A novel multi-objective particle swarm optimization with *K*-means based global best selection strategy

Chenye Qiu[*]
School of Computer, Beijing University of Posts and Telecommunications
No. 10 Xi Tu Cheng Road, Haidian District, Beijing, 100876, China
E-mail: qiuchenye@gmail.com

Chunlu Wang
School of Computer, Beijing University of Posts and Telecommunications
No. 10 Xi Tu Cheng Road, Haidian District, Beijing, 100876, China
E-mail: wangcl@bupt.edu.cn

Xingquan Zuo
School of Computer, Beijing University of Posts and Telecommunications
No. 10 Xi Tu Cheng Road, Haidian District, Beijing, 100876, China
E-mail: zuoxq@bupt.edu.cn

Received 28 February 2012
Accepted 10 January 2013

Abstract

In this paper, a multi-objective particle swarm optimization algorithm with a new global best (*gbest*) selection strategy is proposed for dealing with multi-objective problems. In multi-objective particle swarm optimization, *gbest* plays an important role in convergence and diversity of solutions. A *K*-means algorithm and proportional distribution based approach is used to select *gbest* from the archive for each particle of the population. A symmetric mutation operator is incorporated to enhance the exploratory capabilities. The proposed approach is validated using seven popular benchmark functions. The simulation results indicate that the proposed algorithm is highly competitive in terms of convergence and diversity in comparison with several state-of-the-art algorithms.

Keywords: Particle swarm optimization; Multi-objective optimization; *K*-means algorithm; Global best; Symmetric mutation operator.

1. Introduction

Many real-world optimization problems involve multiple objectives that should be optimized simultaneously. Sometimes, these objectives are even conflicting. Improve one objective would worsen at least one other objective. Contrary to single objective (SO) optimization problem, there is no single optimal solution in multi-objective (MO) optimization problem, but a set of trade-off solutions.

Evolutionary algorithms can deal with a set of possible solutions (so-called population) simultaneously which allows us to find a set of optimal solutions in a single run of the algorithm.[1] So it is very promising to apply evolutionary algorithms to MO problems. Since Schaffer first proposed a Vector Evaluated Genetic

[*]Corresponding Author.

Algorithm (VEGA),[2,3] many evolutionary algorithms have been proposed to solve MO problems, such as NSGA-II, SPEA2, Micro-GA, etc.[4-6] These clever designed methods have been proved to be very effective in dealing with MO optimization problems.

As numerous genetic algorithm based methods have been proposed, many researchers are paying more and more interest in using Particle Swarm Optimization[7] (PSO) to solve MO problems. PSO is a relatively new algorithm, which is inspired by the social interaction of bird flocking. A standard particle swarm optimization includes a swarm of particles. Each particle represents a candidate solution of the problem. Particles fly in a multi-dimensional search space looking for the optimal position according to its own flying experience and the experience of the best particle in the swarm. PSO has been proved to be very effective in a wide variety of optimization problems due to its fast convergence and ease of implementation.[8]

The original PSO was proposed to solve SO problems. Extending original PSO to multi-objective PSO requires a redefinition of the global best (*gbest*) in order to obtain a set of non-dominated solutions. The *gbest* in the PSO has a great impact on convergence and diversity of solutions. Contrary with the SO optimization, there is no single *gbest*, but a set of non-dominated solutions. Choosing the proper *gbest* from the set of non-dominated solutions for each particle in the prime swarm to direct its flight is very important and difficult. The difficulties lie in the following aspects:

(i) It's difficult to define criteria of choosing *gbest* since all the particles in the non-dominated set are pareto optimal. They do not dominate other particles nor dominated by others.[9]
(ii) The appropriate *gbest* should be able to cover the entire non-dominated front and encourage the exploration of the sparse regions.
(iii) The *gbest* selection method should be adapted to different kinds of problems.

Various *gbest* selection methods have been proposed in past years. Hu *et al.*[10] proposed a dynamic neighborhood PSO, which optimizes one objective in one cycle. The *gbest* is chosen according to the objective to be optimized in each cycle. The algorithm may be sensitive to the ordering of objectives. Coello *et al.*[11] introduced an algorithm called MOPSO. An external archive is used to store the non-dominated solutions and the adaptive grid is adopted to choose *gbest* from the archive. It shows good convergence performance and has lower computational cost but falls behind NSGA-II on diversity mechanism. On the basis of MOPSO, Raquel *et al.*[12] introduced the crowding distance operator[4] into MOPSO for the *gbest* selection instead of the adaptive grid. All the non-dominated particles in the archive are sorted based on a decreasing crowding distance. *Gbest* is randomly chosen from the top 10% particles. The new algorithm improves the diversity of solutions compared with MOPSO. Li[13] proposed NSPSO in which the main mechanisms of NSGA-II are adopted in a PSO algorithm. Two methods, niche counts and crowding distance, are used to select *gbest* in this algorithm. A strategy to find *gbest* in MOPSO, named *sigma method*, was proposed in Ref. 14. All the particles both in the population and the archive have a sigma value. For each particle in the population, it chooses its *gbest* from the archive according to the sigma value. A heuristic using a particle swarm optimizer and fitness sharing was proposed in Ref. 15, in which niche count is used for selecting *gbest* and archive pruning. Liu *et al.*[16] proposed a memetic algorithm for MO problems, which combines the global search ability of PSO with a synchronous local search heuristic. In order to provide diversity, they use fuzzy *gbest* instead of a crisp location which can encourage the particles to explore a region beyond that defined by the search trajectory. Shang-Jeng Tsai *et al.*[17] proposed an improved multi-objective particle swarm optimizer named PDJI-MOPSO. A disturbance operation is introduced into the *gbest* selection method which would affect particles to move towards unexpected directions. Yang *et al.*[9] introduced a *gbest* selection method which is a combination of the adaptive grid and the sigma method. The new *gbest* selection method can compromise global and local searching based on the process of evolution. Lei[18] proposed a multi-objective PSO for job shop scheduling problem, in which *gbest* is chosen according to crowding measure. Carvalho *et al.*[19] applied multi-objective PSO to software fault prediction problem. The sigma method is used for *gbest* selection in their approach to maintain diversity of solutions.

In the existing *gbest* selection methods, the adaptive grid[11], the crowding distance method[12] and the sigma method[14] are the most frequently used methods. These methods show promising results, however, there are still some disadvantages in these *gbest* selection methods. The adaptive grid has the advantage of low computational time but it fails in maintaining diversity of the generated solutions when compared with other MO algorithms. The crowding distance method only chooses *gbest* from the sparse regions. The chosen *gbest* cannot represent the entire non-dominated front which may result in bad convergence. The sigma method can guide the particles move to the Pareto front directly and show good convergence ability. However, this behavior may lead to premature and bad diversity when the initial particles of the archive are bad-distributed.[9]

For dealing with the aforementioned disadvantages, a multi-objective particle swarm optimization algorithm (KMOPSO) with a novel *gbest* selection strategy is presented in this paper. A *K*-means algorithm and proportional distribution based *gbest* selection strategy is used to select *gbest* from the external archive. This *gbest* selection strategy considers both global and local information of the non-dominated front. It can capture the whole non-dominated front while encouraging diversity. Particles in the population would move towards the entire non-dominated front uniformly. The simulation results and analysis prove that this strategy can achieve good convergence, diversity and spread.

