Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2015, Article ID 126404, 17 pages
http://dx.doi.org/10.1155/2015/126404

Research Article

Multiobjective Particle Swarm Optimization Based on PAM and Uniform Design

Xiaoshu Zhu, Jie Zhang, and Junhong Feng

School of Computer Science and Engineering, Guangxi Universities Key Lab of Complex System Optimization and Big Data Processing, Yulin Normal University, Yulin 537000, China

Correspondence should be addressed to Jie Zhang; jgxyzjzj@126.com and Junhong Feng; jgxyfjh@126.com

Received 26 November 2014; Accepted 3 March 2015

Academic Editor: Pandian Vasant

In MOPSO (multiobjective particle swarm optimization), to maintain or increase the diversity of the swarm and help an algorithm to jump out of the local optimal solution, PAM (Partitioning Around Medoid) clustering algorithm and uniform design are respectively introduced to maintain the diversity of Pareto optimal solutions and the uniformity of the selected Pareto optimal solutions. In this paper, a novel algorithm, the multiobjective particle swarm optimization based on PAM and uniform design, is proposed. The differences between the proposed algorithm and the others lie in that PAM and uniform design are firstly introduced to MOPSO. The experimental results performing on several test problems illustrate that the proposed algorithm is efficient.

1. Introduction

Many real-world optimization problems often need to simultaneously optimize multiple objectives that are incommensurable and generally conflicting with each other. They can usually be written as

$$\min_{X\in\Omega}\{f_1(X), f_2(X), \ldots, f_M(X)\}, \quad (1)$$

where $X = (x_1, x_2, \ldots, x_N)$ is a variable vector in a real and N-dimensional space, Ω is the feasible solution space, and M is the number of the objective functions. Since the pioneering attempt of Schaffer [1] to solve multiobjective optimization problems, many kinds of multiobjective evolutionary algorithms (MOEAs), ranging from traditional evolutionary algorithms to newly developed techniques, have been proposed and widely used in different applications [2–4].

Multiobjective evolutionary algorithms, MOEAs, have become well-known methods for solving the multiobjective optimization problems that are too complex to be solved by exact methods. The main challenge for MOEAs is to be satisfied with three goals at the same time: (1) the Pareto optimal solutions are as near to true Pareto front, which means the convergence of MOEAs, (2) the nondominated solutions are evenly scattered along the Pareto front, which means the diversity of MOEAs, and (3) MOEAs obtain Pareto optimal solutions in limited evolution times [5].

The particle swarm optimization algorithm, PSO, and MOEAs are both intelligent optimization algorithms. It was proposed by Eberhart and Kennedy in 1995 [6, 7]. It originates from sharing and exchanging of information in the process of searching food among the bird's individuals. Each individual can benefit from the discovery and flight experience of the others. PSO seems particularly suitable for multiobjective optimization mainly because of the high speed of convergence [8, 9].

PAM is one of k-medoids clustering algorithms based on partitioning methods. It attempts to divide data objects into k partitions. Namely, it can divide a swarm into k different subswarms with different features.

This paper proposed a novel multiobjective particle swarm optimization based on PAM and uniform design, abbreviated as UKMOPSO. It first uses PAM to partition the data points into several clusters, and then the smallest cluster is implemented crossover operator based on the uniform design to generate some new data points. When the size of the Pareto solution is larger than the size of the external archive, PAM is used to determine which Pareto solution is to be

removed or appended. The results of the experimental simulation implemented on several well-known test problems indicate that the proposed algorithm is efficient.

The rest of this paper is organized as follows. Section 2 states the preliminaries of the proposed method. Section 3 presents our method in detail. Section 4 gives the numerical results of the proposed method. The conclusion of the work is made in Section 5.

2. Preliminaries

In this section, we describe some concepts concerning particle swarm optimization, multiobjective particle swarm optimization, PAM, and uniform design.

2.1. Particle Swarm Optimization Algorithms. In d-dimensional search space, the position and velocity of the ith particle are, respectively, represented as $X_i = [x_{i,1}, x_{i,2}, \ldots, x_{i,d}]$ and $V_i = [v_{i,1}, v_{i,2}, \ldots, v_{i,d}]$. The optimal positions of the ith particle and the whole swarm, namely, the individual optimal and the global optimal, are denoted as $Pbest = [p_{i,1}, p_{i,2}, \ldots, p_{i,d}]$ and $Gbest = [p_{g,1}, p_{g,2}, \ldots, p_{g,d}]$, respectively. The individuals or particles in the swarm update their velocities and positions according to the following formulas:

$$v_{i,j}(k+1) = w \cdot v_{i,j}(k) + c_1 \cdot r_1 \cdot \left(Pbest_{i,j}(k) - x_{i,j}(k)\right) + c_2 \times r_2 \times \left(Gbest_j(k) - x_{i,j}(k)\right), \tag{2}$$

$$x_{ij}(k+1) = x_{ij}(k) + v_{ij}(k+1), \tag{3}$$

where the inertia weight coefficient w indicates the ability to maintain the previous speed; the acceleration coefficients c_1 and c_2 are used to coordinate the degrees of tracking individual optimal and global optimal; and r_1 and r_2 are two random numbers drawn from the uniform distribution on the interval $(0, 1)$.

The update equation of the velocity consists of the previous velocity component, a cognitive component and a social component. They are mainly controlled by three parameters: the inertia weight and two acceleration coefficients.

From the theoretical analysis for the trajectory of particles in PSO [10], the trajectory of a particle X_i converges to a weighted mean of P_i and P_g. Whenever the particle converges, it will "fly" to the individual best position and the global best position. According to the update equation, the individual best position of the particle will gradually move closer to the global best position. Therefore, all the particles will converge onto the global best particle's position.

2.2. Multiobjective Particle Swarm Optimization. MOPSO is proposed by Coello et al.; it adopts swarm intelligence to optimize MOPs, and it uses the Pareto optimal set to guide the particle's flight [9].

Particle swarm optimization has been proposed for solving a large number of single objective problems. Many researchers are interested in solving multiobjective problems (MOP) using PSO. To modify a single objective PSO to MOPSO, a guide must be redefined in order to obtain a set of nondominated solutions (Pareto front). In MOPSO, the Pareto optimal solutions should be used to determine the guide for each particle. How to select suitable local guides for attaining both convergence and diversity of solutions becomes an important issue.

There have been some publications to use PSO to solve MOP. A dynamic neighborhood PSO was proposed [11], which optimizes only one objective at a time and uses a scheme similar to lexicographic ordering. In addition, this approach also proposes an unconstrained elite archive named dominated tree to store the nondominated solutions. However, it is a difficult issue for this approach to pick up a best local guide from the set of Pareto optimal solutions for each particle of the population. A strategy for finding suitable local guides for each particle was proposed and named Sigma method. The local guide is explicitly assigned to specify particles according to the Sigma value [12]. This results in the desired diversity and convergence, but it is still not close enough to the Pareto front. On the other hand, an enhanced archiving technique to maintain the best (nondominated) solutions found during the course of a MO algorithm was proposed [13]. It shows that using archives in PSO for MO problems will improve their performance directly. A parallel vector evaluated particle swarm optimization (VEPSO) method for multiobjective problems was proposed [14], which adopted a ring migration topology and PVE system to simultaneously work 2–10 CPUs to find nondominated solutions. In [9], MOPSO method was proposed. It incorporates Pareto dominance and a special mutation operator to solve MO problems [15].

Recently, a hybrid multiobjective algorithm combining both genetic algorithm (GA) and particle swarm optimization (PSO) was proposed [16]. A multiobjective particle swarm optimization based on self-update and grid strategy was proposed for improving the function of Pareto set [17]. A new dynamic self-adaptive multiobjective particle swarm optimization (DSAMOPSO) method is proposed to solve binary-state multiobjective reliability redundancy allocation problems [18], which used a modified nondominated sorting genetic algorithm (NSGA-II) method and a customized time-variant multiobjective particle swarm optimization method to generate nondominated solutions.

The MOPSO method is becoming more popular due to its simplicity to be implemented and its ability to quickly converge to a reasonably acceptable solution for problems in science and engineering.

2.3. PAM Algorithm. There are many clustering methods available in data mining. Typical clustering analysis methods are clustering based on partition, hierarchical clustering, clustering based on density, clustering based on grid, and clustering based on model.

The most frequently used clustering methods based on partition are k-means and k-medoids. In contrast to the k-means algorithm, k-medoids chooses data points as centroids, which make k-medoids method more robust than k-means in the presence of noise and outliers. The reason is that k-medoids method is less influenced by outliers or other

Algorithm: PAM, a k-medoids algorithm for partitioning based on medoid or central objects.
Input:
k: the number of clusters,
D: a data set containing n objects.
Output: A set of k clusters.
Method:
(1) Arbitrarily choose k objects in D as the initial representative objects or seeds;
(2) Repeat
(3) Assign each remaining object to the cluster with the nearest representative object;
(4) Randomly select a nonrepresentative object, o random;
(5) Compute the total cost, S, of swapping representative object, o_j, with o random;
(6) if $S < 0$ then swap o_j with o random to form the new set of k representative objects;
(7) Until no change;

Algorithm 1: PAM algorithm.

extreme values than k-means. PAM (Partitioning Around Medoids) is the first and the most frequently used k-medoids algorithms. It is shown in Algorithm 1.

PAM constructs k partitions (clusters) of the given dataset, where each partition represents a cluster. Each cluster may be represented by a centroid or a cluster representative which is some sort of summary description of all the objects contained in a cluster. It needs to determine k partitions for n objects. The process of PAM is by and large as follows. Firstly, randomly select k representative objects, and cluster other objects to the same group as the representative object according to the minimum distances between the representative object and other objects. Then try to replace these representative objects with other nonrepresentative objects in order to minimize squared error. All the possible pairs of objects are analyzed, where one object in each pair is considered a representative object and the other is not. The total cost of the clustering is calculated for each such combination. An object will be replaced with such an object having minimized squared error. The set of best objects in each cluster after iteration forms the representative objects for the next iteration. The final set of representative objects is the respective centroids of the clusters [19–22].

