Applied Soft Computing 9 (2009) 173–181

Contents lists available at ScienceDirect

Applied Soft Computing

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

The development of a sub-population genetic algorithm II (SPGA II) for multi-objective combinatorial problems

Pei-Chann Chang *, Shih-Hsin Chen

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

ARTICLE INFO

Article history:
Received 13 September 2006
Received in revised form 29 March 2008
Accepted 1 April 2008
Available online 8 April 2008

Keywords:
Genetic algorithm
Parallel scheduling problems
Multidimensional knapsack problem
Multi-objective optimization

ABSTRACT

Previous research has shown that sub-population genetic algorithm is effective in solving the multi-objective combinatorial problems. Based on these pioneering efforts, this paper extends the SPGA algorithm with a global Pareto archive technique and a two-stage approach to solve the multi-objective problems. In the first stage, the areas next to the two single objectives are searched and solutions explored around these two extreme areas are reserved in the global archive for later evolutions. Then, in the second stage, larger searching areas except the middle area are further extended to explore the solution space in finding the near-optimal frontiers. Through extensive experimental results, SPGA II does outperform SPGA, NSGA II, and SPEA 2 in the parallel scheduling problems and knapsack problems; it shows that the approach improves the sub-population genetic algorithm significantly. It may be of interests for researchers in solving multi-objective combinatorial problems.

1. Introduction

Combinatorial problems occur in many practical applications and the approaches to these problems can be classified as either exact or approximate algorithms. Exact algorithms are guaranteed to find an optimal solution by systematically searching the solution space. However, due to the NP-completeness of many combinatorial optimization problems, the time needed to solve them may grow exponentially in the worst case. Moreover, if the multiple objectives of the combinatorial problems are considered, they become even more complex. To practically solve these problems, one has to be satisfied with finding good, approximately optimal solutions in reasonable, that is, polynomial time.

Recently, researchers have developed meta-heuristics to solve the combinatorial problems with practical sizes. New development in evolutionary multi-objective optimization provides interesting results as discussed by Deb et al. [1] and Zitzler et al. [2]. Moreover, there are some researchers who propose their sub-population-like approaches, such as segregative genetic algorithms [3], multi-sexual genetic algorithm [4], multi-population genetic algorithm [5], hierarchical fair competition model [6], MO particle swam optimization [7,8], and sub-population GA [9], MOTGA [24], MOEA/D [10], MGSPGA [11], and multi-objective immune algorithm [12].

The new algorithm, i.e., SPGA II, developed in this research extends the result in our previous research [9] to solve the combinatorial problems. According to previous researches of these sub-population algorithms such as segregative genetic algorithms and multi-sexual genetic algorithm, they create a chance for these sub-populations to be able to communicate with each other occasionally thus the solution quality can be further improved. As a result, the major contribution of SPGA II is to develop a mechanism to exchange information within these sub-populations. The mechanism of SPGA II is to apply the Pareto set generated from these sub-populations and to save these Pareto set as a global archive. Therefore, once a sub-population reaches a better non-dominated solution, other sub-populations are able to apply them directly within their searching areas. Then, these Pareto set in the global archive will guide all individuals in the same population searching toward the true Pareto front.

There is a second characteristic that distinguishes SPGA II from SPGA. The SPGA II applies a two-stage approach, which attempts to derive a better convergent, and a diverse effect. The first stage will apply only few sub-populations and focus on exploring solutions near each individual objective function. The solutions derived from Stage I will be regenerated for Stage II as initial solutions and this process will encourage the information cross exchange within each sub-population. At the second stage, SPGA II explores the solution space more extensively. The procedure is identical to that of SPGA. However, to be more efficiently locating the searching areas, some sub-populations located near the middle area (like the weight vector [0.5, 0.5] for these two objective functions) are ignored.

* Corresponding author. Tel.: +886 936101320; fax: +886 34638884.
E-mail address: iepchang@saturn.yzu.edu.tw (P.-C. Chang).

1568-4946/$ – see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.asoc.2008.04.002

Consequently, the advantage of this modified procedure can create convergent solutions in the first stage and then searching for more diverse areas to find the frontiers with better solution quality in the second stage.

The rest of the research is organized as follows: Section 2 introduces the combinatorial problems. Section 3 explains detail procedures of SPGA II. Section 4 is the experimental results for the bi-criteria parallel scheduling problems and knapsack problems. Finally, the conclusions are made and the future researches are provided.

2. Combinatorial optimization problems

Combinatorial optimization problems are characterized by their well-structured problem definition as well as by their huge number of solution spaces in practical application areas. Especially in areas like routing, task allocation, or scheduling such kinds of problems often occur. Approaches by utilizing classical methods of operations research (OR) often fail due to the exponential growth of computational times. Therefore, in practice, heuristics are commonly used even if they are unable to guarantee an optimal solution.

Heuristic techniques that mimic natural processes developed over the last thirty years have produced 'good' results in reasonable short runs for this class of optimization problems. In addition heuristics are much more flexible regarding modifications in the problem description when compared to classical OR methods, and so they are often superior in their results. Heuristics such as genetic algorithms (GAs) attempt to imitate the biological evolution of a species in order to achieve an almost optimal state whereas simulated annealing (SA) is initially inspired by the laws of thermodynamics in order to cool down a certain matter to its lowest energetic state.

Recently, plenty of work has been investigated, in order to introduce new coding standards and operators especially for genetic algorithms. Almost all of these approaches have one thing in common: They are quite problem specific and often they do not challenge the basic principle of genetic algorithms. Considering the advantages and disadvantages of certain heuristic methods in order to combine there favorable attributes in a generic or problem specific way, a generic hybrid heuristic can be created. In the following we will exemplarily consider the main aspects when designing a hybrid heuristic method. Furthermore, we propose a new approach and look upon the concepts of a standard genetic algorithm as an artificial self-organizing process in order to overcome some of the fundamental problems genetic algorithms are concerned with in almost all areas of applications.

3. Sub-population genetic algorithm

Since SPGA II is based on the concepts of Chang et al. [9], the detail procedures of SPGA will be studied in Section 3.1. Moreover, the idea of global Pareto archive will be further explained in Section 3.2 to justify the reason why it can enhance the performance. Furthermore, the two-stage approach is described in Section 3.3. As for the knapsack problem, since GA does not have the ability to exclude the infeasible solutions, the paper develops a repair operator to fix the problem. The repair method is discussed in Section 3.4. Finally, in order to evaluate the solution quality among different algorithms, the performance metric is discussed in Section 3.5.

