Constrained optimization based on hybrid evolutionary algorithm and adaptive constraint-handling technique

Article *in* Structural and Multidisciplinary Optimization · January 2008

DOI: 10.1007/s00158-008-0238-3

CITATIONS
97

READS
142

4 authors, including:

Yong Wang
Central South University
46 PUBLICATIONS **2,525** CITATIONS

SEE PROFILE

Yuren Zhou
South China University of Technology
35 PUBLICATIONS **792** CITATIONS

SEE PROFILE

Zhun Fan
Shantou University
148 PUBLICATIONS **970** CITATIONS

SEE PROFILE

Struct Multidisc Optim (2009) 37:395–413
DOI 10.1007/s00158-008-0238-3

RESEARCH PAPER

Constrained optimization based on hybrid evolutionary algorithm and adaptive constraint-handling technique

Yong Wang · Zixing Cai · Yuren Zhou · Zhun Fan

Received: 1 September 2007 / Revised: 21 November 2007 / Accepted: 21 January 2008 / Published online: 14 March 2008

Abstract A novel approach to deal with numerical and engineering constrained optimization problems, which incorporates a hybrid evolutionary algorithm and an adaptive constraint-handling technique, is presented in this paper. The hybrid evolutionary algorithm simultaneously uses simplex crossover and two mutation operators to generate the offspring population. Additionally, the adaptive constraint-handling technique consists of three main situations. In detail, at each situation, one constraint-handling mechanism is designed based on current population state. Experiments on 13 benchmark test functions and four well-known constrained design problems verify the effectiveness and efficiency of the proposed method. The experimental results show that integrating the hybrid evolutionary algorithm with the adaptive constraint-handling technique is beneficial, and the proposed method achieves competitive performance with respect to some other state-of-the-art approaches in constrained evolutionary optimization.

Y. Wang (✉) · Z. Cai
School of Information Science and Engineering,
Central South University, Changsha 410083,
People's Republic of China
e-mail: wangyong1226@gmail.com

Y. Zhou
School of Computer Science and Engineering,
South China University of Technology,
Guangzhou 516040, People's Republic of China

Z. Fan
Department of Management Engineering,
Technical University of Denmark,
Lyngby DK-2800, Denmark

Keywords Constrained optimization · Hybrid evolutionary algorithm · Constraint-handling technique

1 Introduction

In the real world, many optimization problems involve constraints. The general constrained optimization problem can be expressed as follows:

minimize $f(\vec{x})$, $\vec{x} = (x_1, x_2, \cdots, x_n) \in \Re^n$ subject to inequality constraints

$$g_j(\vec{x}) \leq 0, \quad j = 1, \cdots, l$$

and/or equality constraints

$$h_j(\vec{x}) = 0, \quad j = l+1, \cdots, m$$

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

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

where l_i and u_i are lower and upper bounds for a decision variable x_i, respectively.

For an inequality constraint that satisfies $g_j(\vec{x}) = 0$ ($j \in 1, \cdots, l$) at any point $\vec{x} \in \Omega$, we say it is *active* at $\vec{x}$. All equality constraints $h_j(\vec{x})$ ($j = l+1, \cdots, m$) are considered *active* at all points of Ω.

To deal with equality constraints, each equality constraint $h_j(\vec{x}) = 0$ ($j = l+1, \cdots, m$) is usually converted into a couple of inequality constraints: $h_j(\vec{x}) - \delta \leq 0$ and $-h_j(\vec{x}) - \delta \leq 0$ where δ is a small tolerant value. Thus, the original problem has $2m - l$ inequality constraints.

In general, constrained optimization problems are intractable, especially when the landscape of the objective function is very complex and the feasible region is concave and covers a very small part of the whole search space. Classical gradient-based optimization methods have difficulties to handle this kind of problems, as constrained optimization problems may usually lack an explicit mathematical formulation and have discrete definition domains. Compared with gradient-based optimization methods, evolutionary algorithms (EAs) are population-based global search techniques, not sensitive to the characteristics of the problems and easy to implement. During the past decade, using EAs to solve constrained optimization problems has attracted a lot of research interest, and a large number of constrained optimization evolutionary algorithms (COEAs) have been proposed. An extensive survey of COEAs can be found in Michalewicz and Schoenauer (1996) and Coello Coello (2002).

It is noteworthy that EAs are unconstrained search methods and lack an explicit mechanism to bias the search in constrained search space. This motivates the development of different constraint-handling techniques to cope with constraints (Mezura-Montes and Coello Coello 2005). Coello Coello (2002) classified most constraint-handling techniques into five categories: (1) methods based on preserving feasibility of solutions; (2) methods based on penalty functions; (3) methods making distinction between feasible and infeasible solutions; (4) methods based on decoders; and (5) hybrid methods.

While most constraint-handling techniques are modular, there are also constraint-handling techniques embedded as an integral part of EAs. In essence, COEAs can be considered as constraint-handling techniques plus EAs. Therefore, an effective constraint-handling technique needs to be in conjunction with an efficient EA to obtain competitive performance. Next, a number of the most representative COEAs will be briefly reviewed from the two important aspects, i.e., constraint-handling techniques and EAs. Note that according to the recent progress on constraint-handling techniques, this paper divides them into three categories: methods based on penalty functions, methods based on biasing feasible over infeasible solutions, and methods based on multi-objective optimization concepts.

The most common constraint-handling techniques are penalty-function-based methods because of their simplicity and ease of implementation. In these methods, the individual is penalized based on its constraint violation which is the sum of the violation of all constraints. Based on constraint violation, a penalty term can be constructed. Then, an extended objective function is defined by introducing the penalty term into the original objective function. The aim is to optimize the extended objective function. Farmani and Wright (2005) proposed a two-stage adaptive fitness formulation method. In the first penalty stage, the worst of the infeasible solutions has a penalized objective function value that is higher or equal to that of the best solution in the population. In the second penalty stage, the penalized objective function value of the worst infeasible individual is equal to that of the individual with maximum objective function value in the current population. The goal of the two-stage penalty is to ensure that slightly infeasible solutions with a low objective function value remain fit. The main advantage of this method is that it does not require any parameter tuning. Considering that it is difficult to set appropriate penalty factor for penalty function, Huang et al. (2007) proposed a co-evolutionary differential evolution for constrained optimization. Two populations are used in this method. The first population contains a set of penalty factors and is used to evolve decision solutions, while the second population consists of decision solutions and is employed to adapt penalty factors. These two populations evolve interactively and self-adaptively.

Another method for constraint-handling is to bias feasible over infeasible solutions. Deb (2000) proposed a pairwise comparison used in tournament selection which does not need any penalty parameter. In this method, when comparing pairwise individuals, (1) any feasible solution is preferred to any infeasible solution; (2) between two feasible solutions, the one with better objective function value is chosen; and (3) between two infeasible solutions, the one with smaller constraint violation is chosen. Taking into account the difficulty to determine the penalty parameter, Runarsson and Yao (2000) proposed a stochastic-rank-based approach. In this approach, a probability p_f is introduced. The probability p_f denotes the probability of using only the objective function to compare individuals in the infeasible region of the search space. That is to say, given pairwise individuals, the probability of comparing them according to the objective function is 1 if both individuals are feasible; otherwise, it is p_f. A p_f value of 0.45 is found to provide very good results. Takahama and Sakai (2005) proposed the α constrained method. In this method, a satisfaction level $\mu(\vec{x})$ for the constraints is introduced to indicate how well an individual $\vec{x}$ satisfies the constraints. Then, the α level comparison between individuals is conducted based on their level of satisfaction and the objective function. This method can convert an algorithm for unconstrained optimization

problems into an algorithm for constrained optimization problems by the α level comparison.

Recently, using multi-objective optimization concepts for constrained optimization has become a hot topic. The main characteristics of this kind of methods are twofold: (1) converting the original constrained optimization problems into unconstrained multi-objective optimization problems and (2) exploiting multi-objective optimization techniques to solve the converted problems. Three mechanisms taken from multi-objective optimization are frequently incorporated into constraint-handling techniques (Mezura-Montes and Coello Coello 2002): (1) using Pareto dominance as a selection criterion; (2) using Pareto ranking to assign fitness in such a way that non-dominated individuals are assigned a higher fitness value; and (3) splitting the population into subpopulations that are evaluated with respect to the objective function or with respect to a single constraint of the problem. Constrained optimization by multi-objective genetic algorithms proposed by Surry and Radcliffe (1997) views a constrained optimization problem as a constrained satisfaction problem by ignoring the objective function and as an unconstrained optimization problem by neglecting the constraints. In the former case, the individuals are evaluated by Pareto ranking. This method replaces a proportion p_{cost} of solutions based on fitness and the others based on Pareto ranking. The parameter p_{cost} is adjusted depending on the target proportion of the feasible solutions in the population. Zhou et al. (2003) transformed a constrained optimization problem into a bi-objective optimization problem. In this approach, Pareto strength value is defined for each individual based on Pareto dominance. Pareto strength value of an individual reflects the number of individuals in the population dominated by it. The comparisons between two individuals are firstly based on Pareto strength, if they share the same Pareto strength value, then comparisons occur based only on constraint violations. Venkatraman and Yen (2005) proposed a generic, two-phase framework for constrained optimization problems. In the first phase, the objective function is completely disregarded, and the constrained optimization problem is treated as a constrained satisfaction problem. In the second stage, the constrained optimization problem is considered as a bi-objective optimization problem by simultaneously optimizing the objective function and the constraints. Cai and Wang (2006) proposed a method in which non-dominated individuals are identified from the population over the course of evolution, and only one non-dominated individual is used to replace the corresponding dominated individual. In addition, an infeasible solution archiving and replacement mechanism is proposed, the main aim of which is to motivate the population toward feasible region promptly. Ray and Liew (2003) proposed a method based on the concept of society and civilization. In this method, a society refers to a set of individuals in the solution space, while a civilization is a collection of all such societies. At each generation, intra-society interaction between an average individual and its leader results in an improvement of an individual's performance; however, inter-society information exchange among leaders leads to the migration of leaders to more advanced societies. The identification of leaders in a society is the following: (1) if there are no feasible solutions, the leaders are the ones with constraint rank 1 and (2) if all the individuals are feasible, the leaders are the ones with objective rank less than average objective rank. Besides, the method (Coello Coello 2000a) based on Fonseca and Fleming's Pareto ranking process (Fonseca and Fleming 1999), the method (Coello Coello 2000b) based on population-based multi-objective technique such as VEGA (Schaffer 1985), the method (Coello Coello and Mezura-Montes 2002) based on the niched-Pareto genetic algorithm (Horn et al. 1994), and the method (Aguirre et al. 2004) based on the Pareto archived evolutionary strategy (Knowles and Corne 2000) are also presented for constrained optimization.

