Available online at www.sciencedirect.com
ScienceDirect
Expert Systems with Applications 33 (2007) 762–771
Expert Systems with Applications
www.elsevier.com/locate/eswa

Sub-population genetic algorithm with mining gene structures for multiobjective flowshop scheduling problems

Pei-Chann Chang [a,*], Shih-Hsin Chen [b], Chen-Hao Liu [b]

[a] Department of Information Management, Yuan Ze University, 135 Yuan-Tung Road, Chung-Li, Taoyuan 32026, Taiwan, ROC
[b] Department of Industrial Engineering and Management, Yuan Ze University, 135 Yuan-Tung Road, Chung-Li, Taoyuan 32026, Taiwan, ROC

Abstract

According to previous research of Chang et al. [Chang, P. C., Chen, S. H., & Lin, K. L. (2005b). Two phase sub-population genetic algorithm for parallel machine scheduling problem. *Expert Systems with Applications*, *29*(3), 705–712], the sub-population genetic algorithm (SPGA) is effective in solving multiobjective scheduling problems. Based on the pioneer efforts, this research proposes a mining gene structure technique integrated with the SPGA. The mining problem of elite chromosomes is formulated as a linear assignment problem and a greedy heuristic using threshold to eliminate redundant information. As a result, artificial chromosomes are created according to this gene mining procedure and these artificial chromosomes will be reintroduced into the evolution process to improve the efficiency and solution quality of the procedure. In addition, to further increase the quality of the artificial chromosome, a dynamic threshold procedure is developed and the flowshop scheduling problems are applied as a benchmark problem for testing the developed algorithm. Extensive tests in the flow-shop scheduling problem show that the proposed approach can improve the performance of SPGA significantly.

Keywords: Genetic algorithms; Multiobjective optimization; Pareto optimum solution; Minging gene structures; Scheduling problem

1. Introduction

In the operations research literature, flowshop scheduling is one of the most well-known problems in the area of scheduling. Flowshops are useful tools in modeling manufacturing processes. A permutation flowshop is a job processing facility which consists of several machines and jobs to be processed on the machines. In a permutation flowshop all jobs follow the same machine or processing order and job processing is not interrupted once started. Our objective is to find a sequence for the jobs so that the makespan or the completion time is a minimum.

Various approaches to this problem have been proposed since the pioneering work of Johnson (Affenzeller, 2002). Genetic Algorithms have been applied (Davis, 1991; Goldberg, 1989; Holland, 1975) to combinatorial optimization problems such as the traveling salesman problem (Jog, Suh, & Gucht, 1989; Starkweather, McDaniel, Whitley, Mathias, & Whitley, 1991; Ulder, 1991), and scheduling problems (see for example, Fox & McMahon, 1991; Ishibuchi, Yamamoto, Murata, & Tanaka, 1994). A simulated annealing (SA) approach to the flowshop problem was proposed by Osman and Potts (1989) and Ogbu and Smith (1990). This approach was shown to produce high quality solutions. The performance of these heuristics has been measured on a set of 120 benchmark instances of the PFSP proposed in Taillard (1991). A GA for flowshop scheduling was proposed by Reeves (1995), which was then tested on several categories of problems with time gradients and job correlations and some hard test problems. GA was overall seen to produce results comparable to SA for the flowshop sequencing problem for most types and sizes of problems. Further, GA was shown to perform relatively better for large problems and reaches near-optimal solutions earlier. Both GA and SA outperform other heuristics.

* 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 © 2006 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2006.06.019

2. Approaches in multiobjective scheduling problems

The flow shop scheduling problem is a NP-Hard problem, which all algorithms cannot solve it in a polynomial time. When the problem size is larger than 50, it is time-consuming to solve by exact algorithms, such as Branch-and-Bound algorithm and dynamic programming. Therefore, researchers developed some metaheuristics to solve it. Moreover, if the multiple objectives of a problem are considered, it becomes more complex. Therefore, recent development in evolutionary multiobjective optimization provides interesting results as discussed by Deb, Amrit, Sameer, and Meyarivan (2002) Zitzler, Laumanns, and Bleuler (2004). In addition, some sub-population-like approaches also can be found in related literatures, such as segregative genetic algorithms (Affenzeller, 2002), multisexual genetic algorithm (Lis & Eiben, 1997), multipopulation genetic algorithm (Cochran, Horng, & Fowler, 2003), hierarchical fair competition model (Hu, Goodman, Seo, Fan, & Rosenberg, 2005), MO particle swam optimization (Coello, Pulido, & Lechuga, 2004; Mostaghim & Teich, 2004) and two phases sub-population GA (Chang, Chen, & Lin, 2005b). Therefore, different EMO algorithms are proposed and the efficiency and solution quality are greatly improved.

Since Schaffer (1985) proposed VEGA (vector evaluated genetic algorithm) in 1985, the application of GAs in solving multiobjective problems has been developed and it bloomed in the mid 1990s. The research introduces two of the most famous and well-known approaches of GAs: SPEA-II and NSGA-II in the following, and because the proposed algorithm is established based on SPGA, it will be described as well.

2.1. Strength Pareto evolutionary algorithm-II (SPEA-II)

SPEA-II was proposed by Zitzler, Laumanns, and Thiele (2001) which is improved from SPEA (Zitzler & Thiele, 1999). There are three main differences between SPEA-II and SPEA:

1. *Fitness assignment*: In SPEA-II, each individual has a strength value calculated. The calculation model of SPEA-II for fitness is not only based on the strength value, but also on the density degree.
2. *Density estimation*: In SPEA-II, besides the strength value, each individual has its own σ_i^k, which is called the value of density in our study. The calculation of σ_i^k is to sort first the values of all objectives; each individual σ_i^k is equal to the sum of the value difference between each objective value and its kth neatest neighbor. The fitness value equals the strength value plus the density degree.
3. *Archive truncation method*: In SPEA-II, the size the outside storage file is fixed, and the individual crossover and mutation process can only take place in the storage file.

