Expert Systems with Applications 36 (2009) 7135–7141

Contents lists available at ScienceDirect

Expert Systems with Applications

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

Artificial chromosomes embedded in genetic algorithm for a chip resistor scheduling problem in minimizing the makespan

Pei-Chann Chang [a,c,*], Jih-Chang Hsieh [b], Shih-Hsin Chen [d], Jun-Lin Lin [a], Wei-Hsiu Huang [c]

[a] *Department of Information Management, Yuan Ze University, 135 Yuan Tung Road, Taoyuan 320, Taiwan, ROC*
[b] *Department of Finance, Vanung University, Chung-Li, Tao-Yuan, Taiwan, ROC*
[c] *Department of Information Management, Yuan Ze University, Taoyuan 320, Taiwan, ROC*
[d] *Department of Electronic Commerce Management, Nanhua University, 32 Chungkeng, Dalin, Chiayi 62248, Taiwan, ROC*

ARTICLE INFO

Keywords:
Artificial chromosome
Genetic algorithm
Flowshop scheduling
Makespan

ABSTRACT

The manufacturing processes of a chip resistor are very similar to a flowshop scheduling problem only with minor details which can be modeled using some extra constraints; while permutation flowshop scheduling problems (PFSPs) have attracted much attention in the research works. Many approaches like genetic algorithms were dedicated to solve PFSPs effectively and efficiently. In this paper, a novel approach is presented by embedding artificial chromosomes into the genetic algorithm to further improve the solution quality and to accelerate the convergence rate. The artificial chromosome generation mechanism first analyzes the job and position association existed in previous chromosomes and records the information in an association matrix. An association matrix is generated according to the job and position distribution from top 50% chromosomes. Artificial chromosomes are determined by performing a roulette wheel selection according to the marginal probability distribution of each position. Two types of PFSPs are considered for evaluation. One is a three-machine flowshop in the printing operation of a real-world chip resistor factory and the other is the standard benchmark problems retrieved from OR-Library. The result indicates that the proposed method is able to improve the solution quality significantly and accelerate the convergence process.

1. Introduction

The chip form electrical resistance is designed to be soldered notably on a printed circuit card or on a hybrid circuit substratum as shown in Fig. 1. It includes an electrically insulating substratum of the ceramic type, to which is attached by a layer of adhesive organic resin a sheet of metal or of resistive alloy which is engraved to provide a sinuous resistance. The layer of resin leaves in the area of the two opposite sides of the substratum, two free areas, at the extremities of the engraved resistive sheet. These two parts of the resistive sheet are each covered by a thin layer of a metal or alloy adhering to the resistive sheet, this layer being covered by a second thicker layer of metal or conductive alloy, and this second layer being covered by a third, also thicker layer of a solderable metal, these three superimposed layers spreading equally over both lateral sides opposite the substratum and partially on its face opposite the engraved resistive sheet.

Chip resistor is a key component to electronic appliances manufacturing industry. Electronic appliances today have to cater the need of customers. This makes the specifications of chip resistances diversified. It is not an easy task to manage so many specifications. Actually the printing is the bottleneck operation in this industry. To improve the performance of the bottleneck, it may be of interest to apply a new scheduling technique.

2. Literature reviews

Scheduling problems have been applied in many practical fields. For example, Caccetta and Hill (2003) discussed the open pit mine scheduling problem, Bai and Elhafsi (1996) dealt with a manufacturing system with setup changes, and Dror and Mullaseril (1996) concerned the scheduling problem on civil engineering projects. In this study, the scheduling problem in a real-world chip resistor factory is presented. The production process can be defined as a flowshop system. The properties of a permutation flowshop are described as follows:

A permutation flowshop scheduling problem (PFSP) concerns that n jobs $\{J_1, J_2, \ldots, J_n\}$ are processed on m machines $\{M_1, M_2, \ldots, M_m\}$ in the same order to meet one or some specified objectives. Baker (1974) summarized the assumptions of permutation flowshop scheduling problems including:

1. Each job is allowed to be processed at most on one machine at the same time.

* Corresponding author. Address: Department of Industrial Engineering and Management, Yuan Ze University, 135 Yuan Tung Road, Taoyuan 320, Taiwan, ROC. Tel.: +886 3 4638800x2305; fax: +886 3 4638884.
E-mail address: iepchang@saturn.yzu.edu.tw (P.-C. Chang).

