Neural Comput & Applic (2008) 17:101–109
DOI 10.1007/s00521-007-0118-6

BIC-TA 2006

Multiobjective optimization using population-based extremal optimization

Min-Rong Chen · Yong-Zai Lu · Genke Yang

Received: 30 October 2006 / Accepted: 2 April 2007 / Published online: 26 April 2007

Abstract In recent years, a general-purpose local-search heuristic method called Extremal Optimization (EO) has been successfully applied in some NP-hard combinatorial optimization problems. In this paper, we present a novel Pareto-based algorithm, which can be regarded as an extension of EO, to solve multiobjective optimization problems. The proposed method, called Multiobjective Population-based Extremal Optimization (MOPEO), is validated by using five benchmark functions and metrics taken from the standard literature on multiobjective evolutionary optimization. The experimental results demonstrate that MOPEO is competitive with the state-of-the-art multiobjective evolutionary algorithms. Thus MOPEO can be considered as a viable alternative to solve multiobjective optimization problems.

Keywords Multiobjective optimization · Extremal optimization · Self-organized criticality · Pareto front

1 Introduction

Most real-world engineering optimization problems are multiobjective in nature, since they normally have several (possible conflicting) objectives that must be satisfied at the same time. Instead of aiming to find a single solution, the multiobjective optimization methods try to produce a set of good ''trade-offs'' from which the decision maker may select one.

Over the past two decades, a great amount of multiobjective evolutionary algorithms have been proposed [1]. Evolutionary algorithms seem particularly suitable to solve multiobjective optimization problems, because they deal simultaneously with a set of possible solutions. This allows us to find several Pareto optimal solutions in a single run of the algorithm, instead of performing a series of separate runs as in the case of the traditional mathematical programming techniques [1]. In addition, evolutionary algorithms can easily deal with discontinuous or concave Pareto fronts, whereas these two issues are a real concern for mathematical programming techniques.

Recently, a general-purpose local-search heuristic algorithm named Extremal Optimization (EO) was presented by Boettcher and Percus [2]. EO is based on the Bak–Sneppen model [3], which shows the emergence of self-organized criticality (SOC) [4] in ecosystems. The evolution in this model is driven by a process where the weakest species in the population, together with its nearest neighbors, is always forced to mutate. The dynamics of this extremal process exhibits the characteristics of SOC, such as punctuated equilibrium [3]. EO opens the door to applying non-equilibrium process, while the simulated annealing (SA) applies equilibrium statistical mechanics. In contrast to genetic algorithm (GA) which operates on an entire ''gene-pool'' of huge number of possible solutions, EO successively eliminates those worst components in the sub-optimal solutions. Its large fluctuations provide significant hill-climbing ability, which enables EO to perform well particularly at the phase transitions. EO has been successfully applied to some NP-hard combinatorial optimization problems such as graph bi-partitioning [2], TSP [2], graph coloring [5], spin glasses [6], MAXSAT [7].

M.-R. Chen (✉) · Y.-Z. Lu · G. Yang
Department of Automation, Shanghai Jiao Tong University, Shanghai 200240, China
e-mail: auminrongchen@sjtu.edu.cn

Y.-Z. Lu
e-mail: yzlu@sjtu.edu.cn

G. Yang
e-mail: gkyang@sjtu.edu.cn

So far there have been some papers which studied the multiobjective optimization using extremal dynamics. Ahmed and Elettreby [8] introduced random version of Bak–Sneppen model. They also generalized the single objective Bak–Sneppen model to a multiobjective one by weighted sum of objectives method. The method is easy to implement but its most serious drawback is that it cannot generate proper members of the Pareto-optimal set when the Pareto front is concave regardless of the weights used [9]. Galski et al. [10] presented a multiobjective version of the Generalized Extremal Optimization (GEO) algorithm, called M-GEO. Since the fitness assignment in the M-GEO is not based on the Pareto dominance strategy, M-GEO belongs to the non-Pareto approach [10].

In this paper, we develop a novel Pareto-based algorithm named Multiobjective Population-based Extremal Optimization (MOPEO), which can be considered as a variation of EO to solve multiobjective optimization problems. The fitness assignment of MOPEO is based on the Pareto domination, which is popularly used by many existing multiobjective evolutionary algorithms. Our approach has been validated by five standard test functions reported in the specialized literature and compared against three highly competitive multiobjective evolutionary algorithms: the Nondominated Sorting Genetic Algorithm-II (NSGA-II) [11], the Pareto Archived Evolution Strategy (PAES) [12] and the Strength Pareto Evolutionary Algorithm (SPEA) [13]. The simulation results indicate that MOPEO may be a good alternative to solve the multiobjective optimization problems.