2.2. Non-dominated sorting genetic algorithm-II (NSGA-II)

NSGA-II was proposed by Deb et al. (2002). To improve the fitness assignment of NSGA, it adopts a crowding distance to measure the density of individuals in solution space. The algorithm of NSGA-II can be divided in three parts:

1. *Non-dominated sorting*: N populations and their N subpopulations first compose of a $2N$ population and then they are sorted according to each individual's domination situation.
2. *Crowding distance computation*: Crowding distance computation is used to decrease the competitive ability of the non-dominated solutions with more crowding distance.
3. *Crowded computation operator* $\prec_n$: This operator is used as a selection tool. The selection is based on two comparison rules: (1) the smaller level the individual belongs to, the better the solution is; (2) an individual with greater crowding distance has better solution because the area it belongs to is less crowded.

2.3. Sub-population genetic algorithm (SPGA)

SPGA was proposed by Chang et al. (2005b). In this approach, the original population is divided into several sub-populations which will be assigned with different weights to search optimal solutions in different directions. This way can consider both the expanding and converging natures of the solutions. There are three main characteristics of SPGA:

1. The original population is divided into numerous small sub-populations which are designed to explore the solution space; each sub-population is independent and unrelated to each other.
2. The multiple objectives are scalarized into single objective and each sub-population is assigned with different weight, which stands for different searching direction. An outside storage file is set up to record the non-dominated solutions appeared during the searching process. Certain individuals are chosen from the file in the process of crossover.
3. Replacement process has to be converged after certain generations; that is, the sub-populations will replace the original populations only when they are superior to the original ones.

The searching procedure of SPGA is depicted as shown in Fig. 2.1.

2.4. Mining gene structure (MGS)

There is a lot of useful information hidden in the evolvement process of GAs and each produced solution also

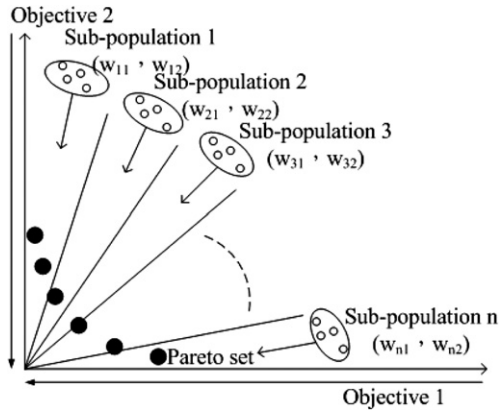

Fig. 2.1. The framework of SPGA (Chang et al., 2005b).

reveals certain useful signals which can be used for further applications; for example, Chang, Wang, and Liu (2005a) proposed a methodology to improve GAs by mining gene structures which was used to solve traveling salesman problem, (TSP). This study will integrate mining gene structures with SPGA such that the convergence of solutions can be speeded and solutions with better quality can be obtained. A "fabrication operator" is proposed to generate a new set of chromosomes by collecting useful information from elite chromosomes generated in previous generations. For each chromosome, a gene represents the job and the sequence the position each job is assigned. The number of times of the job showing up at each different position was counted and recorded in a "dominance matrix". This process was named "voting". The element M_{ij} in the matrix represented the times that job i appeared at position j. Thus a dominance matrix generated from the chromosome base is formed. According to this dominance matrix, an "artificial chromosome" can be generated by the following steps:

Let

E	the elite set, $E \in \{\pi_1, \pi_2, \ldots, \pi_i\}$
Pos	the position of a job
size	the size of the E
length	the length of chromosome
j	the number of job
$M_{j,\mathrm{Pos}}$	the value of the dominance matrix
A	the set of jobs has to be assigned
A'	the set of assigned-jobs
B'	the set of weed-out jobs
$\lvert V_l \rvert$	the highest number of vote of the lth sequence = MAX(V_{kl}), $\forall k$
$\lvert C_l \rvert$	the job with the highest number of vote of the lth sequence

Processes of generate fabricated chromosomes:

Step 0. Set $i = 1$.
Step 1. Set $\pi = E(i)$ and Pos = 1.
Step 2. $j = \pi(\mathrm{Pos})$, $M_{j,\mathrm{Pos}} = M_{j,\mathrm{Pos}} + 1$.
Step 3. If Pos small than the length, then Pos++, go to Step 2.
Else if i small than size, then i++, go to Step 1.
Else go to Step 4.
Step 4. $\lvert V_l \rvert$ is the maximum of M_{JP}.
If there is any other l' where $\lvert V_l \rvert = \lvert V_{l'} \rvert$, go to step 3; otherwise, go to step 2.
Step 5. Remove job $\lvert C_l \rvert$ from A to A', and let it to hold the lth sequence in fabricated chromosomes. Then, go to step 4.
Step 6. Remove jobs $\lvert C_l \rvert$ and $\lvert C_{l'} \rvert$ from A to B'. Then, go to step 4.
Step 7. If $A = \phi$, then, go to step 5; otherwise, go back to step 1.
Step 8. Randomly assign the jobs in B' to the unassigned sequences in fabricated chromosomes.