Some researchers have incorporated *K*-means algorithm to improve the performance of PSO.[20, 21] However, they incorporated *K*-means algorithm in single-objective PSO. In their studies, *K*-means algorithm is used to divide the population into several clusters according to their values in the decision space. In our study, we use *K*-means algorithm in multi-objective PSO to select *gbest* from the archive according to their corresponding objective values.

Also, a symmetric mutation operator is presented to strengthen the exploratory capabilities of the proposed algorithm. PSO is known for its high speed of convergence, which would leads to the loss of population diversity. The behavior can also lead the swarm to be trapped in a local optimum from which they cannot escape. The symmetric mutation operator can help PSO to overcome its disadvantages by providing PSO the ability of jumping out of local optimum.

The remainder of this paper is organized as follows. Section 2 describes the basic concepts of MO problems. Section 3 gives some basic knowledge of PSO. In Section 4, we describe our approach in detail. Section 5 presents a comparative study with other well known MO evolutionary algorithms. Section 6 concludes the paper.

2. Basic Concept of MO Problems

In contrast to SO optimization, MO problems are characterized by a set of optimal solutions, known as *Pareto Optimal* solutions. Without any loss of generality, we consider a multi-objective minimization problem. It can be stated as:

$$\begin{aligned} &Min \quad \mathbf{y} = f(\mathbf{x}) = (f_1(\mathbf{x}),\ldots,f_n(\mathbf{x})) \\ &\mathbf{x} = (x_1,\ldots,x_m) \in \mathbf{X}, \mathbf{y} = (y_1,\ldots,y_n) \in \mathbf{Y} \end{aligned} \tag{1}$$

where $\mathbf{x}$ is the decision vector, $\mathbf{X}$ is the decision space, $\mathbf{y}$ is the objective vector and $\mathbf{Y}$ is the objective space. A vector $\mathbf{x}_k$ is said to dominate another vector $\mathbf{x}_l$, denoted as:

$$\begin{aligned} &\forall i \in 1,2,\ldots,n : f_i(\mathbf{x}_k) \le f_i(\mathbf{x}_l) \\ &\exists i \in 1,2,\ldots,n : f_i(\mathbf{x}_k) < f_i(\mathbf{x}_l) \end{aligned} \tag{2}$$

A decision vector $\mathbf{x}_k$ is called *Pareto optimal* if there does not exist another vector $\mathbf{x}_l \in \mathbf{X}$ that dominates it. The set of all non-dominated vectors in the decision variable set is the *Pareto optimal* set. The corresponding set of objective vectors is called *Pareto optimal* front.

3. Particle Swarm Optimization

PSO is a heuristic technique inspired by the social behavior of bird flocking. A standard particle swarm optimization includes a swarm of particles which represent solutions of the problem. Let $x_i = (x_{i1}, x_{i2}, \cdots, x_{iD})$ be the *i*th particle in the swarm. *D* is the dimension of the search space. Its current velocity is $v_i = (v_{i1}, v_{i2}, \cdots, v_{iD})$. In the basic PSO algorithm, the positions of particles are updated by the following equations:

$$v_{id}^{t+1} = \omega \times v_{id}^t + c_1 \times r_1^t \times (pbest_{id}^t - x_{id}^t) + c_2 \times r_2^t \times (gbest_d^t - x_{id}^t) \tag{3}$$

$$x_{id}^{t+1} = x_{id}^t + v_{id}^{t+1} \tag{4}$$

where v_{id}^t is the *d*th dimension of the velocity of particle *i* in cycle *t*; x_{id}^t the *d*th dimension of the position of particle *i* in cycle *t*; $pbest_{id}^t$ is the *d*th dimension of the position of personal best of particle *i* in cycle *t*; $gbest_d^t$ is the *d*th dimension of the position of *gbest* in cycle *t*; ω is the inertia weight. Inertia weight plays an important role in balancing global and local search. A large inertia weight promotes global search and a small inertia weight is more appropriate for local search. The value is typically set between 0 and 1. c_1 is the cognitive weight and c_2 is the social weight; r_1^t and r_2^t are two random numbers.

4. Proposed Approach

In the beginning of the algorithm, the positions of all the particles are randomly initialized. Their speed is set as 0.Their *pbest* are their present positions. All the non-dominated solutions are stored in an external archive. A *gbest* selection strategy based on *K*-means algorithm and proportional distribution is used to choose *gbest* from the archive in order to spread the particles along the Pareto front. The archive is updated after each cycle. When it reaches its limit, a nearest neighbor based pruning technique is used to control its size. Besides, a symmetric mutation operator is employed to improve the exploratory capability.

The steps of KMOPSO are as follows.

```
% M is the population size
% x is the position of particle
% v is the speed of each particle
% P is the population
% t is the iteration counter
% K is the number of clusters
(i)   Initialize the population:
        a. For $i$ =1 to $M$
        b. Initialize $x_i$ randomly
        c. Initialize $v_i = 0$
(ii)  Evaluate all the particles.
(iii) Store the non-dominated solutions in P into the
        external archive A.
(iv)  Initialize the personal best of each particle i:
        $pbest_i = x_i$
(v)   While t < maximum number of iterations
DO
        (a) Compute the speed of each particle with Eq.
              (3).
        (b) Compute the new position of each particle with
              Eq. (4).
        (c) If $x_i$ goes beyond its search boundaries, we take
              two measures: 1) set the decision variable the
              value of its corresponding lower of upper
              boundary; 2) its velocity is multiplied by -1 in
              order to make it searches the opposite
              direction.
        (d) Apply the mutation operator.
        (e) Evaluate all the particles in population.
        (f) Update the external archive A.
        (g) Update pbest for each particle.
        (h) Increment the loop counter: $t = t + 1$.
End while.
```

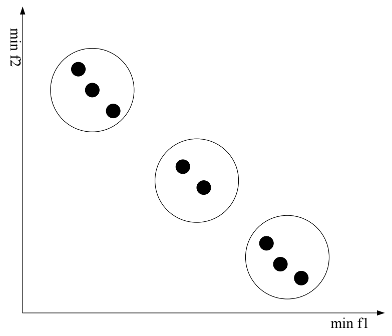


Fig. 1. Divide particles into *K* clusters.

4.1. *Gbest selection strategy based on K-means algorithm and proportional distribution*

In MOPSO, *gbest* is very important in guiding the entire population moving towards the true Pareto front. Different from single objective optimization problem, there exists a set of non-dominated solutions in the external archive. So this leads to the problem of how to choose *gbest* from the non-dominated solutions. This paper introduced a *gbest* selection strategy based on *K*-means algorithm and proportional distribution in order to lead to a diverse and uniformly distributed set of solutions.

As shown in Fig.1, the first step of the *gbest* selection method is to divide the particles in the archive into *K* clusters according to their corresponding objective function values.[22] It operates as follows:

(i) Randomly choose *K* solutions, each of which represents a cluster center.
(ii) For each of the remaining solutions, each solution is assigned to the cluster to which it is the most similar, based on the Euclidean distance between the solution and the cluster center.
(iii) Compute the new center of each cluster.
(iv) Iterate step 2 and 3 until convergence of objective function:

$$E = \sum_{i=1}^{K} \sum_{\mathbf{f} \in C_i} \left| \mathbf{f} - \mathbf{m}_i \right| \tag{5}$$

where E is the sum of the square error for all solutions in the archive; $\mathbf{f}$ is the point in space representing a given solution; $\mathbf{m}_i$ is the mean of the cluster C_i.

(v) In each cluster, find the solution nearest to the centroid, and make it the representative solution of the cluster.