2 Extremal optimization

2.1 Bak–Sneppen model

To make a good understanding of the underlying mechanism of EO, we will first describe the Bak–Sneppen model [3] in detail. Species in the Bak–Sneppen model are located on the sites of a lattice. Each species is assigned a fitness value randomly with uniform distribution. At each update step, the worst adapted species is always forced to mutate. The change in the fitness of the worst adapted species will cause the alteration of the fitness landscape of its neighbors. After a number of iterations, the system evolves to a highly correlated state known as self-organized criticality (SOC). In the SOC state, a little change of one species will result in co-evolutionary chain reactions called ''avalanches''.

2.2 Extremal optimization

Inspired by the Bak–Sneppen model, Boettcher and Percus proposed the EO algorithm for a minimization problem as follows [2]:

1. Randomly generate a solution S. Set optimal solution $S_{\text{best}} = S$ and the minimum cost function $C(S_{\text{best}}) = C(S)$.
2. For the current solution S,

(a) evaluate the fitness λ_i for each variable x_i, $i \in \{1, 2, ..., n\}$,
(b) rank all the fitnesses and find the variable x_j with the lowest fitness, i.e., $\lambda_j \leq \lambda_i$ for all i,
(c) choose one solution S', in the neighborhood of S, such that the jth variable must change its state,
(d) accept $S = S'$, *unconditionally*,
(e) if $C(S) < C(S_{\text{best}})$ then set $S_{\text{best}} = S$.

3. Repeat at Step 2 as long as desired.
4. Return S_{best} and $C(S_{\text{best}})$.

3 Multiobjective population-based extremal optimization

It has been proved that evolutionary algorithms are suitable for the multiobjective optimization problems due to its ability to create multiple Pareto-optimal solutions in a single simulation run [1]. Furthermore, a simple evolutionary algorithm can be extended to maintain a diverse set of solutions also because of its population mechanism. However, the traditional EO performs a search through sequential changes on a single solution, namely, the point-to-point search rather than the population based search applied in GA. In order to extend EO to solve the multiobjective problems, we developed a novel real-coded EO search algorithm, so-called MOPEO, through introducing the population search strategies being popularly used in evolutionary algorithms into EO. Similar to the evolutionary algorithms, MOPEO operates on the evolution of solutions generation after generation. Let each solution consists of n decision variables. Similar to EO, the MOPEO performs only one operation, i.e., mutation, on each variable of every solution.

3.1 Main algorithm

For a multiobjective optimization problem, the proposed MOPEO algorithm works as follows.

1. Generate initial population with N solutions, $S_i = (x_{i1}, x_{i2}, ..., x_{in})$, $i \in \{1, ..., N\}$, randomly and uniformly. Set the external archive empty. Set *iteration* = 0.
2. For each solution S_i, $i \in \{1, ..., N\}$,

(a) generate n offspring of the current solution S_i by performing mutation on each variable one by one;
(b) perform dominance ranking on the n offspring and then obtain their rank numbers, i.e., $r_{ij} \in [0, n-1]$, $j \in \{1, ..., n\}$;

(c) assign the fitness $\lambda_{ij} = r_{ij}$ for each variable x_{ij}, $j \in \{1, ..., n\}$;
(d) if there is only one variable with fitness value of zero, the variable will be considered as the worst adapted species; otherwise, the diversity preservation mechanism is invoked. Assuming that the weakest species is x_{iw} with fitness $\lambda_{iw} = 0$, $w \in \{1, ..., n\}$,
(e) perform mutation only on x_{iw} while keeping other variables unchanged, then get a new solution S_{iw};
(f) accept $S_i = S_{iw}$ *unconditionally*.

3. Identify the nondominated solutions in the new population.
4. Update the external archive by comparing the nondominated solutions in the new population with those in the archive.
5. If the external population has exceeded the maximum allowable capacity, reduce the external archive by crowding-distance computation; else if the iterations reach the predefined maximum number of the generations, go to Step 6; otherwise,set *iteration = iteration + 1, and go to Step 2.*
6. Return the external archive as the Pareto-optimal set.