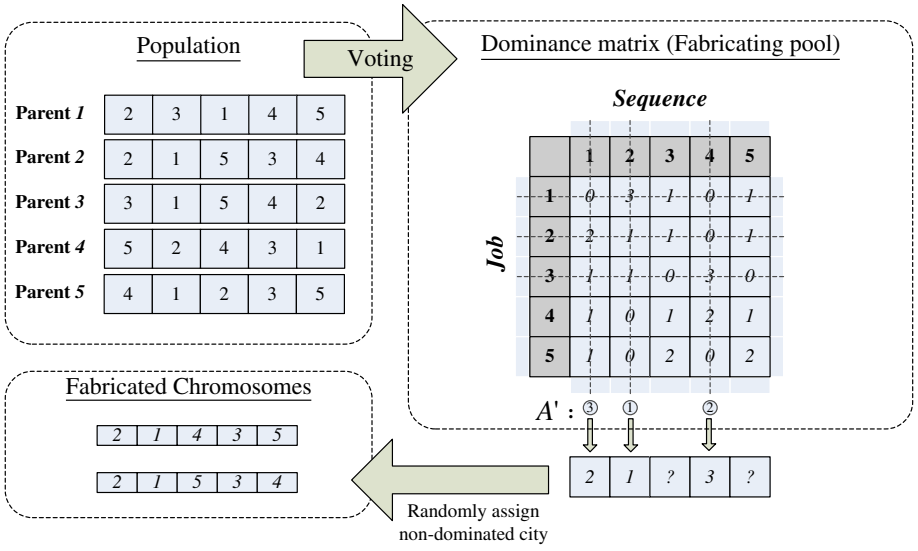

Fig. 2.2. The working procedure of fabrication operator (Chang et al., 2005a).

The following is a simple example for the fabrication operator as shown in Fig. 2.2, i.e., a simple example for th fabrication operator.

This research extends SPGA (Chang et al., 2005b) by integrating the mining gene structure approach with SPGA, which becomes the so-called MGSPGA (mining gene structures on sub-population genetic algorithm), a algorithm with both natures of convergence and diffusivity. Detailed development of the integrated approach will be described in the next section.

3. Methodology

A new GA is developed in this research which is called mining gene structures on sub-population genetic algorithm (MGSPGA). The method is proposed to solve flow-shop scheduling problems and will be compared with SPGA, NSGA-II and SPEA-II. Through literature reviews, we find that SPGA has very good diffusivity when solving multiobjective problems; however, as for convergence, there still remains room for improvement. Thus, the research tries to strengthen the solution convergence of SPGA by mining gene structures. Except for the original mining gene structures (MGS) (Chang et al., 2005a), there are two more methods proposed – weighting minging gene structures (WMGS) and threshold mining gene structures (TMGS); these three methods are later integrated with SPGA. This section is organized as the following: the three approaches of MGS are first introduced, and then the steps of solutions are depicted. Finally, the evaluation approach to use for the comparison of each algorithm is introduced.

3.1. Approaches of mining gene structures

3.1.1. Mining gene structures (MGS)

Mining gene structures was proposed by Chang et al. (2005a) who used statistic sorting of elite chromosomes to find better genetic sorts for the making of artificial chromosomes. When collecting genetic information, the sorting of each elite chromosome will be recorded in a dominance matrix. After several generations, the artificial chromosomes will be generated from voting results. After the first set of artificial chromosomes is generated, all values of the matrix return to zero for re-voting process. The same approach will be integrated with SPGA and applied to solve multiobjective scheduling problems.

3.1.2. Weighting mining gene structures (WMGS)

In the MGS proposed by Chang et al. (2005a), only the elite solutions have the vote power, and the power of each elite solution is equally effective. If this voting model is applied to SPGA, there will have two problems:

1. The elite solutions of SPGA belong to non-dominated solutions of its group and when the number of solutions is not sufficient, the information collected is less complete.
2. The original votes of each elite solution have the same effectiveness, which makes it hard to distinguish which one is superior/inferior.

Based on above two problems, the research proposed a weighting mining gene structures method to overcome this situation. This method differs from the original MGS in that it can give different weights according to the goodness or badness of each solution; that is, each voting solution has different effectiveness; information provided by a better solution has relatively higher value for reference, which will be assigned higher weights. Hence, more solutions can be accepted for voting. The procedure of WMGS is very similar to MGS introduced in Section 2.4; the only difference is in step 2 and it will be substituted by weighting vote, and the calculation of weighting vote is as follows:

$$M_{j,\mathrm{Pos}} = M_{j,\mathrm{Pos}} + (w_{n1} \cdot f(x_1) + w_{n2} \cdot f(x_2)) \quad (3.1)$$

$M_{j,\mathrm{Pos}}$ the voting number of the chromosome
$w_{n1} \cdot f(x_1)$ the fitness function value of the normalization of objective 1
$w_{n2} \cdot f(x_2)$ the fitness function value of the normalization of objective 2

3.1.3. Threshold mining gene structures (TMGS)

When creating the artificial solutions by WMGS, normally the processing order of each objective can be ensured; therefore, the sorting of the produced chromosome should be exactly identical, which causes a waste of searching resources. Thus, a threshold value is set up to limit the number of times each gene showing up in each position and the rest can be assigned in a random way. A variety of artificial chromosomes combination can be obtained by this way. The threshold setting has the following two advantages:

1. To avoid the waste of searching resources.
2. To have similar effectiveness to area searching.

A simple example is given to explain the procedure of TMGS. The voting result after the weights being assigned is shown in Fig. 3.1. It is supposed that the threshold value

The result of considering the threshold effect

Job \ Production Sequence	1	2	3	4	5	6
1	0.1085	3.1427	3.8773	1.7340	1.1773	1
2	1	0.1085	3.1427	2.5957	1.7340	2.4588
3	2.0874	1.4184	0.1085	1.3036	2.1552	3.9665
4	3.0485	1.9101	2.0012	1.2637	1.3978	1.4184
5	2.8163	4.4600	0.9101	2.7448	0.1085	0
6	1.9791	1.3036	1	1.3978	3.1633	2.1959

Fig. 3.1. Voting results (by having considered the threshold effect).

