Journal of Zhejiang University SCIENCE A
ISSN 1673-565X (Print); ISSN 1862-1775 (Online)
www.zju.edu.cn/jzus; www.springerlink.com
E-mail: jzus@zju.edu.cn

# Multiobjective extremal optimization with applications to engineering design*

CHEN Min-rong†[1,2,3], LU Yong-zai[1], YANG Gen-ke[1]

([1]*Department of Automation, Shanghai Jiao Tong University, Shanghai 200240, China*)

([2]*College of Information Science and Technology, Jinan University, Guangzhou 510632, China*)

([3]*College of Automation Science and Engineering, South China University of Technology, Guangzhou 510641, China*)

†E-mail: optmrchen@gmail.com

Received June 20, 2007; revision accepted Oct. 8, 2007

**Abstract:** In this paper, we extend a novel unconstrained multiobjective optimization algorithm, so-called multiobjective extremal optimization (MOEO), to solve the constrained multiobjective optimization problems (MOPs). The proposed approach is validated by three constrained benchmark problems and successfully applied to handling three multiobjective engineering design problems reported in literature. Simulation results indicate that the proposed approach is highly competitive with three state-of-the-art multiobjective evolutionary algorithms, i.e., NSGA-II, SPEA2 and PAES. Thus MOEO can be considered a good alternative to solve constrained multiobjective optimization problems.

**Key words:** Multiobjective optimization, Extremal optimization (EO), Engineering design
**doi:**10.1631/jzus.2007.A1905 **Document code:** A **CLC number:** TP18

## INTRODUCTION

Most engineering design problems are multiobjective in nature, since they normally have several (possible conflicting) objectives that must be satisfied simultaneously. Instead of finding a single solution, the multiobjective optimization methods try to produce a set of good tradeoffs from which the decision maker may select one. Since mid-1980s, a considerable amount of multiobjective evolutionary algorithms (MOEAs) have been presented to solve the multiobjective optimization problems (MOPs) (Coello Coello, 2006). Among them, the evolutionary algorithms (EAs) seem particularly suitable for solving the MOPs, mainly because of their ability to find multiple Pareto-optimal solutions in one single simulation run (Coello Coello, 2006). Additionally, EAs can easily deal with discontinuous or concave Pareto fronts that are a real concern when mathematical programming techniques are applied (Coello Coello, 2006).

In recent years, a general-purpose local-search heuristic algorithm named extremal optimization (EO) was proposed by Boettcher and Percus (2000). EO is based on the Bak-Sneppen (BS) model (Bak and Sneppen, 1993), which shows the emergence of self-organized criticality (SOC) (Bak *et al*., 1987) 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 (Bak and Sneppen, 1993). EO opens the door to applying non-equilibrium process, while the simulated annealing (SA) applies equilibrium statistical mechanics. In contrast to genetic algorithms (GAs) which operate on an entire "gene-pool" of huge number of possible solutions, EO successively eliminates the worst components in the sub-optimal solutions. Its large fluctuations provide

* Project (No. 60574063) supported by the National Natural Science Foundation of China

significant hill-climbing ability, which enables EO to perform well particularly at the phase transitions. EO has been successfully applied to many optimization problems such as graph bi-partitioning (Boettcher and Percus, 2000), production scheduling (Lu *et al*., 2007), function optimization (Chen *et al*., 2006) and dynamic combinatorial problems (Moser and Hendtlass, 2006).

