Applied Mathematics & Information Sciences
An International Journal

http://dx.doi.org/10.12785/amis/070531

An Improved Multi-Objective Genetic Algorithm for Solving Multi-objective Problems

Sheng-Ta Hsieh[1,*], *Shih-Yuan Chiu*[2] *and Shi-Jim Yen*[2]

[1] Department of Communication Engineering, Oriental Institute of Technology, New Taipei City, 220, Taiwan, R.O.C.
[2] Department of Computer Science and Information Engineering, National Dong Hwa University, Hualien County, 974, Taiwan, R.O.C.

Received: 24 Jan. 2013, Revised: 25 May. 2013, Accepted: 26 May. 2013
Published online: 1 Sep. 2013

Abstract: Multi-objective optimization (MO) has been an active area of research in the last two decades. In multi-objective genetic algorithm (MOGA), the quality of newly generated offspring of the population will directly affect the performance of finding the Pareto optimum. In this paper, an improved MOGA, named SMGA, is proposed for solving multi-objective optimization problems. To increase efficiency during solution searching, an effective mutation named *sharing mutation* is adopted to generate potential offspring. Experiments were conducted on CEC-09 MOP test problems. The results show that the proposed method exhibits better performance when solving these benchmark problems compared to related multi-objective evolutionary algorithms (MOEA).

Keywords: Genetic algorithm, multi-objective, optimization, sharing mutation

1 Introduction

Multi-objective optimization (MO) problems contain more than one objective that needs to be achieved simultaneously. Such problems arise in many applications, where two or more, sometimes conflicting and/or incommensurable objective functions have to be minimized concurrently. In such problems, there is no single optimal solution; there is instead a set of potential solutions for each of the objectives, considered alongside the solutions for other objectives. Due to the multi-criteria nature of MO problems, optimal solutions need to be redefined. A set of potential solutions are called *Pareto-optimal* or *non-dominated* solutions in multi-objective optimization problems (MOPs). In contrast to the singleobjective optimization case, MOPs are characterized by trade-offs.

A solution x_1 is said to dominate another solution x_2, if both statements below are satisfied.

–The solution x_1 is no worse than x_2 in all objectives.
–The solution x_1 is strictly better than x_2 in at least one objective.

The plot of the objective function whose non-dominated vectors are in the Pareto optimal set is called the *Pareto front*.

2 Related works

Many evolution-based MO algorithms have been proposed in last two decades. Hajela and Lin proposed a Weight-based Genetic Algorithm (WBGA) [1], which gives a weighted value for each objective function, and accumulates them as the fitness value. This algorithm possesses simple concepts, but had unsatisfactory performance when the optimal solutions are distributed as a non-convex Pareto graph. The Multi-Objective GA (MOGA) [2] was proposed by Fonseca and Fleming. The discovered non-dominated solutions are classified and then ranked to enhance the searching abilities for finding non-dominated solutions, and to maintain the diversity of the discovered solutions. This method cannot ensure that a solution with a poorer rank will always be mapped to a lower fitness value. This attributes the algorithm with slower convergence and more instability. Later, Srinivas and Deb proposed a Non-dominated Sorting Genetic Algorithm (NSGA) [3], which ranks populations according to its characteristic of non-domination, and gives higher fitness values for better non-dominated solutions. The Niched Pareto Genetic Algorithm (NPGA) [4] was proposed by Horn *et al.* It introduced a binary tournament selection but does not assign a definite fitness

[*] Corresponding author e-mail: fo013@mail.oit.edu.tw

value, and problems with more optimized objectives will influence the computational efficiency of NPGA.

Several efficient strategies were then introduced based on these algorithms, such as Elitism, external repository, or archive. Zitzler and Thiele proposed the Strength Pareto Evolutionary Algorithm (SPEA) [5], which introduced an elitism strategy to store an extra population that contains non-dominated solutions. New found non-dominated solutions will be compared with the auxiliary stored population, and the better solution is kept. The SPEA2 [6] is an advanced version of SPEA. SPEA2 inherited the advantages from SPEA and improved fitness assignment to take both dominated and non-dominated solutions into account. SPEA2 also considered the diversity of neighboring solutions to produce more capable guides. Similar to MOGA, all these approaches have the same problem; non-dominated solutions with the same ranks may not have the same status. In [7], Knowles *et al.* proposed Pareto Archive Evolution Strategy (PAES), which employs (1+1) evolution strategy (ES) and uses a mutation operator for local searches. A map of a grid is applied in the algorithm to maintain the diversity of the archive. Thus, there will be a trade-off to define the size of both the external repository and grid of the map. Deb proposed an enhanced NSGA named NSGA-II [8][9] which employs a fast non-dominated approach to assign ranks to individuals and crowded tournament selection for density estimation. In the case of a tie in rank during the selection process, the individual with a lower density count will be chosen.

In recent years, many evolutionary algorithm based multi-objective optimization (MOEA) methods have been suggested and developed. In 2007, Campelo et al. [10] suggested that the negative selection, danger theory and other immune mechanisms may improve the existing MOIAs. In the same year, Elaoud et al. proposed Pareto fitness genetic algorithm (PFGA) [11]. It modified ranking procedure and possessed a promising way of sharing. Zhang and Li [12] proposed an interested MO algorithm named *MOEA/D*. It decomposes a multi-objective optimization problem into a number of scalar optimization sub-problems and optimizes them simultaneously. Recently, for many-objective problems (when there are more than three objectives), Adra and Fleming proposed diversity management mechanisms [13] to investigate solution convergence of such problems.