After the clustering process, the clusters are separated from each other and the solutions in the same cluster are similar. The representative solutions lie in diverse regions of the non-dominated front. The next step is to choose *gbest* from the *K* representative particles. In the *K* clusters, they contain different numbers of particles. If a cluster has a large number of particles, it means this region is crowded. We should encourage particles move towards those less crowded regions. A proportional distribution method is used to select *gbest* from those *K* representative particles. The probability of a representative particle *i* being chosen as *gbest* is calculated as follows:

$$p_i = \frac{1 / num_i}{\sum_{i=1}^{K} 1 / num_i} \tag{6}$$

where p_i is the probability of the *i*th representative particle being chosen as *gbest*. num_i is the number of particles in the *i*th cluster. As shown in Eq. (6), the representative particle *i* has more opportunity to be chosen as the *gbest* if its corresponding cluster has less particles.

This *gbest* selection method considers both the global and local information of the non-dominated front. In the clustering process, we first consider the distribution of all the particles in the non-dominated front. The particles in a small crowded region would be merged into one cluster and the particles in a sparse region will be assigned to one cluster with only a few particles. Then in the proportional distribution, the local information is considered. The representative particle whose corresponding cluster has fewer particles has more opportunity to be chosen as the *gbest*. The *gbest* chosen by this algorithm can represent the distribution of all the non-dominated solutions and encourage the exploration of the sparse regions. The particles in the crowded regions also have the opportunity to be chosen as the *gbest*. Hence this algorithm would not decrease the convergence speed.

4.2. *External archive and the pruning method*

In our algorithm, elitism is implemented by using a fixed-size external archive to prevent the loss of good particles. External archive is used to store the non-dominated particles found during the evolution process. The archive is updated in each cycle. If the candidate solution is not dominated by any solution in the archive, it will enter the archive. Any archive members dominated by this solution will be removed from the archive.

The size of external archive is limited considering the computational cost. When the external archive is full, a pruning method based on the nearest neighbor is carried out to determine which solution in the archive is to be replaced. It is chosen according to the following steps:

(i) For $i = 1$: N (N is the archive size)

(ii) Calculate the Euclidean distance between the *i*th solution and the other N-1 solutions and store the minimum one.

(iii) End for.

(iv) Choose the solution with the minimum distance to another solution. If several solutions tie, then compare their distance to their second nearest neighbors and so forth.

By this pruning method, diversity can be promoted in the archive as the most crowded solution is replaced by a new non-dominated solution.

4.3. *The mutation operator*

PSO is known for its high convergence speed. In MO optimization, such convergence speed may be harmful because this may lead the algorithm converge to a local Pareto front. In some complex MO problems, there exists many local optimal. If *gbest* corresponds to one of the local optima, the swarm would converge to the vicinity of the local optimal rapidly. This motivates the use of a mutation operator. A symmetric mutation operator is adopted to enrich the exploratory abilities. The mutation operator is performed on the entire population.

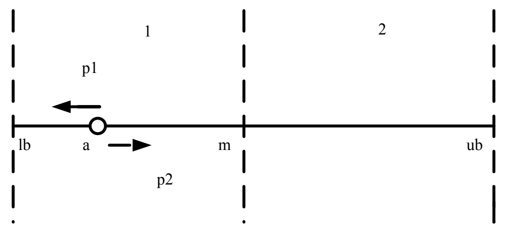


Fig. 2. The mutation operator

The mutation operator is described in Fig. 2. For each decision variable of particle, the search space is divided into two symmetric regions, region 1(r1) and region 2(r2). Particle will mutate in the region in which it locates. As shown in Fig. 2, particle *a* lies in r1and it will mutate inside r1. The new position of *a* will be updated as follows:

(i) Calculate its distance to the lower bound (*lb*) and the middle point (*mp*). Denote *distolb* as the distance to the lower bound. Denote *distomp* as the distance to the middle point.

(ii) Generate a random real number *rand* between 0 and 1. If $rand < \frac{distomp}{distolb + distomp}$, particle *a* will search in the direction of lower bound. The position of particle *a* will be:

$$a' = RandomDouble(lb, a) \tag{7}$$

$RandomDouble(lb, a)$ means generating a random number between *lb* and *a*.

If $rand > \frac{distomp}{distolb + distomp}$, particle *a* will search in the opposite direction. The position of particle *a* will be:

$$a' = RandomDouble(a, m) \tag{8}$$

If particle is nearer to the lower bounder, it has more opportunity to search in the direction of the lower bounder. If particle is nearer to the middle point, it has more opportunity to search in the direction of the middle point. For particles in r2, the mutation operator works in the same way.

4.4. *Redefinition of the pbest*

In MO problems, the traditional way to update *pbest* is as follows: if the current particle dominates its present *pbest*, the *pbest* will be updated. If neither of them dominates the other, one of them is selected as the new *pbest* randomly. Due to the aforementioned searching behavior in MO problems, the *pbest* will usually stay the same in many cycles[16]. In this situation, the *pbest* has no effect on guiding particles move to a new place. Particle would adjust their search strategies only with their *gbest*.

In this paper, the *pbest* is redefined to solve this problem. In section 4.1, *K* representative particles are selected by the *gbest* selection strategy. In each cycle, for each particle in the population, calculate the distance to all *K* representative particles. The nearest representative particle will be its new *pbest*.

5. Experiment and Analysis

5.1. *Performance metrics*

In order to provide a quantitative assessment for the performance of the proposed MO optimization algorithms, three measures are used.

Generational distance (GD): The metric of generational distance was introduced by Van Veldhuizen and Lamont[23] as an indication of the gap between the discovered Pareto front and the true Pareto front. It is defined as:

$$GD = \sqrt{\sum_{i=1}^{n} d_i^2} \Big/ n \tag{9}$$

where *n* is the number of non-dominated solutions found by the algorithm and d_i is the Euclidean distance between the solution *i* and its nearest solution in the true Pareto front. $GD = 0$ means that the generated solutions are in the Pareto optimal set.

Spacing (S): The metric of spacing[24] measures how evenly the solutions are distributed along the discovered front.

$$S = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(\bar{d} - d_i)^2} \tag{10}$$

where $d_i = \min(|f_1^i(\vec{x}) - f_1^j(\vec{x})| + |f_2^i(\vec{x}) - f_2^j(\vec{x})|), i, j = 1, \ldots, n$, $\bar{d}$ is the mean of all d_i, *n* is the number of non-dominated solutions found by the algorithm. $S = 0$ indicates that all the generated solutions are evenly distributed.

Maximum spread (MS): The metric of maximum spread measures how well the true Pareto front is covered by the found Pareto front. It is defined as:

$$MS = \sqrt{\frac{1}{M}\sum_{i=1}^{M}\left[\frac{\min(f_i^{\max}, F_i^{\max}) - \max(f_i^{\min}, F_i^{\min})}{F_i^{\max} - F_i^{\min}}\right]^2} \tag{11}$$

where M is the number of objectives, $f_i^{\max}$ and $f_i^{\min}$ are the maximum and minimum of the ith objective in the found solutions, respectively. $F_i^{\max}$ and $F_i^{\min}$ are the maximum and minimum of the ith objective in the true Pareto solutions, respectively. $MS = 1$ means the generated solutions cover the full extent of the true Pareto front.

5.2. *Benchmark problems*

In this paper, seven benchmark problems SCH, FON, POL, ZDT1, ZDT3, ZDT4, and ZDT6 are selected to examine the performance of the proposed KMOPSO. Many researchers have applied these problems to examine their proposed algorithms.[4, 11, 13, 16] The definition of these problems is summarized in Table 1.