3.1. The concept of SPGA and the global Pareto archive

In order to prevent the searching procedure from being trapped into local optimality for the combinatorial problems, this research extends the sub-population-like GA by Chang et al. [9]. There are two main characteristics of the sub-population-like method: (1) numerous small sub-populations are designed to explore the solution space and (2) the multiple objectives are scalarized into a single objective for each sub-population. Because the sub-populations are designed to explore specific region, the Pareto-optimal solution are scattered uniformly over the frontier. A uniform design method related to the scalarizing idea is employed to distribute the solution uniformly [26]. Fig. 1 shows the framework of the SPGA algorithm.

In SPGA, each sub-population works independently and do not communicate with each other. However, from previous researches of Lis and Eiben [4] and Affenzeller [3], the sub-population should communicate to each other so that it may bring better convergence and diversity. Therefore, this research considers how to exchange information within these sub-populations while they are exploring the different solution space. The proposed approach adopts the global Pareto achieve to solve the combinatorial optimization problems. The concept of the global archive for SPGA is shown in Fig. 1.

Since the original design of SPGA does not allow the Pareto archives sharing their own information with each other, each sub-population will work independently. However, if there is a link among these sub-populations, there are chances that once a sub-population finds a better solution, other sub-populations can also adopt it to further improve the solution quality within its own searching area. Consequently, the main idea of the modified SPGA is to employ the same Pareto set, which is assigned as a global archive, to let each sub-population communicate with each other. Hence, after a new solution is identified, other sub-populations can adopt it directly. Therefore, the main difference between the original SPGA and the SPGA II is the Pareto archive. In SPGA, there are m sub-populations and there are m Pareto archives, but there is only one external archive applied in SPGA II. The second difference between SPGA and SPGA II is the two-stage procedure, which can improve the convergence and diversity of the solution quality at each different stage.

3.2. The two-stage approach

According to some empirical results, it is hard to obtain a widespread solution for multi-objective combinatorial problems. Therefore, some algorithms would like to preserve the solutions that are close to the individual objective space. Take the NSGA II [1] to solve a bi-objective problem for example; it keeps the extreme solutions area in the upper and lower corners during the first stage. In other words, the extreme solutions are saved. In addition, since

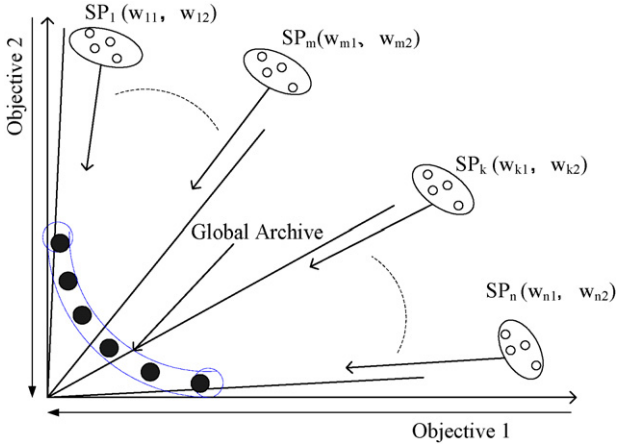

Fig. 1. The framework of the SPGA II.

the multi-population genetic algorithm [5] let the sub-population evolves separately at the second stage, it might be able to explore more extreme solutions. Consequently, the first stage will focus on exploring extreme solutions. Besides, comparing with MPGA, SPGA II applies the global archive technique in the first stages. As a result, even though a sub-population finds a new efficient solution, other sub-populations can also benefit from that.

There is another advantage in adopting few sub-populations at the first stage. Because some sub-populations are initialized in the first stage and they are not re-initialized again at the second stage, these sub-populations still can evolve at the second stage by using the global archive. There are more chances for these global Pareto archive to be convergent since their probabilities of evolving are doubled. In summary, the two-stage approach might result in better converging effect.

In order to simplify the procedure in the second stage, the sum of the number of sub-populations of stage 1 and stage 2 is equal to the total number of sub-populations that is n. The second stage runs all sub-populations that accept some weight vectors located at the middle. For instance, suppose the total number of sub-populations is 35 and there are 10 sub-populations implemented at the first stage. Assume sub-populations from 1 to 5 and from 31 to 35 are executed in the first stage. Then, there are (n-10) sub-populations to be further explored in the second stage, including sub-populations from 1 to 13 and from 24 to 35. The detailed procedures of each stage are explained in the following section.

3.3. Procedures of the SPGA II

The parameters of SPGA II such as N, ns, n, and iteration are total number of chromosomes, total number of sub-populations, number of individuals (chromosomes) in each sub-population, and number of iterations (number of solutions should be examined/ns), respectively. Other parameter set ups for flow-shop scheduling problem in selection, crossover, mutation, objective function calculation, fitness assignment, and weight assignment are binary tournament, two points crossover, shift mutation [25], total makespan and total tardiness-time, and scalarized weight assignment, respectively. The encoding technique of chromosomes is a sequential type for scheduling problems.

As for the knapsack problem, the uniform crossover, bit flip mutation, and total profit of multiple knapsacks are applied. The algorithm is able to handle the flowshop and knapsack problem directly. However, in the Knapsack problem a solution vector x may not be feasible because the constraints are involved; the paper will discuss how to handle the constraint in Section 3.4. The pseudo-code for SPGA II is given as follows:

Algorithm 1. The Main Procedure of SPGA II()

ns: the total number of sub-populations
generations: the total number of generations of each sub-population
p: the last sub-population index of the region 1 at the first stage
q: the first sub-population index of the region 2 at the second stage
m: the last sub-population index of the region 1 at the first stage
k: the first sub-population index of the region 2 at the second stage

```
1: Initialize()
2: DividePopulation()
3: AssignWeightToEachObjectives()
4: Stage One: SPGA (p, q, ns, generations)
5: InitiateSubPopulations(p, m, k, q)
6: Stage Two: SPGA (m, k, ns, generations)
```

