Enhancing MOEA/D with Guided Mutation and Priority Update for Multi-objective Optimization

Chih-Ming Chen
Ying-ping Chen
Qingfu Zhang

NCLab Report No. NCL-TR-2008008
November 2008

Natural Computing Laboratory (NCLab)
Department of Computer Science
National Chiao Tung University
329 Engineering Building C
1001 Ta Hsueh Road
HsinChu City 300, TAIWAN
http://nclab.tw/

Enhancing MOEA/D with Guided Mutation and Priority Update for Multi-objective Optimization

Chih-Ming Chen[1], Ying-ping Chen[1], and Qingfu Zhang[2]
[1]Department of Computer Science
National Chiao Tung University
HsinChu City 300, Taiwan
{ccming, ypchen}@nclab.tw

[2]Department of Computing and Electronic Systems
University of Essex
Colchester, CO4 3SQ, UK
qzhang@essex.ac.uk

November 14, 2008

Abstract

Multi-objective optimization is an essential and challenging topic in the domains of engineering and computation because real-world problems usually include several conflicting objectives. Current trends in the research of solving multi-objective problems (MOPs) require that the adopted optimization method provides as approximation of the Pareto set such that the user can understand the tradeoff between objectives and therefore make the final decision. Recently, an efficient framework, called MOEA/D, combining decomposition techniques in mathematics and optimization methods in evolutionary computation was proposed. MOEA/D decomposes a MOP to a set of single-objective problems (SOPs) with neighborhood relationship and approximates the Pareto set by solving these SOPs. In this paper, we attempt to enhance MOEA/D by proposing two mechanisms. To fully employ the information obtained from neighbors, we introduce a guided mutation operator to replace the differential evolution operator. Moreover, a update mechanism utilizing a priority queue is proposed for performance improvement when the SOPs obtained by decomposition are not uniformly distributed on the Pareto font. Different combinations of these approaches are compared based on the test problem instances proposed for the CEC 2009 competition. The set of problem instances include unconstrained and constrained MOPs with variable linkages. Experimental results are presented in the paper, and observations and discussion are also provided.

1 Introduction

Handling multi-objective optimization problems (MOPs) is a very important issue for real-world applications, because in real-world applications, there are usually two or more objectives which conflict with each other. These conflicting objectives pose a challenge for optimization algorithm developers because there is no general rule to appropriately combine these objectives into a single one and decision makers may wish to know all the possible tradeoffs that they can have. Traditionally in mathematics, the procedure to solve a multi-objective problem is to firstly transform it into a single-objective problem (SOP) by using weights on the objectives. This method makes the problem solvable by many existing, well-developed tools based on mathematics or heuristics. However, such weights oftentimes cannot be pre-determined, especially when the domain knowledge for the problem is unavailable. Furthermore, the best solution to the transformed single-objective problem is merely one solution on the Pareto front (PF) of the MOP. Hence, better optimization frameworks must be developed to fulfill the need of solving MOPs.

Due to the limitation of traditional mathematical methods for MOPs, more and more researchers try to solve MOPs in a direct way and to approximate the PF as complete as possible.

Their goal is to provide a set of solutions which are partially optimal. Many advanced multi-objective algorithms have been proposed in the literature. Some of them try to approximate the PF by using mathematical models [1, 2], and some are developed based on evolutionary algorithms [3, 4, 5, 6, 7, 8]. A hybrid framework makes use of decomposition methods in mathematics and the optimization paradigm in evolutionary computation was proposed and called MOEA/D-SBX [9]. Later, a version of MOEA/D employing the differential evolution (DE) operator, MOEA/D-DE, was proposed and shown to perform well on the MOPs with complicated Pareto set shapes [10].

MOEA/D uses a decomposition method to convert the given MOP into a set of SOPs and tries to approximate the Pareto front by solving these SOPs all together. According to the employed decomposition method, we can calculate the abstract distance between each SOP and define the neighborhood relationship. The SOPs in one neighborhood are assumed to have similar fitness landscapes, and their respective optimal solutions may probably be close to each other. The goal of this paper is to extend the framework of MOEA/D-DE and enhance the utilization of the information shared among neighbors. Firstly, we replace DE operator with a guided mutation operator for reproduction and take the SOP's neighbors, as the guided target. Secondly, we propose a new update mechanism with a priority order. The update mechanism can improve MOEA/D's performance when the SOPs obtained by decomposition are not uniformly distributed on the Pareto font. Finally, the set of test instances for the CEC 2009 competition is adopted to evaluate the performance of the various combinations of these mechanisms.

The remainder of this paper is organized as follows. Section 2 describes the formulation of multi-objective optimization problems. Section 3 introduces the main MOEA/D framework with the guided mutation operator and the priority update mechanism. Experimental results and discussion are given in section 4. Finally, section 5 concludes this paper.

2 Multi-objective Problems

Most real-world problems are multi-objective optimization problems (MOPs), of which single-objective problems are a special case. For example, in many engineering problems, there are usually at least two conflicting objectives, performance and cost. Formally, a MOP can be stated as:

$$\begin{array}{ll} \text{minimize} & F(x) = (f_1(x), \ldots, f_m(x)) \\ \text{subject to} & \begin{cases} x \in \Omega \\ C(x) = (c_1(x), \ldots, c_t(x)) \geq 0 \end{cases} \end{array}, \tag{1}$$

where Ω is called *decision space* or *variable space*, and R^m is the *objective space*. $C(x)$ represents the problem constraints and defines the feasible regions in the decision space according to problem properties [11]. $F : \Omega \rightarrow R^m$ consists of m real-valued objective functions. If Ω is a closed and connected region in R^n and all the objective functions are continuous, we call the problem a continuous MOP.

In order to consider the tradeoff between objectives, the concept of *domination* between solutions is introduced. Let $u = (u_1, \ldots, u_m)$, $v = (v_1, \ldots, v_m) \in R^m$ be two vectors. u is said to *dominate* v if $u_i \leq v_i$ for all $i = 1, \ldots, m$, and $u \neq v$. A point $x^* \in \Omega$ is *Pareto optimal* if there is no $x \in \Omega$ such that $F(x)$ dominates $F(x^*)$. The set of all the Pareto optimal points, is called the *Pareto set* (PS). The set of all the objective vectors corresponding to the PS is called the *Pareto front* (PF), where $PF = \{F(x) \in R^m | x \in PS\}$ [12].

Instead of searching for a single or just a few optimal solutions as in solving single-objective problems, the goal of handling multi-objective problems is to find the Pareto front as well as the Pareto set of the problem. Given the limited computational resource, including time and storage,

how to provide good solutions in terms of both quality and spread is the key and challenging task for multi-objective optimization.