0957-4174/$ - see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2008.08.047

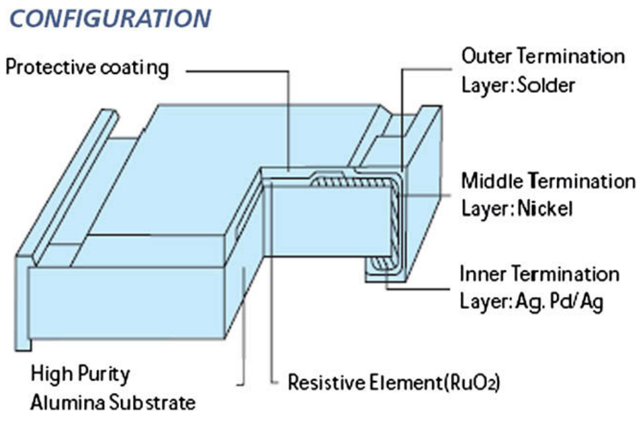

CONFIGURATION
Protective coating
Outer Termination Layer: Solder
Middle Termination Layer: Nickel
Inner Termination Layer: Ag. Pd/Ag
High Purity Alumina Substrate
Resistive Element(RuO2)

Fig. 1. Configuration of a chip resistor.

2. Each machine is allowed to process only one job at a time.
3. No preemption is allowed.
4. All jobs are independent.
5. All jobs are available being processed at time 0.
6. The setup time of each job on every machine can be ignored.
7. Machine breakdown is not considered.

Garey, Johnson, and Sethi (1976) proved that permutation flowshop scheduling problem is NP-complete. Therefore most of the research works emerged to develop effective heuristics and metaheuristics. Framinan, Gupta, and Leisten (2004) reported a review and classification of the heuristics for permutation flowshop scheduling problems. All the heuristics mentioned in this review were developed for minimizing makespan. Reza Hejazi and Saghafian (2005) presented a complete survey of flowshop-scheduling problems and contributions from 1954 to 2004. This survey concerned some exact methods, constructive heuristics, metaheuristics and evolutionary approaches. This paper is a good reference for $n/m/p/C_{\max}$. Ruiz and Maroto (2005) provided a comprehensive review and evaluation of permutation flowshop heuristics. Through reading these review articles, it is apparent that heuristics developed for PFSPs have proposed a remarkable contribution.

Heuristics are developed for some specified situations. They may not work in some unexpected situations. To involve more flexibility, metaheuristics emerge. Among the metaheuristics, genetic algorithms have attracted a lot of attention because of many convincing results. Chen, Vempati, and Aljaber (1995), Reeves (1995) and Murata, Ishibuchi, and Tanaka (1996), did the pioneering work by applying genetic algorithms in solving flowshop scheduling problems. Reeves and Yamada (1998) and Wang and Zheng (2003), Two-phase subpopulation GA by Chang, Chen, and Lin (2005) which simultaneously applies several subpopulation and assigns the weight for these subpopulation to explore the solution space uniformly; mining gene subpopulation GA by Chang, Chen, and Liu (2007) which employs a mining gene technique based on the subpopulation genetic algorithm. Chang, Hsieh, and Liu (2006) hybridized other techniques with genetic algorithms to improve the genetic searching capability. Iyer and Saxena (2004) proposed an improved genetic algorithm to solve permutation flowshop scheduling problem. It is not difficult to search more related genetic algorithm applications in the field of PFSPs. However, those applications are similar in algorithmic structures. Leaving the traditional structures behind, this work intends to improve the effectiveness and efficiency of genetic search by embedding more feedback information in the evolutionary process.

The central task of PFSPs is to arrange jobs to positions to meet one or some specified objectives. In other words, if the association between jobs and positions can be found then PFSPS can be solved immediately. However, scheduling problems belong to discrete optimization problems. Therefore there is no apparent links between jobs and positions. Although it is hopeless to find some deterministic job and position associations, it still can strive for some probabilistic job and position associations. It may have some improvement with embedding the probabilistic job and position associations.

Observing the algorithmic structures of genetic algorithms, it is interesting that each genetic algorithm would evolve many generations and it may be meaningful to analyze the job and position associations from the evolved top chromosomes. To generate artificial chromosomes, it depends on the probability of each job at a certain position. The idea is originated from Chang, Wang, and Liu (2005a) and Chang, Chen, and Fan (2008) which propose a methodology to improve GAs by mining gene structures within a set of elite chromosomes generated in previous generations. Instead of replacing the crossover operator and mutation operator due to efficiency concern, the proposed algorithm is embedded into SGA. Analyzing each job assigned to which position and record the relative frequency of each job assigned to a specified position infers a probability distribution. Then a roulette wheel selection implements according to the probability distribution to generate artificial chromosomes. Embedding artificial chromosomes in a genetic algorithm may improve the genetic search.