2.4. Uniform Design

2.4.1. Uniform Design. The main objective of uniform design is to sample a small set of points from a given set of points such that the sampled points are uniformly scattered [23].

Let there be n factors and q levels per factor. When n and q are given, the uniform design selects q from q^n possible combinations, such that these q combinations are uniformly scattered over the space of all possible combinations. The selected q combinations are expressed as a uniform array $U(n, q) = [U_{i,j}]q \times n$, where $U_{i,j}$ is the level of the jth factor in the ith combination and can be calculated by the following formula

$$U_{i,j} = \left(i\sigma^{j-1} \bmod q\right) + 1, \tag{4}$$

where σ is a parameter given in Table 1.

2.4.2. Improved Generation of Initial Population. An algorithm for dividing the solution space and an algorithm for generating the initial population have been designed [23]. However, Algorithm 2 in [23] considers only the dividing of the solution space, not the dividing of the N-dimension space. This will bring about some serious problems. If we assume that, in Step 2, $Q_0 = 5$ and $N = 10$, then $U(N, Q_0)$ is impossible to be generated because Q_0 must be larger than N. Namely, Algorithm 2 in [23] is only suitable for the low dimensional problem. In order to overcome the shortcomings, the dividing of the N-dimension space is introduced to improve the algorithm. The improved algorithm can be suitable for not only low dimensional but also high dimensional problem. The improved algorithm is shown as follows.

Algorithm A (improved generation of initial population)

Step 1. Judge whether Q_0 is valid or not, as it must be found in the 1st column of Table 1. If it is not valid, it stops and shows error messages, otherwise it continues.

Step 2. Execute Algorithm 1 in [23] to divide $[l, u]$ into S subspaces $[l(1), u(1)], [l(2), u(2)] \cdots [l(S), u(S)]$.

Step 3.1. Judge whether Q_0 is more than N, if yes, turn to Step 3.2, otherwise turn to Step 3.3.

Step 3.2. Quantize each subspace, and apply the uniform array $U(N, Q_0)$ to sample Q_0 points.

Step 3.3. Divide N-dimension space into $\lfloor N/N_0 \rfloor$ parts, where N_0 is an integer more than 1 and less than Q_0 and is generally taken as $Q_0 - 1$. Among $\lfloor N/N_0 \rfloor$ parts, the 1st part corresponds to the dimension from 1 to N_0, and the 2nd part corresponds to the dimension from $N_0 + 1$ to $2 * N_0$, and so forth. If the remainder R of N/N_0 is not equal to 0, then a plus part corresponds to the dimension from $\lfloor N/N_0 \rfloor * N_0 + 1$ to N, whose length is surely less than N_0. Repeat to execute Step 3.4 for each part.

Step 3.4. Quantize each subspace, and then apply uniform array $U(N_0, Q_0)$ to sample Q_0 points. It is noteworthy that

Table 1: Values of the parameter σ for different number of factors and different number of levels per factor.

Number of levels per factors	Number of factors	σ
5	2~4	2
7	2~6	3
11	2~10	7
13	2	5
	3	4
	4~12	6
17	2~16	10
19	2~3	8
	4~18	14
23	2, 13~14, 20~22	7
	8~12	15
	3~7, 15~19	17
29	2	12
	3	9
	4~7	16
	8~12, 16~24	8
	13~15	14
	25~28	18
31	2, 5~12, 20~30	12
	3~4, 13~19	22

$U(N_0, Q_0)$ is replaced with $U(R, Q_0)$ in the plus part, where the remainder R is equal to $N - \lfloor N/N_0 \rfloor * N_0$.

2.4.3. Crossover Operator Based on the Uniform Design. The crossover operator based on the uniform design acts on two parents. It quantizes the solution space defined by these parents into a finite number of points, and then it applies the uniform design to select a small sample of uniformly scattered points as the potential offspring.

For any two parents p_1 and p_2, their minimal values and the maximal values of each dimension are used to form the novel solution space $[l_{\text{parent}}, u_{\text{parent}}]$. Each domain of $[l_{\text{parent}}, u_{\text{parent}}]$ is quantized into Q_1 levels, where Q_1 is a predefined prime number. Then the uniform design is applied to select a sample of points as the potential offspring. The details of the algorithm can be referred to [23].

3. The Proposed Algorithm

3.1. Elitist Selection or Elitism [4]. Elitism means that elite individuals cannot be excluded from the mating pool of the population. A strategy presented can always include the best individual of the current population in the next generation in order to prevent the loss of good solutions found so far. This strategy can be extended to copy the best n individuals to the next generation. This is explanation of the elitism. In MOP, elitism plays an important role.

Two strategies are often used to implement elitism. One maintains elitist solutions in the population; the other stores elitist solutions into an external secondary list or external archive and reintroduces them to the population. The former copies all nondominated solutions in the current population to the next population and then fills the rest of the next population by selecting from the remaining dominated solutions in the current population. The latter uses an external secondary list or external archive to store the elitist solutions. External list stores the nondominated solutions found. It will be updated in the next generation by means of removing elitist solutions dominated by a new solution or adding the new solution if it is not dominated by any existing elitist solution.

```
population paretocreate (pop)
Input: pop indicates the population.
Output: pareto indicates the non-dominated solutions.
  pareto ← pop;
  for ∀ chr1 ∈ pop;
    for ∀ chr2 ∈ pareto ∧ chr2 ≠ chr1;
      if chr1 dominate chr2
        pareto ← pareto − chr2;
      end if
    end for
  end for
  return pareto;
```

Algorithm 2: Pseudocode of selecting elitist.

The work adopts the second strategy, namely, storing elitist solutions to an external secondary list. Its advantage is that it can preserve and dynamically adjust all the nondominated solutions set till the current generation. The pseudocodes of selecting elitist and updating elitist are, respectively, shown in Algorithms 2 and 3.

3.2. Selection Mechanism for the Swarm Based on Uniform Design. This paper adopts the uniform design to select the best T_1 points ($T_1 < T$) from the T points and acquires their objectives fit$V$. The detailed steps are as follows.

Algorithm B

Step 1. Calculate each of M objectives for each of the T points; normalize each of objectives $f_i(x)$ as follows:

$$h_i x = \frac{f_i(x)}{\max_{y \in \psi} \{|f_i(y)|\}}, \tag{5}$$

where ψ is a set of points in the current population and $h_i x$ is the normalized objective.

Step 2. Apply the uniform design to generate the D_0 weight vectors $w_1, w_2, \ldots, w_{D_0}$; each of them is used to compose one fitness function by the following formula, where D_0 is a design parameter and it is prime:

$$\text{fit}_i = w_{i,1} h_1(x) + w_{i,2} h_2(x) + \cdots + w_{i,D_0} h_{D_0}(x), \quad 1 \le i \le D_0. \tag{6}$$

```
population paretoupdate(offspring, pareto)
Input: offspring indicates the offsprings after performing the crossover operator
       and mutation operator; pareto indicates the non-dominated solutions.
Output: pareto indicates the non-dominated solutions.
  offspring ← call paretocreate(offspring);
  for ∀ chr1 ∈ offspring;
    nondominated ← true;
    for ∀ chr2 ∈ pareto;
      if chr1 dominate chr2
        pareto ← pareto − chr2;
      else if chr2 dominate chr1
        nondominated ← false;
      else
        continue
      end if
    end for
    if nondominated
      pareto ← pareto ∪ chr1
    end if
  end for
  return pareto;
```

Algorithm 3: Pseudocode of updating elitist.

Step 3. Based on each of fitness functions, evaluate the quality of the T points. Assume the remainder T_1/D_0 is R_0. For the first R_0 fitness functions, select the best $\lceil T_1/D_0 \rceil$ points; for the rest of fitness functions, select the best $\lfloor T_1/D_0 \rfloor$ points. Overall, a total of T_1 points are selected. The objectives of these selected points are correspondingly stored into fit$V$.

3.3. Selection Mechanism for Gbest Based on PAM. In MOPSO, *Gbest* plays an important role in guiding the entire swarm toward the global Pareto front [24]. In contrast to the single objective PSO having only one global best *Gbest*, MOPSO has multiple Pareto optimality solutions which are nondominated each other. How to select a suitable *Gbest* for each of particles from Pareto optimal solutions is one very key issue.

The paper presents the selection mechanism for *Gbest* based on PAM as follows.

Algorithm C. Assume the population and the numbers of particles are denoted as pop and Npop, and the Pareto optimality and the numbers of Pareto optimality are denoted as pareto and NP.

Step 1. Acquire the number of cluster according to the following formula:

$$K = K_{\min} + (K_{\max} - K_{\min}) \cdot \left(\frac{t}{\max G} \right), \tag{7}$$

where $K_{\min}$ and $K_{\max}$, respectively, denote the minimal and maximal value of K, namely, $K \in [K_{\min}, K_{\max}]$; t and $\max G$ indicate the tth iteration and the maximal iteration number. The formula can acquire the linearly increasing number of cluster so as to accord with the process from the coarse search to the elaborate search.

Step 2. If $NP \leq K$, then for each particle in pop, find the nearest Pareto optimal solution as its *Gbest*. Otherwise, turn to Step 3.

Step 3. Perform PAM described in Section 2.3 to partition Pareto and pop into K clusters, respectively, the cluster centroids of which are denoted as $C_1 = \{c_{1,1}, c_{1,2}, \ldots, c_{1,K}\}$ and $C_2 = \{c_{2,1}, c_{2,2}, \ldots, c_{2,K}\}$.

Step 4. For each $c_{2,j} \in C_2$, find the nearest $c_{1,i} \in C_1$. For all the particles in the cluster represented by $c_{2,j}$, their *Gbest* randomly takes one of Pareto optimal solutions in the cluster represented by $c_{1,i}$.