As previously mentioned, the search algorithm is another important aspect in constrained evolutionary optimization. Traditionally, EAs include three main branches, i.e., genetic algorithm (GA), evolutionary strategy (ES), and evolutionary programming (EP). These three branches have been extensively applied to cope with constrained optimization problems. During the past few years, some new members have been created and added into the community of EAs, such as differential evolution (DE), particle swarm optimization (PSO), cultural algorithm (CA), etc. Brest et al. (2006), Huang et al. (2006), and Mezura-Montes et al. (2006a) exploit DE, Dimopoulos (2007), Krohling and Coelho (2006), and Liang and Suganthan (2006) exploit PSO, and Coello Coello and Becerra (2004) and Becerra and Coello Coello (2006) exploit CA as search algorithms for constrained optimization. In addition, Wang et al. (2007a) proposed a hybrid EA for constrained optimization in which global and local search models are executed iteratively. In this method, the global search model is used to promote high population diversity, and the local search model intends to accelerate the convergence speed. Orthogonal design is also generalized into constrained optimization by Wang et al. (2007b).

In this paper, a hybrid EA and an adaptive constraint-handling technique are proposed for numerical and engineering constrained optimization problems. The hybrid EA incorporates simplex crossover and two mutation operators [diversity mutation and improved breeder GA (BGA) mutation] to generate the offspring population. In addition, the adaptive constraint-handling technique consists of three main situations, i.e., infeasible situation, semi-feasible situation, and feasible situation. Only one situation is applied at each generation according to whether all the individuals are infeasible, there are feasible and infeasible individuals, or all the individuals are feasible. Meanwhile, one constraint-handling mechanism is designed according to the characteristic of the current situation. The method is assessed on a total of seventeen optimization problems to verify its performance. The experimental results indicate that it is very robust and effective for solving constrained optimization problems. The main advantage of the method proposed is its ease of implementation.

Organization of the rest of this paper is as follows. Section 2 presents the details of the proposed method. Section 3 presents and analyzes the experimental results. Furthermore, we compare our method with respect to the state-of-the-art approaches in constrained evolutionary optimization using 13 benchmark test functions and four engineering design problems. The effectiveness of the genetic operators adopted in our approach is also shown in this section by different experiments. In addition, the effect of simplex crossover on performance is demonstrated. Section 4 concludes this paper.

2 Description of the proposed approach

2.1 Algorithm framework

In this paper, the degree of constraint violation of an individual $\vec{x}$ on the jth constraint is calculated using the following expression:

$$G_j(\vec{x}) = \begin{cases} \max\{0, g_j(\vec{x})\}, & 1 \le j \le l \\ \max\{0, |h_j(\vec{x})| - \delta\}, & l+1 \le j \le m \end{cases}. \quad (1)$$

where δ is a positive tolerance value for equality constraints. Then, $G(\vec{x}) = \sum_{j=1}^{m} G_j(\vec{x})$ reflects the degree of constraint violation of the individual $\vec{x}$.

At each generation, the proposed method, referred to as *hybrid evolutionary algorithm* and *adaptive constraint-handling technique* (abbreviated to HEA-ACT) hereafter, maintains: (1) a population of N individuals, i.e., $P_{(t)} = \{\vec{x}_1, \vec{x}_2, \cdots, \vec{x}_N\}$, and (2) their objective function values $f(\vec{x}_1), f(\vec{x}_2), \cdots, f(\vec{x}_N)$, and their degree of constraint violations $G(\vec{x}_1), G(\vec{x}_2), \cdots, G(\vec{x}_N)$.

The basic idea of HEA-ACT is that after the individuals in the parent population undergo crossover and mutation operations, the offspring population is obtained. Subsequently, some potential individuals in both the parent and offspring populations will be selected for the next population based on the adaptive constraint-handling technique. HEA-ACT works as follows:

Step 1. Initialization: Set $t := 0$. Randomly generate an initial population $P_{(0)}$ of size N from the decision space S. Evaluate the f value and G value for each individual in the population $P_{(0)}$.

Step 2. Reproduce: Generate new populations Q_1 and Q_2 by simplex crossover and mutation, respectively. It is worth noting that the mutation operators contain two components, i.e., diversity mutation and improved BGA mutation. An individual takes part in diversity mutation or improved BGA mutation with a probability of 0.5. Thus, no individual is subject to both mutation operations in the same generation. Note also that the crossover and mutation operations are applied in parallel rather than sequentially.

Step 3. Evaluation: Evaluate the f value and G value for each individual in $Q_1 \cup Q_2$.

Step 4. Selection: Select N individuals from $Q_1 \cup Q_2 \cup P_{(t)}$ to form the next population $P_{(t+1)}$ based on the adaptive constraint-handling technique that will be specified later.

Step 5. Set t: = t+1.

Step 6. Stopping criterion: If stopping criterion is met, stop and return the optimal solution in $P_{(t)}$, else go to step 2.

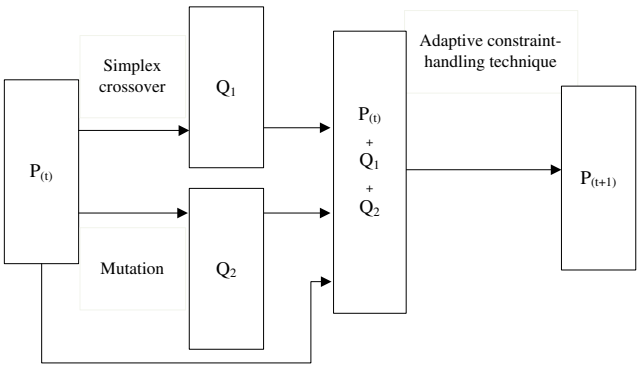

Fig. 1 HEA-ACT framework

The HEA-ACT procedure is also shown in Fig. 1. Next, we will discuss the implementation of reproduction and the adaptive constraint-handling technique of the above method in detail.

2.2 Reproduction

The reproduction procedure includes simplex crossover and two mutation operators, i.e., diversity mutation and improved BGA mutation.

2.2.1 Simplex crossover

Simplex crossover (Tsutsui et al. 1999) is a multi-parent combination operator for real-code genetic algorithm, which generates offspring based on uniform probability distribution and does not need any fitness information. In $\Re^n$, $n+1$ individuals that are independent of each other form a simplex. For simplicity, in a two-dimensional search space three individuals $\vec{x}_1$, $\vec{x}_2$, and $\vec{x}_3$ form a simplex (as shown in Fig. 2). We expand this simplex in each direction by $(1+\varepsilon)$ $(\varepsilon \geq 0)$ times. Let $\vec{o} = \frac{1}{3}\sum_{i=1}^{3}\vec{x}_i$ and $\vec{y}_i = (1+\varepsilon)(\vec{x}_i - \vec{o})$. Thus, $\vec{y}_1$, $\vec{y}_2$, and $\vec{y}_3$ constitute a new simplex. We then randomly choose an individual $\vec{z}$ from the new simplex, i.e., $\vec{z} = k_1\vec{y}_1 + k_2\vec{y}_2 + k_3\vec{y}_3 + \vec{o}$, where k_1, k_2, and k_3 are randomly selected within the range [0,1] and satisfy the condition $k_1 + k_2 + k_3 = 1$. Thus, we obtain an offspring $\vec{z}$ from these three parents $\vec{x}_1$, $\vec{x}_2$, and $\vec{x}_3$ by simplex crossover. This procedure for producing offspring can be generalized into n-dimensional search space.

In general, simplex crossover is specified as $SPX-\mu-\lambda-\varepsilon$, where μ is the number of parents chosen for crossover operation, λ is the number of the offspring created, and ε is a control parameter that defines the amplification rate.

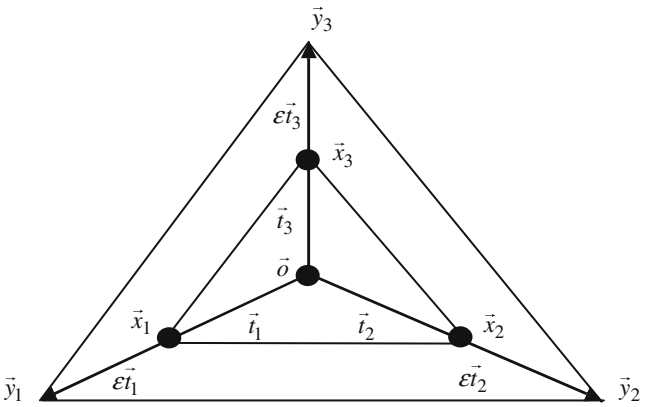

Fig. 2 SPX with three-parent in two-dimensional space

Simplex crossover can maintain a balance between exploration and exploitation (Tsutsui et al. 1999). At the early stage of the search, the variance among individuals randomly selected in the population is large. Therefore, the creation of several individuals from the chosen parents using simplex crossover results in widening the search region. On the other hand, at the later stage of the search the variance in the population is small, thereby ensuring a focused exploitation in the vicinity of the optimal solution using simplex crossover. In addition, the computational complexity of simplex crossover for producing an offspring is only $O(n)$.

