ELSEVIER

Expert Systems with Applications 29 (2005) 705–712

Expert Systems with Applications

www.elsevier.com/locate/eswa

Two-phase sub population genetic algorithm for parallel machine-scheduling problem

Pei-Chann Chang*, Shih-Hsin Chen, Kun-Lin Lin

Department of Industrial Engineering and Management, Yuan-Ze University, 135 Yuan-Tung Road, Taoyuan 32026, Taiwan, ROC

Abstract

This paper introduces a two-phase sub population genetic algorithm to solve the parallel machine-scheduling problem. In the first phase, the population will be decomposed into many sub-populations and each sub-population is designed for a scalar multi-objective. Sub-population is a new approach for solving multi-objective problems by fixing each sub-population for a pre-determined criterion. In the second phase, non-dominant solutions will be combined after the first phase and all sub-population will be unified as one big population. Not only the algorithm merges sub-populations but the external memory of Pareto solution is also merged and updated. Then, one unified population with each chromosome search for a specific weighted objective during the next evolution process. The two phase sub-population genetic algorithm is applied to solve the parallel machine-scheduling problems in testing of the efficiency and efficacy. Experimental results are reported and the superiority of this approach is discussed.

Keywords: Scheduling problem; Genetic algorithm; Multi-objective optimization; Evolution strategy

1. Introduction

The literature of parallel machine-scheduling problems has been extensively reviewed by Cheng and Sin (1990). Garey and Johnson (1979) showed that scheduling jobs on two identical machines to minimize the makespan is NP-hard (NP is the abbreviation of Non-Polynomial). As identified by Brucker (1998) when the number of machine is greater than two, the problem is even strong NP-hard. Therefore, the parallel machine-scheduling problem presents a great challenge to the industrial practitioners and academic researchers. As a result, efficient heuristic algorithm should be developed in order to deal with practical industrial scheduling problems especially in drilling operation-scheduling problems of Printed Circuit Board (PCB) industries as presented by Hsieh, Chang, and Hsu (2003).

Owing to the development in genetic algorithm, it provides a new method and new direction for scheduling researchers to apply this new tool. Successful application examples can be found in Carlos and Peter (1995), Chang, Hsieh, and Lin (2002), Chang, Hsieh, and Wang (2003), Lo and Bavarian (1992), Neppali, Chen, and Gupta (1996), Sridhar and Rajendran (1996), and Wang (2003, 2005). However, in practical application the goal in PCB industries is always multi-objective which includes makespan, due-date and flowtime. Recent development in Evolutionary Multi-objective Optimization provides interesting results as discussed by Deb, Amrit Pratap, and Meyarivan (2000) and Zitzler, Laumanns, and Bleuler (2004). In that, different EMO algorithms are proposed and efficiency and solution quality are greatly improved.

Inspired by these pioneer works as discussed above, this research proposes a two phase sub-population genetic algorithm to solve the parallel machine-scheduling problem. In the first phase, the population will be decomposed into many sub-populations and each sub-population is designed for a scalar multi-objective. Sub-population is a new approach for solving multi-objective problems by fixing each sub-population for pre-determined criteria. In the second phase, non-dominant solutions will be combined after the final evolutions. One unified population by combining these sub-populations will be applied for regular evolution.

The rest of the paper is organized as follows: Section 2 gives literature review. Section 3 introduces the TPSPGA

* Corresponding author. Tel.: +886 3 4352654; fax: +886 3 4559378.
E-mail address: iepchang@saturn.yzu.edu.tw (P.-C. Chang).

0957-4174/$ - see front matter © 2005 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2005.04.033

algorithm. Then experimental results are given in Section 4. Finally, the conclusion is discussed and performance of the algorithm is evaluated.

2. Literature review

The genetic algorithm has been widely discussed. The following related works present the efforts of multi-objective genetic algorithm and its application on scheduling problem.

Holland (1975) proposed the Genetic Algorithm (GA) that imitates the natural evolution progress, including the selection, crossover, and mutation. The procedure produces better offspring in accommodating to the environment. Schaffer (1985) proposed VEGA (Vector Evaluated Genetic Algorithm) to solve the Pareto-optimal solution of multi-objective problem. The VEGA is the first method modifying the GA to solve multi-objective problems. It selects better chromosomes from separate sub-mating pools so that the selected chromosomes satisfy different objectives. However, the drawbacks of the algorithm are: (1) the algorithm ca not guarantee that all the solutions are Pareto-optimal solution; and (2) the algorithm cannot keep the diversity of the solutions during the evolving process.

