A New Multiobjective Evolutionary Algorithm

Ruhul Sarker, Ko-Hsin Liang and Charles Newton
School of Computer Science
The University of New South Wales
ADFA Campus, Canberra, ACT 2600, Australia
Email: {ruhul, liangk, csn}@cs.adfa.edu.au

Abstract

The Pareto-based approaches have shown some success in designing multiobjective evolutionary algorithms. Their methods of fitness assignment are mainly from the information of dominated and nondominated individuals. On the top of the hierarchy of multiobjective evolutionary algorithms, the Strength Pareto Evolutionary Algorithm (SPEA) has been elaborately designed with this principle in mind. In this paper, we propose a $(\mu+\lambda)$ multiobjective evolutionary algorithm ($(\mu+\lambda)$MEA), which discards the dominated individuals in each generation. The comparisons of the experimental results demonstrate that the $(\mu+\lambda)$MEA outperforms SPEA on five benchmark functions with less computational efforts.

1 Introduction

In multiobjective optimization problems, we define the vector-valued objective function $\mathbf{f}: R^n \to R^m$ where $m > 1$, n is the dimension of the decision vector, $\mathbf{x}$, and m is the dimension of the objective vector, $\mathbf{y}$. For the case of minimization problems, we intend to minimize

$$\mathbf{y} = \mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), \cdots, f_m(\mathbf{x})),$$

where $\mathbf{x} \in R^n$, and $\mathbf{y} \in R^m$. A solution $\mathbf{y_a}$ is said to dominate solution $\mathbf{y_b}$ if $y_{ai} \leq y_{bi}$, $\forall i \in \{1, \cdots, m\}$ and $y_{ai} < y_{bi}$, $\exists i \in \{1, \cdots, m\}$. In most cases, the objective functions are in conflict, because in order to decrease any of the objective functions, we need to increase other objective functions. When we have a solution which is not dominated by any other solution in the feasible space, we call it Pareto-optimal. The set of all Pareto-optimal solutions is also termed the Pareto-optimal set, efficient set or admissible set. Their corresponding vectors are called the nondominated set.

In [1], Fonseca and Fleming categorized the evolutionary based techniques for multiobjective optimization into three approaches, plain aggregating approaches, population-based non-Pareto approaches, and Pareto-based approaches. Recently, a similar but more comprehensive survey was also presented in [2] by Coello.

Scalarizing the objective vector into a single objective (also called the aggregating function) has the advantage of obtaining a single compromise solution when applying any traditional optimizers or evolutionary algorithms (EAs). However, these approaches cannot deal properly with non-convex Pareto fronts [3]. In addition, the single solution may not satisfy the decision makers, and the importance of each objective function must be known prior to setting the proper weights for each objective function. Hence solving the multiple objectives concurrently is a favorable option to most decision makers. This demand fits with the EA's characteristic of population. Using evolutionary based approaches to solve multiobjective problems has the merit of generating multiple solutions in one run.

The early work exploiting the population of EAs to search for multiple nondominated solutions can be dated back to Schaffer's Vector Evaluated Genetic Algorithm (VEGA) [4]. To properly handle every objective function is the main difficulty of the population-based non-pareto approaches. VEGA uses m subpopulations to optimize each objective independently using proportional selection. These subpopulations, then, are shuffled together to perform crossover and mutation operators. A similar approach was developed by Kursawe [5]. His multiobjective version of evolution strategies (ESs) applies a probability vector to determine the objective in one of the selection steps.

The Pareto-based approaches basically use the objective functions to distinguish between the nondominated and dominated solutions in the current population. The fitness assignment of an individual is based on the information from both the nondominated and dominated sets. Goldberg [6] was the first to suggest a nondominated ranking procedure to decide the fitness of the individuals. Based on his idea, Srinivas and Deb [7] developed the Nondominated Sorting Genetic Algorithms (NSGA). The nondominated individuals are removed layer by layer from the population. Before removing them, the different ranks are assigned to each layer. The individuals in the first layer would have the best fitness.

A slightly different scheme was proposed in Fonseca and Fleming's multiobjective EA (FFEA) [8]. An individual's rank is determined by the number of individuals dominating it. Without using any nondominated ranking methods, the Niched Pareto Genetic Algorithm (NPGA) proposed by Horn, Nafpliotis and Goldberg [9] directly use a set of randomly picked individuals from the current population and place the best of this subset in the next population. Different from binary tournament selection, the fitness of the two randomly selected individuals is decided according to whether they are dominated by any of the individuals from the comparison set(a set of individuals is also picked from the population). If one individual is dominated by the comparison set, and the other is not, the later is selected for reproduction. If neither or both are dominated by the comparison set, a niched method [10] is used for selection.

A common feature of the Pareto-based approaches mentioned above is that the Pareto-optimal solutions in each generation are assigned the same fitness or rank. Some sharing or niche techniques have to be applied in the selection procedure. Recently, Zitzler and Thiele [11] proposed a Pareto-based method, the Strength Pareto Evolutionary Algorithm (SPEA), which elaborately assigns different fitness to all the individuals including the nondominated ones. The detailed characteristics of this algorithm are discussed in a later section. The comparisons among SPEA and other evolutionary based algorithms were made by solving 9 multiobjective 0/1 knapsack problems [12] and 6 test functions [13] constructed by following Deb's guidelines [14]. In these comparisons, SPEA clearly outperforms the other multiobjective EAs.

Most recently, Knowles and Corne [15, 16] proposed a simple Evolution Strategies, (1+1)-ES, known as PAES (Pareto Archived Evolution Strategy), that keeps a record of limited nondominated individuals. The nondominated individuals are accepted for recording based on the degree of crowdiness in their grid (defined regions on the Pareto frontier) location to ensure diversity of individuals in the final solution. The algorithm is strictly confined to local search i.e. it should use a small change (mutation) operator only, and move from a current solution to a nearby neighbour. As they reported, the algorithm works well, specially for the problems of low computational complexity. They also proposed an extension to this basic approach, which took them to some variants of a $(\mu + \lambda)$ -ES. The performance of the algorithm is judged, by solving several test problems, and analyzing the results using a statistical comparison technique. $PEAS$ was also compared with $SPEA$ using the same representation and operators as $SPEA$ on all the test functions.

