Engineering, 2010, 2, 65-77
doi:10.4236/eng.2010.21009 Published Online January 2010 (http://www.SciRP.org/journal/eng/).

An Adaptive Differential Evolution Algorithm to Solve Constrained Optimization Problems in Engineering Design

Youyun AO[1], Hongqin CHI[2]
[1]*School of Computer and Information, Anqing Teachers College, Anqing, China*
[2]*Department of Computer, Shanghai Normal University, Shanghai, China*
Email: youyun.ao@gmail.com, chihq@shnu.edu.cn
Received July 28, 2009; *revised August* 23, 2009; *accepted August* 28, 2009

Abstract

Differential evolution (DE) algorithm has been shown to be a simple and efficient evolutionary algorithm for global optimization over continuous spaces, and has been widely used in both benchmark test functions and real-world applications. This paper introduces a novel mutation operator, without using the scaling factor *F*, a conventional control parameter, and this mutation can generate multiple trial vectors by incorporating different weighted values at each generation, which can make the best of the selected multiple parents to improve the probability of generating a better offspring. In addition, in order to enhance the capacity of adaptation, a new and adaptive control parameter, i.e. the crossover rate *CR*, is presented and when one variable is beyond its boundary, a repair rule is also applied in this paper. The proposed algorithm ADE is validated on several constrained engineering design optimization problems reported in the specialized literature. Compared with respect to algorithms representative of the state-of-the-art in the area, the experimental results show that ADE can obtain good solutions on a test set of constrained optimization problems in engineering design.

Keywords: Differential Evolution, Constrained Optimization, Engineering Design, Evolutionary Algorithm, Constraint Handling

1. Introduction

Many real-world optimization problems involve multiple constraints which the optimal solution must satisfy. Usually, these problems are also called constrained optimization problems or nonlinear programming problems. Engineering design optimization problems are constrained optimization problems in engineering design. Like a constrained optimization problem, an engineering design optimization problem can be generally defined as follows [1–4]:

Minimize $f(\vec{x})$, $\vec{x} = [x_1, x_2, ..., x_n] \in \Re^n$

$$\text{Subject to } \begin{array}{l} g_j(\vec{x}) \le 0, j = 1,2,...,q \\ h_j(\vec{x}) = 0, j = q+1, q+2,...,m \end{array} \quad (1)$$

where $L_i \le x_i \le U_i, i = 1,2,...,D$

Here, n is the number of the decision or parameter variables (that is, $\vec{x}$ is a vector of size D), the ith variable x_i varies in the range $[L_i, U_i]$. The function $f(\vec{x})$ is the objective function, $g_j(\vec{x})$ is the jth inequality constraint and $h_j(\vec{x})$ is the jth equality constraint. The decision or search space S is written as $S = \prod_{i=1}^{D}[L_i, U_i]$, the feasible space expressed as $F = \{\vec{x} \in S \mid g_j(\vec{x}) \le 0, j = 1,2,...,q; h_j(\vec{x}) = 0, j = q+1, q+2,...,m\}$ is one subset of the decision space S (obviously, $F \subseteq S$) which satisfies the equality and inequality constraints.

Population-based evolutionary algorithm, mainly due to its ease to implement and use, and its less susceptibleness to the characteristics of the function to be optimized, has been very popular and successfully applied to constrained optimization problems [5]. And many successful applications of evolutionary algorithms to solve engineering design optimization problems in the specialized literature have been reported. Ray and Liew [6] used a swarm-like based approach to solve engineering optimization problems. He *et al.* [7] proposed an improved particle swarm optimization to solve mechanical design

optimization problems. Zhang *et al.* [8] proposed a differential evolution with dynamic stochastic selection to constrained optimization problems and constrained engineering design optimization problems. Akhtar *et al.* [9] proposed a socio-behavioural simulation model for engineering design optimization. He and Wang [10] proposed an effective co-evolutionary particle swarm optimization for constrained engineering design problems. Wang and Yin [11] proposed a ranking selection-based particle swarm optimizer for engineering design optimization problems. Differential evolution (DE) [12,13], a relatively new evolutionary technique, has been demonstrated to be simple and powerful and has been widely applied to both benchmark test functions and real-world applications [14]. This paper introduces an adaptive differential evolution (ADE) algorithm to solve engineering design optimization problems efficiently.

The remainder of this paper is organized as follows. Section 2 briefly introduces the basic idea of DE. Section 3 describes in detail the proposed algorithm ADE. Section 4 presents the experimental setup adopted and provides an analysis of the results obtained from our empirical study. Finally, our conclusions and some possible paths for future research are provided in Section 5.

2. The Basic DE Algorithm

Let's suppose that $\vec{x}_i^t = [x_{i,1}^t, x_{i,2}^t, ..., x_{i,D}^t]$ are solutions at generation t, $P^t = \{\vec{x}_1^t, \vec{x}_2^t, ..., \vec{x}_N^t\}$ is the population, where D denotes the dimension of solution space, N is the population size. In DE, the child population P^{t+1} is generated through the following operators [12,15]:

1) Mutation Operator: For each $\vec{x}_i^t$ in parent population, the mutant vector $\vec{v}_i^{t+1}$ is generated according to the following equation:

$$\vec{v}_i^{t+1} = \vec{x}_{r_1}^t + F \times (\vec{x}_{r_2}^t - \vec{x}_{r_3}^t) \tag{2}$$

where $r_1, r_2, r_3 \in \{1,2,...,N\} \setminus i$ are randomly chosen and mutually different, the scaling factor F controls amplification of the differential variation $(\vec{x}_{r_2}^t - \vec{x}_{r_3}^t)$.

