Engineering Optimization
Vol. 39, No. 6, September 2007, 715–736

An orthogonal design based constrained evolutionary optimization algorithm

YONG WANG*†, HUI LIU†, ZIXING CAI† and YUREN ZHOU‡

†School of Information Science and Engineering, Central South University, Changsha 410083, P.R. China
‡School of Computer Science and Engineering, South China University of Technology, Guangzhou 516040, P.R. China

(Received 8 March 2006; in final form 22 January 2007)

Solving constrained optimization problems (COPs) via evolutionary algorithms (EAs) has attracted much attention. In this article, an orthogonal design based constrained optimization evolutionary algorithm (ODCOEA) to tackle COPs is proposed. In principle, ODCOEA belongs to a class of steady state evolutionary algorithms. In the evolutionary process, several individuals are chosen from the population as parents and orthogonal design is applied to pairs of parents to produce a set of representative offspring. Then, after combining the offspring generated by different pairs of parents, non-dominated individuals are chosen. Subsequently, from the parent's perspective, it is decided whether a non-dominated individual replaces a selected parent. Finally, ODCOEA incorporates an improved BGA mutation operator to facilitate the diversity of the population. The proposed ODCOEA is effectively applied to 12 benchmark test functions. The computational experiments show that ODCOEA not only quickly converges to optimal or near-optimal solutions, but also displays a very high performance compared with another two state-of-the-art techniques.

Keywords: Constrained optimization; Orthogonal design; Multi-objective optimization; Non-dominated individuals

1. Introduction

Constrained optimization problems (COPs) belong to a kind of mathematical programming problem, which is frequently encountered in the disciplines of science and engineering application. Without loss of generality, the general non-linear programming problem has the following form (in minimization sense):

$$\text{Find } \vec{x}(\vec{x} = (x_1, x_2, \ldots, x_n) \in \Re^n) \text{ which optimizes } f(\vec{x})$$

where $\vec{x} \in \Omega \subseteq S$, and S is an n-dimensional rectangle space in $\Re^n$ defined by the parametric constraints:

$$l_i \leq x_i \leq u_i, \quad 1 \leq i \leq n.$$

*Corresponding author. Email: ywang@csu.edu.cn

Engineering Optimization
ISSN 0305-215X print/ISSN 1029-0273 online © 2007 Taylor & Francis
http://www.tandf.co.uk/journals
DOI: 10.1080/03052150701280541

The feasible region $\Omega \subseteq S$ is defined by a set of m additional linear or non-linear constraints ($m \geq 0$):

$$g_j(\vec{x}) \leq 0, \quad j = 1, \ldots, q \quad \text{and} \quad h_j(\vec{x}) = 0, \quad j = q+1, \ldots, m$$

where q is the number of inequality constraints and $m - q$ is the number of equality constraints. If an inequality constraint that satisfies $g_j(\vec{x}) = 0$ ($j \in \{1, \ldots, q\}$) at any point $\vec{x} \in \Omega$, it is considered *active* at $\vec{x}$. All equality constraints $h_j(\vec{x})$ ($j = q+1, \ldots, m$) are considered *active* at all points of Ω.

Over the past decade, evolutionary algorithms (EAs) have been successfully used to solve COPs, and researchers have proposed a large number of constrained optimization evolutionary algorithms (COEAs) (Michalewicz and Schoenauer 1996, Michalewicz *et al.* 2000, Coello 2002). In essence, COEAs can be generalized as constraint-handling techniques plus EAs, *i.e.* an effective constraint-handling technique needs to be considered in conjunction with an efficient EA.

Much previous work has been done on constraint-handling techniques, which can be classified into the following categories according to the way the constraints are treated.

1. **Methods based on penalty functions** are among the most common approaches to solve COPs. The principal idea of this type of method is to reformulate a COP as an unconstrained case by introducing a penalty term into the original objective function to penalize constraint violations. In general, a penalty term is based on the degree of constraint violation of an individual. Let

$$G_j(\vec{x}) = \begin{cases} \max\{0, g_j(\vec{x})\}, & 1 \leq j \leq q \\ \max\{0, |h_j(\vec{x})| - \delta\}, & q+1 \leq j \leq m \end{cases} \tag{1}$$

   where δ is a positive tolerance value for equality constraints. Then

$$G(\vec{x}) = \sum_{j=1}^{m} G_j(\vec{x}) \tag{2}$$

   reflects the degree of constraint violation of the individual $\vec{x}$. However, Yu *et al.* (2003) pointed out that even the most dynamic setting methods, which start with a low parameter value and end up with a high value, are unlikely to work well for problems where the unconstrained global optimum is far away from the constrained global optimum.
2. **Methods based on biasing feasible over infeasible solutions**: Runarsson and Yao (2000) introduced a stochastic ranking based method to balance the objective and penalty functions. A probability parameter p_f is used to compare individuals as follows: given pairs of adjacent individuals, (i) if both individuals are feasible, the one with the better objective function value wins, else (ii) if a uniformly generated random number u between 0 and 1 is less than p_f, the one with the better objective function value wins, otherwise the one with a smaller degree of constraint violation wins. This approach significantly improves the search performance without any special constrained operator. Deb (2000) proposed a simple method that requires no penalty parameter. This method uses a tournament selection operator, where pairs of solutions are compared using the following criteria: (i) any feasible solution is preferred to any infeasible solution; (ii) between two feasible solutions, the one with the better objective function value is preferred; (iii) between two infeasible solutions, the one with the smaller degree of constraint violation is preferred. Based on this method,

Mezura-Montes and Coello (2005) introduced a simple diversity mechanism to add some infeasible solutions into the next population.

3. **Methods based on multi-objective optimization techniques**: taking advantage of multi-objective optimization techniques to handle constraints is a new idea that has emerged in recent years. In this case, constraints can be regarded as one or more objectives. If constraints are treated as one objective, then the original problem will be transformed into a multi-objective optimization problem with two objectives. In general, one is the original objective function $f(\vec{x})$, and the other is the degree of constraint violation of an individual $\vec{x}$, *i.e.* $G(\vec{x})$. Alternatively, each constraint can be treated as an objective and, in this case, the original problem will be transformed into a multi-objective optimization problem which has $m + 1$ objectives. Thus a new vector $\vec{v} = (f(\vec{x}), f_1(\vec{x}), \ldots, f_m(\vec{x}))$ should be optimized, where $f_1(\vec{x}), \ldots, f_m(\vec{x})$ are the constraints of the given problem. Several typical paradigms are reviewed below.

By combining the vector evaluated genetic algorithm (GA) (Schaffer 1985) with Pareto ranking, Surry and Radcliffe (1997) proposed a method called COMOGA. COMOGA uses a single population but randomly decides whether to consider the original problem as a constraint satisfaction problem or as an unconstrained optimization problem. Also, the relative likelihood of adopting each view is adjusted using a simple adaptive mechanism that tries to set a target proportion (*e.g.* 0.1) of feasible solutions in the population. Its main advantage is that it does not require fine tuning of penalty factor.

Venkatraman and Yen (2005) presented a two-phase framework for solving COPs. In the first phase, COP is treated as a constrained satisfaction problem, and the genetic search is directed toward minimizing the constraint violations of the solutions. In the second phase, COP is treated as a bi-objective optimization problem, and non-dominated sorting, as described by Deb *et al.* (2002), is adopted to rank the individuals. This algorithm has the advantage of being problem independent and does not rely on parameter tuning.

Zhou *et al.* (2003) proposed a novel method which uses Pareto dominance to assign an individual's Pareto strength.

DEFINITION 1 (Pareto dominance) *A vector* $\vec{u} = (u_1, \ldots, u_k)$ *is said to Pareto dominate another vector* $\vec{v} = (v_1, \ldots, v_k)$, *denoted as* $\vec{u} \prec \vec{v}$, *if*

$$\forall\, i \in \{1, \ldots k\},\ \ u_i \leq v_i \quad \text{and} \quad \exists\, j \in \{1, \ldots, k\},\ \ u_j < v_j.$$

The Pareto strength of an individual means the number of individuals in the population dominated by it. Based on Pareto strength, the ranking of individuals in the population is conducted in such a manner that, upon comparing each individual's strength: (i) the one with high strength wins; and (ii) if the strength is equal, the one with lower degree of constraint violation is better.

Other multi-objective optimization techniques have been developed to handle COPs. Some examples include a method (Coello 2000a) based on population-based multi-objective techniques such as the vector evaluated GA (Schaffer 1985), a method (Coello 2000b) based on the Pareto ranking process of Fonseca and Fleming (1999), a method (Coello and Mezura-Montes 2002) based on the niched-Pareto GA (Horn *et al.* 1994), and a method (Aguirre *et al.* 2004) based on the Pareto archived evolutionary strategy (Knowles and Corne 2000).

