
StartingGen: It determines when does the AC works.
Interval: The frequency to generate artificial chromosomes.

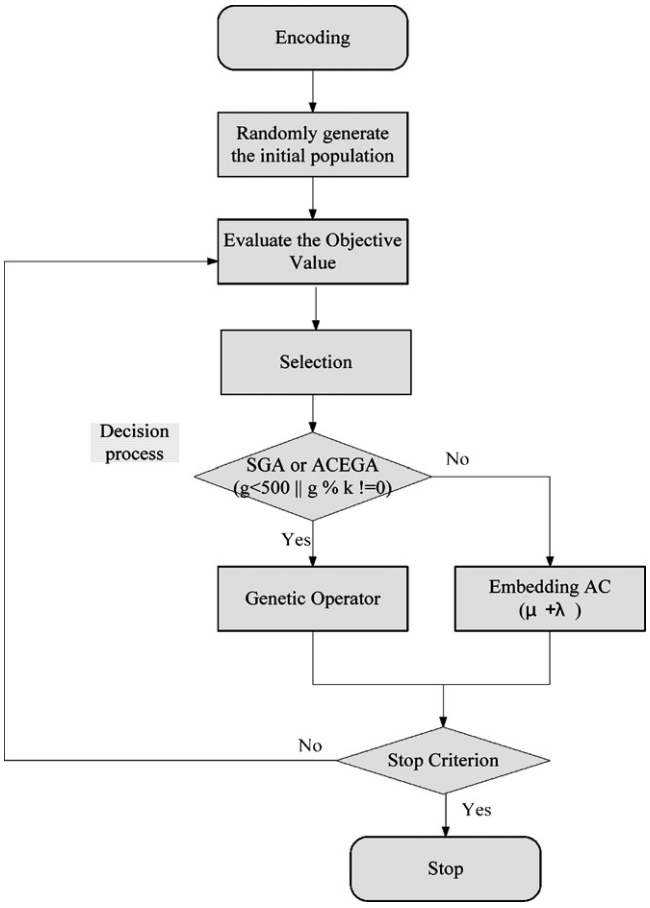

Fig. 1. The framework of the GA with injecting artificial chromosomes.

```
1.  Initiate Population
2.  counter ← 0
3.  while counter < generations do
4.      Evaluate Objective and Fitness()
5.      FindEliteSolutions(i)
6.      if counter < startingGen or counter %
    Interval! = 0 do
7.          Selection with Elitism Strategy()
8.          Crossover()
9.          Mutation()
10.     TotalReplacement()
11.     else
12.         CalculateAverageFitness()
13.         CollectGeneInformation()
14.         GenerateArtificialChromsomomes()
15.         μ + λ  Replacement()
16.     End if
17.     counter ← counter + 1
18. end while
```

The following is the detail procedures of the AC approach:

Step 1: Initial population generation

An initial population consists of a number of chromosomes. A chromosome represents a processing sequence for the scheduling problem. Therefore, it is feasible to use integer numbers and the permutation of the integers to represent the jobs and feasible sequences, respectively. Taking a five-job case as an example, {1, 2, 3, 4, and 5} represents the jobs to be processed. If the sequence is determined and represented as 2-1-5-4-3, that means the processing order is job 2, job 1, job 5, job 4, and job 3. To keep diversity in the initial population, random number generation is applied to generate initial sequences.

Step 2: Evaluate objective and fitness

The objective function of this scheduling problem is $\sum_{j=1}^{n}(E_j + T_j)$ for a set of job, which is explained in Section 1. Because this problem is a single objective problem, this objective value is also represented as the fitness of a chromosome.

Step 3: Genetic operators

After the fitness of each chromosome is calculated, the selection operator will choose better chromosomes to be survived. Although there are several selection operators, such as tournament selection, proportionate selection, and ranking selection. Since tournament selection has better convergence and computational time-complexity properties than others by ref. [16]. Therefore, a binary tournament operator is employed, which selects the better chromosomes with lower objective values in this minimization problem.

In the crossover step, two chromosomes are randomly selected and a random number r_c is generated first. If r_c is smaller than P_c, then crossover implements on this pair, else no crossover. In refs. [28,29], they reported that two-point crossover is effective in scheduling problems. Therefore, this study applies the two-point crossover operator to mate chromosomes.

Finally, a swap mutation operator is used because of its simplicity. There are two positions of a chromosome, which are randomly selected and a random number r_m is generated first. If r_m is smaller than P_m, then mutation implements on this pair, else no mutation. For example, a sequence is 2-9-5-3-4-8-10-6-7-1, and swapping positions 2 and 7 are assigned. The corresponding jobs at positions 2 and 7 are job 9 and job 10. Then the two jobs are interchanged and the sequence becomes 2-10-5-3-4-8-9-6-7-1 after mutation.

Step 4: Artificial chromosome operators

The artificial chromosome operator has two parameters, i.e., startingGen and generation interval, to be setup. The parameter configuration is done by design of experiment (DOE), which shows there is no significant difference. So the startingGen and generation interval are set to 500 and 50, respectively. The detail procedures of the artificial chromosome operator are described as follows:

Step 4.1: To calculate average fitness and to convert gene information into dominance matrix

Instead of collecting all gene information from a population, we select better chromosomes which are compared with the average fitness of the population in the current generation. For a better chromosome, if job i exists at position j, the number of occurrence of X_{ij} is incremented by 1. For chromosome k,

$$X_{ij} = \begin{cases} 1 & \text{if job } i \text{ assigned to position } j \\ 0 & \text{if job } i \text{ not in position } j \end{cases} \quad (1)$$

There are m chromosomes and the frequency of job i on position j (f_{ij}) within the dominance matrix $\mathbf{F}$, will be calculated as follows:

$$f_{ij}(t) = \sum_{k=1}^{n} X_{ij}^{k}, \quad i = 1, \ldots, n; \quad j = 1, \ldots, n; \quad k = 1, \ldots, m \quad (2)$$

Step 4.2: Generate artificial chromosomes

As soon as we collect gene information into dominance matrix, we are going to assign jobs onto the positions of each artificial chromosome. A probability matrix is further derived from the dominance matrix by the following:

$$P_{ij}(t) = \frac{\sum_{k=1}^{N} X_{ij}^{k}}{N} \quad (3)$$

where $P_{ij}(t)$ is the probability of job i to be assigned in position j.

$$P(t) = \begin{pmatrix} P_{11}(t) & \cdots & P_{1n}(t) \\ \vdots & \ddots & \vdots \\ P_{n1}(t) & \cdots & P_{nn}(t) \end{pmatrix} \quad (4)$$

The assignment sequence for every position is assigned randomly, which is able to diversify the artificial chromosomes.