Although mass MO approaches have been developed, the efficiency of MO algorithms during solution searching is still an important issue. In this paper, an efficient mutation method called *sharing mutation* (SM) is adopted to assist multi-objective genetic algorithms in the exploration for optimal solutions. It can significantly improve the solution searching abilities of a multi-objective optimizer. Chromosomes will be more efficient, and discover more solutions located on or near the Pareto front.

The rest of the paper is organized as follows; Section 2 describes genetic algorithm briefly, Section 3 describes the proposed method, Section 4 presents the experimental results and Section 5 of the paper contains the conclusion.

3 Genetic Algorithm

The traditional genetic algorithm (TGA) possesses the following features:

–A bit string representation.
–Proportional selection.
–Cross-over as the primary method to produce new individuals.
–Mutation for disturbing evolution to avoid solutions falling into local search.
–Elitism policies employed.

A brief description of genetic algorithm will be introduced in this section.

3.1 Chromosome Representation

Considering that a problem is presented as $f(x_1,x_2...x_N)$ which consists of N tunable parameters to be optimized. The problem can be encoded by a vector representation in GA (i.e., chromosome) as $\mathbf{C}^m[x_1,x_2...x_N]$, $m = 1,2,...,p$, where p denotes the population size. For high-dimension or complex problems, GA will require a larger population to ensure uniform distribution of population in the searching space; otherwise potential solutions may go unfounded. The value of p is always given experimentally.

3.2 Initial Population

For most optimization techniques, the final solutions are often restricted by the initialization. However, GA is able to overcome this drawback with the cross-over and mutation operation. Chromosomes can therefore be scattered across an area in the first generation. The initial population will be used to generate p chromosomes which will be uniformly distributed across the searching space.

3.3 Cross-over

The purpose of the cross-over operation is to produce new chromosomes (offspring) by mating two random parent chromosomes, but it does not guarantee that the offspring produced is fitter than the parent. However, after adopting “exploration” and “exploitation” during the performance of cross-over, optimal results can be ensured because the offspring will be generated around fitter parents. The detail of “exploration” and “exploitation” are described in Section 4.1. The number of individuals which will be joined during cross-over is based on a pre-defined parameter r_c which is called *cross-over rate*. Thus, there will be *round* $(p \times r_c)$ individuals (parents) joined to perform cross-over.

For example, assume that two chromosomes $\mathbf{C}_1$, $\mathbf{C}_2$ are randomly picked from population for cross-over, and $\mathbf{C}_1$ is better than $\mathbf{C}_2$. The offspring $\mathbf{O}_c$ can be obtained by extrapolation cross-over using:

$$\mathbf{O}_c = \mathbf{C}_1 + \alpha(\mathbf{C}_1 - \mathbf{C}_2) \tag{1}$$

where α is a random value between [0, 1]. On the other hand, the offspring $\mathbf{O}_c$ can also be obtained by interpolation cross-over using:

$$\mathbf{O}_c = \mathbf{C}_1 - \alpha(\mathbf{C}_1 - \mathbf{C}_2) \tag{2}$$

3.4 Mutation

The mutation operator exists to randomly alter some subparts of a chromosome. When GA is learning, the chromosomes will move to the nearest optimal solution to itself, but that may be not a global optimization. Therefore, some disturbances to aid in extending the search range are quite important. In general, the offspring of mutation $\mathbf{O}_m$ is generated inside the search space randomly as

$$\mathbf{O}_m = \alpha \tag{3}$$

where α denotes a mutation vector with random components of uniform distribution in the search space. The number of parents which joins mutation is based on a predefined parameter r_m which is called *mutation rate*. Thus, there are *round* $(p \times r_m)$ individuals (parents) that will be joined to perform mutation.

In general, the fitness of mutated offspring can be better or worse than their parents and/or any cross-over offspring. On the other hand, adopting the mutation operation will extend the search range in order to explore unsearched areas in the search space in order to find the potential optimal solution.

3.5 Selection

After cross-over and mutation operations, all chromosomes, including parents and offspring in a population, will be larger than the initialization. In order to produce better offspring, the elitism operation is adapted to select p better chromosomes which will survive in the next generation.

The GA optimization is combined with operations mentioned above and repeats the evolution process until it reaches the pre-defined terminating conditions. The pseudo code of GA is given as follows.

```
Initiate population $\mathbf{C}(0)$
Evaluate the fitness values of $\mathbf{C}(0)$
**Repeat** $g = 1 : max\_generations$
**for** each chromosome
Generate offspring $\mathbf{O}_c$ using (1) or (2)
Mutate offspring $\mathbf{O}_m$ by (3).
Evaluate the fitness values of $\mathbf{O}_c$ and $\mathbf{O}_m$.
Assemble all chromosomes including $\mathbf{C}(g)$, $\mathbf{O}_c$ and $\mathbf{O}_m$,
Pick up $M$ better chromosomes and named $\mathbf{C}(g+1)$
**endfor**
**Until** Terminating condition is met
```

4 Proposed Method

Although there are numerous approaches of MOGA, premature convergence, diversity, and solutions located on/near the Pareto front when solving MO problems are still major deficiencies. In MO problems, more than one conflicting objectives need to be optimized simultaneously. Thus, non-dominated solutions which are located on/near the Pareto front will be more than one. Each non-dominated solution can provide its position information to guide the current population in finding better solutions.