2) Crossover Operator: For each individual $\vec{x}_i^t$, a trial vector $\vec{u}_i^{t+1}$ is generated by the following equation:

$$u_{i,j}^{t+1} = \begin{cases} v_{i,j}^{t+1}, & \text{if } (rand \le CR \parallel j = rand[1,D]) \\ x_{i,j}^t, & \text{otherwise} \end{cases} \tag{3}$$

where $rand$ is a uniform random number distributed between 0 and 1, $rand[1,D]$ is a randomly selected index from the set $\{1,2,...,D\}$, the crossover rate $CR \in [0,1]$ controls the diversity of the population.

3) Selection Operator: The child individual $\vec{x}_i^{t+1}$ is selected from each pair of $\vec{x}_i^t$ and $\vec{u}_i^{t+1}$ by using greedy selection criterion:

$$\vec{x}_i^{t+1} = \begin{cases} \vec{u}_i^{t+1}, & \text{if } (f(\vec{u}_i^{t+1}) < f(\vec{x}_i^t)) \\ \vec{x}_i^t, & \text{otherwise} \end{cases} \tag{4}$$

where the function f is the objective function and the condition $f(\vec{u}_i^{t+1}) < f(\vec{x}_i^t)$ means the individual $\vec{u}_i^{t+1}$ is better than $\vec{x}_i^t$.

Therefore, the conventional DE algorithm based on scheme DE/rand/1/bin is described in Figure 1 [15].

3. The Proposed Algorithm ADE

3.1. Generating Initial Population Using Orthogonal Design Method

Usually, the initial population $P^0 = \{\vec{x}_1^0, \vec{x}_2^0, ..., \vec{x}_N^0\}$ of evolutionary algorithms is randomly generated as follows:

$$\forall i \le N, \forall j \le D : x_{i,j}^0 = L_j + \mathrm{r}_j \times (U_j - L_j) \tag{5}$$

where N is the population size, D is the number of variables, r_j is a random number between 0 and 1, the jth variable of $\vec{x}_i^0$ is written as $x_{i,j}^0$, which is initialized in the range $[L_j, U_j]$. In order to improve the search efficiency, this paper employs orthogonal design method to generate the initial population, which can make some points closer to the global optimal point and improve the diversity of solutions. The orthogonal design method is described as follows [16]:

For any given individual $\vec{x} = [x_1, x_2, ..., x_D]$, the ith

1: Generate initial population $P^0 = \{\vec{x}_1^0, \vec{x}_2^0, ..., \vec{x}_N^0\}$
2: Let $t = 0$
3: **repeat**
4: **for** each individual $\vec{x}_i^t$ in the population P^t **do**
5: Generate three random integers r_1, r_2 and
6: $r_3 \in \{1,2,...,N\} \setminus i$, with $r_1 \ne r_2 \ne r_3$
7: Generate a random integer $j_{rand} \in \{1,2,...,D\}$
8: **for** each parameter j **do**
9: $u_{i,j}^{t+1} = \begin{cases} x_{r_3,j}^t + F \times (x_{r_1,j}^t - x_{r_2,j}^t), \\ \quad \text{if } (rand \le CR \parallel j = rand[1,D]) \\ x_{i,j}^t, \quad \text{otherwise} \end{cases}$
10: **end for**
11: Replace $\vec{x}_i^t$ with the child $\vec{u}_i^{t+1}$ in the population P^{t+1},
12: if $\vec{u}_i^{t+1}$ is better, otherwise $\vec{x}_i^t$ is retained
13: **end for**
14: $t = t + 1$
15: **until** the termination condition is achieved

Figure 1. Pseudocode of differential evolution based on scheme DE/rand/1/bin.

decision variable x_i varies in the range $[L_i, U_i]$. Here, each x_i is regarded as one factor of orthogonal design. Suppose that each factor holds Q levels, namely, quantize the domain $[L_i, U_i]$ into Q levels $\alpha_1, \alpha_2, ..., \alpha_Q$. The jth level of the ith factor is written as $\alpha_{i,j}$, which is defined as follows:

$$a_{i,j} = \begin{cases} L_i & , j = 1 \\ L_i + (j-1)(\frac{U_i - L_i}{Q-1}) & , 2 \le j \le Q-1 \\ U_i & , j = Q \end{cases} \quad (6)$$

And then, we create the orthogonal array $M = (b_{i,j})_{N \times D}$ with D factors and Q levels, where N is the number of level combinations. The procedure of constructing one orthogonal array $M = (b_{i,j})_{N \times D}$ is described in Figure 2.

Therefore, the initial population $P^0 = (x_{i,j}^0)_{N \times D}$ is generated by using the orthogonal array $M = (b_{i,j})_{N \times D}$, where the jth variable of individual $\vec{x}_i^0$ is $x_{i,j}^0 = a_{j,b_{i,j}}$.