2.2.2 Diversity mutation

To perform diversity mutation on a chosen individual $\vec{x} = (x_1, x_2, \cdots, x_n)$, randomly generate an integer i_{rand} between 1 and n with probability $1/n$ and a real number between l_i and u_i, and then replace the i_{rand}th component of the chosen individual by the real number to get a new chromosome $\vec{x}' = (x_1', x_2', \cdots, x_n')$. The above procedure can be denoted by the following expression:

$$x_i' = \begin{cases} l_i + \beta(u_i - l_i) & i = i_{\text{rand}} \\ x_i & \text{otherwise} \end{cases}, \quad i = 1, \cdots, n \tag{2}$$

where $\beta \in U(0,1)$ and $U(0,1)$ is a uniform random number generator in the range [0,1].

The purpose of this mutation operator is to facilitate the high diversity of the population, so it is called diversity mutation.

2.2.3 Improved BGA mutation

BGA mutation is proposed by Mühlenbein and Schlierkamp-Voosen (1993). Suppose $\vec{x} = (x_1, x_2, \cdots, x_n)$ is a chromosome and x_i is a variable to be mutated. A new value x_i' is computed according to:

$$x_i' = x_i \pm \text{rang}_i \cdot \alpha, \tag{3}$$

where rang_i defines the mutation range and it is normally set to $0.1 \cdot (u_i - l_i)$, the + or - sign is chosen with a probability of 0.5. In (3), α is set by the following expression

$$\alpha = \sum_{k=0}^{15} \alpha_k 2^{-k}, \tag{4}$$

where $\alpha_k \in \{0, 1\}$. Before BGA mutation is applied, we set each α_k equal to 0, and then each α_k is mutated to 1 with probability $p(\alpha_k = 1) = 1/16$. Only $\alpha_k = 1$ contributes to the sum. On average, there will be just one α_k with value 1, say α_j. Then α is fixed to 2^{-j}.

In this paper, BGA mutation has been improved by replacing the normal setting of rang_i (i.e., $0.1 \cdot (u_i - l_i)$) with the modified setting

$$(u_i - l_i) \cdot \text{rand}\big(0, (1 - \text{current\_gen}/\text{total\_gen})^7\big) \tag{5}$$

where current_gen denotes the current number of generation and total_gen denotes the total number of generation. The power "7" in (5) is set experimentally. The improved BGA mutation is called IBGA mutation.

To perform the IBGA mutation on a chosen individual $\vec{x} = (x_1, x_2, \cdots, x_n)$, randomly generate an index i_{rand} between 1 and n with probability $1/n$, and then replace the i_{rand}th variable of the chosen individual using (3), (4), and (5) to get a new chromosome $\vec{x}' = (x'_1, x'_2, \cdots, x'_n)$. This procedure can be expressed as follows:

$$x'_i = \begin{cases} x_i \pm (u_i - l_i) \cdot \text{rand}\big(0,(1\text{-current\_gen}/\text{total\_gen})^7\big) \times \sum_{k=0}^{15} \alpha_k 2^{-k}, & i = i_{\text{rand}} \\ x_i, & \text{otherwise} \end{cases}, \quad i = 1, \cdots, n \tag{6}$$

The IBGA mutation is intended to enhance the local search capability of the population, as in this operator, the probability of generating a neighborhood of $\vec{x}$ is very high. Moreover, the setting of rang_i is dynamically controlled by a nonlinear function. Note, however, that in the normal setting, rang_i is always kept to $0.1 \cdot (u_i - l_i)$, so the local tuning ability is not satisfactory.

Remark 1 The main highlight of simplex crossover is its capability to maintain a balance between exploration and exploitation. However, when the population is trapped in a local optimum, simplex crossover might not have the ability in helping the population to jump out of the local optimum, as simplex crossover only expands the simplex formed by several individuals with a constant amount. Especially when the individuals in the population are very similar, the global search ability of simplex crossover will drastically decrease. The diversity mutation has the global search capability since it can search the whole space. Nevertheless, its local search ability is not good enough. The goal of the IBGA mutation is to improve the capability of local search. Moreover, the dynamic setting of rang_i makes the population favor exploration at the early stage and exploitation at the later stage of the search. It is expected that combing these operators can enhance the performance of HEA-ACT concretely.

2.3 Adaptive constraint-handling technique

In general, the population will experience three different situations in constrained optimization and, consequently, we can design alternate constraint-handling mechanisms for different situations. When the population contains only infeasible solutions, it is called infeasible situation. When the population consists of a combination of feasible and infeasible solutions, it is called semi-feasible situation. And when the population is composed of only feasible solutions, it is called feasible situation. Next, the constraint-handling mechanisms will be introduced for these three situations step by step.

2.3.1 Infeasible situation

As in this situation there are no feasible solutions in the population, the constrained optimization problem under this condition can be treated as a constraint satisfaction problem. As a result, finding feasible solutions is the most important objective in this situation. To achieve this objective, we only concern the constraint violations $G(\vec{x})$ of the individuals in the population, and the objective function $f(\vec{x})$ is disregarded completely. Firstly, the individuals in the parent population are ranked based on their constraint violations in ascending order, and then some excellent individuals with the least constraint violations are selected and form the offspring population.

The above constraint-handling mechanism in the infeasible situation is very suitable for the highly constrained optimization problems in which the feasible region is very small compared with the entire search space and it is extremely difficult to find a feasible solution.

2.3.2 Semi-feasible situation

As for the semi-feasible situation, to keep the diversity and balance the exploration and exploitation ability of the population, a reasonable proportion between feasible and infeasible solutions in the population should be maintained, and some potential feasible and infeasible solutions should survive into the next generation. An adaptive constraint-handling mechanism is proposed to accomplish this goal.

Firstly, the population is divided into the feasible group K_1 and the infeasible group K_2. The best feasible

solution $\vec{x}_{\text{best}}$ and the worst feasible solution $\vec{x}_{\text{worst}}$ are identified from the feasible group K_1, respectively. Afterward, the objective function $f(\vec{x})$ of the population is converted into the following form (Wang et al. 2008):

$$f'(\vec{x}_i) = \begin{cases} f(\vec{x}_i), & \vec{x}_i \in K_1 \\ \max\left\{\varphi * f(\vec{x}_{\text{best}}) + (1-\varphi) * f(\vec{x}_{\text{worst}}),\ f(\vec{x}_i)\right\}, & \vec{x}_i \in K_2 \end{cases} \tag{7}$$

where φ denotes the proportion of feasible solutions in the last population $P_{(t)}$.

The objective function values of the individuals in the population are then normalized:

$$f_{\text{nor}}(\vec{x}_i) = \frac{f'(\vec{x}_i) - \min\limits_{\vec{x}\in K_1\cup K_2} f'(\vec{x})}{\max\limits_{\vec{x}\in K_1\cup K_2} f'(\vec{x}) - \min\limits_{\vec{x}\in K_1\cup K_2} f'(\vec{x})}, \vec{x}_i \in K_1 \cup K_2, \tag{8}$$

To scale the constraint violations to the same order of magnitude as the objective function, the constraint violations of the individuals in the population are normalized using the following expression:

$$G_{\text{nor}}(\vec{x}_i) = \begin{cases} 0, & \vec{x}_i \in K_1 \\ \dfrac{G(\vec{x}_i) - \min\limits_{\vec{x}\in K_2} G(\vec{x})}{\max\limits_{\vec{x}\in K_2} G(\vec{x}) - \min\limits_{\vec{x}\in K_2} G(\vec{x})}, & \vec{x}_i \in K_2 \end{cases}. \tag{9}$$

It is possible that a single infeasible solution exists in the population. Under this condition, the normalized constraint violation G_{nor} of such individual will always be equal to 0. To overcome this crash, the normalized constraint violation G_{nor} of such individual is set to a value uniformly chosen between 0 and 1.

Finally, the fitness function is obtained by adding the normalized objective function values and constraint violations together:

$$f_{\text{final}}(\vec{x}_i) = f_{\text{nor}}(\vec{x}_i) + G_{\text{nor}}(\vec{x}_i). \tag{10}$$

When selecting offspring for next generation in the semi-feasible situation, the individuals in the parent population are firstly ranked based on (10) in ascending order, and then some excellent individuals with the lowest $f_{\text{final}}(\vec{x})$ are selected for the offspring population.

It is noteworthy that (7) reflects the adaptive characteristic of the constraint-handling mechanism, as the parameter is adapted based on the feasibility proportion of the last population $P_{(t)}$ which may change from generation to generation. If the value of φ is large, the objective function values of the infeasible solution will be relatively smaller, thus increasing the probability for the infeasible solution to survive into the offspring population. In contrast, if the value of φ is small, the objective function values of the infeasible solution will be relatively larger, so the feasible solution will be selected with a higher probability. By the adaptive transformation for fitness function, the population may maintain a reasonable proportion between feasible and infeasible solutions, which, in turn, facilitates more effective search for the global optimal solution.

In addition, normalization for both the objective function and the constraint violations is necessary, as the objective function and the constraint violations are of different orders of magnitude originally.

2.3.3 Feasible situation

In this situation, the constrained optimization problem is equivalent to the unconstrained optimization problem, as the population only contains feasible solutions. With respect to the feasible situation, the comparisons of individuals are based solely on the objective function $f(\vec{x})$. Afterward, some excellent individuals with the least objective function values are selected for the next generation.

Remark 2 The proposed constraint-handling mechanisms in these three different situations constitute the adaptive constraint-handling technique for constrained optimization, which switches smoothly from the first situation to the third situation based on a simple conditional statement. In principle, the adaptive constraint-handling technique proposed in this paper belongs to the penalty function method. Apart from the feature of adaptation, the proposed technique is easy to understand and use. Furthermore, the computational time complexity is only $O(N\log(N))$. Compared with Wang et al. (2008), this paper adopts the same idea of converting the objective function values of feasible and infeasible individuals in the semi-feasible situation; however, the rest of the proposed approach is different.