Recently, we have developed a fast novel elitist multiobjective optimization algorithm, called "multiobjective extremal optimization" (MOEO) (Chen and Lu, 2007), through elegantly introducing the fitness assignment based on Pareto dominance strategy to EO for the first time. In our previous work (Chen and Lu, 2007), MOEO was successfully applied to solving a number of difficult unconstrained benchmark problems. Moreover, we proposed a new hybrid mutation operator, so-called hybrid *GC* mutation operator which combines the Gaussian mutation with the Cauchy mutation perfectly. In this study, we extend MOEO to deal with constrained MOPs. Our approach is validated using three constrained benchmark problems. Furthermore, the proposed approach is successfully applied to three multiobjective engineering design problems reported in the specialized literature. The simulation results demonstrate that the proposed approach is highly competitive with three state-of-the-art MOEAs, i.e., the nondominated sorting genetic algorithm-II (NSGA-II) (Deb *et al*., 2002), the strength Pareto evolutionary algorithm2 (SPEA2) (Zitzler *et al*., 2001) and the Pareto archived evolution strategy (PAES) (Knowles and Corne, 1999). As a result, MOEO may be a good alternative to solve constrained MOPs.

## EXTREMAL OPTIMIZATION

Inspired by the Bak-Sneppen model, Boettcher and Percus (2000) proposed the EO algorithm for a minimization problem as Fig.1.

Note that in the EO algorithm, each decision variable in the current solution $S$ is considered "species". In this study, we adopt the term "component" to represent "species", which is usually used in biology. From the above algorithm, it can be seen that unlike GAs which work with a population of candidate solutions, EO evolves a single solution $S$ and makes local modification to the worst components. This requires that a suitable representation be selected which permits the solution components to be assigned a quality measure (i.e. fitness). This differs from holistic approaches such as evolutionary algorithms that assign equal-fitness to all components of a solution based on their collective evaluation against an objective function. It is worth pointing out that the component, which provides the best result after mutation, will be regarded as the worst one. In this way, EO can drive a solution evolving towards the global optimal region by always mutating the worst components.

1. Randomly generate a solution $S=(x_1, x_2, \ldots, x_n)$. Set optimal solution $S_{best}=S$ and the minimum cost function $C(S_{best})=C(S)$.
2. For the current solution $S$,
   (1) evaluate the fitness $\lambda_i$ for each variable $x_i$, $i\in\{1, \ldots, n\}$;
   (2) rank all the fitnesses and find the variable $x_j$ with the lowest fitness, i.e., $\lambda_j\leq\lambda_i$ for all $i$;
   (3) choose one solution $S'$ in the neighborhood of $S$, so that the $j$th variable must change its state;
   (4) accept $S=S'$ unconditionally;
   (5) if $C(S)<C(S_{best})$, then set $S_{best}=S$ and $C(S_{best})=C(S)$.
3. Repeat Step 2 as long as desired.
4. Return $S_{best}$ and $C(S_{best})$.

**Fig.1 The pseudo-code of extremal optimization (EO) algorithm**

## MULTIOBJECTIVE EXTREMAL OPTIMIZATION

In order to further extend MOEO to solve the constrained MOPs, in this work, we present an extended MOEO algorithm, through introducing a constraint handling strategy. In the following sections, main algorithm, constraint handling, fitness assignment, diversity preservation, external archive and mutation operator will be addressed in some details.

### Main algorithm

For a constrained multiobjective optimization problem, the extended MOEO algorithm works as Fig.2.

### Constraint handling

In this work, we adopt the constrained-domination principle proposed by Deb *et al*.(2002) for solving the constrained MOPs. This principle can be incorporated into the dominance comparison between any two solutions. A solution $i$ is said to constrained-

dominate a solution $j$, if any of the following conditions is true (Deb *et al*., 2002):

(1) Solution $i$ is feasible while solution $j$ is not;

(2) Solutions $i$ and $j$ are both infeasible, but solution $i$ has a smaller overall constraint violation;

(3) Solutions $i$ and $j$ are feasible and solution $i$ dominates solution $j$.