Muruta and Ishibuchi (1996) employed the structure of genetic algorithm in searching the multi-objective problem, and the algorithm is named MOGA (Multi-Objective Genetic Algorithm). One characteristic of MOGA is using the dynamic weighting to transform the multiple objectives into single objective, which randomly assigns different weight value to different objectives. The algorithm also applies the elite preserving strategy that randomly selects chromosomes from Pareto set. This technique prevents from sinking into local optimal. Muruta, Ishibuchi, and Tanaka (1996) found the MOGA was superior to VEGA or GA on multi-objective problem of flow shop scheduling problem. NSGA2 (Non-dominated Sorting Genetic Algorithm-II) was proposed by Deb et al. (2000), where the Elitism strategy was adopted. Besides, in order to keep the solution diversity, the algorithm also provided a crowding distance to measure the density of individuals in solution space.

Coello Coello et al. (2001) proposed micro-GA, which refers to small-population genetic algorithm with reinitialization. They found that micro-GA is able to converge to better solution although there are few individuals in each population. The more individuals in a population, algorithm needs more computational time. Therefore, it saved the computational effort and increases the efficiency. The performance of GA that accompanied vector evaluated approach (from the concept of Schaffer, 1985) and weighted criteria approach (linear combination of objectives) on multi-objective scheduling problem was evaluated by Neppali et al. (1996). The evaluation result showed that the presented vector evaluated approach was better than weighted criteria approach. Muruta and Lshibuchi (1994) utilized MOGA to solve the flowline scheduling problem, which emphasizes on random weight assignment and Elitism. The research pointed out that the Elitism was able to find out the near Pareto set more efficiently and fast.

Funda and Ulusoy (1999) employed the GA and considered the total weighted earliness and tardiness on the mutliobjective scheduling problem of parallel machine. Their suggestion is that if the local search was included in the GA, the performance may become better. Gupta, Neppalli, and Werner (2001) discussed the multi-objective scheduling problem of parallel machines, which attempts to minimize the make span under the consideration of minimizing the flow time. They proposed Two-Machine Optimization Procedure, Longest Processing Time Procedure, Multi-fit Procedure, and Hierarchical Criteria Algorithm. The experiment result showed the combination of Hierarchical Criteria Algorithm and Multi-fit Procedure was more efficient when compared to others combinations.

Motivated by the literature discussed above, this research introduces a two-phase sub population genetic algorithm to solve the parallel machine-scheduling problem. Detailed procedures of TPSPGA are presented in Section 3.

3. Two phase sub-population genetic algorithm (TPSPGA)

According to Simoes and Costa (2002), the loss of diversity may mean pre-mature of evolution algorithm. In order to prevent the searching procedure from being trapped into local optimality, this research proposes a two phase sub-population genetic algorithm (TPSPGA) to solve the parallel machine-scheduling problems. The basic idea of TPSPGA is to decompose the population into several sub-populations, which are assigned different weights by scalarizing multiple objectives into single objective. Each sub-population is just like a squad team with a pre-assigned goal and hopefully they will be marching in the right direction to reach the high peak of the landscape in terms of fitness performance. By the effort, every sub-population concentrates on specific exploring space and thus the diversity of the population can be kept among these sub-populations. After certain evolutions of the searching process, to further improve the solution quality, all the sub-population will be reunified and each individual chromosome will be reassigned a scalarized objective to further expand the searching process until the final near-optimal solution is found.

As discussed above, there are three main characteristics of the TPSPGA method: (1) numerous small sub-populations are designed to explore the solution space; (2) the multiple objectives are scalarized into single objective for each sub-population; and (3) two phase implementation is applied to deal with the gradually narrowing down of the searching space. The framework of TPSPGA is illustrated in