To verify the effectiveness and efficiency of artificial chromosomes embedded in genetic algorithm (ACEGA), two types of PFSPs are applied. One is a real-world chip resistance factory and the other is the standard benchmark instances provided by Reeves (1995). The real-world chip resistance factory has a bottleneck operation. This operation consists of three printing workstations. It is expected that applied the proposed method is able to improve the scheduling performance. The standard benchmark instances are provided with optimum solutions. Comparing with the optimum solutions shows the absolute performance of ACEGA.

3. A chip resistor scheduling problem

The major steps of chip resistor manufacturing process consist of printing layers of colloid on alumina substrates. Printing has to be processed layer by layer. Each layer is processed in a specified workstation. The processing order of printing operations can not be changed.

There are three workstations in printing including R workstation, G1 workstation, G2/GM workstation. R workstation prints resistances on the alumina substrates. This process is an imperative step to set resistance. G1 workstation prints the internal layer of collotype. G2/GM workstation prints the external layer of collotype. A sample of a chip resistance of this case factory is depicted in Fig. 2.

Jobs have to be processed in the same order through the three workstations. This can be treated as a flowshop production system. The production information can be retrieved from the shop floor directly.

The chip resistor scheduling problem can be defined as follows: if $p(i,j)$ is the processing times for job i on machine j, and a job permutation $\pi_1, \pi_2,\ldots,\pi_n$, where there are n jobs and m machines, then the completion times $C(\pi_i, j)$ is calculated as follows:

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

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

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

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

The makespan is finally defined as

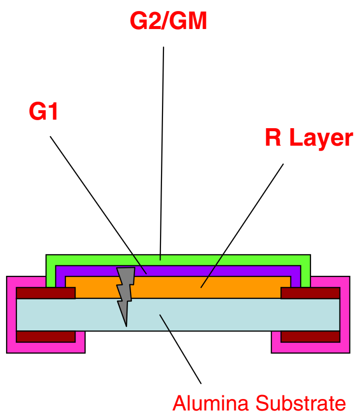


Fig. 2. A profile of chip resistor.

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

Then, the chip resistor scheduling problem is to find a permutation π^* in the set of all permutations Π *such* that

$$C_{\max}(\pi^*) \leqslant C_{\max}(\pi) \quad \forall \pi \in \Pi \tag{6}$$

The π^* is the optimal makespan for a PFSP.

The Chip resistor scheduling problem can be presented using a mixed integer programming model and the model is formulated as follows:

$$\min\left(\sum_{i=1}^{m-1}\sum_{j=1}^{n} x_{j1}p_{ij} + \sum_{j=1}^{n-1} I_{mj}\right) \tag{7}$$

subject to

$$\begin{aligned}
&\sum_{j=1}^{n} x_{jk} = 1 \quad k = 1,\ldots,n,\\
&\sum_{k=1}^{n} x_{jk} = 1 \qquad j = 1,\ldots,n,\\
&I_{ik} + \sum_{j=1}^{n} x_{j,k+1}p_{ij} + W_{i,k+1} - W_{ik} - \sum_{j=1}^{n} x_{jk}p_{i+1,j} - I_{i+1,k} = 0\\
&\quad k = 1,\ldots,n-1; \quad i = 1,\ldots,m-1,\\
&W_{i1} = 0 \quad i = 1,\ldots,m-1, \quad I_{1k} = 0 \quad k = 1,\ldots,n-1
\end{aligned} \tag{8}$$

where x_{jk} is a decision variable. It equals 1 if job j is the k_{th} job in the sequence and 0 otherwise. I_{ik} is an auxiliary variable that denotes the idle time on machine i between the processing of the jobs in the k_{th} position and $(k+1)_{\text{th}}$ position. W_{ik} is an auxiliary variable that denotes the waiting time of the job in the k_{th} position in between machine i and i+1. The first set of constraints represents that exactly only one job has to be assigned to position k for any k. The second set of constraints represents that job j has to be assigned to exactly only one job. The third set of constraints represents the relationships of decision variable x_{jk} with the physical constraints.

4. Artificial chromosome embedded in genetic algorithm

It is observed that genetic algorithms evolve generation by generation. Once the number of generations is considerably large, one genetic algorithm should ever search a huge number of solutions. Then it would be meaningful to discuss the job and position association among the searched solutions. Although it is not easy to find a deterministic job and position association, an alternative is to find a probabilistic association between jobs and positions. As long as the number of searched solutions is large enough, the frequency of each job assigned to each position can be summarized.