1. Randomly generate an initial feasible solution $S=(x_1, x_2, \ldots, x_n)$ and add $S$ to the external archive. Set *iteration*=0.
2. Generate $n$ offspring of the current solution $S$ by performing mutation on each component one by one, while keeping other components unchanged.
3. Perform dominance ranking on the $n$ offspring and then obtain their rank numbers, i.e., $r_j \in [0, n-1], j \in \{1, \ldots, n\}$.
4. Assign the fitness $\lambda_j=r_j$ for each component $x_j, j \in \{1, \ldots, n\}$.
5. If there is only one component with fitness value of zero, the component will be considered the worst one. Otherwise, if there are two or more components with fitness value of zero, the diversity preservation mechanism is invoked to decide the worst component. Assuming that the worst component is $x_w$, and $S_w$ is the corresponding offspring generated by performing mutation only on $x_w$, $w \in \{1, \ldots, n\}$.
6. Accept $S=S_w$ unconditionally.
7. If $S$ is a feasible solution, then apply the archiving logic to updating the external archive. Otherwise, go to Step 8.
8. If the iterations reach the predefined maximum number of the generations, go to Step 9; otherwise, set *iteration*=*iteration*+1, and go to Step 2.
9. Output the external archive as the Pareto-optimal set.

**Fig.2 The pseudo-code of multiobjective extremal optimization (MOEO) algorithm**

## Fitness assignment

MOEO (Chen and Lu, 2007) uses the dominance ranking method proposed by Fonseca and Fleming (1993) to determine the fitness value for each solution. In order to extend MOEO to handle the constrained MOPs, the above constrained-domination principle should be incorporated into the dominance ranking method. That is, the fitness value of one solution equals the number of other solutions by which it is constrained-dominated. Therefore, the nondominated solutions are ranked as zero, whilst the worst possible ranking is the number of all solutions minus one.

It is important to note that there exists only one solution in the search process of MOEO. In order to find out the worst component via fitness assignment, we generate a population of new offspring by performing mutation on the components of current solution one by one. Then the dominance ranking is carried out in the newly generated offspring. In MOEO, each component of the current solution will be assigned a fitness value, which is defined as the dominance ranking number of its corresponding offspring. It is worth pointing out that the component corresponding to the nondominated offspring (i.e., its fitness value equals zero) is considered the weakest one, as the best result can be produced via changing this component. Then the nondominated offspring will replace the current solution in the next generation.

If there are two or more components with the same fitness value of zero, then the following diversity-preserving mechanism will be invoked to decide the worst component.

## Diversity preservation

The goal of introducing the diversity preservation mechanism is to maintain a good spread of solutions in the obtained set of solutions. Note that this diversity preservation mechanism is only invoked when more than one nondominated offspring has been generated. It should be pointed out that those offspring dominated by the current solution will increase the searching time to approach the Pareto-optimal set. As a result, if there exists more than one nondominated offspring, MOEO will first randomly pick one from those offspring which are not dominated by the current solution. The worst case is that all the nondominated offspring are dominated by the current solution, and then MOEO will randomly choose one from them as the new solution in the next generation. For more details, the interested readers are referred to (Chen and Lu, 2007).

## External archive

The main objective of the external archive is to keep a historical record of the nondominated solutions found along the search process. Note that the solutions added to the external archive should be feasible when dealing with the constrained MOPs. MOEO adopts a fixed-size archive consisting of two main components as follows.

(1) Archiving logic. The function of the archiving logic is to decide whether the current solution should be added to the archive or not. The archiving logic works as follows. The newly generated solution $S$ is compared with the archive to check if it dominates any member of the archive. If some members of the archive are dominated by $S$, all the dominated

solutions are eliminated from the archive and $S$ is added to the archive. If at least one member of the archive dominates $S$, the archive does not need to be updated and the iteration continues. However, if $S$ and any member of the archive do not dominate each other, there exist three cases:

(i) If the archive is not full, $S$ is added to the archive.

(ii) If the archive is full and $S$ resides in the most crowded region in the objective space among the members of the archive, the archive does not need to be updated.

(iii) If the archive is full and $S$ does not reside in the most crowded region of the archive, the member in the most crowded region of the archive is replaced by $S$.

(2) Crowding-distance metric. MOEO adopts the crowding-distance metric proposed by Deb *et al*. (2002) to judge whether the current solution resides in the most crowded region of the archive. If $S$ is the one with the smallest crowding distance, $S$ will be considered lying in the most crowded region.