Table 1 . Test problems used in this paper.

Test Problem	Objective Functions
SCH	$f_1(x) = x^2$ $f_2(x) = (x-2)^2$ $where\ -10^3 \le x \le 10^3.$
FON	$f_1(x) = 1 - \exp(-\sum_{i=1}^{3}(x_i - 1/\sqrt{3})^2)$ $f_2(x) = 1 - \exp(-\sum_{i=1}^{3}(x_i + 1/\sqrt{3})^2)$ $where\ \ -4 \le x_i \le 4.$
POL	$f_1(x) = 1 + (A_1 - B_1)^2 + (A_2 - B_2)^2$ $f_2(x) = (x_1+3)^2 + (x_2+1)^2$ $A_1 = 0.5\sin 1 - 2\cos 1 + \sin 2 - 1.5\cos 2$ $A_2 = 1.5\sin 1 - \cos 1 + 2\sin 2 - 0.5\cos 2$ $B_1 = 0.5\sin x_1 - 2\cos x_1 + \sin x_2 - 1.5\cos x_2$ $B_2 = 1.5\sin x_1 - \cos x_1 + 2\sin x_2 - 0.5\cos x_2$ $where\ \ -\pi \le x_i \le \pi.$
ZDT1	$f_1(x) = x_1$ $f_2(x) = 1 - \sqrt{f_1/g}$ $g(x) = 1 + 9(\sum_{i=2}^{m} x_i)/(m-1)$ $where\ \ m = 30, and\ 0 \le x_i \le 1$
ZDT3	$f_1(x) = x_1$ $f_2(x) = 1 - \sqrt{f_1/g} - (f_1/g)\sin(10\pi f_1)$ $g(x) = 1 + 9(\sum_{i=2}^{m} x_i)/(m-1)$ $where\ \ m = 10, and\ 0 \le x_i \le 1$
ZDT4	$f_1(x) = x_1$ $f_2(x) = 1 - \sqrt{f_1/g}$ $g(x) = 1 + 10(m-1) + \sum_{i=2}^{m}(x_i^2 + 10\cos(4\pi x_i))$ $where\ \ m = 10, and\ 0 \le x_1 \le 1, -5 \le x_i \le 5 (i = 2,\ldots,m)$
ZDT6	$f_1(x) = 1 - \exp(-4x_1)\sin^6(6\pi x_1)$ $f_2(x) = 1 - -(f_1/g)^2$ $g(x) = 1 + 9((\sum_{i=2}^{m} x_i/(m-1))^{0.25})$ $where\ \ m = 10, and\ 0 \le x_i \le 1$

5.3. *Comparative algorithms*

The performance of KMOPSO is compared to three MO algorithms, including NSGA-II, MOPSO, and MOPSO-CD. NSGA-II[4] is an improved version of the NSGA[25] (non-dominated sorting genetic algorithm). It incorporates elitism and a crowding distance operator that keeps diversity. It has been very popular in the last a few years, becoming a landmark against which other MO optimization algorithms have to be compared. MOPSO[10] was proposed by Coello *et al.*. MOPSO shows competitive results compared to PAES[26] and NSGA-II. MOPSO-CD[11] is a revised version of MOPSO. The experimental results show this algorithm is competitive in converging towards the Pareto front and generating a well distributed set of non-dominated solutions.

5.4. *Parameter settings*

The experiments are performed on a machine with Intel(R) Core(TM) 2 Quad CPU E7300 at 2.66 GHz and 2.00 GB of RAM. The operating system is MS Windows XP and the compiler is VC++ 6.0.

In this study, the NSGA-II used a population size of 100, a crossover rate 0.8, tournament selection, and a mutation rate of $1/L$, where L is the number of decision variables. The MOPSO used a population of 100 particles, 30 divisions for the adaptive grid, and a mutation rate of 0.5. ω, r_1, and r_2 were 0.4, 1.0, and 1.0, respectively. The MOPSO-CD was run using 100 particles in the population, and a mutation rate of 0.5. ω, r_1, and r_2 were 0.4, 1.0, and 1.0, respectively. KMOPSO used 100 particles in the population. ω, r_1, and r_2 were 0.3, 1.5, and 1.5, respectively. The number of cycles in the K-means algorithm was 20. The mutation rate was 0.1 except in ZDT4 (the mutation rate was set as 0.2 in this test function.) The number of K is crucial to the algorithm. If K is too small, the chosen K particles are unable to cover the distribution of the whole non-dominated front. If K is too large, the chosen particles lose representativeness. In our research, the number of clutters was time varying. It is shown as follows:

$$K = \begin{cases} |A| & when \;\; 0 < A \le 3 \\ 3 & when \;\; 3 < A \le 10 \\ 5 & when \;\; 10 < A \le 30 \\ 10 & when \;\; 30 < A \le 100 \end{cases} \tag{12}$$

According to our experiment, when the number of particles in the archive is more than 30, the number of K should be between 10 and15 in order to achieve good performance.

The total number of function evaluation was 10,000 for SCH, FON, KUR, and POL, 40,000 for ZDT1, and ZDT4, 20,000 for ZDT3 and ZDT6. Each experiment was repeated 30 times to restrict the influence of random effects. In this paper, the best average results obtained with respect to each metric are shown in **boldface**.

5.5. *Results and discussions*

Figs. 3, 5, 7, 9, 11, 13, and 15 show the Pareto fronts generated by KMOPSO on the seven test problems. The solutions shown here correspond to the median value with respect to the generational distance. Figs. 4, 6, 8, 10, 12, 14, and 16 show the box plots of KMOPSO, NSGA-II, MOPSO, and MOPSO-CD on these test problems. In these seven figures, X-axis is the label of each algorithm. Y-axis denotes the value of each algorithm considering different metrics. Tables 2-22 show the comparison of the three algorithms considering the three metrics.

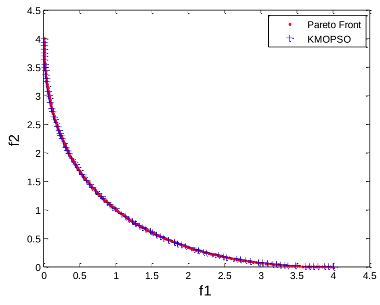


Fig. 3. Pareto fronts produced by KMOPSO for SCH.

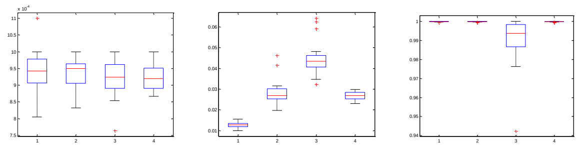


Fig. 4. Box plots for the metrics of generational distance (a), spacing (b), maximum spread(c), KMOPSO (1), NSGA-II (2), MOPSO (3), MOPSO-CD (4).

Table 2 . Results of the generational distance metric for SCH.

GD	KMOPSO	NSGA-II	MOPSO	MOPSO-CD
Best	8.05E-04	8.32E-04	7.64E-04	8.67E-04
Worst	0.0011	1.00E-03	1.00E-03	1.00E-03
Average	9.44E-04	9.39E-04	**9.21E-04**	9.24E-04
Median	9.43E-04	9.50E-04	9.24E-04	9.20E-04
Std. Dev.	5.78E-05	4.32E-05	5.16E-05	3.74E-05

Table 3 . Results of the spacing metric for SCH.