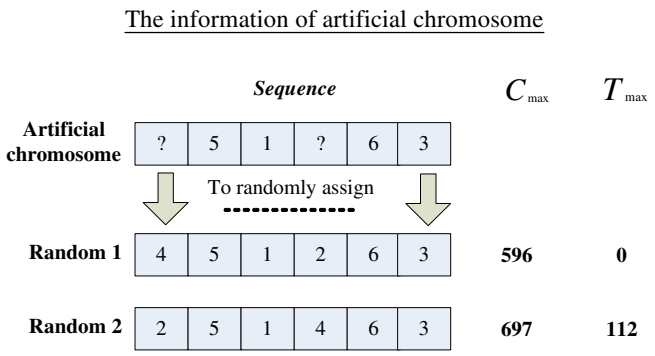

Fig. 3.2. Generation of artificial chromosomes.

is set as 2, which means the weight assignment stops when the unsorted sequence is smaller than or equals 2. The same assignment procedure is carried out for the sorting of manufacturing sequences until the completion of all jobs to be sorted or the threshold value to be achieved. Hence the sorting result is: $? \rightarrow 5 \rightarrow 1 \rightarrow ? \rightarrow 6 \rightarrow 3$, with two positions left for random sorting. Fig. 3.2 shows two possibilities of the sorting information released from artificial chromosomes.

3.2. Mining gene on sub-population genetic algorithm (MGSPGA)

3.2.1. Characteristics of MGSPGA

In this algorithm, the original population is divided in several sub-populations to search solutions separately. By assigning different weights, each sub-population can focus on its own searching in a specific small area. This way can help to strengthen the solution diffusivity effectively.

1. Multiple objectives are transformed into a single objective function by using weighting model. The solution space can be divided into several small areas according to the assigned weights. The weighting combination of each sub-population is (w_{n1}, w_{n2}), in which n stands for the ordinal number of sub-populations. Besides, the concept of SPEA-II is used to record the non-dominated solutions appearing in the searching process in an outside storage file and these solutions later become the elite solutions by elitist mechanism.
2. Mining gene structure is used to extract useful information from chromosomes. In the process of mutation, a set of artificial chromosomes is made to mutate simultaneously. Normally artificial chromosomes will have good sorting results, which is beneficial for GAs to search better solution spaces and find better solutions. The artificial chromosomes can help to strengthen the solution convergence effectively.
3. After a certain number of generations, sub-populations are enforced to converge; the sub-populations can replace their parent population only if they are superior to the original population.

3.2.2. The procedure of MGSPGA

The procedure of MGSPGA is depicted as shown in Fig. 3.3. The parameters of the algorithm N, ns, n, and iteration are number of chromosomes, number of sub-populations, number of individuals in each sub-population, and number of iterations (number of solutions should be examined/ns) respectively. The methods for selection, crossover, mutation, objective function calculation, fitness assignment, and weight assignment are the same as defined by Chang et al. (2005b), which are binary tournament, two points crossover, shift mutation, total makespan and total tardiness time, and scalarized weight assignment respectively. Voting is to record the solution information in the dominance matrix. There are three methods included in fabrication operator: MGS, WMGS and TMGS. The detailed procedure is shown as in Section 2.4. The encoding techniques of chromosomes are sequential type for the scheduling problems. The procedure of the MGSPGA is explained as the following:

(N_s is the number of sub-populations; g is the number of generations; k is the interval number of artificial chromosomes' generations.)

```
1.   Initialize ()
2.   DividePopulation()
3.   AssignWeightToEachObjectives()
4.   counter ← 0
5.   while counter < Iteration do
6.     for i = 1 to ns do
7.       FindPareto(i)
8.       Fitness(i)
9.       Elitism(i)
10.      Voting(i)
11.      Selection(i)
12.      Fabrication(i)
13.      Crossover(i)
14.      Mutation(i)
15.      Replacement(i)
16.    end for
17.    counter ← counter + 1
18.  end while
19.  counter ← 0
```

Compared with SPGA, this approach is different in that it has the mechanism of creating artificial chromosomes and the sorting information of chromosomes in each mutation is recorded for the use of creating artificial chromosomes and placing them in the mating pool for evolution.

3.3. Performance measure

3.3.1. By $D1_R$ value

$D1_R$ is a metric which considers the convergence and diversity simultaneously (Knowles & Corne, 2002) and it is adopted in this research to evaluate the solution quality. Its main concept is to calculate the shortest distance