As previously pointed out, the search algorithm is another important aspect of COEAs. Recently, as an efficient search technique, orthogonal design (Wu 1978) has been combined with EAs for different kinds of optimization problems (Zhang and Leung 1999, Ho and Chen 2001, Leung and Wang 2001, Ho *et al.* 2004, Tsai *et al.* 2004, Zeng *et al.* 2004, Zhong *et al.* 2004). Observing that some major steps of a GA (such as the crossover operator) can

be considered to be 'experiments', Zhang and Leung (1999) proposed incorporating experimental design methods into the GA, so that the resulting algorithm would be more robust and statistically sound. Leung and Wang (2001) designed a GA called the orthogonal GA with quantization for global numerical optimization with continuous variables. They developed a quantization technique to complement the orthogonal design, so that the resulting methodology would enhance GAs for optimization with continuous variables. Ho and Chen (2001) proposed an efficient EA with a novel orthogonal array crossover for obtaining the optimal solution to the polygonal approximation problem. Tsai *et al.* (2004) presented a hybrid Taguchi–genetic algorithm for global numerical optimization, where the Taguchi method involving a two-level orthogonal array and a signal-to-noise-ratio is inserted between the crossover and mutation operations of a traditional GA. More recently, orthogonal design has been generalized to deal with multi-objective optimization problems (Ho *et al.* 2004, Zeng *et al.* 2004).

An orthogonal design based constrained optimization evolutionary algorithm (ODCOEA), which belongs to category 3 of the constraint-handling techniques, is proposed in this article. To the authors' knowledge, ODCOEA is the first attempt to integrate orthogonal design and multi-objective optimization into EAs for solving COPs. In principle, ODCOEA is a kind of steady state evolutionary algorithm. That is, at each generation some individuals are chosen from the population as parents. Then orthogonal design is applied to pairs of parents to produce a set of representative offspring. After combining the offspring created by different sets of parents, non-dominated individuals are chosen as the new potential offspring. Subsequently, from the parent's perspective, it is decided whether a non-dominated individual replaces a chosen parent; this behaviour makes the replacement operation more reasonable and effective. Finally, an improved BGA mutation operator is devised to enhance the diversity of the population. The empirical results suggest that ODCOEA achieves a good performance on 12 benchmark test functions. The primary benefits of ODCOEA are efficiency and simplicity.

The remainder of the article is organized as follows. The orthogonal design adopted is described in section 2. The proposed ODCOEA is introduced in detail in section 3. In section 4, the experimental study of 12 benchmark problems of constrained optimization is presented, and ODCOEA is also compared with two state-of-the-art approaches. Finally, section 5 concludes this article.

2. Orthogonal design

A complete factorial experiment would require all possible combinations of levels, which would be so large that another, more practical, approach is proposed using orthogonal design (OD). OD is a fractional factorial experiment with the capability of sampling a small, but representative, set of level combinations. The primary purpose of OD is to utilize the properties of the fractional factorial experiment for the efficient determination of the best combination of levels.

OD provides a series of orthogonal arrays, each of which is a fractional factorial matrix and ensures a balanced combination of levels for any factor. Let $L_M(Q^N)$ be an orthogonal array of N factors and Q levels, where L denotes a Latin square, Q^N is the full size of the orthogonal array, and M is the number of combinations of levels. $L_M(Q^N)$ is a matrix of numbers arranged in M rows and N columns, where each row represents a combination of levels, and each column represents a specific factor that can be changed. The array is called orthogonal because all columns can be evaluated independently of one another. For convenience, let $L_M(Q^N) = [a_{i,j}]_{M\times N}$, where the jth factor in the ith combination has level $a_{i,j}$ and $a_{i,j} \in \{1, 2, \ldots, Q\}$.

For example, an orthogonal array $L_9(3^4)$ is depicted as follows:

$$L_9(3^4) = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 2 & 2 \\ 1 & 3 & 3 & 3 \\ 2 & 1 & 2 & 3 \\ 2 & 2 & 3 & 1 \\ 2 & 3 & 1 & 2 \\ 3 & 1 & 3 & 2 \\ 3 & 2 & 1 & 3 \\ 3 & 3 & 2 & 1 \end{bmatrix} \tag{3}$$

In $L_9(3^4)$, there are four factors, three levels per factor, and nine combinations of levels. In the first combination, the four factors have levels 1, 1, 1, 1; in the second combination, the four factors have levels 1, 2, 2, 2, etc. In this example, there are 81 combinations to be tested. However, the orthogonal array $L_9(3^4)$ is applied to select nine representative combinations to be tested.

Although many orthogonal arrays have been tabulated in the literature, it is impossible to store all of them for the proposed algorithm. Therefore a special class of orthogonal arrays $L_M(Q^N)$ is introduced, where Q is prime and $M = Q^J$, where J is a positive integer satisfying

$$N = \frac{Q^J - 1}{Q - 1}. \tag{4}$$

In the following, a simple permutation method is designed to construct orthogonal arrays of this class. Let $\vec{a}_j$ denote the jth column of the orthogonal array $[a_{i,j}]_{M\times N}$. Columns $\vec{a}_j$ for

$$j = 1, 2, \frac{Q^2 - 1}{Q - 1} + 1, \frac{Q^3 - 1}{Q - 1} + 1, \ldots, \frac{Q^{J-1} - 1}{Q - 1} + 1$$

are called the basic columns, and the others are called the non-basic columns. The algorithm first constructs the basic columns, and then constructs the non-basic columns. The details are shown in table 1.

Next, an approach showing how to design orthogonal crossover is introduced on the basis of orthogonal array and quantization technique. Suppose that the search space defined by any two parents $\vec{p}_1 = (p_{1,1}, p_{1,2}, \ldots, p_{1,n})$ and $\vec{p}_2 = (p_{2,1}, p_{2,2}, \ldots, p_{2,n})$ is $[\vec{l}_{parent}, \vec{u}_{parent}]$ where

$$\begin{cases} \vec{l}_{parent} = [\min(p_{1,1}, p_{2,1}), \min(p_{1,2}, p_{2,2}), \ldots, \min(p_{1,n}, p_{2,n})] \\ \vec{u}_{parent} = [\max(p_{1,1}, p_{2,1}), \max(p_{1,2}, p_{2,2}), \ldots, \max(p_{1,n}, p_{2,n})] \end{cases} \tag{5}$$

Each domain of $[\vec{l}_{parent}, \vec{u}_{parent}]$ is quantized into Q_1 levels such that the difference between any two successive levels is the same. For example, the domain of the ith dimension is quantized into $\beta_{i,1}, \beta_{i,2}, \ldots, \beta_{i,Q_1}$ where

$$\beta_{i,j} = \begin{cases} \min(p_{1,i}, p_{2,i}), & j = 1 \\ \min(p_{1,i}, p_{2,i}) + (j-1)\left(\dfrac{|p_{1,i} - p_{2,i}|}{Q_1 - 1}\right), & 2 \le j \le Q_1 - 1 \\ \max(p_{1,i}, p_{2,i}), & j = Q_1 \end{cases} \cdot \tag{6}$$

Table 1. Pseudocode for constructing the orthogonal array.

Step 1 Construct the basic columns as follows:
for k =1 to J **do**
$j = \dfrac{Q^{k-1}-1}{Q-1} + 1;$
for $i = 1$ to Q^J **do**
$a_{i,j} = \left\lfloor \dfrac{i-1}{Q^{J-k}} \right\rfloor \bmod Q;$
end for
end for
Step 2 construct the non-basic columns as follows:
for $k = 2$ to J **do**
$j = \dfrac{Q^{k-1}-1}{Q-1} + 1;$
for $s = 1$ to $j-1$ **do**
for $t = 1$ to $Q-1$ **do**
$\vec{a}_{j+(s-1)(Q-1)+t} = (\vec{a}_s \times t + \vec{a}_j) \bmod Q$
end for
end for
end for
Step 3 Increment $a_{i,j}$ by 1 for all $1 \le i \le M$ and $1 \le j \le N$

Then $F-1$ integers $k_1, k_2, \ldots, k_{F-1}$ are randomly generated such that $1 < k_1 < k_2 < \cdots < k_{F-1} < n$, and the following F factors are created for any chromosome $\vec{x} = (x_1, x_2, \ldots, x_n)$:

$$\begin{cases} \vec{f}_1 = (x_1, \ldots, x_{k_1}) \\ \vec{f}_2 = (x_{k_1+1}, \ldots, x_{k_2}) \\ \cdots \\ \vec{f}_F = (x_{k_{F-1}+1}, \ldots, x_n) \end{cases} \tag{7}$$

Thereafter, Q_1 levels for the ith factor $\vec{f}_i$ are defined as follows:

$$\begin{cases} \vec{f}_i(1) = (\beta_{k_{i-1}+1,1}, \beta_{k_{i-1}+2,1}, \ldots, \beta_{k_i,1}) \\ \vec{f}_i(2) = (\beta_{k_{i-1}+1,2}, \beta_{k_{i-1}+2,2}, \ldots, \beta_{k_i,2}) \\ \cdots \\ \vec{f}_i(Q_1) = (\beta_{k_{i-1}+1,Q_1}, \beta_{k_{i-1}+2,Q_1}, \ldots, \beta_{k_i,Q_1}) \end{cases} \tag{8}$$

The orthogonal array $L_{M_1}(Q_1^F) = [b_{i,j}]_{M_1 \times F}$ is constructed as in table 2 and used to generate the following M_1 individuals out of Q_1^F possible individuals:

$$\begin{cases} (\vec{f}_1(b_{1,1}), (\vec{f}_2(b_{1,2}), \ldots, (\vec{f}_F(b_{1,F}) \\ (\vec{f}_1(b_{2,1}), (\vec{f}_2(b_{2,2}), \ldots, (\vec{f}_F(b_{2,F}) \\ \cdots \\ (\vec{f}_1(b_{M_1,1}), (\vec{f}_2(b_{M_1,2}), \ldots, (\vec{f}_F(b_{M_1,F}) \end{cases} \tag{9}$$

In this work, the parameters Q_1 and F are chosen to be 3 and 4, respectively. Consequently, parameter J_1 has a value of 2. Then each crossover operator applies the orthogonal array $L_9(3^4)$ to produce nine candidate solutions. The number of individuals generated is reasonable with the parameter values assigned above. For more details about OD, please see Zhang and Leung (1999).