3.4. Adjustment of Pareto Optimal Solutions Based on the Uniform Design and PAM. The number of Pareto optimal solutions in the external archive will increasingly enlarge with evolution; namely, the size of the external archive will become very large with the iteration number. This will increase the computation complexity and the execution time of an algorithm. Therefore, the size of the external archive must be controlled. In the meanwhile, Pareto optimal solutions in the external archive are possibly close to each other in the objective space. If the solutions are close to each other, they are nearly the same choice. It is desirable to find the Pareto optimal solutions scattered uniformly over the Pareto front, so that the decision maker can have a variety of choices. Therefore, how to control the size of the external archive and select representative Pareto optimal solutions scattered uniformly over the Pareto front is a key issue.

The paper presents the adjustment algorithm of Pareto optimal solutions based on the uniform design and PAM. The algorithm firstly implements PAM to partition the Pareto front into K clusters in the objective space and then

implements the uniform crossover operator on the minimal cluster so as to generate more Pareto optimality in the minimal cluster; finally it keeps all the points in the lesser clusters and discards some points in the larger clusters. The detailed steps of the algorithm are shown as follows.

Algorithm D. Assume the Pareto optimality, their values of M functions, and the numbers of Pareto optimality are denoted as pareto, Fpareto, and NP. The size of the external archive is assumed as NP lim. The number of clusters is K.

Step 1. If the numbers of data points in the maximal and the minimal clusters have too many differences or are both very small, select all the data points from the minimal cluster and the same number of points from the cluster which is the nearest to the minimal cluster. For each pair of data points, perform the uniform crossover operator described in Section 2.4.3. Update pareto, Fpareto, and NP according to Algorithm 3.

Step 2. If $NP > NP$ lim, implement PAM to partition the Fpareto into K clusters in the objective space and let N div = NP lim $/K$; it indicates the average points to select from each of K clusters. We can classify three situations according to the number of points in each cluster and N div as follows.

(i) Situation 1: the numbers of points in all clusters are larger than or equal to N div.

(ii) Situation 2: the number of data points is larger than or equal to N div only in one cluster.

(iii) Situation 3: the number of the clusters, in which the number of data points is larger than N div, is larger than 1 and less than K.

Step 3. For Situation 1, sort all clusters according to their containing points in ascending order. Select N div points from the previous $K - 1$ clusters, and select the remainder points, NP lim $-N$ div $*(K - 1)$, from the last cluster. Turn to Step 7.

Step 4. For Situation 2, keep the points of all the clusters, in which the number of the points is less than or equal to N div, and the remainder points are selected from that only cluster. Turn to Step 7.

Step 5. For Situation 3, keep all the points of all the clusters, in which the number of points is less than or equal to N div. Turn to Step 6 to select the remainder point.

Step 6. Recalculate N div, and let N div = remP/K', where rem$P$ and K' indicate the number of the remainder points and clusters. According to the new N div, turn to Step 3 or Step 4.

Step 7. Save the selected NP lim points and terminate the algorithm.

3.5. Thoughts on the Proposed Algorithm. In MOP, for M objectives $f_1, f_2, \ldots, f_M$ and any two points x and y in the feasible solution space Ω, if each objective is satisfied with $f_i(x) \leq f_i(y)$ and at least one objective is satisfied with $f_i(x) < f_i(y)$, namely, x is at least as good as y with respect to all the objectives, and x is strictly better than y with respect to at least one objective, then we say that x dominates y. If no other solution is strictly better than x, then x is called a nondominated solution or Pareto optimal solution.

In single objective problems, there is only one objective to be optimized. Therefore, the global best (*Gbest*) of the whole swarm is only one. In multiobjective problems, there are multiple consistent or conflicting objectives to be optimized. Therefore, there exist very large or infinite numbers of solutions which cannot dominate each other. These solutions are nondominated solutions or Pareto optimal solutions. Therefore, *Gbest* of the whole swarm is more than one. How to select suitable *Gbest* is a very key issue.

When it is not possible to find all these solutions, it may be desirable to find as many solutions as possible in order to provide more choices to the decision maker. However, if the solutions are close to each other, they are nearly of the same choice. It is desirable to find the Pareto optimal solutions scattered uniformly over the Pareto front, so that the decision maker can have a variety of choices. The paper introduces uniform design to ensure that the acquired solutions scatter uniformly over the Pareto front in the objective space.

For the MOPSO, the diversity of the population is a very important factor. It has a key impact on the convergence of an algorithm and uniformly distribution of Pareto optimal solution. It can effectively get rid of the premature of the algorithms. The paper introduces PAM clustering algorithms to maintain the diversity of the population. The particles in the same cluster have similar features, whereas ones in different clusters have dissimilar features. Thus, choosing particles from different clusters may necessarily increase the diversity of the population.

3.6. Steps of the Proposed Algorithm

Step 1. According to the population size Npop, determine the number of subintervals S and the population size Q_0 in each subinterval, such that $Q_0 * S$ is more than Npop; Q_0 is a prime and must exist in Table 1. Execute Algorithm A in Section 2.4.2 to generate a temporary population containing $Q_0 * S \geq N$pop points.

Step 2. Perform Algorithm B described in Section 3.2 to select the best Npop points from the $Q_0 * S$ points as the initial population pop and acquire their objectives fit$V$.

Step 3. Initialize the speed and *Pbest* of the particle i by $V[i]$ = pop$[i]$ and *Pbest*$[i]$ = pop$[i]$; $fPbest[i]$ = fit$V[i]$ where $i = 1 \cdots N$pop; $fPbest[i]$ denotes objectives of *Pbest*$[i]$.

Step 4. According to Algorithm 2, select elitist or Pareto optimal solutions from pop and store them to the external secondary list, pareto, the size of which is assumed as NP.

Step 5. Choose the suitable *Gbest* for each of the particles from pareto in terms of Algorithm C described in Section 3.3.

Step 6. Update the position pop and velocity V for each of the particles, respectively, using formulas (2) and (3), where formula (2) is modified as follows:

$$v_{i,j}(k+1) = w \cdot v_{i,j}(k) + c_1 \cdot r_1 \cdot \left(Pbest_{i,j}(k) - x_{i,j}(k)\right) + c_2 \times r_2 \times \left(Gbest_{i,j}(k) - x_{i,j}(k)\right). \quad (8)$$

Step 7. For the jth dimensional variable of the ith particle, if it goes beyond its lower boundary or upper boundary, then it takes the jth dimensional value of its corresponding boundary and the jth dimensional value of its velocity takes the opposite value.

Step 8. Calculate each of M objectives for each of the particles, and update them into fit$V$.

Step 9. Update $Pbest$ of each particle as follows.

For the ith particle, if fit$V[i]$ dominates $fPbest[i]$, then let $fPbest[i]$ = fit$V[i]$ and $Pbest[i]$ = pop$[i]$; otherwise, $Pbest[i]$ and $fPbest[i]$ are kept unchanged. If neither of them is dominated by the other, then randomly select one of them as $fPbest[i]$ and update its corresponding $Pbest[i]$.

Step 10. Update the external archive pareto storing Pareto optimality and its size NP according to Algorithm 3.

Step 11. Implement Algorithm D described in Section 3.4 to adjust the Pareto optimal solutions such that the number of Pareto optimal solutions is less than or equal to the size of the external archive NP lim.

Step 12. If the stop criterion is satisfied, terminate algorithm; otherwise, turn to Step 5 and continue.

4. Numerical Results

In order to evaluate the performances of the proposed algorithm, we compare it with two outstanding algorithms, UMOGA [23] and NSGA-II [25].

4.1. Test Problems. Several well-known multiobjective test functions are used to test the performance of the proposed algorithm.

Biobjective problem: FON [25, 26], KUR [25, 27], ZDT1, ZDT2, ZDT3, and ZDT6 [25, 28–31].

Three-objective problems: DTLZ1 and DTLZ [2, 32].

The definitions of them are as follows.

FON is defined as

$$f_1(x) = 1 - \exp\left(-\sum_{i=1}^{3}\left(x_i - \frac{1}{\sqrt{3}}\right)^2\right),$$
$$f_2(x) = 1 - \exp\left(-\sum_{i=1}^{3}\left(x_i + \frac{1}{\sqrt{3}}\right)^2\right), \quad (9)$$

where $x_i \in [-4, 4]$.

This test function has a nonconvex Pareto optimal front.

KUR is defined as

$$f_1(x) = \sum_{i=1}^{n-1}\left(-10\exp\left(-0.2 \cdot \sqrt{x_i^2 + x_{i+1}^2}\right)\right),$$
$$f_2(x) = \sum_{i=1}^{n}\left(|x_i|^{0.8} + 5 \cdot \sin x_i^3\right), \quad (10)$$

where $n = 3$ and $x_i \in [-5, 5]$.

This test function has a nonconvex Pareto optimal front, in which there are three discontinuous regions.

ZDT1 is defined as

$$f_1(x_1) = x_1,$$
$$g(x_2, \ldots, x_n) = 1 + \frac{9\sum_{i=2}^{n} x_i}{(n-1)}, \quad (11)$$
$$h(f_1, g) = 1 - \sqrt{\frac{f_1}{g}},$$

where $n = 30$ and $x_i \in [0, 1]$.

This test function has a convex Pareto optimal front.

ZDT2 is defined as

$$f_1(x_1) = x_1,$$
$$g(x_2, \ldots, x_n) = 1 + \frac{9\sum_{i=2}^{n} x_i}{(n-1)}, \quad (12)$$
$$h(f_1, g) = 1 - \left(\frac{f_1}{g}\right)^2,$$

where $n = 30$ and $x_i \in [0, 1]$.

This test function has a nonconvex Pareto optimal front.

ZDT3 is defined as

$$f_1(x_1) = x_1,$$
$$g(x_2, \ldots, x_n) = 1 + \frac{9\sum_{i=2}^{n} x_i}{(n-1)}, \quad (13)$$
$$h(f_1, g) = 1 - \sqrt{\frac{f_1}{g}} - \left(\frac{f_1}{g}\right)\sin(10\pi f_1),$$

where $n = 30$ and $x_i \in [0, 1]$.