Step 4.3: In order to enhance the convergence ability of the proposed algorithm, a $\mu + \lambda$ replacement strategy is applied. μ is the parent chromosomes and λ is the artificial chromosomes generated in Step 2. After we evaluate the fitness of artificial chromosomes, the parent chromosomes and artificial chromosomes are combined into together, whose population size is $\mu + \lambda$. Then, we select the size of μ from the $\mu + \lambda$ chromosomes deterministically. The new population becomes the parent chromosomes and the proposed algorithm employs it to continually evolve. Consequently, better solutions are preserved to the next generation.

To demonstrate the working theory of the artificial chromosome generation procedure, a five-job problem is illustrated. Suppose there are 10 sequences (chromosomes) whose fitness is better than average fitness. Then, we accumulate the gene information from these ten chromosomes to form a dominance matrix. As shown in the left-hand side of Fig. 2, there are two job 1, two job 2, two job 3, one job 4, and three job 5 on position. Again, there are three job 1, one job 2, two job 3, three job 4, and one job 5 on position 2. The procedure will repeat for the rest of the position. Finally, the dominance matrix contains the gene information from better chromosomes is illustrated in the right-hand side of Fig. 2.

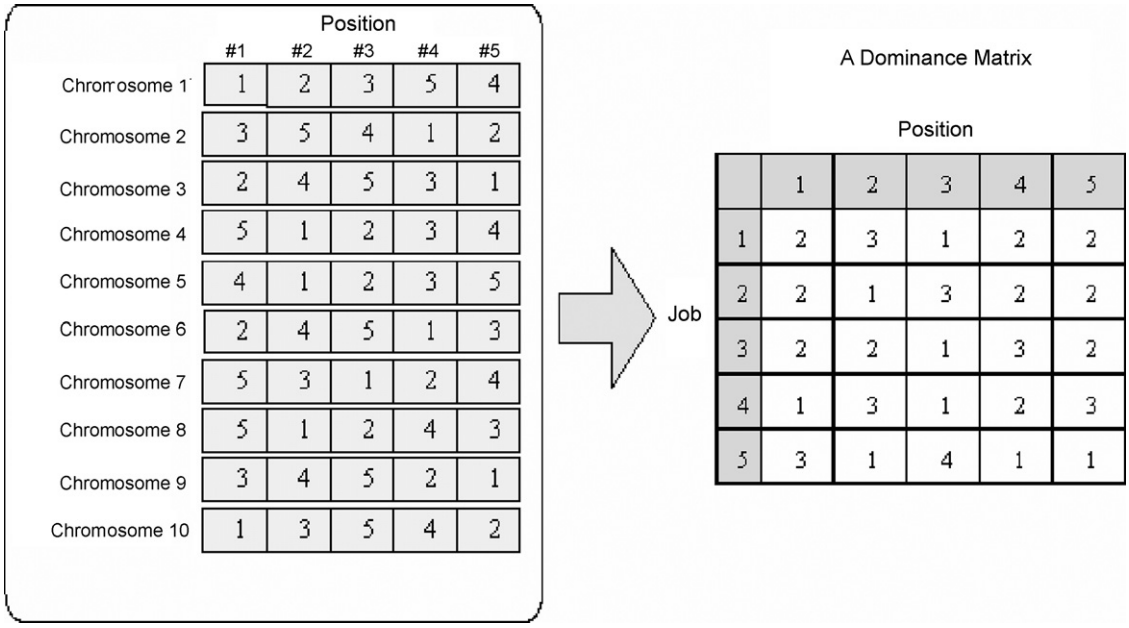

Fig. 2. Collecting gene information and converting it into a dominance matrix.

Then, the probabilistic matrix is further derived according to Eq. (3) from the dominance matrix and it shown in Table 1.

The probability distribution of each job assigned to each specific position is again shown in Fig. 3. In the earlier population, the diversity of the chromosome is still very wild therefore the probability distribution of each job onto each position is very even, i.e., close to 0.2(1/5).

To generate a sequence, each job has to be assigned a specific position according to a pre-defined assignment strategy. This paper studied two kinds of assignment strategies which are sequential (e.g., 1-2-3-4-5) and random assignment (e.g., 3-2-5-1-4). After the assignment strategy is determined, the job selected will be assigned to each position by a roulette wheel selection method based on the probability of each job on this position. After a job is assigned to a specific position, the data in the dominance matrix for that particular job and specific position are removed. Then, next job will be selected for assignment until all jobs are assigned.

For example, if the first job will be assigned at position 3 as shown in Fig. 3, the frequency of each job shown up at position 3 is [1, 3, 1, 1, and 4]. The corresponding probability for job 1 is 1/10; job 2 is 3/10, and so on. Then, the probability from job 1 to 5 is accumulated. The probability and accumulated probability of each job in position 3 is shown in Table 2.

According to the roulette wheel selection method, if a random probability generated is 0.6, then job 4 will be assigned to position 3. Assume the position 2 is the next one to be assigned, an updated dominance matrix is shown in Table 3. Again, following the same procedure, the probability of each job in position 2 is calculated as well as the accumulated probability. Then, a roulette wheel selection method will select a job based on the accumulated probability of each job. Consequently, all jobs will be assigned a specific position. Finally, a new job sequence according to the dominance matrix is generated.