Table 2. Pseudocode for constructing the orthogonal array $L_{M_1}(Q_1^F)$.

Step 1 Select the smallest J_1 fulfilling $(Q_1^{J_1}-1)/(Q_1-1) \geq F$
Step 2 If $(Q_1^{J_1}-1)/(Q_1-1) = F$, then $F' = F$ else $F' = (Q_1^{J_1}-1)/(Q_1-1)$
Step 3 Execute algorithm shown in table 1 to construct the orthogonal array $L_{Q_1^{J_1}}(Q_1^{F'})$
Step 4 Delete the last $F'-F$ columns of $L_{Q_1^{J_1}}(Q_1^{F'})$ to obtain $L_{M_1}(Q_1^F)$ where $M_1 = Q_1^{J_1}$

3. Orthogonal design based constrained optimization evolutionary algorithm (ODCOEA)

It is well known that OD is an efficient approach to solving global optimization problems. It has been proved to be optimal for additive and quadratic models, and the selected combinations are good representatives for all the possible combinations. Herein, solution of COPs using OD that is incorporated with multi-objective optimization techniques is described. It is important to note that, in this article, COPs are recast as bi-objective optimization problems to minimize the original objective function $f(\vec{x})$ and the degree of constraint violation $G(\vec{x})$ simultaneously. For clarity, let $\mathbf{f}(\vec{x}) = (f(\vec{x}), G(\vec{x}))$. The details of the proposed ODCOEA will be described in the sequel.

3.1 *Generation of the initial population*

Since there is no prior knowledge of the location of the global optimum, it is desirable that the individuals in the initial population are scattered uniformly over the search space so that the algorithm can explore it evenly. To do this, an approach for generating the initial population is introduced (Tsai *et al.* 2004).

Let $\vec{x}_i = (x_1, x_2, \ldots, x_n)$ $(i \in \{1, 2, \ldots, N\})$, denote an individual in the initial population P of size N, where n is the number of decision variables. Let $\vec{x}_i = \vec{l} + (\vec{u} - \vec{l}) \otimes \vec{\beta}$ $(i \in \{1, 2, \ldots, N\})$, where $\vec{u} = (u_1, \ldots, u_n)$ and $\vec{l} = (l_1, \ldots, l_n)$ are the upper and lower bounds of the decision variables, respectively, $\otimes$ denotes point-wise vector multiplication, and $\vec{\beta} = (\beta_1, \ldots, \beta_n)$, where a random value β_i $(i \in \{1, 2, \ldots, N\})$, is selected from $\{0, 0.1, 0.2, \ldots, 1\}$. This process is repeated N times, so that an initial population with high diversity is obtained.

3.2 *Generation of potential offspring*

The next step is to produce potential offspring. Since ODCOEA makes use of multi-objective optimization techniques to handle constraints, the next three related definitions regarding multi-objective optimization are given in the context of the approach proposed in this article.

DEFINITION 2 (Pareto optimality) *$\vec{x}_u \in S$ is said to be Pareto optimal (in S) if and only if $\neg\, \exists\, \vec{x}_v \in S$, $\vec{v} \prec \vec{u}$, where $\vec{v} = \mathbf{f}(\vec{x}_v) = (v_1, v_2)$, $\vec{u} = \mathbf{f}(\vec{x}_u) = (u_1, u_2)$.*

DEFINITION 3 (Pareto optimal set) *The Pareto optimal set, denoted ρ^*, is defined as follows*:

$$\rho^* = \{\vec{x}_u \in S | \neg\, \exists\, \vec{x}_v \in S, \vec{v} \prec \vec{u}\}. \tag{10}$$

The vectors included in the Pareto optimal set are called non-dominated individuals.

DEFINITION 4 (Pareto front) *According to the Pareto optimal set, the Pareto front, denoted* ρf^*, *is defined as follows*:

$$\rho f^* = \{\vec{u} = \mathbf{f}(\vec{x}_u) | \vec{x}_u \in \rho^*\}. \tag{11}$$

Clearly, the Pareto front is the image of the Pareto optimal set in objective space.

In traditional GAs, two randomly selected parents are used to create offspring using certain recombination operators. In this work, nine offspring are obtained by performing OD on pairs of parents as described in section 2. Let these nine offspring constitute the offspring population. This work is only concerned with the non-dominated individuals, since they represent the most important feature of the population to which they belong (Cai and Wang 2006). Thus it is not necessary to assign a rank value to each individual in the population, as in multiobjective evolutionary algorithms (MOEAs), and hence the proposed algorithm is more efficient. The characteristics of non-dominated individuals can be summarized as follows.

THEOREM 1 *There exists at most one feasible solution in non-dominated individuals in a population.*

Proof Assume that there are $k > 1$ feasible solutions in non-dominated individuals in a population. Then the feasible solution with the minimum $f(\vec{x})$ in these k individuals must dominate other feasible solutions in these k individuals based on Pareto dominance. This is a contradiction. ■

PROPERTY 1 *Non-dominated individuals in a population may consist of one feasible solution, infeasible solutions, or a combination of feasible and infeasible solutions.*

PROPERTY 2 *Feasible solutions of non-dominated individuals in the offspring population can dominate either feasible or infeasible solutions in the parent population. However, infeasible solutions of non-dominated individuals in the offspring population can only dominate infeasible solutions in the parent population.*

Property 2 can be obtained from the definition of Pareto dominance.

As stated, the population of nine offspring is obtained after performing OD on two parents. Non-dominated individuals are then identified from the offspring population as the prior candidates, and are used to replace the individuals dominated by them in the two parents. However, it should be noted that in this work the comparisons among individuals are based on Pareto dominance. Thus, if the size of non-dominated individuals in the offspring population and the size of parents are both very small, the replacement may not take place very often. This is because non-dominated individuals in the offspring population might always be non-dominated relative to the parents as shown in figure 1. Obviously, such behaviour influences the convergence speed.

To address this problem, $\mu > 2$ individuals are chosen from the population as the parent population each time. The parents are mated into $\mu/2$ couples at random, and orthogonal crossover is applied to each pair of parents to generate nine offspring. Hence the offspring population is composed of $9\mu/2$ individuals. Thereafter, non-dominated individuals are chosen from the offspring population as the new potential offspring. This process is explained in figure 2.

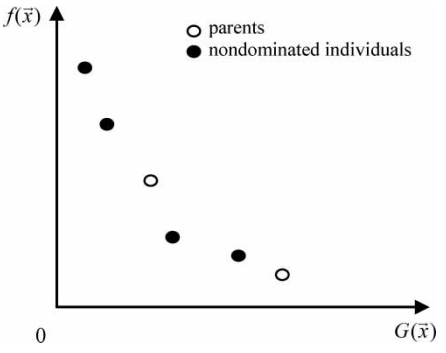

Figure 1. Suppose that there are four non-dominated individuals in the offspring population created by performing OD on two parents. Clearly, the six individuals (four non-dominated individuals and two parents) are non-dominated with each other according to the definition of Pareto dominance.

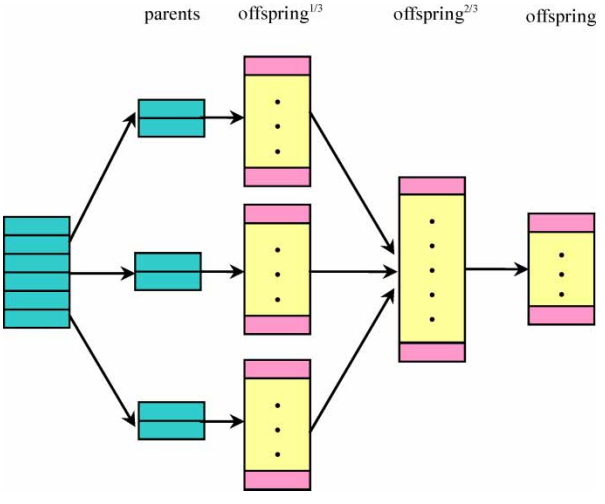

Figure 2. The procedure for producing the potential offspring with $\mu = 6$.

3.3 *Replacing selected parents with appropriate offspring*

A replacement operator is an important part of EAs. For constrained optimization, the point of the replacement operator is that both the optimization of the objective function and the handling of constraint should be taken into account. The replacement operator is expected to have the property of being able to allow low-fitness nearly feasible solutions to survive into the next population.

In this work a new approach to replacement is proposed. The approach of previous replacement schemes is that a better offspring can replace one of inferior parents, *i.e.* the replacement occurs mainly from the offspring's perspective. However, in this work the replacement is carried out from the parent's perspective, *i.e.* we decide which offspring has the opportunity to replace a chosen parent. Next, this idea will be generalized into constrained optimization.