3 Experimental study

3.1 Benchmark test functions

At first, 13 well-known benchmark test functions mentioned in Runarsson and Yao (2000) are optimized to inspect the performance of the proposed method. The main characteristics of these test cases are reported in Table 1. From Table 1, it is obvious that the test

Table 1 Main characteristics of 13 benchmark functions

Function	Number	Type of f	ρ (%)	LI	NE	NI	α
g01	13	Quadratic	0.0003	9	0	0	6
g02	20	Nonlinear	99.9965	1	0	1	1
g03	10	Nonlinear	0.0000	0	1	0	1
g04	5	Quadratic	26.9356	0	0	6	2
g05	4	Nonlinear	0.0000	2	3	0	3
g06	2	Nonlinear	0.0064	0	0	2	2
g07	10	Quadratic	0.0003	3	0	5	6
g08	2	Nonlinear	0.8640	0	0	2	0
g09	7	Nonlinear	0.5256	0	0	4	2
g10	8	Linear	0.0005	3	0	3	3
g11	2	Quadratic	0.0000	0	1	0	1
g12	3	Quadratic	0.0197	0	0	9^3	0
g13	5	Nonlinear	0.0000	0	3	0	3

functions include different types of objective function (e.g., linear, nonlinear, and quadratic) and constraints [e.g., linear inequality (LI), nonlinear equalities (NE), and nonlinear inequalities (NI)]. The feasibility ratio $\rho = |F \cap S|/|S|$ is determined experimentally by calculating the percentage of feasible solutions among 1,000,000 randomly generated individuals. Note that test functions g02, g03, g08, and g12 are maximization problems, and the others are minimization problems. In this study, the maximization problems are transformed into minimization using $-f(\vec{x})$. In addition, only test functions g03, g05, g11, and g13 contain equality constraints. For these problems, a dynamic setting of the parameter δ for equality constraints is adopted like Mezura-Montes and Coello Coello (2005). The parameter δ decreases from generation to generation using the following equation:

$$\delta_{(t+1)} = \begin{cases} \dfrac{\delta_{(t)}}{\delta'} & if\ \delta_{(t)} > 1E-10 \\ 1E-10 & \text{otherwise} \end{cases}, \tag{11}$$

where the initial $\delta_{(0)}$ is set to 5 and the value of δ' is set to 1.035.

The following parameters are established experimentally for the best performance of HEA-ACT: N = 60, μ, λ, and ε in simplex crossover are set to 10, 5, and 10, respectively. It is worth noticing that simplex crossover is executed 40 times in each generation, so the size of population Q_1 is equal to 200. When implementing simplex crossover at each time, the parents of size μ are randomly chosen from the population. In addition, as the diversity mutation or the IBGA mutation occurs with a probability of 0.5 for each individual in the population, the size of Q_2 is equal to 60. The number of fitness function evaluations (FFEs) is fixed to 200,000. The above parameter settings are kept for all experiments. In this paper, 30 independent runs are performed for each test function in MATLAB (the source code may be obtained from the authors upon request).

3.1.1 Experimental results

The statistical results of HEA-ACT are summarized in Table 2. The table shows the "known" optimal solution for each test function and the "best", "median", "mean", "worst", and standard deviations of the objective function values found.

As shown in Table 2, HEA-ACT is able to find the global optima consistently in 12 test functions over 30 runs with the exception of test function g02. With respect to test function g02, although the optimal solutions are not consistently found, the best result achieved is very close to the global optimal solution. The resulting solutions achieved for test function g02 have been exhibited in Fig. 3. It is noteworthy that the standard deviations in Table 2 are fairly small. In particular, the standard deviation for test functions g12 is equal to 0. Moreover, HEA-ACT is an efficient method, as the number of FFEs is 200,000. In addition, as the objective function and constraints are treated separately in HEA-ACT, it does not need to evaluate the objective function for the infeasible situation in the adaptive constraint-handling technique, which makes HEA-ACT more efficient. Finally, feasible solutions are consistently found for all test functions in 30 runs.

The above discussion validates that HEA-ACT is an effective and efficient approach for constrained optimization, and that it is capable of providing competitive results.

As discussed previously, HEA-ACT cannot consistently reach the optimal solution for test function g02. The main characteristic of this test function is that there are many local optima with high peak near the global

Table 2 Experimental results obtained by HEA-ACT for 13 benchmark test functions over 30 independent runs

Function	Optimal	Results of HEA-ACT				
		Best	Median	Mean	Worst	SD
g01	−15.000	−15.000	−15.000	−15.000	−15.000	7.7E-11
g02	−0.803619	−0.803582	−0.767844	−0.758182	−0.673096	3.2E-02
g03	−1.000	−1.000	−1.000	−1.000	−1.000	5.2E-15
g04	−30665.539	−30665.539	−30665.539	−30665.539	−30665.539	7.4E-12
g05	5126.498	5126.498	5126.498	5126.498	5126.498	9.3E-13
g06	−6961.814	−6961.814	−6961.814	−6961.814	−6961.814	4.6E-12
g07	24.306	24.306	24.306	24.306	24.306	1.9E-11
g08	−0.095825	−0.095825	−0.095825	−0.095825	−0.095825	2.8E-17
g09	680.630	680.630	680.630	680.630	680.630	5.8E-13
g10	7049.248	7049.248	7049.248	7049.248	7049.248	1.4E-05
g11	0.750	0.750	0.750	0.750	0.750	3.4E-16
g12	−1.000	−1.000	−1.000	−1.000	−1.000	0.0E+00
g13	0.0539498	0.0539498	0.0539498	0.0539498	0.0539498	1.5E-15

optimal solution. It is commonly accepted that when solving multi-modal problems (such as test function g02), the diversity of the population has a significant effect on finding the optimal solution, as the population may converge to a local optimum if the diversity of the population is not good enough. In general, selection pressure has a direct relationship with the diversity of the population. If the selection pressure is high, the diversity of the population may decrease rapidly. It is necessary to note that in HEA-ACT, the selection strategy is similar to $(\mu + \lambda)$-evolutionary strategy, and therefore, the selection pressure of the population is relatively high. Another experiment (denoted as HEA-ACT_1) has been done to show the change of the results for 13 test functions when the selection pressure of the population decreases. For HEA-ACT_1, instead of using the whole population $P_{(t)}$, only five best individuals chosen from $P_{(t)}$ are combined with $Q_1 \cup Q_2$ for selecting the next population in the step 4 of Section 2.1. The comparisons of results between HEA-ACT and HEA-ACT_1 have been shown in Table 3. It can be seen from Table 3 that the results derived from HEA-ACT_1 for test function g02 are of a much higher quality than HEA-ACT. Moreover, the global optimal solution has been found by HEA-ACT_1 for test function g02. However, the lower selection pressure causes the performance degradation for some test functions, such as g01, g07, g10, and g13. More importantly, the feasible solution cannot be found consistently over 30 runs for test function g13. The analyses above indicate that different test functions may need different selection pressure, for instance, for test function g02, the lower selection pressure is beneficial, but for test functions g01, g07, g10, and g13, the higher selection pressure is beneficial. This phenomenon is in agreement with the no-free lunch theorem (Wolpert and Macready 1997). Based on the above discussion, how to dynamically choose the selection pressure for different test functions will be part of our future work.

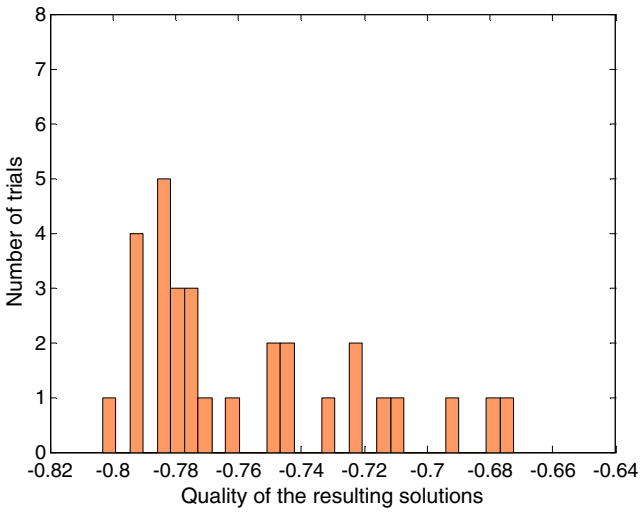

Fig. 3 Plots show the number of trials versus the quality of the resulting solutions achieved for test function g02

3.1.2 Comparison with the α simplex and the CDE

In the experiments, HEA-ACT is compared with α Simplex (Takahama and Sakai 2005), which has been introduced in Section 1. In α Simplex, 30 independent runs are performed, the population size N is 90, the mutation rate $P_m = 0.06$, the maximum number of FFEs is 30,000 for test function g12 and from 290,000 to 330,000 for the other test functions, and all equality constraints

Table 3 Experimental comparison between HEA-ACT and HEA-ACT_1 on 13 test functions over 30 independent runs