First, the top quality chromosomes are collected together and the frequency of each job assigned to which position is recorded in a matrix cell. The whole cells in the matrix can be filled by the same procedure. Dividing the frequency of each job by the total frequency of each position obtains the marginal probability distribution of each position. Then to assign which job to this position can be determined by a roulette wheel selection according to the marginal probability distribution of this position. The higher the probability, the higher chance of this job being assigned to this position. Through this mechanism, new chromosomes can be created. The new created chromosomes are named artificial chromosomes. It is expected that artificial chromosomes are able to find higher quality solutions. The artificial chromosomes are embedded in a simple genetic algorithm (SGA) to further improvement on solution quality. The overall procedure of embedding artificial chromosomes in a simple genetic algorithm is depicted in Fig. 3. The procedure is described in two subsections. Subsection 4.1 describes the simple genetic algorithm procedure. Subsection 4.2 describes the artificial chromosome mechanism embedded in the genetic algorithm.

4.1. Simple genetic algorithm

In this subsection, the steps to implement a simple genetic algorithm are introduced. The key steps include encoding, initial population generation, fitness function, reproduction, crossover, mutation, and stopping criterion.

4.1.1. Encoding: integer coding

Encoding relates to an appropriate chromosome representation. Effective representation perhaps reduces computational complexity. In PFSPs, jobs are processed by a series of machines in an identical order. PFSPs only concern the sequences of jobs. The sequences of machines are not of interest. That is, it only needs to show the processing order of jobs in the chromosomes. Therefore it is feasible to use integer numbers and the permutation of the integers to represent the jobs and feasible sequences respectively. Taking a 5-job case as an example, {1, 2, 3, 4, 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.

4.1.2. Initial population generation: random number generation

An initial population consists of a number of chromosomes. A chromosome represents a processing sequence for the scheduling problem. To keep diversity in the initial population, random number generation is applied to generate initial sequences.

4.1.3. Fitness function and reproduction

Because the type of this problem is a single objective problem, the objective value of each chromosome can be used as fitness directly. Then, the binary tournament selection proposed by Goldberg and Deb (1991) is employed in the reproduction operation. The criterion to reproduce better offsprings is dependent on their own fitness; the individual whose fitness is lower will be reproduced. As a result, the reproduction operation selects better chromosomes into the mating pool.

4.1.4. Crossover: two-point crossover

In the crossover step, two chromosomes are randomly selected and a random number rc is generated first. If rc is smaller than Pc, then crossover implements on this pair, else no crossover.

Syswerda (1991) proposed the concept of position-based crossover including single-point crossover and two-point crossover. Murata and Ishibuchi (1994) and Murata et al. (1996) reported that two-point crossover is effective for flowshop scheduling problems.