3.2 Fitness assignment

It is important to note that, in MOPEO, each decision variable in one solution is considered as a species. Being different from the multiobjective evolutionary algorithm, the fitness assignment of MOPEO is carried out in each solution rather than in the population. That is, we will find out all the worst adapted species for all solutions in the current population via fitness assignment and perform mutation on them. In our work, we adopt the Pareto-based fitness assignment strategy. We use the dominance ranking [14], i.e., the number of solutions by which a solution is dominated, to determine the fitness value for each solution. In MOPEO, the dominance ranking is carried out in the offspring, which are generated by mutating the variables of the current solution in turn. Therefore, the nondominated offspring are ranked as zero, whilst the worst possible ranking is the number of decision variables minus one. So the species corresponding to the offspring with the fitness value of zero is considered as the weakest one and will be chosen to mutate.

To be clearer, we illustrate the process of dominance ranking in MOPEO using Fig. 1a. The locations of the solutions in the objective space (marked with black solid circles) will change to new ones (shown with grey dashed circles) in the next generation via mutating their weakest species. For example, given a solution $\mathbf{S}_i = (x_1, x_2, x_3, x_4)$, of which the location in the objective space is denoted by the circle i, we can identify the weakest species by mutating the four variables one by one and then performing dominance ranking on the newly generated offspring. First, an offspring $\mathbf{S}_{i\mathrm{A}} = (x_1', x_2, x_3, x_4)$ can be obtained by mutating x_1 to x_1', keeping other variables unchanged. Similarly, the other three offspring, i.e., $\mathbf{S}_{i\mathrm{B}} = (x_1, x_2', x_3, x_4)$, $\mathbf{S}_{i\mathrm{C}} = (x_1, x_2, x_3', x_4)$, $\mathbf{S}_{i\mathrm{D}} = (x_1, x_2, x_3, x_4')$, are generated. The four white dashed circles (i.e., A, B, C, D) in Fig. 1a indicate the locations of four newly generated offspring (i.e., $\mathbf{S}_{i\mathrm{A}}$, $\mathbf{S}_{i\mathrm{B}}$, $\mathbf{S}_{i\mathrm{C}}$, $\mathbf{S}_{i\mathrm{D}}$) in the objective space, respectively. The next location of the solution $\mathbf{S}_i$ in the objective space depends on the dominance ranking number of the four newly generated offspring. It can be seen from Fig. 1a that, for the circle i, circle A is nondominated by the other three dashed circles (i.e., B, C, D). Thus, the rank number of A is zero. Hence, the species x_1 corresponding to A is considered as the weakest species and the solution $\mathbf{S}_i$ will change to $\mathbf{S}_{i\mathrm{A}}$ in the next step.

If there exists more than one worst adapted species, i.e., at least two variables own the same fitness value of zero, then the following diversity preservation mechanism will be invoked.

3.3 Diversity preservation

The goal of introducing the diversity preservation mechanism is to maintain a good spread of solutions in the obtained set of solutions. In this study, we propose a new approach to keep good diversity of nondominated solutions. It is worth pointing out that our approach does not

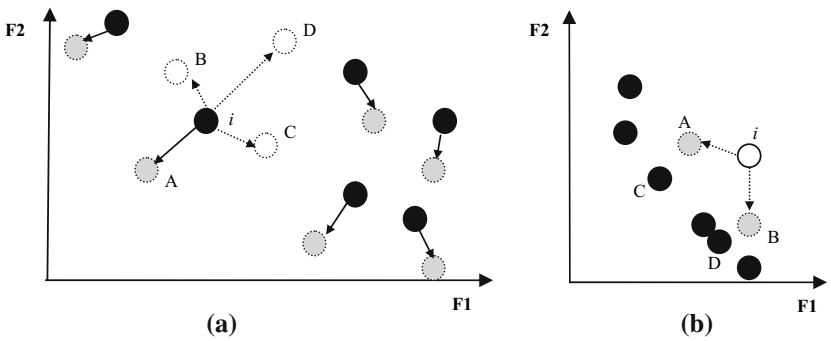

Fig. 1 **a** Shows the process of dominance ranking in MOPEO; **b** shows the diversity preservation in MOPEO

require any user-defined parameter for maintaining diversity among population members. Assuming that one solution has at least two species with the same fitness value of zero, we consider the species, by which the new solution generated approaches the less crowded region in the external archive, as the weakest one. As shown in Fig. 1b, the solution i has two offspring (i.e., A and B) who do not dominate each other. Suppose two nondominated solutions, i.e., C and D, who exist in the external archive, have the minimum Euclidean distances from A and B, respectively. Then we calculate the density value of C and D in the external population by using the k-nearest neighbor method. It can be seen from Fig. 1b that C is less crowded than D. So we choose the one corresponding to C, i.e., A, as the new location of solution i in the next step.