The replacement scheme in ODCOEA works as follows. For each chosen parent, in the first step, non-dominated individuals who dominate it are identified from the offspring population (if they exist). Subsequently, the individual with the lowest degree of constraint violation in those non-dominated individuals is qualified to eliminate it. Note, however, that to avoid a

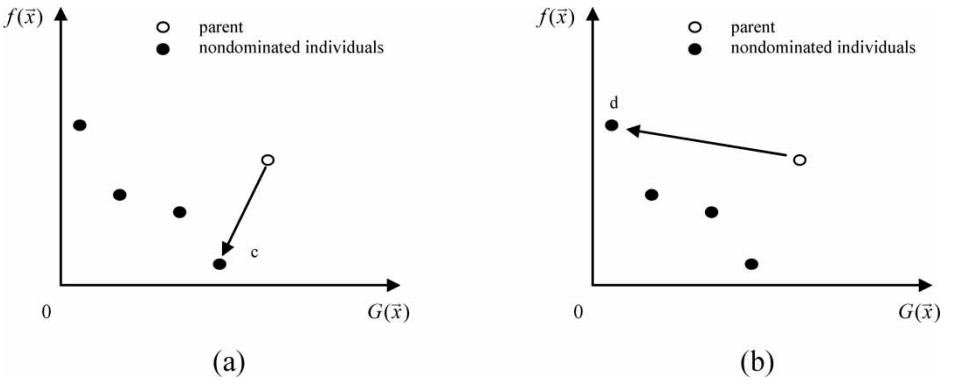

Figure 3. (a) In the previous replacement schemes, the chosen parent might be eliminated by the non-dominated individual 'c'. (b) Based on the replacement scheme presented in this work, the chosen parent is eliminated by the non-dominated individual 'd'. Clearly, the replacement proposed in this work is more reasonable.

greater selection pressure, only two parents randomly selected from the μ parents are specified to be replaced.

The two contrasting approaches to replacement are illustrated and compared in figure 3.

3.4 *Two special operations*

In addition to the replacement operation described above, two special operations are inserted so as to provide a greater assurance of obtaining feasible solutions.

3.4.1 Special operation 1. If non-dominated individuals selected from the offspring population are infeasible, this means that the offspring population is also infeasible, and simultaneously implies that the corresponding parent population, or a vast majority of individuals in the corresponding parent population, are still far from the feasible region. In this case, the individual with the lowest degree of constraint violation in the non-dominated individuals is used to replace an individual chosen from the parent population at random. The purpose of this operation is to move the population rapidly towards feasibility.

3.4.2 Special operation 2. Suppose that the non-dominated individuals selected from the offspring population contain one feasible solution. In this situation, in order to ensure that this feasible solution can survive into the next population, it is used to eliminate the worst solution in the parent population (if it is better than the worst solution). This process happens in 20 out of every 100 cycles during the evolution. The selection of the worst solution in the parent population is based on the following criteria (denoted as feasibility criteria).

(i) Between two feasible solutions, the one with lower objective function value wins.
(ii) If one solution is feasible and the other is infeasible, the feasible solution wins.
(iii) If both solutions are infeasible, the one with lower sum of constraint violation is preferred.

It is evident that in the feasibility criteria, feasible solutions are always considered to be better than infeasible solutions. Note that comparison of the worst solution in the parent population with the feasible solution in the non-dominated individuals also takes place in light of the feasibility criteria.

3.5 *Mutation operator*

An improved version of the BGA mutation operator (Mühlenbein and Schlierkamp-Voosen, 1993) is proposed in this article. Assume that $\vec{x} = (x_1, x_2, \ldots, x_n)$ is a chromosome to be mutated. A new chromosome $\vec{x}' = (x_1', x_2', \ldots, x_n')$ is generated as follows:

$$x_i' = x_i \pm \ rang_i \bullet \sum_{k=0}^{15} \alpha_k 2^{-k} \quad i = 1, \ldots, n. \tag{12}$$

where $rang_i$ defines the mutation range and is set to $(u_i - l_i) \bullet rand\ (0, 0.3)$. The plus or minus sign is chosen with a probability of 0.5 and $\alpha_k \in \{0, 1\}$ is randomly produced with $P(\alpha_k = 1) = 1/16$. The improvement proposed here is the dynamic setting of the

Table 3. Pseudocode of ODCOEA (flip(p) is a function that returns TRUE with probability p).

```
1.  Initialize the population P = (x⃗_1, x⃗_2, ..., x⃗_n);
2   Repeat
3     Randomly select μ individuals from the population as the parent population, denoted as
      A = (x⃗'_1, ..., x⃗'_μ);
4     P = P\A;
5     B = φ;
6     for i = 1 : μ/2 do
7.        B = orthogonal _crossover (x⃗'_{2i−1}, x⃗'_{2i});
8     end
9     Choose non-dominated individuals from the set B, assuming that these non-dominated
      individuals form a new set C;
10    if ¬∃ feasible individuals in the set C
11.     Select two individuals from the set A randomly. These two individuals are replaced with
        the corresponding individuals in the set C using the replacement scheme depicted in
        section 3.3;
12.     Use the individual with the lowest degree of constraint violation in the set C to randomly
        remove an individual in the set A; /* special operation 1*/
13.   else
14.     if flip(0.8)
15.       Select two individuals from the set A at random. These two individuals are
          replaced with the corresponding individuals in set C via the replacement scheme
          depicted in section 3.3;
16.     else/* special operation 2 */
17.       Choose the worst individual from the set A based on the feasibility criteria,
          denoted as x̂;
18.       Choose the feasible solution from the set C, denoted as x̃;
19.       if x̃ is better than x̂ (based on the feasibility criteria)
20        A = A ∪ {x̃}\{x̂};
21.       end if
22.     end if
23.   end if
24.   Each chromosomes in the set A undergoes the improved BGA mutation operator with
      probability p_m. To prevent the best fitness value from deteriorating, mutation is not applied to
      the best individuals, the best individual is defined based on the feasibility criteria;
25.   P = P ∪ A;
26. Until a predetermined number of fitness function evaluations (FFEs) is reached
```

parameter $rang_i$, which is intended to produce a diverse explorative behaviour throughout the evolution.

3.6 *Implementation of the proposed ODCOEA*

First, a good initial population P with N chromosomes is generated. ODCOEA is a steady-state evolutionary algorithm, *i.e.* only a few individuals in the population are selected and evolve in each iteration. After executing orthogonal crossover on the chosen parents, non-dominated individuals are selected from the offspring population as the new excellent offspring. Subsequently, a series of mechanisms is designed for replacement based on the different states of the non-dominated individuals. Finally, an improved BGA mutation is added. The details of the overall algorithm are shown in table 3.

4. Empirical study

4.1 *Test functions and the experimental conditions*

The performance of the proposed ODCOEA was evaluated using a set of 11 test functions (Koziel and Michalewicz 1999). These test cases include various types (linear, non-linear, and quadratic) of objective function with different numbers n of decision variables and a range of types (linear inequalities (LI), non-linear equalities (NE), and non-linear inequalities (NI)) and number of constraints. The main characteristics of the test cases are given in table 4, where a is the number of constraints *active* at the optimal solution and ρ is the ratio of the size of the feasible search space to that of the entire search space, *i.e.*

$$\rho = \frac{|\Omega|}{|S|} \tag{13}$$

where $|S|$ is the number of solutions randomly generated from S and $|\Omega|$ is the number of feasible solutions out of these $|S|$ solutions. In the experimental setup, $|S| = 1\,000\,000$.

These 11 problems pose a challenge for constraint-handling methods and provide a good test of their ability. ODCOEA performed 30 independent runs for each test function. The parameters of ODCOEA were $N = 100$, $\mu = 6$, and $p_m = 0.2$. The number of fitness function evaluations (FFEs) is fixed at 150 000. An equality constraint is usually replaced by pairwise inequality constraints $h(\vec{x}) - \delta \leq 0$ and $-h(\vec{x}) - \delta \leq 0$, where δ is a small positive value. In this work, $\delta = 0.001$.

Table 4. Summary of 11 benchmark problems.

Function	n	Type of f	ρ (%)	LI	NE	NI	a
g01	13	Quadratic	0.0003	9	0	0	6
g02	20	Non-linear	99.9965	1	0	1	1
g03	10	Non-linear	0.0000	0	1	0	1
g04	5	Quadratic	26.9356	0	0	6	2
g05	4	Non-linear	0.0000	2	3	0	3
g06	2	Non-linear	0.0064	0	0	2	2
g07	10	Quadratic	0.0003	3	0	5	6
g08	2	Non-linear	0.8640	0	0	2	0
g09	7	Non-linear	0.5256	0	0	4	2
g10	8	Linear	0.0005	3	0	3	3
g11	2	Quadratic	0.0000	0	1	0	1

4.2 *General performance of the proposed algorithm*

The experimental results obtained using the above parameters are summarized in table 5. This table shows the known 'optimal' solutions for each problem, and records the best, median, mean, and worst of the objective function values, and the standard deviations (SDs), found over 30 independent runs.