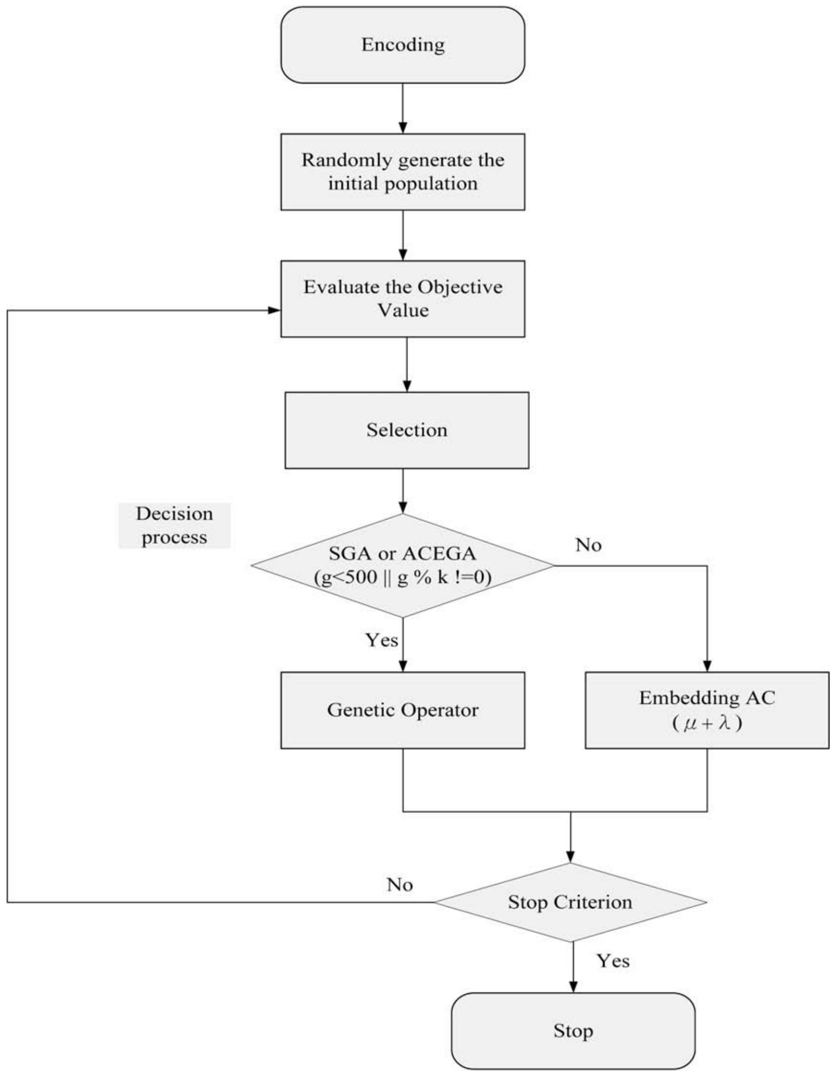

Encoding
Randomly generate the initial population
Evaluate the Objective Value
Selection
Decision process
SGA or ACEGA (g<500 || g % k !=0)
No
Yes
Genetic Operator
Embedding AC ($\mu+\lambda$)
No
Stop Criterion
Yes
Stop

Fig. 3. The framework of artificial chromosome embedded in genetic algorithm.

The steps to implement a two-point crossover are described as follows:

Select two chromosomes Parent 1 and Parent 2.

Randomly assign two cutting points, suppose the cutting points are located at i_{th} and j_{th} positions respectively. Genes beyond the cutting points in Parent 1 are directly duplicated to the offspring.

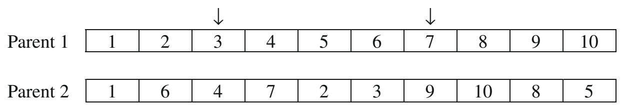
The vacant positions in the offspring are duplicated from Parent 2.

For example, two 10-job chromosomes namely Parent 1 and Parent 2 are shown in Fig. 4. Two cutting points are assigned at position 3 and position 7. Jobs before position 3 and jobs after position 7 in Parent 1 are duplicated to the offspring as Fig. 5. The sequence of vacant positions in the offspring is duplicated from Parent 2 and the precedence is still kept. Then the crossover finishes and the complete offspring is shown in Fig. 6.

4.1.5. Mutation: swap mutation

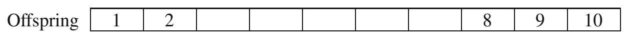
In the mutation step, two positions of a chromosome are randomly selected and a random number rm is generated first. If rm is smaller than Pm, then mutation implements on this pair, else no mutation.

Swap mutation is to randomly select two positions in a chromosome and interchange the two positions. For example, position 1 and 2 are assigned in Fig. 7. Before mutation, the sequence is 2-



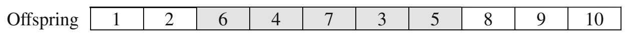
			↓				↓			
Parent 1	1	2	3	4	5	6	7	8	9	10
Parent 2	1	6	4	7	2	3	9	10	8	5

Fig. 4. Parent 1 and Parent 2.



Offspring	1	2						8	9	10

Fig. 5. Duplication of genes from Parent 1.



Offspring	1	2	6	4	7	3	5	8	9	10

Fig. 6. Duplication of genes from Parent 2.

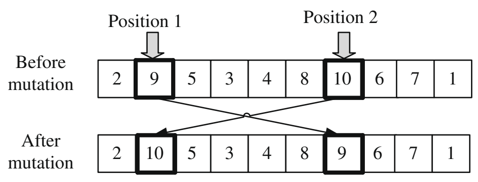


Fig. 7. Swap mutation.

9-5-3-4-8-10-6-7-1. The corresponding jobs at position 1 and 2 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.

4.1.6. Stopping criterion

Maximum number of generations (G_{max}) is specified in advance. If the number of generations is not greater than G_{max} then turn back to evaluate the fitness of the chromosomes in the new population, else the algorithm is terminated.

4.2. Artificial chromosomes

4.2.1. Association matrix

Top quality chromosomes are collected together and recorded the frequency of each job on each position. Taking a 5-job case as an example, 10 chromosomes are analyzed and the frequency of each job on each position is summarized in the association matrix in Fig. 8.