The k-nearest neighbor method requires sorting the external population according to each objective function value in ascending order of magnitude. The density values of the boundary solutions equal to the Euclidean distances of them from their nearest neighbors, while the density value of all other intermediate solutions can be calculated by the following:

$$\widehat{Y}_k(\mathbf{X}) = \frac{1}{k} \sum_{x_i \in N_k(\mathbf{X})} y_i, \tag{1}$$

where y_i is Euclidean distance of x_i from $\mathbf{X}$, $N_k(\mathbf{X})$ neighborhood of $\mathbf{X}$ that contains exactly k neighbors. To the best of our knowledge, so far there has been no rational method to determine the value of k, which may be one drawback of k-nearest neighborhood method. Note that the computational cost will increase with the value of k. So, in this study, we set k to 2.

3.4 External archive

The main objective of the external archive is to keep a historical record of the nondominated individuals found along the search process. The external repository consists of two main components: the archive controller and the crowding-distance metric.

– **The archive controller**: Inspired by [15], we introduce the archive controller into our approach. The function of the archive controller is to decide whether the nondominated solutions found in the new population should be added to the archive or not. The decision-making process is similar to that in [15]. After comparing all the nondominated solutions in the new population with respect to the external population, if the external population has exceeded its maximum allowable capacity, then the crowding-distance computation procedure is invoked.

– **Crowding-distance metric**: In our approach, we adopt the crowding-distance metric proposed by Deb et al. [11] to truncate the external archive when the external population has exceeded its maximum allowable capacity. For more details, the readers can refer to [11].

3.5 Mutation operator

There is merely mutation operator in MOPEO. Therefore, the mutation plays a key role in MOPEO search that generates new solutions through adding or removing genes at the current solutions, i.e., chromosomes. Till now, there have been many mutation operations proposed, such as Gaussian mutation, Cauchy mutation and nonuniform mutation. The mechanisms of Gaussian and Cauchy mutation operations have been studied by Yao et al. [16]. They pointed out that Cauchy mutation is better at coarse-grained search while Gaussian mutation is better at fine-grained search. The non-uniform mutation introduced by Janikow and Michalewicz [17] is designed for fine-tuning the solutions. At early generations, the mutation range is relatively large, while at the latter generations, it is tightened for local refinement. Thus, the non-uniform mutation operation combines the advantages of coarse-grained search and fine-grained search.

In this study, we use the non-uniform mutation as mutation operator. Suppose the kth variable is chosen to mutate, the value of kth variable after mutation is calculated as follows [17]:

$$x_k^{t+1} = \begin{cases} x_k^t + \Delta(t, u_k - x_k^t), & a > 0.5 \\ x_k^t - \Delta(t, x_k^t - l_k), & a \leq 0.5 \end{cases} \tag{2}$$

where $k = 1, \ldots, n$, n is number of decision variables, a random number in [0,1], l_k and u_k lower and upper bounds of the kth variable, respectively. The function $\Delta(t,y)$ is as follows:

$$\Delta(t, y) = yr^{(1-t/T)b} \tag{3}$$

where r is random number with range of [0,1], T maximal generation number, and b system parameter determining the degree of non-uniformity. Thus, there is only one adjustable parameter b in the non-uniform mutation. The function $\Delta(t,y)$ returns a value in the range $[0,y]$ such that the probability of $\Delta(t,y)$ being close to 0 increases as t increases (t is generation number). This property causes this operator to search the space uniformly initially (when t is small), and very locally at the later stages. The value of b affects the performance of non-uniform mutation. As we can see from Eq. 3, the searching step size, i.e., $\Delta(t,y)$, increases with the decreasing value of b. So the value of b

can adjust the searching step size of non-uniform mutation. As can be seen from the aforementioned, the non-uniform mutation is helpful for our approach to perform exploration and exploitation.