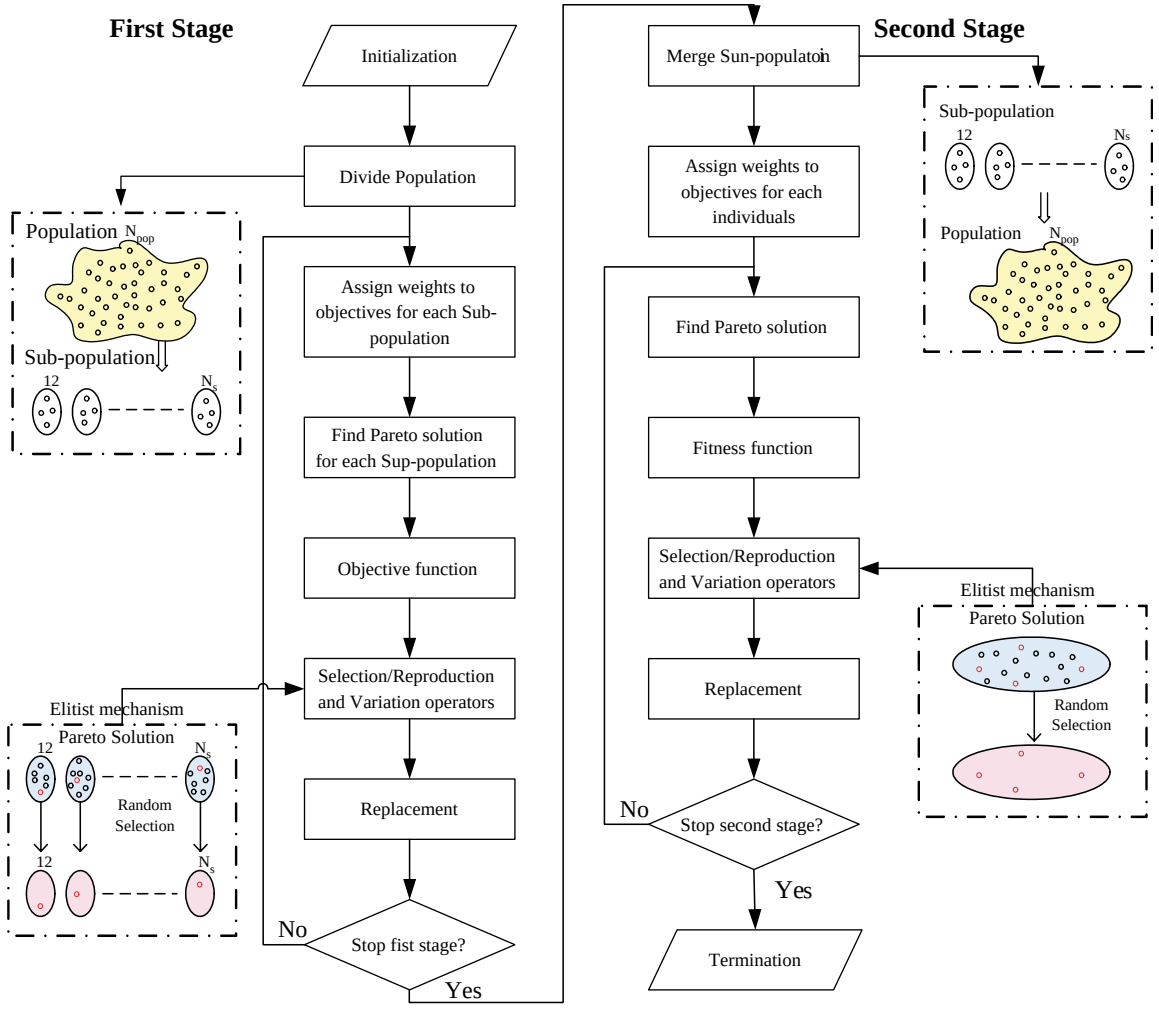

Fig. 1. The framework of TPSPGA.

Fig. 1 and the detail procedure will be explained in Sections 3.1–3.5.

As illustrated in Fig. 1, there are two stages in TPSPGA: at the first stage, there are two objectives that need to be minimized. Suppose the number of individuals of original population is N. Because the algorithm divides the population into n small sub-populations, each sub-population contains N/n individuals. Furthermore, every sub-population is assigned with two different weights for these two objectives. For example, all chromosomes in sub-population n is assigned with the weight value (W_{n1}, W_{n2}), W_{n1} and W_{n2} stand for the weight of the first and second objective, respectively. Therefore, we scalarize the two objectives into a single one by a linear combination of these two objectives. Consequently, these sub-populations will search for different solution areas as shown in Fig. 2.