**Hybrid *GC* mutation operator**

In the hybrid *GC* mutation (Chen and Lu, 2007), the Cauchy mutation is first adopted. It means that the large step size will be taken first at each mutation. If the new generated variable after mutation goes beyond the range of variable, the Cauchy mutation will be used repeatedly for *TC* times, until the new generated offspring falls into the range. Otherwise, Gaussian mutation will be carried out repeatedly for *TG* times, until the offspring satisfies the requirement. That is, the step size will become smaller than before. If the new generated variable after mutation still goes beyond the range of variable, then the upper or lower bound of the decision variable is chosen as the new generated variable. Thus, this hybrid mutation operator combines the advantages of coarse-grained search and fine-grained search (Chen and Lu, 2007).

In the hybrid *GC* mutation, the values of parameters *TC* and *TG* are set by the user beforehand. *TC* decides the coarse-grained searching time, while *TG* has an effect on the fine-grained searching time. Therefore, both values of the two parameters cannot be large because it will prolong the search process and hence increase the computational overhead. According to our experimental experience, the moderate values of *TC* and *TG* can be set to 2~4.

## TEST PROBLEMS AND EXPERIMENTAL RESULTS

**Test problems**

In this paper, the proposed approach was testified by three constrained benchmark functions, i.e., CONSTR, SRN and TNK, reported in (Deb *et al*., 2002), and three nonlinear mechanical component design problems, i.e., two bar truss design (Deb *et al*., 2000) ("Two Bar" for short), welded beam design (Deb *et al*., 2000) ("Welded Beam" for short) and machine tool spindle design (Coello Coello, 1996) ("Spindle" for short). All the problems have two objective functions.

**Performance measures**

In this article, we use the following two performance metrics to assess the performance of our approach.

(1) Front spread (*FS*) (Bosman and Thierens, 2003): It indicates the size of the objective space covered by an approximate set. A larger *FS* metric is preferable. The *FS* metric for an approximation set $S$ is defined to be the maximum Euclidean distance inside the smallest $m$-dimensional bounding-box that contains $S$. Here $m$ is the number of objectives. This distance can be computed using the maximum distance among the solutions in $S$ in each dimension separately.

$$FS(S)=\sqrt{\sum_{i=0}^{m-1}\max_{(z^0,z^1)\in S\times S}\left(f_i(z^0)-f_i(z^1)\right)^2}. \quad (1)$$

(2) Coverage of two sets (Zitzler and Thiele, 1999): Without loss of generality, assume a minimization problem and consider two decision vectors $\boldsymbol{a}$, $\boldsymbol{b}\in X$. Then $\boldsymbol{a}$ is said to cover $\boldsymbol{b}$ (written as $\boldsymbol{a}\preceq\boldsymbol{b}$) if $\boldsymbol{a}$ dominates $\boldsymbol{b}$ (also written as $\boldsymbol{a}\prec\boldsymbol{b}$) or $\boldsymbol{a}$ equals $\boldsymbol{b}$. Let $X'$, $X''\subseteq X$ be two sets of decision vectors. The function $C$ maps the ordered pair ($X'$, $X''$) to the interval [0, 1].

$$C(X',X'')=\frac{\left|\{\boldsymbol{a}''\in X'';\exists\boldsymbol{a}'\in X':\boldsymbol{a}'\preceq\boldsymbol{a}''\}\right|}{|X''|}, \quad (2)$$

$C(X', X'')=1$ means that all points in $X''$ are covered by points in $X'$. The opposite, $C(X', X'')=0$ represents the situation when none of points in $X''$ are covered by the

set $X'$. Note that both $C(X', X'')$ and $C(X'', X')$ have to be considered, since $C(X', X'')$ is not necessarily equal to $C(X'', X')$ [e.g., if $X'$ dominates $X''$, then $C(X', X'')=1$ and $C(X'', X')=0$].