Algorithm 2. SPGA(*index1, index2, ns, generations*)

```
1: counter ← 0
2: while counter < generations do
3:   for i = 1 to index1 and i = index2 to ns do
4:       Evaluate Objectives and Fitness(i)
5:       FindPareto(i)
6:       Selection with Elitism Strategy(i)
7:       Crossover(i)
8:       Mutation(i)
9:     Replacement(i)
10:  end for
11:  counter ← counter +1
12: end while
```

Algorithm 3. *InitiateSubPopulations (start1, end1, start2, end2)*

```
1: for i = start1 to end1and i = start2 to end2 do
2:   for j = 1 to populationSize do
3:     Solution = tournamentSelection(p, q, ns)
4:     setSolution (i, j, Solution)
5:   end for
6: end for
```

The procedure *initialize* is to generate a set of chromosomes according to the population size in the beginning. The procedure *DividePopulation* is to divide the original population into *ns* sub-populations. The procedure *AssignWeightToEachObjectives* is to assign different weight values to each sub-population and the individuals in the same sub-population will share the same weight value. Since the problem to be solved in this research is a bi-criteria problem, the vector size is set to two. The combination of weight vector is formulated as follows:

$$(w_{n1}, w_{n2}) = (\frac{1}{+1}\, n,\ 1 - \frac{1}{+1}\, n) \tag{1}$$

where n is the nth sub-population.

After the weight assignment, the corresponding scalarized objective value of these two objectives, i.e., $f_1(x)$ and $f_2(x)$, in each sub-population is defined as Eq. (2). Because the total tardiness and makespan are the two objectives to be considered in the flowshop scheduling problem, we will replace $f_1(x)$ and $f_2(x)$ by Z_{TT} and Z_{TC}, where Z_{TT} and Z_{TC} denote total tardiness time and makespan for each solution x. Z_{TT} and Z_{TC} are defined in Eqs. (4) and (5), respectively.

$$\text{Min/Max } f(x) = w_{n1} f_1(x) + w_{n2} f_2(x) \tag{2}$$

$$\text{Min } f(x) = w_{n1} Z_{TT}(x) + w_{n2} Z_{TC}(x) \tag{3}$$

$$Z_{TT} = \sum_{i=1}^{n} T_i; \quad T_i = \max\{C_i - d_i, 0\} \tag{4}$$

$$Z_{TC} = \max\{F_1, F_2, \cdots, F_n\} \tag{5}$$

It is the same for the Knapsack problem, and $f_1(x)$ and $f_2(x)$ are replaced by $P_1(x)$ and $P_2(x)$, respectively. However, the objective of the knapsack problem is to maximize its profit.

Because the scales of the two objectives are different, the objective values are normalized into a single unit interval. The normalization equations for the first objective function and the

second one are formulated as follows:

$$f_1(x) = \frac{X_{t1}^i - f_{t1}^W}{f_{t1}^B - f_{t1}^W}, \quad 0 \le f_1(x) \le 1 \tag{6}$$

$$f_2(x) = \frac{X_{t2}^i - f_{t2}^W}{f_{t2}^B - f_{t2}^W}, \quad 0 \le f_2(x) \le 1 \tag{7}$$

The *Elitism* strategy adopted at the first stage randomly selects a number of individuals from non-dominated set into the mating pool.

The binary tournament selection is employed at the *selection* operation. Because flowshop scheduling problem is a minimization problem, the smaller fitness value has better chance to be selected during the evolutionary process. Whereas the knapsack problem is a maximization problem, to solve this type of problem we have to transform the objective value by the following formulation:

$$\min f_1(x)' = \mathrm{Max}_1 - f_1(x) \tag{8}$$

$$\min f_2(x)' = \mathrm{Max}_2 - f_2(x) \tag{9}$$

where the Max_1 and Max_2 are the maximum objective value of a population corresponding to the first and second objective function.

There are numerous crossover and mutation methods and two-point crossover and moving position mutation will be applied at the *Crossover* procedure and the *Mutation* procedure, respectively. As claimed by Murata et al. both of them are the best approaches for the flowshop scheduling problems. In addition, one-point crossover and bit flip mutation are applied in the knapsack problem as suggested by Chu and Beasley [13].

After the first stage is finished, the algorithm initiates other sub-populations by tournament selection from all existing solutions generated in stage 1. If the tournament size is 10, the initial solutions for each sub-population are randomly selected from those solutions generated in stage 1. The fitness of these solutions will be re-calculated according to the weight vector of each sub-population and the best one will be selected for that particular sub-population. Thus those selected solutions will be further evolved in stage 2. The advantage of this approach is to exchange the information from existing sub-populations. The approach not only increases the convergent ability but also exchanging the solutions among different sub-populations. Thus, a better solution quality from SPGA II is expected. Fig. 2 depicts the flow diagram of SPGA II.

The differences between SPGA II and SPGA are listed as follows:

1. A global Pareto archive: As soon as a new Pareto front is found, the solution is inserted into the global Pareto archive which can be further applied by other sub-populations in later evolutions.
2. Initiate solutions in stage 2 by tournament selection from all existing solutions: After the evolution process in stage 1, set of Pareto fronts is obtained. In stage 2, initial solutions for these sub-populations will be generated by selecting those better ones from these existing solutions, which will speed up the convergent progression.