4. Experimental result

To test the effectiveness of ACGA, we compared this algorithm with dominance properties, and GA with dominance properties. In addition, in order to make sure the proposed algorithm works well, single machine scheduling problems with the objective to minimize the early–tardy cost are presented. The testing instances are taken from [33] for

Table 1
A probabilistic matrix transformed from a dominance matrix

Jobs	Position				
	1	2	3	4	5
1	0.2	0.3	0.1	0.2	0.2
2	0.2	0.1	0.3	0.2	0.2
3	0.2	0.2	0.1	0.3	0.2
4	0.1	0.3	0.1	0.2	0.3
5	0.3	0.1	0.4	0.1	0.1

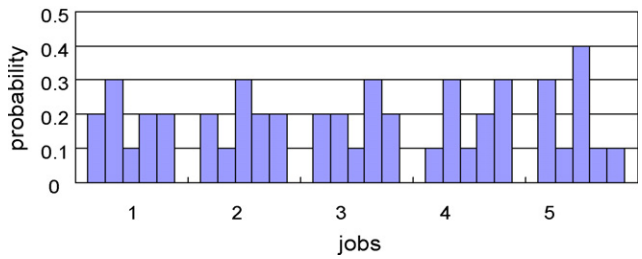

Fig. 3. The probability distribution of each job assigned to each specific position.

Table 2
The probability matrix and accumulated probability of each job in position 3

Job	Position 3	Probability	Accumulated
1	1	1/10	1/10
2	3	3/10	4/10
3	1	1/10	1/10
4	1	1/10	1/10
5	4	4/10	10/10

Table 3
An updated dominance matrix after assigning job 4 at position 3

Job \ Position	1	2	3	4	5
1	2	3	1	2	2
2	2	1	3	2	2
3	2	2	1	3	2
~~4~~	~~1~~	~~3~~	~~1~~	~~2~~	~~3~~
5	3	1	4	1	1

Table 4
The factors and the treatment of each factor

Factor	Treatments
Instance	20, 30, 40, 50, 60, 90
StartingGen	200, 500
Interval	25, 50
Assignseq	0 (sequential), 1 (random)

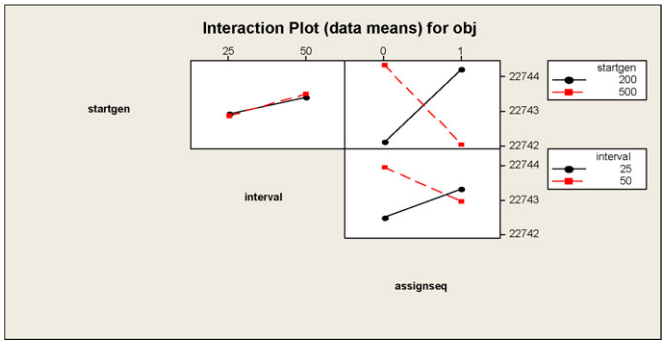

Fig. 4. The interaction plot of the ACGA in single machine scheduling.

benchmark tests. Since there are some parameters in the proposed algorithm, we did a design-of-experiment to determine the parameter settings in Section 4.1. The experimental comparison is shown in Section 4.2.

4.1. Parameter configurations in single machine problem

There are two parameters of the ACGA, including the starting generations and the interval or frequency of the ACGA executed. In addition, there are two methods to determine the assignment sequence for each position, which are sequential and random assignment. Three of them are the factors in the DOE. Each combination of these factors is replicated 30 times. Table 4 is the list of these factors and its corresponding levels.

Table 5 shows the ANOVA result of ACGA in single machine problem. There are four different factors to be evaluated and they are starting generation (when the AC procedure starts to function), interval (how many generations AC to be injected again), assigning sequence (the sequence for a job to be assigned to a specific position), and instance (the block factor and it is not necessary to analyze this effect). As can be observed from the table, the F- and P-value all shows that the interaction factor starting generation, assigning sequences and instance × interval × assignseq are significant. The interaction factor is significant if P-value is less than or equal to 0.05. Because the F-value of startingGen × assignseq is higher, i.e., 4.91, this interaction factor will be determined first.

Through the interaction plot as shown in Fig. 4, when the starting generation is 500 and the assignment method is 1 (the

Table 5
ANOVA result of ACGA in single machine problem

Source	d.f.	Seq SS	Adj SS	Adj MS	F	P
instance	5.30E+01	7.23E+12	7.23E+12	1.36E+11	42,643,119	**0.000**
startinggen	1	3	3	3.00	0.00	0.977
interval	1	1,008	1,008	1008.00	0.32	0.575
assignseq	1	27	27	27.00	0.01	0.927
instance × startingGen	53	93,711	93,711	1768.00	0.55	0.997
instance × interval	53	80,583	80,583	1520.00	0.48	1.000
Instance × assignseq	53	114,672	114,672	2164.00	0.68	0.966
startinggen × interval	1	19	19	19.00	0.01	0.938
startinggen × assignseq	1	15,712	15,712	15712.00	4.91	**0.027**
interval × assignseq	1	2,730	2,730	2730.00	0.85	0.356
instance × startingGen × interval	53	143,992	143,992	2717.00	0.85	0.774
instance × startingGen × assignseq	53	89,478	89,478	1688.00	0.53	0.998
instance × interval × assignseq	53	233,927	233,927	4414.00	1.38	**0.035**
startinggen × interval × assignseq	1	1,086	1,086	1086.00	0.34	0.560
instance × startingGen × interval × assignseq	53	101,760	101,760	1920.00	0.60	0.991
Error	12,528	4.01E+07	4.01E+07	3,199		
Total	12,959	7.23E+12				

Table 6
The parameter settings of ACGA in the single machine problem

Factor	Parameter setting
Starting generation	500
Interval	50
Assignment method	Random assign

random assignment), it yields a better result. Since the assignment method is determined, the interval of AC generation is set to every 50 generation randomly since it is not a significant factor.

Finally, the parameter setting of ACGA from ANOVA result is shown at Table 6.