One of the first observations is that, for each problem, the best solution is almost equivalent to the optimal solution. For problems g01, g03, g08, and g11, optimal solutions are consistently found in all 30 runs. For problems g02, g04, g06, g07, and g09, near-optimal solutions are found in all 30 runs. The optimal solution is not consistently found for problem g05; the primary feature of this function is that it contains three equality constraints. Another problem whose optimum is not found consistently is g10; the main characteristics of this problem are its relatively large search space and three *active* constraints at the optimum.

In addition, the standard deviations over 30 runs for all the problems other than g05 and g10 are relatively small. These results confirm that ODCOEA has a substantial capability of handling various COPs and its solution quality is quite stable.

4.3 *Convergence dynamics of the proposed ODCOEA*

Having discussed the quality and robustness of the proposed approach, it is necessary to verify the rate at which the algorithm is able to achieve optimal or near-optimal solutions, *i.e.* the convergence dynamics of the approach. The convergence result for each problem is shown in figures 4 and 5 so that the efficiency of ODCOEA can be demonstrated more explicitly.

As can be seen, for six problems (g01, g02, g03, g04, g06 and g10) ODCOEA converges to a stable value after about 2000 generations (*i.e.* about 56 000 FFEs). Furthermore, ODCOEA achieves the convergence at a very early stage for the remaining problems. These results can be attributed to the following two merits of ODCOEA. Firstly the uniform distribution of the initial population in the search space can be guaranteed by the initialization procedure. Secondly, OD can evenly scan the search space once to locate the good points for further exploration in subsequent iterations.

4.4 *Comparison with the self-adaptive fitness formulation and the generic framework for constrained optimization*

ODCOEA is compared with two state-of-the-art approaches: the self-adaptive fitness formulation (SAFF) (Farmani and Wright 2003) and the generic framework for constrained

Table 5. Statistical results obtained by ODCOEA for the 11 benchmark test problems over 30 independent runs.

Function	Optimal	Best	Median	Mean	Worst	SD
g01	−15.000	−15.000	−15.000	−14.999	−14.999	2.1×10^{-4}
↑g02	0.803619	0.802996	0.793227	0.792587	0.762508	9.8×10^{-3}
↑g03	1.00	1.00	1.00	1.00	1.00	7.1×10^{-4}
g04	−30665.539	−30665.536	−30665.472	−30665.428	−30664.909	1.2×10^{-1}
g05	5126.498	5126.498	5139.441	5158.922	5249.496	3.4×10^{1}
g06	−6961.814	−6961.699	−6960.700	−6960.669	−6959.281	6.5×10^{-1}
g07	24.306	24.585	25.625	25.886	27.262	7.3×10^{-1}
↑g08	0.095825	0.095825	0.095825	0.095825	0.095825	2.6×10^{-13}
g09	680.630	680.636	680.744	680.774	680.998	9.0×10^{-2}
g10	7049.248	7054.547	7360.659	7552.225	8428.351	4.4×10^{2}
g11	0.75	0.75	0.75	0.75	0.75	1.5×10^{-5}

↑Problem that should be maximized.

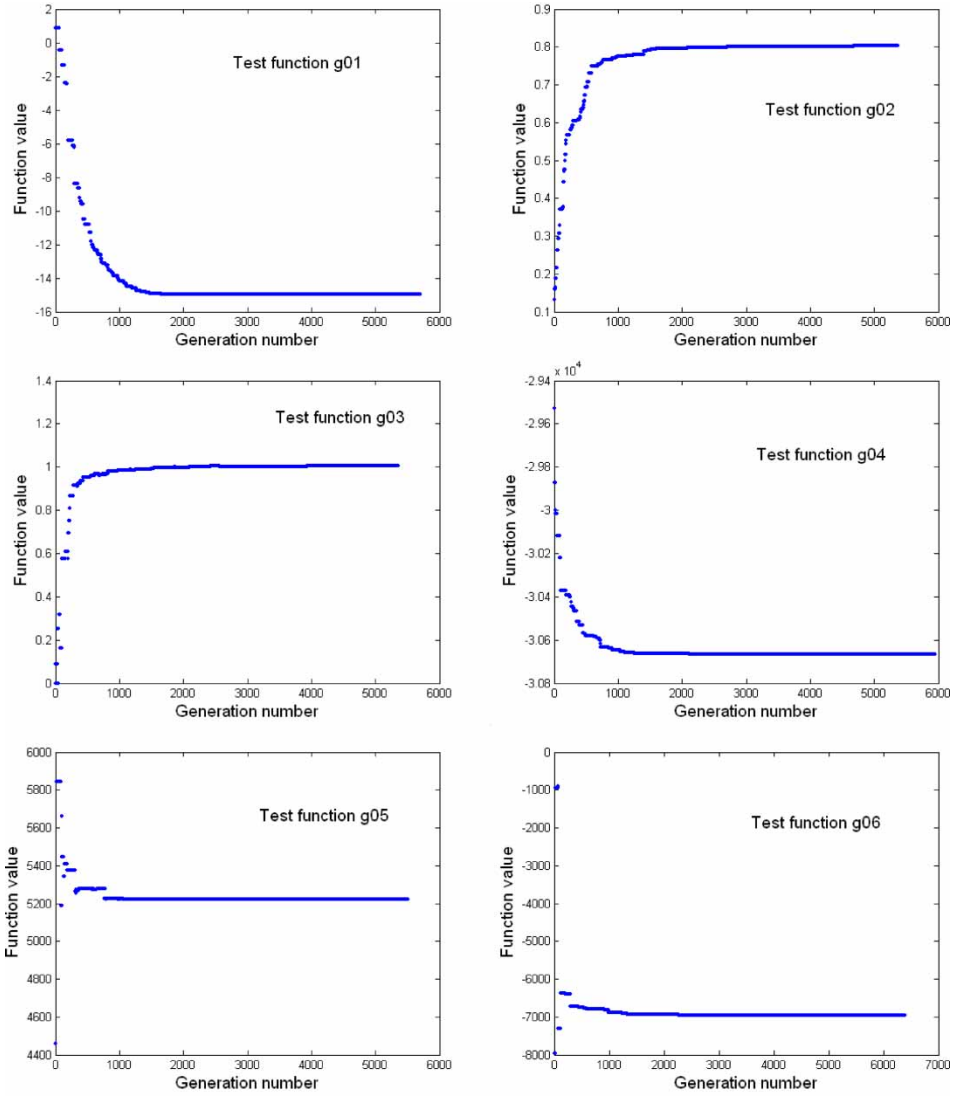

Figure 4. Convergence results for test problems g01–g06.

optimization (GFCO) (Venkatraman and Yen 2005). In principle, SAFF belongs to the class of methods based on adaptive penalty functions and is an enhanced version of the algorithm in Wright and Farmani (2001), where the penalty is divided into two stages. The improved approach eliminates the fixed penalty factor for the second penalty stage proposed by Wright and Farmani (2001), by assigning the penalized objective function value of the worst individual to be equal to that of the individual with maximum objective function value in the current population. This makes the method self-adaptive, as well as more dynamic, because the individual with maximum objective function value may vary from one generation to another. GFCO is illustrated in section 1.

As shown in tables 6 and 7, the performance of ODCOEA is compared in detail with SAFF and GFCO by examining the selected performance metrics.

With respect to SAFF, the proposed approach finds better ‘best’ results in four problems (g02, g04, g05, and g10) and similar ‘best’ results in five problems (g01, g03, g08, g09,