A Multiobjective Evolution Strategy (MOBES) was developed by Binh and Korn [17, 18] for solving constrained multiobjective optimization problems. In this method, the decision variables, strategy parameter vector and the objective function vector are considered as the key elements. The detail of this method is not reported here as we are interested in unconstrained problems only.

In this paper, we develop a Pareto-based $(\mu + \lambda)$ multiobjective evolutionary algorithm where the values of μ vary from generation to generation and λ is equal to μ times a fixed ratio. In each generation, all dominated individuals are discarded. Offspring are only produced by nondominated individuals, stored in an external memory, using recombination and mutation operators. The algorithm is compared to SPEA on five benchmark test functions used in [13]. We use the concept of 'coverage metric', as stated in Zitler and Thiele [12], to judge the performance of our algorithm. The experimental results clearly demonstrate that the $(\mu + \lambda)$ multiobjective evolutionary algorithm outperforms SPEA on all test functions.

The rest of this paper is organized as follows. Section 2 describes the $(\mu + \lambda)$ multiobjective evolutionary algorithm in detail. The empirical results and discussions are presented in section 3. The performance analysis is provided in section 4. Finally, section 5 concludes the paper with some remarks and future research directions.

2 The $(\mu + \lambda)$ Multiobjective Evolutionary Algorithm

The $(\mu + \lambda)$ multiobjective evolutionary algorithm, or for short: $(\mu + \lambda)$MEA, has a very different design from other Pareto-based approaches. It does not use any information from the dominated individuals. In each generation, we only preserve nondominated individuals. The number of generated offspring is the number of the nondominated individuals multiplied by a fixed ratio. Hence, the population size in each generation could be different. However, we limit the number of the nondominated individuals using a nearest neighbour distance function, which can help to disperse the nondominated individuals.

The $(\mu + \lambda)$ multiobjective evolutionary algorithm is described as follows.

Step 1: Generate μ_{init} individuals randomly.

Step 2: Collect non-dominated solutions from the population. The number of non-dominated solutions is μ_{nd}.

Step 3: If $\mu_{nd} > \mu_{max}$, select μ_{max} individuals using the nearest neighbour distance function, and then $\mu_{nd} = \mu_{max}$.

Step 4: Generate $\mu_{nd} \times \nu$ offspring using discrete recombination and Gaussian mutation.

Step 5: Go to Step 2 if the termination criteria are not met.

This algorithm uses the concept of the $(\mu + \lambda)$-ES [19]. However, μ is the number of non-dominated individuals, μ_{nd}, which varies from generation to generation, and λ is equal to $\mu_{nd} \times \nu$, where ν is the fixed ratio of offspring and parents in each generation.

Instead of the binary string implementation normally found in GAs, we use the real number representation in our algorithm. Since many problems in the real world are expressed in real variables, faster computation can be obtained without conversions between different representations. More discussions on binary versus real coding are provided in section 4.

In Step 3, we control the number of non-dominated individuals so as not to exceed a maximum number, μ_{max}. Hence the generated offspring are under control. To filter better individuals from the non-dominated set, we evenly distribute the individuals on the Pareto front. Using the nearest neighbour distance function the individuals crowds together resulting in a lower fitness value thereby reducing the probability of selection to the next generation. So an individual far away from any other individual will have a better chance of survival. The nearest neighbour distance function is defined as follows.

$$D(\mathbf{x}) = \frac{\min \sum_{j=1}^{m} \{\|\mathbf{x} - \mathbf{x}_j\| : \mathbf{x}, \mathbf{x}_j \in \{\mathbf{x}_1, \cdots, \mathbf{x}_{\mu_{nd}}\}\}}{m},$$

where $\mathbf{x} \neq \mathbf{x}_j$. In the case of two objective problems, the above function can be simplified as

$$D(\mathbf{x}) = (\min \|\mathbf{x} - \mathbf{x}_i\| + \min \|\mathbf{x} - \mathbf{x}_j\|)/2,$$

defining the nearest neighbour distance to be the average of the distance from $\mathbf{x}$ to the two closest individuals $\mathbf{x}_i$ and $\mathbf{x}_j$, where $\mathbf{x}_i \neq \mathbf{x}_j$, and $\mathbf{x}, \mathbf{x}_i, \mathbf{x}_j \in \{\mathbf{x}_1, \cdots, \mathbf{x}_{\mu_{nd}}\}$. This rule will allow n or less closely grouped individuals to stay in the population, and expel those more than n.

The discrete recombination is most often used in ESs [20, 21]. The discrete recombination has also produced good results with real-coded GAs (such as Breeder GA) [22]. The implementation is that each new individual's objective variable is randomly decided from one of the two pre-selected parents. If parent-1 is $(x_1, \cdots, x_n)$ and parent-2 is $(y_1, \cdots, y_n)$. We generate offspring $(z_1, \cdots, z_n)$, where $z_i = \{x_i\}$ or $\{y_i\}$, $\forall i \in \{1, \cdots, n\}$, x_i or y_i are chosen with probability 0.5.

Parts of the mutation operator have been inspired by the breeder genetic algorithm (BGA) [22]. For each objective variable, the probability to perform a mutation is p_m. We use $p_m = 1/n$. The Gaussian mutation is implemented as follows.

$$z_i = x_i + 0.1 \cdot (x_i^U - x_i^L) \cdot N(0, 1),$$

where x_i^U and x_i^L are the upper and lower boundaries of variable x_i, respectively, and $i \in \{1, \cdots, n\}$. $N(0, 1)$ denotes a Gaussian distributed one-dimensional random number with mean 0 and standard deviation 1. The number 0.1 adopted from BGA is regarded as a fixed step size. Using a fixed step size, we can avoid designing self-adaptive step sizes, which may be a difficult job. For multi-objective problems, the adaptive step sizes may not be adapted well enough to fit different objective spaces. This could be an interesting topic for future research.