3.2. Multi-Parent Mutation Scheme

According to the different variants of mutation, there are several different DE schemes often used, which are formulated as follows [12]:

"DE/rand/1/bin": $$\vec{v}_i^{t+1} = \vec{x}_{r_1}^t + F \times (\vec{x}_{r_2}^t - \vec{x}_{r_3}^t) \quad (7)$$

"DE/best/1/bin": $$\vec{v}_i^{t+1} = \vec{x}_{best} + F \times (\vec{x}_{r_1}^t - \vec{x}_{r_2}^t) \quad (8)$$

"DE/current to best/2/bin":

$$\vec{v}_i^{t+1} = \vec{x}_i^t + F \times (\vec{x}_{best} - \vec{x}_i^t) + F \times (\vec{x}_{r_1}^t - \vec{x}_{r_2}^t) \quad (9)$$

"DE/best/2/bin":

$$\vec{v}_i^{t+1} = \vec{x}_{best} + F \times (\vec{x}_{r_1}^t - \vec{x}_{r_2}^t) + F \times (\vec{x}_{r_3}^t - \vec{x}_{r_4}^t) \quad (10)$$

"DE/rand/2/bin":

$$\vec{v}_i^{t+1} = \vec{x}_{r_1}^t + F \times (\vec{x}_{r_2}^t - \vec{x}_{r_3}^t) + F \times (\vec{x}_{r_4}^t - \vec{x}_{r_5}^t) \quad (11)$$

1: **for** ($i = 1; i \le N; i++$)
2: { $b_{i,1} =$ **int**($(i-1)/Q$) **mod** Q ; $b_{i,2} = (i-1)$ **mod** Q }
3: **for** ($j = 3; j \le D; j++$)
4: **for** ($i = 1; i \le N; i++$)
5: { $b_{i,j} = (b_{i,1} \times (j-2) + b_{i,2})$ **mod** Q }
6: Increment $b_{i,j}$ by one for $1 \le i \le N, 1 \le j \le D$

Figure 2. Procedure of constructing one orthogonal array $M = (b_j^i)_{N \times D}$.

where $\vec{x}_{best}$ is the best individual of the current population. Usually, based on both the control parameter *F* and the selected multiple parents, using these DE schemes can only generate a vector after a single mutation. Tsutsui *et al.* [17] proposed a multi-parent recombination with simplex crossover in real coded genetic algorithms to utilize the selected multiple parents and improve the diversity of offspring. Inspired by multi-parent recombination with simplex crossover, this paper proposes a novel multi-parent mutation in differential evolution. The multi-parent mutation is described in the following.

For each individual $\vec{x}_i^t$ from the population P^t with population size N, $i = 1, 2, ..., N$. A perturbed vector $\vec{v}_i^{t+1}$ is generated according to the following formula:

$$\vec{v}_i^{t+1} = \vec{x}_i^t + \sum_{k=1}^{K} w_k \times (\vec{x}_{r_k}^t - \vec{x}_{r_{k+1}}^t) \quad (12)$$

where $r_1, r_2, ..., r_K \in \{1, 2, ..., N\} \setminus i$, K randomly chosen integers are mutually different, and $\vec{x}_{r_{K+1}}^t = \vec{x}_{r_1}^t$. The weighted value w_k is defined as follows:

$$\vec{\xi} = randn(1, K), \vec{w} = \vec{\xi} / sum(\vec{\xi}) \quad (13)$$

where $randn(1, K)$ is a 1-by-K matrix with normally distributed random numbers, $sum(\vec{\xi})$ is used for calculating the sum of all components of the vector $\vec{\xi}$, and $\vec{w} = [w_1, w_2, ..., w_K]$.

According to the varying $\vec{w}$, repeat Formulas (13) and (12) for K times, K new vectors $\vec{v}_i^{t+1}\{1\}$, $\vec{v}_i^{t+1}\{2\}$, ..., $\vec{v}_i^{t+1}\{K\}$ are generated from these K selected parents. And then K vectors $\vec{x}_i^{t+1}\{1\}$, $\vec{x}_i^{t+1}\{2\}$, $\cdots$, $\vec{x}_i^{t+1}\{K\}$ are created by crossover, repair and constraint handling described in Subsections 3.3-3.5 respectively. Finally, an offspring individual $\vec{x}_i^{t+1}$ of the $(t+1)$th generation population P^{t+1} is obtained by selecting the best individual from these K offspring and their common parent $\vec{x}_i^t$.

3.3. Adaptive Crossover Rate *CR*

In conventional DE, the crossover rate *CR* is a constant value between 0 and 1. This paper proposes an adaptive crossover rate *CR*, which is defined as follows:

$$CR = CR_0 \times \exp(-a(\frac{t}{T})\text{^}\, b) \quad (14)$$

where the initial crossover rate CR_0 is a constant value and usually is set to 0.8 or 0.85, t is the current genera-

tion number and T is the maximal generation number, b is a shape parameter determining the degree of dependency on the generation number, a and b are positive constants, usually a is set to 2, b is set to 2 or 3. At the early stage, DE uses a bigger crossover rate CR to preserve the diversity of solutions and prevent premature; at the later stage, DE employs a smaller crossover rate CR to enhance the local search and prevent the better solutions found from being destroyed.

3.4. Repair Method

After crossover, if one or more of the variables in the new vector $\vec{u}_i^{t+1}$ are beyond their boundaries, the violated variable value $\vec{u}_{i,j}^{t+1}$ is either reflected back from the violated boundary or set to the corresponding boundary value using the repair rule as follows [18,19]:

$$u_{i,j}^{t+1}=\begin{cases}\dfrac{L_j+u_{i,j}^{t+1}}{2}, & \text{if}(p\le 1/3)\wedge(u_{i,j}^{t+1}<L_j)\\ L_j, & \text{if}(1/3<p\le 2/3)\wedge(u_{i,j}^{t+1}<L_j)\\ 2L_j-u_{i,j}^{t+1}, & \text{if}(p>2/3)\wedge(u_{i,j}^{t+1}<L_j)\\ \dfrac{U_j+u_{i,j}^{t+1}}{2}, & \text{if}(p\le 1/3)\wedge(u_{i,j}^{t+1}>U_j)\\ U_j, & \text{if}(1/3<p\le 2/3)\wedge(u_{i,j}^{t+1}>U_j)\\ 2U_j-u_{i,j}^{t+1}, & \text{if}(p>2/3)\wedge(u_{i,j}^{t+1}>U_j)\end{cases} \quad (15)$$

where p is a probability and uniformly distributed random number in the range $[0,1]$.