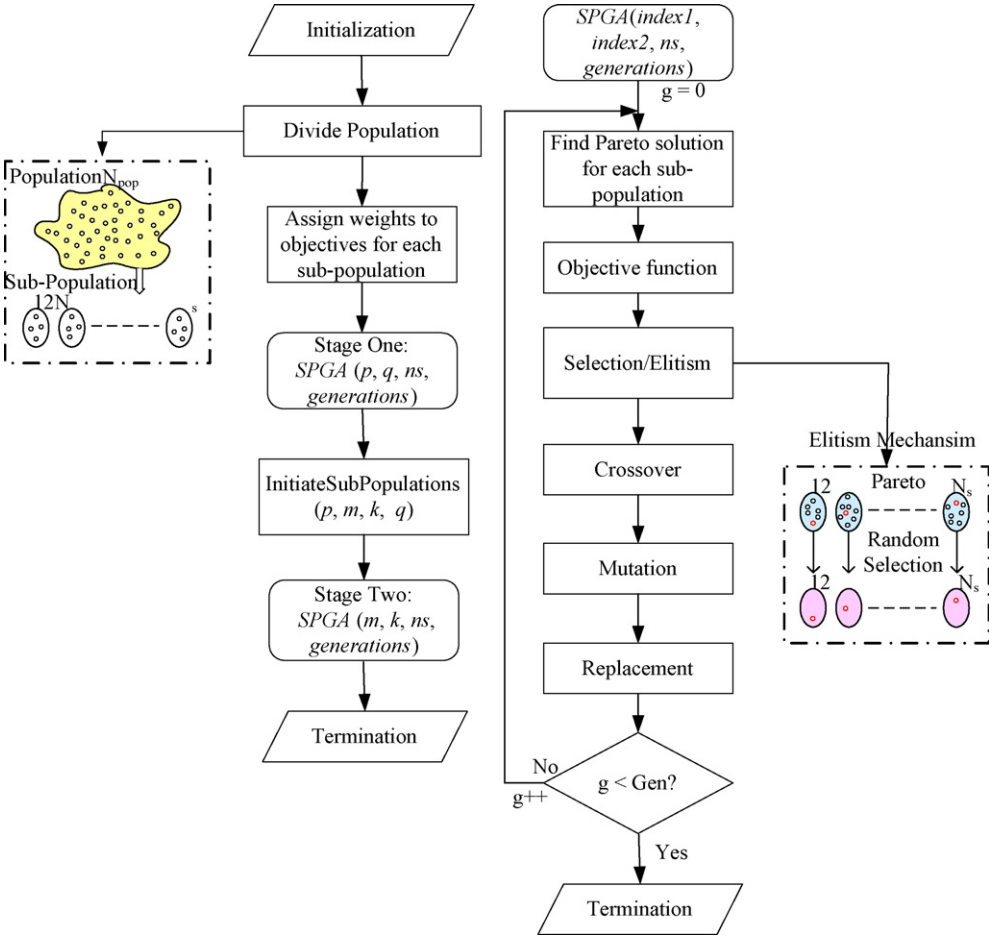

Fig. 2. The flow diagram of SPGA II.

3. Two stages approach: The sub-populations at the first stage explore specific solution space near the respective objective function. These sub-populations will be continually evolved at the second stage. The existing solutions are randomly selected into the sub-populations created at stage 2 by tournament selection. This approach enables the system to recombine different species from different sub-population, thus a more diversified solutions with better quality are produced.

3.4. The constraint handling methods for Knapsack problems

The model of the multidimensional Knapsack problem (MKP) or k-dimensional Knapsack problem is defined as follows:

$$\text{Maximize} \sum_{i=1}^{k}\sum_{j=1}^{m} w_{ni}\, p_{ij} x_j \tag{10}$$

$$\text{Subject to} \sum_{j=1}^{m} r_{ij} x_j \le b_i, \quad i = 1,2,\ldots,k. \tag{11}$$

$$x_j \in \{0,1\}, \quad j = 1,2,3,\ldots,m. \tag{12}$$

where I is the index of knapsack, j the index of each item, p_{ij} is the item profit of item j in knapsack I and r_{ij} is the resource weight of item j in knapsack i.

Eq. (10) is the weighted scalar objective function of k-dimensional nth sub-population. The k-dimensional constraint shown in Eq. (11) is a knapsack constraint. The solutions generated by GA may not be feasible because of the knapsack constraints. Thus, a constraint handling method or a repair operator is needed in handling these infeasible solutions. Both methods are widely applied in GAs.

The constraint handling methods are divided into three categories, including the greedy repair, penalty functions, and permutation coding in knapsack problems [14]. Though the constraint handling functions achieve the best result with capacity of half the total weight [15] in the single objective 0/1 knapsack problem, it fails on the problem with more restricted capacity. Therefore [2] used the Lamarckian implementation scheme, which is implicitly applied to solve the multi-objective 0/1 knapsack problems without using constraint-handling function [16–20].

There is another repair scheme by Baldwinian implementation, which is better than the Lamarckian implementation scheme [14]. The difference between these two is that the repair solution of Lamarckian implementation scheme replaces the infeasible solution while the Baldwinian implementation scheme does not. The concept of Baldwinian implementation scheme is shown below (Fig. 3).

The research of Ishibuchi et al. [14] compares two implementation schemes (i.e., Baldwinian and Lamarckian) and two repair methods (i.e., maximum ratio repair and weighted scalar repair),

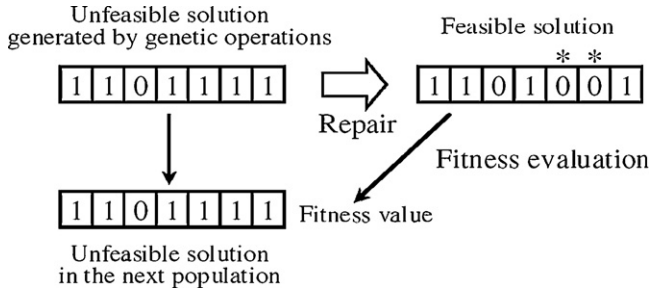

Fig. 3. The Baldwinian implementation [14].

the combination of Baldwinian implementation scheme and the weighted scalar repair is the best among the four possible combinations. In addition, the partial Baldwinian repair further improves the solution quality with a so-called 5% rule. The rule means that there are 5% solutions which are actually repaired. Thus, this research will apply the result of Ishibuchi et al. [14] to deal with infeasible solutions.

The weight vector, w_{ni}, can be directly applied in the weighted repair method because it corresponds to the weight vector of each sub-population. According to the weight vector, the following ratio q_j is subsequently computed and they are sorted in the ascending order:

$$q_j = \frac{\sum_{i=1}^{k} w_{ni}\, p_{ij}}{\sum_{i=1}^{k} r_{ij}}, \quad j = 1,2,3,\ldots,m. \tag{13}$$

In Chu and Beasley [13], they applied the DROP phase and ADD phase for each item to build their repair operator according to q_j value. The add phase distinguishes itself from most repair operators in previous researches. Through add phase, the solution quality should be further improved. The procedures of the repair operator are described as follows:

Algorithm 4. *Repair Operator* ()

R_i: The cumulated resource weight $\sum_{j=1}^{m} r_{ij} x_j$ of knapsack i, $i = 1, 2 \ldots k$.
b_i: The maximum weight capacity of knapsack i.

```
1: Calculate R_i and q_j
2: while (R_i > b_i, for any i = 1 to k) then/* DROP phase */
3: for j = 1 to m do/* Ascending order of q_j*/
4:     if x_j = 1 then
5:         x_j ← 0
6:         R_i ← R_i − r_ij, for any i = 1 to k
7:     End if
8:   End for
9: End While
10: for j = m to 1 do/* ADD phase and descending order of q_j*/
11: if (x_j = 0) and (R_i + r_ij ≤ b_i, for any i = 1 to k) then
12:         x_j ← 1
13:         R_i ← R_i + r_ij, for any i = 1 to k
14:     End if
15: End for
```

The steps 2–9 are the DROP phase and the ADD phase is from step 10–15. Step 2 continually tests the feasibility solution of solution x. If one constraint is violated, the method removes an item greedily in the ascending order of q_j. After the DROP phase is done, the repair operator tries to add more items into knapsack in the descending order of q_j. The $R_i + r_{ij} \le b_i$ guarantees that the ADD phase will not violate the constraints so that the repair operator produces a feasible solution.