In the second stage, it is possible that TPSPGA may miss some important searching spaces. To further improve the solution quality of TPSPGA, we will restore these sub-populations into one big population in the beginning. Then, we randomly specify a linear combination of weight of these two objectives for each individual in the whole population as shown in Fig. 3.

The detailed procedures of each stage are explained in the following section.

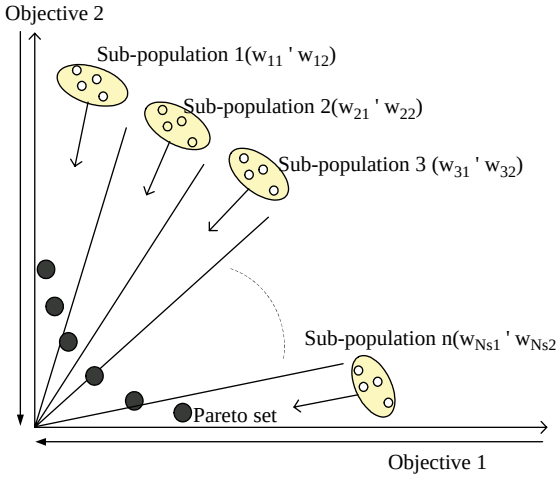

Fig. 2. TPSPGA at the first phase.

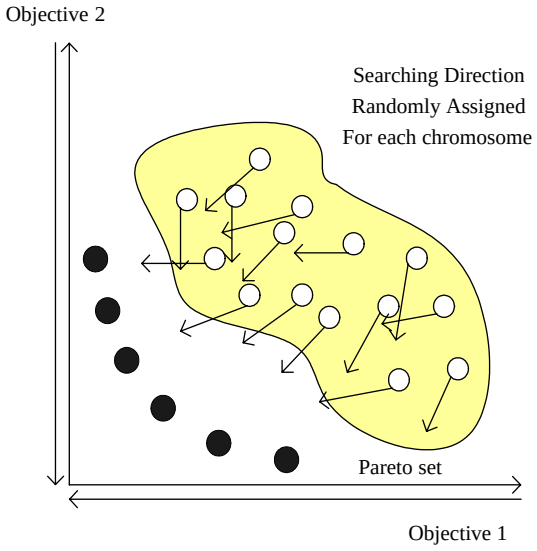

Fig. 3. TPSPGA searching directions in phase two.

3.1. Parameters of the algorithm

Parameters of the algorithm are listed as follows:

N	number of chromosomes.
ns	number of sub-populations.
n	number of individuals in each sub-population.
Iteration1	number of iterations in phase one.
Iteration2	number of iterations in phase two.

3.2. General scheme of TPSPGA

The detailed procedure of TPSPGA is explained as follows:

Algorithm 1. TPSPGA()

```
1. Initialize()
2. DividePopulation()
3. AssignWeightToEachObjectives()
4. counter ← 0
5. while counter < Iteration1 do
6.    for i = 1 to ns do
7.       FindParetoi()
8.       Fitnessi()
9.       Elitismi()
10.      Selectioni()
11.      Crossoveri()
12.      Mutationi()
13.      Replacementi()
14.   end for
15.   counter ← counter + 1
16. end while
17. counter ← 0
18. Merge()
19. while counter < Iteration2 do
20.    FindPareto()
21.    Fitness()
22.    Elitism()
23.    Selection()
24.    Crossover()
25.    Mutation()
26.    Replacement()
27.    counter ← counter + 1
28. end while
```

3.3. The first phase

The procedure *initialization* is used to generate the chromosomes of a population, which is according to the population size defined by the user. The procedure *Divide Population* is to divide the original population into *ns* sub-populations.

At the procedure *Assign Weight To Each Objectives*, each sub-population is assigned different weights and each individual in the same sub-population share the same weight. The scalarized weight of each sub-population is defined as below

$$(W_{n1}, W_{n2}) = \left(\frac{1}{N_s + 1} \times n,\ 1 - \frac{1}{N_s + 1} \times n\right) \tag{1}$$

where n is the nth sub-population.