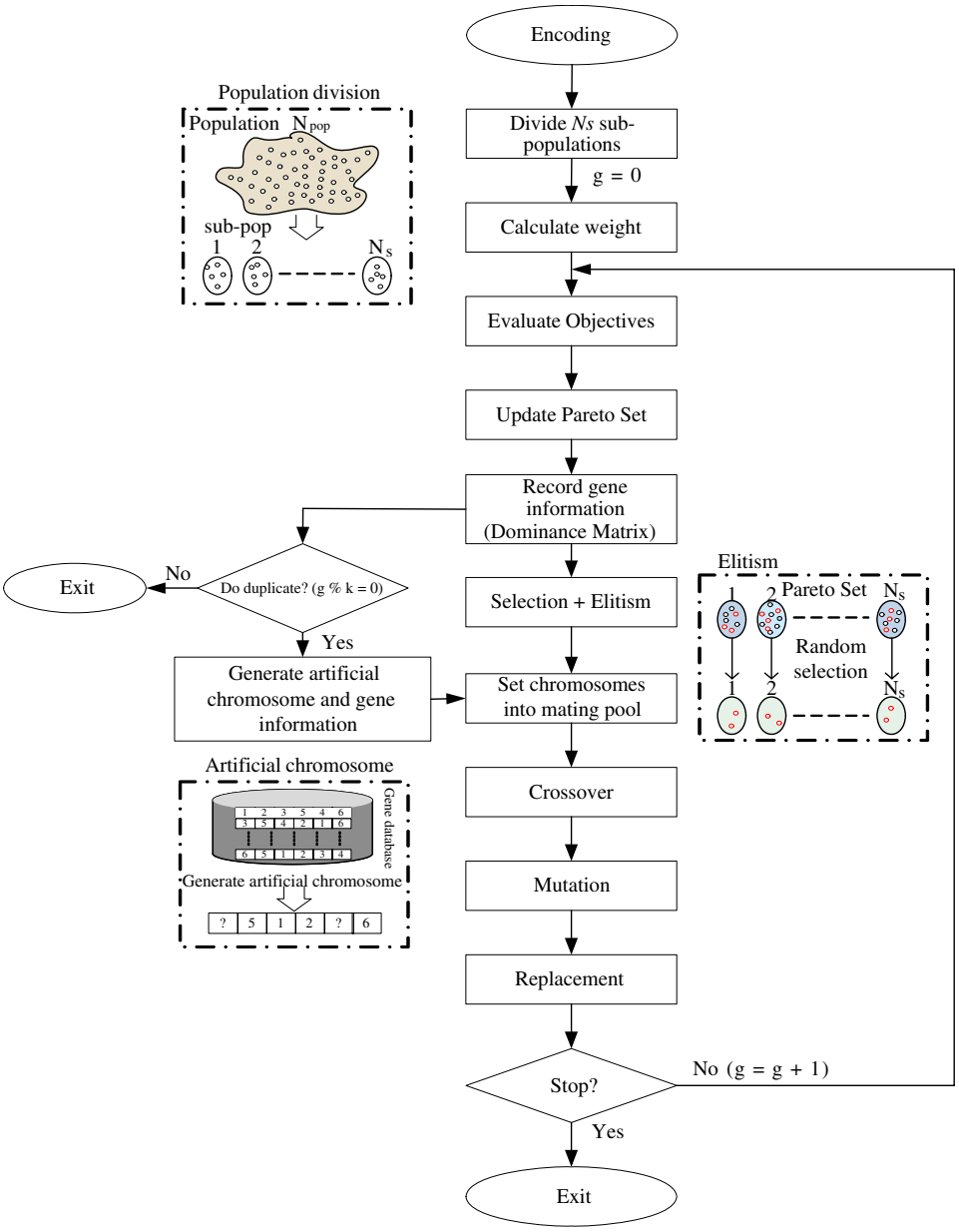

Fig. 3.3. The algorithm procedure of MGSPGA.

between each solution in the Pareto-Solution set and the set to be compared with, and then calculate the mean value. The smaller $D1_R$ is the better. The procedure is explained as follows:

A_j is the number of the sets to be compared, and the $j = 1, 2, \ldots, J$. Therefore, if there are J sets to be compared, then the $D1_R$ value of each set has to be calculated. Z^* is the referred or real set of the Pareto Solution. The formula of $D1_R$ is as follows:

$$D1_R(A_j) = \frac{1}{|Z^*|} \sum_{y \in Z^*} \min\{d_{xy} | x \in A_j\} \tag{3.2}$$

d_{xy} is the distance of each solution in set A_j and the formula is

$$d_{xy} = \sqrt{(f_1^*(y) - f_1(x)) + \cdots + (f_N^*(y) - f_N(x))} \tag{3.3}$$

In formula (3.3), N is the function number of the objectives; in this research, $N = 2$. The distance is not computed directly from the calculation of the objective value; instead, d_{xy} is computed indirectly after normalization. The normalization procedure is to convert all objective values between 0 and 100.

3.3.2. Non-dominated solution numbers and completion time

When comparing the advantages and disadvantages of different algorithms, this research also compares the number of the final non-dominated solutions of each algorithm. The completion time of each method is provided as well.

The method with the shortest completion time to find the non-dominated solution number is the best because if the complexity of a case becomes higher, saving the time for decision making can help the decision maker to respond the current problems quickly.

4. Experimental results and analysis

The research uses the flowshop scheduling case study by Ishibuchi, Yoshida, and Murata (2003) in which four types were included in the bi-objective flow-shop problems; they were 20 jobs in 20 machines, 40 jobs in 20 machines, 60 jobs in 20 machines and 80 jobs in 20 machines. The processing and completion time are the same as used in Ishibuchi et al.'s (2003). The experimental results will be compared with those of SPGA, NSGA-II and SPEA-II.

4.1. Parameter design

4.1.1. Common parameters

The approach proposed in this research and those to be compared have certain common parameters, for example, the number of original population, crossover rate, mutation rate and termination condition, etc. After simple experimental design, the setting of common parameters is listed as shown in Table 4.1.

4.1.2. Individual parameters

MGSPGA differs from SPGA in that it has two more parameters to be considered. In MGSPGA, because of the added MGS, the number of created artificial chromosomes and interval generation number of the solutions generated have to be set. Besides, when using TMGS, one more threshold value has to be set, this is set to be 4 after experiments. Three factors are tested for the experimental design: four processing levels in the number of artificial chromosomes, three processing levels in the interval generation number and three approaches of MGS. The combination sample problems of four jobs and the number of machines are also tested. The processing level of each factor is shown in Table 4.2.

After combining the 3-genes MGS approach with SPGA to be MGSPGA, the parameter combination $3 \times 4 \times 3 = 36$ is used for experiments and we find that only $D1_R$ has significant impact on MGSPGA (see Table 4.3). According to Table 4.1, we can find that the effectiveness of TMGS is the best, WMGS the second, the MGS the last. According to Table 4.2, there is no such reciprocal function between these three factors. Since both interval generation number and artificial chromosome number have no significant impacts on the experimental results, it is no need to set them. Thus, in the following tests, the parameters are set as 20 artificial chromosomes and 40 interval generations. TMGS, the approach with the better effectiveness, is used to be compared with the rest algorithms. According to the main effect plot at Fig. 4.1 and the interaction plot at Fig. 4.2, the approach with the better effectiveness is applied for the comparisons with other algorithms.