3 Methodology

In this section, we will firstly introduce the general framework of MOEA/D for handling multi-objective optimization problems. The new operator and updating mechanism proposed in this paper for enhancing MOEA/D are then described in the following sections.

3.1 MOEA/D Framework

One of the key ideas of MOEA/D is the use of a decomposition method to transform a MOP into a number of single-objective optimization problems (SOPs). MOEA/D attempts to optimize these single-objective collectively and simultaneously instead of trying to directly approximate the Pareto front as many other evolutionary algorithms do because each optimal solution to these SOPs is a Pareto optimal solution to the given MOP. The collection of these optimal solutions is an approximation of the Pareto front. Weighted sum, Tchebycheff approach, boundary intersection, and any other decomposition approaches can serve this purpose. In the present work, the Tchebycheff approach [12] in adopted. A single-objective optimization problem obtained by decomposing the given MOP can be represented as

$$\begin{array}{ll} \text{minimize} & g(x|\lambda, z^*) = \max_{1\leq i\leq m}\{\lambda_i|f_i(x) - z_i^*\} \\ \text{subject to} & x \in \Omega \end{array} \tag{2}$$

where $\lambda = (\lambda_1, \ldots, \lambda_m)$ is a vector of weights, i.e., $\lambda_i \geq 0$ for all $i = 1, \ldots, m$ and $\sum_{i=1}^{m} \lambda_i = 1$. $z^* = (z_1^*, \ldots, z_m^*)$ is the reference point, i.e., $z_i^* = \min\{f_i(x)|x \in \Omega\}$ for each $i = 1, \ldots, m$.

Let $\lambda^1, \ldots, \lambda^N$ be a set of N weight vectors. If we use a large N and select the weight vectors properly, all the optimal solutions of the SOPs from decomposition will well approximate the Pareto front. Moreover, we can define a neighborhood relationship for each of the SOPs by computing Euclidean distances between weighted vectors. SOPs which are considered neighbors will have similar fitness landscapes and their optimal solutions should be close in the decision space. MOEA/D exploits the information sharing among subproblems which are neighbors to accomplish the optimization task effectively and efficiently. The framework of MOEA/D can be described as follows:

- Global structure:
  - a population of N search points $x^1, \ldots, x^N \in \Omega$, where x^i is the solution to the ith subproblem.
  - $FV^1, \ldots, FV^N$, where FV^i is the F-value of x^i, i.e., $FV^i = F(x^i)$ for each $i = 1, \ldots, N$.
  - $z = (z_1, \ldots, z_m)$, where z_i is the best value found so far for objective f_i.
  - a priority queue for subproblem indexes Q.
- Inputs:
  - the multi-objective problem.
  - stopping criteria.
  - N: the number of subproblems.
  - T: the number of neighbors for each subproblem.

- δ: the probability with which the parent solutions are selected from the neighborhood.
- n_r: the maximal copies of a new child in update.
- p_m: the mutation ratio for SBX in guided mutation.

- Outputs:
  - Approximation to the PS: $x^1, \ldots, x^N$.
  - Approximation to the PF: $F(x^1), \ldots, F(x^N)$.

- Procedure
  - **Step 1) Initialization**
    - **Step 1.1)** Compute the Euclidean distances between any two weight vectors and determine the T closest weight vectors to each weight vector. For each $i = 1, \ldots, N$, set $B(i) = \{i_1, \ldots, i_T\}$, where $\lambda^{i_1}, \ldots, \lambda^{i_T}$ are the T closest weight vectors to λ^i.
    - **Step 1.2)** Generate an initial population x^1, ..., x^N by uniformly randomly sampling from Ω. Set $FV^i = F(x^i)$.
    - **Step 1.3)** Insert the subproblem indexes into Q at random.
    - **Step 1.4)** Initialize $z = (z_1, \ldots, z_m)$ by setting
    $$z_j = \min_{1 \le i \le N} f_j(x^i).$$
  - **Step 2) Update**
    - **Step 2.1) Selection of targets:** Generate a random number r which is uniformly distributed in $[0, 1]$ and set
    $$P = \begin{cases} B(i) & \text{if } r < \delta, \\ \{1, \ldots, N\} & \text{otherwise.} \end{cases}$$
    - **Step 2.2) Reproduction:** Randomly select an index from P as the guided target and generate a new solution y by using guided mutation.
    - **Step 2.3) Repair:** If an element of y is out of the boundary of Ω, its value is reset to be a random value within the range.
    - **Step 2.4) Update of reference point z:** For each $j = 1, \ldots, m$, if $z_j > f_i(y)$, set $z_j = f_i(y)$.
    - **Step 2.5) Update of solutions:** Set $c = 0$ and for each index j of the subproblem in the priority queue, conduct the following steps:
      - 1) If $g(y|\lambda^j, z) \le g(x^j|\lambda^j, z)$, set $x^j = y$, $FV^j = F(y)$ and $c = c + 1$.
      - 2) If $c = n_r$, go to **Step 3**.
  - **Step 3) Stopping criteria checking** If the stopping criterion is satisfied, stop and output $x^1, \ldots, x^N$ and $F(x^1), \ldots, F(x^N)$. Otherwise, go to **Step 2**.

There are some structures to be maintained at each generation of MOEA/D, include the whole population, reference point z, and priority queue Q. Each subproblem i in the population has its own solution point x^i, fitness value FV^i, evaluation function with a weight vector λ^i, and a neighbor list $B(i)$. In step 1, solution points of generation 0 are initialized by sampling in search space at random. The corresponding fitness value and global z are determined accordingly. After initialization, the evolutionary process begins from step 2. For different operators, offspring are produced and repaired to ensure the feasibility. The whole population and reference point are updated by these newly created offspring in step 2.5. If some stoping condition is satisfied in step 3, the algorithm terminates. Otherwise, it goes back to step 2.1.