S	KMOPSO	NSGA-II	MOPSO	MOPSO-CD
Best	0.01	0.0198	0.0322	0.0231
Worst	0.0156	0.0461	0.0641	0.0299
Average	**0.0126**	0.0284	0.0442	0.0268
Median	0.0128	0.0268	0.0435	0.0268
Std. Dev.	0.0014	0.0057	0.0071	0.0019

TABLE 4 . Results of the maximum spread metric for SCH.

MS	KMOPSO	NSGA-II	MOPSO	MOPSO-CD
Best	1	1	1	1
Worst	0.9993	0.9992	0.9423	0.9992
Average	**0.9999**	0.9998	0.9906	0.9998
Median	0.9999	0.9999	0.9938	0.9998
Std. Dev.	1.60E-04	1.83E-04	0.0113	2.03E-04

From Fig. 4(a), it can be observed that all four algorithms can converge to the true Pareto front. The average performance of MOPSO is the best with respect to the GD. But its advantage over other algorithm is not obvious. In terms of S, KMOPSO is the best. Other algorithms have much higher spacing values, especially the MOPSO. KMOPSO can cover the full extent of the true Pareto front, as illustrated by the high value of MS. NSGA-II and MOPSO-CD place slightly below KMOPSO, while MOPSO falls behind.

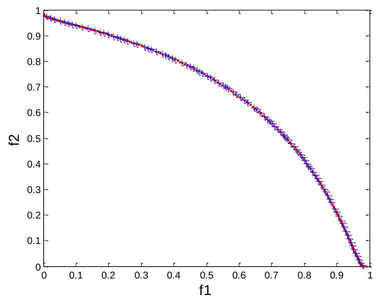
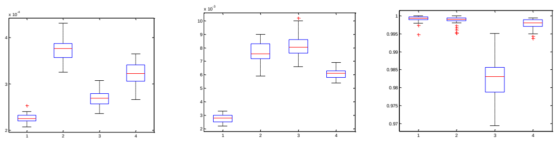


Fig. 5. Pareto fronts produced by KMOPSO for FON.



(a) (b) (c)

Fig. 6. Box plots for the metrics of generational distance (a), spacing (b), maximum spread(c), KMOPSO (1), NSGA-II (2), MOPSO (3), MOPSO-CD (4).

Table 5 . Results of the generational distance metric for FON.

GD	KMOPSO	NSGA-II	MOPSO	MOPSO-CD
Best	2.07E-04	3.25E-04	2.36E-04	2.66E-04
Worst	2.53E-04	4.31E-04	3.07E-04	3.65E-04
Average	**2.26E-04**	3.74E-04	2.70E-04	3.23E-04
Median	2.25E-04	3.77E-04	2.69E-04	3.22E-04
Std. Dev.	9.73E-06	2.44E-05	1.71E-05	2.34E-05

Table 6 . Results of the spacing metric for FON.

S	KMOPSO	NSGA-II	MOPSO	MOPSO-CD
Best	0.0022	0.0059	0.0066	0.0054
Worst	0.0033	0.0090	0.0102	0.0069
Average	**0.0027**	0.0077	0.0082	0.0061
Median	0.0028	0.0075	0.0080	0.0061
Std. Dev.	0.0003	0.0007	0.0009	0.0004

Table 7 . Results of the maximum spread metric for FON.

MS	KMOPSO	NSGA-II	MOPSO	MOPSO-CD
Best	1.0000	1.0000	0.9951	0.9994
Worst	0.9948	0.9952	0.9694	0.9937
Average	**0.9990**	0.9987	0.9827	0.9977
Median	0.9993	0.9990	0.9830	0.9980
Std. Dev.	0.0010	0.0013	0.0071	0.0015

It can be seen from Fig. 6(a) that the average performance of KMOPSO is the best with respect to the GD. And it's also able to evolve a diverse and well-distributed solution set, as shown in Fig. 6(b) and Fig. 6(c). Other algorithms fall behind KMOPSO in all three metrics in this test function.

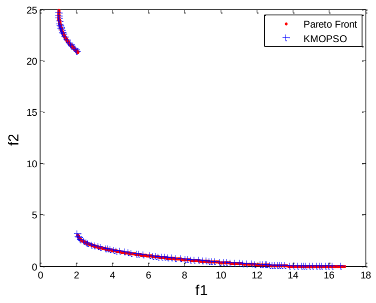
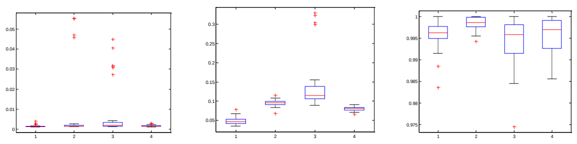


Fig. 7. Pareto fronts produced by KMOPSO for POL.



(a) (b) (c)

Fig. 8. Box plots for the metrics of generational distance (a), spacing (b), maximum spread(c), KMOPSO (1), NSGA-II (2), MOPSO (3), MOPSO-CD (4).

Table 8 . Results of the generational distance metric for POL.

GD	KMOPSO	NSGA-II	MOPSO	MOPSO-CD
Best	0.0011	0.0013	0.0013	0.0011
Worst	0.004	0.0554	0.0449	0.003
Average	**0.0015**	0.0082	0.0084	0.0017
Median	0.0013	0.0015	0.0019	0.0015
Std. Dev.	6.46E-04	0.0171	0.0135	4.54E-04

Table 9 . Results of the spacing metric for POL.

S	KMOPSO	NSGA-II	MOPSO	MOPSO-CD
Best	0.0349	0.0682	0.0891	0.0662
Worst	0.0783	0.1153	0.3296	0.0911
Average	**0.0492**	0.0947	0.141	0.0802
Median	0.0466	0.0951	0.1147	0.081
Std. Dev.	0.0099	0.0088	0.0708	5.50E-03

Table 10 **.** Results of the maximum spread metric for POL

MS	KMOPSO	NSGA-II	MOPSO	MOPSO-CD
Best	1	1	1	1
Worst	0.9836	0.9943	0.9745	0.9856
Average	0.9957	**0.9983**	0.9942	0.9957
Median	0.9962	0.9986	0.9958	0.997
Std. Dev.	0.0035	0.0016	0.0057	0.0041

It can be seen from Table 8 that the average performance of KMOPSO is the best with respect to the GD. NSGA-II and MOPSO don't perform well in several trials which affect their average values. Furthermore, the solutions found by KMOPSO are more uniformly distributed than the other algorithms. Other algorithms fail to evolve a diverse and well-distributed solution set compared to KMOPSO. By looking at maximum spread, KMOPSO plays second, tied with MOPSO-CD.

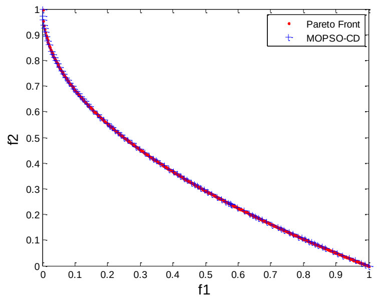


Fig. 9. Pareto fronts produced by KMOPSO for ZDT1.

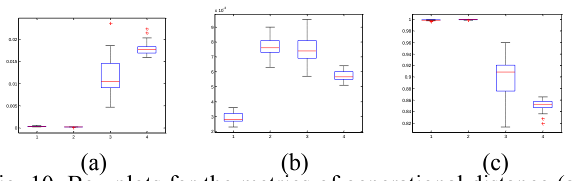


Fig. 10. Box plots for the metrics of generational distance (a), spacing (b), maximum spread(c), KMOPSO (1), NSGA-II (2), MOPSO (3), MOPSO-CD (4).