The first metric *FS* can be used to measure the spread of an approximate front, while the second metric *C* can be used to show that the outcomes of one algorithm dominate those of another algorithm.

**Experimental settings**

In this study, the simulation results of our approach will be compared with those of NSGA-II, SPEA2 and PAES. To avoid any bias or misinterpretation when implementing each of the other three approaches, we adopted the public-domain versions of NSGA-II, PAES and SPEA2 (the former two are available at http://www.lania.mx/~ccoello/EMOO/EMOOsoftware.html and the last one available at http://www.tik.ee.ethz.ch/pisa/selectors/spea2/spea2_c_source.html). In addition, to conduct a fair comparison, all algorithms should be performed under the same fitness function evaluations (*FFE* for short). Note that for MOEO, *FFE*=maximum generation× problem dimension, for PAES, *FFE*=maximum generation, and for NSGA-II and SPEA2, *FFE*=maximum generation×population size. All the algorithms were run for a maximum of 50 000 *FFE* for problem TNK, 30 000 for Two Bar, 20 000 for SRN, and 40 000 for the rest three problems. For MOEO, it was encoded in the floating point representation and an archive of size 100 was used. The parameters in the hybrid *GC* mutation, i.e., *TC* and *TG*, were both set to 3. For NSGA-II (real-coded), the population size was 100. For SPEA2, we used a population of size 80 and an external population of size 20, so that the overall population size became 100. For PAES, we adopted a depth value $d$ equal to 4 and the archive size of 100. The crossover probability of $p_c$=0.9 and a mutation probability of $p_m$=0.05 for real-coded GAs or $p_m$=1/$l$ for binary-coded GAs (where $l$ is the string length) were used. Note that all the 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. Additionally, all the algorithms were executed 50 times independently on each problem.

**Experimental results and discussion**

Table 1 shows the mean and standard deviations of the *FS* metric obtained using four algorithms, i.e., MOEO, NSGA-II, SPEA2 and PAES. As can be seen from Table 1, MOEO is capable of finding a wider spread of solutions than any other algorithms on the problems CONSTR, TNK, Welded Beam and Spindle. This indicates that our approach can find a wider distributed set of nondominated solutions than many other state-of-the-art MOEAs. In all cases with MOEO, the standard deviation of *FS* metric in 50 runs is also small except for the problem Two Bar.

The direct comparison of the different algorithms based on the *C* metric is shown in Table 2. It can be observed from Table 2 that the MOEO can perform better than NSGA-II on the problems SRN and Spindle concerning the *C* metric. It is interesting to note that MOEO outperforms SPEA2 on all the problems in terms of the *C* metric. It is also clear from Table 2 that MOEO significantly outperforms PAES on four problems, i.e., SRN, TNK, Two Bar and Spindle, with respect to the *C* metric.

For illustration, we also show one of the 50 runs of all the algorithms on each problem in Figs.3a~3f. The figures show the tradeoff fronts obtained by MOEO, NSGA-II, SPEA2 and PAES. The insets in all the figures show the parts which may be unclear in the main plots. From Figs.3a~3f, we can see that MOEO can converge nearly as well as or better than the other three algorithms on all the problems. Moreover, MOEO can find a well-distributed set of nondominated solutions on all the problems.

**Table 1 Comparisons of *FS* metric**

| Algorithm | CONSTR | | SRN | | TNK | | Two Bar | | Welded Beam | | Spindle | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Mean | SD | Mean | SD | Mean | SD | Mean | SD | Mean | SD | Mean | SD |
| MOEO | **7.87** | 0.12 | 343.98 | 2.33 | **9.43** | 0.08 | 8.47E4 | 4.2E3 | **37.45** | 3.22 | **1.143E6** | 1.2E4 |
| NSGA-II | 7.36 | 0.69 | 343.22 | 3.29 | 9.37 | 0.18 | **9.15E4** | 1.0E2 | 33.62 | 1.89 | 1.129E6 | 2.6E4 |
| SPEA2 | 7.74 | 0.27 | **344.47** | 2.63 | 9.42 | 0.13 | **9.15E4** | 8.4E1 | 33.81 | 1.52 | 1.136E6 | 2.5E4 |
| PAES | 2.19 | 2.05 | 336.03 | 49.29 | 5.01 | 3.04 | 8.80E4 | 3.7E3 | 31.35 | 4.90 | 1.073E6 | 7.7E4 |