Table 7
The minimum, mean, and maximum objective value of the four algorithms

Instance	SGA			GADP			ACGA			ACGADP		
	Min	Mean	Max	Min	Mean	Max	Min	Mean	Max	Min	Mean	Max
sks222a	5,286	5,402	5,643	5,286	5,291	5,298	5,286	5,289	5,298	5,286	5288.7	5,298
sks225a	3,958	4,174	4,389	3,958	3,959	3,977	3,958	3,958	3,958	3,958	3,958	3,958
sks228a	2,085	2,156	2,749	2,085	2,085	2,085	2,085	2,085	2,085	2,085	2,085	2,085
sks252a	4,052	4,195	4,508	3,947	3,947	3,947	3,947	3,979	4,067	3,947	3,947	3,947
sks255a	2,388	2,489	2,787	2,372	2,372	2,372	2,372	2,380	2,388	2,372	2372.5	2,388
sks258a	1,184	1,250	1,371	1,184	1,242	1,248	1,184	1,200	1,248	1,184	1192.7	1,248
sks282a	4,348	4,435	4,695	4,348	4,353	4,355	4,348	4,351	4,355	4,348	4353.8	4,355
sks285a	4,452	4,643	4,895	4,452	4,452	4,452	4,452	4,452	4,452	4,452	4,452	4,452
sks288a	3,421	3,518	3,847	3,421	3,421	3,421	3,421	3,421	3,421	3,421	3,421	3,421
sks322a	11,623	12,066	12,916	11,568	11,572	11,622	11,568	11,577	11,622	11,568	11,570	11,622
sks325a	7,615	8,152	9,650	7,587	7,703	7,904	7,587	7,587	7,587	7,587	7,587	7,587
sks328a	3,195	3,556	4,810	3,164	3,164	3,164	3,164	3,164	3,164	3,164	3,164	3,164
sks352a	7,588	8,203	9,063	7,395	7,395	7,395	7,392	7,394	7,395	7,392	7394.2	7,395
sks355a	6,202	6,849	7,693	6,056	6,068	6,212	6,056	6,065	6,193	6,056	6057.5	6,058
sks358a	3,104	3,283	3,787	3,069	3,074	3,076	3,069	3,073	3,076	3,069	3072.5	3,076
sks382a	11,157	11,319	11,558	11,140	11,152	11,160	11,140	11,149	11,222	11,140	11,142	11,154
sks385a	9,157	9,212	9,389	9,148	9,148	9,148	9,148	9,148	9,148	9,148	9,148	9,148
sks388a	11,321	11,499	11,789	11,317	11,317	11,317	11,317	11,317	11,317	11,317	11,317	11,317
sks422a	25,656	26,211	27,462	25,656	25,658	25,712	25,656	25,659	25,704	25,656	25,657	25,687
sks425a	12,725	13,592	15,198	12,601	12,604	12,605	12,601	12,606	12,668	12,601	12,601	12,601
sks428a	7,237	7,741	8,761	7,129	7,129	7,141	7,129	7,129	7,129	7,129	7,129	7,129
sks452a	11,804	12,634	14,053	11,367	11,367	11,367	11,367	11,406	11,581	11,367	11,367	11,367
sks455a	6,573	7,566	9,435	6,405	6,405	6,405	6,405	6,427	6,666	6,405	6,405	6,405
sks458a	4,424	5,587	8,331	4,294	4,303	4,319	4,294	4,321	4,391	4,294	4300.4	4,306
sks482a	19,656	20,122	20,605	19,559	19,580	19,735	19,559	19,573	19,735	19,559	19,562	19,572
sks485a	15,459	16,023	16,576	15,256	15,309	15,480	15,256	15,338	15,480	15,256	15,350	15,480
sks488a	17,374	17,999	18,668	16,862	16,881	16,906	16,862	16,863	16,888	16,862	16,863	16,888
sks522a	29,485	30,623	32,340	29,309	29,322	29,507	29,309	29,312	29,396	29,309	29,310	29,311
sks525a	25,693	26,113	26,713	25,433	25,436	25,517	25,433	25,438	25,469	25,433	25,434	25,444
sks528a	11,154	12,037	13,121	10,798	10,821	10,823	10,798	10,838	11,129	10,798	10,804	10,823
sks552a	23,491	24,827	26,241	22,863	22,863	22,863	22,863	22,894	23,148	22,863	22,863	22,863
sks555a	10,877	12,233	14,626	10,207	10,243	10,446	10,187	10,216	10,299	10,207	10,209	10,226
sks558a	5,776	7,345	9,358	5,269	5,298	5,416	5,269	5,269	5,269	5,269	5,269	5,269
sks582a	28,375	29,154	30,551	27,939	28,117	28,352	27,939	27,947	28,056	27,939	27,986	28,284
sks585a	25,200	25,862	26,447	24,828	24,830	24,853	24,828	24,839	24,853	24,828	24,850	24,853
sks588a	25,453	26,422	28,197	24,844	24,846	24,856	24,844	24,845	24,861	24,844	24,845	24,861
sks622a	43,930	45,018	46,017	43,048	43,048	43,048	43,048	43,120	43,479	43,048	43,048	43,048
sks625a	25,563	26,672	27,957	25,253	25,253	25,253	25,229	25,260	25,307	25,253	25,253	25,253
sks628a	17,463	18,431	20,707	17,047	17,057	17,123	17,047	17,059	17,162	17,047	17,055	17,123
sks652a	31,292	33,022	35,426	30,801	30,801	30,801	30,801	30,871	31,080	30,801	30,801	30,801
sks655a	17,409	19,137	23,546	16,158	16,158	16,158	16,158	16,218	16,635	16,158	16,158	16,158
sks658a	9,948	12,715	18,469	9,623	9,623	9,626	9,623	9,655	9,724	9,623	9623.2	9,626
sks682a	38,930	39,722	41,137	38,836	38,940	39,109	38,714	38,749	38,863	38,744	38,923	39,109
sks685a	38,736	39,744	41,345	38,084	38,096	38,166	38,084	38,103	38,166	38,084	38,090	38,166
sks688a	34,456	35,826	37,229	33,551	33,654	33,665	33,551	33,639	33,665	33,551	33,646	33,665
sks922a	91,516	93,966	96,966	88,994	89,606	90,514	88,841	88,894	89,067	88,866	89,315	90,493
sks925a	74,327	76,438	79,979	72,038	72,045	72,141	72,038	72,065	72,114	72,038	72,055	72,120
sks928a	38,676	41,879	49,091	33,825	33,992	34,159	33,830	33,973	34,138	33,903	34,019	34,195
sks952a	73,718	76,863	79,847	68,150	68,188	68,441	68,150	68,288	68,674	68,150	68,179	68,253
sks955a	35,647	40,444	45,820	30,660	30,664	30,700	30,582	30,683	31,312	30,590	30,663	30,697
sks958a	23,553	30,662	39,623	19,945	19,972	20,028	19,950	20,025	20,201	19,954	20,012	20,108
sks982a	1E+05	1E+05	1E+05	98,613	99,041	99,349	98,613	98,644	98,832	98,613	99,081	99,349
sks985a	82,254	84,966	87,173	78,296	78,442	78,532	78,296	78,414	78,520	78,296	78,445	78,557
sks988a	88,094	91,422	96,318	81,984	81,993	82,097	81,984	82,002	82,053	81,984	81,995	82,045