Table 11 . Results of the generational distance metric for ZDT1.

GD	KMOPSO	NSGA-II	MOPSO	MOPSO-CD
Best	2.60E-04	8.95E-05	4.70E-03	1.59E-02
Worst	5.85E-04	2.96E-04	2.36E-02	2.23E-02
Average	3.82E-04	**2.31E-04**	1.16E-02	1.79E-02
Median	3.64E-04	2.48E-04	1.05E-02	1.76E-02
Std. Dev.	8.69E-05	4.95E-05	4.50E-03	1.50E-03

Table 12 . Results of the spacing metric for ZDT1.

S	KMOPSO	NSGA-II	MOPSO	MOPSO-CD
Best	0.0023	0.0063	0.0057	0.0051
Worst	0.0036	0.0090	0.0095	0.0064
Average	**0.0029**	0.0076	0.0074	0.0057
Median	0.0028	0.0076	0.0074	0.0057
Std. Dev.	3.59E-04	6.02E-04	9.02E-04	3.67E-04

Table 13 . Results of the maximum spread metric for ZDT1.

MS	KMOPSO	NSGA-II	MOPSO	MOPSO-CD
Best	1.0000	1.0000	0.9599	0.8652
Worst	0.9964	0.9990	0.8132	0.8192
Average	0.9988	**0.9997**	0.9018	0.8509
Median	0.9990	0.9998	0.9089	0.8529
Std. Dev.	0.0009	0.0003	0.0352	0.0108

Unfortunately, MOPSO and MOPSO-CD are unable to find the true Pareto front on ZDT1. KMOPSO and NSGA-II can capture the true Pareto front. Although the average performance of NSGA-II on generational distance is slightly better than KMOPSO, it fails to generate a diverse and well-distributed solution set, as illustrated by a relatively large value of S. KMOPSO has much smaller value of S than NSGA-II. In terms of MS**,** KMOPSO and NSGA-II can both cover the full extent of the Pareto front.

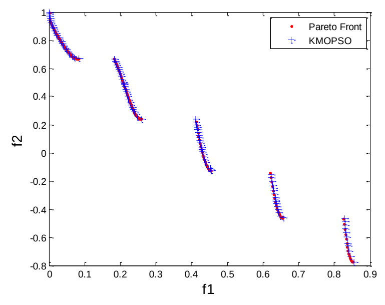


Fig. 11. Pareto fronts produced by KMOPSO for ZDT3.

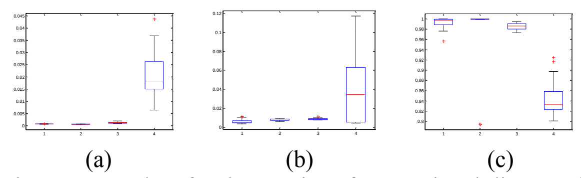


Fig. 12. Box plots for the metrics of generational distance (a), spacing (b), maximum spread(c), KMOPSO (1), NSGA-II (2), MOPSO (3), MOPSO-CD (4).

Table 14 . Results of the generational distance metric for ZDT3.

GD	KMOPSO	NSGA-II	MOPSO	MOPSO-CD
Best	6.03E-04	5.53E-04	7.59E-04	6.40E-03
Worst	8.09E-04	7.01E-04	2.00E-03	4.37E-02
Average	6.67E-04	**6.11E-04**	1.30E-03	2.04E-02
Median	6.47E-04	6.10E-04	1.20E-03	1.79E-02
Std. Dev.	5.66E-05	3.69E-05	3.20E-04	8.90E-03

Table 15 . Results of the spacing metric for ZDT3.

S	KMOPSO	NSGA-II	MOPSO	MOPSO-CD
Best	3.50E-03	5.90E-03	7.60E-03	4.30E-03
Worst	1.11E-02	9.40E-03	1.15E-02	1.17E-01
Average	**5.90E-03**	7.70E-03	8.80E-03	3.92E-02
Median	5.30E-03	7.50E-03	8.60E-03	3.46E-02
Std. Dev.	1.90E-03	9.83E-04	9.44E-04	3.49E-02

Table 16 . Results of the maximum spread metric for ZDT3.

MS	KMOPSO	NSGA-II	MOPSO	MOPSO-CD
Best	1.0000	1.0000	0.9948	0.9245
Worst	0.9569	0.7942	0.9729	0.8007
Average	**0.9923**	0.9724	0.9855	0.8451
Median	0.9967	0.9998	0.9860	0.8335
Std. Dev.	0.0097	0.0709	0.0059	0.0328

Except MOPSO-CD, other algorithms can explore the Pareto front within 20,000 function evaluations on ZDT3. With respect to the GD, KMOPSO plays second, slightly below NSGA-II. In terms of spacing, KMOPSO is much better than NSGA-II and MOPSO. Besides, KMOPSO covers the whole Pareto front in all 30 runs. NSGA-II failed to cover the whole Pareto front in several test runs, as shown in Fig. 12(c) and Table 16.

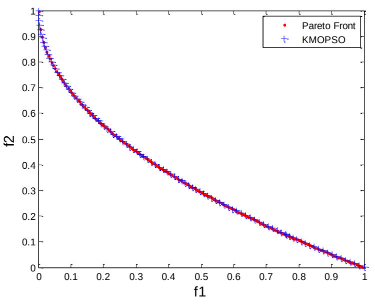
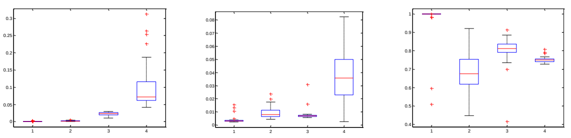


Fig. 13. Pareto fronts produced by KMOPSO for ZDT4.



(a) (b) (c)

Fig. 14. Box plots for the metrics of generational distance (a), spacing (b), maximum spread(c), KMOPSO (1), NSGA-II (2), MOPSO (3), MOPSO-CD (4).

Table 17 . Results of the generational distance metric for ZDT4.

GD	KMOPSO	NSGA-II	MOPSO	MOPSO-CD
Best	4.66E-04	7.37E-04	1.05E-02	4.16E-02
Worst	3.30E-03	5.90E-03	2.97E-02	3.13E-01
Average	**7.08E-04**	2.00E-03	2.28E-02	1.08E-01
Median	5.11E-04	1.40E-03	2.38E-02	7.20E-02
Std. Dev.	6.54E-04	1.40E-03	5.00E-03	7.32E-02

Table 18 . Results of the spacing metric for ZDT4.

S	KMOPSO	NSGA-II	MOPSO	MOPSO-CD
Best	2.40E-03	4.50E-03	5.80E-03	2.70E-03
Worst	1.54E-02	2.36E-02	3.10E-02	8.23E-02
Average	**4.20E-03**	9.60E-03	8.10E-03	3.58E-02
Median	3.20E-03	8.10E-03	7.00E-03	3.58E-02
Std. Dev.	3.10E-03	4.60E-03	4.70E-03	2.02E-02

Table 19 . Results of the maximum spread metric for ZDT4.