Function	Method	Best	Median	Mean	Worst	SD
g01	HEA-ACT	−15.000	−15.000	−15.000	−15.000	7.7E-11
	HEA-ACT_1	−15.000	−15.000	−14.998	−14.966	6.0E-03
g02	HEA-ACT	−0.803582	−0.767844	−0.758182	−0.673096	3.2E-02
	HEA-ACT_1	−0.803619	−0.794896	−0.794358	−0.730711	1.4E-02
g03	HEA-ACT	−1.000	−1.000	−1.000	−1.000	5.2E-15
	HEA-ACT_1	−1.000	−1.000	−1.000	−1.000	4.3E-06
g04	HEA-ACT	−30,665.539	−30,665.539	−30,665.539	−30,665.539	7.4E-12
	HEA-ACT_1	−30,665.539	−30,665.539	−30,665.539	−30,665.539	7.4E-12
g05	HEA-ACT	5,126.498	5,126.498	5,126.498	5,126.498	9.3E-13
	HEA-ACT_1	5,126.498	5,126.498	5,126.498	5,126.498	3.2E-11
g06	HEA-ACT	−6,961.814	−6,961.814	−6,961.814	−6,961.814	4.6E-12
	HEA-ACT_1	−6,961.814	−6,961.814	−6,961.814	−6,961.814	4.6E-12
g07	HEA-ACT	24.306	24.306	24.306	24.306	1.9E-11
	HEA-ACT_1	25.225	26.366	26.497	27.992	7.8E-01
g08	HEA-ACT	−0.095825	−0.095825	−0.095825	−0.095825	2.8E-17
	HEA-ACT_1	−0.095825	−0.095825	−0.095825	−0.095825	2.8E-17
g09	HEA-ACT	680.630	680.630	680.630	680.630	5.8E-13
	HEA-ACT_1	680.630	680.630	680.630	680.630	7.2E-09
g10	HEA-ACT	7,049.248	7,049.248	7,049.248	7,049.248	1.4E-05
	HEA-ACT_1	7,175.103	7,378.483	7,400.906	7,751.922	1.6E+02
g11	HEA-ACT	0.750	0.750	0.750	0.750	3.4E-16
	HEA-ACT_1	0.750	0.750	0.750	0.750	3.4E-16
g12	HEA-ACT	−1.000	−1.000	−1.000	−1.000	0.0E+00
	HEA-ACT_1	−1.000	−1.000	−1.000	−1.000	0.0E+00
g13	HEA-ACT	0.0539498	0.0539498	0.0539498	0.0539498	0.0E+00
	HEA-ACT_1	(11)[a]				

[a]Denotes the number of trials that feasible solutions are found.

are relaxed using $\delta = 1E-4$. In addition, the α level is controlled based on the following equation:

$$\alpha_{(t)} = \begin{cases} \frac{1}{2}\left(\max_i \mu(\vec{x}_i) + \frac{\sum_i \mu(\vec{x}_i)}{N}\right), & if \quad t = 0 \\ (1-\beta)\alpha_{(t-1)} + \beta, & if\ 0 < t < \frac{T_{\max}}{2} \\ & and\ (T \bmod T_\alpha) = 0 \\ \alpha_{(t-1)}, & if\ 0 < t < \frac{T_{\max}}{2} \\ & and\ (T \bmod T_\alpha) \neq 0 \\ 1, & if\ t > \frac{T_{\max}}{2} \end{cases} \tag{12}$$

where $\mu(\vec{x}_i)$ denotes the satisfaction level of individual $\vec{x}_i$ which indicate how well the individual $\vec{x}_i$ satisfies the constraints, $\beta = 0.3$, $T_\alpha = 50$, the maximum iterations $T_{\max} = 8{,}500$ for test function g12 and for the other test functions, $T_{\max} = 85{,}000$ is used.

HEA-ACT is also compared with cultural differential evolution (CDE; Becerra and Coello Coello 2006). CDE combines cultural algorithm with differential evolution to solve constrained optimization problems. Cultural algorithms involve two main components: the population space and the belief space. The population space consists of a set of possible solutions, and the belief space is the information repository. Both spaces are linked through a communication protocol. In this method, the cultural algorithm is used to extract knowledge from the population during the evolutionary process and to accelerate the convergence, and the population space is modeled using differential evolution. For each test function, 30 independent runs are performed with 100,100 FFEs, the population size is 100, the maximum number of generations is 1,000, $F = 0.5$, $CR = 1$, the maximum depth of k-d tree is 12, the length of best cell list is the number of decision variables of the problem, the size of the list in the history knowledge $w = 5$, α and β can be fixed to 0.4 or 0.45, and the percentage of accepted individuals at the end of the evolutionary process $\%p = 0.2$.

The results have been shown in Tables 4 and 5. Compared with α Simplex, HEA-ACT finds similar "best", "mean", and "worst" results for ten test functions (g01, g03, g04, g05, g06, g08, g09, g10, g11, and g12). For test function g02, better "best", "mean," and "worst" results are found by α Simplex. The "worst" result provided

Table 4 Comparing HEA-ACT with respect to α Simplex on 13 benchmark test functions

Function	Optimal	Best result		Mean result		Worst result	
		HEA-ACT	α Simplex	HEA-ACT	α Simplex	HEA-ACT	α Simplex
g01	−15.000	−15.000	−15.000	−15.000	−15.000	−15.000	−15.000
g02	−0.803619	−0.803582	**−0.803619**	−0.758182	**−0.784187**	−0.673096	**−0.754259**
g03	−1.000	−1.000	−1.001	−1.000	−1.001	−1.000	−1.001
g04	−30,665.539	−30,665.539	−30,665.539	−30,665.539	−30,665.539	−30,665.539	−30,665.539
g05	5,126.498	5,126.498	5,126.497	5,126.498	5,126.497	5,126.498	5,126.497
g06	−6,961.814	−6,961.814	−6,961.814	−6,961.814	−6,961.814	−6,961.814	−6,961.814
g07	24.306	24.306	24.306	24.306	24.306	**24.306**	24.307
g08	−0.095825	−0.095825	−0.095825	−0.095825	−0.095825	−0.095825	−0.095825
g09	680.630	680.630	680.630	680.630	680.630	680.630	680.630
g10	7,049.248	7,049.248	7,049.248	7,049.248	7,049.248	7,049.248	7,049.248
g11	0.750	0.750	0.750	0.750	0.750	0.750	0.750
g12	−1.000	−1.000	−1.000	−1.000	−1.000	−1.000	−1.000
g13	0.0539498	0.0539498	0.053942	**0.0539498**	0.066770	**0.0539498**	0.438803

Result in boldface indicates that a better result is reached.

by HEA-ACT is of a higher quality for test function g07. In addition, HEA-ACT finds better "mean" and "worst" results for test function g13.

With respect to CDE, HEA-ACT provides similar "best", "mean", and "worst" results for eight test functions (g01, g04, g06, g07, g08, g09, g10, and g12). A better "best" result is found by CDE in test function g02; however, HEA-ACT reaches better "mean" and "worst" results. HEA-ACT finds better "best", "mean" and "worst" results for three test functions (g03, g05 and g13). Finally, HEA-ACT finds better "mean" and "worst" results for test function g11. It can be observed that with regard to CDE, premature convergence tends to occur for constrained optimization problems with equality constraints (g03, g05, g11, and g13).

It is very difficult to solve constrained optimization problems with equality constraints without relaxing the equality constraints. Nevertheless, the transformation from equality constraints into inequality constraints may have an impact on performance of the method. For example, as the transformation of equality constraints, the results provided by α Simplex for test functions g03, g05, and g13 are better than the "known" optima. This does not mean that "new" optima have been found by α Simplex. It is necessary to notice that for four test functions with equality constraints (g03, g05, g11, and

Table 5 Comparing HEA-ACT with respect to CDE on 13 benchmark test functions

Function	Optimal	Best result		Mean result		Worst result	
		HEA-ACT	CDE	HEA-ACT	CDE	HEA-ACT	CDE
g01	−15.000	−15.000	−15.000	−15.000	−15.000	−15.000	−15.000
g02	−0.803619	−0.803582	**−0.803619**	**−0.758182**	−0.724886	**−0.673096**	−0.590908
g03	−1.000	**−1.000**	−0.995	**−1.000**	−0.789	**−1.000**	−0.640
g04	−30,665.539	−30,665.539	−30,665.539	−30,665.539	−30,665.539	−30,665.539	−30,665.539
g05	5,126.498	**5,126.498**	5,126.571	**5,126.498**	5,207.411	**5,126.498**	5,327.390
g06	−6,961.814	−6,961.814	−6,961.814	−6,961.814	−6,961.814	−6,961.814	−6,961.814
g07	24.306	24.306	24.306	24.306	24.306	24.306	24.306
g08	−0.095825	−0.095825	−0.095825	−0.095825	−0.095825	−0.095825	−0.095825
g09	680.630	680.630	680.630	680.630	680.630	680.630	680.630
g10	7,049.248	7,049.248	7,049.248	7,049.248	7,049.248	7,049.248	7,049.248
g11	0.750	0.750	0.750	**0.750**	0.758	**0.750**	0.796
g12	−1.000	−1.000	−1.000	−1.000	−1.000	−1.000	−1.000
g13	0.0539498	**0.0539498**	0.056180	**0.0539498**	0.288324	**0.0539498**	0.392100

Result in boldface indicates that a better result is reached.

Table 6 Mean results obtained by HEA-ACT, HEA-ACT with only simplex crossover (version 1), HEA-ACT with simplex crossover and diversity mutation (version 2), and HEA-ACT with simplex crossover and IBGA mutation (version 3) for 13 benchmark test functions over 30 independent runs

Function	HEA-ACT	Version 1	Version 2	Version 3
g01	**−15.000**	(16)[a]	−14.981	**−15.000**
g02	−0.758182	−0.530551	**−0.789723**	−0.699993
g03	−1.000	−1.000	−1.000	−1.000
g04	**−30,665.539**	−30,661.841	−30,665.300	**−30,665.539**
g05	5,126.498	5,126.498	5,126.498	5,126.498
g06	−6,961.814	−6,961.814	−6,961.814	−6,961.814
g07	24.306	24.306	24.306	24.306
g08	−0.095825	−0.095825	−0.095825	−0.095825
g09	680.630	680.630	680.630	680.630
g10	**7,049.248**	8,910.618	7,151.997	7,049.252
g11	**0.750**	**0.750**	(28)[a]	**0.750**
g12	−1.000	−1.000	−1.000	−1.000
g13	**0.0539498**	(0)[a]	(18)[a]	0.0667799

Result in boldface indicates that a better result is reached.
[a]Denotes the number of trials that feasible solutions are found

g13), the results obtained by HEA-ACT are almost equal to the "known" optima. It is because the value of δ is extremely small ($1E-10$) when the stopping criterion is met.

In addition, regarding the computation cost (measured by the number of FFEs), CDE is the most efficient method, as it has the minimum computational cost (100,100 FFEs), and HEA-ACT has the median computation cost (200,000 FFEs).