4 Experiments and test results

4.1 Test problems

We choose five problems out of six test problems proposed by Zitzler et al. [18] and call them ZDT1, ZDT2, ZDT3, ZDT4, and ZDT6. All problems have two objective functions. None of these problems have any constraint. We describe these problems in Table 1. The table also shows the number of variables, their bounds, the Pareto-optimal solutions, and the nature of Pareto-optimal front for each problem.

Deb et al. [11] have compared the performance of NSGA-II with PAES and SPEA and obtained their experimental results in [11]. The experimental results in [11] demonstrated that, in most problems, NSGA-II is able to find much better spread of solutions and better convergence near the true Pareto-optimal front compared to PAES and SPEA. In this paper, we apply the MOPEO to solve five difficult test problems in Table 1 and compare our simulation results with those obtained in [11] under the same conditions. To be fair, we adopt the identical parameter settings as suggested in [11]. For four test problems (ZDT1, ZDT2, ZDT3 and ZDT6), the adjustable parameter b in the

Table 1 Test problems in this study

Problem	n	Variable bounds	Objective functions	Optimal solutions	Pareto front
ZDT1	30	[0,1]	$f_1(X) = x_1$ $f_2(X) = g(X)\left(1 - \sqrt{x_1/g(X)}\right)$ $g(X) = 1 + 9\left(\sum_{i=2}^{n} x_i\right)/(n-1)$	$x_1 \in [0,1]$ $x_2 = \cdots = x_n = 0$	Convex
ZDT2	30	[0,1]	$f_1(X) = x_1$ $f_2(X) = g(X)\left(1 - (x_1/g(X))^2\right)$ $g(X) = 1 + 9\left(\sum_{i=2}^{n} x_i\right)/(n-1)$	$x_1 \in [0,1]$ $x_2 = \cdots = x_n = 0$	Nonconvex
ZDT3	30	[0,1]	$f_1(X) = x_1$ $f_2(X) = g(X)\left(1 - \sqrt{x_1/g(X)} - \frac{x_1}{g(X)}\sin(10\pi x_1)\right)$ $g(X) = 1 + 9\left(\sum_{i=2}^{n} x_i\right)/(n-1)$	$x_1 \in [0,1]$ $x_2 = \cdots = x_n = 0$	Convex, disconnected
ZDT4	10	$x_1 \in [0,1]$ $x_i \in [-5,5]$, $i = 2, \cdots, n$	$f_1(X) = x_1$ $f_2(X) = g(X)\left(1 - \sqrt{x_1/g(X)}\right)$ $g(X) = 1 + 10(n-1) + \sum_{i=2}^{n}(x_i^2 - 10\cos(4\pi x_i))$	$x_1 \in [0,1]$ $x_2 = \cdots = x_n = 0$	Nonconvex
ZDT6	10	[0,1]	$f_1(X) = 1 - \exp(-4x_1)\sin^6 6\pi x_1$ $f_2(X) = g(X)\left(1 - (f_1(X)/g(X))^2\right)$ $g(X) = 1 + 9\left(\left(\sum_{i=2}^{n} x_i\right)/(n-1)\right)^{0.25}$	$x_1 \in [0,1]$ $x_2 = \cdots = x_n = 0$	Nonconvex, nonuniformly spaced

All objective functions are to be minimized.

non-uniform mutation is set to 0.9, while b is set to 0.1 for ZDT4. In this study, all the algorithms developed were encoded in the floating point representation. The fitness function evaluations for all the algorithms is 25,000. The maximum population size of the external archive is 100. Additionally, ten independent runs were carried out. The source codes of all experiments were coded in JAVA.

4.2 Performance measures

In this article, we used the two performance metrics proposed by Deb et al. [11] to assess the performance of our approach. For more detail, the readers can refer to [11]. The first metric Υ measures the extent of convergence to a known set of Pareto-optimal solutions. In all simulations, we address the average $\bar{\Upsilon}$ and variance σ_{Υ} of this metric obtained in ten independent runs. Deb et al. [11] has pointed out that even when all solutions converge to the Pareto-optimal front, the convergence metric does not have a value of zero. The metric will yield zero only when each obtained solution lies exactly on each of the chosen solutions.