4.1 Cross-over Operations

In general, cross-over operation in GA is employed to generate new (better) offspring (chromosomes) based on their parents. It combines the information of two chromosomes, which were also generated by chromosomes in a previous generation and evaluated by the cost function. Finally, the better chromosomes will be kept in the population. If a chromosome discovers a new probable solution, its offspring will move closer to it to explore the region deeply in the proceeding cross-over process. Thus, in this paper, both the exploration and exploitation are adopted to generate new offspring [14]. Exploitation strategies will restrict the searching range and reduce its size, while the exploration strategies will extend and expand the searching range. The exploration and exploitation cross-over strategies are shown in Fig. 1.

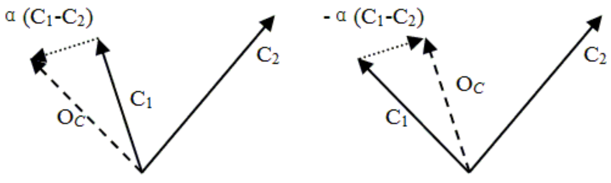


Figure 1 Exploration and exploitation cross-over strategies

The cross-over operation of proposed MOGA is that all the genes, N-dimensions, are involved to produce new chromosomes. Each pair of parents will produce two offspring which are generated by either exploration or exploitation randomly.

4.2 Sharing Mutation

In general, mutation is adopted to generate new chromosomes, mutating one or more genes, to prevent chromosomes from falling into the local optimum. It can help populations to better explore additional potential search spaces and to produce more potential solutions. Furthermore, it can also aid in the exploration of unsearched solution space. The performance of mutation will affect solution searching directly.

In order to improve mutation, *sharing mutation* (SM) is adopted for increased efficiency while exploring the solution space. The proposed sharing mutation can be classified into two versions: local sharing and global sharing. The activating probability of local and global sharing is 2 to 1. The main difference between the local sharing mutation and global sharing mutation is dimension selection.

For the local version of sharing mutation, one of the dimensions will be picked randomly; the mutating chromosome's corresponding dimension will be perturbed and restricted as this dimension's solution for all chromosomes. For example, a randomly selected dimension ($d1$) of the chromosome $i$will be perturbed in the range between $[S_{d1_{\min}}, S_{d1_{\max}}]$, where$S_{d1_{\min}}$and $S_{d1_{\max}}$are the minimal and maximal solution of $d1$ of all chromosomes respectively. In other words, the local version is the sharing of searching ranges of selected dimensions among chromosomes to efficiently generate new solutions. This will ignore other dimensions but can fine tune the solutions of specific dimensions one by one in the chromosome.

The same principle applies for the global version of sharing mutation; the current chromosome's dimension $d1$ will be perturbed and restricted as the initial boundary. For example, the chromosome $i$will be perturbed in the range between $[X_{Min}, X_{Max}]$, where X_{Min} and X_{Max} are the minimal and maximal initial boundary respectively. The global version of sharing mutation can prevent solutions of a particular dimension from being trapped in the local optimum.

Whether it's the local version or the global version of sharing mutation, they will ignore other dimensions but can fine tune the solutions of chosen dimensions one by one in the chromosomes.

Just as generic mutation rates in GA, different mutation rates will affect the performance of solution exploration directly. Subsequently, after numerous generations, chromosomes will gather in several clusters, and therefore increase the demands of SM. A lower SM activating rate may be inefficient at rescuing chromosomes trapped in local minimal, and a higher one would deeply interfere with the convergence of chromosomes. To ensure better solutions can constantly be obtained efficiently and also to prevent chromosomes from perform local searches, the concept of linearly variation inertia weight for PSO [15][16] is adopted. The mutation rate r_m for each generation is defined as follows:

$$r_m(g) = 0.001 + \frac{0.009 * g}{\text{max\_gens}} \tag{4}$$

where g denotes the generation number and max_gens is maximum generations. The probability of mutation will keep increasing linearly, from 0.001 to 0.01, during the solution searching process.

4.3 Archive

The function of the repository controller is to make decisions about whether certain solutions should be included into the archive or not. The decision-making process is stated as follows.

1. If the archive is empty, any new solution N_S found will always be accepted and stored in archive (*Case* 1, in Fig. 2).
2. If the new solution is dominated by any individual in the archive, then such a solution will be discarded (*Case* 2, in Fig. 2).
3. If none of the solutions contained in the archive dominates the new solution, then such a solution will be stored in the archive (*Case* 3, in Fig. 2).
4. Otherwise, if there are solutions in the archive that are dominated by the new solution, then such dominated solutions will be removed from the archive (*Case* 4, in Fig. 2).

Finally, after updating all non-dominated solutions, the cluster procedure will then be activated to eliminate similar solutions to ensure a lower diversity of non-dominated solutions.