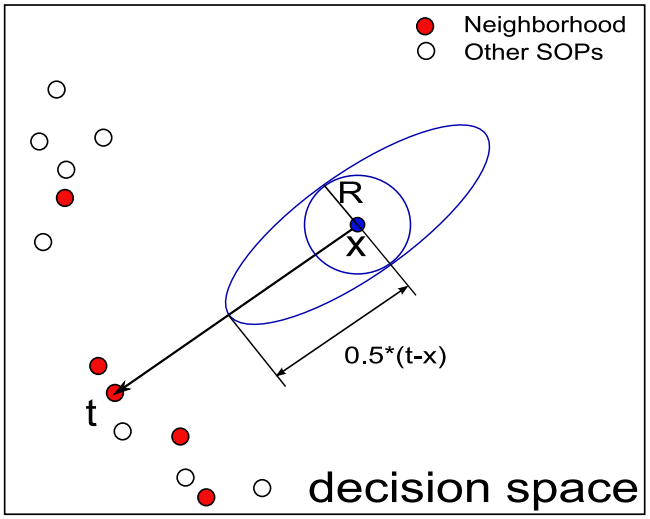


Figure 1: Illustration of guided mutation

3.2 Guided Mutation

A reproduction operator is used in step 2.2 of the MOEA/D framework. When MOEA/D was initially proposed, a simple simulated binary crossover (SBX) was adopted and implemented. [10] introduces the differential evolution (DE) operator into MOEA/D, and the integration is called MOEA/D-DE. In this paper, we attempt to replace the DE operator with guided mutation, which was proposed in [13]. The neighborhood relationship is very important in MOEA/D because it models and maintains the structure of those SOPs obtained by decomposition. Such a property is considered highly compatible with the operation of guided mutation. As aforementioned, the optimal solutions of subproblems which are neighbors will be close to each other, because their fitness landscapes are similar. It is important to choose a neighbor as the guided target when new solutions are created. At each operation of guided mutation in step 2.2, a neighbor or a subproblem t is selected from P based on the probability δ, and then a new solution y can be generate according to Equation (4).

$$\begin{aligned} & H = (H_1, \ldots, H_m) \\ \text{where} \quad & H_i = \begin{cases} N(0,1) & \text{with probability } p_m \\ 0 & \text{with probability } 1 - p_m \end{cases} \end{aligned} \tag{3}$$

$$\begin{aligned} & y = x + 0.5(t - x) \times N(0,1) + R * H \\ \text{where } & R = \begin{cases} 0.1|t - x| & \text{if } 0.1|t - x| > \mu \\ \mu & \text{otherwise.} \end{cases} \end{aligned} \tag{4}$$

New solution y is composed of three components. The first part is the current position of x. The second part is the guided vector derived from target t, and it is also the main direction of search. However, the new solution will get stuck in some local optimum if x is very close to t. For the third part, we attempt to avoid the situation by appending a simple mutation. The mutation step R here is decided by distance, $|t - x|$, and bounded by parameter μ. Figure 1 illustrates how a new solution is generated. The ellipse represents the contour line of equal probability density.

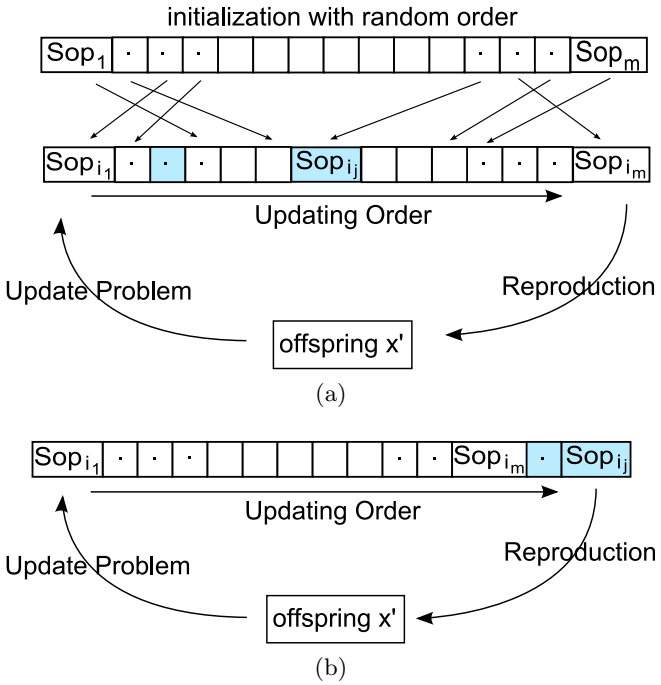


Figure 2: (a) The update queue is initialized at random, and the processing order is shown; (b) The queue state is shown after two successful updates

3.3 Priority Update

The original update mechanism in MOEA/D is to randomly pick an index from P in step 2.1 and do the update checking. Parameter n_r limits the maximal time of success updates, so it is possible that some subproblem is not updated for indefinitely many generations. Such a situation is similar to the issue of "starvation" in the process scheduling in operating systems and may lead those subproblems to be far away from other search points of the population in the decision space. The probability to update them will become lower and lower. A uniform update mechanism should be able to take care of this problem. In the present work, a new update mechanism is introduced. Initially, all subproblems are randomly allocated in a priority queue. When each time we update the best solution to a subproblem, the element at the head of the queue will be checked first and then others in the order specified by the queue. If the best solution of any subproblem is successfully updated, the subproblem will be moved back to the tail of the queue, similar to the process adopted in the priority-scheduling using in operating systems. Another difference of this mechanism lies in the selection of parent solutions. For each evaluation, we take the element at the queue tail, which is the newly updated one, as a parent, because we are more interested in the area close to the newly generated individual. Figure 2(a) shows the initialization and processing flow of priority update. Figure 2(b) displays the queue state change of Figure 2(a) after some SOPs are updated.

4 Experimental Results

For a fair comparison, a new set of test problem instances is used to verify the MOEA/D enhancements proposed in this paper. We extend MOEA/D-DE to MOEA/D-GM and adopt the new update mechanism. There are four combinations shown in Table 1 tested in the series of experiments. In section 4.1, we firstly introduce the test problem instances designed for the CEC2009 competition. Section 4.2 describes all the parameter settings used in the experiments

Table 1: Combinations of different operators and update mechanism

Legend	Description
DE	MOEA/D with differential evolution [10]
GM	MOEA/D with guided mutation
QDE	MOEA/D-DE + priority update
QGM	MOEA/D-GM + priority update

and provides some guidelines of these parameters. Section 4.3 shows the experimental results, and finally some discussion are given in section 4.4.

4.1 Test Instances of the CEC 2009 Competition

In order to develop robust, effective, and efficient evolutionary algorithms to solve multi-objective optimization problems, we are in need of various problem models. There have been several test problem sets proposed in the literature. Continuous multi-objective problems with geometrical shape of the Pareto set were widely used in early days [14, 15, 16]. However, most of their PS shapes are strikingly simple. Some researchers started to consider that the test instances with more complicated PS shapes are necessary for simulating real-world problems and that variable linkages should be introduced into the test instances [17, 18, 19]. For the CEC 2009 competition, Zhang et al. propose a new set of test problem instances, including constrained and unconstrained problems with complicated PS shapes [20]. Each problem explicitly defines the objective functions, variable dimensions, search spaces, and constraint conditions. By adopting the benchmark, we compare different approaches proposed in the study in terms of a performance metric called *IGD* value. Let P^* be a set of points uniformly distributed on the Pareto front and A be the approximation obtained by the algorithm. IGD represents the average distance from P^* to A and is defined as

$$IGD(A, P^*) = \frac{\sum_{v \in P^*} d(v, A)}{|P^*|} , \tag{5}$$

where $d(v, A)$ is the minimum Euclidean distance between v and the points in A. If the points in set P^* can appropriately represents the Pareto front, IGD can measure both the diversity and convergence of set A. Obviously, set A with a low IGD value must closely fit the Pareto front and may not miss any part of the whole Pareto front.