3.5. Constraint Handling Technique of Feasibility-Based Rule

In evolutionary algorithms for solving constrained optimization problems, the most common method to handle constraints is to use penalty functions. In general, the constraint violation function of one individual $\vec{x}$ is transformed by m equality and inequality constraints as follows [4]:

$$G(\vec{x})=\sum_{j=1}^{q}w_j\max(0,g_j(\vec{x}))^{\beta}+\sum_{j=q+1}^{m}w_j\max(0,|h_j(\vec{x})|-\varepsilon)^{\beta} \quad (16)$$

where the exponent β is usually set to 1 or 2, ε is a tolerance allowed (a very small value) for the equality constraints and the coefficient w_j is greater than zero. If $\vec{x}$ is a feasible solution, $G(\vec{x})=0$, otherwise $G(\vec{x})>0$. The function value $G(\vec{x})$ shows that the degree of constraints violation of individual $\vec{x}$. β is set to 2 and w_j is set to 1 in this study.

In this study, a simple and efficient constraint handling technique of feasibility-based rule is introduced, which is also a constraint handling technique without using parameters. When two solutions are compared at a time, the following criteria are always applied [1]:

1) If one solution is feasible, and the other is infeasible, the feasible solution is preferred;

2) If both solutions are feasible, the one with the better objective function value is preferred;

3) If both solutions are infeasible, the one with smaller constraint violation function value is preferred.

3.6. Algorithm Framework

The general framework of the proposed algorithm ADE is described in Figure 3.

4. Experimental Study

4.1. Constrained Optimization Problems in Engineering Design

In order to validate the proposed algorithm ADE, we use six benchmark test problems, which are commonly used

1: Generate initial population $P^0=\{\vec{x}_1^0,\vec{x}_2^0,...,\vec{x}_N^0\}$ using
2: orthogonal design method, set CR_0 and let $t=0$
3: **repeat**
4: **for** each individual $\vec{x}_i^t$ in the population P^t **do**
5: Generate K random integers r_1, r_2, …, r_K
6: $\in\{1,2,...,N\}\setminus i$, they are also mutually different
7: **for** each $k\in\{1,2,...K\}$ **do**
8: Apply multi-parent mutation to generate new
9: vector $\vec{v}_i^{t+1}\{k\}$
10: **for** each parameter j **do**
11: $u_{i,j}^{t+1}\{k\}=\begin{cases}\vec{v}_{i,j}^{t+1}\{k\}, & \text{if } (rand\le CR \,\|\, j=rand[1,D])\\ x_{i,j}^t, & \text{otherwise}\end{cases}$
12: If $u_{i,j}^{t+1}$ is beyond its lower or upper boundaries,
13: repair rule is enforced
14: **end for**
15: **end for**
16: Find out the best one $\vec{u}_i^{t+1}$ of the children
17: $\{\vec{u}_i^{t+1}\{1\},\vec{u}_i^{t+1}\{2\},...,\vec{u}_i^{t+1}\{K\}\}$ /*apply the
18 feasibility–based rule */
19: Replace $\vec{x}_i^t$ with $\vec{u}_i^{t+1}$ in the population P^{t+1},
20: if $\vec{u}_i^{t+1}$ is better, otherwise $\vec{x}_i^t$ is retained
21: **end for**
22: $t=t+1$
23: **until** the termination condition is achieved

Figure 3. The general framework of the ADE algorithm.

in the specialized literature, and which are described in the following.

1) Three-bar truss design [8]:

Minimize $f(\vec{x}) = (2\sqrt{2}x_1 + x_2) \times l$

Subject to $g_1(\vec{x}) = \dfrac{\sqrt{2}x_1 + x_2}{\sqrt{2}x_1^2 + 2x_1x_2} P - \sigma \le 0$,

$$g_2(\vec{x}) = \frac{x_2}{\sqrt{2}x_1^2 + 2x_1x_2} P - \sigma \le 0,$$

$$g_3(\vec{x}) = \frac{1}{x_1 + \sqrt{2}x_2} P - \sigma \le 0$$

where $0 \le x_1 \le 1$ and $0 \le x_2 \le 1$; $l = 100\text{cm}$, $P = 2\text{KN/cm}^2$, and $\sigma = 2\text{KN/cm}^2$.

2) Spring design [8]:

Minimize $f(\vec{x}) = (x_3 + 2)x_1x_2^2$

Subject to $g_1(\vec{x}) = 1 - \dfrac{x_1^3 x_3}{71785x_2^4} \le 0$,

$$g_2(\vec{x}) = \frac{4x_1^2 - x_1x_2}{12566(x_1x_2^3 - x_2^4)} + \frac{1}{5108x_2^2} - 1 \le 0,$$

$$g_3(\vec{x}) = 1 - \frac{140.45x_2}{x_1^2 x_3} \le 0,$$

$$g_4(\vec{x}) = \frac{x_1 + x_2}{1.5} - 1 \le 0$$

where $0.25 \le x_1 \le 1.3$, $0.05 \le x_2 \le 2.0$, and $2 \le x_3 \le 15$.