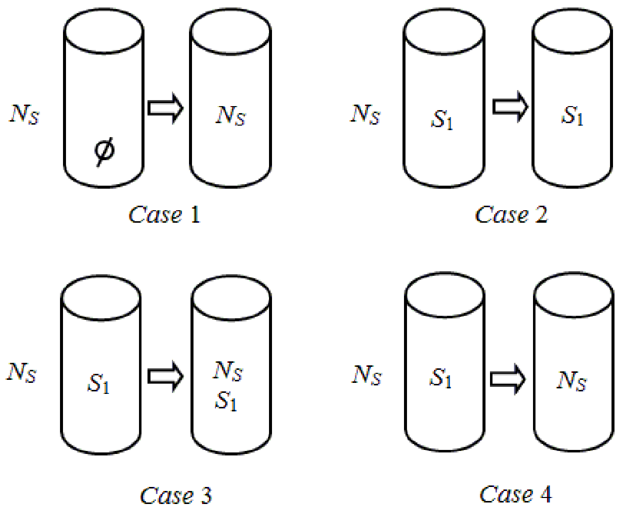


Figure 2. Possible Cases for Archive Controller

5 Experiments

5.1 Test Functions

Ten unconstrained (bound constrained) MOP test problems of CEC 2009 technic report [17] were adopted for testing the proposed method with the results compared to MOEA/D [12] and NSGA-II [9]. The test problems are listed as follows:

Problem 1

$$\begin{aligned}
&f_1 = x_1 + \tfrac{2}{|J_1|}\sum_{j\in J_1}\left[x_j - \sin(6\pi x_1 + \tfrac{j\pi}{n})\right]^2\\
&f_2 = 1-\sqrt{x_1} + \tfrac{2}{|J_2|}\sum_{j\in J_2}\left[x_j - \sin(6\pi x_1 + \tfrac{j\pi}{n})\right]^2\\
&J_1 = \{j|j \textit{ is odd and } 2\le j\le n\} \textit{ and}\\
&J_2 = \{j|j \textit{ is even and } 2\le j\le n\}\\
&\textit{search space is } [0,1]\mathrm{x}[-1,1]^{n-1}\\
&n = 30
\end{aligned} \tag{5}$$

Problem 2

$$\begin{aligned}
&f_1 = x_1 + \tfrac{2}{|J_1|}\sum_{j\in J_1} y_j^2\\
&f_2 = 1-\sqrt{x_1} + \tfrac{2}{|J_2|}\sum_{j\in J_2} y_j^2\\
&J_1 = \{j|j \textit{ is odd and } 2\le j\le n\} \textit{ and}\\
&J_2 = \{j|j \textit{ is even and } 2\le j\le n\}\\
&y_j = \begin{cases} x_j - [0.3x_1^2\cos(24\pi x_1 + \frac{4j\pi}{n}) + 0.6x_1]\cos(6\pi x_1 + \frac{j\pi}{n}) & j\in J_1\\ x_j - [0.3x_1^2\cos(24\pi x_1 + \frac{4j\pi}{n}) + 0.6x_1]\sin(6\pi x_1 + \frac{j\pi}{n}) & j\in J_2\end{cases}\\
&\textit{search space is } [0,1]\mathrm{x}[-1,1]^{n-1}\\
&n = 30
\end{aligned} \tag{6}$$

Problem 3

$$\begin{aligned}
&f_1 = x_1 + \tfrac{2}{|J_1|}(4\sum_{j\in J_1} y_j^2 - 2\prod_{j\in J_1}\cos(\tfrac{20y_j\pi}{\sqrt{j}}) + 2)\\
&f_2 = 1-\sqrt{x_1} + \tfrac{2}{|J_2|}(4\sum_{j\in J_2} y_j^2 - 2\prod_{j\in J_2}\cos(\tfrac{20y_j\pi}{\sqrt{j}}) + 2)\\
&J_1 = \{j|j \textit{ is odd and } 2\le j\le n\} \textit{ and}\\
&J_2 = \{j|j \textit{ is even and } 2\le j\le n\}\\
&y_j = x_j - x_1^{0.5(1.0+\frac{3(j-2)}{n-2})}, j = 2,3,4,5,\ldots,n\\
&\textit{search space is } [0,1]^n\\
&n = 30
\end{aligned} \tag{7}$$

Problem 4

$$\begin{aligned}
&f_1 = x_1 + \tfrac{2}{|J_1|}\sum_{j\in J_1} h(y_j)\\
&f_2 = 1-x_1^2 + \tfrac{2}{|J_2|}\sum_{j\in J_2} h(y_j)\\
&J_1 = \{j|j \textit{ is odd and } 2\le j\le n\} \textit{ and}\\
&J_2 = \{j|j \textit{ is even and } 2\le j\le n\}\\
&y_j = x_j - \sin(6\pi x_1 + \tfrac{j\pi}{n}), j = 2,\ldots,n\\
&h(t) = \tfrac{|t|}{1+e^{2|t|}}\\
&\textit{search space is } [0,1]\mathrm{x}[-2,2]^{n-1}\\
&n = 30
\end{aligned} \tag{8}$$