Finally, when SPGA II is applied to solve the knapsack problem, Algorithm 4, i.e., *Repair Operator*, is used before the evaluate objectives and fitness (i) function in Algorithm 2.

3.5. The evaluation metric for multi-objective algorithm

This research adopts $D1_R$ and generational distance (GD) to evaluate the solution quality. The $D1_R$ is a metric, which considers convergence and diversity at the same time [21]. After a run, an algorithm obtains a set of Pareto solutions, which is compared with a reference set. Thus, the $D1_R$ value is obtained. The lower the $D1_R$ value is; the better the solution quality is. Therefore, the $D1_R$ provides a basis for comparing the performances among different

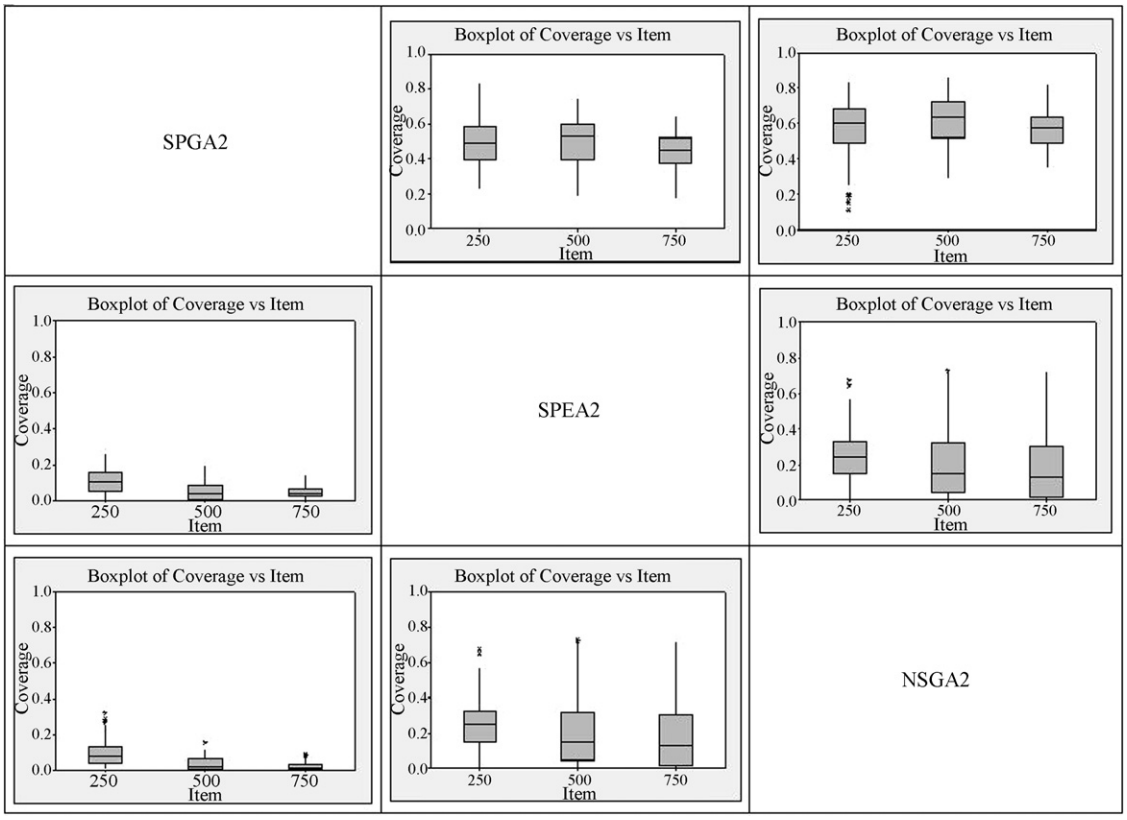

Fig. 4. The comparisons of different algorithms in *C* metric.

algorithms in the study. The $D1_R$ is formulated as follows:

$$D1_R(A_j) = \frac{1}{|Z^*|}\sum_{y\in Z^*}\min\{d_{xy}|x\in A_j\} \tag{14}$$

$$d_{xy} = \sqrt{(f_1(x) - f_1(y))^2 + (f_2(x) - f_2(y))^2} \tag{15}$$

where A_j is the set of Pareto solution obtained by an algorithm; Z^* is the reference solution or true Pareto solution; $|Z^*|$ is the number of reference solution.

GD is developed by Van Veldhuizen (1999) [27] and it has several advantages such as strong out performance, complete out performance, non-inducing, non-cardinality, and relatively reasonable calculation effort [20]. The equation of GD is defined as follows:

$$GD(S) = \frac{1}{|S|}\sum_{x\in S}\min\{d_{xy}|y\in S^*\} \tag{16}$$

4. Experimental results

In order to validate the performance of the SPGA II, the study takes the parallel scheduling instances[1] and the knapsack instance provided by Zitzler and Laumanns.[2] In addition, the proposed algorithm is compared with original version of SPGA and two well-known multi-objectives GA, which are the NSGA II [1] and SPEA 2 [22]. The comparison results of different algorithms in parallel machine scheduling problem and knapsack problem are shown in Section 4.1 and 4.2, respectively. Finally, the design of experiment is applied to set up the parameters in SPGA II.

[1] http://ppc.iem.yzu.edu.tw/download.html.
[2] http://www.tik.ee.ethz.ch/~zitzler/testdata.html.