This test function represents the discreteness feature. Its Pareto optimal front consists of several noncontiguous convex parts. The introduction of the sine function in h causes discontinuity in the Pareto optimal front.

ZDT6 is defined as

$$f_1(x_1) = 1 - \exp(-4x_1)\sin^6(6\pi x_1),$$
$$g(x_2, \ldots, x_n) = 1 + 9\left(\frac{\sum_{i=2}^{n} x_i}{(n-1)}\right)^{0.25}, \quad (14)$$
$$h(f_1, g) = 1 - \left(\frac{f_1}{g}\right)^2,$$

where $n = 10$ and $x_i \in [0, 1]$.

This test function has a nonconvex Pareto optimal front. It includes two difficulties caused by the nonuniformity of the search space. Firstly, Pareto optimal solutions are nonuniformly distributed along the global Pareto optimal front (the front is biased for solutions in which $f_1(x_1)$ is near one). Secondly, the density of the solutions is the lowest near the Pareto optimal front and the highest away from the front.

DTLZ1 is defined as

$$f_1(x) = 0.5x_1x_2(1 + g(x)),$$
$$f_2(x) = 0.5x_1(1 - x_2)(1 + g(x)),$$
$$f_3(x) = 0.5(1 - x_1)(1 + g(x)),$$
$$g(x) = 100 \cdot \left[5 + \sum_{i=3}^{n}\left((x_i - 0.5)^2 - \cos(20\pi(x_i - 0.5))\right)\right], \tag{15}$$

where $n = 7$ and $x_i \in [0, 1]$.

This test function has difficulty in the convergence to the Pareto optimal hyperplane and contains $(11^5 - 1)$ local Pareto optimal fronts.

DTLZ2 is defined as

$$f_1(x) = (1 + g(x))\cos(0.5\pi x_1)\cos(0.5\pi x_2),$$
$$f_2(x) = (1 + g(x))\cos(0.5\pi x_1)\sin(0.5\pi x_2),$$
$$f_3(x) = (1 + g(x))\sin(0.5\pi x_1), \tag{16}$$
$$g(x) = \sum_{i=3}^{n}(x_i - 0.5)^2,$$

where $n = 12$ and $x_i \in [0, 1]$.

The Pareto optimal solutions of this test function must lie inside the first octant of the unit sphere in a three-objective plot. It is more difficult than DTLZ1.

4.2. Parameter Values. The parameter values of the proposed algorithm, UKMOPSO, are adopted as follows.

(i) *Parameters for PSO*: the linearly descending inertia weight [33, 34] is within the interval [0.1, 1], and the acceleration coefficients c_1 and c_2 are both taken as 2.

(ii) *Parameter for PAM*: the minimal and maximal numbers of clusters are $K_{\min} = 2$ and $K_{\max} = 10$.

(iii) *Population size*: the population size Npop is 200.

(iv) *Parameters for the uniform design*: the number of subintervals S is 64; the number of the sample points or the population size of each subinterval Q_0 is 31; set $D_0 = 31$ and $Q_1 = 5$.

(v) *Stopping condition*: the algorithm terminates if the number of iterations is larger than the given maximal generations of 20.

All the parameter values in UMOGA [23] are set equal to the original values in [23]; the different and additional parameter values between UMOGA and UKMOPSO are as follows: the number of subintervals S is 16; $F = N$ (the number of variables); $D_1 = 7$; $p_m = 0.02$.

NSGA-II in [25] adopted a population size of 100, a crossover probability of 0.8, a mutation probability of $1/n$ (where n is the number of variables), and maximal generations of 250. In order to make the comparisons fair, we have used population size of 100 and maximal generations of 40 in NSGA-II, so that the total number of function evaluations in NSGA-II, UKMOPSO, and UMOGA is the same.

4.3. Performance Metric. Based on the assumption that the true Pareto front of a test problem is known, many kinds of performance metrics have been proposed and used by many researchers such as [2, 9, 24, 25, 29, 35–40]. Three of them are adopted in the paper to compare the performance of the proposed algorithm with them. The first metric is C metric [29, 38]. It is taken as the quantitative metric of the solution quality and often used to show that the outcomes of one algorithm dominating the outcomes of another algorithm. It is defined as

$$C(A, B) = \frac{|b \in B : \exists a \in A : a \prec b|}{|B|}, \tag{17}$$

where $a \prec b$ represents b being dominated by a; A represents the Pareto front obtained by Algorithm A, and B represents the Pareto front obtained by Algorithm B in a typical run. Furthermore, $|B|$ represents the number of elements of B.

The metric value $C(A, B) = 1$ means that all points in B are dominated by or equal to points in A. In contrast, $C(A, B) = 0$ means that none of the points in B are covered by the ones in A.

The second metric is the IGD metric (namely, Inverted Generational Distance) [39–41]. It is the mean value of the distances from the set of solutions uniformly distributed in the true Pareto front to the set of solutions obtained by an algorithm in objective space. Let P^* be a set of uniformly distributed points along the true PF (Pareto Front). Let A be an approximate set to the PF; the average distance from P^* to A is defined as

$$\mathrm{IGD}(A, P^*) = \frac{\sum_{v \in P^*} d(v, A)}{|P^*|}, \tag{18}$$

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, $\mathrm{IGD}(A, P^*)$ could measure both the diversity and convergence of A in a sense. To have a low value of $\mathrm{IGD}(A, P^*)$, the set A must be very close to the PF and cannot miss any part of the whole PF.

The smaller the IGD value for the set of obtained solutions is, the better the performance of the algorithm will be.

The third measure is the measure of maximum spread (MS) [2, 24, 35], which is proposed in [42, 43]. It can measure how well the true Pareto front ($\mathrm{PF}_{\mathrm{true}}$) is covered by the discovered Pareto front ($\mathrm{PF}_{\mathrm{known}}$) through hyperboxes

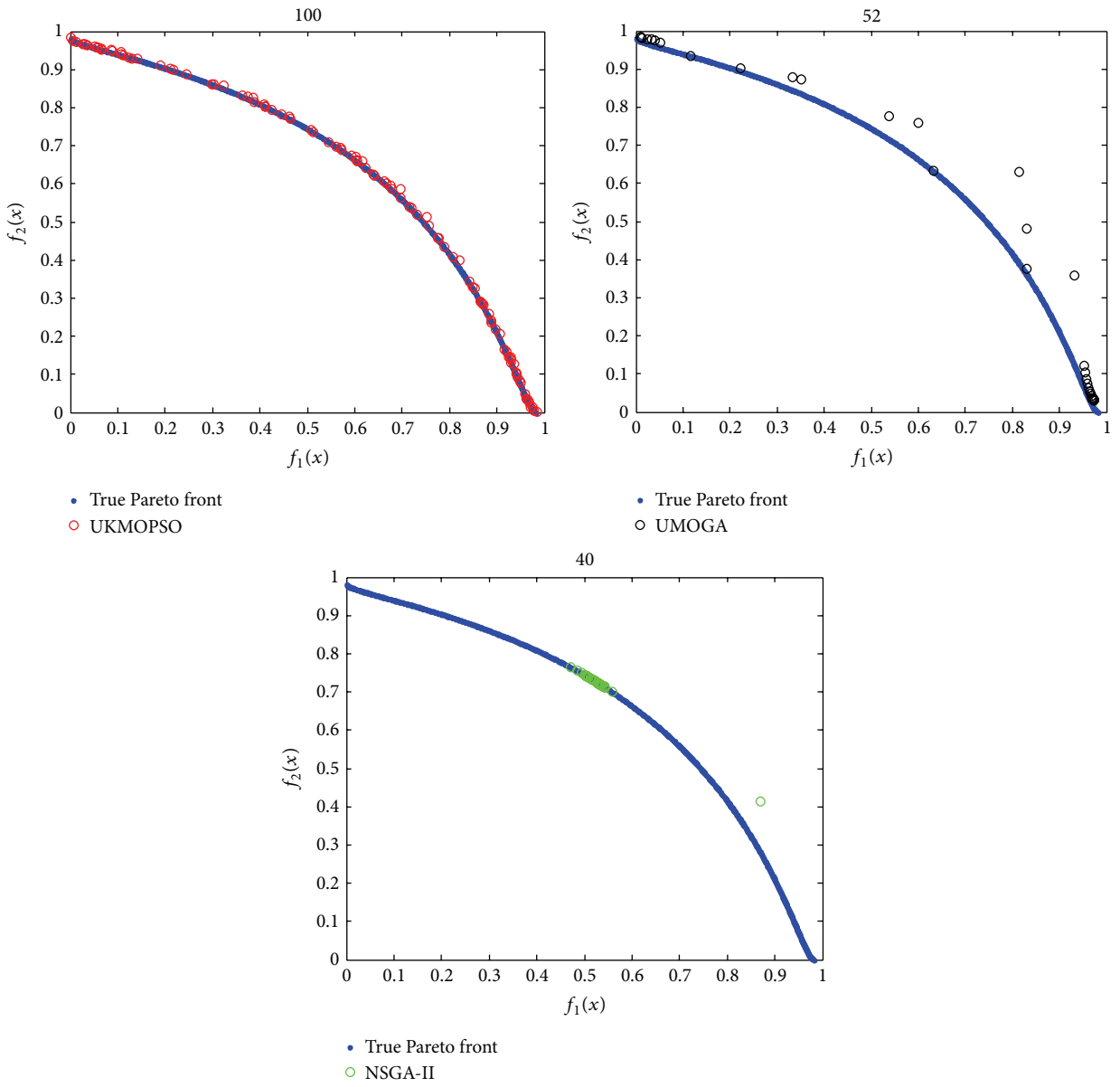

FIGURE 1: Pareto fronts obtained by different algorithms for test problem FON.

formed by the extreme function values observed in the PF_{true} and PF_{known}. It is defined as

$$MS = \sqrt{\frac{1}{M}\sum_{i=1}^{M}\delta_i},$$
$$\delta_i = \left(\frac{\min\left(f_i^{max}, F_i^{max}\right) - \max\left(f_i^{min}, F_i^{min}\right)}{F_i^{max} - F_i^{min}}\right)^2, \tag{19}$$

where M is the number of objectives, f_i^{max} and f_i^{min} are the maximum and minimum values of the ith objective in PF_{known}, respectively, and F_i^{max} and F_i^{min} are the maximum and minimum values of the ith objective in PF_{true}, respectively.