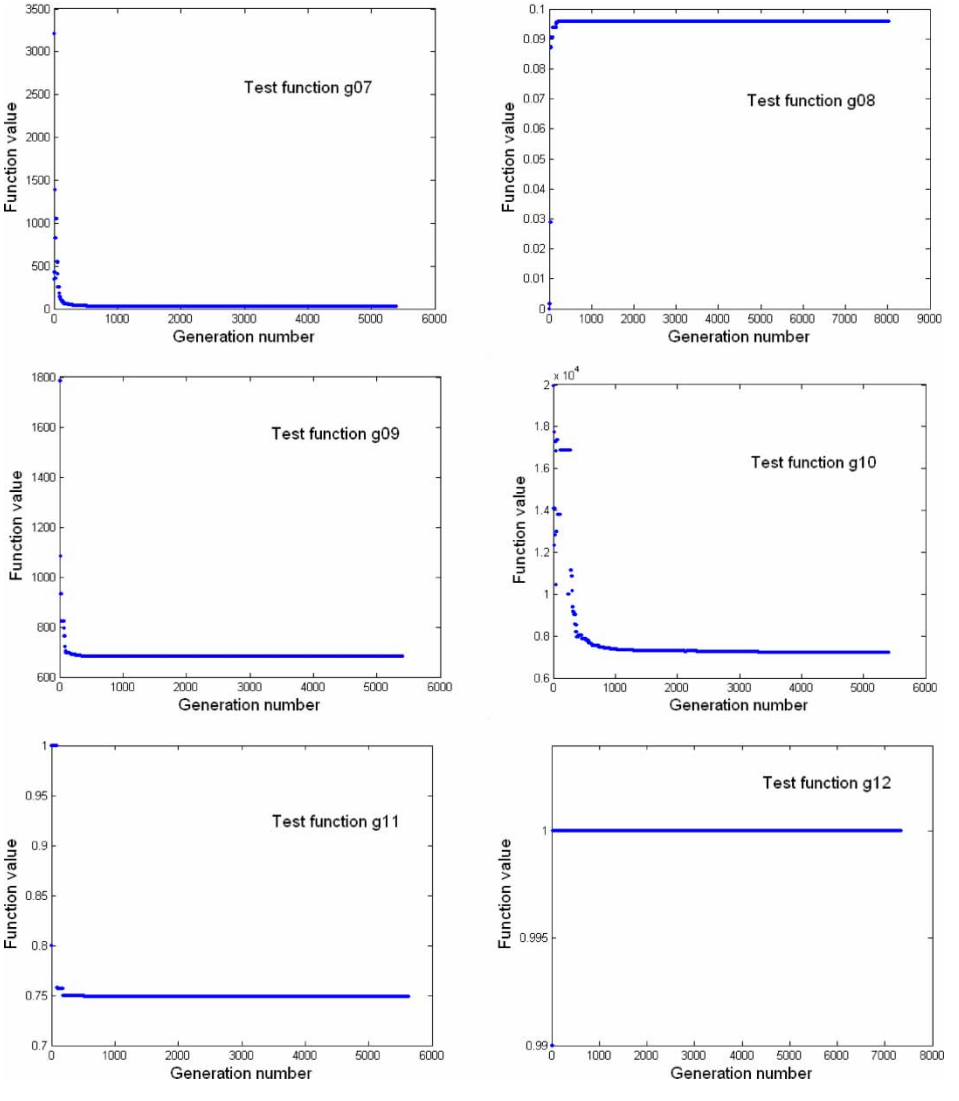

Figure 5. Convergence results for test problems g07–g12.

Table 6. Comparison of ODCOEA with SAFF (Farmani and Wright 2003) for 11 benchmark functions.

		Best result		Mean result		Worst result	
Function	Optimal	ODCOEA	SAFF	ODCOEA	SAFF	ODCOEA	SAFF
g01	−15.000	**−15.000**	**−15.000**	−14.999	**−15.000**	−14.999	**−15.000**
↑g02	0.803619	**0.802996**	0.80297	**0.792587**	0.79010	**0.762508**	0.76043
↑g03	1.00	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
g04	−30665.539	**−30665.536**	−30665.50	**−30665.428**	−30665.20	**−30664.909**	−30663.30
g05	5126.498	**5126.498**	5126.989	**5158.922**	5432.080	**5249.496**	6089.430
g06	−6961.814	−6961.699	**−6961.800**	−6960.669	**−6961.800**	−6959.281	**−6961.800**
g07	24.306	24.585	**24.48**	**25.886**	26.58	**27.262**	28.40
↑g08	0.095825	**0.095825**	**0.095825**	**0.095825**	**0.095825**	**0.095825**	**0.095825**
g09	680.630	**680.636**	**680.64**	680.774	**680.72**	680.998	**680.87**
g10	7049.248	**7054.547**	7061.34	**7552.225**	7627.89	8428.351	**8288.79**
g11	0.75	**0.75**	**0.75**	**0.75**	**0.75**	**0.75**	**0.75**

Bold type indicates the better result between the two compared algorithms or that the global optimum was reached.

Table 7. Comparison of ODCOEA with GFCO (Venkatraman and Yen 2005) for 11 benchmark functions.

		Best result		Mean result		Worst result	
Function	Optimal	ODCOEA	SAFF	ODCOEA	SAFF	ODCOEA	SAFF
g01	−15.000	**−15.000**	**−15.000**	**−15.000**	**−15.000**	**−14.999**	−12.000
↑g02	0.803619	0.802996	**0.803190**	**0.793227**	0.755332	**0.762508**	0.672169
↑g03	1.00	**1.00**	**1.00**	**1.00**	0.95	**1.00**	0.79
g04	−30665.539	**−30665.536**	−30665.531	**−30665.472**	−30663.364	**−30664.909**	−30651.960
g05	5126.498	**5126.498**	5126.510	**5139.441**	5170.529	**5249.496**	6112.223
g06	−6961.814	**−6961.699**	−6961.179	**−6960.700**	−6959.568	**−6959.281**	−6954.319
g07	24.306	24.585	**24.411**	**25.625**	26.736	**27.262**	35.882
↑g08	0.095825	**0.095825**	**0.095825**	**0.095825**	**0.095825**	**0.095825**	**0.095825**
g09	680.630	**680.636**	680.762	**680.744**	681.706	**680.998**	684.131
g10	7049.248	**7054.547**	7060.553	**7360.659**	7723.167	**8428.351**	12097.408
g11	0.75	**0.75**	**0.75**	**0.75**	**0.75**	**0.75**	0.809

Bold type indicates the better result between the two compared algorithms or that the global optimum was reached.

and g11). SAFF finds better 'best' solutions in the remaining two problems (g06 and g07). The proposed technique achieves better 'mean' and 'worst' results in four problems (g02, g04, g05, and g07). Similar 'mean' and 'worst' results are found in three problems (g03, g08, and g11). SAFF attains better 'mean' and 'worst' solutions in three problems (g01, g06, and g09). Note that, for problem g10, the proposed technique has a better 'mean' solution, while SAFF has a better 'worst' solution. In addition, for problems g03, g08, and g11, the optimal solutions are consistently found by the two compared methods.

With respect to GFCO, the method proposed here provides better 'best' solutions in five problems (g04, g05, g06, g09, and g10) and similar 'best' results in four problems (g01, g03, g08, and g11). Better 'best' results are found by GFCO in problems g02 and g07. The proposed method finds better 'medium' results in eight problems (g02, g03, g04, g05, g06, g07, g09, and g10) and similar 'medium' results in remaining three (g01, g08, and g11). It also finds better 'worst' results in ten problems, except for problem g08 in which the the 'worst' result is similar to that obtained with GFCO.

As far as the computational cost (*i.e.* the number of FFEs) is concerned, GFCO has the minimum cost (50 000 FFEs) for all the test functions, ODCOEA has the medium cost (150 000 FFEs) for all the test functions, and SAFF has the highest cost (1 400 000 FFEs) for all the test functions. The computational cost of SAFF is nearly 10 times more than that of ODCOEA.

In summary, we can conclude that ODCOEA is slightly superior to SAFF and substantially superior to GFCO in terms of the quality of the resulting solutions. However, GFCO is the most efficient method. In general, ODCOEA can be considered as a good trade-off between efficiency and effectiveness for constrained evolutionary optimization.

4.5 *Performance of the algorithm in a discontinuous search space*

The problem below is adopted to further test the performance of the proposed ODCOEA in solving problems in a discontinuous search space:

$$\text{Maximize} \quad f(\vec{x}) = \frac{100 - (x_1 - 5)^2 - (x_2 - 5)^2 - (x_3 - 5)^2}{100}$$

$$\text{subject to} \quad g(\vec{x}) = (x_1 - p)^2 + (x_2 - q)^2 + (x_3 - r)^2 - 0.0625 \leq 0$$

where

$$0 \leq x_i \leq 10 \quad (i = 1, 2, 3) \text{ and } p, q, r = 1, , 2, \ldots, 9.$$

The feasible region of the search space consists of 9^3 disjointed spheres. A point (x_1, x_2, x_3) is feasible if and only if there exist p, q, r such that the above inequality holds. The optimum solution is located at $\vec{x}^*(5, 5, 5)$ where $f(\vec{x}^*) = 1$. The solution lies within the feasible region.

For this test problem, 30 independent runs were performed with the parameters used in the previous experiments. During the evolutionary process, ODCOEA easily reached the global optimal solution. The best, average, and worst results and the standard deviations obtained over 30 runs are 1, 1, 1, and 0, respectively. The convergence trace of this test problem (denoted g12) is shown in figure 5. Note, however, that the corresponding results for SAFF are 1, 0.994 375, 0.999 718, and 5.34×10^{-4}, respectively, after 350 000 FFEs, and match the results obtained with ODCOEA only after 1 400 000 FFEs. This suggests that ODCOEA can efficiently deal with different types of COPs, even in a disjoint search space.

5. Conclusion

A new approach to handle constraints in evolutionary optimization, which incorporates orthogonal design and multi-objective optimization techniques, is proposed in this article. In this approach only a few individuals are selected from the population as the parents at each generation. Thereafter, pairs of parents are mated at random and orthogonal crossover is used to produce a set of offspring. After combining the offspring generated from different pairs of parents, non-dominated individuals are selected as the potential offspring and used to replace the individuals chosen from the parents by various replacement mechanisms. In addition, an improved BGA mutation operator is added to promote the diversity of the population.

It is shown that this approach can be used to solve a range of COPs with linear/nonlinear equality/inequality constraints, as well as continuous/discontinuous search spaces. Its overall capability is superior to that of two other COEAs which are representative of the state-of-the-art in constrained evolutionary optimization. More importantly, this approach is easy to implement and its computational cost is relatively low.

Acknowledgements

The authors acknowledge support from National Natural Science Foundation of China under Grants 60234030 and 60673062, National Basic Scientific Research Funds under Grant A1420060159, and Hunan S&T Funds under Grant 05IJY3035.