4.1. Parallel machine scheduling problem

According to Chang et al. [23] which examines the parameter configuration and clone strategy for sub-population genetic algorithm, the paper directly applies these suggested results as shown in Table 1.

We compare SPGA II with SPGA, NSGA II [1], and SPEA 2 (Zitzler, 2001), in three testing instances. Table 2 shows the minimum, average, and maximum values of different algorithms for these three instances. The instances include scheduling problems with 35 jobs and 10 machines, 50 jobs and 15 machines, and 65 jobs and 18 machines. From these three instances, SPGA II is superior to SPGA, NSGA II, and SPEA 2 in minimum, average, and maximum value of D1r metric. In addition, it is reasonable that the computation time of SPGA-Like algorithm is less than NSGA II and SPEA 2. The reason is that NSGA II and SPEA 2 need to calculate the Pareto dominance which the time-complexity is $O(n^2)$. Because the fitness evaluation of SPGA-Like is depended on the weighted scalarization, the time-complexity is only constant time. As a result, SPGA and SPGA II work more efficiently than NSGA II and SPEA 2. When compared SPGA, the CPU time of SPGA II is more

Table 1
The suggested parameters set up and clone strategy for SPGA 2 in scheduling problem

Factor	Treatment
Crossover rate	0.9
Mutation rate	0.1
Population size	210
Number of sub-population	40
Clone Strategy	Swap mutation
Time to clone	In the beginning

Table 2
The min, average, and max value of different algorithms of the three instances

Instance	Algorithm	Min	Avg.	Max	CPU
35/10	SPGA 2	0.3	1.295	3.682	0.99
	SPGA	3.57	5.8	9.34	0.77
	NSGA II	5.16	11.82	22.22	1.47
	SPEA 2	4.8	10.39	22.48	2.23
50/15	SPGA 2	1.5592	2.6272	3.6806	1.45
	SPGA	8.57	9.68	10.78	1.05
	NSGA II	9.68	11.74	13.79	1.46
	SPEA 2	7.65	10.27	12.89	2.23
65/18	SPGA 2	5.598	11.51	16.183	1.95
	SPGA	17.88	18.98	20.08	1.27
	NSGA II	20.97	23.08	25.43	1.47
	SPEA 2	7.7	10.3	12.9	2.29

Table 3
The suggested parameter and clone strategy for the SPGA 2 in Knapsack problem

Factor	Treatment
Population size	200
Crossover rate	0.8
Mutation rate	0
Population size	200
Number of sub-population	15
Clone strategy	Adjust fitness
Time to clone	In the beginning

demanding because SPGA II requires extra efforts in transferring existing solutions to a new sub-population.

4.2. Bi-Criteria knapsack problem

Based on the knapsack instance of Zitzler et al. [22], the item 250, 500, and 750 of two knapsacks are adopted in the experiment. A preliminary experiment is done to obtain better setting for SPGA II in the knapsack problem. The parameter settings for these experiments can be obtained in http://ppc.iem.yzu.edu.tw/download.html. The general result of the parameter setting is shown in Table 3. Zitzler et al. [22] specify the maximum number of evaluations for the three instances. Thus the SPGA II, SPEA 2, and NSGA II evaluate the same specific number of solutions. Table 4 demonstrates the parameter values of the population size and the maximum number of evaluations in different knapsack instance.

The performance evaluation along with the coverage measure *C* by box-plot is presented in Fig. 4. In addition, the evaluations of *R* metric are depicted from Figs. 5–7 for item 250, 500, and 750.

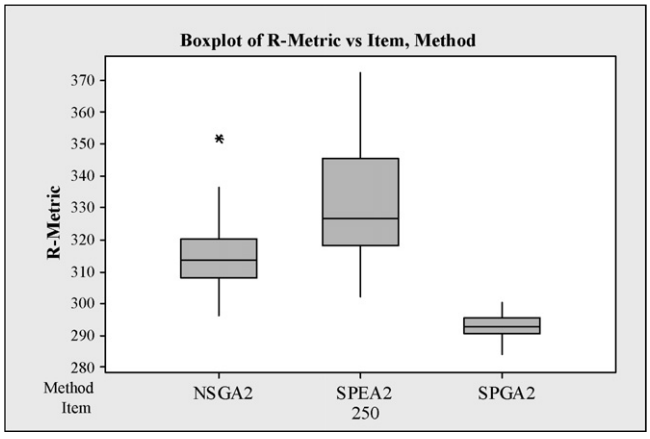

Fig. 5. The *R* metric of different algorithms for item 250.

Table 4
The parameter settings for SPEA 2 and NSGA II and relative parameter settings

Instance	Population size		Maximum number of evaluations	Reference point
	SPEA 2	NSGA II		
2–250	120	150	75,000	[9898.86, 10107.3]
2–500	160	200	100,000	[20086.5, 20494.6]
2–750	200	250	125,000	[30130.1, 20037.2]

As shown in Fig. 5, SPGA II performs much better than SPEA 2 and NSGA II in R metric. The third quartet (Q3) of SPGA II is lower than the Q1 of SPEA 2 and NSGA II. In addition, the width between the first quartet (Q1) and Q3 of SPGA II is smaller which means that SPGA II outperforms SPEA 2 and NSGA II.

According to the *C* metric, SPGA II is covered more points by SPEA 2 and NSGA II while SPGA II covers only small proportion of points from SPEA 2 and NSGA II. To further analyze the result, we plot the Pareto solutions obtained from these three algorithms as shown in Fig. 8. The diagram shows that the Pareto solutions SPGA II, SPEA 2 and NSGA II are all covered to each other. In addition, some efficient points of SPEA 2 and NSGA II dominate the Pareto solutions of SPGA II. However, the advantage of SPGA II is that it generates wider spread of efficient solutions than SPEA 2 and NSGA II. Thus, it is quite obvious that C metric concerns more about the convergence effect.