4.2. Experimental results of the single machine scheduling problem

Sourd and Sidhoum [33] provided numerous data sets, including 20, 30, 40, 50, 60, and 90 for test. Each job set of 20-job to 50-job problems contains 49 combinations while there are only 9 instances in the job set for 60-job and 90-job problems. We carried out our experiment on these total 214 instances and each instance is replicated for 30 times. The stopping criterion is the number of examined solutions reaching

Table 8
The median and standard deviation of the four algorithms

Instance	SGA		GADP		ACGA		ACGADP	
	Median	S.D.	Median	S.D.	Median	S.D.	Median	S.D.
sks222a	5,372	88.3	5,286	5.53	5,286	5.16	5,286	5.16
sks225a	4,179	104.4	3,958	3.47	3,958	0	3,958	0
sks228a	2127.5	124.3	2,085	0	2,085	0	2,085	0
sks252a	4,166	126.4	3,947	0	3,947	54	3,947	0
sks255a	2447.5	113.9	2,372	0	2,380	8.14	2,372	2.92
sks258a	1,248	35.8	1,248	19.5	1,184	27.8	1,184	22.1
sks282a	4,399	105.1	4,355	3.15	4,348	3.49	4,355	2.65
sks285a	4,656	133.4	4,452	0	4,452	0	4,452	0
sks288a	3490.5	97.4	3,421	0	3,421	0	3,421	0
sks322a	12,048	269	11,568	13.7	11,568	18.7	11,568	9.86
sks325a	8058.5	421.2	7,587	155.4	7,587	0	7,587	0
sks328a	3,475	375.1	3,164	0	3,164	0	3,164	0
sks352a	8170.5	412.5	7,395	0	7,395	1.49	7,395	1.35
sks355a	6824.5	369.9	6,058	39.2	6,058	31.5	6,058	0.9
sks358a	3180.5	210.6	3,076	3.01	3,076	3.43	3072.5	3.56
sks382a	11,329	130	11,154	6.64	11,140	15.9	11,140	4.84
sks385a	9,199	46.5	9,148	0	9,148	0	9,148	0
sks388a	11,438	146	11,317	0	11,317	0	11,317	0
sks422a	26,079	401	25,656	10.2	25,656	12.2	25,656	5.67
sks425a	13,656	501	12,605	1.38	12,601	14	12,601	0
sks428a	7720.5	319.2	7,129	2.19	7,129	0	7,129	0
sks452a	12,658	552	11,367	0	11,367	60.3	11,367	0
sks455a	7,412	661	6,405	0	6,405	55.4	6,405	0
sks458a	5,394	890	4,305	9.74	4,306	27.4	4,306	6.09
sks482a	20,103	262	19,572	43.1	19,572	31.2	19,559	5.29
sks485a	16,034	331	15,256	90.5	15,256	96.7	15,430	90.3
sks488a	18,014	355	16,888	17.4	16,862	4.75	16,862	4.75
sks522a	30,487	694	29,311	44.9	29,309	15.8	29,309	0.9
sks525a	26,157	281	25,433	15.3	25,433	12.4	25,433	2.79
sks528a	11,963	489	10,823	6.34	10,798	94.7	10,798	10.8
sks552a	24,705	740	22,863	0	22,874	63.9	22,863	0
sks555a	11,997	988	10,227	44.7	10,207	28.6	10,207	5.8
sks558a	7,159	984	5,269	53.4	5,269	0	5,269	0
sks582a	29,138	537	28,167	118	27,939	22.6	27,939	119
sks585a	25,840	273	24,828	6.84	24,828	12.6	24,853	8.64
sks588a	26,241	576	24,844	3.98	24,844	3.46	24,844	3.46
sks622a	45,172	604	43,048	0	43,048	112	43,048	0
sks625a	26,759	545	25,253	0	25,253	17.2	25,253	0
sks628a	18,026	837	17,047	21.3	17,047	34.8	17,047	21.7
sks652a	32,983	962	30,801	0	30,801	106	30,801	0
sks655a	18,879	1,346	16,158	0	16,158	152	16,158	0
sks658a	12,159	1,921	9,623	0.548	9,624	40.7	9,623	0.761
sks682a	39,572	571	38,934	94.2	38,716	53.2	38,923	109
sks685a	39,819	607	38,084	27.3	38,084	33.8	38,084	20.3
sks688a	35,858	646	33,665	34.2	33,665	48.7	33,665	42.3
sks922a	94,129	1,356	89,408	499	88,853	75.3	89,388	353
sks925a	76,389	1,365	72,041	18.7	72,065	24.4	72,043	21.6
sks928a	41,419	2,535	34,015	73.1	33,955	86.4	34,024	66.2
sks952a	76,780	1,446	68,150	66.2	68,220	153	68,150	45.9
sks955a	40,066	2,457	30,660	8.98	30,645	143	30,667	18.2
sks958a	29,897	3,751	19,963	22	20,012	59	20,007	41.1
sks982a	102,129	1,420	99,064	258	98,637	52.3	99,147	272
sks985a	85,093	1,220	78,434	46	78,434	65.4	78,435	39.9
sks988a	91,902	1,838	81,985	21.7	81,989	22.2	81,987	15.4