After the weight value assignment, the objective value of sub-population n is defined as follows

$$f(x) = W_{n1} \cdot f(x_1) + W_{n2} \cdot f(x_2) \tag{2}$$

where

$f(x_1)$ the first objective function
$f(x_2)$ the second objective function

To calculate the fitness value, we have to calculate the objective value of the sub-population and to normalize the objective value. In this study, we concern two scheduling objectives, including the makespan and tardiness of all jobs. The metrics are as follows

$$Z_{\mathrm{TT}} = \sum_{i=1}^{n} T_i; \quad T_i = \max\{C_i - d_i, 0\} \tag{3}$$

$$Z_{\mathrm{TC}} = \max\{F_1, F_2, \ldots, F_n\} \tag{4}$$

where

Z_{TT} total tardiness time
Z_{TC} total completion time or makespan

Because the scale of each objective is different, in order to evaluate the solution quality, this research will normalize

these objective values between 0 and 1. The equations are listed as follows:

$$f(x_1) = \frac{X_{t1}^i - f_{t1}^W}{f_{t1}^B - f_{t1}^W} \quad 0 \le f(x_1) \le 1 \tag{5}$$

$$f(x_2) = \frac{X_{t2}^i - f_{t2}^W}{f_{t2}^B - f_{t2}^W} \quad 0 \le f(x_2) \le 1 \tag{6}$$

The Elitism strategy adopted at the first stage is randomly selecting a number of individuals from non-dominated set into mating pool; therefore we can pick these individuals while in the crossover procedure. The Elitism strategy of each sub-population is independent.

The binary tournament selection is employed in the selection operation of the first phase. The reason why we do not employ the roulette wheel in the first phase is that there is few individuals in each sub-population so that we apply the binary tournament may save time than roulette wheel in this phase. The smaller objective value of each chromosome has a better chance to be selected. Every sub-population works independently so that they will not influence each other.

There are numerous crossover and mutation methods. We utilize the two-point crossover and moving position mutation for the Crossover procedure and the Mutation procedure, respectively, because Muruta and Ishibuchi (1994) found both of them were the best approaches for these two objectives.

3.4. The second phase

The unique characteristic of the second phase is that the second phase merges these sub-populations into one big population. In other words, the size of the new population is equal to the original population size. Not only the algorithm merges sub-populations but the external memory of Pareto solution is also merged and updated.

There are two other differences compared to the first phase. The first is the vector of weight value, which is randomly assigned to each individual population. For example, we have a weight vector (W_{n1},W_{n2}) for the individual n and W_{n1} is the weight value for first objective and W_{n2} for the second objective, respectively. We generate the random value for W_{n1}. Since $W_{n1}+W_{n2}$ should equal to one, W_{n2} is equivalent to $1-W_{n1}$ or 1-random value. Therefore, the assignment of weight value for individual population n can be written as the following:

$$\begin{aligned}(W_{n1}, W_{n2}) &= (W_{n1}, 1 - W_{n2}) \\ &= (\text{random value},\ 1\text{-random value})\end{aligned} \tag{7}$$

The other difference is that the selection strategy employs roulette wheel selection, which is proposed by Goldberg (1989). The selection method in the first phase applies the binary tournament selection. Other than the above differences, the rest of the procedures are the same as in phase one. Fig. 2 shows all individual population approaching the optimal Pareto set in different directions.

3.5. Performance measure

There are various metrics, which are applied to compare the non-dominated sets of evolution algorithms. Hansen and Jaszkiewicz (1998) defined the performance relations, which are weak outperformance, strong outperformance, and complete outperformance. They express the relationship between two sets of internally non-dominated objective vectors. Knowles and Corne (2002) compared several metrics based on the outperformance relations. This research selects the generational distance (GD) to measure the superiority of the non-dominated set and the method is developed by Van Veldhuizen (1999). It is because the method is compatible with the strong outperformance and complete outperformance, non-inducing, non-cardinality, and has relatively cheap calculation effort according to Knowles and Corne (2002). The equation of GD is defined as follows:

$$\text{GD} = \frac{\sqrt{\sum_{i=1}^{n} d_i^2}}{n} \tag{8}$$

where n is the number of vectors in the approximation set and d_i is the distance in the objective space between vector i and the nearest member of Z^* (the reference solution set). The lower value of GD, the better of solution quality we have.

4. Experimental results

The experiment to be conducted in this paper is the scheduling problem of drilling operation in a printed circuit board factory located in Chung-Li, Taiwan, ROC. Numerical data of 30-job 10-machine problem including job information and machine flow are presented in Table 1 and Fig. 4.