Although the $(\mu+\lambda)$MEA only keeps a fixed number of nondominated solutions in each generation, it also maintains an external storage to store the nondominated individuals found in the entire optimization run. The size of the external storage is not restricted. The same external storage is applied in SPEA as well. SPEA maintains a main external set used in the evolution process, and a secondary external set used as storage for all the nondominated solutions only. The purpose of keeping all the nondominated solutions is to facilitate the comparisons among EAs by focusing on their offline performance [12].

3 Experimental Results

In our study, five numerical multi-objective optimization problems were applied to test the $(\mu+\lambda)$MEA. These benchmark problems as listed in the Appendix were used by Zitzler, Deb and Thiele [13]. Since the $(\mu+\lambda)$MEA is mainly designed for solving numerical problems, we only tested functions T_1–T_4 and T_6. The test function T_5 is a deceptive problem with bit string representation, which currently is unsuitable to be solved by the $(\mu+\lambda)$MEA.

For the $(\mu+\lambda)$MEA, the initial population size, μ_{init}, is set to 100, which is a common value found in EAs experiments. The maximum number of parents, μ_{max}, is 20, and the ratio of offspring and parents, ν, is 4. Therefore, the maximum possible number of $(\mu+\lambda)$ in each generation will be limited to 100. Each problem is executed 30 times. The algorithm terminates when 10000 function evaluations are used.

To evaluate the performance of the $(\mu+\lambda)$MEA, we compare it with the results of SPEA [1]. We use the same methods of performance measures as applied in [13].

3.1 Graphical Presentation and Coverage Comparisons

It is well accepted that the multiobjective optimization problems require multiple, but evenly distributed, solution points to form a Pareto trade-off front to make them useful to the human decision makers. When comparing two algorithms, these two consequences (number of solution points and their distributions) must be considered in addition to the quality of solutions . However, there are a number of methods available to compare the performance of different algorithms. The *Error Ratio* and the *Generational Distance* are used as the performance measure indicators when the Pareto optimal solutions are known [23]. The *Spread* measuring techniques express the distribution of individuals over the nondominated region [7]. The method is based on the chi-square-like deviation form distribution measure, and it requires several parameters to be estimated before calculating the Spread indicator.

The method of *coverage metrics* is proposed by Zitzler and Theiele for comparing the performances of different multiobjective evolutionary algorithms [12]. The method shows whether the outcomes of one algorithm dominate the outcomes of another algorithm without indicating how much better it is. Most recently, Knowles and Corne [16] proposed a statistical method to compare the performances of two or more algorithms by analyzing the superiority in different regions of the attainment surfaces. However, the concept of $attainmentsurfaces$ and counting the superiority of different regions of the attainment surfaces was first proposed by Fonseca and Fleming [24] without incorporating any proper statistical analysis.

We used two methods: (i) graphical presentation for visual inspection and (ii) the method of coverage metrics to show the quantitative dominance. Using the method of coverage metrics, Zitzler and Theiele claimed that their algorithm is the best among the well known multiobjective evolutionary algorithms [12]. This encourages us to choose the same test problems as Zitzler and Theiele and compare their outcomes with ours using the same comparison technique.

The first method is the graphical presentation. We unify the outcomes of the first five runs of each test function from each algorithm. The dominated solutions are deleted from the union set, and all the nondominated ones are plotted as shown in Figures 1 to 3.

It is clear that the $(\mu+\lambda)$MEA outperforms SPEA on all 5 test functions. For functions T_1 and T_2, the Pareto-optimal solutions of the $(\mu+\lambda)$MEA are closer to the Pareto trade-off surface. Its solution distributions cover almost the whole convex and non-convex Pareto-optimal fronts. The results of function T_3 from the $(\mu+\lambda)$MEA and SPEA are a bit close. To show the discrepancy, we enlarged the region of closest proximity

[1] The test results of SPEA are obtained from "http://www.tik.ee.ethz.ch/∼zitzler/testdata.html" established by Zitzler.

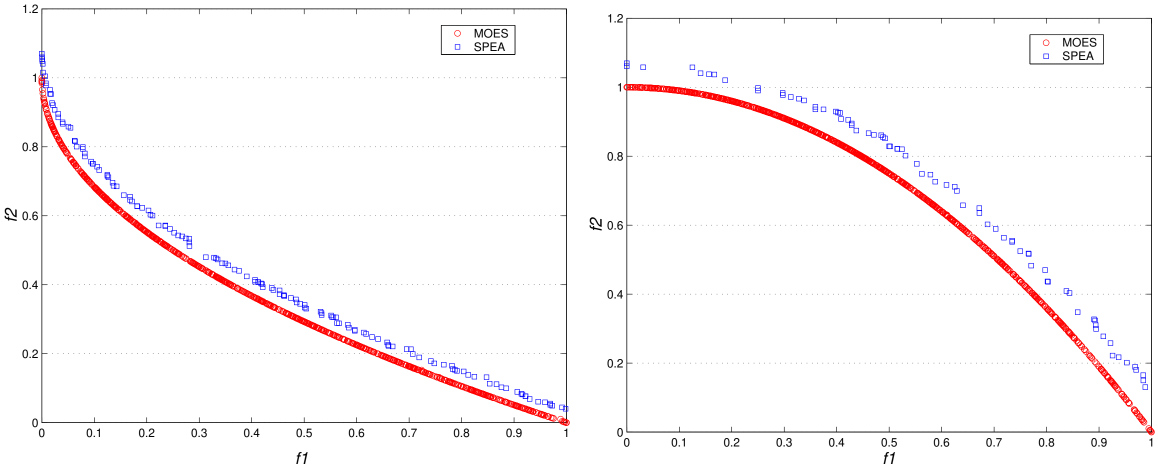

Figure 1: Test function T_1 (left) and test function T_2 (right).