4.2.2. Job assignment

According to the association matrix, the jobs can be assigned one by one. Suppose position *i* is selected first, which job will be assigned can be made by the following steps:

(1) Select a position *i*.
(2) Find the marginal probability distribution of position *i*.
(3) Calculate the accumulated probability.
(4) Apply roulette wheel selection to select a job.
(5) Go to step (1) until all the positions are occupied.

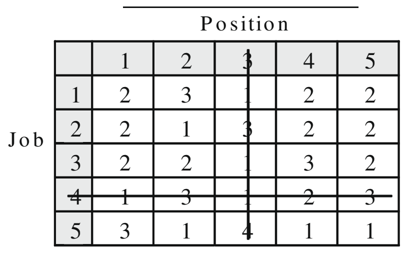
Continue the 5-job example and suppose position 3 is selected. The frequency of each job on position 3 is transferred in the "Frequency" column in Fig. 9. Summing up the frequency of each job obtains total frequency. Dividing each frequency by total frequency obtains probability of each job (Shown in "Probability" column in Fig. 9). Calculate the accumulated probability of each job (Shown in "Accumulated Probability" column in Fig. 9) and apply roulette wheel selection to select a job assigned to position 3.

Suppose job 4 is selected, and then eliminate job in row 4 and position in column 3 from the association matrix. The new matrix after elimination is shown in Fig. 10.

After assigning job 4 on position 3, the steps will be repeated until all the positions are occupied.

Jobs	Frequency	Probability	Accumulated Probability
1	1	0.10	0.10
2	3	0.30	0.40
3	1	0.10	0.50
4	1	0.10	0.60
5	4	0.40	1.00

Fig. 9. The probability and accumulated probability of each job for position 3.



Position

Job	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

Fig. 10. Eliminate job in row 4 and position in column 3 from the association matrix.

5. Numerical experiments

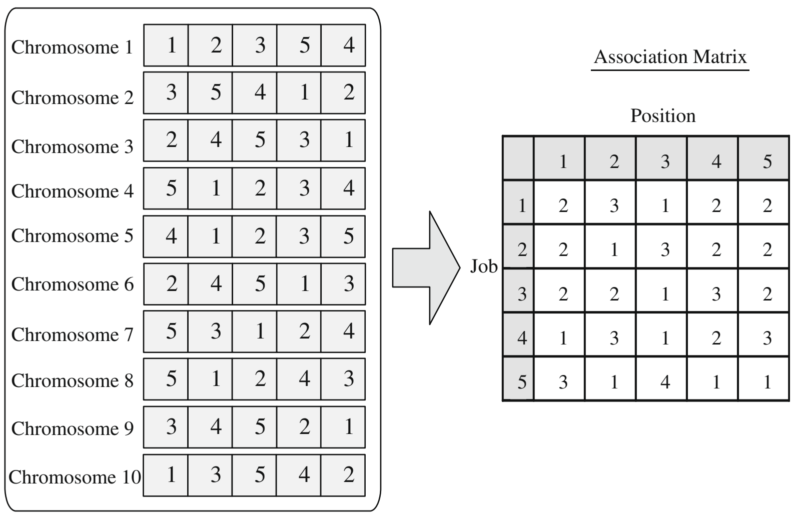
Two types of flowshop scheduling problems are considered to evaluate the proposed genetic algorithm. The first one is the case study of a chip resistor factory and the second one is from the standard benchmark instances. The first type is a real-life case. The production information is retrieved from the shop floor. The second type is the standard benchmark instances found in OR-Library.

Reeves (1995) provided 21 instances (titled rec01-rec41) as standard benchmark for flowshop scheduling problems with minimizing makespan. These instances are used to evaluate the performance of the proposed artificial chromosome embedded in genetic algorithm. The standard benchmark instances are able to be found in the OR-Library (http://www.ms.ic.ac.uk/info.html).

The numerical experiments are done for chip resistor factory and standard benchmark instances separately.

5.1. A chip resistor factory

The flowshop production system in the chip resistor factory consists of three printing machines. Production data are retrieved from the shop floor directly. The number of machines is fixed as



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

Association Matrix

Position

Job	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

Fig. 8. Summarizing the job and position association in an association matrix.

Table 1
Parameter values for SGA

Parameters	Values
Population size	100
Crossover rate	0.9
Mutation rate	0.5
$G_{\max}$	1000

Table 2
Average objective values of SGA and ACEGA in the chip resistor case

# of jobs	SGA avg. obj value	ACEGA avg. obj value
20	34,386	34,270
40	203,505	201,819
60	581,420	579,634
80	1,112,965	1,108,485
100	1,746,115	1,738,541

well as the shop floor and the number of jobs has five cases: 20, 40, 60, 80, and 100. The key parameter values for SGA were determined by conducting an design of experiment approach and the optimal setting is reported in Table 1. The population size, crossover rate, mutation rate, and maximum number of generations ($G_{\max}$) are 100, 0.9, 0.5, and 1000, respectively.