Table 9
The Duncan grouping result for the four algorithms

Duncan	Grouping	Mean	N	Method
	A	13982.894	6,420	SGA
	B	12827.096	6,420	GADP
	B			
C	B	12816.471	6,420	ACGADP
C				
C		12813.276	6,420	ACGA

Table 10
Wilcoxon scores (Rank Sums) for objective values classified by these four algorithms

Method	N	Sum of scores	Expected under H_0	S.D. under H_0	Mean score
ACGADP	6,420	81,162,861	82,436,010	514410.5	12642.19
ACGA	6,420	81,180,297	82,436,010	514410.5	12644.91
GADP	6,420	81,256,352	82,436,010	514410.5	12656.75
SGA	6,420	86,144,531	82,436,010	514410.5	13418.15

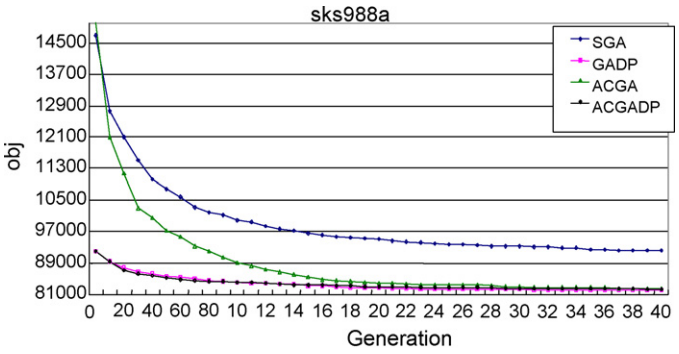

Fig. 5. The convergence diagram of the four algorithms for 90-job problem (sks988a).

100,000 solutions. The parameters of GA include the crossover rate, mutation rate, and population size is set as 0.8, 0.5, and 100, respectively. The proposed algorithm is compared with a simple genetic algorithm (SGA), a genetic algorithm with dominance properties (GADP) that was proposed by our previous work, a GA with injecting AC (ACGA), and a hybrid algorithm ACGADP. The GADP applies a heuristic to generate a good initial population in the beginning and it is able to enhance the exploration ability of simple genetic algorithm. Finally, an average relative error is applied as a performance metric that shows each average objective with respect to its optimal solution. The equation is calculated by (avgObj − Opt)/Opt × 100%, where the avgObj is the average objective