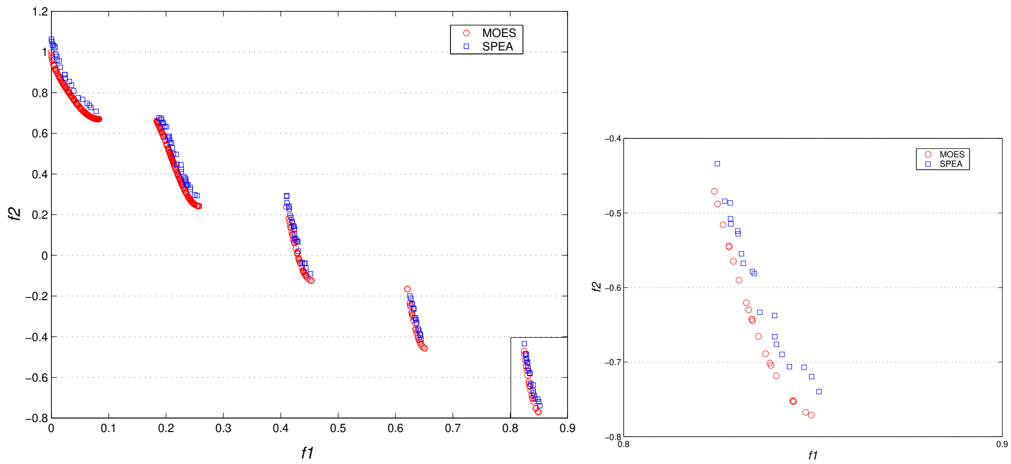

Figure 2: Test function T_3 (left) and enlarged rectangle of test function T_3 from left (right).

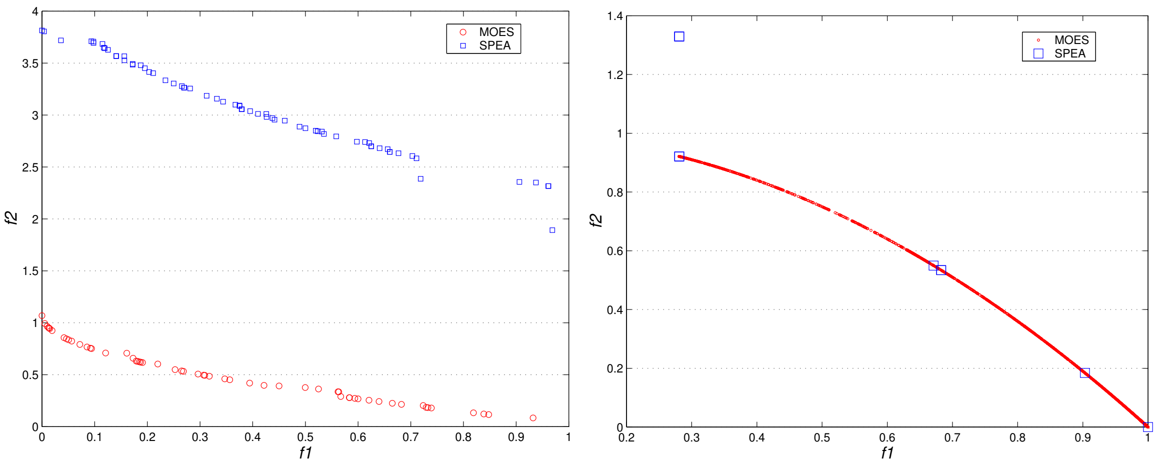

Figure 3: Test function T_4 (left) and test function T_6 (right).

located between 0.8 and 0.9 of f_1 and displayed it on the right of Figure 2. It is clear that the results of the $(\mu+\lambda)$MEA are closer to the Pareto-optimal front with similar solution distribution.

For function T_4, the $(\mu+\lambda)$MEA clearly outperforms SPEA. Since the solutions of function T_6 from both algorithms are very close to the Pareto-optimal front, it is difficult to judge from the quality of the solutions. However, for SPEA, there are only 12 solutions found and 6 points are displayed. That is, they group into 6 different areas. Therefore, the solution distribution of SPEA is not satisfactory.

The second comparison method is to measure the coverage of two solution sets from different algorithms. We say a solution $\mathbf{y_a}$ *covers* solution $\mathbf{y_b}$ when $\mathbf{y_a}$ dominates $\mathbf{y_b}$ or $\mathbf{y_a} = \mathbf{y_b}$. The performance measures are represented by the $\mathcal{C}$ metric used in [12]. $\mathcal{C}(Y', Y'')$ is defined as the number of solutions in Y'' covered by any solutions in Y' divided by the solution number in Y''.

Figure 4 shows the box plots based on the $\mathcal{C}$ metric measured from the results of both algorithms. The box plots for each function are drawn from 30 $\mathcal{C}$ values, which are calculated from the 30 algorithm pairs from 30 runs. The box plots demonstrate that $(\mu+\lambda)$MEA's results covered SPEA except for a few occasions. On the other hand, the answers from SPEA never covered any of the $(\mu+\lambda)$MEA's solutions.

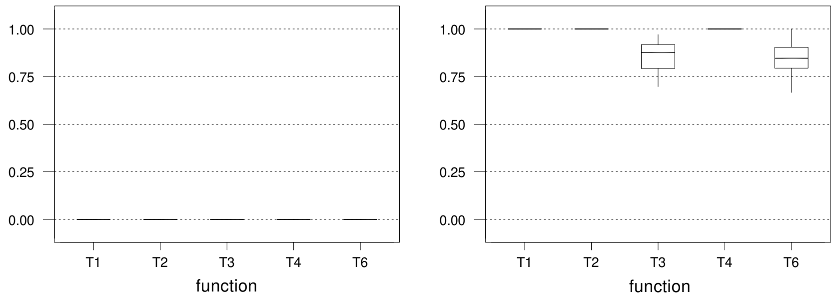

Figure 4: Box plots based on the $\mathcal{C}$ metric from 5 test functions are shown. $\mathcal{C}(Y_{SPEA}, Y_{(\mu+\lambda)MEA})$ is on the left, and $\mathcal{C}(Y_{(\mu+\lambda)MEA}, Y_{SPEA})$ is on the right.