The numerical results of SGA and ACEGA are reported in Table 2. Table 2 indicates that ACEGA outperforms SGA in all the cases.

5.2. Standard benchmark instances

Standard benchmark instances obtained from the OR-Library are solved by SGA and ACEGA, respectively. The parameter values of genetic algorithm part in ACEGA are the same as the settings of SGA. There are three parameters of artificial chromosomes part of ACEGA. The parameters are listed in Table 3. The first parameter "Starting Generation" represents that artificial chromosome mechanism activates from the "Starting Generation" on. The second parameter "Interval" denotes how long the artificial chromosome mechanism collects beset quality chromosomes. The last parameter "Assignment Method" denotes the method to select a position is dependent on the random number generated.

The optimum of each instance is listed in the third column of Table 4. The average objective values of SGA and ACEGA are shown in the fourth and fifth columns, respectively. Then we calculate the difference of SGA and ACEGA with optimum for each instance in terms of error rate. The formula of error rate is shown as follows:

$$\text{Error rate} = \frac{\text{Average objective value} - \text{Optimum}}{\text{Optimum}} \times 100\% \qquad (9)$$

The error rates of ACEGA are evidently smaller than the error rates of SGA in all 21 instances. On average, there is a 1.458% difference between the performance of SGA and ACEGA. In other words, the overall error rate of ACEGA is almost one half of that of SGA. This result represents that ACEGA is very promising.

Besides the solution quality, the convergent trends of SGA and ACEGA are depicted in Fig. 11. In this figure, the data of SGA and ACEGA are denoted by "□" and "▓" respectively. The convergent trends reflect that both of the two methods converge rapidly. ACEGA converges to the final results faster than SGA after the 90th generation and the performance of ACEGA leads afterward.

Table 3
The suggested parameter settings of ACEGA

Parameters	Values
Starting generation	500
Interval	50
Assignment method	Random assign

Table 4
Average objective values of SGA and ACEGA in solving standard benchmark

Instance	(n, m)	Optimum	SGA avg. obj value	ACEGA avg. obj value	Error rate (%)	
					SGA	ACEGA
rec01	(20,5)	1247	1250	1249	0.200	0.160
rec03	(20,5)	1109	1111	1110	0.207	0.090
rec05	(20,5)	1242	1247	1245	0.370	0.242
rec07	(20,10)	1566	1584	1574	1.137	0.511
rec09	(20,10)	1537	1564	1554	1.783	1.106
rec11	(20,10)	1431	1445	1433	0.985	0.140
rec13	(20,15)	1930	1965	1946	1.824	0.829
rec15	(20,15)	1950	1986	1964	1.831	0.718
rec17	(20,15)	1902	1951	1932	2.581	1.577
rec19	(30,10)	2093	2154	2124	2.924	1.481
rec21	(30,10)	2017	2069	2050	2.578	1.636
rec23	(30,10)	2011	2076	2047	3.212	1.790
rec25	(30,15)	2513	2615	2566	4.059	2.109
rec27	(30,15)	2373	2450	2408	3.224	1.475
rec29	(30,15)	2287	2387	2336	4.368	2.143
rec31	(50,10)	3045	3215	3136	5.580	2.989
rec33	(50,10)	3114	3185	3143	2.286	0.931
rec35	(50,10)	3277	3306	3279	0.882	0.061
rec37	(75,20)	4951	5382	5205	8.701	5.130
rec39	(75,20)	5087	5436	5278	6.861	3.755
rec41	(75,20)	4960	5411	5217	9.095	5.181
Overall error rate					3.080	1.622

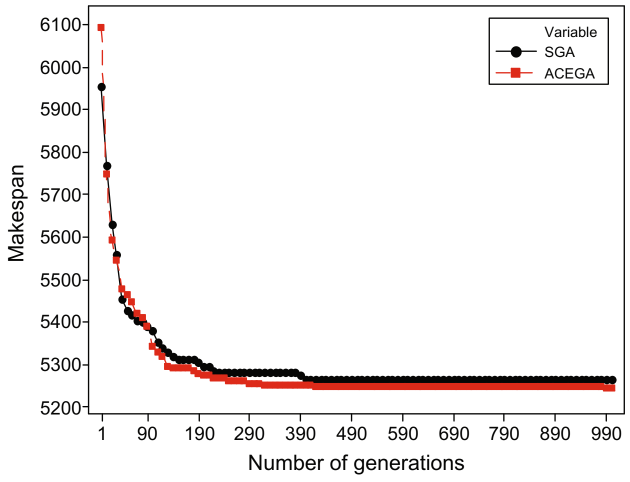


Fig. 11. Convergent trends of SGA and ACEGA.