As a general remark on the comparison above, HEA-ACT shows a very competitive performance with respect to two state-of-the-art approaches in terms of the quality, the robustness, and the efficiency of search. Concretely, HEA-ACT exhibits a good trade-off between effectiveness and efficiency.

3.1.3 Effectiveness of the genetic operators in HEA-ACT

In this subsection, some trials have been performed to study the effectiveness of the genetic operators adopted in HEA-ACT (i.e., simplex crossover, diversity mutation, and IBGA mutation) by using some operators separately. Three different versions of HEA-ACT have been tested:

1. Version 1: HEA-ACT with only simplex crossover;
2. Version 2: HEA-ACT with simplex crossover and diversity mutation;
3. Version 3: HEA-ACT with simplex crossover and IBGA mutation.

Table 7 Mean results obtained by HEA-ACT with two-point crossover, HEA-ACT with BLX-α crossover, and HEA-ACT with simplex crossover for 13 benchmark test functions over 30 independent runs

Function	HEA-ACT with two-point crossover	HEA-ACT with BLX-α	HEA-ACT with simplex crossover
g01	−14.633	**−15.000**	**−15.000**
g02	−0.616297	−0.723299	**−0.758182**
g03	−0.991	−0.999	**−1.000**
g04	−30,396.177	−30,817.915	**−30,665.539**
g05	(0)[a]	(0)[a]	**5,126.498**
g06	(28)[a]	−6,961.788	**−6,961.814**
g07	37.157	26.674	**24.306**
g08	−0.093602	**−0.095825**	**−0.095825**
g09	691.688	682.137	**680.630**
g10	9,164.792	8,704.273	**7,049.248**
g11	0.755	0.756	**0.750**
g12	−1.000	−1.000	−1.000
g13	(27)[a]	(26)[a]	**0.0539498**

Result in boldface indicates that a better result is reached.
[a]Denotes the number of trials that feasible solutions are found

Table 8 Mean results obtained by HEA-ACT with varying μ in simplex crossover over 30 independent runs

Function	6	8	10	12	14
g01	−15.000	−15.000	−15.000	−15.000	−15.000
g02	−0.755016	−0.755971	**−0.758182**	−0.744841	−0.746268
g03	−1.000	−1.000	−1.000	−1.000	−1.000
g04	−30,665.539	−30,665.539	−30,665.539	−30,665.539	−30,665.539
g05	**5,126.498**	**5,126.498**	**5,126.498**	**5,126.498**	(29)[a]
g06	−6,961.814	−6,961.814	−6,961.814	−6,961.814	−6,961.814
g07	24.306	24.306	24.306	24.306	24.306
g08	−0.095825	−0.095825	−0.095825	−0.095825	−0.095825
g09	680.630	680.630	680.630	680.630	680.630
g10	7,049.248	7,049.248	7,049.248	7,049.248	7,049.248
g11	0.750	0.750	0.750	0.750	0.750
g12	−1.000	−1.000	−1.000	−1.000	−1.000
g13	**0.0539498**	**0.0539498**	**0.0539498**	**0.0539498**	0.0667798

Result in boldface indicates that a better result is reached
[a]Denotes the number of trials that feasible solutions are found

The parameters used in these three versions are exactly the same as those used in the experiments described at the beginning of Section 3.1. Furthermore, the number of FFEs is also the same to have a fair comparison. The experimental results have been recorded in Table 6.

Table 6 shows that compared with HEA-ACT, the performance deterioration takes place in test functions g02, g04, and g10 for version 1. Moreover, version 1 cannot find feasible solutions consistently for test functions g01 and g13. Although version 2 provides results of a higher quality for test function g02, it degrades its performance for test functions g01, g04, and g10. More importantly, feasible solutions cannot be found consistently for test functions g11 and g13. There exists a negative effect when using version 3. For instance, the results of test functions g02, g10, and g13 are worse than those provided by HEA-ACT. This may be because the diversity of the population is not as good as expected.

Based on the analyses above, it can be concluded that HEA-ACT outperforms these three versions on the whole, which also verifies that simultaneously employing these three genetic operators increases the chance for improving the search performance.

3.1.4 Discussion about simplex crossover

To ascertain whether simplex crossover is really suitable for the framework proposed, the proposed approach is compared with the same approach, but

Table 9 Mean results obtained by HEA-ACT with varying λ in simplex crossover over 30 independent runs

Function	1	3	5	7	9
g01	−15.000	−15.000	−15.000	−15.000	−15.000
g02	−0.740535	−0.750761	−0.758182	**−0.762383**	−0.755297
g03	**−1.000**	**−1.000**	**−1.000**	(21)[a]	(19)[a]
g04	−30,665.539	−30,665.539	−30,665.539	−30,665.539	−30,665.539
g05	**5,126.498**	**5,126.498**	**5,126.498**	(27)[a]	(26)[a]
g06	−6,961.814	−6,961.814	−6,961.814	−6,961.814	−6,961.814
g07	24.306	24.306	24.306	24.306	24.306
g08	−0.095825	−0.095825	−0.095825	−0.095825	−0.095825
g09	680.630	680.630	680.630	680.630	680.630
g10	7,049.248	7,049.248	7,049.248	7,049.248	7,049.248
g11	0.750	0.750	0.750	0.750	0.750
g12	−1.000	−1.000	−1.000	−1.000	−1.000
g13	0.105270	**0.0539498**	**0.0539498**	**0.0539498**	**0.0539498**

Result in boldface indicates that a better result is reached.
[a]Denotes the number of trials that feasible solutions are found

Table 10 Mean results obtained by HEA-ACT with varying ε in simplex crossover over 30 independent runs

Function	6	8	10	12	14
g01	−14.933	**−15.000**	**−15.000**	**−15.000**	**−15.000**
g02	−0.680891	−0.735283	−0.758182	−0.753172	**−0.760076**
g03	−1.000	−1.000	−1.000	−1.000	−1.000
g04	−30,619.116	**−30,665.539**	**−30,665.539**	**−30,665.539**	**−30,665.539**
g05	(8)[a]	(29)[a]	**5,126.498**	**5,126.498**	**5,126.498**
g06	−6,952.819	**−6,961.814**	**−6,961.814**	**−6,961.814**	**−6,961.814**
g07	26.728	**24.306**	**24.306**	**24.306**	**24.306**
g08	−0.095825	−0.095825	−0.095825	−0.095825	−0.095825
g09	681.604	**680.630**	**680.630**	**680.630**	**680.630**
g10	7,148.091	**7,049.248**	**7,049.248**	**7,049.248**	7,050.864
g11	0.750	0.750	0.750	0.750	0.750
g12	−1.000	−1.000	−1.000	−1.000	−1.000
g13	0.0539587	**0.0539498**	**0.0539498**	**0.0539498**	**0.0539498**

Result in boldface indicates that a better result is reached.
[a]Denotes the number of trials that feasible solutions are found

replacing simplex crossover with two traditional ones, i.e., two-point crossover and BLX-α crossover (Eshelman and Schaffer 1993). The compared approaches have the same number of FFEs (i.e., 200,000 FFEs) to have a fair comparison. In BLX-α crossover, the parameter α is fixed to 0.5, which is used as a standard value in Eshelman and Schaffer (1993). The mean results obtained by three different experiments, HEA-ACT with two-point crossover, HEA-ACT with BLX-α crossover, and HEA-ACT with simplex crossover, have been shown in Table 7. As described in Table 7, the best overall performance is exhibited by HEA-ACT with simplex crossover, followed by HEA-ACT with BLX-α crossover, and finally, HEA-ACT with two-point crossover seems to have the worst overall performance. HEA-ACT with two-point crossover and HEA-ACT with BLX-α crossover can only consistently hit the global optima for test function g12 and for test functions g01, g08, and g12, respectively. Furthermore, HEA-ACT with two-point crossover and HEA-ACT with BLX-α crossover cannot consistently find feasible solutions for test functions g05, g06, and g13 and for test functions g05 and g13, respectively. The results above indicate that on the whole, HEA-ACT has more stable performance and is more suitable for the hybrid framework proposed than the other two crossover operators.

It is important to note that in this paper, the constraint-handling technique and both mutation

Table 11 Mean results obtained by HEA-ACT with different number of simplex crossover performed per generation over 30 independent runs

Function	20	30	40	50	60
g01	−14.933	**−15.000**	**−15.000**	**−15.000**	**−15.000**
g02	−0.746141	−0.746763	−0.758182	**−0.763888**	−0.760410
g03	**−1.000**	**−1.000**	**−1.000**	(18)[a]	(18)[a]
g04	−30,665.539	−30,665.539	−30,665.539	−30,665.539	−30,665.539
g05	**5,126.498**	**5,126.498**	**5,126.498**	(29)[a]	(27)[a]
g06	−6,961.814	−6,961.814	−6,961.814	−6,961.814	−6,961.814
g07	24.306	24.306	24.306	24.306	24.306
g08	−0.095825	−0.095825	−0.095825	−0.095825	−0.095825
g09	680.630	680.630	680.630	680.630	680.630
g10	7,049.248	7,049.248	7,049.248	7,049.248	7,049.248
g11	0.750	0.750	0.750	0.750	0.750
g12	−1.000	−1.000	−1.000	−1.000	−1.000
g13	0.0667799	**0.0539498**	**0.0539498**	(29)[a]	**0.0539498**

Result in boldface indicates that a better result is reached
[a]Denotes the number of trials that feasible solutions are found.

Table 12 Experimental results obtained by HEA-ACT for four engineering optimization problems over 30 independent runs

Problem	Results of HEA-ACT				
	Best	Median	Mean	Worst	SD
Welded beam design	2.38095723	2.38096560	2.38097103	2.38102095	1.3E-05
Spring design	0.012665233	0.012665234	0.012665234	0.012665240	1.4E-09
Speed reducer design	2,994.499107	2,994.599748	2,994.613368	2,994.752311	7.0E-02
Three-bar truss design	263.895843	263.895848	263.895865	263.896099	4.9E-05

operators are parameter-free. Hence, simplex crossover is responsible for the algorithm not to be completely adaptive. Next, four additional experiments have been performed to illustrate the effect of the four parameters used by simplex crossover (i.e., the number of parents μ, the number of offspring λ, the control parameter ε, and the number of times crossover is performed per generation) on the approach's performance. The mean results obtained by four different experiments have been summarized in Tables 8, 9, 10, 11.