3.2 Discussions

In our experiments, the $(\mu+\lambda)$MEA stopped when 10000 function evaluations were used. For SPEA in [12], the population sizes of 80 and 250 generations were applied, which used 20000 function evaluations to finish one run of the experiment. We used half of the number of function evaluations used in SPEA, since the nondominated individuals converged pretty early to the Pareto trade-off surface.

Figure 5 shows the trace of the nondominated solutions of the $(\mu+\lambda)$MEA on function T_2 in one run. In this example, the program ended after 367 generations had evolved. At the 200-th generation, 948 function evaluations were used, and one of the Pareto-optimal solutions was $(f_1, f_2) = (0, 1.1232)$. This shows that the individuals can evolve to the Pareto trade-off surface rapidly. One possible reason for this faster search speed is that the $(\mu+\lambda)$MEA does not save any dominated individuals, which might reduce the possible waste made from the dominated ones. However, two issues are raised. What benefits could be obtained by keeping dominated individuals in the population? That is, what is the probability of a nondominated individual being derived from a dominated individual? Another issue is how harmful will the rejection of the dominated set be. The above questions could be interesting topics for future research. However, when we watched the processes of the $(\mu+\lambda)$MEA, we often observed that the number of the nondominated individuals was very low in the beginning, sometimes resulting in only one nondominated individual being left in the middle of the process. In the example of Figure 5, the $(1+\lambda)$MEA $(\mu = 1)$ were first appeared in the 186-th generation. The number of nondominated individuals began to increase when the evolved individuals were close to the Pareto trade-off surface.

We may approximately divide the $(\mu+\lambda)$MEA into two search stages, the convergent and divergent stages. The convergent stage makes the individuals to rapidly approach the Pareto front. The divergent stage disperses the individuals and maintains a well distributed population. In the above case, 1/10 and 9/10 of the process were dedicated to the convergent and divergent stages, respectively. On the second stage, the ability of the $(\mu+\lambda)$MEA to find multiple nondominated solutions is mainly driven by the Gaussian mutation. Using the

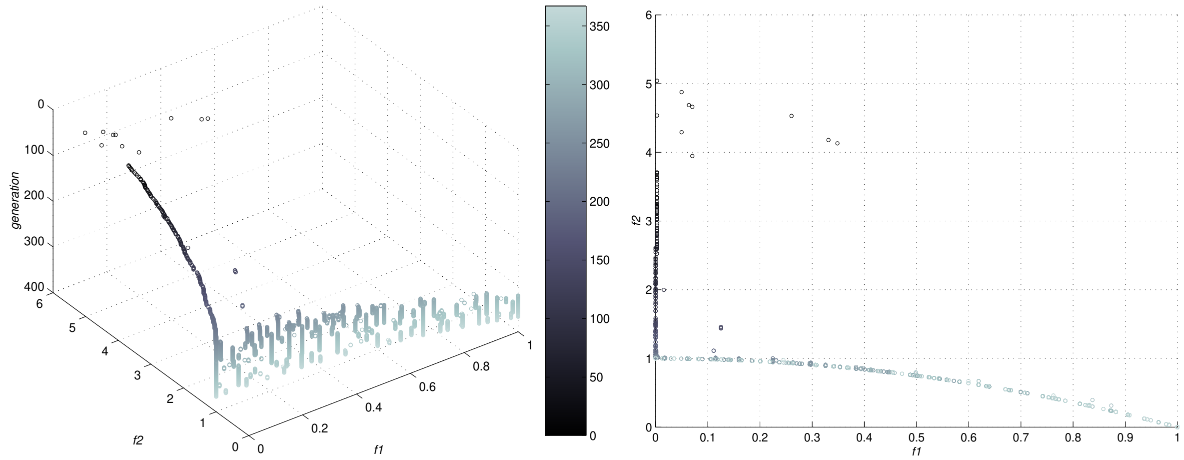

Figure 5: The trace of one run of the $(\mu+\lambda)$MEA on function T_2 is shown, a 3D view on the left and a 2D view on the right. The colour bars note for the number of generations evolved.

continuous search space, the Gaussian mutation has the advantage of recovering good individuals from the near neighborhood.

One possible weakness of the $(\mu+\lambda)$MEA was uncovered in Figure 2. A closer inspection of the five discrete Pareto fronts, showed that the solutions are not evenly distributed. Therefore, the search without dominated individuals may not have enough diversity to cover some solution space, especially when the Pareto-optimal set is discretely located.

4 Performance Analysis

Since $(\mu+\lambda)$MEA performed significantly better than SPEA, it will be beneficial to know what contributes the success of $(\mu+\lambda)$MEA. The differences between $(\mu+\lambda)$MEA and SPEA are described as follows.

Nondominated Solution Limitation: In both algorithms, the number of nondominated solutions in each generation is limited to a given maximum. SPEA uses the clustering methods to prune the exceeding nondominated solutions, and $(\mu+\lambda)$MEA retains preferable nondominated solutions by way of the nearest neighbor distance function.

Evolutionary Operators: The evolutionary operators employed by SPEA and $(\mu+\lambda)$MEA are different. SPEA uses bit string representation, traditional one point crossover, and bit flipping mutation. $(\mu+\lambda)$MEA uses numerical representation, global discrete recombination and Gaussian mutation with fixed step size control.

Parent Selection: Another major difference is the way of generating the parent population for the next population. In SPEA, an intermediate population is formed first by binary tournament selection before performing any operators. An individual's fitness is calculated by the strength of covering other individuals. In $(\mu+\lambda)$MEA, no strength calculation is needed. The nondominated solutions naturally become the parents of the next generation.

The nondominated solution limitation is a mechanism to distribute the solutions evenly. The algorithm's search ability is mainly contributed by the evolutionary operators and the parent selection schemes. To compare the performances of different evolutionary operators, we can refer Bäck and Schwefel's work [20] on comparing GAs, ESs and evolutionary programming (EP). The performances of ESs and EP are both better than GAs, where the evolutionary operators have played a significant role. It seems the operators in $(\mu+\lambda)$MEA provide major roles while outperforming SPEA.

To investigate which part of $(\mu+\lambda)$MEA makes more contribution to the search results, two empirical analysis are conducted. The empirical results will clarify the question: is the better performance of $(\mu+\lambda)$MEA caused by its operators or/and its parent selection schemes?