4.2. Experimental results

When testing flowshop scheduling problems, the scheduling performance measure index is minimum with maximum completion time and maximum with minimum completion time. Four different types are tested. Among them, the termination condition for different algorithms is the same: when the total searching number equals one million, it stops searching. The following is the results of 30 times testing.

4.2.1. 20 Jobs in 20 machines

The testing result of the sample problem of 20 jobs in 20 machines is depicted in Table 4.4. For the column of $D1_R$, Min is the minimum value after 30 times testing under the parameter combination set as mentioned above; on the other hand, Max is the maximum value; Avg is the average

Table 4.1
Parameter setting

Parameter	Value
Population size	200
Crossover rate	0.9
Mutation rate	0.6
Number of sub-populations	20
Stop criterion	To evaluate 1,000,000 solutions

Table 4.3
The ANOVA result of the experiment

Source	DF	Seq SS	Adj SS	Adj MS	*F*	*P*
Int	2	87.1	87.1	43.6	0.33	0.717
Qua	3	7.2	7.2	2.4	0.02	0.997
Met	2	6892.4	6892.4	3446.2	26.37	**0.000**
Int * Qua	6	58.9	58.9	9.8	0.08	0.998
Int * Met	4	394.9	394.9	98.7	0.76	0.557
Qua * Met	6	21.6	21.6	3.6	0.03	1.000
Int * Qua * Met	12	38.2	38.2	3.2	0.02	1.000
Error	105	13,719.9	13,719.9	130.7		
Total	143	366,308.1				

Table 4.2
The experimental factors

Factor	Treatments			
The number of generation to produce artificial chromosome (Int)	20	30	40	
The number of chromosome to be generated (Qua)	15	20	25	30
Mining gene methods (Met)	1:MGS	2:WMGS	3:TMGS	

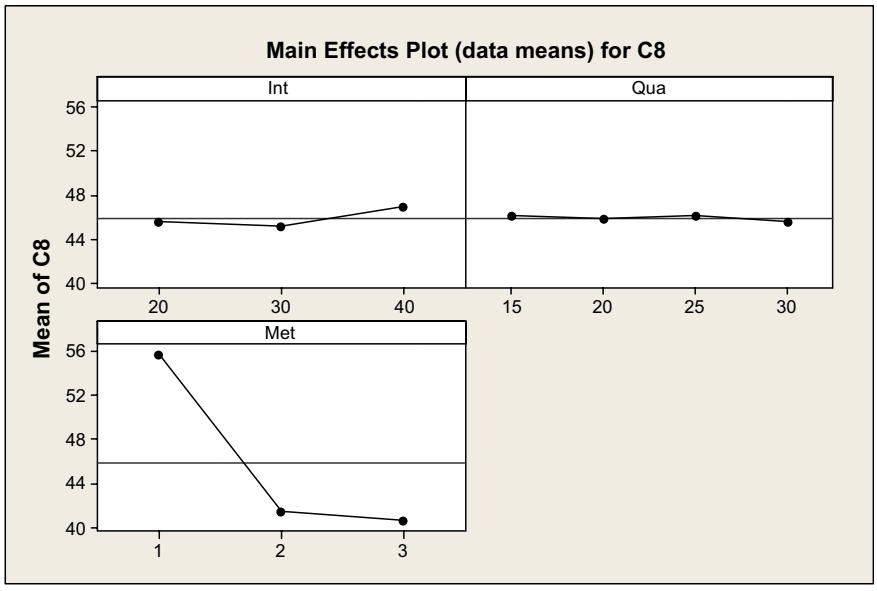

Fig. 4.1. The main effect plot of the three factors.

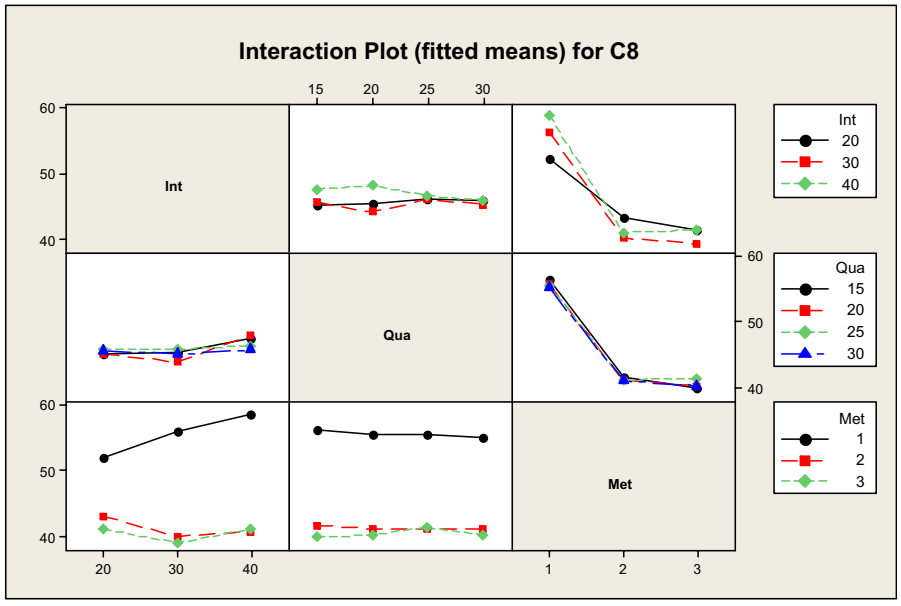

Fig. 4.2. The interaction plot of the three factors.