Problem 5

$$\begin{aligned}
&f_1 = x_1 + (\tfrac{1}{2N}+\varepsilon)\,|\sin(2N\pi x_1)| + \tfrac{2}{|J_1|}\sum_{j\in J_1} h(y_j)\\
&f_2 = 1-x_1 + (\tfrac{1}{2N}+\varepsilon)\,|\sin(2N\pi x_1)| + \tfrac{2}{|J_2|}\sum_{j\in J_2} h(y_j)\\
&J_1 = \{j|j \textit{ is odd and } 2\le j\le n\} \textit{ and}\\
&J_2 = \{j|j \textit{ is even and } 2\le j\le n\}.\ \text{N is an integer, } \varepsilon\text{¿}0\\
&y_j = x_j - \sin(6\pi x_1 + \tfrac{j\pi}{n}), j = 2,\ldots,n\\
&h(t) = 2t^2 - \cos(4\pi t) + 1\\
&\textit{search space is } [0,1]\mathrm{x}[-1,1]^{n-1}\\
&N = 10, \varepsilon = 0.1, n = 30
\end{aligned} \tag{9}$$

Problem 6

$$\begin{aligned}
&f_1 = x_1 + \max\{0, 2(\tfrac{1}{2N}+\varepsilon)\sin(2N\pi x_1)\}\\
&\quad + \tfrac{2}{|J_1|}(4\sum_{j\in J_1} y_j^2 - 2\prod_{j\in J_1}\cos(\tfrac{20y_j\pi}{\sqrt{j}}) + 2)\\
&f_2 = 1-x_1 + \max\{0, 2(\tfrac{1}{2N}+\varepsilon)\sin(2N\pi x_1)\}\\
&\quad + \tfrac{2}{|J_2|}(4\sum_{j\in J_2} y_j^2 - 2\prod_{j\in J_2}\cos(\tfrac{20y_j\pi}{\sqrt{j}}) + 2)\\
&J_1 = \{j|j \textit{ is odd and } 2\le j\le n\}, J_2 = \{j|j \textit{ is even and } 2\le j\le n\}.\\
&y_j = x_j - \sin(6\pi x_1 + \tfrac{j\pi}{n}), j = 2,\ldots,n\\
&\textit{search space is } [0,1]\mathrm{x}[-1,1]^{n-1}\\
&N = 2, \varepsilon = 0.1, n = 30
\end{aligned} \tag{10}$$

Problem 7

$$\begin{aligned}
&f_1 = \sqrt[5]{x_1} + \tfrac{2}{|J_1|}\sum_{j\in J_1} y_j^2\\
&f_2 = 1-\sqrt[5]{x_1} + \tfrac{2}{|J_2|}\sum_{j\in J_2} y_j^2\\
&J_1 = \{j|j \textit{ is odd and } 2\le j\le n\}, J_2 = \{j|j \textit{ is even and } 2\le j\le n\}.\\
&y_j = x_j - \sin(6\pi x_1 + \tfrac{j\pi}{n}), j = 2,\ldots,n\\
&\textit{search space is } [0,1]\mathrm{x}[-1,1]^{n-1}\\
&n = 30
\end{aligned} \tag{11}$$

Problem 8

$$\begin{aligned}
&f_1 = \cos(0.5x_1\pi)\cos(0.5x_2\pi) + \tfrac{2}{|J_1|}\sum_{j\in J_1}(x_j - 2x_2\sin(2\pi x_1 + \tfrac{j\pi}{n}))^2\\
&f_2 = \cos(0.5x_1\pi)\sin(0.5x_2\pi) + \tfrac{2}{|J_2|}\sum_{j\in J_2}(x_j - 2x_2\sin(2\pi x_1 + \tfrac{j\pi}{n}))^2\\
&f_3 = \sin(0.5x_1\pi) + \tfrac{2}{|J_3|}\sum_{j\in J_3}(x_j - 2x_2\sin(2\pi x_1 + \tfrac{j\pi}{n}))^2\\
&J_1 = \{j|3\le j\le n, \textit{ and } j-1 \textit{ is a multuplication of } 3\}\\
&J_2 = \{j|3\le j\le n, \textit{ and } j-2 \textit{ is a multuplication of } 3\}\\
&J_3 = \{j|3\le j\le n, \textit{ and } j \textit{ is a multuplication of } 3\}\\
&\textit{search space is } [0,1]^2\mathrm{x}[-2,2]^{n-2}\\
&n = 30
\end{aligned} \tag{12}$$

Problem 9

$$\begin{aligned}
&f_1 = 0.5[\max\{0,(1+\varepsilon)(1-4(2x_1-1)^2)\}+2x_1]x_2\\
&\quad+\tfrac{2}{|J_1|}\sum_{j\in J_1}(x_j-2x_2\sin(2\pi x_1+\tfrac{j\pi}{n}))^2\\
&f_2 = 0.5[\max\{0,(1+\varepsilon)(1-4(2x_1-1)^2)\}+2x_1]x_2\\
&\quad+\tfrac{2}{|J_2|}\sum_{j\in J_2}(x_j-2x_2\sin(2\pi x_1+\tfrac{j\pi}{n}))^2\\
&f_3 = 1-x_2+\tfrac{2}{|J_3|}\sum_{j\in J_3}(x_j-2x_2\sin(2\pi x_1+\tfrac{j\pi}{n}))^2\\
&J_1=\{j|3\le j\le n,\ and\ j-1\ is\ a\ multuplication\ of\ 3\}\\
&J_2=\{j|3\le j\le n,\ and\ j-2\ is\ a\ multuplication\ of\ 3\}\\
&J_3=\{j|3\le j\le n,\ and\ j\ is\ a\ multuplication\ of\ 3\}\\
&search\ space\ is\ [0,1]^2\mathrm{x}[-2,2]^{n-2}\\
&n=30,\varepsilon=0.1
\end{aligned} \tag{13}$$