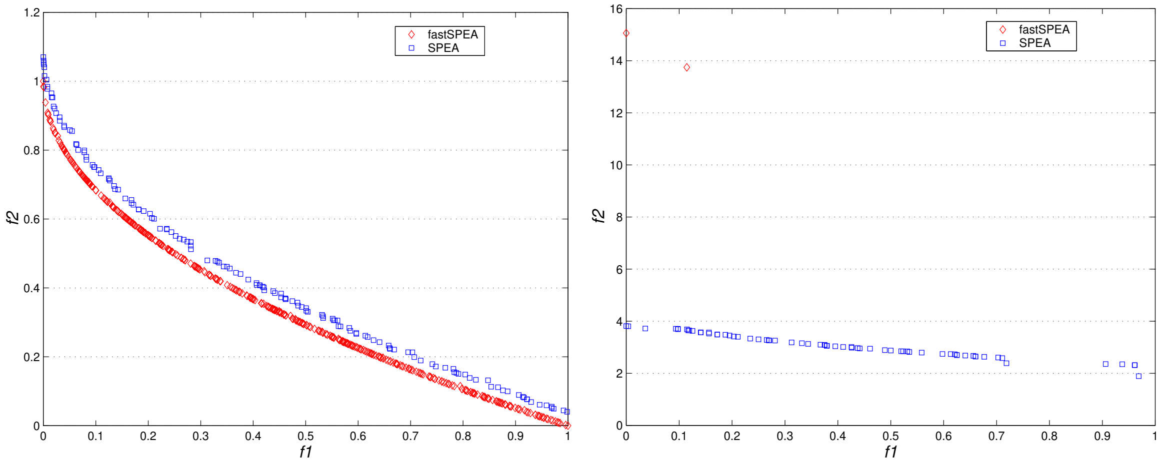

Figure 6: Comparison between SPEA and fastSPEA for test function T_1 (left) and test function T_4 (right).

4.1 Comparison Between Evolutionary Operators

In the first experiment, two SPEAs with different evolutionary operators are compared. The first version of SPEA is the original SPEA proposed by Zitzler *et al.* [12]. The operators are one point crossover and bit flipping mutation. The second version of SPEA replaces the above operators with discrete recombination and Gaussian mutation, which were the same ones used in $(\mu+\lambda)$MEA. We name the second SPEA as "fastSPEA". Since the binary tournament selection has to be retained (where the strength is used), the discrete recombination can only operate with two individuals (the discrete recombination in $(\mu+\lambda)$MEA operates with whole population). The experimental setup is the same as the original SPEA in [12].

Since the operators are the only part changed, the comparison between both SPEAs can clearly demonstrate the influence of different operators. We only compare the performances on test functions T_1 and T_4, since they represent two extreme types of functions, easy (convex) and difficult (multimodal). The comparison methods are the same as used in Section 3. The experimental results are shown in Figures 6 and 7.

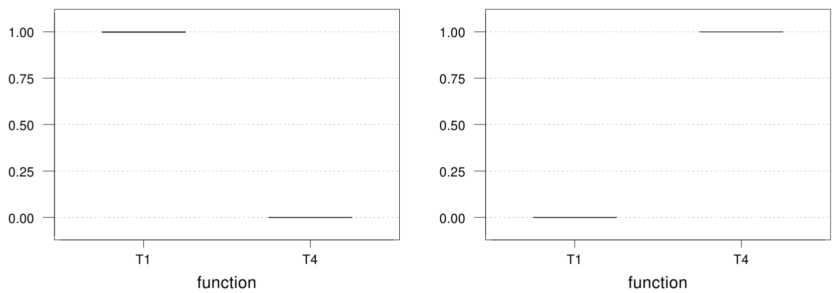

Figure 7: Box plots based on the $\mathcal{C}$ metric from functions T_1 and T_4 are shown. $\mathcal{C}(Y_{fastSPEA}, Y_{SPEA})$ is on the left, and $\mathcal{C}(Y_{SPEA}, Y_{fastSPEA})$ is on the right.

Obviously, fastSPEA performs better than SPEA on test function T_1, and worse on test function T_4. For test function T_1, the numerical operators in fastSPEA demonstrated the fast convergence rate. However, these operators made the search trapped in some local minima on the case of test function T_4. Since the discrete recombination is limited under the binary tournament selection, its performance on multimodal problems was affected. In general, global discrete recombination can provide good results in multimodal problems [22]. Salomon [25] explained in his paper why breeder GA performs well on some benchmark problems.

Summarizing the experimental results, we cannot make any decisive conclusions to the performance of different evolutionary operators. However, from the viewpoint of convergence speed we are certain that the numerical operators are better than traditional GA operators on numerical problems.

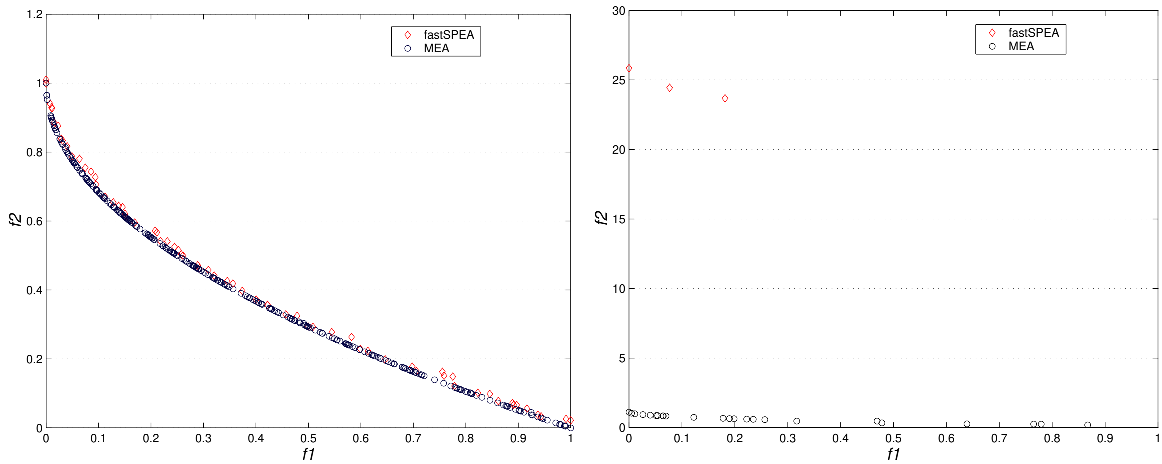

Figure 8: Comparison between fastSPEA and MEA using 5000 function evaluations on test function T_1 (left) and test function T_4 (right).