SD=standard deviation

**Table 2 Comparisons of *C* metric**

| Coverage of two sets | CONSTR | | SRN | | TNK | | Two Bar | | Welded Beam | | Spindle | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Mean | SD | Mean | SD | Mean | SD | Mean | SD | Mean | SD | Mean | SD |
| *C*(NSGA-II,MOEO) | 0.30 | 0.04 | 0.06 | 0.02 | 0.35 | 0.05 | 0.38 | 0.04 | 0.52 | 0.11 | 0.01 | 0.01 |
| *C*(MOEO,NSGA-II) | 0.09 | 0.03 | 0.09 | 0.03 | 0.16 | 0.04 | 0.15 | 0.04 | 0.09 | 0.09 | 0.03 | 0.03 |
| *C*(SPEA2,MOEO) | 0.03 | 0.02 | 0.01 | 0.01 | 0.06 | 0.02 | 0.06 | 0.03 | 0.09 | 0.04 | 0.004 | 0.01 |
| *C*(MOEO,SPEA2) | 0.39 | 0.08 | 0.46 | 0.13 | 0.38 | 0.10 | 0.58 | 0.10 | 0.32 | 0.17 | 0.29 | 0.10 |
| *C*(PAES,MOEO) | 0.25 | 0.03 | 0.03 | 0.02 | 0.01 | 0.01 | 0.15 | 0.07 | 0.38 | 0.18 | 0.002 | 0.005 |
| *C*(MOEO,PAES) | 0.01 | 0.02 | 0.26 | 0.20 | 0.45 | 0.24 | 0.53 | 0.15 | 0.23 | 0.27 | 0.62 | 0.26 |