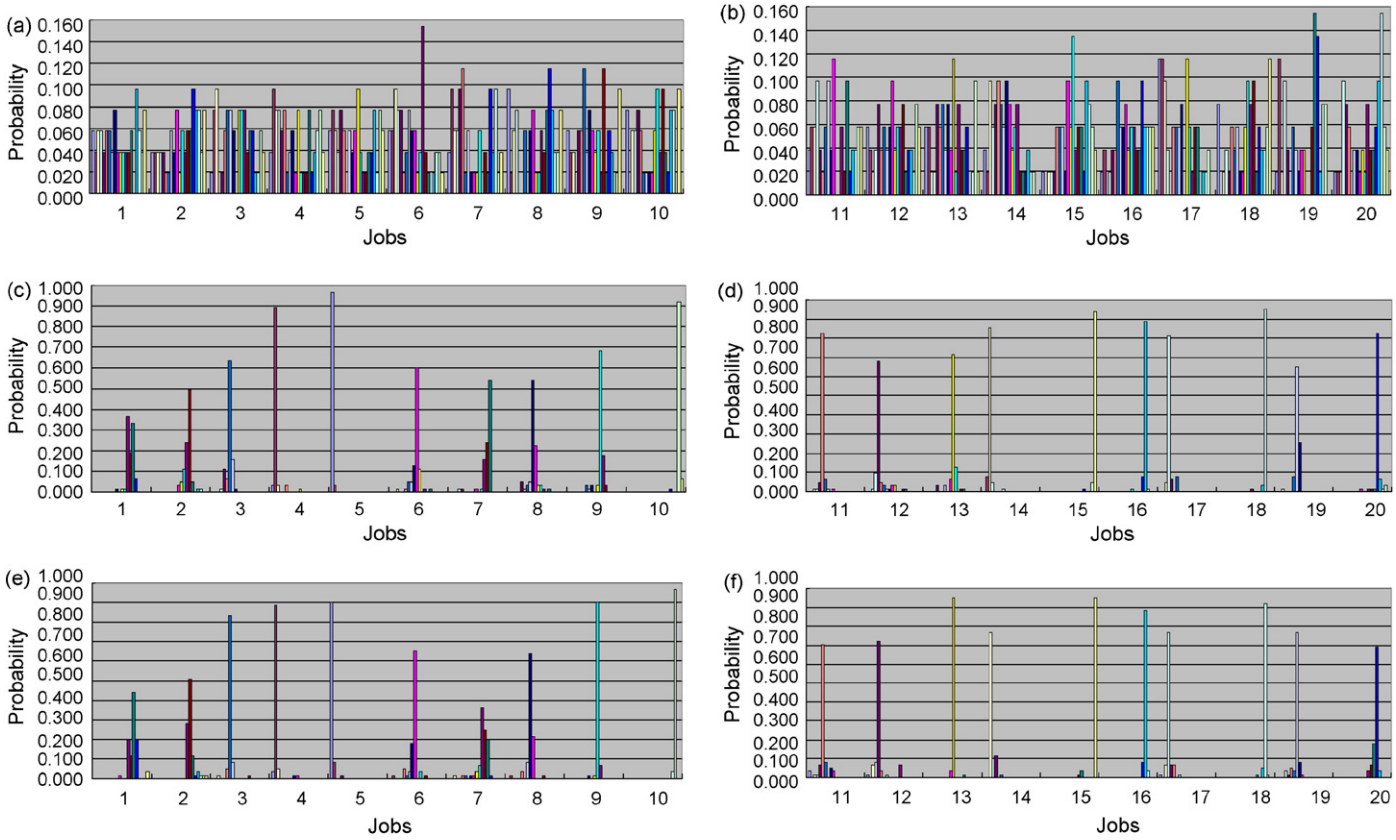

Fig. 6. The probability distribution of a 20-job case in different position for generation, 0, 500 and 1000 (a–f). (a) The probability distribution of job 1 to job 10 in each different position for generation 0 (a different color means a different position). (b) The probability distribution of job 11 to job 20 in each different position for generation 0 (a different color means a different position). (c) The probability distribution of job 1 to job 10 in each different position for generation 500 (a different color means a different position). (d) The probability distribution of job 11 to job 20 in each different position for generation 500 (a different color means a different position). (e) The probability distribution of job 1 to job 10 in each different position for generation 1000 (a different color means a different position). (f) The probability distribution of job 11 to job 20 in each different position for generation 1000 (a different color means a different position).

value obtained by each algorithm. These results are depicted in Tables 7 and 8. The completed test results are available on our website.[1]

GADP, ACGA, and ACGADP outperform the SGA in the average error ratio because the total average ratio of SGA is 9.971% while other three algorithms is less than 0.26%. Then, the standard deviation of ACGADP is smaller than others. To compare the performance of these four algorithms, the ANOVA test shows there is a significant difference among these methods. Thus, the Duncan grouping method, Kruskall–Wallis test, and Rank Sums are further applied to compare these algorithms using SAS software package. Table 9 is the Duncan grouping result, which shows that there is no difference between ACGA and ACGADP. In addition, there is no difference between ACGADP and GADP. However, ACGA, ACGADP, and SGA are not in the same group, it means there is a significant difference between each of them. Consequently, ACGA is the best algorithm, ACGADP and GADP are second rank, and SGA is the worst.

The Kruskall–Wallis test shows the Chi-square of the four algorithms is 51.9875 and the corresponding P-value is less than 0.0001, which shows there is a significant difference among these algorithms. Table 10 presents the Rank Sums of the algorithms. The sum of scores of SGA is 86,144,531 which is far from the expected under H_0 (say 82,436,010) compared to ACGADP, ACGA, and GADP. Hence, the SGA is indeed worse method than others.

To show the convergence process for these difference algorithms, i.e., SGA, GADP, ACGA and ACGADP, instance sks988a is applied as a demonstration. Fig. 5 shows that GADP has the quickest convergence then ACGA and lastly is SGA. However, after 20 generation ACGA has almost the same solution quality as GADP.

In addition, the probabilistic matrix of a 20-job case is shown in Fig. 6 for initial generation, 500 generation and 999 generation. The figure shows that each job will be converged to a finite position or finite positions. That is the most of the jobs will have a large probability which is very close to one for one particular position while the rest of the probabilities will be very close to zero in other positions.

5. Conclusions

This research develops a genetic algorithm with injecting artificial chromosomes in solving the single machine scheduling problems with the objective of minimizing the total deviation. From the experimental results, we find out that the proposed algorithm is able to obtain a very good solution quality when compared with SGA and GADP. Without any complex mathematic calculation and proofing procedure, ACGA can solve the problem in as good as or a better solution quality than GADP. The reason is that the dominance matrix can truly capture the gene information and prohibits jobs that are assigned to inappropriate positions. Then, jobs are potentially to be assigned to a position with higher probability by roulette wheel selection method.

One of the major advantages of injecting artificial chromosomes to GA is that it can start with any good initial solutions generated from Meta heuristics or dominance properties. ACGA will create a much diversified population according to the probability matrix transformed from the best initial seeds and finally converge to reach a near-optimal solution. Consequently, after the intensive experiments in the single machine scheduling problem, the result is very satisfactory and convincing and we expect to apply the ACGA to other combinatorial problems in the near future.

[1] http://ppc.iem.yzu.edu.tw/publication/sourceCodes/InjectionArtificialChromosomes/.

References

[1] T. Abdul-Razaq, C.N. Potts, Dynamic programming state-space relaxation for single machine scheduling, J. Operat. Res. Soc. 39 (1988) 141–152.
[2] M.S. Akturk, D. Ozdemir, A new dominance rule to minimize total weighted tardiness with unequal release dates, Eur. J. Operat. Res. 135 (2001) 394–412.
[3] M.S. Arumugam, M.V.C. Rao, R. Palaniappan, New hybrid genetic operators for real coded genetic algorithm to compute optimal control of a class of hybrid systems, Appl. Soft Comput. 6 (1) (2005) 38–52.
[4] M. Azizoglu, S. Kondakci, K. Ömer, Bicriteria scheduling problem involving total tardiness and total earliness penalties, Int. J. Prod. Econ. 23 (1–3) (1991) 17–24.
[5] W. Banzhaf, P. Nordin, R.E. Keller, F.D. Francone, Genetic Programming. An Introduction—On the Automatic Evolution of Computer Programs and its Applications, Morgan Kaufmann, CA, 1998.
[6] H. Belouadah, M.E. Posner, C.N. Potts, Scheduling with release dates on a single machine to minimize total weighted completion time, Discrete Appl. Math. 36 (1992) 213–231.
[7] Z. Bingul, Adaptive genetic algorithms applied to dynamic multiobjective problems, Appl. Soft Comput. 7 (3) (2007) 791–799.
[8] P.C. Chang, A branch and bound approach for single machine scheduling with earliness and tardiness penalties, Comput. Math. Appl. 37 (1999) 133–144.
[9] P.C. Chang, H.C. Lee, A greedy heuristic for bi-criterion single machine scheduling problems, Comput. Ind. Eng. 22 (2) (1992) 121–131.
[10] P.C. Chang, H.C. Lee, A two phase approach for single machine scheduling: minimizing the total absolute deviation, J. Chin. Inst. Eng. 15 (6) (1992) 735–742.
[11] P.-C. Chang, J.C. Hsieh, Y.W. Wang, Genetic algorithms applied in BOPP film scheduling problems, Appl. Soft Comput. 3 (2003) 139–148.
[12] P.-C. Chang, J.C. Hsieh, C.Y. Wang, Adaptive multi-objective genetic algorithms for scheduling of drilling operation in printed circuit board industry, Appl. Soft Comput. 7 (3) (2007) 800–806.
[13] P.-C. Chang, J.C. Hsieh, C.H. Liu, A case-injected genetic algorithm for single machine scheduling problems with release time, Int. J. Prod. Econ. 103 (2) (2006) 551–564.
[14] P.-C. Chang, S.H. Chen, K.L. Lin, Two phase sub-population genetic algorithm for parallel machine scheduling problem, Expert Syst. Appl. 29 (3) (2005) 705–712.
[15] P. D'Haeseleer, Context preserving crossover in genetic programming, in: Proceedings of the 1994 IEEE World Congress on Computational Intelligence, vol. 1, 1993, pp. 256–261.
[16] D.E. Goldberg, K. Deb, A comparison of selection schemes used in genetic algorithms, in: G.J.E. Rawlins (Ed.), Foundations of Genetic Algorithms, Morgan Kaufmann Publishers, San Mateo, CA, USA, 1991 pp. 69–93.
[17] A.M.A. Hariri, C.N. Potts, Scheduling with release dates on a single machine to minimize total weighted completion time, Discrete Appl. Math. 36 (1983) 99–109.