3) Pressure vessel design [9,20]:

Minimize $f(\vec{x}) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 + 19.84x_1^2x_3$

Subject to $g_1(\vec{x}) = -x_1 + 0.0193x_3 \le 0$,

$$g_2(\vec{x}) = -x_2 + 0.00954x_3 \le 0,$$

$$g_3(\vec{x}) = -\pi x_3^2 x_4 - \frac{4}{3}\pi x_3^3 + 1{,}296{,}000 \le 0,$$

$$g_4(\vec{x}) = x_4 - 240 \le 0$$

where $x_1 = 0.0625n_1$, $x_2 = 0.0625n_2$, $1 \le n_1 \le 99$, $1 \le n_2 \le 99$, $10 \le x_3 \le 200$, $10 \le x_4 \le 200$.

4) Welded beam design [9]:

Minimize

$$f(\vec{x}) = 1.10471x_1^2x_2 + 0.04811x_3x_4(14.0 + x_2)$$

Subject to $g_1(\vec{x}) = \tau(x) - \tau_{\max} \le 0$,

$$g_2(\vec{x}) = \sigma(x) - \sigma_{\max} \le 0,$$

$$g_3(\vec{x}) = x_1 - x_4 \le 0$$

$$g_4(\vec{x}) = 0.10471x_1^2 + 0.04811x_3x_4(14.0 + x_2) - 5.0 \le 0,$$

$$g_5(\vec{x}) = 0.125 - x_1 \le 0$$

$$g_6(\vec{x}) = \delta(x) - \delta_{\max} \le 0,$$

$$g_7(\vec{x}) = P - P_c(x) \le 0$$

The other parameters are defined as follows:

$$\tau(\vec{x}) = \sqrt{(\tau')^2 + \frac{2\tau'\tau''x_2}{2R} + (\tau'')^2}, \quad \tau' = \frac{P}{\sqrt{2}x_1x_2}, \quad \tau'' = \frac{MR}{J},$$

$$M = P(L + \frac{x_2}{2}), \quad R = \sqrt{\frac{x_2^2}{4} + (\frac{x_1 + x_3}{2})^2},$$

$$J = 2\left\{\frac{x_1x_2}{\sqrt{2}}\left[\frac{x_2^2}{12} + \left(\frac{x_1 + x_3}{2}\right)^2\right]\right\}, \qquad \sigma(\vec{x}) = \frac{6PL}{x_4x_3^2},$$

$$\delta(\vec{x}) = \frac{4PL^3}{Ex_4x_3^3},$$

$$P_c(\vec{x}) = \frac{4.013\sqrt{EGx_3^2x_4^6/36}}{L^2}\left(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right),$$

where $P = 6000$ lb., $L = 14$ in, $\delta_{\max} = 0.25$ in., $E = 30 \times 10^6$ psi, $G = 12 \times 10^6$ psi, $\tau_{\max} = 13{,}600$ psi, $\sigma_{\max} = 30{,}000$ psi, $0.1 \le x_1 \le 2.0$, $0.1 \le x_2 \le 10.0$, $0.1 \le x_3 \le 10.0$, and $0.1 \le x_4 \le 2.0$.

5) Speed reducer design [8]:

Minimize

$$f(\vec{x}) = 0.7854x_1x_2^2(3.3333x_3^2 + 14.9334x_3 - 43.0934) - 1.508x_1(x_6^2 + x_7^2) + 7.4777(x_6^3 + x_7^3) + 0.7854(x_4x_6^2 + x_5x_7^2)$$

Subject to $g_1(\vec{x}) = \dfrac{27}{x_1x_2^2x_3} - 1 \le 0$,

$$g_2(\vec{x}) = \frac{397.5}{x_1x_2^2x_3^2} - 1 \le 0,$$

$$g_3(\vec{x}) = \frac{1.93x_4^3}{x_2x_3x_6^4} - 1 \le 0,$$

$$g_4(\vec{x}) = \frac{1.93x_5^3}{x_2x_3x_7^4} - 1 \le 0,$$

$$g_5(\vec{x}) = \frac{[(745x_4/(x_2x_3))^2 + 16.9 \times 10^6]^{1/2}}{110.0x_6^3} - 1 \le 0,$$

$$g_6(\vec{x}) = \frac{[(745x_5/(x_2x_3))^2 + 157.5 \times 10^6]^{1/2}}{85.0x_7^3} - 1 \le 0,$$

$$g_7(\vec{x}) = \frac{x_2x_3}{40} - 1 \le 0, \; g_8(\vec{x}) = \frac{5x_2}{x_1} - 1 \le 0,$$

$$g_9(\vec{x}) = \frac{x_1}{12x_2} - 1 \le 0,$$

$$g_{10}(\vec{x}) = \frac{1.5x_6 + 1.9}{x_4} - 1 \le 0,$$

$$g_{11}(\vec{x}) = \frac{1.1x_7 + 1.9}{x_5} - 1 \le 0$$

where $2.6 \le x_1 \le 3.6$, $0.7 \le x_2 \le 0.8$, $17 \le x_3 \le 28$, $7.3 \le x_4 \le 8.3$, $7.3 \le x_5 \le 8.3$, $2.9 \le x_6 \le 3.9$, $5.0 \le x_7 \le 5.5$.

6) Himmelblau's Nonlinear Optimization Problem [21]:

This problem was proposed by Himmelblau and similar to problem $g04$ [22] of the benchmark except for the second coefficient of the first constraint. There are five design variables. The problem can be stated as follows:

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

$$\text{Subject to } g_1(\vec{x}) = 85.334407 + 0.0056858x_2x_5 + 0.00026x_1x_4 - 0.0022053x_3x_5, \quad -92 \le 0$$

$$g_2(\vec{x}) = -85.334407 - 0.0056858x_2x_5 - 0.00026x_1x_4 + 0.0022053x_3x_5 \le 0,$$

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

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

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

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

where $78 \le x_1 \le 102$, $33 \le x_2 \le 45$, and $27 \le x_i \le 45$ $(i = 3,4,5)$.