MS	KMOPSO	NSGA-II	MOPSO	MOPSO-CD
Best	1.0000	0.9203	0.9137	0.8088
Worst	0.5097	0.4476	0.4155	0.7283
Average	**0.9690**	0.6891	0.8022	0.7523
Median	1.0000	0.6742	0.8129	0.7496
Std. Dev.	0.1137	0.1157	0.0844	0.0177

It can be shown from Table 17 that NSGA-II, MOPSO, and MOPSO-CD have relatively large GD at the end of 40,000 evaluations. KMOPSO can jump out of the local optimal of ZDT4, resulting in a low value of GD. Table 18 indicates that KMOPSO also have good diversity with lower value of S than other algorithms. KMOPSO can cover the full extent of the Pareto front in 28 of the 30 runs.

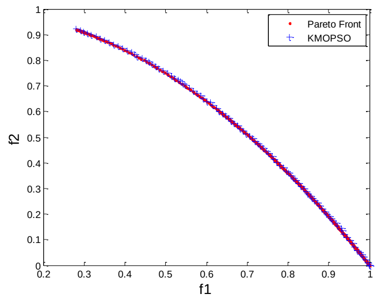
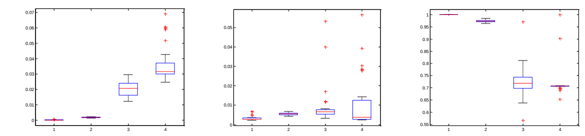


Fig. 15. Pareto fronts produced by KMOPSO for ZDT6.



(a) (b) (c)

Fig. 16. Box plots for the metrics of generational distance (a), spacing (b), maximum spread(c), KMOPSO (1), NSGA-II (2), MOPSO (3), MOPSO-CD (4).

Table 20 . Results of the generational distance metric for ZDT6.

GD	KMOPSO	NSGA-II	MOPSO	MOPSO-CD
Best	1.19E-04	1.40E-03	1.23E-02	2.48E-02
Worst	7.90E-04	2.40E-03	2.96E-02	6.92E-02
Average	**2.82E-04**	1.90E-03	2.08E-02	3.74E-02
Median	2.00E-04	1.90E-03	2.08E-02	3.16E-02
Std. Dev.	2.11E-04	2.86E-04	4.80E-03	1.22E-02

Table 21 . Results of the spacing metric for ZDT6.

S	KMOPSO	NSGA-II	MOPSO	MOPSO-CD
Best	0.0022	0.0043	0.0033	0.0023
Worst	0.0068	0.0067	0.0533	0.0566
Average	**0.0034**	0.0055	0.0096	0.0106
Median	0.0028	0.0055	0.0066	0.0038
Std. Dev.	0.0013	0.0006	0.0106	0.0136

Table 22 . Results of the maximum spread metric for ZDT6.

MS	KMOPSO	NSGA-II	MOPSO	MOPSO-CD
Best	1.0000	0.9844	0.9714	1.0000
Worst	1.0000	0.9633	0.5663	0.6526
Average	**1.0000**	0.9735	0.7243	0.7199
Median	1.0000	0.9737	0.7177	0.7077
Std. Dev.	0.0000	0.0045	0.0627	0.0651

From Table 20, it can be seen that NSGA-II, MOPSO, and MOPSO-CD fail to capture the true Pareto front. KMOPSO is the only algorithm which can converge to the true Pareto front. KMOPSO also shows good diversity and spread, as illustrated by Table 21 and Table 22.

In general, we can see that KMOPSO shows competitive results with respect to three other algorithms. For all test problems, KMOPSO can approximate the true Pareto front. The solutions generated by KMOPSO show the best diversity as KMOPSO takes the first place in all test problems in terms of the spacing metric. KMOPSO can cover the full Pareto front in all test problems. It places first in five problems and second in the rest two problems.

5.6. *Computational time*

In this section, the computational time of the four multi-objective optimization algorithms are compared. The results are shown in Table 23.

Table 23 . Computational time (in seconds) of three algorithms

		KMOPSO	NSGA-II	MOPSO	MOPSO-CD
SCH	Best	1.734	0.256	0.639	0.098
	Worst	1.921	0.534	0.672	0.436
	Average	1.8453	0.3913	0.656	**0.1843**
	Median	1.859	0.387	0.654	0.14
	Std. Dev.	0.0614	0.006	0.0118	0.1083
FON	Best	2.79	0.998	1.06	0.133
	Worst	3.728	1.509	2.388	0.378
	Average	3.2286	1.2519	1.5011	**0.322**
	Median	3.2395	1.2215	1.465	0.338
	Std. Dev.	0.2782	0.1753	0.3836	0.0693
POL	Best	2.328	0.329	1.017	0.093
	Worst	3.562	0.765	2.385	0.265
	Average	2.7931	0.5447	1.5378	**0.206**
	Median	2.755	0.548	1.1095	0.218
	Std. Dev.	0.3945	0.1435	0.5868	0.0501
ZDT1	Best	10.793	6.445	5.081	25.073
	Worst	16.462	8.984	7.716	29.12
	Average	14.635	7.4795	**6.2548**	27.4398
	Median	15.377	7.4755	6.391	27.569
	Std. Dev.	1.9065	0.7453	0.917	1.3203
ZDT3	Best	4.139	1.109	2.073	1.437
	Worst	7.841	1.876	4.446	1.921
	Average	6.1336	**1.4465**	2.5854	1.7361
	Median	6.279	1.345	2.2385	1.75
	Std. Dev.	1.1407	0.235	0.7712	0.1356
ZDT4	Best	4.718	2.468	2.509	0.296
	Worst	18.619	3.451	5.003	0.92
	Average	12.081	2.8898	3.3959	**0.4464**
	Median	11.8535	2.84	3.3435	0.358
	Std. Dev.	5.0865	0.3327	0.6831	0.1924
ZDT6	Best	4.998	1.11	1.374	0.109
	Worst	8.888	1.865	3.922	1.809
	Average	7.0211	1.4718	2.3535	**0.3726**
	Median	7.1775	1.5095	2.1635	0.234
	Std. Dev.	1.479	0.2474	0.986	0.4836

In the first three test functions, all four algorithms can find the Pareto front. MOPSO-CD is the fastest among them. It takes much less time than other three algorithms. In ZDT1, KMOPSO and NSGA-II are the only two algorithms can converge to the Pareto front while NSGA-II is much faster. In ZDT3, all four algorithms except MOPSO-CD can converge to the Pareto front. NSGA-II is the fastest one among them. In ZDT4 and ZDT6, KMOPSO is the only algorithm can achieve the Pareto front. MOPSO-CD converges very fast in these two test functions. However, it is unable to find the true Pareto front and its GD value is very high.

The reasons of the relatively low computation speed of KMOPSO is that it adopts the *K*-means guide selection strategy and nearest neighbor based pruning method which are a little more complex than the crowding distance in MOPSO-CD and NSGA-II and the adaptive grid in MOPSO.

6. Conclusions

This paper proposes a KMOPSO to solve multi-objective optimization problems which employs a novel *gbest* selection strategy based on *K*-means algorithm and proportional distribution. This *gbest* selection method can encourage particles converge fast towards the Pareto front while maintaining diversity. A symmetric mutation operator is employed to improve the exploratory abilities and prevent particles from falling into local optima. A new *pbest* is defined in order to help particles search more regions in the search space. Seven test functions were adopted to test the effectiveness of our method. The experimental results show the approach has several advantages comparing with some other MO evolutionary algorithms:

(i) KMOPSO has a good search capability, and is able to converge to the true Pareto front of all seven test functions.
(ii) KMOPSO is able to find non-dominated solutions distributed uniformly along the Pareto front and has the best diversity among the comparative algorithms.
(iii) KMOPSO is the only one that can cover the entire Pareto fronts of all test functions among the comparative algorithms.