The second metric Δ measures the extent of spread of the obtained nondominated solutions. It is desirable to get a set of solutions that spans the entire Pareto-optimal region. The second metric Δ can be calculated as follows [11]:

$$\Delta = \frac{d_f + d_l + \sum_{i=1}^{N-1} |d_i - \bar{d}|}{d_f + d_l + (N-1)\bar{d}} \tag{4}$$

where, d_f and d_l are Euclidean distances between the extreme solutions and the boundary solutions of the obtained nondominated set, d_i Euclidean distance between consecutive solutions in the obtained nondominated set of solutions, and $\bar{d}$ is the average of all distances d_i (i = 1, 2, ..., N–1), assuming that there are N solutions on the best nondominated front. Note that a good distribution would make all distances d_i equal to $\bar{d}$ and would make $d_f = d_l = 0$ (with existence of extreme solutions in the nondominated set). Consequently, for the most widely and uniformly spreadout set of nondominated solutions, Δ would be zero.

4.3 Discussion of the results

Table 2 shows the mean and variance of the convergence metric Υ obtained using four algorithms, i.e., MOPEO, NSGA-II (real-coded), SPEA and PAES. Here, all the experimental results of NSGA-II (real-coded), SPEA and PAES come from [11].

As can be observed from Table 2, MOPEO is able to converge better than the other algorithms in three problems, i.e., ZDT1, ZDT3 and ZDT6 and it is the next-best algorithm for problem ZDT2. For problem ZDT4, MOPEO performs worse than NSGA-II and PAES in terms of the convergence to the Pareto-optimal front. In all cases with MOPEO, the variance of the convergence metric in ten runs is very small except in ZDT4.

Table 3 shows the mean and variance of the diversity metric Δ obtained using all the algorithms. On problems

Table 2 Mean (first rows) and variance (second rows) of the convergence metric Υ

Algorithm	ZDT1	ZDT2	ZDT3	ZDT4	ZDT6
MOPEO	0.001062	0.001657	0.004175	2.802010	0.013400
	7.53E-05	0.016471	0.012972	1.940834	0.020698
NSGA-II (real-coded)	0.033482	0.072391	0.114500	0.513053	0.296564
	0.004750	0.031689	0.007940	0.118460	0.013135
SPEA	0.001799	0.001339	0.047517	7.340299	0.221138
	0.000001	0	0.000047	6.572516	0.000449
PAES	0.082085	0.126276	0.023872	0.854816	0.085469
	0.008679	0.036877	0.00001	0.527238	0.006664

Table 3 Mean (first rows) and variance (second rows) of the diversity metric Δ

Algorithm	ZDT1	ZDT2	ZDT3	ZDT4	ZDT6
MOPEO	0.453562	0.483912	0.654732	1.663724	0.562433
	0.070312	0.041922	0.041333	0.350441	0.336753
NSGA-II (real-coded)	0.390307	0.430776	0.738540	0.702612	0.668025
	0.001876	0.004721	0.019706	0.064648	0.009923
SPEA	0.784525	0.755148	0.672938	0.798463	0.849389
	0.004440	0.004521	0.003587	0.014616	0.002713
PAES	1.229794	1.165942	0.789920	0.870458	1.153052
	0.004839	0.007682	0.001653	0.101399	0.003916

Table 4 Mean (first rows) and variance (second rows) of the running times (in milliseconds)

Algorithm	ZDT1	ZDT2	ZDT3	ZDT4	ZDT6
MOPEO	2,756.33	2,811.42	3,121.36	2,544.78	2,913.96
	221.24	167.83	315.65	154.95	325.47
NSGA-II (real-coded)	2,906.90	2,889.83	3,154.90	2,475.57	2,746.61
	159.51	287.70	489.04	166.31	401.19
SPEA	2,912.56	2,913.48	2,923.57	2,561.72	2,834.15
	292.34	185.27	437.69	282.11	289.38
PAES	12,051.63	12,088.77	8,890.13	2,661.80	25,716.33
	1,044.91	240.52	357.17	139.36	1,685.56

ZDT3 and ZDT6, MOPEO is able to find a better spread of solutions than any other algorithm. MOPEO is the next-best algorithm for problems ZDT1 and ZDT2 while NSGA-II is the best algorithm. For problem ZDT4, MOPEO performs worse than any other algorithm with respect to diversity. In all cases with MOPEO, the variance of the diversity metric in ten runs is also small.