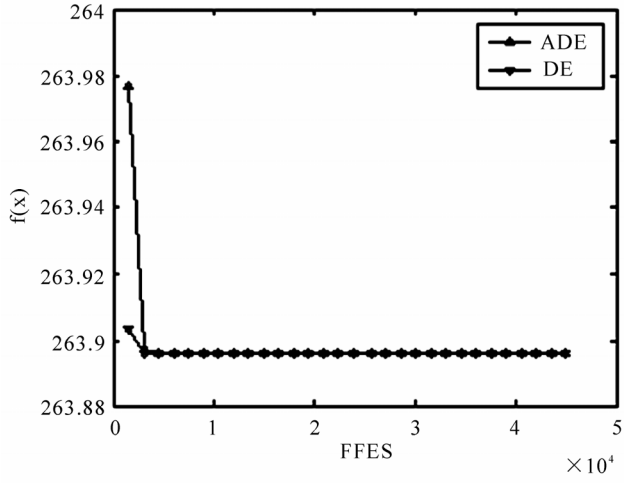

Figure 4. Convergence graph for three-bar truss design.

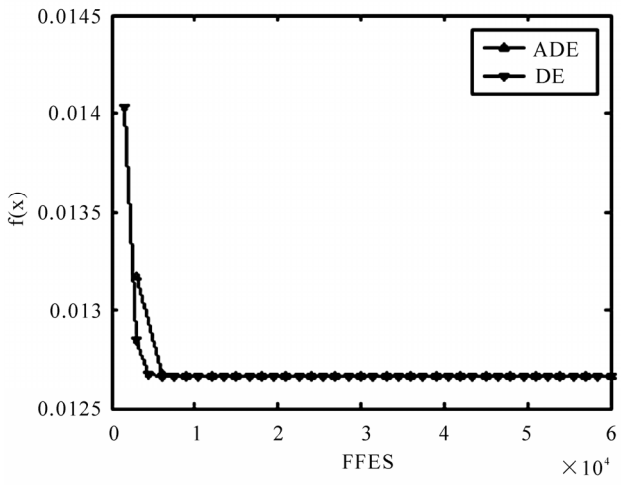

Figure 5. Convergence graph for spring design.

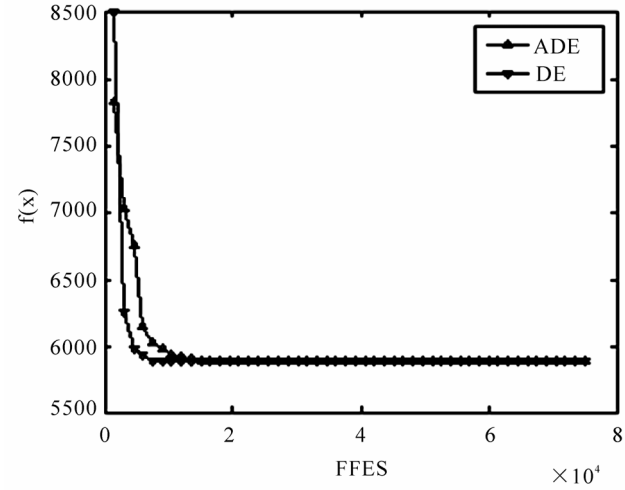

Figure 6. Convergence graph for pressure vessel design.

4.2. Convergence of ADE

In this section, Figures 4-9 depict the convergence graphs for 6 engineering optimization problems described above respectively. From Figures 4-6, we know that ADE and DE all can be quickly convergent. In the figures, FFES is the number of fitness function evaluations.

4.3. Comparing ADE with Respect to Some State-of-the-Art Algorithms

In this experimental study, the parameter values used in ADE are set as follows: the population size $N = 50$, the maximal generation number $T = 300$, the level number $Q = \lfloor \sqrt{N} \rfloor$, the mutation parent number $K = D + 1$, the

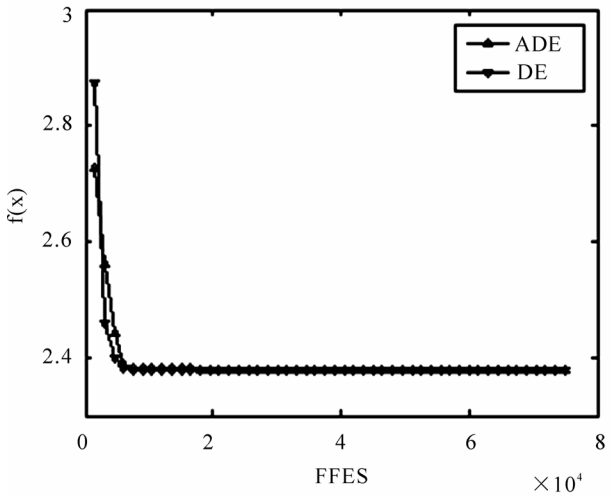

Figure 7. Convergence graph for welded beam design.

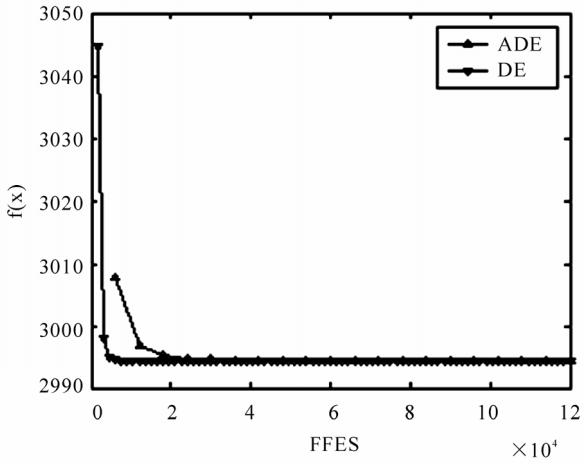

Figure 8. Convergence graph for speed reducer design.