This factory has developed a simple computerized scheduling system incorporating simple heuristics such as Early Due Date (EDD) or Shortest Processing Times (SPT) while the solution quality is poor and it not flexible enough

Table 1
Numerical data of 30-job problem

Job number	Processing time (mins)	Due date (mins)
1	1020	1920
2	816	1920
3	770	1440
4	1500	1440
5	1000	1440
⋮	⋮	⋮
30	384	960

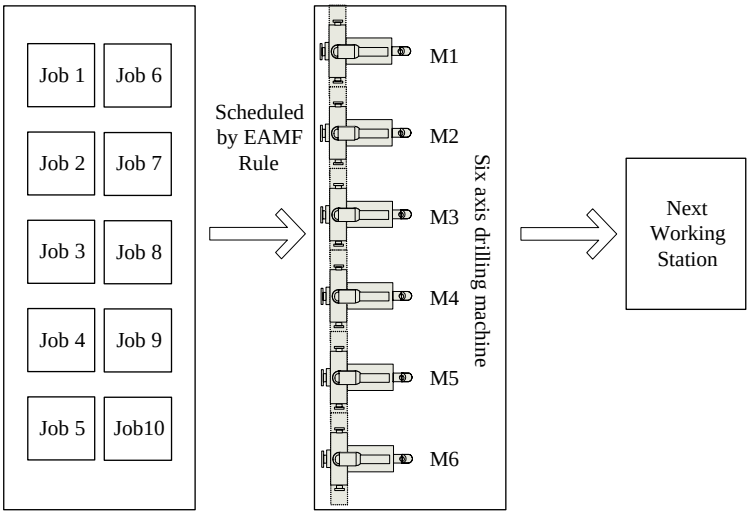

Fig. 4. Machine flow.

to react to the dynamic environment, i.e. the major incentive for us to develop a more efficient and effective system such as TPSPGA to improve the system.

In this section, TPSPGA is compared to MOGA and NSGA II with three test problem sets. They are 30 jobs and 10 machines, 50 jobs and 15 machines, and 65 jobs and 18 machines.

The parameters' setting of TPSPGA is as following:

N	300
ns	30
n	10
Iteration1	400
Iteration2	600
Crossover rate	0.9
Mutation rate	0.5
Elitist	top 20% of population

Because there are 300 chromosomes in a population and we are going to separate it into 30 sub-populations, there will be 10 chromosomes in each sub-population. Furthermore, we apply the Elitism preserving 20% of a population. Consequently, there will be two better individuals that are kept in each sub-population at the first phase and 60 better ones are stored into external memory at the second phase. The following sections are the experimental result of the three test cases.

4.1. The 30 jobs and 10 machines

Fig. 5 shows that the performance of TPSPGA is superior to that of NSGA2 and NSGA2 is better than MOGA. From the figure, the solutions of TPSPGA dominate all solutions in NSGA2 and MOGA. Table 2 represents the GD values of each method and we repeat the experiment for five times.

Table 2
The GD values of the 30 jobs and 10 machines problem

	TPSPGA	NSGA2	MOGA
Seed 1	73.50	445.30	687.46
Seed 2	68.38	468.18	413.20
Seed 3	77.46	529.26	426.86
Seed 4	85.59	459.16	400.30
Seed 5	360.21	437.21	507.90
Average	133.03	467.82	487.14

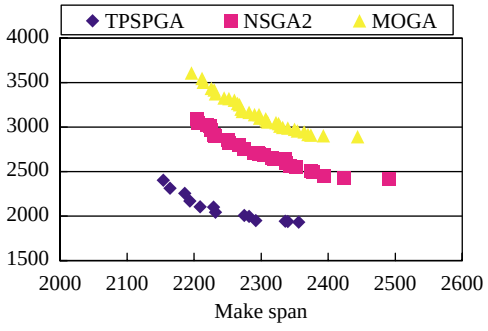

Fig. 5. The comparison of the 30 jobs and 10 machines problem.

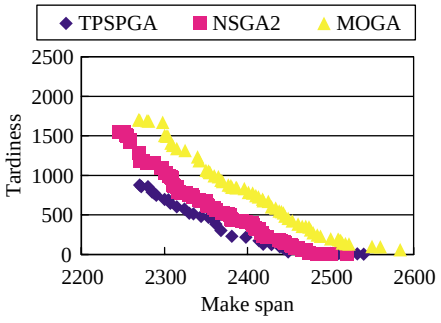

Fig. 6. Comparisons of the 50 jobs and 15 machines.