Finally, Table 5 is the computation time of the three algorithms. It shows that the SPGA II takes longer time which

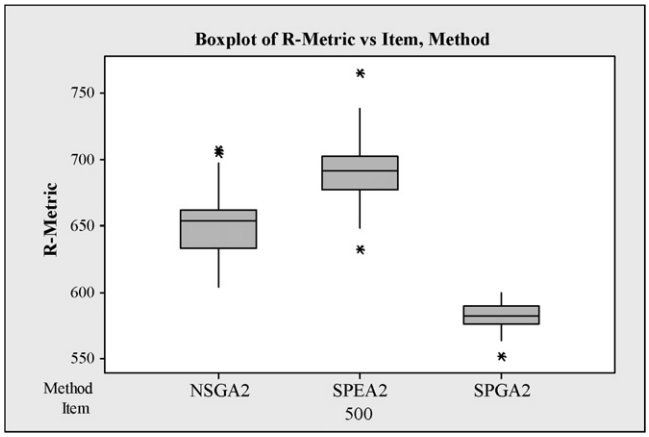

Fig. 6. The *R* metric of different algorithms for item 500.

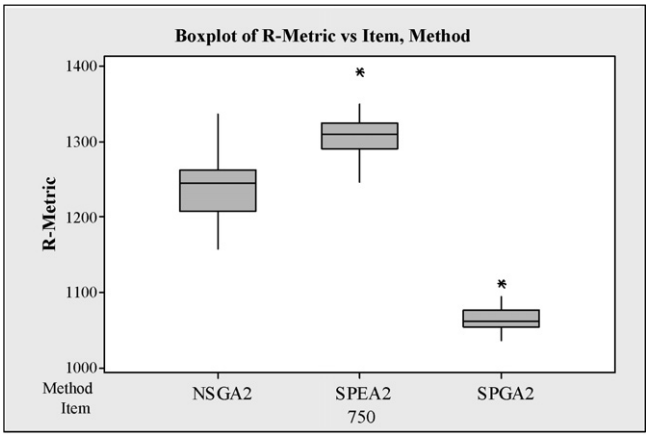

Fig. 7. The *R* metric of different algorithms for item 750.

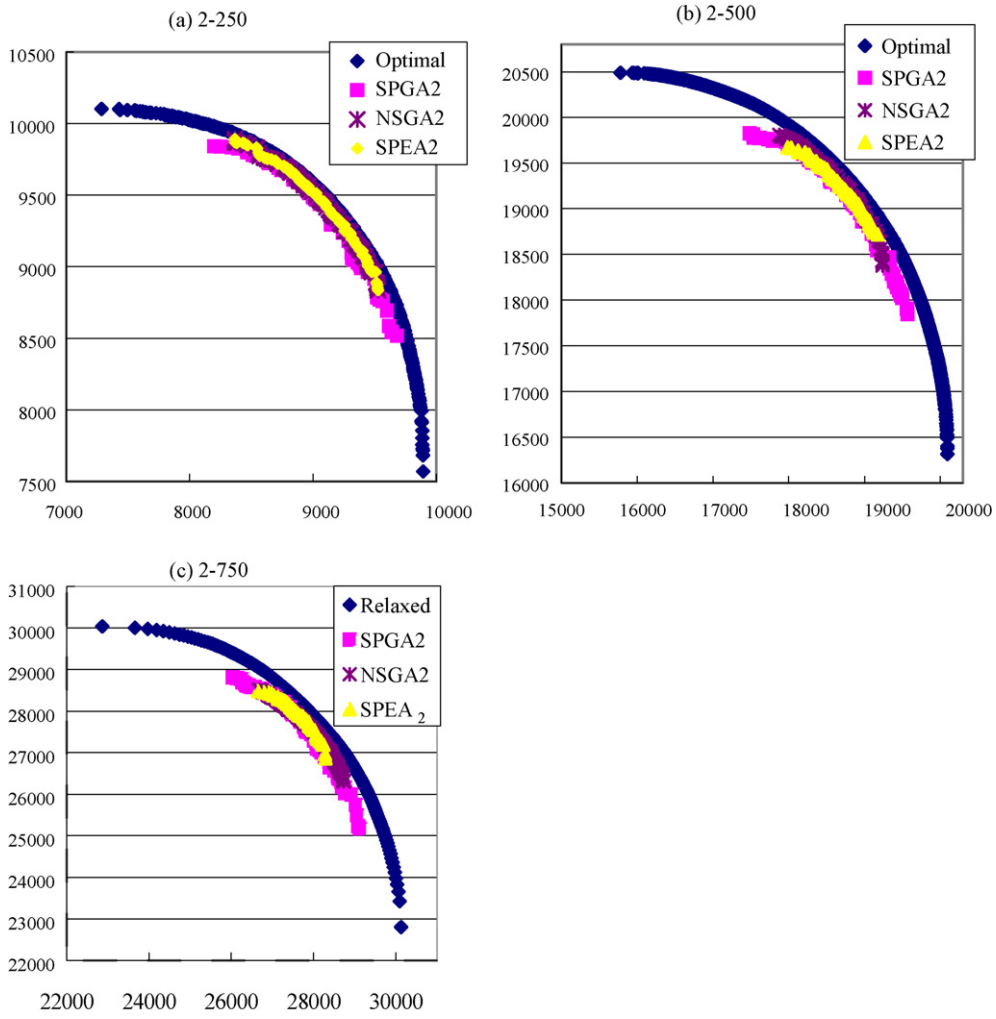

Fig. 8. The Pareto fronts of three different algorithms in a single run.

Table 5
Computation time of the three algorithms (s)

Instance	Algorithm	CPU
2–250	SPGA 2	9.19
	NSGA II	0.41
	SPEA 2	0.63
2–500	SPGA 2	17.40
	NSGA II	0.43
	SPEA 2	0.62
2–750	SPGA 2	29.65
	NSGA II	0.42
	SPEA 2	0.65

is because the SPGA 2 uses a repair operator to deal with the constraint handling while NSGA II and SPEA 2 which implicitly used the Lamarckian implementation scheme instead of using a repair operator. It is obviously that although this repair operator proposed by Chu and Beasley [13] could improve the solution quality; the repair operator will causes significant computational times.

5. Conclusions

This paper extends SPGA algorithm with a global Pareto archive technique and a two-stage approach to solve the multi-objective problems. In the first stage, the areas next to the two single objectives are searched and solutions explored around these two extreme areas are reserved in the global archive for later evolutions. Then, in the second stage, larger searching areas except the middle area are further extended to explore the solution space in finding the near-optimal frontiers. In order to know which factors may influence the solution quality and to obtain better parameter configurations for SPGA II, a two-stage design of experiment (DOE) is employed.