Table 8 indicates that increasing the parameter μ causes convergence instability. For instance, the algorithm cannot consistently offer feasible solutions for test function g05 when $\mu = 14$. Moreover, in this case, premature convergence occurs for test function g13. These are because a large number of parents may result in the rapid loss of population diversity. Tsutsui et al. (1999) pointed out that a large number of parents have sampling biases that reflect biases in the population distribution too much. Recently, Noman and Iba (2008) used simplex crossover with three parents to improve the local search ability of differential evolution. Their method shows high performance. From the results of Table 8, we can induce that a value of μ between 6 and 12 is an appropriate setting for HEA-ACT.

For Table 9, in the cases of $\lambda = 1$, the results for test functions g01 and g13 are much worse than other results. In addition, in the case of $\lambda = 7$ and 9, the methods cannot converge to the feasible solutions consistently. The above behaviors are not difficult to understand, as the exploration and exploitation ability of the population is poor if the value of λ is smaller (such as 1); meanwhile, a higher value of λ (such as 7 and 9) means a lower number of iterations, which leads to an incomplete convergence of the population. These results encourage the use of a value of λ between 3 and 5 for HEA-ACT.

The control parameter ε defines the amplification rate of simplex crossover and adjusts the exploration and exploitation ability of the population. If the control parameter ε is smaller, the exploration ability of the population is not good; on the contrary, if the control parameter ε is bigger, the exploitation capability is poor, so a suitable value should be chosen for this parameter. The results in Table 10 have verified the above analysis. In the case of $\varepsilon = 6$, the results are of a much worse quality compared with other algorithms. In the case of $\varepsilon = 8$, the quality of results is better than those provided by $\varepsilon = 6$; however, it still shows poor performance for test functions g02 and g05. In the case of $\varepsilon = 14$, the result for test function g10 is worse than that of $\varepsilon = 8$, 10, and 12. Thus, a value of ε between 10 and 12 is appropriate for HEA-ACT.

The number that crossover is executed per generation also has a significant impact on performance. If the number of crossover executed per generation is less, the competence of simplex crossover cannot be exhibited completely. However, if the number of crossover executed per generation is more, the number of FFEs per generation will increase remarkably, and hence, the total iteration number of the population will decrease accordingly. The results in Table 11 also

Table 13 Results of welded beam design

Method	Best	Median	Mean	Worst	SD	The number of FFEs
HEA-ACT	2.38095723	2.38096560	2.38097103	2.38102095	1.3E-05	30,000
Ray and Liew (2003)	2.3854347	3.0025883	3.2551371	6.3996785	9.6E-01	33,095
FSA (Hedar and Fukushima 2006)	2.381065	NA	2.404166	2.488967	NA	56,243
Deb (2000)	2.38119	2.39289	NA	2.64583	NA	40,080

NA not available

Table 14 Comparison of results for welded beam design

	HEA-ACT	Ray and Liew (2003)	FSA (Hedar and Fukushima 2006)	Deb (2000)
x_1	0.2443688943	0.2444382760	0.24435257	NA
x_2	6.2175179741	6.2379672340	6.2157922	NA
x_3	8.2914773014	8.2885761430	8.2939046	NA
x_4	0.2443689510	0.2445661820	0.24435258	NA
Best	2.38095723	2.3854347	2.381065	2.38119

NA not available

Table 15 Results of spring design

Method	Best	Median	Mean	Worst	SD	The number of FFEs
HEA-ACT	0.012665233	0.012665234	0.012665234	0.012665240	1.4E-09	24,000
Ray and Liew (2003)	0.012669249	0.012922669	0.012922669	0.016717272	5.9E-04	25,167
FSA (Hedar and Fukushima 2006)	0.012665258	NA	0.012665299	0.012665338	2.2E-08	49,531

NA not available

Table 16 Comparison of results for spring design

	HEA-ACT	Ray and Liew (2003)	FSA (Hedar and Fukushima 2006)
x_1	0.3567292035	0.368158695	0.3580047835
x_2	0.0516895376	0.0521602170	0.0517425034
x_3	11.2882937035	10.6484422590	11.2139073628
Best	0.012665233	0.012669249	0.012665258

Table 17 Results of speed reducer design

Method	Best	Median	Mean	Worst	SD	The number of FFEs
HEA-ACT	2,994.499107	2,994.599748	2,994.613368	2,994.752311	7.0E-02	40,000
Ray and Liew (2003)	2,994.744241	3,001.758264	3,001.758264	3,009.964736	4.0E+00	54,456
Mezura-Montes et al. (2006a)	2,996.356689	NA	2,996.367220	2,996.390137	8.2E-03	24,000
Akhtar et al. (2002)	3,008.08	NA	3,012.12	3,028.28	NA	19,154

NA not available

Table 18 Comparison of results for speed reducer design

	HEA-ACT	Ray and Liew (2003)	Mezura-Montes et al. (2006a)	Akhtar et al. (2002)
x_1	3.5000228993	3.50000681	3.500010	3.506122
x_2	0.7000003924	0.70000001	0.700000	0.700006
x_3	17.0000128592	17	17	17
x_4	7.3004277414	7.32760205	7.300156	7.549126
x_5	7.7153774494	7.71532175	7.800027	7.859330
x_6	3.3502309666	3.35026702	3.350221	3.365576
x_7	5.2866636970	5.28665450	5.286685	5.289773
Best	2,994.499107	2,994.744241	2,996.356689	3,008.08

Table 19 Results of three-bar truss design

Method	Best	Median	Mean	Worst	SD	The number of FFEs
HEA-ACT	263.895843	263.895848	263.895865	263.896099	4.9E-05	15,000
Ray and Liew (2003)	263.8958466	263.8989	263.9033	263.96975	1.3E-02	17,610

demonstrate the above analyses. In the case of the number of crossover executed per generation being equal to 20, the results for test functions g01, g02, and g13 show slight performance degradation. In the case of the number of crossover executed per generation being equal to 50 and 60, for some test functions, the population is unable to enter the feasible region consistently. Therefore, a value of the number of crossover executed per generation between 30 and 40 is suitable for HEA-ACT.

3.2 Engineering optimization problems

For studying the performance of HEA-ACT on real-world engineering constrained optimization problems, four well-studied engineering design examples chosen from Ray and Liew (2003) are solved using this approach. All parameter settings are the same as the previous experiments for 13 benchmark test functions except for the number of FFEs. The number of FFEs for these four engineering optimization problems is 30,000, 24,000, 40,000, and 15,000, respectively.

The simulation results are shown in Table 12. As described in Table 12, in terms of the selected performance measures, it can be seen that the robutstness of HEA-ACT is very good, as it can converge to similar results with a very small number of FFEs. In particular, HEA-ACT can almost reach the same result for spring design problem.

3.2.1 Welded beam design problem

The approaches applied to this problem for comparison include Ray and Liew (2003), fast simulated annealing (FSA; Hedar and Fukushima 2006), and Deb (2000). Their results are shown in Table 13, and the best solutions obtained by the above approaches and HEA-ACT are listed in Table 14. With respect to HEA-ACT, the constraints are [–0.001583 –0.039724 –0.000000 –0.234241 –0.001140] for the best result obtained.

From Table 13, it is clear that the best, median, mean, and worst results provided by HEA-ACT are better than those found by other methods. Note that even the worst result found by HEA-ACT is better than the best results found by other methods. Moreover, HEA-ACT is more efficient than other methods in terms of the number of FFEs.

3.2.2 Spring design problem

The approaches applied to this problem for comparisons include Ray and Liew (2003) and FSA (Hedar and Fukushima 2006). Their results are shown in Table 15, and the best solutions obtained by the above approaches and HEA-ACT are listed in Table 16. The constraints are [0.00000 –0.000000 –4.053808 –0.727721] based on the best result derived from HEA-ACT.

From Table 15, it can be seen that HEA-ACT has performed with more robustness in terms of the quality of results obtained and more efficiency in terms of the number of FFEs. In addition, even the worst result found by HEA-ACT is better than the best results found by other methods.

3.2.3 Speed reducer design problem

The approaches applied to this problem for comparison include Ray and Liew (2003), Mezura-Montes et al. (2006a), and Akhtar et al. (2002). Their results are shown in Table 17, and the best solutions obtained by the above approaches and HEA-ACT are listed in Table 18. The constraints are [–0.073923 –0.198006 –0.499095 –0.904643 –0.000014 –0.000005 –0.702500 –0.000006 –0.583331 –0.051378 –0.000006] based on the best result provided by HEA-ACT.

As shown in Table 17, while the number of FFEs for HEA-ACT is larger than those provided by Mezura-Montes et al. (2006a) and Akhtar et al. (2002), the best, median, mean, and worst results found by HEA-ACT are apparently better than those found by other methods.

Table 20 Comparison of results for three-bar truss design

	HEA-ACT	Ray and Liew (2003)
x_1	0.7886803456	0.7886210370
x_2	0.4082335517	0.4084013340
Best	263.895843	263.8958466

Table 21 Experimental results obtained by HEA-ACT for four engineering optimization problems over 30 independent runs with 100,000 FFEs

Problem	Results of HEA-ACT				
	Best	Median	Mean	Worst	SD
Welded beam design	2.38095658	2.38095658	2.38095658	2.38095658	3.8E-12
Spring design	0.012665233	0.012665233	0.012665233	0.012665233	5.3E-18
Speed reducer design	2,994.471066	2,994.471066	2,994.471066	2,994.471066	1.2E-09
Three-bar truss design	263.895843	263.895843	263.895843	263.895843	8.3E-12

3.2.4 Three-bar truss design problem

The approaches applied to this problem for comparative purpose is Ray and Liew (2003). Their results are shown in Table 19, and the best solutions obtained by the above approach and HEA-ACT are listed in Table 20. The constraints are [–0.000000 –1.464118 –0.535881] based on the best result provided by HEA-ACT.