In order to compare the running times of MOPEO with those of other three algorithms, i.e., NSGA-II, SPEA and PAES, we also show the mean and variance of running times of each algorithm in ten runs in Table 4. To avoid any bias or misinterpretation when implementing each of the three other approaches, we adopted the public-domain versions of NSGA-II, PAES and SPEA. Note that all the

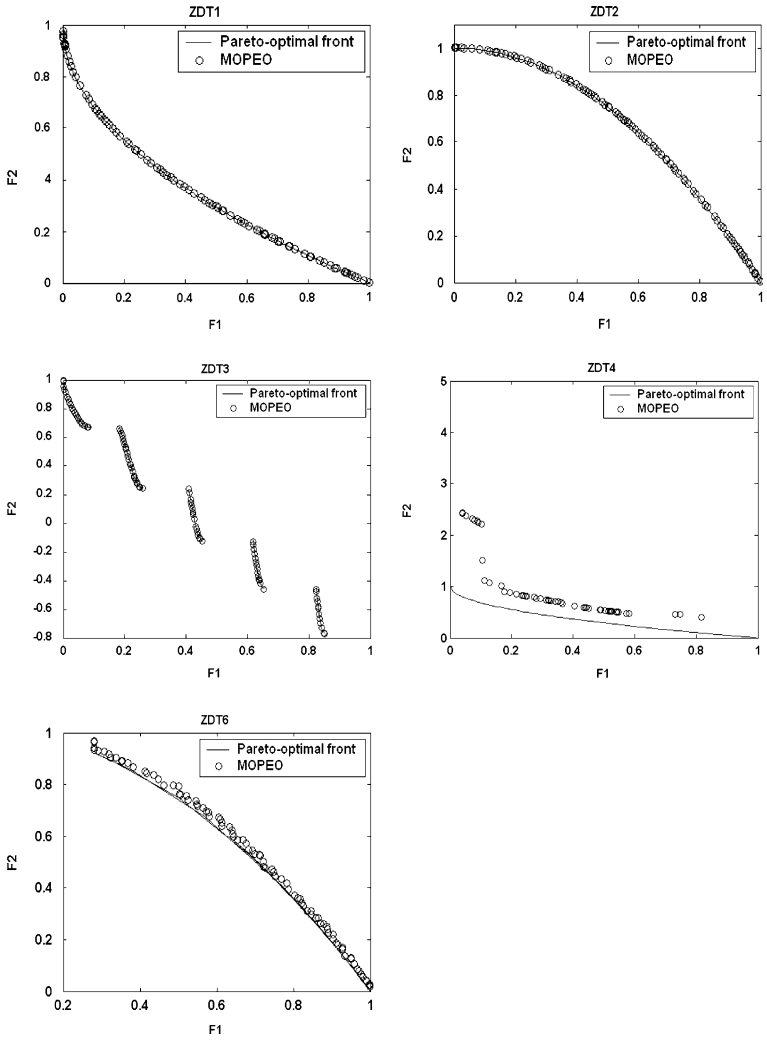

Fig. 2 Nondominated solutions with MOPEO on ZDT1, ZDT2, ZDT3, ZDT4 and ZDT6, respectively

algorithms were run on the same hardware (i.e., Intel Pentium M with 900 MHz CPU and 256 M memory) and software (i.e., JAVA) platform. The fitness function evaluations for all the algorithms is 25,000. In order to compare the running times of each algorithm fairly, we use the crowding-distance metric as the archive truncation method for MOPEO, SPEA and PAES, and as the diversity preservation mechanism for NSGA-II. From Table 4, we can see that, in terms of running times, MOPEO is competitive with other three algorithms on all the problems.

For illustration, we show one of ten runs of MOPEO on the five test problems in Fig. 2. From Fig. 2, we can see that, with respect to convergence and diversity of the obtained set of nondominated solutions, MOPEO performs well in all test problems except in ZDT4. Note that the problem ZDT4 has 21^9 different local Pareto-optimal fronts in the search space, of which only one corresponds to the global Pareto-optimal front. As can be seen from Fig. 2, MOPEO got stuck into the local Pareto-optimal fronts of problem ZDT4.

4.4 Advantages of proposed approach

From the above analysis, it can be seen that our approach has the following advantages:

- Our approach is capable of finding multiple Pareto-optimal solutions in one single simulation run due to its EO and population mechanism.
- There exists only one adjustable parameter, i.e., the system parameter b in the non-uniform mutation. Note that b can be tuned easily. This makes our approach easier in real applications than other state-of-the-art methods.
- Only one operator, i.e., mutation operator, exists in our approach, which makes our approach simple and convenient.
- Similar to evolutionary algorithms, MOPEO is less susceptible to the shape or continuity of the Pareto front.
- The proposed approach has a good performance in both aspects of convergence and distribution of solutions.