SD=standard deviation

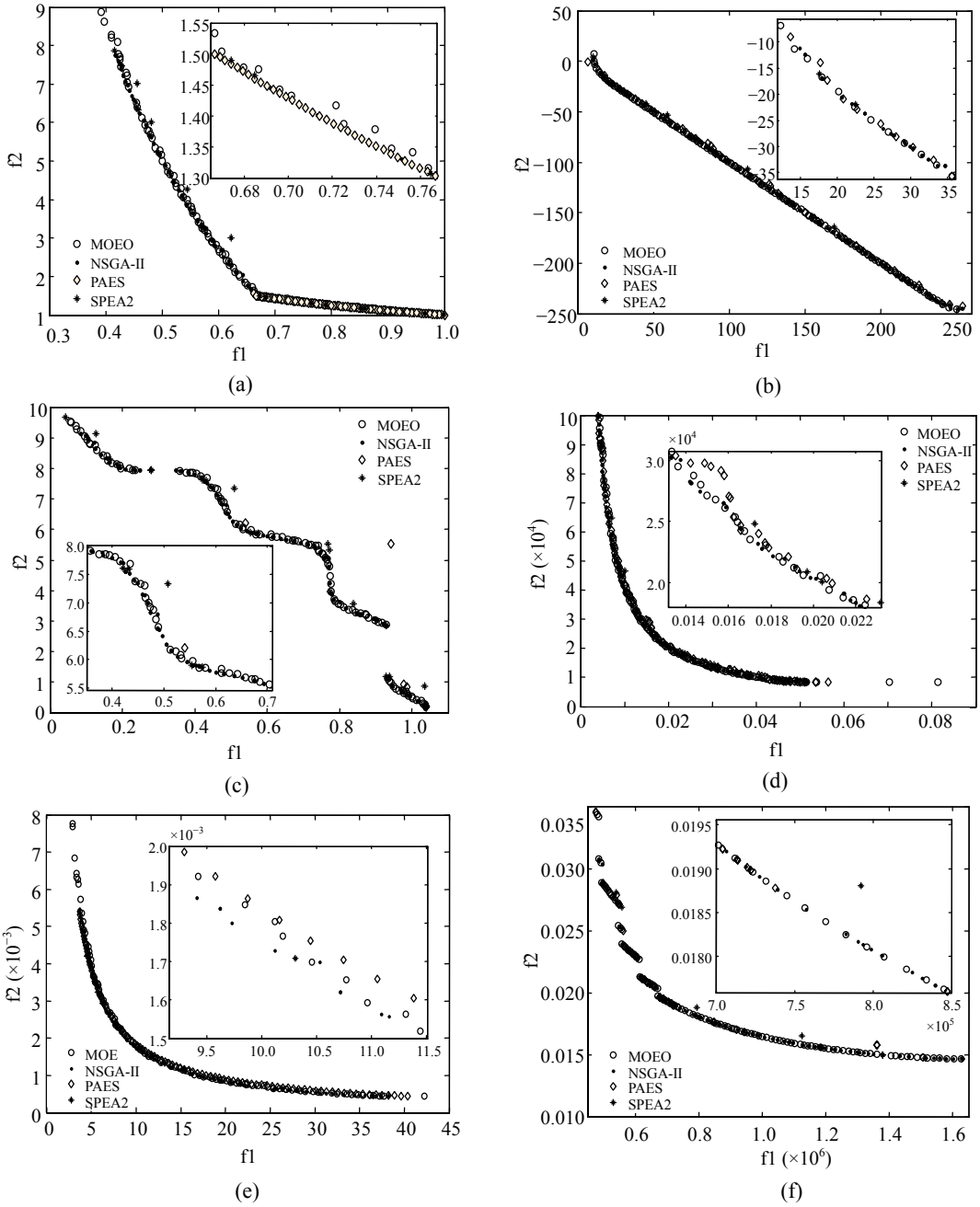


**Fig.7 Tradeoff fronts. (a) CONSTR; (b) SRN; (c) TNK; (d) Two Bar; (e) Welded Beam; (f) Spindle**

It is interesting to note that MOEO can perform well in both aspects of convergence and diversity of solutions on the problem TNK. The problem TNK is slightly harder, since the Pareto front is discontinuous and, as shown by Deb *et al*.(2002), some algorithms have difficulty in finding the entire central continuous region. Fig.3c shows the tradeoff fronts obtained by all the algorithms on this problem. It is obvious that MOEO performs significantly better than SPEA2 and PAES in terms of the convergence and diversity of solutions. Note that MOEO and NSGA-II nearly converge to the same tradeoff front. However, our approach can find a much more diverse set of nondominated solutions than NSGA-II.

Note that the problem Spindle has two continuous and two discrete decision variables. As can be observed from Table 1, Table 2 and Fig.3f, MOEO seems to be the best performer in both aspects of convergence and diversity of solutions on this problem. Two extreme nondominated solutions found by MOEO are ($4.767\times10^5$, 0.0346) and ($1.607\times10^6$, 0.014 65), respectively, which are better than those found by other techniques listed in (Baykasoglu, 2006). Consequently, MOEO may be a good choice to solve those problems with mixed continuous or discrete decision variables.

## CONCLUSION AND FUTURE WORK

In this paper, we extend the MOEO algorithm to solve the constrained MOPs. The proposed approach is validated by three constrained benchmark problems and three engineering design problems. Simulation results indicate that MOEO is highly competitive with three state-of-the-art MOEAs, i.e., NSGA-II, SPEA2 and PAES in terms of convergence and diversity of solutions. The results of this study are encouraging. Thus, MOEO may be a good alternative to deal with constrained MOPs and be well-suited for handling those engineering design problems with multiple objectives. The future work includes the studies on how to extend MOEO to handle those MOPs with more than two objectives. It is desirable to further apply MOEO to solving those more complex engineering optimization problems in the real world.