Problem 10

$$\begin{aligned}
&f_1=\cos(0.5x_1\pi)\cos(0.5x_2\pi)\\
&\quad+\tfrac{2}{|J_1|}\sum_{j\in J_1}[4y_j^2-\cos(8\pi y_i)+1]\\
&f_2=\cos(0.5x_1\pi)\sin(0.5x_2\pi)\\
&\quad+\tfrac{2}{|J_2|}\sum_{j\in J_2}[4y_j^2-\cos(8\pi y_i)+1]\\
&f_3=\sin(0.5x_1\pi)+\tfrac{2}{|J_3|}\sum_{j\in J_3}[4y_j^2-\cos(8\pi y_i)+1]\\
&y_j=x_j-2x_2\sin(2\pi x_1+\tfrac{j\pi}{n}),\ j=3,\dots,n\\
&J_1=\{j|3\le j\le n,\ and\ j-1\ is\ a\ multuplication\ of\ 3\}\\
&J_2=\{j|3\le j\le n,\ and\ j-2\ is\ a\ multuplication\ of\ 3\}\\
&J_3=\{j|3\le j\le n,\ and\ j\ is\ a\ multuplication\ of\ 3\}\\
&search\ space\ is\ [0,1]^2\mathrm{x}[-2,2]^{n-2}\\
&n=30
\end{aligned} \tag{14}$$

5.2 Parameter Settings and Initialization

All the MOGA algorithms were implemented using MATLAB 2010a. The experiments were executed on Core-2 Quad 2.66 GHz (Hyper Threading function enabled) with 4GB RAM on Windows 7 professional operating system. MOGAs' parameters are listed in Table I.

The maximal Number of Function Evaluations (FEs) is set to be 300,000 for every problem. Each algorithm was executed 30 times. Their mean values and standard deviation for the results were recorded. The initial and the variation of mutation rate (r_m) for sharing mutation were set as 0.001 and 0.009 respectively. Thus, the SM rate will keep increasing linearly, from 0.001 to 0.01, while the solution searching is in progress.

Table I Parameters' Setting of MOGAs

Methods	**Cross-over rate**	**Mutation rate**
Proposed Method	1.0	vary linearly from 0.001 to 0.01
MOEA/D	1.0	0.005
NSGA-II	1.0	0.005

5.3 Performance Metric (IGD) [17]

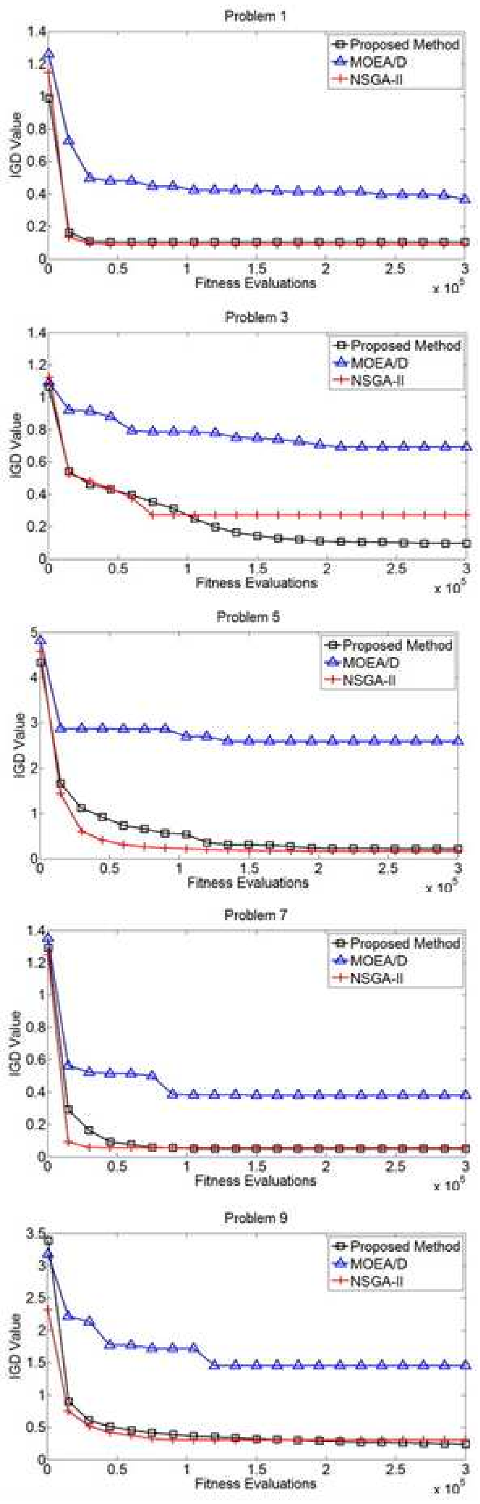
Let $P*$be a set of uniformly distributed points along the PF (in the objective space). Let Abe an approximate set to the PF, the average distance from $P*$to $A$is defined as:

$$IGD(A,\ P*)=\frac{\sum_{v\in p*}d(v,\ A)}{|P*|} \tag{15}$$

where $d(v,\ A)$ is the minimum Euclidean distance between $v$and the points in A. If $|P*|$ is large enough to represent the PF very well, $IGD(A{,}P*)$ could measure both the diversity and convergence of Ain a sense. To have a low value of $D(A{,}P*)$, The set Amust be very close to the *PF* and cannot miss any part of the whole *PF*.