The first stage is the screening experiment, which considers several factors at the same time. Because the number of combinations is high, it is time-consuming to study all the combinations. Instead, the research uses the 2^{6-1}_{IV} design, which greatly decreases the experimental efforts. The second stage is to explore more detailed parameter configuration. Since there are only two factors considered here, the study applies the full factorial design to provide more accurate experiment results.

Finally, through extensive experimental results, SPGA II does outperform SPGA (There is only one exception, i.e., the first instance, that the performance of SPGA 2 is not better than the SPGA), NSGA II, and SPEA 2 in the parallel scheduling problems and knapsack problems. It shows that the approach improves the sub-population genetic algorithm significantly. It may be of interests for researchers in solving multi-objective combinatorial problems.

References

[1] K. Deb, S.A. Amrit Pratap, T. Meyarivan, A fast and elitist multi-objective genetic algorithm-NSGA-II, in: Proceedings of the parallel problem solving from nature VI conference, 2000, pp. 849–858.

[2] E. Zitzler, M. Laumanns, S. Bleuler, A tutorial on evolutionary multi-objective optimization, in: Proceedings of the workshop on multiple objective metaheuristics, 2004.

[3] M. Affenzeller, New Generic Hybrids Based Upon Genetic Algorithms, Institute of Systems, Science Systems Theory and Information Technology, Johannes Kepler University, 2001.

[4] J. Lis, A.E. Eiben, A multisexual genetic algorithm for multicriteria optimization, in: Proceedings of the 4th IEEE Conference on Evolutionary Computation, 1997, pp. 59–64.

[5] J.K. Cochran, S.M. Horng, J.W. Fowler, A multi-population genetic algorithm to solve multi-objective scheduling problems for parallel machines, Comput. Oper. Res. 30 (7) (2003) 1087–1102.

[6] J. Hu, E. Goodman, K. Seo, Z. Fan, R. Rosenberg, The hierarchical fair competition framework for sustainable evolutionary algorithms, Evol. Comput. 13 (2) (2005) 241–277.

[7] C.A.C. Coello, G.T. Pulido, M.S. Lechuga, Handling multiple objectives with particle swarm optimization, IEEE Trans. Evol. Comput. 8 (3) (2004) 256–279.

[8] S. Mostaghim, J. Teich, Covering Pareto-optimal fronts by subswarms in multi-objective particle swarm optimization, Evol. Comput. 2 (2004) 1404–1411.

[9] P.C. Chang, S.H. Chen, K.L. Lin, Two-phase sub population genetic algorithm for parallel machine-scheduling problem, Exp. Syst. Appl. 29 (3) (2005) 705–712.

[10] Q. Zhang, H. Li, MOEA/D: a multiobjective evolutionary algorithm based on decomposition, IEEE Trans. Evol. Comput. 11 (6) (2007) 712–731.

[11] P.C. Chang, S.H. Chen, C.H. Liu, Sub-population genetic algorithm with mining gene structures for multi-objective flowshop scheduling problems, Exp. Syst. Appl. 33 (3) (2007) 762–771.

[12] K.C. Tan, C.K. Goh, A.A. Mamun, E.Z. Ei, An evolutionary artificial immune system for multi-objective optimization, Eur. J. Oper. Res. 187 (2) (2008) 371–392.

[13] P.C. Chu, J.E. Beasley, A genetic algorithm for the multidimensional knapsack problem, J. Heuristics 4 (1) (1998) 63–86.

[14] H. Ishibuchi, S. Kaige, K. Narukawa, Comparison between Lamarckian and Baldwinian repair on multiobjective 0/1 knapsack problems", in: Proceedings of the 3rd International Conference EMO, vol. 3410, 2005, pp. 370–385.

[15] Z. Michalewicz, J. Arabas, Genetic algorithms for the 0/1 knapsack problem, in: Methodologies for Intelligent Systems (ISMIS '94), vol. 869, 1994, 134–143.

[16] C.L. Mumford, Comparing representations and recombination operators for the multi-objective 0/1 Knapsack problem, Proceedings of the International Congress on Evolutionary Computation, (2003), pp. 854–861.

[17] J.B. Zydallis, G.B. Lamont, Explicit building-block multiobjective evolutionary algorithms for NPC problems, in: Proceedings of the International Congress on Evolutionary Computation, 2003, pp. 2685–2695.

[18] H. Ishibuchi, S. Kaige, Effects of repair procedures on the performance of EMO algorithms for multiobjective 0/1 Knapsack problems, in: Proceedings of the International Congress on Evolutionary Computation, 2003, pp. 2254–2261.

[19] A. Jaszkiewicz, On the performance of multiple-objective genetic local search on the 0/1 Knapsack problem—a comparative experiment, IEEE Trans. Evol. Comput. 6 (2002) 402–412.

[20] J.D. Knowles, D.W. Corne, A comparison of diverse approaches to mimetic multi-objective combinatorial optimization, in: Proceedings of the Genetic and Evolutionary Computation Conference Workshop Program, 2000, pp. 103–108.

[21] J.D. Knowles, D.W. Corne, On metrics for comparing non-dominated sets, in: Proceedings of the international congress on evolutionary computation conference (CEC02), IEEE Press, New York, 2002, pp. 711–716.

[22] E. Zitzler, M. Laumanns, L. Thiele, SPEA 2: improving the strength Pareto evolutionary algorithm, TIK report 103, computer engineering and networks laboratory (TIK), 2001.

[23] P.C. Chang, S.H. Chen, J.C. Hsieh, A global archive sub-population genetic algorithm with adaptive strategy in multi-objective parallel-machine scheduling problem, 4221, 2006, 730–739.

[24] M.J. Alves, M. Almeida, MOTGA: a multi-objective Tchebycheff based genetic algorithm for the multidimensional Knapsack problem, Comput. Oper. Res. 34 (11) (2007) 3458–3470.

[25] H. Ishibuchi, N. Yamamoto, T. Murata, H. Tanaka, Genetic algorithms and neighborhood search algorithms for fuzzy flowshop scheduling problems, Fuzzy Sets Syst. 67 (1) (1994) 81–100.

[26] Y.W. Leung, Y.P. Wang, Multiobjective programming using uniform design and genetic algorithm, IEEE Trans. Syst. Man Cybernet. C: Appl. Rev. 30 (3) (2000) 293–304.

[27] D.A. Van Veldhuizen, Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations, in: Ph. D. thesis, Graduate School of Engineering of the Air Force Institute of Technology, Air University, 1999.