4.2 Parameter Settings

All control parameters in the MOEA/D framework are listed in Table 2. Suggestion values are also provided. T defines the number of neighbors of one SOP, and the value is highly dependent on the decomposition size N as well as the shape of the Pareto front. According to the assumption, the fitness landscapes of neighbors should be similar. If a larger N is adopted and the SOPs distributes among the PF uniformly, there are more SOPs with similar problem structures. We can use a larger T value for promoting information sharing. δ is the probability used to decide whether the candidate of parents comes from only the neighbors or from all the SOPs. In other words, it controls the weight of exploitation and exploration. Mutation rate p_m is a control factor for simulated binary crossover. It indicates probability that each variable x_i is changed or not. We set the value $1/M$ for expecting at least one dimension is mutated in a uniform way. n_r limits the maximum time of the successful updates. Another effect of n_r is

Table 2: General settings for the parameters

Parameter	Value
T	$0.1N$
δ	0.9
p_m	$1/M$
n_r	2
μ	0.005

Table 3: IGD values for the unconstrained problems in 30 independent runs (mean/standard deviation)

Instance	DE	GM	QDE	QGM
UF1	0.00521 / 0.00036	0.00633 / 0.00102	0.00533 / 0.00047	0.00615 / 0.00113
UF2	0.01252 / 0.00231	0.00671 / 0.00078	0.01058 / 0.00131	0.00643 / 0.00043
UF3	0.02302 / 0.01221	0.05611 / 0.06593	0.00942 / 0.00622	0.04293 / 0.03407
UF4	0.07161 / 0.00618	0.05167 / 0.00262	0.07234 / 0.00634	0.04756 / 0.00222
UF5	0.34555 / 0.16121	1.43955 / 0.31818	0.47337 / 0.10724	1.79191 / 0.51240
UF6	0.40846 / 0.20139	0.47195 / 0.19595	0.51346 / 0.19437	0.55634 / 0.14701
UF7	0.01916 / 0.05794	0.00773 / 0.00091	0.06546 / 0.14380	0.00755 / 0.00094
UF8	0.11889 / 0.02048	0.13253 / 0.04404	0.13275 / 0.03548	0.24456 / 0.08542
UF9	0.16424 / 0.03586	0.17016 / 0.04411	0.17394 / 0.02549	0.18784 / 0.02872
UF10	0.49816 / 0.08300	0.50213 / 0.06963	0.52501 / 0.08757	0.56460 / 0.10165

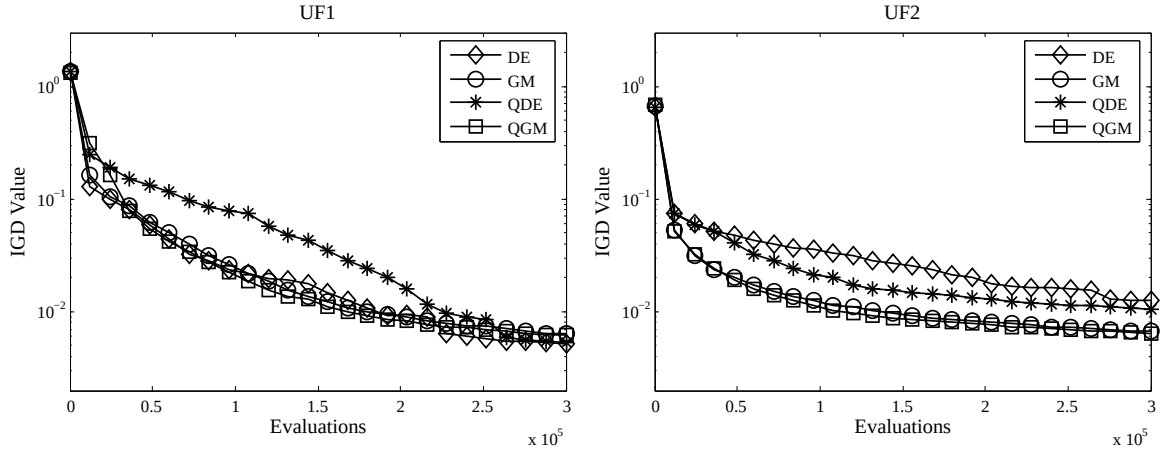
to control the number of copies of the offspring in next generation. Clearly, a large n_r leads to fast convergence, and the population will probably lose diversity in a sort time. μ defines the lower bound of the step size of guided mutation. The functionality of μ is to avoid ineffective searching in a local region when the guided target is very close to the parent.

4.3 Results

The detailed definition of the CEC 2009 multi-objective test instances is presented in [20], and the set $P^* \in PF$ for IGD calculation is also available. Tables 3 and 4 record the final results after 300,000 function evaluations in 30 independent runs. The first value is the mean and the second value is the standard deviation. Furthermore, we show the evolution of IGD values in Figures 3 and 4. The computing environment is a Windows PC with 1.8Ghz AMD CPU and 2GB RAM. Both the algorithm and test instances are programmed in C++. We also compare the CPU time in Table 5.

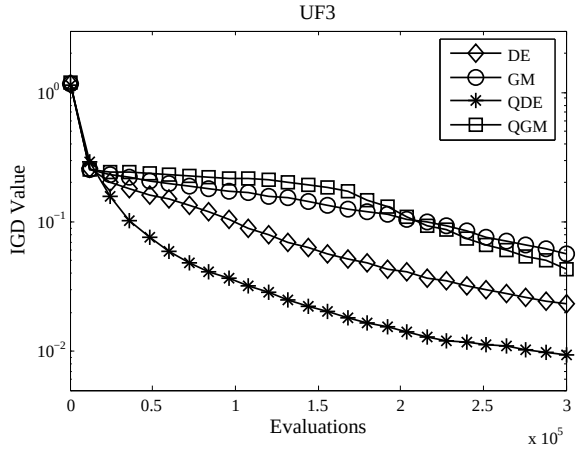
4.4 Discussion

In this section, we will discuss some observations of the results. There are four combinations with different operators and update methods. None of them outperforms the others on all test instances. In the test set, each test instance has very different properties as well as PS shapes. Some instances are well handled by MOEA/D-DE, but some are better handled by MOEA/D-GM. The main difference of GM from DE is that GM can generate offspring in a broader region.