Table II Results of Ten CEC 2009 Test Problems

Test Problems	**Proposed Method**	**MOEA/D** [12]	**NSGA-II** [8]
P_1	**2.10e-01** ± **9.70e-02**	3.85e-01 ± 3.76e-02	1.26e+00 ± 2.30e-01
P_2	**1.11e-01** ± **4.36e-02**	3.17e-01 ± 1.91e-02	5.83e-01 ± 1.28e-01
P_3	**3.10e-01** ± **2.89e-02**	6.89e-01 ± 3.12e-02	1.18e+00 ± 6.77e-02
P_4	**7.50e-02** ± **1.14e-02**	1.13e-01 ± 2.42e-03	1.45e-01 ± 3.43e-02
P_5	**3.20e-01** ± **9.00e-02**	2.67e+00 ± 9.98e-02	5.06e+00 ± 9.42e-01
P_6	**2.90e-01** ± **7.61e-02**	1.65e+00 ± 1.25e-01	5.77e+00 ± 6.64e-01
P_7	**2.70e-01** ± **1.85e-01**	3.99e-01 ± 6.80e-02	1.36e+00 ± 1.61e-01
P_8	**3.71e-01** ± **1.01e-01**	1.29e+00 ± 2.39e-01	2.56e+00 ± 1.00e+00
P_9	**4.29e-01** ± **6.69e-02**	1.47e+00 ± 2.33e-01	2.57e+00 ± 9.22e-01
P_{10}	**8.34e-01** ± **3.15e-01**	8.39e+00 ± 1.09e+00	1.39e+01 ± 2.92e+00



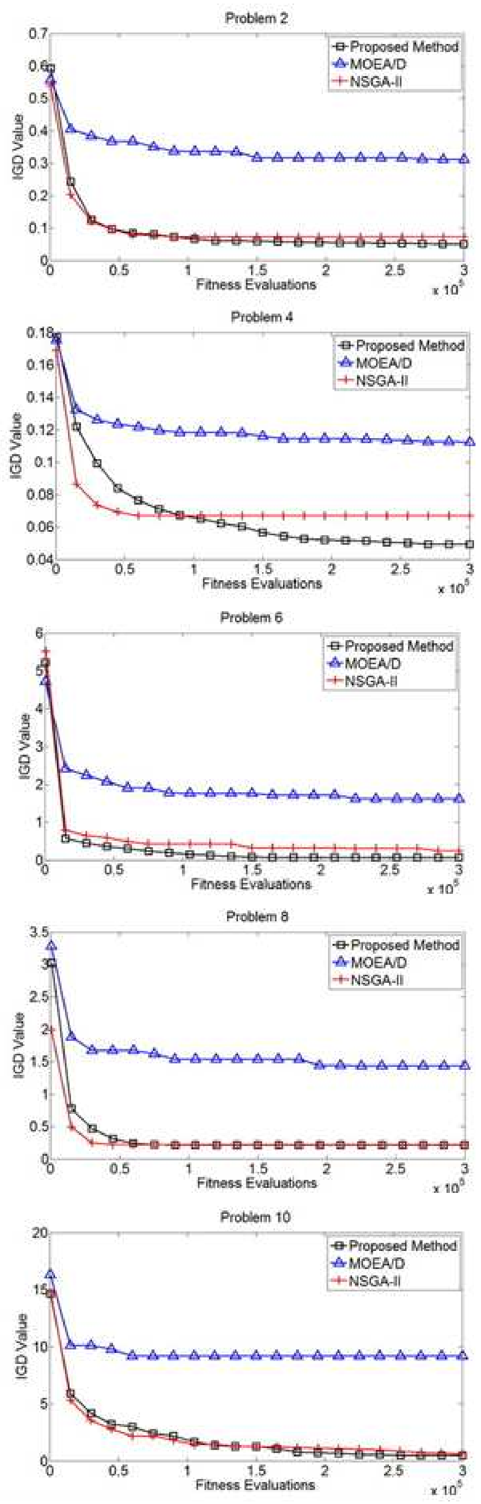


Figure 3a. Convergence characteristics of $P_1 - P_5$

Figure 3b. Convergence characteristics $P_6 - P_{10}$

5.4 Experimental Results

Table II presents the mean and standard deviation of 30 runs of the proposed MOGA, MOEA/D [12] and [8] on the ten test problems. The best results among the three approaches are shown in bold. From the results, it can be observed that the proposed MOGA performed with better results. The proposed method surpasses all other algorithms in solving all functions and shows a significant improvement on the results of problems 4, 5, 6, 8, 9 and 10. The convergence characteristics of each approach on all test functions are shown in Fig. 3. The Proposed method exhibits superior convergence than other approaches on most test functions.