Table 3
GD values of the 50 jobs and 15 machines problem

	TPSPGA	NSGA2	MOGA
Seed 1	12.88	42.28	110.57
Seed 2	23.79	40.07	156.22
Seed 3	43.93	69.09	73.87
Seed 4	17.44	9.67	88.53
Seed 5	39.20	41.62	60.60
Average	27.45	40.55	97.96

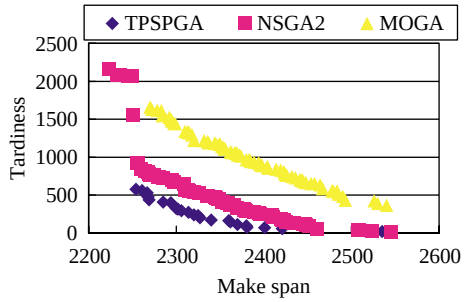

Fig. 7. Comparisons of the 65 jobs and 18 machines.

The GD values show that the TPSPGA is better than the other two in the case.

4.2. The 50 jobs and 15 machines

Fig. 6 shows that the TPSPGA is superior to NSGA2 and NSGA2 is better than MOGA. Table 3 represents the GD values with five times repeated experiment, illustrating TPSPGA is better than NSGA2 and MOGA except the seed number 4.

4.3. The 65 jobs and 18 machines

Fig. 7 shows that the TPSPGA is superior to NSGA2 and NSGA2 is better than MOGA. However, NSGA2 indeed has better diversity in the case. Table 4 represents the GD values with five times repeated experiment. In this case, the TPSPGA is also better than the other two except the seed number 2.

TPSPGA has been compared with the NSGA2 and MOGA on three test problems and each problem has five replicates. From the computational result, performances of TPSPGA are better than those of NSGA2 and MOGA according to the GD value. For example, TPSPGA is able to dominate all solutions of NSGA 2 and MOGA shown in Fig. 3 in the first test problem. GD values calculated for these three algorithms also prove that TPSPGA has better performance. Thus, TPSPGA completely outperforms NSGA2 and MOGA. As far as the second and third test problem, there is only one case that NSGA2 is superior to TPSPGA. Therefore, TPSPGA has better performances in average case.

Table 4
GD values of the 65 jobs and 18 machines problem

	TPSPGA	NSGA2	MOGA
Seed 1	25.34	77.79	172.60
Seed 2	55.40	26.54	130.54
Seed 3	23.73	40.85	121.34
Seed 4	70.30	96.68	105.94
Seed 5	18.90	59.66	144.75
Average	38.73	60.30	135.03

5. Discussion and conclusions

TPSPGA is developed in this research for solving the multi-objective scheduling problem. The algorithm is divided into two phases. The first phase applies sub-populations, which concentrates on specific search space and prevents all individuals from being converged into a local optimal. Then, in order to explore solution space ignored or missed in the first phase, sub-population is regrouped as a single big population. Each individual chromosome in this big population of the second phase is randomly assigned a weight value to explore for more solution spaces.

This research compares the solution quality among TPSPGA, NSGA2, and MOGA using three different test experiments. The results show that TPSPGA is not only able to find Pareto-optimal solution but also has better performance than the other two in average performance. Therefore, it is an inspiration and encouragement for researchers who are interested in the area. In the future, TPSPGA can be further extended to three objectives or multi-dimensional continuous problems. The concept of sub-population can be further embedded in local search procedure to improve the solution quality of the algorithm.

References

Brucker, P. (1998). *Scheduling algorithm*. Berlin: Springer.

Carlos, M. F., & Peter, J. F. (1995). An overview of evolutionary algorithms in multiobjective optimization. *Evolutionary Computation*, *3*(1), 1–16.

Chang, P. C., Hsieh, J. C., & Lin, S. G. (2002). The development of gradual priority weighting approach for the multi-objective flow shop scheduling problem. *International Journal of Production Economics*, *79*(3), 171–181.

Chang, P. C., Hsieh, J. C., & Wang, Y. W. (2003). Genetic algorithms applied in BOPP film scheduling problems. *Applied Soft Computing*, *3*, 139–148.