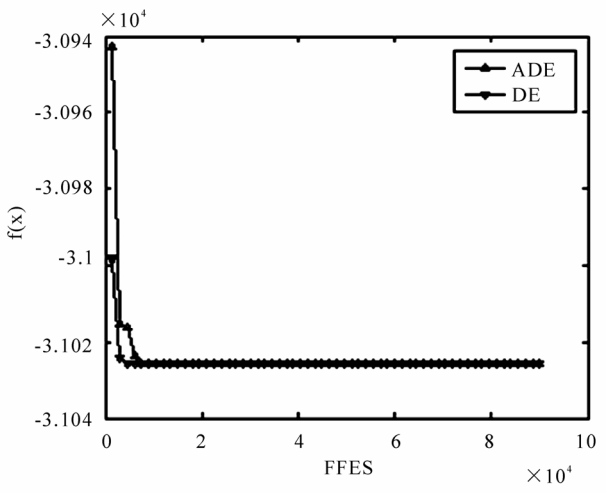

Figure 9. Convergence graph for Himmelblau's nonlinear optimization problem.

initial crossover rate $CR_0 = 0.8$, the coefficient $a = 2$, the shape parameter $b = 3$, the exponent $\beta = 2$. The number of fitness function evaluations (FFES) is equal to $N \times T \times K$. The achieved solution at the end of $N \times T \times K$ FFES is used to measure the performance of ADE. ADE is independently run 30 times on each test problem above. The optimized objective function values (of 30 runs) arranged in ascending order and the 15th value in the list is called the median optimized function value. Experimental results are presented in Tables 1-12. And NA is the abbreviation for "Not Available".

For three-bar truss design problem, the experimental results are given in Tables 1-2. According to Table 1, ADE and DSS-MDE [8] can obtain the approximate best and median values, which are slightly better than those obtained by Ray

Table 1. Comparison of statistical results for three-bar truss design over 30 runs.

Algorithms	Best	Median	Mean	Worst	Std	FFES
ADE	263.89584338	263.89584338	263.89584338	263.89584338	4.72e-014	45,000
DSS-MDE [8]	263.8958434	263.8958434	263.8958436	263.8958498	9.72e-07	15,000
Ray and Liew [6]	263.8958466	263.8989	263.9033	263.96975	1.26e-02	17.610

Table 2. Comparison of best solutions found for three-bar truss design.

Function	ADE	DSS-MDE [8]	Ray and Liew [6]	ECT [23]	Ray and Saini [24]
x_1	0.7886751376014	0.7886751359	0.7886210370	0.78976441	0.795
x_2	0.4082482819599	0.4082482868	0.4084013340	0.40517605	0.395
$f(x)$	263.895843376	263.8958434	263.8958466	263.896710000	264.300
FFES	45,000	15,000	17,610	55,000	2712

Table 3. Comparison of statistical results for spring design over 30 runs.

Algorithms	Best	Median	Mean	Worst	Std	FFES
ADE	0.0126652328	0.0126652458	0.0129336018	0.02064372078	1.46e-03	60,000
SiC-PSO [20]	0.012665	NA	0.0131	NA	4.1e-04	24,000
FSA [25]	0.012665285	NA	0.012665299	0.012665338	2.2e-08	49,531
DSS-MDE [8]	0.012665233	0.012665304	0.012669366	0.012738262	1.25e-05	24,000
Ray and Liew [6]	0.01266924934	0.012922669	0.012922669	0.016717272	5.92e-04	25,167
Coello [26]	0.01270478	0.01275576	0.01276920	0.01282208	NA	900,000

Table 4. Comparison of best solutions found for spring design.

Function	ADE	SiC-PSO [20]	DSS-MDE [8]	FSA [25]	He *et al*. [7]
x_1	0.35674653865	0.354190	0.3567177469	0.35800478345599	0.356750
x_2	0.05169025814	0.051583	0.0516890614	0.05174250340926	0.051690
x_3	11.28727756428	11.438675	11.2889653382	11.21390736278739	11.287126
$f(x)$	0.0126652328	0.012665	0.01265233	0.012665285	0.012665
FFES	60,000	24,000	24,00	49,531	15,000

Table 5. Comparison of statistical results for pressure vessel design over 30 runs.

Algorithms	Best	Median	Mean	Worst	Std	FFES
ADE	5885.3327736	5885.3327785	5885.3349564	5885.3769425	8.66e-03	75,000
SiC-PSO [20]	6059.714335	NA	6092.0498	NA	12.1725	24,000
Ray and Liew [6]	6171.00	NA	6335.05	NA	NA	20,000
He *et al.* [7]	6059.714	NA	6289.929	NA	3.1e+2	30,000
Montes *et al.* [3]	6059.702	6059.702	6059.702	6059.702	1.0e-12	24,000

and Liew [6] respectively. The mean and worst values obtained by ADE are the best among three algorithms, while the FFES (45,000) of ADE is also the highest. And we also find that these algorithms can find the near-optimal solutions. From Table 2, we can see that ADE can find the best value when compared with respect to DSS-MDE [8], Ray and Liew [6], ECT [22] and Ray and Saini [23]. The best result obtained by ADE is