Note that if $f_i^{min} \geq F_i^{max}$, then $\delta_i = 0$. Algorithms with larger MS values are desirable and MS = 1 means that the true Pareto front is totally covered by the obtained Pareto front.

4.4. Results. For each test problem, we perform the proposed algorithm (called UKMOPSO) for 30 independent runs and compare its performance with UMOGA [23] and NSGA-II [25]. The values of several metrics, the C metric, IGD metric, and MS metric, are shown in Tables 2, 3, 4, and 5. The Pareto fronts obtained by several algorithms implemented on several test functions are illustrated in Figures 1–6.

For brevity, C(UKMOPSO, UMOGA), C(UMOGA, UKMOPSO), C(UKMOPSO, NSGA-II), and C(NSGA-II, UKMOPSO) are, respectively, marked as C_{12}, C_{21}, C_{13}, and C_{31}.

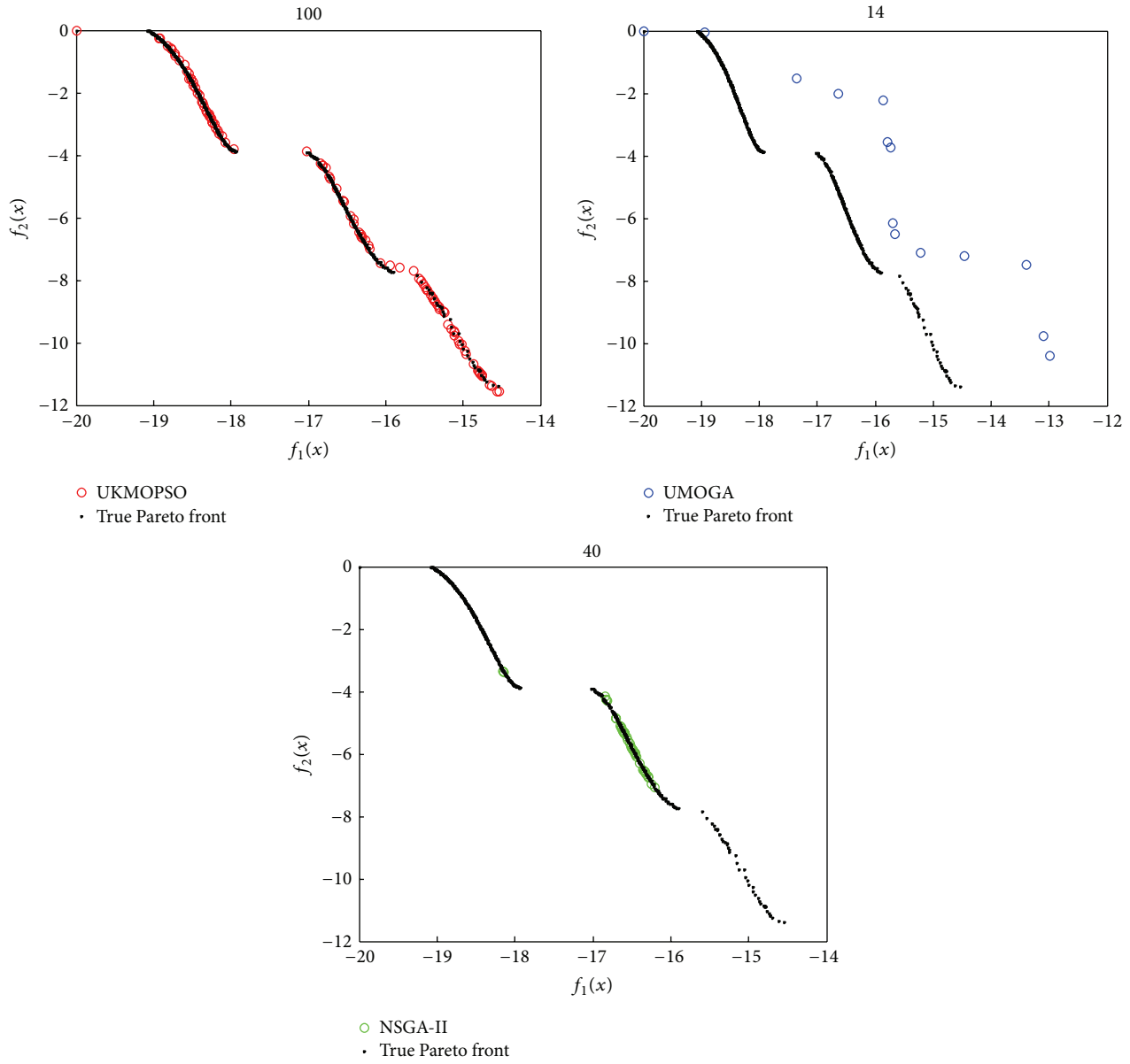

Figure 2: Pareto fronts obtained by different algorithms for test problem KUR.

Table 2: Comparison of C metric between UKMOPSO and UMOGA.

	C (UKMOPSO, UMOGA)	C (UMOGA, UKMOPSO)
FON	0.769	0.01
KUR	0.929	0.01
ZDT1	0.12	0.13
ZDT2	0.063	0.03
ZDT3	0.139	0.098
ZDT6	0.063	0.01
DTLZ1	1	0
DTLZ2	1	0

Table 3: Comparison of C metric between UKMOPSO and NSGA-II.

	C (UKMOPSO, NSGA-II)	C (NSGA-II, UKMOPSO)
FON	0.025	0.01
KUR	0.075	0.11
ZDT1	1	0
ZDT2	1	0
ZDT3	1	0
ZDT6	1	0
DTLZ1	0	0.13
DTLZ2	0.025	0.06

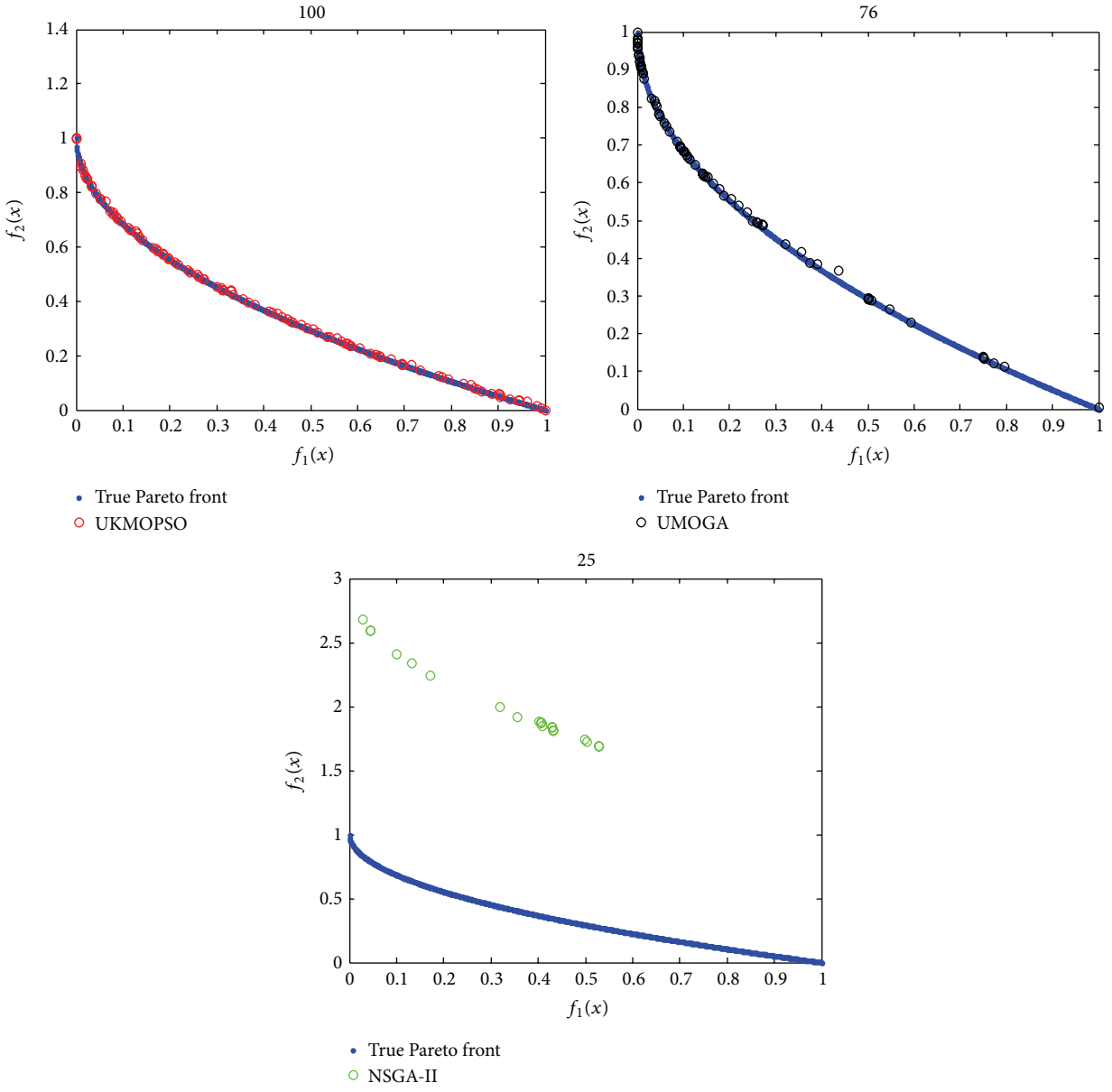

Figure 3: Pareto fronts obtained by different algorithms for test problem ZDT1.

Table 4: Comparison of IGD metric.