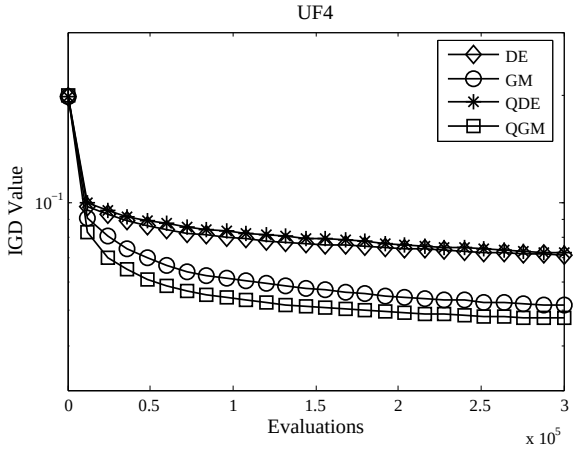


(a) UF1

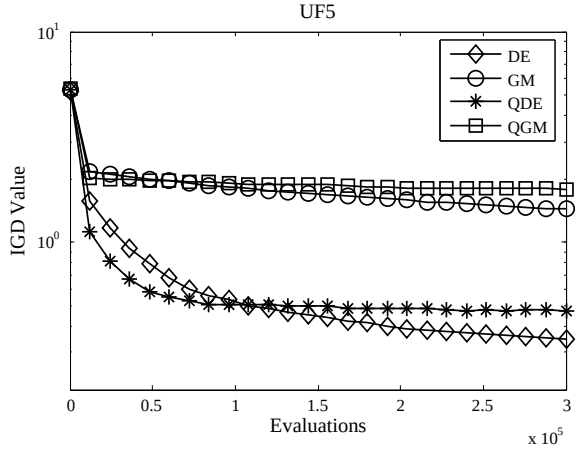
(b) UF2



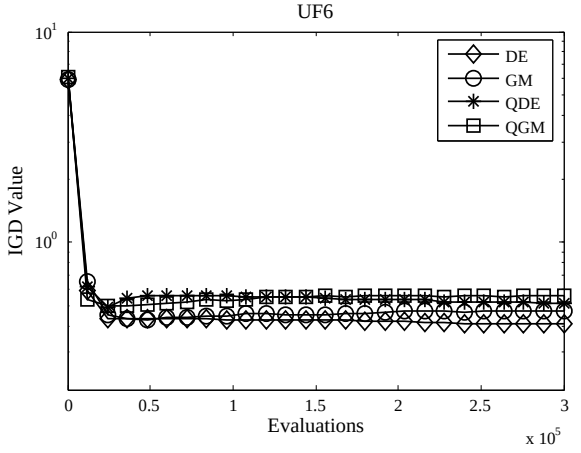
(c) UF3



(d) UF4

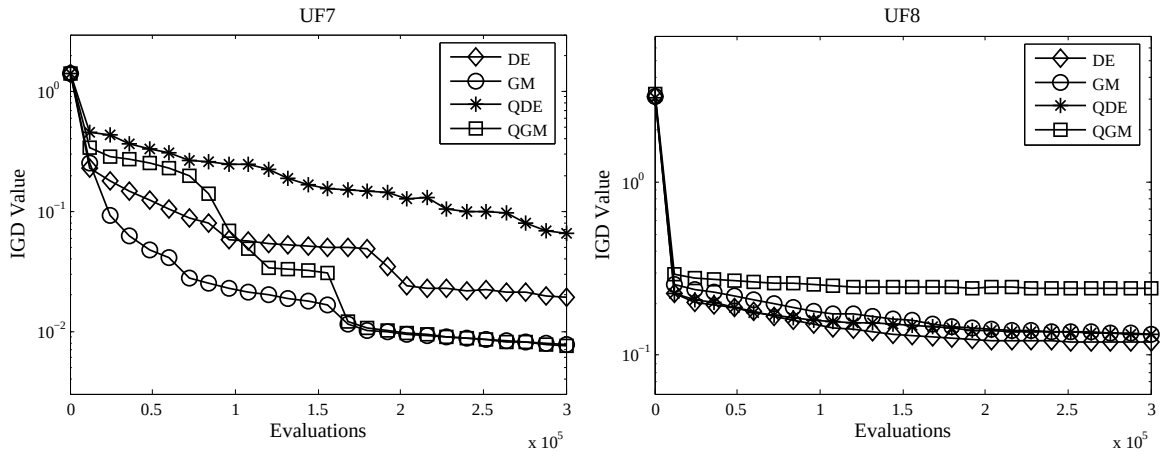


(e) UF5



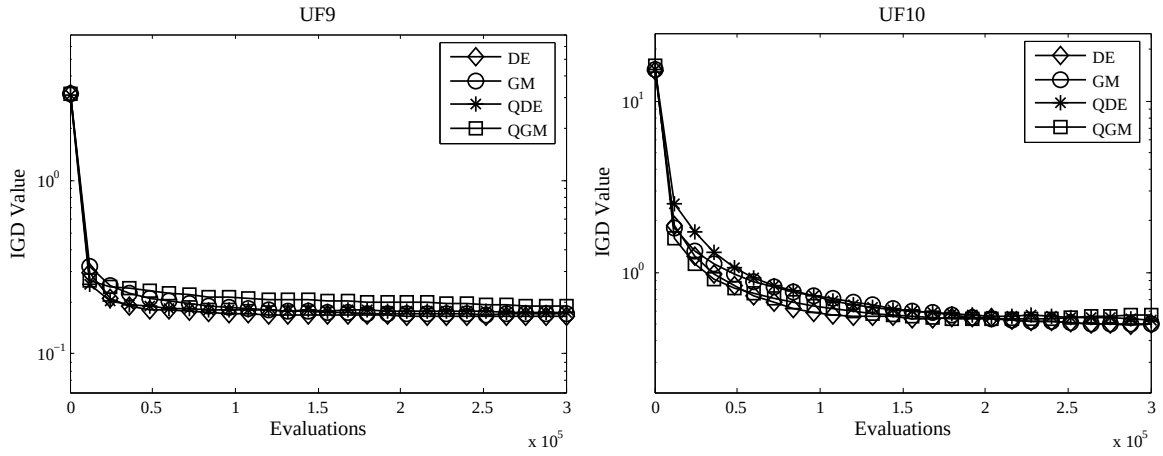
(f) UF6

Figure 3: Mean IGD values for the unconstrained problems in 30 independent runs



(g) UF7

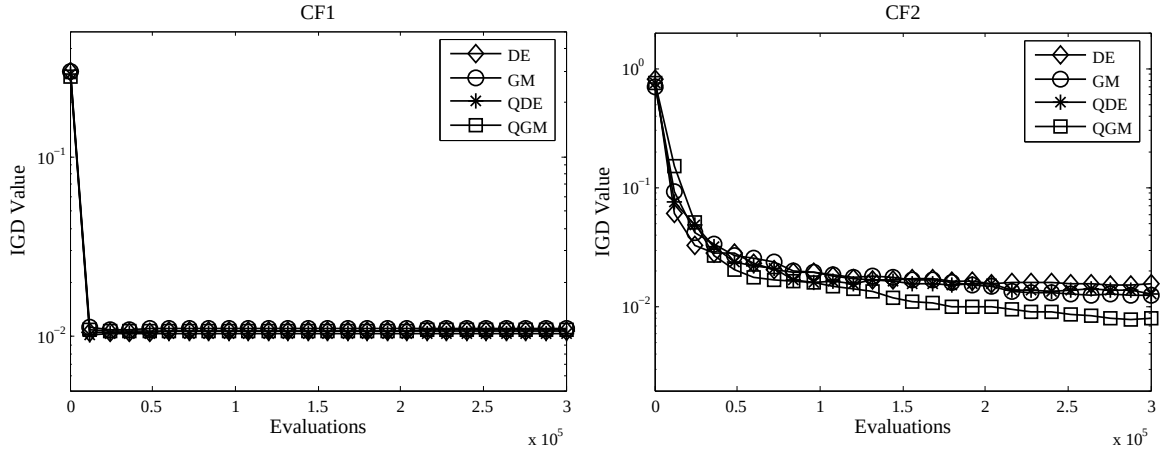
(h) UF8



(i) UF9

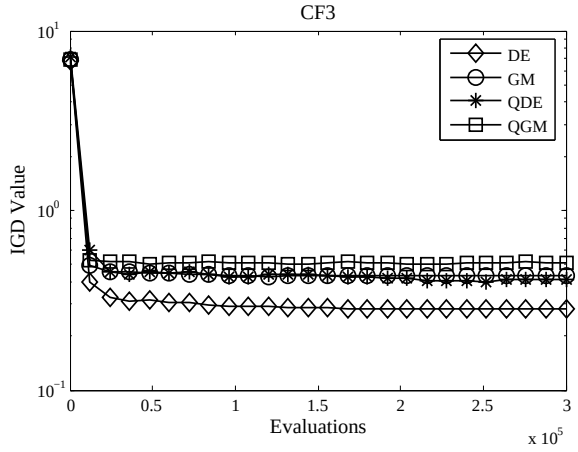
(j) UF10

Figure 3: Mean IGD values for the unconstrained problems in 30 independent runs

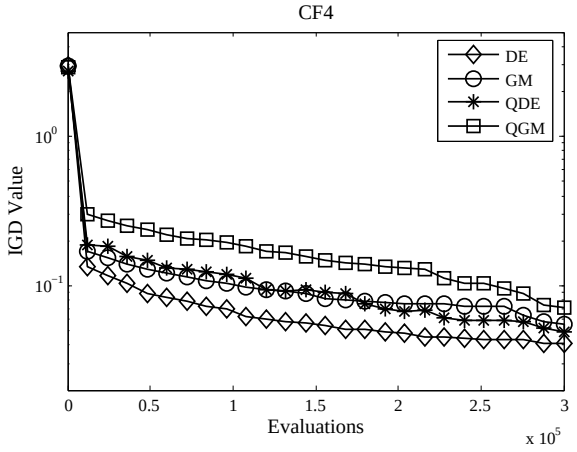


(a) CF1

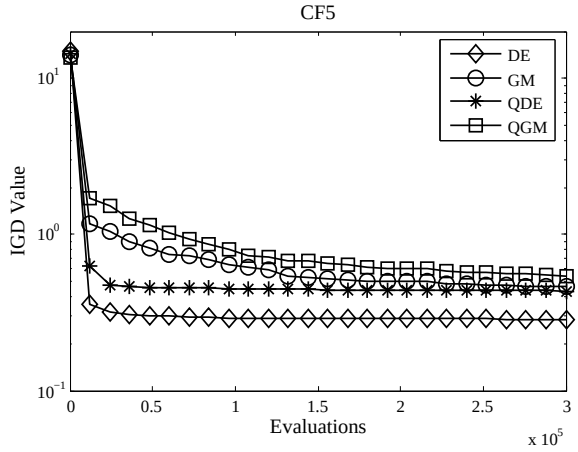
(b) CF2



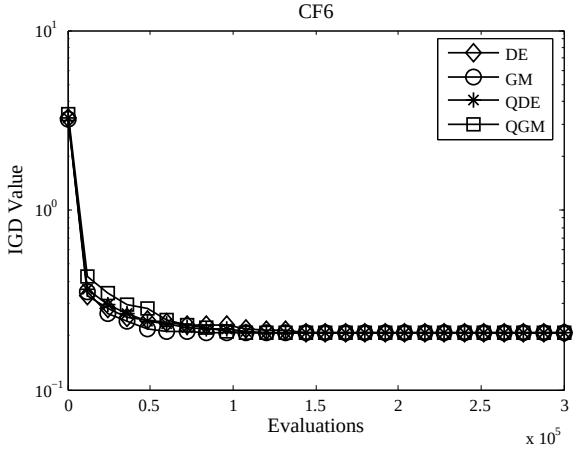
(c) CF3



(d) CF4

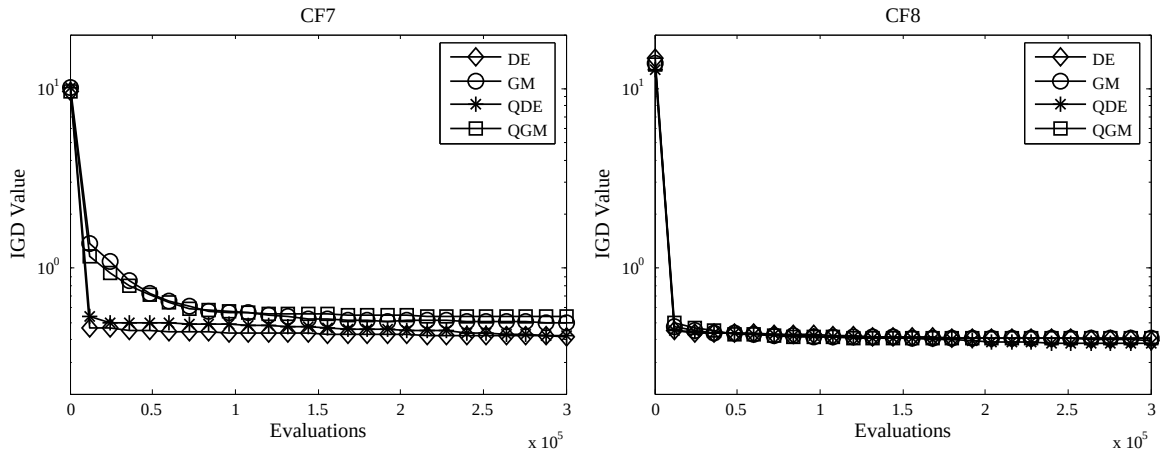


(e) CF5

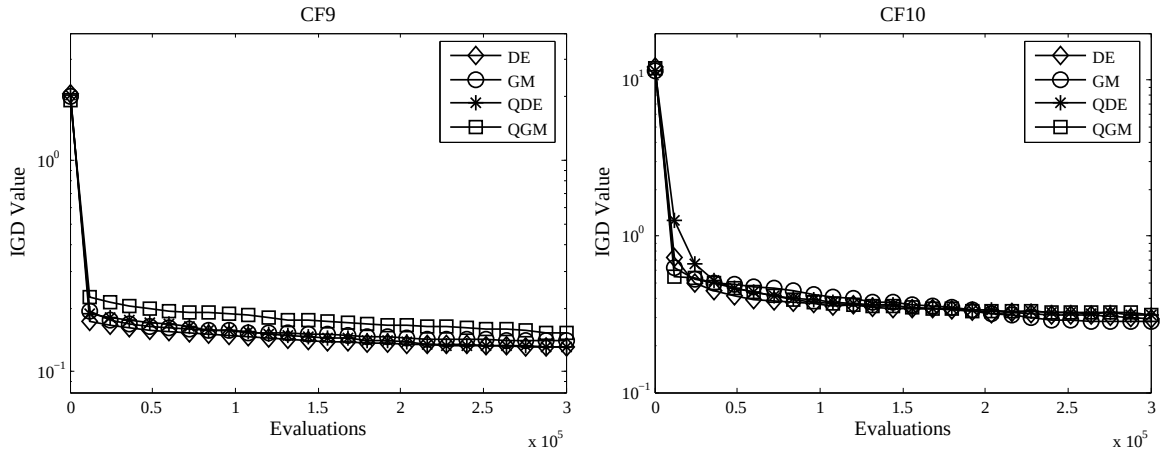


(f) CF6

Figure 4: Mean IGD values for the constrained problems in 30 independent runs



(g) CF7 (h) CF8



(i) CF9 (j) CF10

Figure 4: Mean IGD values for the constrained problems in 30 independent runs

Table 4: IGD values for the constrained problems in 30 independent runs (mean/standard deviation)

Instance	DE	GM	QDE	QGM
CF1	0.01064 / 0.00304	0.01103 / 0.00279	0.01035 / 0.00297	0.01081 / 0.00250
CF2	0.01555 / 0.01054	0.01264 / 0.01318	0.01274 / 0.01039	0.00800 / 0.00999
CF3	0.28376 / 0.13444	0.43359 / 0.11548	0.40955 / 0.15647	0.51344 / 0.07143
CF4	0.04085 / 0.02022	0.05561 / 0.05579	0.04881 / 0.04749	0.07075 / 0.10144
CF5	0.28718 / 0.13048	0.46447 / 0.18780	0.43485 / 0.12327	0.54456 / 0.17231