6 Conclusions

In this paper, the proposed method has been presented to solve multi-objective optimization problems. Sharing mutation was adopted to improve the searching abilities of chromosomes. It also makes the proposed SMGA more robust, and prevents chromosomes from falling into the local optimum. Ten unconstrained (bound constrained) MOP test problems from the CEC 2009 technical report were selected for experiments. The experiment results show that the proposed method can find more solutions located on/near the Pareto front.

Acknowledgement

This work was supported in part by National Science Council of Taiwan, Taiwan, R.O.C. under Grant NSC 101-2221-E-161-011.

References

[1] J. D. Schaffer, Multiple objective optimization with vector evaluated genetic algorithms, Proceedings of 1st IEEE Int. Conf. Genetic Algorithms, 93100 (1985).

[2] C. M. Fonseca and P. J. Fleming, Genetic algorithms for multiobjective Optimization: Formulation, discussion and generalization, Proceedings of the 5th International Conference on Genetic Algorithms, 416-423 (1993).

[3] N. Srinivas and K. Deb, Multiobjective optimization using nondominated sorting in genetic algorithms, Evolutionary Computation, **2**, 221-248 (1994).

[4] J. Horn, N. Nafpliotis and D.E. Goldberg, A niched pareto genetic algorithm for multiobjective optimization, Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence, **1**, 82-87 (1994).

[5] E. Zitzler and L. Thiele, Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach, IEEE Transactions on Evolutionary Computation, **37**, 257-271 (1999).

[6] E. Zitzler, M. Laumanns, and L. Thiele, SPEA2: Improving the strength pareto evolutionary algorithm, Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems, 95-100 (1993). In K. Giannakoglou et al., editor, EUROGEN, (2001).

[7] J. D. Knowles and D. W. Corne, Approximarting the nondominated front using the pareto archived evolution strategy, Evolutionary Computation, **8**, 149-172 (2000).

[8] K. Deb, S. Agrawal, A. Pratab and T. Meyarivan, A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II, Proceedings of the Parallel Problem Solving from Nature VI Conference, 849-858 (2000). Springer, Lecture Notes in Computer Science, (1917).

[9] K. Deb, A. Pratap, S. Agarwal and T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Transactions on Evolutionary Computation, **6**, 182-197 (2002).

[10] F. Campelo, F. G. Guimaraes and H. Igarashi, Overview of artificial immune systems for multi-objective optimization, Springer-Verlag Lecture Notes in Computer Science, **4403**, 937-951 (2007).

[11] S. Elaoud, T. Loukil and J. Teghem, The Pareto fitness genetic algorithm: Test function study, European Journal of Operational Research, **177**, 1703-1719 (2007).

[12] Q. Zhang and H. Li, MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition, IEEE Transactions on Evolutionary Computation, **11**, (2007).

[13] S. F. Adra and P. J. Fleming, Diversity Management in Evolutionary Many-Objective Optimization, IEEE Trans. on Evolutionary Computation, **5**, 183-195 (2011).

[14] Z. Michalewicz, T. Logan, and S. Swaminathan, Evolutionary operations for continuous convex parameter spaces, Proceeding of the 3rd Annual Conference on Evolutionary Programming, 84-97 (1994).

[15] F. van den Bergh and A. P. Engelbrecht, A cooperative approach to particle swarm optimization, IEEE Transactions on Evolutionary Computation, **8**, 225-239 (2004).

[16] J. J. Liang, A. K. Qin, P. N. Suganthan and S. Baskar, Comprehensive learning particle swarm optimizer for global optimization of multimodal functions, IEEE Transactions on Evolutionary Computation, **10**, 281-296 (2006).

[17] Technical Report CES-487, http://www.ntu.edu.sg/home/EPNSugan.

Sheng-Ta Hsieh received the B.Sc. degree in electrical engineering from the National Taiwan University of Science and Technology in 2002 and received Ph.D. degrees in electrical engineering from National Dong Hwa University, Hualien, Taiwan, in 2007. He is currently an assistant professor in the department of Department of Communication Engineering, Oriental Institute of Technology, Taipei County, Taiwan, R.O.C. His interest in research includes signal processing, embedded systems and computational intelligence.

Shih-Yuan Chiu is a Ph.D. candidate in the Department of Computer Science and Information Engineering, National Dong Hwa University, Hualien, Taiwan, and he is also an instructor in the Committee for General Education in National Dong Hwa University . His research interests include computer games, puzzle game solver and computational intelligence. He is also s 4-dan Go player.

Shi-Jim Yen is an associate professor of Department of Computer Science and Information Engineering in National Dong Hwa University. He received a Ph.D degree in Computer Science and Information Engineering from National Taiwan University, Taiwan. He is an IEEE CIS member. He specialized in Artificial Intelligence and computer games. In these areas, he has published over 50 papers in international journals or conference proceedings. He is a 6-dan Go player. He served as a workshop chair on 5th international conference on Grid and Pervasive Computing in 2010, and a workshop chair of 2010 International Taiwanese Association for Artificial Intelligence (TAAI) conference. He serves as a workshop co-chair of 2011 IEEE International Conference on Fuzzy Systems. He is the Chair of the IEEE Computational Intelligence Society (CIS) Emergent Technologies Technical Committee (ETTC) Task Force on Emerging Technologies for Computer Go in 2010.