value of these 30-time tests. From Table 4.4, it is obvious that MGSPGA has better $D1_R$ value than other approaches in general, and has more average non-dominated solution numbers. Its completion time is slightly

Table 4.4
The algorithm comparison of 20 jobs and 20 machines flowshop problem

Algorithm	$D1_R$				Number	Sec.
	Min	Avg	Max	Std		
NSGA-II	11.5	13.5	15.1	2.0	20.5	33.4
SPEA-II	9.4	14.4	20.6	2.6	19.8	87.6
SPGA	4.8	6.3	8.3	0.8	24.4	16.5
MGSPGA	3.9	5.8	7.1	0.8	26.2	16.9

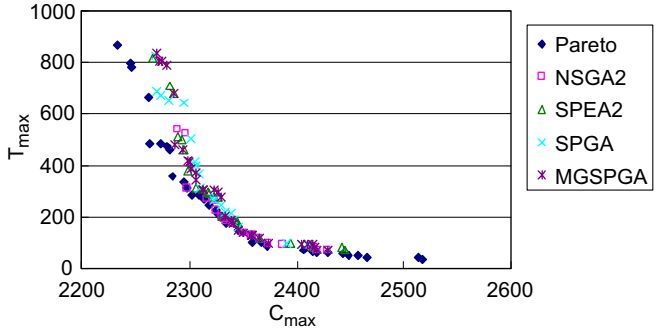

Fig. 4.3. The plot of algorithms with reference Pareto set of 20 jobs and 20 machines.

slower than SPGA but much faster than NSGA-II and SPEA-II. Fig. 4.3 presents the Pareto set solved by these three algorithms.

4.2.2. 40 Jobs in 20 machines

The testing result of this sample problem is depicted in Table 4.5. From Table 4.5, it is obvious that MGSPGA has the best performance of all. Fig. 4.4 illustrates the Pareto set solved by these three algorithms.

4.2.3. 60 Jobs in 20 machines

The testing result of this sample problem is depicted in Table 4.6. This sample is more complex so the results generated from each algorithm are not very close to the referred Pareto optimal solutions. However, in comparison, MGSPGA still has better performance than other approaches. The Pareto set is depicted at Fig. 4.5.

4.2.4. 80 Jobs in 20 machines

The testing result of this sample problem is depicted in Table 4.7. From the numbers shown in Table 4.7, it can be found that MGSPGA has great improvement effectiveness. The Pareto set solved by the three algorithms is shown at Fig. 4.6.

Table 4.5
The algorithm comparison of 40 jobs and 20 machines flowshop problem

Algorithm	$D1_R$				Number	Sec.
	Min	Avg	Max	Std		
NSGA-II	28.2	30.2	32.3	2.6	21.0	54.6
SPEA-II	25.1	28.8	32.5	4.5	21.0	102.5
SPGA	21.5	26.6	32.8	2.7	25.3	31.1
MGSPGA	14.7	18.7	24.7	2.4	22.6	31.7

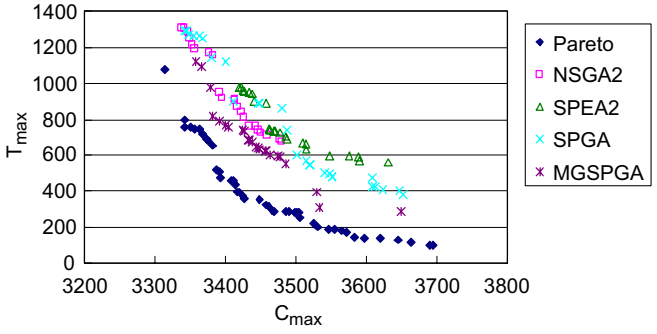

Fig. 4.4. The plot of algorithms with reference Pareto set of 40 jobs and 20 machines.

Table 4.6
The algorithm comparison of 60 jobs and 20 machines flowshop problem

Algorithm	$D1_R$				Number	Sec.
	Min	Avg	Max	Std		
NSGA-II	30.3	31.6	32.8	1.8	18.5	93.0
SPEA-II	28.6	31.1	32.1	2.4	19.0	113.8
SPGA	27.2	31.3	34.4	1.6	22.7	46.7
MGSPGA	18.5	22.4	25.6	1.9	20.4	47.8

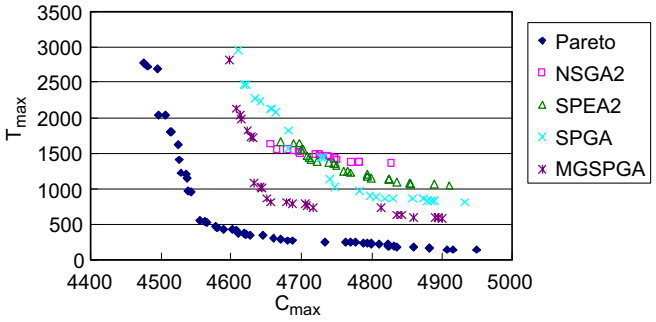

Fig. 4.5. The plot of algorithms with reference Pareto set of 60 jobs and 20 machines.

Table 4.7
The algorithm comparison of 80 jobs and 20 machines flowshop problem

Algorithm	$D1_R$				Number	Sec.
	Min	Avg	Max	Std		
NSGA-II	198.5	209.1	219.8	15.2	10.9	104.1
SPEA-II	188.5	199.5	210.5	13.8	15.5	118.9
SPGA	156.0	202.6	227.3	16.6	23.1	64.8
MGSPGA	84.0	122.3	173.4	25.2	14.1	66.3

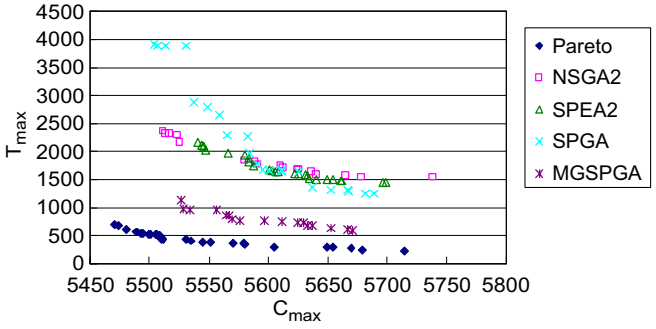

Fig. 4.6. The plot of algorithms with reference Pareto set of 80 jobs and 20 machines.