CF6	0.20718 / 0.00014	0.20705 / 0.00010	0.20719 / 0.00013	0.20712 / 0.00010
CF7	0.41195 / 0.10696	0.50025 / 0.09402	0.41989 / 0.15710	0.53558 / 0.10030
CF8	0.39420 / 0.10749	0.40612 / 0.11071	0.38222 / 0.09630	0.40564 / 0.12824
CF9	0.13038 / 0.01364	0.14077 / 0.01493	0.13034 / 0.01664	0.15193 / 0.04125
CF10	0.29824 / 0.08157	0.28541 / 0.08603	0.31539 / 0.12300	0.31387 / 0.10384

GM embeds a random factor in the mutation step and appends a simple mutation for the other dimensions. GM is therefore more flexible to approximate the PS as a complicated curve like instance UF2, UF4, and UF7. For UF3, the PS shape is a simple curve, and DE can efficiently handle this problem. For the update mechanism proposed in this paper, according the results, we can find that priority update also helps MOEA/D to get better performance in certain test instances. From observations, MOEA/D has a weakness caused by decomposition when the SOPs are not uniformly distributed on the Pareto front. Figure 5 illustrates such a situation. The distances between these SOPs are not similar, even if we uniformly divide the weights of objectives. Moreover, the difficulty degree of these SOPs will also not be identical. There are two points in the update. One is to update the population with a priority order, and the other is to take the latest individual as the parent in reproduction. If a SOP which is rarely updated, it stays in front of the queue and gets more chances to be verified by a new child. Sometimes a few SOPs are so simple that they got the optimal solutions and will not be updated any more. Only the latest individual can reproduce offspring in this situation. Computational resource will be automatically re-allocated and each SOP costs different numbers of evaluations based on the difficulty. We can observe the situation in Table 5 as the additional cost of new updates takes more CPU time.

5 Conclusions

This paper proposed extensions of the MOEA/D framework. We used the guided mutation operator as the reproduction method to replace differential evolution in MOEA/D-DE. Guided mutation makes use of neighborhood information efficiently. We also modified the update step and proposed a new method utilizing a priority order implemented as a queue structure. Different approaches were implemented and tested for performance on the test instances designed for the CEC 2009 competition. The experimental results indicated that none of the combinations outperforms the others on all problems. MOEA/D-GM gained some performance improvement on the problems with a curvy Pareto set. Priority update enhanced the ability of the MOEA/D framework to handle the non-uniform distribution of SOPs on the Pareto set. It is a critical issue because we have no idea or information of the Pareto set before attempting to solve the problem. MOEA/D uses an identical way to decompose all MOPs with different Pareto set shapes. Furthermore, there are still many test instances difficult to handle, especially the

Table 5: Mean CPU time for each test instance (seconds)