	UKMOPSO	UMOGA	NSGA-II
FON	0.0072	0.0432	0.3707
KUR	0.0668	1.1576	1.2323
ZDT1	0.0067	0.0265	1.3933
ZDT2	0.0067	0.0727	2.1409
ZDT3	0.1964	0.2151	0.2816
ZDT6	0.0033	0.0139	5.9223
DTLZ1	7.4827	22.965	5.4182
DTLZ2	0.1123	0.4319	0.2724

Table 5: Comparison of maximum spread (MS) metric.

	UKMOPSO	UMOGA	NSGA-II
FON	0.9769	0.9754	0.2368
KUR	0.9789	0.9580	0.3414
ZDT1	0.9429	0.9984	0.5495
ZDT2	0.9999	0.9986	0.0019
ZDT3	0.9276	0.9275	0.7963
ZDT6	1	0.9974	0.3738
DTLZ1	1	1	0.6588
DTLZ2	1	1	0.7125

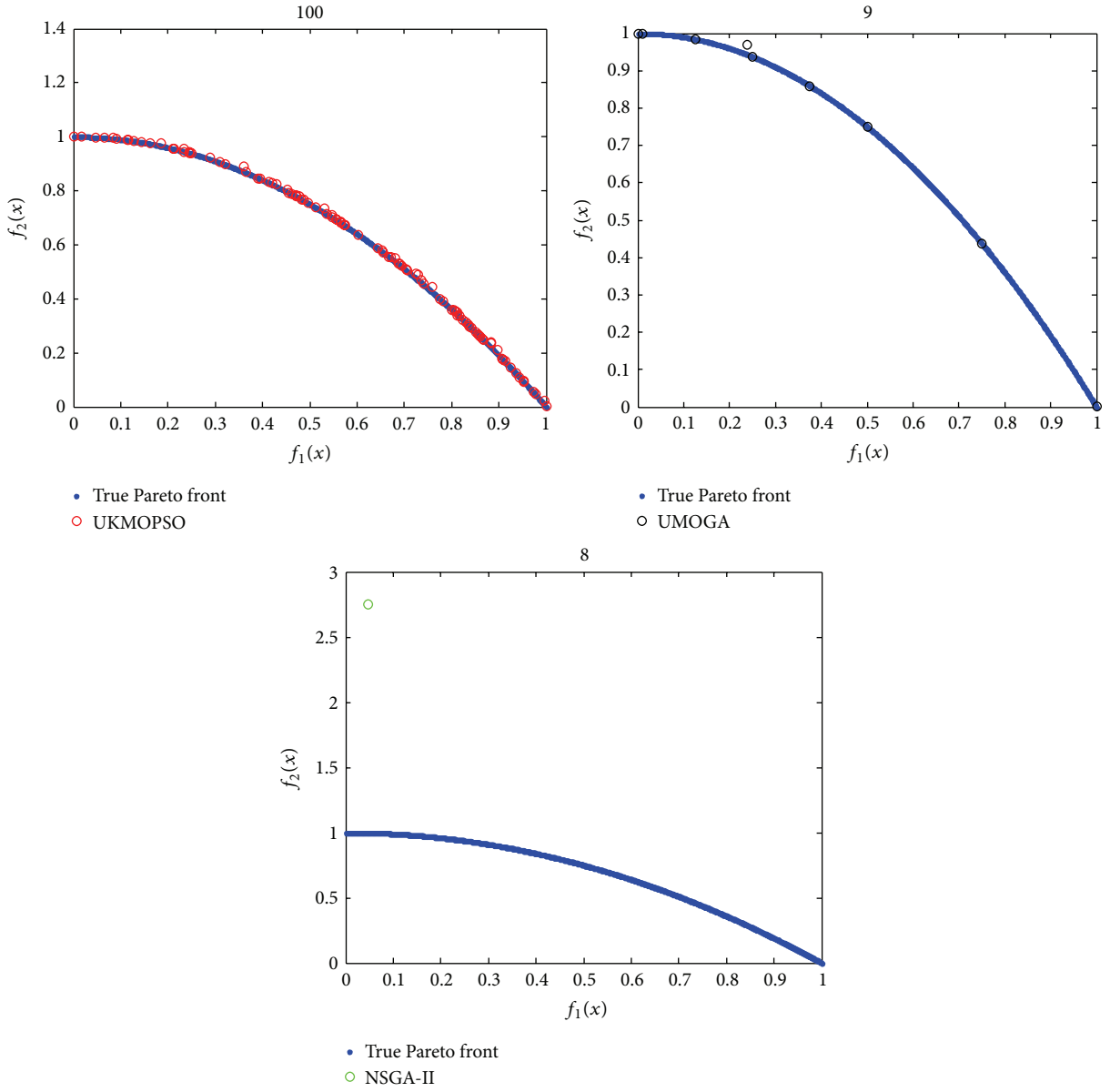

Figure 4: Pareto fronts obtained by different algorithms for test problem ZDT2.

As shown in Table 2, for the test functions, Fon and KUR, $C_{12} = 0.769$ and $C_{12} = 0.929$ mean that 76.8% and 92.9% of the solutions obtained by UMOGA are dominated by those obtained by UKMOPSO. Similarly, $C_{21} = 0.01$ means that 1% of the solutions obtained by UKMOPSO are dominated by those obtained by UMOGA. For DTLZ1 and DTLZ2, $C_{12} = 1$ and $C_{21} = 0$ mean that all solutions obtained by UMOGA are dominated by those obtained by UKMOPSO, but none of solutions from UKMOPSO is dominated by those from UMOGA. It means that the solution quality via UKMOPSO is much better than that via UMOGA for the above test functions. For ZDT1, ZDT2, ZDT3, and ZDT6, the values of the C metrics do not have too many differences between UKMOPSO and UMOGA. It means the solution qualities via both are almost identical.

From Table 3, we can see that for ZDT1, ZDT2, ZDT3, and ZDT6, $C_{13} = 1$ and $C_{31} = 0$ mean that all solutions obtained by NSGA-II are dominated by those obtained by UKMOPSO, but none of solutions from UKMOPSO is dominated by those from NSGA-II. It means that the solution quality via UKMOPSO is much better than that via NSGA-II for the above test functions. For the rest of the test functions, the solution qualities via both are almost identical.

In Table 4, for all test problems, the IGD values of UKMOPSO are the smallest among UKMOPSO, UMOGA, and NSGA-II. This means the PF found by UKMOPSO is the nearest to the true PF compared with the PF obtained by the other two algorithms; namely, the performance of UKMOPSO is the best in the three algorithms. The IGD values of UMOGA are all larger than those of NSGA-II for

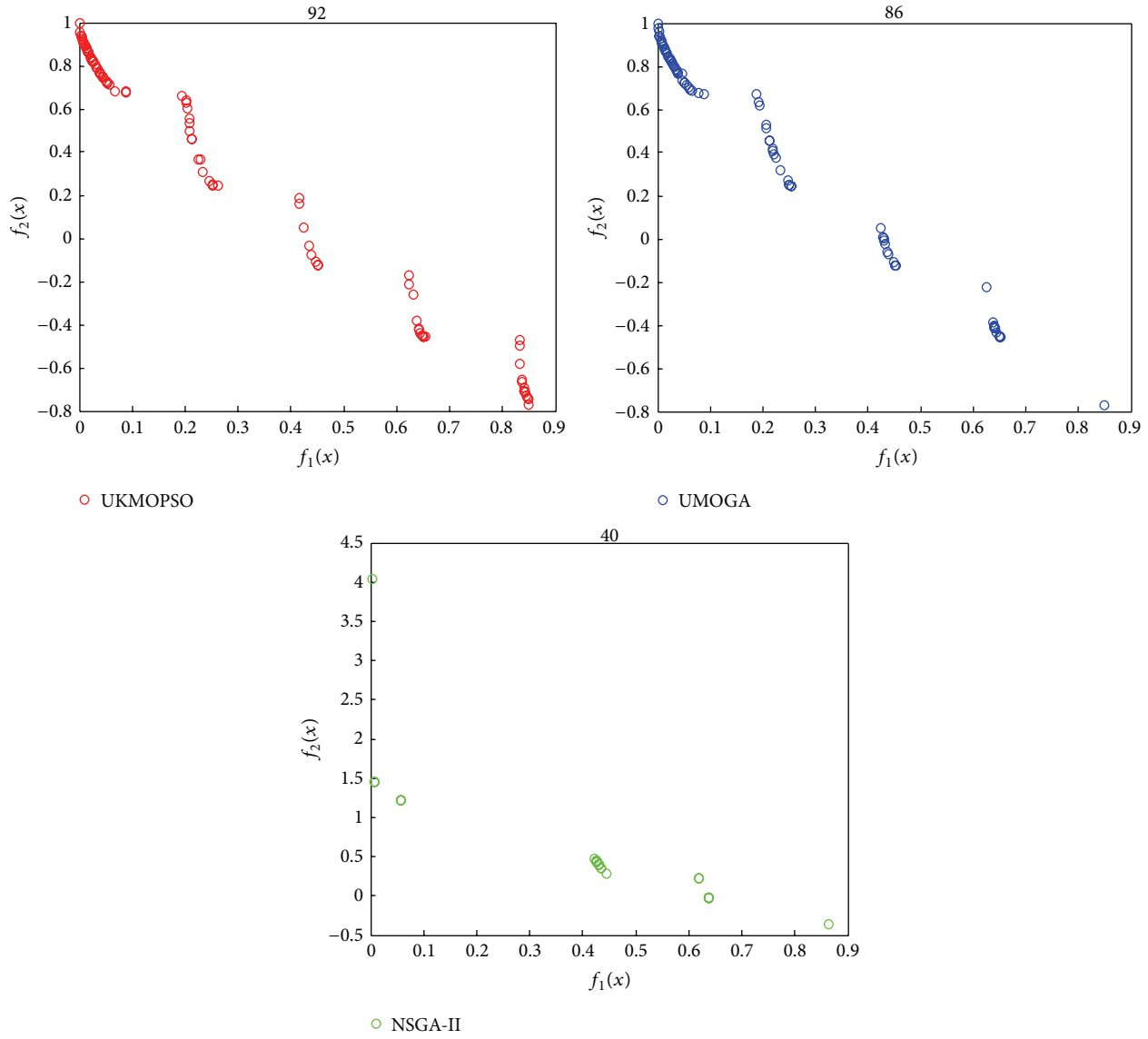

Figure 5: Pareto fronts obtained by different algorithms for test problem ZDT3.

all test functions. It means that the PF found by UMOGA is closer to the true PF than the PF obtained by NSGA-II.