4.2 Comparison Between fastSPEA and $(\mu + \lambda)$MEA

The second experiment is to compare fastSPEA and $(\mu + \lambda)$MEA. Since both algorithms have the similar evolutionary operators, the different performance will be contributed to the nondominated solution limitation or/and parent selection schemes. To prevent overrunning the experiments, 5000 function evaluations are limited per run. Test functions T_1 and T_4 are again tested. The comparisons of experimental results are shown in Fugures 8 and 9.

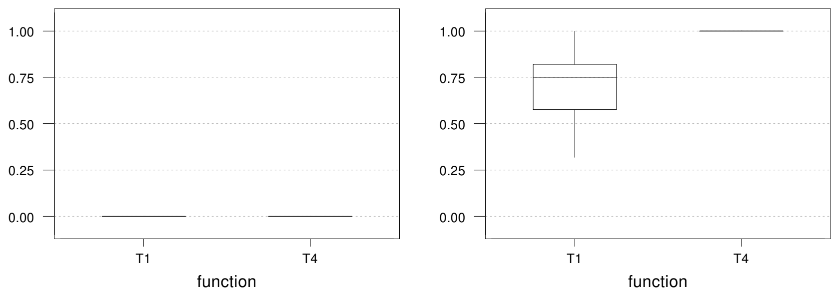

Figure 9: Box plots based on the $\mathcal{C}$ metric from solving functions T_1 and T_4 with 5000 function evaluations are shown. $\mathcal{C}(Y_{fastSPEA}, Y_{(\mu+\lambda)MEA})$ is on the left, and $\mathcal{C}(Y_{(\mu+\lambda)MEA}, Y_{fastSPEA})$ is on the right.

The methods of nondominated solution limitation mainly work for solutions distribution. Since the solution of both algorithms are evenly distributed near the Pareto front on T_1 in Figure 8, we cannot judge which algorithm has better distribution. However, we should have no doubt about the success of both nondominated solution limitation methods.

Excluding the effect of the nondominated solution limitation, we found that the $(\mu+\lambda)$MEA's solutions are closer to the Pareto front than fastSPEA. In fact, the $\mathcal{C}$ metrics on T_1 in Figure 9 show that few $(\mu+\lambda)$MEA's solutions are covered by fastSPEA, and most fastSPEA's solutions are covered by $(\mu+\lambda)$MEA. We can conclude that $(\mu + \lambda)$MEA outperforms fastSPEA. Therefore, we may infer that $(\mu + \lambda)$MEA's parent selection scheme is the major key to its success. The advantages of $(\mu + \lambda)$MEA's parent selection scheme are two-fold.

1. No extra fitness calculation is needed. The mechanism of determining nondominated solutions is applied to decide the individual's fitness.

2. No large population size is maintained. If only one nondominated solution was found in certain generations,

the population size would be simply ν. The smaller population size during the run usually saves a lot of unnecessary computation in generating dominated offspring.

From the empirical studies, we can conclude that $(\mu + \lambda)$MEA's parent selection scheme plays the major role as an effective and efficient multiobjective evolutionary algorithm.

5 Conclusions

In this paper, we developed a $(\mu + \lambda)$ multiobjective evolutionary algorithm. Without reserving the dominated individuals, the $(\mu + \lambda)$MEA appeared to have a very fast search speed. We compared its performance with SPEA using five benchmark functions. The results have shown that the $(\mu + \lambda)$MEA performs better than SPEA in both objective functions values and search speed. We conducted two empirical analysis to find which part of $(\mu+\lambda)$MEA makes more contribution to the search results. From the empirical studies, we can conclude that $(\mu + \lambda)$MEA's parent selection scheme plays a major role to produce quality solutions. Although the fast search speed results from the rejection of the dominated individuals, the search diversity may not be enough for some problems with discrete Pareto-optimality. So more tests on different numerical multiobjective problems will be one area of our future work. To solve discrete or combinatorial problems, the development of other suitable mutation operators is also part of our future research.

Acknowledgements

The authors gratefully acknowledge the support by a UC(UNSW, ADFA) Special Research Grant 1999-2000 awarded to Dr. R. Sarker. Ko-Hsin Liang also thanks Dr. Zitzler for discussion about SPEA. The authors like to thank two unknown referees for their many useful comments and constructive suggestions.

References

[1] C. Fonseca and P. Fleming, "An overview of evolutionary algorithms in multiobjective optimization," *Evolutionary Computation*, vol. 3, pp. 1–16, 1995.

[2] C. Coello, "A Comparative Survey of Evolutionary-based Multiobjective Optimization Techniques," *Knowledge and Information Systems*, vol. 1, pp. 269–308, 1999.

[3] I. Das and J. Dennis, "A closer look at drawbacks of minimizing weighted sums of objectives for Pareto set generation in multicriteria optimization problems," *Strictural Optimization*, vol. 14, pp. 63–69, 1997.

[4] J. Schaffer, "Multiple objective optimization with vector evaluated genetic algorithms," in *Genetic Algorithms and their Applications: Proceedings of the First International Conference on Genetic Algorithms*, pp. 93–100, Lawrence Erlbaum, 1985.

[5] F. Kursawa, "A variant of evolution strategies for vector optimization," in *Parallel Problem Solving from Nature – PPSN I (H.-P. Schwefel and R. Manner, eds.), vol. 496 of Lecture Notes in Computer Science (Berlin)*, pp. 193–197, Springer-Verlag, 1991.