Instance	DE	GM	QDE	QGM
UF1	6.507	5.468	13.052	12.159
UF2	7.083	6.062	13.870	12.915
UF3	7.657	7.373	14.320	13.387
UF4	7.135	6.062	13.941	12.921
UF5	6.638	5.567	12.062	12.246
UF6	7.309	6.630	13.342	12.980
UF7	6.847	6.454	13.007	13.313
UF8	15.212	19.658	28.196	27.069
UF9	14.601	19.475	28.419	27.343
UF10	15.418	16.380	28.466	28.139
CF1	5.024	4.145	11.887	10.793
CF2	5.277	4.469	12.268	11.063
CF3	5.454	4.606	11.774	10.873
CF4	5.436	4.395	11.989	10.949
CF5	5.460	4.517	12.078	11.188
CF6	5.453	4.563	12.143	11.090
CF7	5.456	4.557	11.716	10.852
CF8	7.596	9.492	24.706	24.170
CF9	10.015	11.891	28.658	28.512
CF10	9.593	11.538	27.895	27.226

ones with discontinuous Pareto fronts and the constrained problems. In such cases, simple decomposition techniques like Tchebycheff are not applicable. Hence, along this line, developing flexible decomposition methods should be considered an important future research direction.

Acknowledgments

The authors are grateful to the National Center for High-performance Computing for computer time and facilities.

References

[1] M. M. Wiecek, W. Chen, and J. Zhang, "Piecewise quadratic approximation of the non-dominated set for bi-criteria programs," *Journal of Multi-Criteria Decision Analysis*, vol. 10, no. 1, pp. 35–47, 2001.

[2] S. Ruzika and M. Wiecek, "Approximation methods in multiobjective programming," *Journal of Optimization Theory and Applications*, vol. 126, no. 3, pp. 473–501, 2005.

[3] G. B. Lamont and D. A. V. Veldhuizen, *Evolutionary Algorithms for Solving Multi-Objective Problems*. Kluwer Academic Publishers, 2002.

[4] K. C. Tan, E. F. Khor, and T. H. Lee, *Multiobjective Evolutionary Algorithms and Applications (Advanced Information and Knowledge Processing)*. Springer-Verlag, 2005.

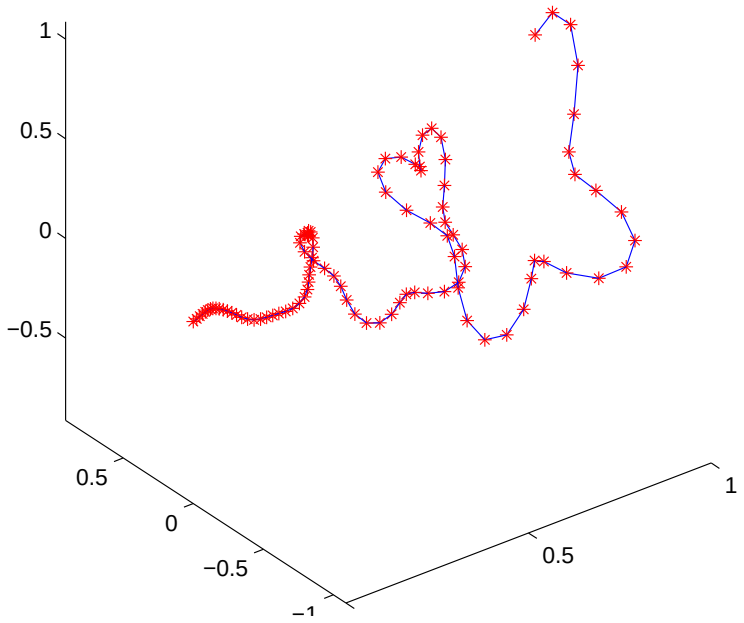


Figure 5: Pareto set of UF2 with non-uniform decomposition

[5] C. A. Coello Coello, "An updated survey of ga-based multiobjective optimization techniques," *ACM Computing Surveys*, vol. 32, no. 2, pp. 109–143, 2000.

[6] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[7] H. Lu and G. G. Yen, "Rank-density-based multiobjective genetic algorithm and benchmark test function study," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 4, pp. 325–343, 2003.

[8] C. A. Coello Coello, G. T. Pulido, and M. S. Lechuga, "Handling multiple objectives with particle swarm optimization," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 256–279, 2004.

[9] Q. Zhang and H. Li, "MOEA/D: A multiobjective evolutionary algorithm based on decomposition," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 6, pp. 712–731, 2007.

[10] H. Li and Q. Zhang, "Multiobjective optimization problems with complicated pareto set, MOEA/D and NSGA-II," *IEEE Transactions on Evolutionary Computation*, 2008, in press.

[11] K. Deb, A. Pratap, and T. Meyarivan, "Constrained test problems for multi-objective evolutionary optimization," in *First International Conference on Evolutionary Multi-Criterion Optimization*. Springer Verlag, 2001, pp. 284–298.

[12] K. Miettinen, *Nonlinear Multiobjective Optimization*. Kluwer Academic, 1999.

[13] C.-T. Hsieh, C.-M. Chen, and Y.-p. Chen, "Particle swarm guided evolution strategy," in *Proceedings of ACM SIGEVO Genetic and Evolutionary Computation Conference 2007 (GECCO-2007)*, 2007, pp. 650–657.

[14] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, "Scalable multi-objective optimization test problems," in *Proceedings of IEEE Congress on Evolutionary Computation (CEC 2002)*, 2002, pp. 825–830.

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

[16] V. L. Huang, A. K. Qin, K. Deb, E. Zitzler, P. N. Suganthan, J. J. Liang, M. Preuss, and S. Huband, "Problem definitions for performance assessment of multi-objective optimization algorithms: Special session on constrained real-parameter optimization," Nanyang Technological University, Singapore," Technical Report, 2007.

[17] K. Deb, A. Sinha, and S. Kukkonen, "Multi-objective test problems, linkages, and evolutionary methodologies," in *Proceedings of ACM SIGEVO Genetic and Evolutionary Computation Conference 2006 (GECCO-2006)*, 2006, pp. 1141–1148.

[18] H. Li and Q. Zhang, "A multiobjective differential evolution based on decomposition for multiobjective optimization with variable linkages," in *Proceedings of the 9th International Conference on Parallel Problem Solving from Nature (PPSN IX)*, 2006, pp. 583–592.

[19] Q. Zhang, A. Zhou, and Y. Jin, "RM-MEDA: A regularity model-based multiobjective estimation of distribution algorithm," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 1, pp. 41–63, 2008.

[20] Q. Zhang, A. Zhou, S. Zhao, P. N. Suganthan, W. Liu, and S. Tiwari, "Multiobjective optimization test instances for the CEC 2009," Department of Computing and Electronic Systems, University of Essex, UK," Working Report, 2008.