From Table 5, we can see that all the MS values of UKMOPSO are almost close to 1 for each test function. It means that almost all the true PF is totally covered by the PF obtained by UKMOPSO. UMOGA is similar to UKMOPSO. The MS values of NSGA-II are much lesser than UKMOPSO and UMOGA, especially MS = 0.0019 for ZDT2.

Figures 1–6 all demonstrate that the PF found by UKMOPSO is the nearest to the true PF and scatters most uniformly in those obtained by three algorithms. Most of the points in the PF found by UKMOPSO overlap the points in the true PF. It means the solution quality obtained by UKMOPSO is very high.

4.5. Influence of the Uniform Crossover and PAM. In order to find out the influence of the uniform crossover on the proposed algorithm, we compare the distribution and number of Pareto optimal solutions before and after performing the uniform crossover. One of the simulating results on the test function ZDT1 is shown in Figure 7.

From Figure 7, it can be seen that, before and after performing the uniform crossover, the number of data points in the 1st cluster and the 2nd cluster varies from 7 to 15 and from 19 to 26, respectively; namely, many new Pareto optimal solutions having not been found before performing the uniform crossover are generated, and the differences of the data points between two clusters have been decreased. This will directly influence the uniformity of the PF and acquire more uniform Pareto solutions.

PAM is used to determine which Pareto solutions are to be removed from or be inserted into the external archive. This is to maintain the diversity of Pareto optimal solutions. The

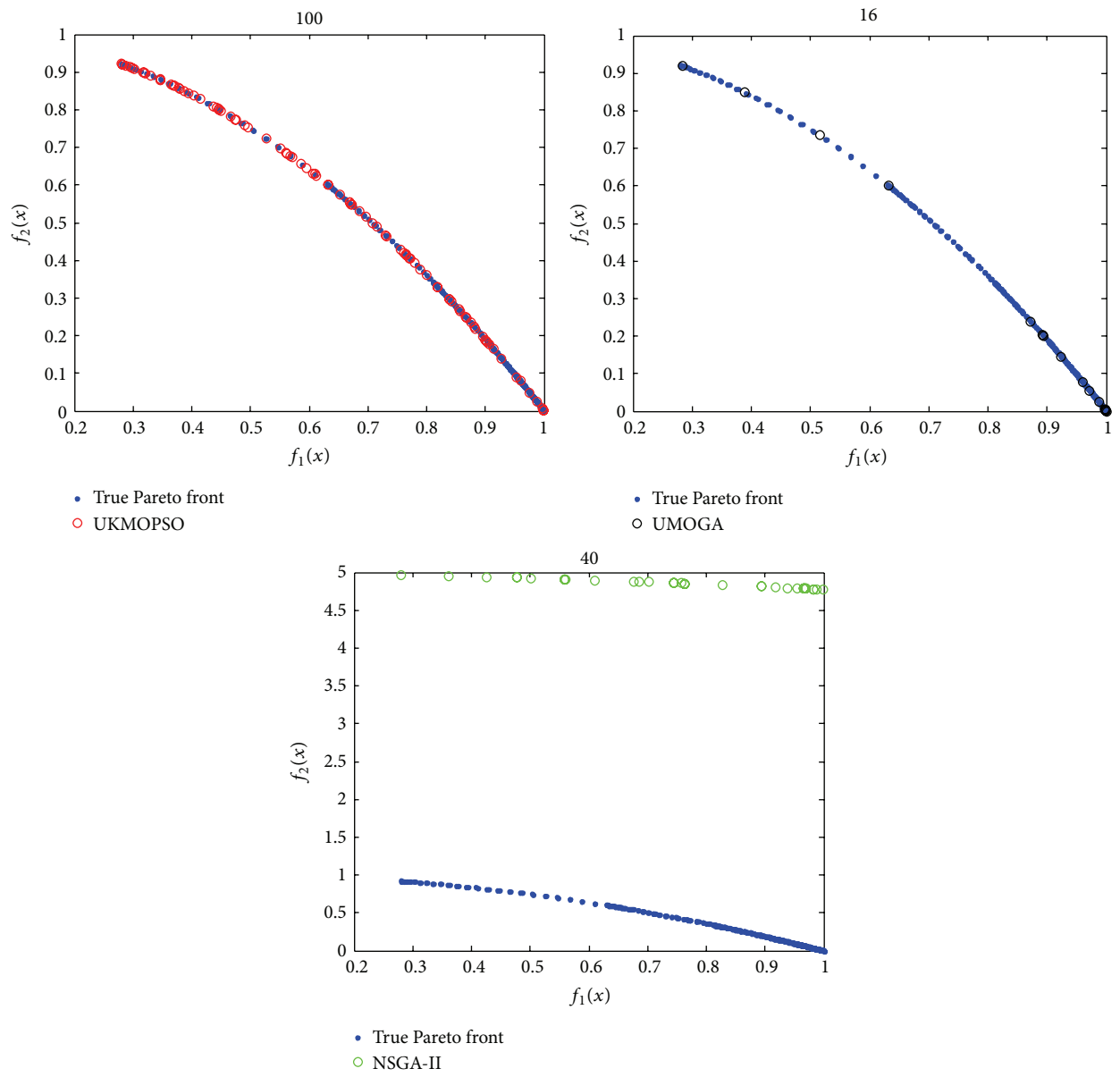

Figure 6: Pareto fronts obtained by different algorithms for test problem ZDT6.

diversity can be computed by the "distance-to-average-point" measure [44] defined as

$$\text{diversity}(S) = \frac{1}{|S|}\sum_{i=1}^{|S|}\sqrt{\sum_{j=1}^{N}\left(p_{ij} - \overline{p}_j\right)^2}, \tag{20}$$

where S is the population, $|S|$ is the swarm size, N is the dimensionality of the problem, p_{ij} is the j'th value of the i'th particle, and $\overline{p}_j$ is the j'th value of the average point $\overline{p}$.

If the number of Pareto optimality is less than or equal to the size of the external archive, PAM has no influence on the proposed algorithm. Otherwise, it is used to select the different type of Pareto optimality from several clusters. Therefore, The diversity of Pareto optimality will certainly increase. We monitor the diversity of Pareto optimality before and after performing PAM on ZDT1 at a certain time. The values are, respectively, 87.62 and 117.52. This fully demonstrates that PAM can improve the diversity of Pareto optimality.

5. Conclusion and Future Work

In this paper, a multiobjective particle swarm optimization based on PAM and uniform design is presented. It firstly implements PAM to partition the Pareto front into K clusters in the objective space; and then it implements the uniform crossover operator on the minimal cluster so as to generate more Pareto optimality in the minimal cluster. When the size of the Pareto solution is larger than that of the external archive, PAM is used to determine which Pareto solutions are to be removed from or be inserted into the external archive.

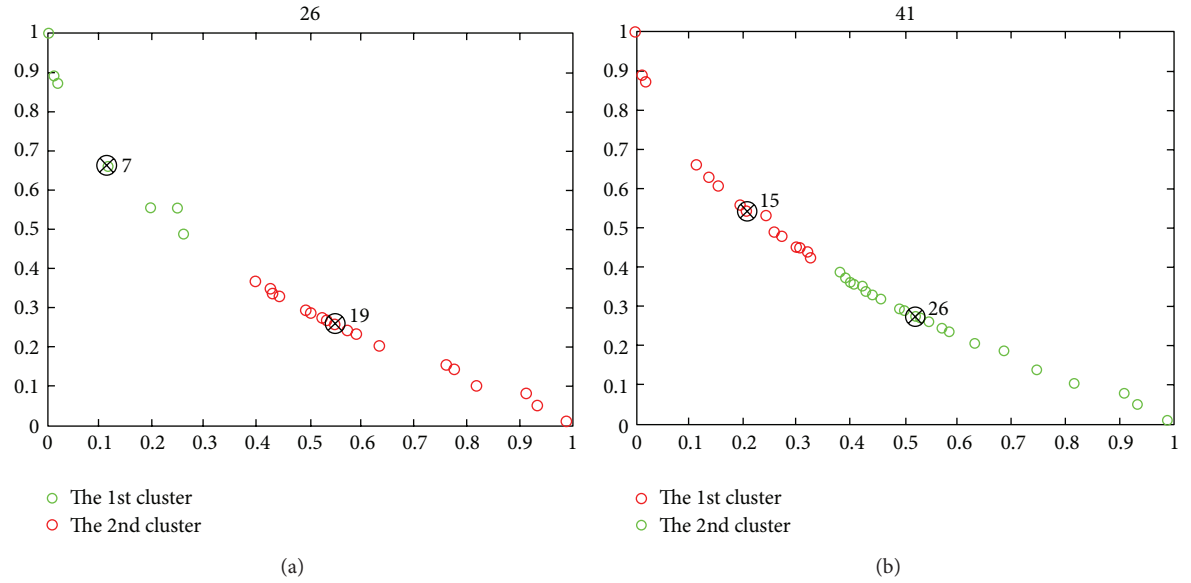

Figure 7: Pareto optimal solutions before (a) and after (b) performing the uniform crossover.

Finally, it keeps all the points in the lesser clusters and discards some points in the larger clusters. This can ensure that each of the clusters will contain approximately the same data points. Therefore, the diversity of the Pareto solutions will increase, and they can scatter uniformly over the Pareto front. The results of the experimental simulation performed on several well-known test problems indicate that the proposed algorithm obviously outperforms the other two algorithms.