As described in Table 19, HEA-ACT outperforms Ray and Liew (2003) with regards to all performance metrics.

Based on the above comparisons, one can conclude that HEA-ACT is of superior search quality and robustness for constrained engineering optimization problems. Furthermore, on the whole, the performance of HEA-ACT is much better than the compared methods.

Finally, to inspect whether the search quality can be further improved in engineering optimization problems, HEA-ACT has been tested using a large number of FFEs (100,000 FFEs). The results have been shown in Table 21. From Table 21, it can be found that the results obtained are of a much higher quality compared with the results in Table 12, and the standard deviations are extremely small. The above observation implies that HEA-ACT can keep improving its performance over time if more computational cost is allowed.

4 Conclusion

The research reported here proposes a hybrid evolutionary algorithm and an adaptive constraint-handling technique for constrained numerical and engineering optimization problems. The hybrid evolutionary algorithm includes simplex crossover, diversity mutation, and IBGA mutation. It is worthwhile to note that the crossover operation and the mutation operation are implemented concurrently and that the diversity mutation or the IBGA mutation is applied to an individual in the population with a probability of 0.5. In addition, the adaptive constraint-handling technique consists of three situations, i.e., the infeasible situation, the semi-feasible situation, and the feasible situation. The proposed approach first judges to which situation the current population belongs, and then the selection of individuals is based on the corresponding constraint-handling mechanism.

The method has been tested experimentally based on numerical and engineering constrained optimization problems. The experimental results indicate that the proposed method is very suitable for constrained optimization problems with different types and that it is superior to or competitive with the compared approaches. The effectiveness of the genetic operators adopted in HEA-ACT has been investigated by different experiments. It is found that the combination of these operators reaches more competitive results than when only using one or two of them, which suggests that these operators can be combined for constrained optimization problems. In addition, the effect of simplex crossover on performance is demonstrated.

Acknowledgements The authors would like to thank the two anonymous reviewers for their very careful and constructive comments and suggestions. The authors also gratefully acknowledge the support of the National Natural Science Foundation of China under Grant 60234030 and Grant 60673062, the National Basic Scientific Research Founds under Grant A1420060159, and the Otto Mønsted Fond for this research.

References

Aguirre AH, Rionda SB, Coello Coello CA, Lizáraga GL, Mezura-Montes E (2004) Handling constraints using multiobjective optimization concepts. Int J Numer Methods Eng 59(15):1989–2017

Akhtar S, Tai K, Ray T (2002) A socio-behavioural simulation model for engineering design optimization. Eng Optim 34(4):341–354

Becerra RL, Coello Coello CA (2006) Cultured differential evolution for constrained optimization. Comput Methods Appl Mech Eng 195(33–36):4303–4322

Brest J, Zumer V, Maucec MS (2006) Self-adaptive differential evolution algorithm in constrained real-parameter optimization. In: Proceedings of the congress on evolutionary computation (CEC'2006). IEEE Press, Sheraton Vancouver Wall Centre Hotel, Vancouver, BC, Canada, pp 215–222, July

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

Coello Coello CA (2000a) Constraint handling using an evolutionary multiobjective optimization technique. Civ Eng Environ Syst 17(4):319–346

Coello Coello CA (2000b) Treating constraints as objectives for single-objective evolutionary optimization. Eng Optim 32(3):275–308

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

Coello Coello CA, Becerra RL (2004) Efficient evolutionary optimization through the use of a cultural algorithm. Eng Optim 36(2):219–236

Coello Coello CA, Mezura-Montes E (2002) Constraint-handling in genetic algorithms through the use of dominance-based tournament selection. Adv Eng Inf 16(3):193–203

Deb K (2000) An efficient constraint handling method for genetic algorithms. Comput Methods Appl Mech Eng 18(2–4): 311–338

Dimopoulos GG (2007) Mixed-variable engineering optimization based on evolutionary and social metaphors. Comput Methods Appl Mech Eng 196(4–6):803–817

Eshelman LJ, Schaffer JD (1993) Real-coded genetic algorithms and interval-schemata. Foundations of genetic algorithms 2. Morgan Kaufman Publishers, San Mateo, pp 187–202

Farmani R, Wright JA (2005) Self-adaptive fitness formulation for constrained optimization. IEEE Trans Evol Comput 7(5):445–455

Fonseca CM, Fleming PJ (1999) Multiobjective optimization and multiple constraint handling with evolutionary algorithms-part I: a unified formulation. IEEE Trans Syst Man Cybern A, Syst Humans 28(1):26–37

Hedar AR, Fukushima M (2006) Derivative-free filter simulated annealing method for constrained continuous global optimization. J Glob Optim 35(4):521–649

Horn J, Nafpliotis N, Goldberg D (1994) A niched Pareto genetic algorithm for multiobjective optimization. In: Proceedings of the first IEEE conference on evolutionary computation. IEEE Press, Piscataway, NJ, pp 82–87

Huang F, Wang L, He Q (2007) An effective co-evolutionary differential evolution for constrained optimization. Appl Math Comput 186(1):340–356

Huang VL, Qin AK, Suganthan PN (2006) Self-adaptive differential evolution algorithm for constrained real-parameter optimization. In: Proceedings of the congress on evolutionary computation (CEC'2006). IEEE Press, Sheraton Vancouver Wall Centre Hotel, Vancouver, BC, Canada, pp 17–24, July

Knowles JD, Corne DW (2000) Approximating the nondominated front using the Pareto archived evolutionary strategy. Evol Comput 8(2):149–172

Krohling RA, Coelho LS (2006) Coevolutionary particle swarm optimization using Gaussian distribution for solving constrained optimization problems. IEEE Trans Syst Man Cybern B Cybern 36(6):1407–1416

Liang JJ, Suganthan PN (2006) Dynamic multi-Swarm particle swarm optimizer with a novel constraint-handling mechanism. In: Proceedings of the congress on evolutionary computation (CEC'2006). IEEE Press, Sheraton Vancouver Wall Centre Hotel, Vancouver, BC, Canada, pp 9–16, July

Mezura-Montes E, Coello Coello CA (2002) A numerical comparison of some multiobjective-based techniques to handle constraints in genetic algorithms. Technical Report EVOCINV-03-2002, Evolutionary Computation Group at CINVESTAV, Sección de Computación, Departamento de Ingeniería Eléctrica, CINVESTAV-IPN, México

Mezura-Montes E, Coello Coello CA (2005) A simple multimembered evolution strategy to solve constrained optimization problems. IEEE Trans Evol Comput 9(1):1–17

Mezura-Montes E, Coello Coello CA, Reyes JV (2006a) Increasing successful offspring and diversity in differential evolution for engineering design. In: Proceedings of the seventh international conference on adaptive computing in design and manufacture (ACDM 2006), pp 131–139, April

Mezura-Montes E, Velázquez-Reyes J, Coello Coello CA (2006b) Modified differential evolution for constrained optimization. In: Proceedings of the congress on evolutionary computation (CEC'2006), IEEE Press, Sheraton Vancouver Wall Centre Hotel, Vancouver, BC, Canada, pp 332–339, July

Michalewicz Z, Schoenauer M (1996) Evolutionary algorithm for constrained parameter optimization problems. Evol Comput 4(1):1–32

Mühlenbein H, Schlierkamp-Voosen D (1993) Predictive models for the breeder genetic algorithm I: Continuous parameter optimization. Evol Comput 1(1):25–49

Noman N, Iba H (2008) Accelerating differential evolution using an adaptive local search. IEEE Trans Evol Comput 12: 107–125

Ray T, Liew KM (2003) Society and civilization: an optimization algorithm based on the simulation of social behavior. IEEE Trans Evol Comput 7(4):386–396

Runarsson TP, Yao X (2000) Stochastic ranking for constrained evolutionary optimization. IEEE Trans Evol Comput 4(3):284–294

Schaffer JD (1985) Multiple objective optimization with vector evaluated genetic algorithms. In: Grefenstette JJ (ed) Proceedings of the 1st international conference on genetic algorithms and their applications. Earlbaum, Hillsdale, NJ, pp 93–100

Surry PD, Radcliffe NJ (1997) The COMOGA method: constrained optimization by multiobjective genetic algorithm. Control Cybern 26(3):391–412

Takahama T, Sakai S (2005) Constrained optimization by applying the α constrained method to the nonlinear simplex method with mutations. IEEE Trans Evol Comput 9(5): 437–451

Tsutsui S, Yamamure M, Higuchi T (1999) Multi-parent recombination with simplex crossover in real coded genetic algorithms. In: Proceedings of the genetic and evolutionary computation conference (GECCO'99), pp 657–664

Venkatraman S, Yen GG (2005) A generic framework for constrained optimization using genetic algorithms. IEEE Trans Evol Comput 9(4):424–435

Wang Y, Cai Z, Guo G, Zhou Y (2007a) Multiobjective optimization and hybrid evolutionary algorithm to solve constrained optimization problems. IEEE Trans Syst Man Cybern B Cybern 37(3):560–575

Wang Y, Liu H, Cai Z, Zhou Y (2007b) An orthogonal design based constrained optimization evolutionary algorithm. Eng Optim 39(6):715–736

Wang Y, Cai Z, Zhou Y, Zeng W (2008) An adaptive tradeoff model for constrained evolutionary optimization. IEEE Trans Evol Comput 12(1):80–92

Wolpert DH, Macready WG (1997) No free lunch theorems for optimization. IEEE Trans Evol Comput 1(1):67–82

Zhou Y, Li Y, He J, Kang L (2003) Multiobjective and MGG evolutionary algorithm for constrained optimization. In: Proceedings of the congress on evolutionary computing 2003 (CEC'2003). IEEE Press, Piscataway, NJ, pp 1–5