[6] D. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading, MA: Addison Weslay, 1989.

[7] N. Srinivas and K. Deb, "Multiobjective Optimization using Nondominated Sorting in Genetic Algorithms," *Evolutionary Computation*, vol. 2, no. 3, pp. 221–248, 1994.

[8] C. Fonseca and P. Fleming, "Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization," *Proceedings of the Fifth International Conference on Genetic Algorithms, San Mateo, California*, pp. 416–423, 1993.

[9] J. Horn, N. Nafpliotis, and D. Goldberg, "A niched pareto genetic algorithm for multipleobjective optimization," in *Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence, Piscataway, NJ*, pp. 82–87, Lawrence Erlbaum, 1994.

[10] K. Deb and D. Goldberg, "An investigation of niche and spices formation in genetic function optimization," in *Proceedings of the third international Conference on Genetic Algorithms, George Mason University, (Sat Mateo, California)*, pp. 42–50, Lawrence Erlbaum, 1989.

[11] E. Zitzler and L. Thiele, "An Evolutionary Algorithm for Multiobjective optimization: The stength pareto approach," in *Tech. Rep. 43, Computer Engineering and Communication Network Lab (TIK), Swiss Federal Institute of Technology (ETH)*, Zurich, Gloriastrasse, CH-8092 Zurich, 1998.

[12] E. Zitzler and L. Thiele, "Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 4, pp. 257–271, 1999.

[13] E. Zitzler, K. Deb, and L. Thiele, "Comparison of multiobjective evolutionary algorithms: Empirical results," *Evolutionary Computation*, vol. 8, no. 2, pp. 173–195, 2000.

[14] K. Deb, "Multi-objective genetic algorithms: Problem difficulties and construction of test problems," *Evolutionary Computation*, vol. 7, no. 4, pp. 205–230, 1999.

[15] J. Knowles and D. Corne, "The Pareto archived evolution strategy: a new baseline algorithm for multiobjective optimization," *In 1999 Congress on Evolutionary Computation, Washington D.C., IEEE Service Centre*, pp. 98–105, 1999.

[16] J. Knowles and D. Corne, "Approximating the nondominated front using the Pareto archived evolution strategy," *Evolutionary Computation*, vol. 8, no. 2, pp. 149–172, 2000.

[17] T. Binh and U. Korn, "An evolution strategy for the multiobjective optimization," in *Proceedings of the second international Conference on Genetic Algorithms (Mendel96), Brno, Czech Republic*, pp. 23–28, 1996.

[18] T. Binh and U. Korn, "MOBES: A multiobjective evolution strategy for constrained optimization problems," in *Proceedings of the third international Conference on Genetic Algorithms (Mendel97), Brno, Czech Republic*, pp. 176–182, 1997.

[19] H.-P. Schwefel, *Evolution and optimum seeking.* New York: Wiley, 1995.

[20] T. Back and H.-P. Schwefel, "An overview of evolutionary algorithms for parameter optimization," *Evolutionary Computation*, vol. 1, no. 1, pp. 1–23, 1993.

[21] T. Back, *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms.* New York: Oxford University, 1996.

[22] H. Muhlenbein and D. Schlierkamp-Voosen, "Predictive models for the breeder genetic algorithm I. continuous parameter optimization," *Evolutionary Computation*, vol. 1, no. 1, pp. 25–49, 1993.

[23] D. V. Veldhuizen and G. Lamont, "Multiobjective Evolutionary Algorithm Test Suites," *In J. Carroll, H. Haddad, D. Oppenheim, B. Bryant and G. Lamont, editors*, pp. 351–357, 1999.

[24] C. M. Fonseca and P. J. Fleming, "On the Performance Assessment and Comparison of Stochastic Multiobjective Optimizers," in *Parallel Problem Solving from Nature—PPSN IV, ed. Hans-Michael Voigt and Werner Ebeling and Ingo Rechenberg and Hans-Paul Schwefel*, pp. 584–593, Lecture Notes in Computer Science, Springer-Verlag, Berlin, Germany, 1996.

[25] R. Salomon, "Some comments on evolutionary algorithm theory," *Evolutionary Computation*, vol. 4, no. 4, pp. 405–415, 1997.

Appendix: Test Functions

Five benchmark test functions are listed here. For all functions the Pareto-optimal front is formed with $g = 1$. n is the number of decision vector,$\mathbf{x}$.

No.	Function	n	boundries	characteristic
T_1	$f_1(\mathbf{x}) = x_1$ $f_2(\mathbf{x}) = g * (1 - \sqrt{f_1/g})$ $g = 1 + 9 \cdot (\sum_{i=2}^{n} x_i)/(n-1)$	30	$x_i \in [0, 1]$	convex
T_2	$f_1(\mathbf{x}) = x_1$ $f_2(\mathbf{x}) = g * (1 - (f_1/g)^2$ $g = 1 + 9 \cdot (\sum_{i=2}^{n} x_i)/(n-1)$	30	$x_i \in [0, 1]$	non-convex
T_3	$f_1(\mathbf{x}) = x_1$ $f_2(\mathbf{x}) = g * (1 - \sqrt{f_1/g} - (f_1/g)\sin(10\pi f_1))$ $g = 1 + 9 \cdot (\sum_{i=2}^{n} x_i)/(n-1)$	30	$x_i \in [0, 1]$	discrete
T_4	$f_1(\mathbf{x}) = x_1$ $f_2(\mathbf{x}) = g * (1 - \sqrt{f_1/g})$ $g = 1 + 10(n-1) + \sum_{i=2}^{n}(x_i^2 - 10\cos(4\pi x_i))$	10	$x_1 \in [0, 1]$ $x_2, \cdots, x_n \in [-5, 5]$	multimodal
T_6	$f_1(\mathbf{x}) = 1 - \exp(-4x_1)\sin^6(6\pi x_1)$ $f_2(\mathbf{x}) = g * (1 - (f_1/g)^2)$ $g = 1 + 9 \cdot ((\sum_{i=2}^{n} x_i)/(n-1))^{0.25}$	10	$x_i \in [0, 1]$	non-uniform