## References


Bak, P., Tang, C., Wiesenfeld, K., 1987. Self-organized criticality. *Phys. Rev. Lett.*, **59**:381-384. [doi:10.1103/PhysRevLett.59.381]

Bak, P., Sneppen, K., 1993. Punctuated equilibrium and criticality in a simple model of evolution. *Phys. Rev. Lett.*, **71**(24):4083-4086. [doi:10.1103/PhysRevLett.71.4083]

Baykasoglu, A., 2006. Applying multiple objective tabu search to continuous optimization problems with a simple neighborhood strategy. *Int. J. Numer. Methods Eng.*, **65**:406-424. [doi:10.1002/nme.1455]

Boettcher, S., Percus, A.G., 2000. Nature's way of optimizing. *Artif. Intell.*, **119**:275-286. [doi:10.1016/S0004-3702(00)00007-2]

Bosman, P.A.N., Thierens, D., 2003. The balance between proximity and diversity in multiobjective evolutionary algorithms. *IEEE Trans. on Evol. Comput.*, **7**(2):174-188. [doi:10.1109/TEVC.2003.810761]

Chen, M.R., Lu, Y.Z., Yang, G., 2006. Population-based Extremal Optimization with Adaptive Lévy Mutation for Constrained Optimization. Proc. Int. Conf. on Computational Intelligence and Security, p.258-261. [doi:10.1109/ICCIAS.2006.294132]

Chen, M.R., Lu, Y.Z., 2007. A novel elitist multiobjective optimization algorithm: multiobjective extremal optimization. *Eur. J. Operat. Res.*, in press. [doi:10.1016/j.ejor.2007.05.008]

Coello Coello, C.A., 1996. An Empirical Study of Evolutionary Techniques for Multiobjective Optimization in Engineering Design. Ph.D Thesis, Department of Computer Science, Tulane University, New Orleans, LA.

Coello Coello, C.A., 2006. Evolutionary multiobjective optimization: a historical view of the field. *IEEE Comput. Intell. Mag.*, **1**(1):28-36. [doi:10.1109/MCI.2006.1597059]

Deb, K., Patrap, A., Moitra, S., 2000. Mechanical Component Design for Multi-objective Using Elitist Non-dominated Sorting GA. KanGAL Report No. 200002. Indian Institute of Technology Kanpur, India.

Deb, K., Pratab, A., Agrawal, S., Meyarivan, T., 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. on Evol. Comput.*, **6**(2):182-197. [doi:10.1109/4235.996017]

Fonseca, C.M., Fleming, P.J., 1993. Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization. Proc. 5th Int. Conf. on Genetic Algorithms. Morgan Kauffman, San Mateo, CA, p.416-423.

Knowles, J., Corne, D., 1999. The Pareto Archived Evolution Strategy: A New Baseline Algorithm for Multiobjective Optimization. Proc. Congress on Evolutionary Computation, p.98-105.

Lu, Y.Z., Chen, M.R., Chen, Y.W., 2007. Studies on Extremal Optimization and its Applications in Solving Real World Optimization Problems. Proc. IEEE Symp. on Foundations of Computational Intelligence, p.162-168. [doi:10.1109/FOCI.2007.372163]

Moser, I., Hendtlass, T., 2006. Solving Problems with Hidden Dynamics-Comparison of Extremal Optimization and Ant Colony System. Proc. IEEE Conf. on Evolutionary Computation, p.1248-1255. [doi:10.1109/CEC.2006.1688452]

Zitzler, E., Thiele, L., 1999. Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. *IEEE Trans. on Evol. Comput.*, **3**(4):257-271. [doi:10.1109/4235.797969]

Zitzler, E., Laumanns, M., Thiele, L., 2001. SPEA2: Improving the Performance of the Strength Pareto Evolutionary Algorithm. Technical Report 103, Computer Engineering and Communication Networks Lab (TIK), Swiss Federal Institute of Technology (ETH) Zurich.