References

Aguirre, A.H., Rionda, S.B., Coello, C.A.C., Lizáraga, G.L. and Montes, E.M. Handling constraints using multi-objective optimization concepts. *Int. J. Numer. Methods Eng.*, 2004, **59**(15), 1989–2017.

Cai, Z. and Wang, Y., A multiobjective optimization-based evolutionary algorithm for constrained optimization. *IEEE Trans. Evol. Comput.*, 2006, **10**(6), 658–675.

Coello, C.A.C., Treating constraints as objectives for single-objective evolutionary optimization. *Eng. Optimiz.*, 2000a, **32**(3), 275–308. f

Coello, C.A.C., Constraint handling using an evolutionary multi-objective optimization technique. *Civil Eng. Environ. Systems.*, 2000b, **17**, 319–346.

Coello, C.A.C., Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. *Comput. Methods Appl. Mech. Eng.*, 2002, **191**(11–12), 1245–1287.

Coello, C.A.C. and Mezura-Montes, E., Constraint-handling in genetic algorithms through the use of dominance-based tournament Selection. *Adv. Eng. Inform.*, 2002, **16**(3), 193–203.

Deb, K., An efficient constraint handling method for genetic algorithms. *Comput. Methods Appl. Mech. Eng.*, 2000, **18**, 311–338.

Deb, K., Pratab, A., Agrawal, S. and Meyarivan, T., A fast and elitist non-dominated sorting genetic algorithm for multiobjective optimization: NSGA II. *IEEE Trans. Evol. Comput*., 2002, **6**(2), 182–197.
Farmani, R. and Wright, J.A., Self-adaptive fitness formulation for constrained optimization. *IEEE Trans. Evol. Comput*., 2003, **7**(5), 445–455.
Fonseca, C.M. and Fleming, P.J., Multi-objective optimization and multiple constraint handling with evolutionary algorithms-part I: a unified formulation. *IEEE Trans. Systems Man Cybern.*, 1999, **28**(1), 26–37.
Ho, S.Y. and Chen, Y.C., An efficient evolutionary algorithm for accurate polygonal approximation. *Pattern Recogn*., 2001, **34**(12), 2305–2317.
Ho, S.Y., Shu, L. and Chen, J.H., Intelligent evolutionary algorithms for large parameter optimization problems. *IEEE Trans. Evol. Comput*., 2004, **8**(6), 522–541.
Horn J., Nafpliotis, N. and Goldberg, D., A niched Pareto genetic algorithm for multi-objective optimization, in *Proceedings of the 1st IEEE Conf on Evolutionary Computation*, pp. 82–87 1994 (IEEE Press: Piscataway, NJ).
Knowles, J.D. and Corne, D.W., Approximating the non-dominated front using the Pareto archived evolutionary strategy. *Evol. Comput*., 2000, **8**(2), 149–172.
Koziel, S. and Michalewicz, Z., Evolutionary algorithm, homomorphous mappings, and constrained parameter optimization. *Evol. Comput*., 1999, **7**(1), 19–44.
Leung, Y.W. and Wang, Y., An orthogonal genetic algorithm with quantization for global numerical optimization. *IEEE Trans. Evol. Comput*., 2001, **5**(1), 41–53.
Mezura-Montes, E. and Coello, C.A.C., A simple multi-membered evolution strategy to solve constrained optimization problems. *IEEE Trans. Evol. Comput*., 2005, **9**(1), 1–17.
Michalewicz, Z. and Schoenauer, M., Evolutionary algorithm for constrained parameter optimization problems. *Evol. Comput*., 1996, **4**(1), 1–32.
Michalewicz, Z., Deb, K., Schmidt, M. and Stidsen, T., Test-case generator for constrained parameter optimization techniques. *IEEE Trans. Evol. Comput.*, 2000, **4**(3), 197–215.
Mühlenbein, H. and Schlierkamp-Voosen, D., Predictive models for the breeder genetic algorithm I: continuous parameter optimization. *Evol Comput.*, 1993, **1**(1), 25–49.
Runarsson, T.P. and Yao, X., Stochastic ranking for constrained evolutionary optimization. *IEEE Trans. Evol. Comput*., 2000, **4**(3), 284–294.
Schaffer, J.D., Multiple objective optimization with vector evaluated genetic algorithms, in *Proceedings of the 1st International Conference on Genetic Algorithms and their Applications*, edited by J.J. Grefenstette, pp. 93–100, 1985 (Lawrence Erlbaum: Mahwah, NJ).
Surry, P.D. and Radcliffe, N.J., The COMOGA method: constrained optimization by multi-objective genetic algorithm. *Control Cybernet.*, 1997, **26**(3), 391–412.
Tsai, J.T., Liu, T.K. and Chou, J.H., Hybrid Taguchi–genetic algorithm for global numerical optimization. *IEEE Trans. Evol. Comput*., 2004, **8**(4), 365–377.
Venkatraman, S. and Yen, G.G., A generic framework for constrained optimization using genetic algorithms. *IEEE Trans. Evol. Comput*., 2005, **9**(4), 424–435.
Wright, J.A. and Farmani, R., Genetic algorithm: a fitness formulation for constrained minimization, in *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 725–732, 2001 (San Francisco, CA: Morgan Kaufmann).
Wu, Q., On the optimality of orthogonal experimental design. *Acta Math. Appl Sin.*, 1978, **1**(4), 283–299.
Yu, J.X., Yao, X., Choi, C. and Gou, G., Materialized view selection as constrained evolutionary optimization. *IEEE Trans. Systems Man, Cybern. C*, 2003, **33**(4), 458–467.
Zeng, S.Y., Kang, L.S. and Ding, L.X., An orthogonal multi-objective evolutionary algorithm for multi-objective optimization problems with constraints. *Evol. Comput*., 2004, **12**(1), 77–98.
Zhang, Q. and Leung, Y.W., Orthogonal genetic algorithm for multimedia multicast routing. *IEEE Trans. Evol. Comput.*, 1999, **3**(1), 53–62.
Zhong, W., Liu, J., Xue, M. and Jiao, L., A multi-agent genetic algorithm for global numerical optimization. *IEEE Trans. Systems Man Cybern. B*, 2004, **34**(2), 1128–1141.
Zhou, Y., Li, Y., He, J. and Kang, L., Multi-objective and MGG evolutionary algorithm for constrained optimization, in *Proceedings of the Congress on Evolutionary Computing 2003 (CEC'2003)*, pp. 1–5, 2003 (IEEE Press: Piscataway, NJ).

Appendix. Test functions

Function g01

Minimize

$$f(\vec{x}) = 5\sum_{i=1}^{4} x_i - 5\sum_{i=1}^{4} x_i^2 - \sum_{i=5}^{13} x_i$$

subject to

$$g_1(\vec{x}) = 2x_1 + 2x_2 + x_{10} + x_{11} - 10 \leq 0$$

$$g_2(\vec{x}) = 2x_1 + 2x_3 + x_{10} + x_{12} - 10 \leq 0$$

$$g_3(\vec{x}) = 2x_2 + 2x_3 + x_{11} + x_{12} - 10 \leq 0$$

$$g_4(\vec{x}) = -8x_1 + x_{10} \leq 0$$

$$g_5(\vec{x}) = -8x_2 + x_{11} \leq 0$$

$$g_6(\vec{x}) = -8x_3 + x_{12} \leq 0$$

$$g_7(\vec{x}) = -2x_4 - x_5 + x_{10} \leq 0$$

$$g_8(\vec{x}) = -2x_6 - x_7 + x_{11} \leq 0$$

$$g_9(\vec{x}) = -2x_8 - x_9 + x_{12} \leq 0$$

where the bounds are $0 \leq x_i \leq 1 \ \ (i = 1, \ldots, 9)$, $0 \leq x_i \leq 100 \ \ (i = 10, 11, 12)$, and $0 \leq x_{13} \leq 1$. The global minimum is at $\vec{x}^* = (1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3, 3, 1)$, where $f(\vec{x}^*) = -15$. Constraints g_1, g_2, g_3, g_7, g_8, and g_9 are active.

Function g02

Maximize

$$f(\vec{x}) = \left| \frac{\sum_{i=1}^{n} \cos^4(x_i) - 2\Pi_{i=1}^{n} \cos^2(x_i)}{\sqrt{\sum_{i=1}^{n} i x_i^2}} \right|$$

subject to

$$g_1(\vec{x}) = 0.75 - \Pi_{i=1}^{n} x_i \leq 0$$

$$g_2(\vec{x}) = \sum_{i=1}^{n} x_i - 7.5n \leq 0$$

where $n = 20$ and $0 \leq x_i \leq 10 \ \ (i = 1, \ldots, n)$. The global maximum is unknown; the best reported solution is $f(\vec{x}^*) = 0.803619$. Constraint g_1 is close to being active ($g_1 = -10^{-8}$).

Function g03

Maximize

$$f(\vec{x}) = (\sqrt{n})^n \prod_{i=1}^{n} x_i$$

subject to

$$h(\vec{x}) = \sum_{i=1}^{n} x_i^2 - 1 = 0$$

where $n = 10$ and $0 \le x_i \le 1$ $(i = 1, \ldots, n)$. The global maximum is at $x_i^* = 1/\sqrt{n}(i = 1, \ldots, n)$, where $f(\vec{x}^*) = 1$.