This algorithm is going on for further enhancement and improvement. One attempt is to use a more efficient or approximate clustering algorithm to speed up the execution time of this algorithm. Another attempt is to extend its application scopes.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This research was supported by Guangxi Natural Science Foundation (no. 2013GXNSFAA019337), Guangxi Universities Key Project of Science and Technology Research (no. KY2015ZD099), Key project of Guangxi Education Department (no. 2013ZD055), Scientific Research Staring Foundation for the PHD Scholars of Yulin Normal University (no. G2014005), Special Project of Yulin Normal University (no. 2012YJZX04), and Key Project of Yulin Normal University (no. 2014YJZD05). The authors are grateful to the anonymous referee for a careful checking of the details and for helpful comments that improved this paper.

References

[1] J. D. Schaffer, "Multiple objective optimization with vector evaluated genetic algorithms," in *Proceedings of the 1st International Conference on Genetic Algorithms*, pp. 93–100, 1985.

[2] L. Tang and X. Wang, "A hybrid multiobjective evolutionary algorithm for multiobjective optimization problems," *IEEE Transactions on Evolutionary Computation*, vol. 17, no. 1, pp. 20–45, 2013.

[3] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan, "A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II," in *Parallel Problem Solving from Nature PPSN VI*, vol. 1917 of *Lecture Notes in Computer Science*, pp. 849–858, Springer, Berlin, Germany, 2000.

[4] J. Zhang, Y. Wang, and J. Feng, "Attribute index and uniform design based multiobjective association rule mining with evolutionary algorithm," *The Scientific World Journal*, vol. 2013, Article ID 259347, 16 pages, 2013.

[5] S. Jiang and Z. Cai, "Faster convergence and higher hypervolume for multi-objective evolutionary algorithms by orthogonal and uniform design," in *Advances in Computation and Intelligence*, vol. 6382 of *Lecture Notes in Computer Science*, pp. 312–328, Springer, Berlin, Germany, 2010.

[6] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, pp. 1942–1948, December 1995.

[7] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proceedings of the 6th International Symposium on Micro Machine and Human Science*, pp. 39–43, IEEE, Nagoya, Japan, October 1995.

[8] L. Yang, "A fast and elitist multi-objective particle swarm algorithm: NSPSO," in *Proceedings of the IEEE International Conference on Granular Computing (GRC '08)*, pp. 470–475, August 2008.

[9] C. A. C. 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.

[10] M. Clerc and J. Kennedy, "The particle swarm-explosion, stability, and convergence in a multidimensional complex space," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 1, pp. 58–73, 2002.

[11] H. Xiaohui and R. Eberhart, "Multiobjective optimization using dynamic neighborhood particle swarm optimization," in *Proceedings of the Congress on Evolutionary Computation (CEC '02)*, vol. 2, pp. 1677–1681, May 2002.

[12] S. Mostaghim and J. Teich, "Strategies for finding good local guides in multi-objective particle swarm optimization (MOPSO)," in *Proceedings of the IEEE Swarm Intelligence Symposium (SIS '03)*, pp. 26–33, Indianapolis, Ind, USA, April 2003.

[13] T. Bartz-Beielstein, P. Limbourg, J. Mehnen, K. Schmitt, K. E. Parsopoulos, and M. N. Vrahatis, "Particle swarm optimizers for Pareto optimization with enhanced archiving techniques," in *Proceedings of the Congress on Evolutionary Computation (CEC '03)*, vol. 3, pp. 1780–1787, December 2003.

[14] K. E. Parsopoulos, D. K. Tasoulis, and M. N. Vrahatis, "Multiobjective optimization using parallel vector evaluated particle swarm optimization," in *Proceedings of the IASTED International Conference on Artificial Intelligence and Applications (AIA '04)*, vol. 2, pp. 823–828, February 2004.

[15] S.-J. Tsai, T.-Y. Sun, C.-C. Liu, S.-T. Hsieh, W.-C. Wu, and S.-Y. Chiu, "An improved multi-objective particle swarm optimizer for multi-objective problems," *Expert Systems with Applications*, vol. 37, no. 8, pp. 5872–5886, 2010.

[16] A. A. Mousa, M. A. El-Shorbagy, and W. F. Abd-El-Wahed, "Local search based hybrid particle swarm optimization algorithm for multiobjective optimization," *Swarm and Evolutionary Computation*, vol. 3, pp. 1–14, 2012.

[17] J. Wang, W. Liu, W. Zhang, and B. Yang, "Multi-objective particle swarm optimization based on self-update and grid strategy," in *Proceedings of the 2012 International Conference on Information Technology and Software Engineering*, vol. 211 of *Lecture Notes in Electrical Engineering*, pp. 869–876, Springer, Berlin, Germany, 2013.

[18] K. Khalili-Damghani, A.-R. Abtahi, and M. Tavana, "A new multi-objective particle swarm optimization method for solving reliability redundancy allocation problems," *Reliability Engineering & System Safety*, vol. 111, pp. 58–75, 2013.

[19] J. Zhang, Y. Wang, and J. Feng, "Parallel particle swarm optimization based on PAM," in *Proceedings of the 2nd International Conference on Information Engineering and Computer Science (ICIECS '10)*, Wuhan, China, December 2010.

[20] J. Han and M. Kamber, *Data Mining: Concepts and Techniques*, Morgan Kaufmann, 2006.

[21] L. F. Ibrahim, "Using of clustering algorithm CWSP-PAM for rural network planning," in *Proceedings of the 3rd International Conference on Information Technology and Applications (ICITA '05)*, vol. 1, pp. 280–283, IEEE, July 2005.

[22] P. Chopra, J. Kang, and J. Lee, "Using gene pair combinations to improve the accuracy of the PAM classifier," in *Proceedings of the IEEE International Conference on Bioinformatics and Biomedicine (BIBM '09)*, pp. 174–177, November 2009.

[23] Y.-W. Leung and Y. Wang, "Multiobjective programming using uniform design and genetic algorithm," *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, vol. 30, no. 3, pp. 293–304, 2000.

[24] D. Liu, K. C. Tan, C. K. Goh, and W. K. Ho, "A multiobjective memetic algorithm based on particle swarm optimization," *IEEE Transactions on Systems, Man, and Cybernetics. Part B: Cybernetics*, vol. 37, no. 1, pp. 42–50, 2007.

[25] 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.

[26] C. M. Fonseca and P. J. Fleming, "Multiobjective optimization and multiple constraint handling with evolutionary algorithms. II. Application example," *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, vol. 28, no. 1, pp. 38–47, 1998.

[27] F. Kursawe, "A variant of evolution strategies for vector optimization," in *Parallel Problem Solving from Nature*, vol. 496 of *Lecture Notes in Computer Science*, pp. 193–197, Springer, Berlin, Germany, 1991.

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

[29] H.-L. Liu, Y. Wang, and Y.-M. Cheung, "A Multi-objective evolutionary algorithm using min-max strategy and sphere coordinate transformation," *Intelligent Automation & Soft Computing*, vol. 15, no. 3, pp. 361–384, 2009.

[30] N. Chakraborti, B. S. Kumar, V. S. Babu, S. Moitra, and A. Mukhopadhyay, "A new multi-objective genetic algorithm applied to hot-rolling process," *Applied Mathematical Modelling*, vol. 32, no. 9, pp. 1781–1789, 2008.

[31] K. Deb, "Multi-objective genetic algorithms: problem difficulties and construction of test problems," *Evolutionary Computation*, vol. 7, no. 3, pp. 205–230, 1999.

[32] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, "Scalable test problems for evolutionary multiobjective optimization," in *Evolutionary Multiobjective Optimization*, Advanced Information and Knowledge Processing, pp. 105–145, Springer, London, UK, 2005.

[33] Y. Shi and R. Eberhart, "Modified particle swarm optimizer," in *Proceedings of the IEEE World Congress on Computational Intelligence*, pp. 69–73, May 1998.

[34] C. S. Feng, S. Cong, and X. Y. Feng, "A new adaptive inertia weight strategy in particle swarm optimization," in *Procedings of the IEEE Congress on Evolutionary Computation (CEC '07)*, pp. 4186–4190, IEEE, Singapore, September 2007.

[35] C. K. Goh, K. C. Tan, D. S. Liu, and S. C. Chiam, "A competitive and cooperative co-evolutionary approach to multi-objective particle swarm optimization algorithm design," *European Journal of Operational Research*, vol. 202, no. 1, pp. 42–54, 2010.

[36] P. Cheng, J.-S. Pan, L. Li, Y. Tang, and C. Huang, "A survey of performance assessment for multiobjective optimizers," in *Proceedings of the 4th International Conference on Genetic and Evolutionary Computing (ICGEC '10)*, pp. 341–345, December 2010.

[37] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. Da Fonseca, "Performance assessment of multiobjective optimizers: an analysis and review," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 2, pp. 117–132, 2003.

[38] 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.

[39] H. Li and Q. Zhang, "Multiobjective optimization problems with complicated pareto sets, MOEA/D and NSGA-II," *IEEE*

Transactions on Evolutionary Computation, vol. 13, no. 2, pp. 284–302, 2009.

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

[41] 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.

[42] C.-S. Tsou, H.-H. Fang, H.-H. Chang, and C.-H. Kao, "An improved particle swarm Pareto optimizer with local search and clustering," in *Simulated Evolution and Learning*, vol. 4247 of *Lecture Notes in Computer Science*, pp. 400–407, 2006.

[43] U. Wickramasinghe and X. Li, "Choosing leaders for multiobjective PSO algorithms using differential evolution," in *Proceeding of the 7th International Conference Simulated Evolution and Learning*, vol. 5361 of *Lecture Notes in Computer Science*, pp. 249–258, Springer, Berlin, Germany, 2008.

[44] T. Krink, J. S. Vesterstrom, and J. Riget, "Particle swarm optimisation with spatial particle extension," in *Proceedings of the Congress on Evolutionary Computation (CEC '02)*, pp. 1474–1479, May 2002.