Although KMOPSO shows good performance in the quality of the solutions, it still has some drawbacks. The most important drawback of the KMOPSO is its computational complexity. The computational time of KMOPSO is the longest compared with other comparative algorithms. Despite of this, the algorithm is highly competitive in terms of its convergence, diversity and spread of the generated solutions.

Acknowledgements

This project was supported by the National High Technology Research and Development Program (863 Project) (Grant No. 2009AA04Z120).

Appendix

Table 24 gives the list of symbols used in this paper.

Table 24. List of symbols

M	Population size
x	Position of particle
v	Speed of particle
ω	Inertial weight
c_1/c_2	Cognitive/social weight
gbest	Global best
pbest	Personal best
N	Archive size
t	Iteration counter
K	Number of clusters
p	Probability of being chosen as *gbest*
num_i	Number of particles in cluster i
C_i	Cluster i
$\mathbf{m}_i$	Mean of cluster C_i
lb	Lower bounder
ub	Upper bounder
mp	Middle point
GD	Generational distance
S	Spacing
MS	Maximum spread
distolb	distance to the lower bound
distomp	distance to the middle point
d_i	Euclidean distance between solution i and its nearest Pareto optimal solution
$\overline{d}$	Mean value of all d_i
$f_i^{\max} / f_i^{\min}$	Maximum/minimum of the ith objective in the found solutions
$F_i^{\max} / F_i^{\min}$	Maximum/minimum of the ith objective in the Pareto solutions

References

1. C. A. Coello Coello, Evolutionary multi-objective optimization: A historical view of the field, *IEEE Computational Intelligence Magazine* 1(1) (2006) 28-36.
2. Schaffer, J. D., Multiple objective optimization with vector evaluated genetic algorithms. PhD thesis, Vanderbilt University, 1984.
3. Schaffer, J. D., Multiple objective optimization with vector evaluated genetic algorithms, in: *Proc. IEEE International Conference on Genetic Algorithms*, (1985), pp. 93–100.
4. Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal and T. Meyarivan, A fast and elitist multiobjective genetic algorithm NSGA-II, *IEEE Transactions on Evolutionary Computation.* 6 (2002) 182–197.
5. Eckart Zitzler, Marco Laumanns and Lothar Thiele, SPEA2: Improving the strength pareto evolutionary algorithm, Computer Engineering and Communication Networks Lab (TIK), Swiss Fed. Inst. Technol. (ETH), Zurich, Switzerland, Tech. Rep. 103, 2001.
6. C. A. Coello Coello and G. T. Pulido, Multiobjective optimization using a micro-genetic algorithm, in: *Proc. Genetic and Evolutionary Computation Conference, GECCO'2001*, (San Francisco, CA, 2001), pp. 274–282.
7. J. Kennedy and R. C. Eberhart, Particle swarm optimization, in: *Proc. IEEE International Conference on Neural Networks*, (Perth, Australia, 1995), pp.1942–1948.
8. J. Kennedy and R. C. Eberhart, *Swarm Intelligence* (San Mateo, CA: Morgan Kaufmann, 2001).
9. Junjie Yang, Jianzhong Zhou, Li Liu and Yinghai Li, A novel strategy of pareto-optimal solution searching in multi-objective particle swarm optimization (MOPSO), *Computers and Mathematics with Applications*. 57 (2009) 1995-2000.
10. X. Hu and R. Eberhart, Multiobjective optimization using dynamic neighborhood particle swarm optimization, in: *Proc. IEEE Congress on Evolutionary Computation, CEC'2002*, (Honolulu, HI, 2002), pp. 1677–1681.
11. C. A. C. Coello, G. T. Pulido and M. S. Lechuga, Handling multiple objectives with particle swarm optimization, *IEEE Transactions on Evolutionary Computation.* 8(3) (2004) 256–279.
12. C. R. Raquel, P.C. Naval, An effective use of crowding distance in multiobjective particle swarm optimization, in: *Proc. Genetic and Evolutionary Computation Conference, GECCO'2005*, (Washington DC, USA, 2005), pp. 257–264.
13. Xiaodong Li, A Nondominated Sorting Particle Swarm Optimizer for Multiobjective Optimization, in: *Proc. Genetic and Evolutionary Computation Conference, GECCO'2003, in Lecture Notes in Computer Science*, (Berlin, Germany, 2003), pp. 37-48.
14. Mostaghim, S, Teich, J, Strategies for finding good local guides in multi-objective particle swarm optimization (MOPSO), in: *Proc. 2003 IEEE Swarm Intelligence Symposium*, (Indianapolis, 2003), pp.26- 33.
15. Maximino Salazar-Lechuga and Jonathan E. Rowe, Particle swarm optimization and fitness sharing to solve multi-mbjective optimization problems, in: *Proc. IEEE Congress on Evolutionary Computation, CEC'2005*, (Edinburgh, Scotland, 2005), pp. 1204-1211.
16. Dasheng Liu, K. C. Tan and C. K. Goh, W. K. Ho, A Multiobjective Memetic Algorithm Based on Particle Swarm Optimization, *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*. 37(1) (2007) 42 – 50.
17. Shang-Jeng Tsai, Tsung-Ying Sun, Chan-Cheng Liu, Sheng-Ta Hsieh, Wun-Ci Wua and Shih-Yuan Chiu, An improved multi-objective particle swarm optimizer for multi-objective problems, *Expert Systems with Applications*. 37 (2010) 5872–5886.

18. D. Lei, A Pareto archive particle swarm optimization for multi-objective job shop scheduling, *Computers & Industrial Engineering*. 54(4) (2008) 960–971.
19. A. B. De Carvalho, A. Pozo and S. R. Vergilio, A symbolic fault-prediction model based on multi-objective particle swarm optimization. *Journal of Systems and Software*. 83(5) (2010) 868–882.
20. James Kennedy, Stereotyping: improving particle swarm performance with cluster analysis, in: *Proc. IEEE Congress on Evolutionary Computation, CEC'2000*, (La Jolla, CA, 2000), pp. 1204-1211.
21. Alessandro Passaro and Antonina Starita, Particle swarm optimization for multimodal functions: a clustering approach, *Journal of Artificial Evolution and Applications*. (2008) 1-15.
22. X. Wu, V. Kumar, J. Quinlan, J. Gosh, Q. Yang, H. Motoda, G. MacLachlan, A. Ng, B. Liu, P. Yu, Z. Zhou, M. Steinbach, D. Hand and D. Steinberg, Top 10 algorithms in data mining, *Knowledge and Information Systems 14*. (2008) 1–37.
23. D. A. Van Veldhuizen and G. B. Lamont, Multiobjective evolutionary algorithm research: a history and analysis, Dept. Elec. Comput. Eng.,Graduate School of Eng., Air Force Inst. Technol., Wright-Patterson AFB, OH, Tech. Rep. TR-98-03, 1998.
24. J.R. Schott, Fault tolerant design using single and multicriteria genetic algorithm optimization, Master's Thesis, Cambridge, Massachusetts, 1995.
25. N. Srinivas and K. Deb, Multiobjective function optimization using nondominated sorting genetic algorithms, *IEEE Transactions on Evolutionary Computation*. 2(3) (1995) 221–248.
26. J. D. Knowles and D.W. Corne, Approximating the nondominated front using the Pareto archived evolution strategy, *Evolutionary Computation*, 8(2000) 149–172.