[18] L.C. Jiao, L. Wang, A novel genetic algorithm based on immunity, IEEE Trans. Syst. Man Cybern. Part A 30 (2000) 552–561.
[19] H.J. Kim, K.S. Shin, A hybrid approach based on neural networks and genetic algorithms for detecting temporal patterns in stock markets, Appl. Soft Comput. 7 (2) (2007) 569–576.
[20] P. Larrañaga, J.A. Lozano, Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation, Kluwer, Norwell, MA, 2001.
[21] J.K. Lenstra, A.H.G. Rinnooy Kan, P. Brucker, Complexity of machine scheduling problems, Ann. Discrete Math. 1 (1977) 343–362.
[22] G. Li, Single machine earliness and tardiness scheduling, Eur. J. Operat. Res. 96 (1997) 546–558.
[23] C.-F. Liaw, A branch-and-bound algorithm for the single machine earliness and tardiness scheduling problem, Comput. Operat. Res. 26 (1999) 679–693.
[24] J.A. Lozano, P. Larrañaga, I. Inza, E. Bengoetxea, Towards a New Evolutionary Computation, Springer, 2006.
[25] K. Mathias, D. Whitley, Genetic operators, the fitness landscape and the traveling salesman problem, in: R. Männer, B. Manderick (Eds.), Proceedings of Parallel Problem Solving from Nature, vol. 2, 1992, pp. 219–228.
[26] M. Mitchell, S. Forrest, Genetic algorithms and artificial life, Artif. Life 1 (3) (1994) 267–289.
[27] M. Mitchell, An Introduction to Genetic Algorithms, MIT Press, 1996.
[28] T. Murata, H. Ishibuchi, Performance evolution of genetic algorithms for flowshop scheduling problems, in: Proceedings of 1st IEEE International Conference on Evolutionary Computation, 1994, pp. 812–817.
[29] T. Murata, H. Ishibuchi, H. Tanaka, Genetic algorithms for flowshop scheduling problems, Comput. Ind. Eng. 30 (1996) 1061–1071.
[30] P.S. Ow, E.T. Morton, The single machine early/tardy problem, Manage. Sci. 35 (1989) 177–191.
[31] P.S. Ow, T.E. Morton, Filtered beam searches in scheduling, Int. J. Prod. Res. 26 (1988) 35–62.
[32] J.M. Robles, V. Pena, P. Larranaga, V. Herves, F. Rosales, M.S. Perez, GA-EDA: hybrid evolutionary algorithm using genetic and estimation of distribution algorithms, Lecture Notes Artif. Intell. (2004) 361–371.
[33] F. Sourd, S.K. Sidhoum, An Efficient Algorithm for the Earliness Tardiness, 2005. http://www-poleia.lip6.fr/sourd/project/et/.
[34] L.H. Su, P.C. Chang, A heuristic to minimize a quadratic function of job lateness on a single machine, Int. J. Prod. Econ. 55 (2) (1998) 169–175.
[35] L.H. Su, P.C. Chang, Scheduling *n* jobs on one machine to minimize the maximum lateness with a minimum number of tardy jobs, Comput. Ind. Eng. 40 (4) (2001) 349–360.
[36] J.M.S. Valente, R.A.F.S. Alves, Heuristics for the early/tardy scheduling problem with release dates, Working Paper 129, Faculdade de Economia do Porto, Portugal, 2003.
[37] J.M.S. Valente, R.A.F.S. Alves, Improved heuristics for the early/tardy scheduling problem with no idle time, Working Paper 126, Faculdade de Economia do Porto, Portugal, 2003.
[38] C.X. Wang, D.U. Cui, D.S. Wan, L. Wang, A novel genetic algorithm based on gene therapy theory, Trans. Inst. Meas. Control 28 (3) (2006) 253–262.
[39] S.D. Wu, R.H. Storer, P.C. Chang, One machine rescheduling heuristic with efficiency and stability as criteria, Comput. Operat. Res. 20 (1) (1993) 1–14.
[40] H. Yang, L.S. Kang, Y.P. Chen, A gene-based genetic algorithm for TSP, Chin. J. Comput. 26 (2003) 1753–1758.
[41] J. Zhang, K.Y. Szeto, Mutation Matrix in Evolutionary Computation: An Application to Resource Allocation Problem, vol. 3612, Springer, Berlin, 2005, pp. 112–119.