6. Conclusion and future works

A new design, artificial chromosome, to improve the performance of genetic search is proposed in this paper and applied to a permutation flowshop scheduling problem in minimizing the make span. Extensive numerical experiments were conducted. The result indicates that artificial chromosome embedded in genetic algorithm is effective and efficiency.

Further investigation will be carried out to examine whether it is possible to generate elite chromosome through better mining algorithm. It is also suggested that different types of flowshop problem can be further tested such as the minimization of the sum of job completion times, and those with more complex requirements such as sequence dependent setup times, and to explore alternative ways of reintroducing the fabricated chromosome into the evolution process.

References

Bai, S. X., & Elhafsi, M. (1996). Transient and steady-state analysis of a manufacturing system with setup changes. *Journal of Global Optimization, 8*, 349–378.

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

Caccetta, L., & Hill, S. P. (2003). An application of branch and cut to open pit mine scheduling. *Journal of Global Optimization, 27*, 349–365.

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

Chang, P. C., Chen, S. H., & Lin, K. L. (2005). Two-phase sub population genetic algorithm for parallel machine-scheduling problem. *Expert Systems with Applications, 29*(3), 705–712.

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

Chang, P. C., Hsieh, J. C., & Liu, C. H. (2006). A case-injected genetic algorithm for single machine scheduling problems with release times. *International Journal of Production Economics, 103*, 551–564.

Chang, P. C., Wang, Y. W., & Liu, C. H. (2005a). New operators for faster convergence and better solution quality in modified genetic algorithm. *Lecture Notes in Computer Science, 3611*, 983–991.

Chen, C. L., Vempati, V. S., & Aljaber, N. (1995). An application of genetic algorithms for flow shop problems. *European Journal of Operational Research, 80*, 389–396.

Dror, M., & Mullaseril, P. A. (1996). Three stage generalized flowshop: Scheduling civil engineering projects. *Journal of Global Optimization, 9*, 321–344.

Framinan, J. M., Gupta, J. N. D., & Leisten, R. (2004). A review and classification of heuristics for permutation flow-shop scheduling with makespan objective. *Journal of the Operational Research Society, 55*, 1243–1255.

Garey, M. R., Johnson, D. S., & Sethi, R. (1976). The complexity of flowshop and jobshop scheduling. *Mathematics of Operations Research, 1*, 117–129.

Goldberg, D. E., & Deb, K. (1991). A comparison of selection schemes used in genetic algorithms. In Rawlins, G. J. E. (Ed.), *Foundations of genetic algorithms* (pp. 69–93).

Iyer, S. K., & Saxena, B. (2004). Improved genetic algorithm for the permutation flowshop scheduling problem. *Computer and Operations Research, 31*, 593–606.

Murata, T., & Ishibuchi, H. (1994). Performance evolution of genetic algorithms for flowshop scheduling problems. *Proceedings of first IEEE international conference on evolutionary computation*, 812–817.

Murata, T., Ishibuchi, H., & Tanaka, H. (1996). Genetic algorithms for flowshop scheduling problems. *Computers and Industrial Engineering, 30*, 1061–1071.

Reeves, C. R. (1995). A genetic algorithm for flowshop sequencing. *Computer and Operations Research, 22*, 5–13.

Reeves, C. R., & Yamada, T. (1998). Genetic algorithms, path relinking, and the flowshop sequencing problem. *Evolutionary Computation, 6*, 45–60.

Reza Hejazi, S., & Saghafian, S. (2005). Flowshop-scheduling problems with makespan criterion: A review. *International Journal of Production Research, 43*, 2895–2929.

Ruiz, R., & Maroto, C. (2005). A comprehensive review and evaluation of permutation flowshop heuristics. *European Journal of Operational Research, 165*, 479–494.

Syswerda, G. (1991). Scheduling optimization using genetic algorithm. *Handbook of Genetic Algorithm*, 332–349.

Wang, L., & Zheng, D. Z. (2003). An effective hybrid heuristic for flow shop scheduling. *The International Journal of Advanced Manufacturing Technology, 21*, 38–44.