According to the above-mentioned four testing results, we find that when solving more complex problems, it is harder to find the improving effectiveness of MGSPGA. Taking the sample of 20 jobs in 20 machines as an example, although both SPGA and MGSPGA achieve the referred Pareto solutions, the improving performance is not very significant. Along with the increasing number of jobs, the problems become more complex and thus the improving effectiveness of MGSPGA can be obviously noticed.

5. Conclusions

Through this study, we can verify that by combining MGS with SPGA, multiobjective scheduling problems can be solved more effectively. In the future, MGSPGA can be further extended to three objectives or multidimensional continuous problems. The concept of sub-populations can be further embedded in local searching procedure to improve the solution quality of the algorithm.

Further investigation will be carried out to examine whether it is possible to generate elite chromosomes through better mining algorithms. It is also suggested that different objectives of flowshop scheduling problems can be further tested such as the minimization of the sum of job completion time, and those with more complex requirements such as sequence dependent setup times.

References

Affenzeller, M. (2002). New generic hybrids based upon genetic algorithms. *Lecture Notes in Computer Science, 2527*, 329–339.

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.

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

Cochran, J. K., Horng, S. M., & Fowler, J. W. (2003). A multi-population genetic algorithm to solve multi-objective scheduling problems for parallel machines. *Computers and Operations Research, 30*, 1087–1102.

Coello, C. A., Pulido, G. T., & Lechuga, M. S. (2004). Handling multiple objectives with particle swarm optimization. *IEEE Transactions on Evolutionary Computation, 8*(3), 256–279.

Davis, L. (1991). *Handbook of genetic algorithm*. New York: Van Nostrand Reinhold.

Deb, K., Amrit, P., Sameer, A., & Meyarivan, T. (2002). A fast and elitist multi-objective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation, 6*(2), 182–197.

Fox, B. R., & McMahon, M. B. (1991). Genetic operators for sequencing problems. In G. J. E. Raolins (Ed.), *Foundations of genetic algorithms* (pp. 284–300). San Marco: Morgan Kaufman Publishers.

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

Holland, J. H. (1975). *Adoption in natural and artificial systems*. Ann Arbor: The University of Michigan Press.

Hu, J., Goodman, E., Seo, K., Fan, Z., & Rosenberg, R. (2005). The hierarchical fair competition framework for sustainable evolutionary algorithms. *Evolutionary Computation, 13*(2), 241–277.

Ishibuchi, H., Yamamoto, N., Murata, T., & Tanaka, H. (1994). Genetic algorithms and neighbourhood search algorithms for fuzzy & flowshop scheduling problems. *Fuzzy Sets and Systems, 67*, 81–100.

Ishibuchi, H., Yoshida, T., & Murata, T. (2003). Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling. *IEEE Transactions on Evolutionary Computation, 7*(2), 204–223.

Jog, J., Suh, Y., & Gucht, D. V. (1989). The effects of population size, heuristic crossover and local improvement on a genetic algorithm for the travelling salesman problem. In *Proceedings of the Third IGCA*, San Mateo (pp. 110–115).

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)* (pp. 711–716). New York: IEEE Press.

Lis, J., & Eiben, A. E. (1997). A multisexual genetic algorithm for multicriteria optimization. In *Proceedings of the 4th IEEE conference on evolutionary computation* (pp. 59–64).

Mostaghim, S., & Teich, J. (2004). Covering Pareto-optimal fronts by subswarms in multi-objective particle swarm optimization. *Evolutionary Computation, 2*, 1404–1411.

Ogbu, F. A., & Smith, D. K. (1990). The application of the simulated annealing algorithm to the solution of the $n/m/C_{\max}$ flowshop problem. *Computers and Operations Research, 17*, 243–253.

Osman, I. H., & Potts, C. N. (1989). Simulated annealing for permutation flow-shop scheduling. *OMEGA, 17*, 551–557.

Reeves, C. R. (1995). Genetic algorithm for flowshop sequencing. *Computers and Operations Research, 15*, 5–23.

Schaffer, J. D. (1985). Multiple objective optimizations with vector evaluated genetic algorithms. In *Proceedings of 1st international conference on genetic algorithms* (pp. 93–100).

Starkweather, T., McDaniel, S., Whitley, C., Mathias, K., & Whitley, D. (1991). A comparison of genetic sequencing operators. In *Proceedings of Fourth IGCA* (pp. 69–76).

Taillard, E. (1991). Some efficient heuristics methods for flow-shop sequencing problem. *European Journal of Operations Research, 47*, 65–74.

Ulder, N. L. J. (1991). Genetic local search algorithms for the travelling salesman problem. In H. P. Schwefel & R. Manner (Eds.), *Parallel problem solving for nature* (pp. 109–116). Berlin: Springer.

Zitzler, E., Laumanns, M., & Thiele, L. (2001). SPEA2: Improving the strength Pareto evolutionary algorithm. Technical Report 103, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich, Gloriastrasse 35, CH-8092 Zurich, Switzerland.

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

Zitzler, E., & Thiele, L. (1999). Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. *IEEE Transactions on Evolutionary Computation, 3*(4), 257–271.