5 Conclusion and future work

In this paper, we present a novel Pareto-based algorithm, called MOPEO, to extend EO to handle multiobjective problems. Our approach has many advantages such as less adjustable parameters, only mutation operation, easily implementation. Our approach is validated to be competitive with respect to three algorithms representative of the state-of-the-art in the area. In the future work, it is desirable to improve our approach to solving those problems with multiple local Pareto-optimal fronts. In addition, we will extend MOPEO to solve constrained or discrete multiobjective optimization problems, e.g., multiobjective knapsack problems. The future work also includes the applications of MOPEO to solving those complex engineering optimization problems in the real world, e.g., multiobjective portfolio optimization problems.

Acknowledgments The authors would like to thank the anonymous referees for their many useful comments and constructive suggestions. This work is supported by the National Natural Science Foundation of China under Grant No. 60574063.

References

1. Coello CAC (2001) A short tutorial on evolutionary multiobjective optimization. Proceedings of the 1st international conference on evolutionary multi-criterion optimization (EMO 2001), Lecture notes in computer science, 1993, Zurich, Springer, Heidelberg, pp 21-40
2. Boettcher S, Percus AG (2000) Nature's way of optimizing. Artif Intell 119:275–286
3. Bak P, Sneppen K (1993) Punctuated equilibrium and criticality in a simple model of evolution. Phys Rev Lett 71(24):4083–4086
4. Bak P, Tang C, Wiesenfeld K (1987) Self-organized criticality. Phys Rev Lett 59:381–384
5. Boettcher S, Percus AG (2004) Extremal optimization at the phase transition of the 3-coloring problem. Phys Rev E 69:066703
6. Boettcher S (2005) Extremal Optimization for the Sherrington-Kirkpatrick Spin Glass. Eur Phys J B 46:501–505
7. Menai ME, Batouche M (2003) Efficient initial solution to extremal optimization algorithm for weighted MAXSAT problem. IEA/AIE 2003, pp 592–603
8. Ahmed E, Elettreby MF (2004) On multiobjective evolution model. Int J Modern Phys C 15:1189
9. Das I, Dennis J (1997) A close look at drawbacks of minimizing weighted sum of objectives for Pareto set generation in multicriteria optimization problems. Struct Optim 14:63
10. Galski RL, de Sousa FL, Ramos FM (2005) Application of a new multiobjective evolutionary algorithm to the optimum design of a remote sensing satellite constelation. In: Proceedings of the 5th international conference on inverse problems in engineering: theory and practice, Cambridge, vol II, G01
11. Deb K, Pratab A, Agrawal S, Meyarivan T (2002) A fast and elitist multiobjective genetic algorithm: NSGACII. IEEE Trans Evolut Comp 6(2):182–197
12. Knowles J, Corne D (1999) The Pareto archived evolution strategy: a new baseline algorithm for multiobjective optimization. In: Proceedings of the 1999 congress on evolutionary computation. Piscataway, IEEE Press, pp 98–105
13. Zitzler E, Thiele L (1998) Multiobjective optimization using evolutionary algorithms—a comparative case study. In: Eiben VAE, Bäck T, Schoenauer M, Schwefel H-P (eds) Parallel problem solving from nature. Springer, Berlin, pp 292–301
14. Fonseca CM, Fleming PJ (1993) Genetic algorithms for multiobjective optimization: formulation, discussion and generalization. In: Forrest S (ed) Proceedings of the fifth international conference on genetic algorithms. Morgan Kauffman, San Mateo, pp 416–423

15. Coello CAC, Pulido GT, Leehuga MS (2004) Handling multiple objectives with particle swarm optimization. IEEE Trans Evolut Comp 8(3):256–279
16. Yao X, Liu Y, Lin G (1999) Evolutionary programming made faster. IEEE Trans Evolut Comp 3(2):82–102
17. Janikow C, Michalewicz Z (1991) Experimental comparison of binary and floating point representations in genetic algorithms. In: Proceedings of the fourth international conference of genetic algorithms, pp 151–157
18. Zitzler E, Deb K, Thiele L (2000) Comparison of multiobjective evolutionary algorithms: empirical results. Evolut Comp 8(2):173–195