Cheng, T., & Sin, C. (1990). A state-of-the-art review of parallel-machine scheduling research. *European Journal of Operation Research*, *47*, 271–292.

Coello Coello, C. A., & Toscano Pulido, Gregorio (2001). A micro-genetic algorithm for multiobjective optimization. *First international conference on evolutionary multi-criterion optimization* (pp. 126–140).

Deb, K., Amrit Pratap, S. A., & Meyarivan, T. (2000). A fast and elitist multi objective genetic algorithm-NSGA-II. *Proceedings of the parallel problem solving from nature VI conference* (pp. 849–858).

Funda, S. S., & Ulusoy, G. (1999). Parallel machine scheduling with earliness and tardiness penalties. *Computers and Operations Research*, *26*(8), 773–787.

Garey, M. R., & Johnson, D. S. (1979). *Computers and intractability: A guide to the theory of NP-completeness*. San Francisco, CA: New York Freeman.

Goldberg, D. E. (1989). *Genetic algorithms in search, optimization and machine learning*. Reading, MA: Addison Wesley.

Gupta, J. N. D., Neppalli, V. R., & Werner, F. (2001). Minimizing makespan subject to minimum flowtime on two identical parallel machines. *Computers & Operations Research*, *28*, 705–717.

Hansen, M. P., & Jaszkiewicz, A., (1988). Evaluating the quality of approximations to the non-dominated set. *Technical report IMM-REP-1998-7*, Technical University of Denmark.

Holland, J. H. (1975). *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control and artificial intelligence* (4th ed.). Ann Arbor, MI: The University of Michigan in press.

Hsieh, J. C., Chang, P. C., & Hsu, L. C. (2003). Scheduling of drilling operations in printed-circuit-board factory. *Computers and Industrial Engineering*, *44*(3), 461–473.

Knowles, J. D., & Corne, D. W. (2002). On metrics for comparing non dominated sets. *In Proceedings of the 2002 congress on evolutionary computation conference (CEC02)*. 711–716 New York: IEEE Press.

Lo, Z. P., & Bavarian, B. (1992). Optimization of job scheduling on parallel machines by simulated annealing algorithms. *Expert Systems with Application*, *4*(3), 323–328.

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

Muruta, T., & Ishibuchi, H. (1996). MOGA: Multi-objective genetic algorithm. *Proceedings of second IEEE international conference on evolutionary computation* (pp. 170–175).

Muruta, T., Ishibuchi, H., & Tanaka, H. (1996). Genetic algorithm for flowshop scheduling problem. *International Journal of Computers and Industrial Engineering*, *30*, 1061–1071.

Neppali, V. R., Chen, C. L., & Gupta, J. N. D. (1996). Genetic algorithms for the two-stage bicriteria flowshop problem. *European Journal of Operational Research*, *95*, 356–373.

Schaffer, J. D. (1985). Multiple objective optimization with vector evaluated genetic algorithms. *Proceedings of first international conference on genetic algorithms* (pp. 93–100).

Simoes, A., & Costa, E. (2002). Parametric study to enhance genetic algorithm's performance when using transformation. *In Proceedings of the genetic algorithm and evolution computation conference (GECCO'02)*. New york: Morgan Kaufmann Publishers (pp. 9–13).

Sridhar, J., & Rajendran, C. (1996). Scheduling in flowshop and cellular manufacturing systems with multiple objectives—a genetic algorithm approach. *Production Planning & Control*, *7*, 374–382.

Van Veldhuizen, D. A., & David A., (1999). Multiobjective evolutionary algorithm: Classification, analyses, and new innovations. PhD Thesis, Department of Electrical and Computer Engineering, Graduate School of Engineering, Force Institute of Technology, Wright-Patterson AFB, OH.

Wang, Y. Z. (2003). Using genetic algorithm methods to solve course scheduling problems. *Expert Systems with Applications*, *25*(1), 39–50.

Wang, Y. Z. (2005). A GA-based methodology to determine an optimal curriculum for schools. *Expert Systems with Applications*, *28*(1), 163–174.

Zitzler, E., Laumanns, M., & Bleuler, S. (2004). A tutorial on evolutionary multiobjective optimization. *Proceedings of the workshop on multiple objective metaheuristics*.