Function g04

Minimize

$$f(\vec{x}) = 5.3578547x_3^2 + 0.8356891x_1x_5 + 37.293239x_1 - 40792.141$$

subject to

$$g_1(\vec{x}) = 85.334407 + 0.0056858x_2x_5 + 0.0006262x_1x_4 - 0.0022053x_3x_6 - 92 \le 0$$

$$g_2(\vec{x}) = -85.334407 - 0.0056858x_2x_5 - 0.0006262x_1x_4 + 0.0022053x_3x_6 \le 0$$

$$g_3(\vec{x}) = 80.51249 + 0.0071317x_2x_5 + 0.0029955x_1x_2 + 0.0021813x_3^2 - 110 \le 0$$

$$g_4(\vec{x}) = -80.51249 - 0.0071317x_2x_5 - 0.0029955x_1x_2 - 0.0021813x_3^2 + 90 \le 0$$

$$g_5(\vec{x}) = 9.300961 + 0.0047026x_3x_5 + 0.0012547x_1x_3 + 0.0019085x_3x_4 - 25 \le 0$$

$$g_6(\vec{x}) = -9.300961 - 0.0047026x_3x_5 - 0.0012547x_1x_3 - 0.0019085x_3x_4 + 20 \le 0$$

where $78 \le x_1 \le 102$, $33 \le x_2 \le 45$, and $27 \le x_i \le 45$ $(i = 3, 4, 5)$. The optimum solution is $\vec{x}^* = (78, 33, 29.995256025682, 45, 36.775812905788)$, where $f(\vec{x}^*) = -30665.539$. Constraints g_1 and g_6 are active.

Function g05

Minimize

$$f(\vec{x}) = 3x_1 + 0.000001x_1^3 + 2x_2 + (0.000002/3)x_2^3$$

subject to

$$g_1(\vec{x}) = -x_4 + x_3 - 0.55 \le 0$$

$$g_2(\vec{x}) = -x_3 + x_4 - 0.55 \le 0$$

$$h_3(\vec{x}) = 1000\sin(-x_3 - 0.25) + 1000\sin(-x_4 - 0.25) + 894.8 - x_1 = 0$$

$$h_4(\vec{x}) = 1000\sin(x_3 - 0.25) + 1000\sin(x_3 - x_4 - 0.25) + 894.8 - x_2 = 0$$

$$h_5(\vec{x}) = 1000\sin(x_4 - 0.25) + 1000\sin(x_4 - x_3 - 0.25) + 1294.8 = 0$$

where $0 \le x_1 \le 1200$, $0 \le x_2 \le 1200$, $-0.55 \le x_3 \le 0.55$, and $-0.55 \le x_4 \le 0.55$. The best known solution is $\vec{x}^* = (679.9453, 1026.067, 0.1188764, -0.3962336)$, where $f(\vec{x}^*) = 5126.4981$.

Function g06

Minimize

$$f(\vec{x}) = (x_1 - 10)^3 + (x_2 - 20)^3$$

subject to

$$g_1(\vec{x}) = -(x_1 - 5)^2 - (x_2 - 5)^2 + 100 \leq 0$$

$$g_2(\vec{x}) = (x_1 - 6)^2 + (x_2 - 5)^2 - 82.81 \leq 0$$

where $13 \leq x_1 \leq 100$ and $0 \leq x_2 \leq 100$. The optimum solution is $\vec{x}^* = (14.095, 0.84296)$, where $f(\vec{x}^*) = -6961.81388$. Both constraints are active.

Function g07

Minimize

$$f(\vec{x}) = x_1^2 + x_2^2 + x_1x_2 - 14x_1 - 16x_2 + (x_3 - 10)^2 + 4(x_4 - 5)^2 + (x_5 - 3)^2 + 2(x_6 - 1)^2 + 5x_7^2 + 7(x_8 - 11)^2 + 2(x_9 - 10)^2 + (x_{10} - 7)^2 + 45$$

subject to

$$g_1(\vec{x}) = -105 + 4x_1 + 5x_2 - 3x_7 + 9x_8 \leq 0$$

$$g_2(\vec{x}) = 10x_1 - 8x_2 - 17x_7 + 2x_8 \leq 0$$

$$g_3(\vec{x}) = -8x_1 + 2x_2 + 5x_9 - 2x_{10} - 12 \leq 0$$

$$g_4(\vec{x}) = 3(x_1 - 2)^2 + 4(x_2 - 3)^2 + 2x_3^2 - 7x_4 - 120 \leq 0$$

$$g_5(\vec{x}) = 5x_1^2 + 8x_2 + (x_3 - 6)^2 - 2x_4 - 40 \leq 0$$

$$g_6(\vec{x}) = x_1^2 + 2(x_2 - 2)^2 + -2x_1x_2 + 14x_5 - 6x_6 \leq 0$$

$$g_7(\vec{x}) = 0.5(x_1 - 8)^2 + 2(x_2 - 4)^2 + 3x_5^2 - x_6 - 30 \leq 0$$

$$g_8(\vec{x}) = -3x_1 + 6x_2 + 12(x_9 - 8)^2 - 7x_{10} \leq 0$$

where $-10 \leq x_i \leq 10$ $(i = 1, \ldots, 10)$. The optimum solution is $\vec{x}^* = (2.171996, 2.363683, 8.773926, 5.095984, 0.9906548, 1.430574, 1.321644, 9.828726, 8.280092, 8.375927)$ where $f(\vec{x}^*) = 24.3062091$. Constraints g_1, g_2, g_3, g_4, g_5, and g_6 are active.

Function g08

Minimize

$$f(\vec{x}) = \frac{\sin^3(2\pi x_1)\sin(\pi x_2)}{x_1^3(x_1 + x_2)}$$

subject to

$$g_1(\vec{x}) = x_1^2 - x_2 + 1 \leq 0$$

$$g_2(\vec{x}) = 1 - x_1 + (x_2 - 4)^2 \leq 0$$

where $0 \leq x_1 \leq 10$ and $0 \leq x_2 \leq 10$. The optimum solution is located at $\vec{x}^* = (1.2279713, 4.2453733)$, where $f(\vec{x}^*) = 0.095825$. The solution lies within the feasible region.

Function g09

Minimize

$$f(\vec{x}) = (x_1 - 10)^2 + 5(x_2 - 12)^2 + x_3^4 + 3(x_4 - 11)^2 + 10x_5^6 + 7x_6^2 + x_7^4 - 4x_6x_7 - 10x_6 - 8x_7$$

subject to

$$g_1(\vec{x}) = -127 + 2x_1^2 + 3x_2^4 + x_3 + 4x_4^2 + 5x_5 \leq 0$$

$$g_2(\vec{x}) = -282 + 7x_1 + 3x_2 + 10x_3^2 + x_4 - x_5 \leq 0$$

$$g_3(\vec{x}) = -196 + 23x_1 + x_2^2 + 6x_6^6 - 8x_7 \leq 0$$

$$g_4(\vec{x}) = 4x_1^2 + x_2^2 - 3x_1x_2 + 2x_3^2 + 5x_6 - 11x_7 \leq 0$$

where $-10 \leq x_i \leq 10$ $(i = 1, \ldots, 7)$. The optimum solution is $\vec{x}^* = (2.330499, 1.951372, -0.4775414, 4.365726, -0.6244870, 1.1038131, 1.594227)$, where $f(\vec{x}^*) = 680.6300573$. Constraints g_1 and g_4 are active.

Function g10

Minimize

$$f(\vec{x}) = x_1 + x_2 + x_3$$

subject to

$$g_1(\vec{x}) = -1 + 0.0025(x_4 + x_6) \leq 0$$

$$g_2(\vec{x}) = -1 + 0.0025(x_5 + x_7 - x_4) \leq 0$$

$$g_3(\vec{x}) = -1 + 0.01(x_8 - x_5) \leq 0$$

$$g_4(\vec{x}) = -x_1x_6 + 833.33252x_4 + 100x_1 - 83333.333 \leq 0$$

$$g_5(\vec{x}) = -x_2x_7 + 1250x_5 + x_2x_4 - 1250x_4 \leq 0$$

$$g_6(\vec{x}) = -x_3x_8 + 1250000 + x_3x_5 - 2500x_5 \leq 0$$

where $100 \leq x_1 \leq 10\,000$, $1000 \leq x_i \leq 10\,000$ $(i = 2, 3)$, and $10 \leq x_i \leq 10000 (i = 4, \ldots, 8)$. The optimum solution is $\vec{x}^* = (579.19, 1360.13, 5109.92, 182.0174, 295.5985, 217.9799, 286.40, 395.5979)$, where $f(\vec{x}^*) = 7049.248$. Constraints g_1, g_2, and g_3 are active.

Function g11

Minimize

$$f(\vec{x}) = x_1^2 + (x_2 - 1)^2$$

subject to

$$h(\vec{x}) = x_2 - x_1^2 = 0$$

where $-1 \leq x_1 \leq 1$ and $-1 \leq x_2 \leq 1$. The optimum solution is $\vec{x}^* = (\pm 1/\sqrt{2}, 1/2)$, where $f(\vec{x}^*) = 0.75$.