$$f(\vec{x})=263.8958433764684,$$

corresponding to

$$\vec{x}=[x_1, x_2]=[0.78867513760142, 0.40824828195990]$$

and constraints

$$[g_1(\vec{x}),\ g_2(\vec{x}),\ g_3(\vec{x})\ ]$$

$$=[0, -1.46410162480516, -0.53589837519484].$$

For spring design problem, the experimental results are given in Tables 3-4. According to Table 3, ADE, Sic-PSO [20], FSA [24], DSS-MDE [8] can find out the best value when compared with respect to Ray and Liew [6] and Coello [25]. The median value obtained by ADE is better than obtained by other methods, but the mean and worst values are worse, this is because that ADE can only find 29 near-optimal solutions in 30 runs and the other is an exception solution (i.e., the worst value is 0.02064372078). Table 4 presents the detail of each best value obtained by ADE, SiC-PSO [20], DSS-MDE [8], FSA [24] or He *et al.* [7] respectively. The best result obtained by ADE is

$$f(\vec{x})=0.01266523278832,$$

corresponding to

$$\vec{x}=[x_1, x_2, x_3]$$

$$=[0.35671785021031, 0.05168906567225, 11.28895927857073]$$

and constraints

$[g_1(\vec{x}), g_2(\vec{x}), g_3(\vec{x}), g_4(\vec{x})]$
=[-2.220446049250313e-016, -4.4408 92098500626e-016, 4.05378584839796, -0.72772872274496].

For pressure vessel design problem, the experimental results are given in Tables 5-6. According to Table 5, the best, median, mean, worst and standard deviation of values obtained by ADE are the best when compared with respect to Sic-PSO [20], Ray and Liew [6], He *et al.* [7], and Montes *et al.* [3], while the FFES (75,000) of ADE is also the highest. Table 6 presents the detail of each best value obtained by ADE, SiC-PSO [20], Ray and Liew [6], He *et al.* [7] or Montes *et al.* [3] respectively. The best result obtained by ADE is

$$f(\vec{x}) = 5885.332773616458,$$

corresponding to

$$\vec{x} = [x_1, x_2, x_3, x_4]$$
$$= [0.778168641375, 0.384649162628, 40.319618724099, 200]$$

and constraints

$[g_1(\vec{x}), g_2(\vec{x}), g_3(\vec{x}), g_4(\vec{x})]$
=[-1.110223024625157e-016,0,0,-40].

For welded beam design problem, the experimental results are provided with Tables 7-8. According to Table 7, the best, median, mean, worst and standard derivation of values obtained by ADE are slightly worse than those obtained by DSS-MDE [8] and are better than those obtained by Ray and Liew [6], FSA [25] and Deb [1]. However, the FFES (75,000) of ADE is the highest. Table 8 presents the detail of each best value obtained by DSS-MDE [8], He *et al.* [7], FSA [25], Ray and Liew [6], and Akhtar *et al.* [9] respectively. The best result obtained by ADE is

$$f(\vec{x}) = 2.3809565\ 8032252,$$

corresponding to

$$\vec{x} = [x_1, x_2, x_3, x_4]$$
$$= [0.24436897580173, 6.21751971517460, 8.29147139048\ 684, 0.24436897580173]$$

and constraints

$[g_1(\vec{x}), g_2(\vec{x}), g_3(\vec{x}), g_4(\vec{x}), g_5(\vec{x}), g_6(\vec{x}), g_7(\vec{x})]$
=[-1.091393642127514e-011, -3.310560714453459e-010, -1.387778780781446e-016, -3.02295458760400, -0.11936897580173, -0.23424083488769, -1.273292582482100e-011].

Table 6. Comparison of best solutions found for pressure vessel design.

Function	ADE	Sic-PSO [20]	Ray and Liew [6]	He *et al.* [7]	Montes *et al.* [3]
x_1	0.7781686414	0.812500	0.8125	0.8125	0.8125
x_2	0.3846491626	0.437500	0.4375	0.4375	0.4375
x_3	40.319618724	42.098445	41.9768	42.098446	42.098446
x_4	200	176.636595	182.9768	176.636052	176.636047
$f(x)$	5885.3327736	6059.714335	6171.0	6059.7143	6059.701660
FFES	75,000	24,000	20,000	30,000	24,000

Table 7. Comparison of statistical results for welded beam design over 30 runs.

Algorithms	Best	Median	Mean	Worst	Std	FFES
ADE	2.380956580	2.380956580	2.380956585	2.380956708	2.35e-08	75,000
DSS-MDE [8]	2.38095658	2.38095658	2.38095658	2.38095658	3.19e-10	24,000
Ray and Liew [6]	2.3854347	3.2551371	3.0025883	6.3996785	0.959078	33,095
FSA [25]	2.381065	NA	2.404166	2.488967	NA	56.243
Deb [1]	2.38119	2.39289	NA	2.64583	NA	40,080

Table 8. Comparison of best solutions found for welded beam design.

Function	ADE	DSS-MDE [8]	He *et al.* [7]	FSA [25]	Ray and Liew [6]	Akhtar *et al.* [9]
x_1	0.24436897580	0.2443689758	0.244369	0.24435257	0.244438276	0.2407
x_2	6.21751971517	6.2175197152	6.217520	6.2157922	6.237967234	6.4851
x_3	8.29147139049	8.2914713905	8.291471	8.2939046	8.288576143	8.2399
x_4	0.24436897580	0.2443689758	0.244369	0.24435258	0.244566182	0.2497
$f(x)$	2.38095658032	2.38095658	2.380957	2.381065	2.3854347	2.4426
FFES	75,000	24,000	30,000	56,243	33,095	19,259

Table 9. Comparison of statistical results for speed reducer design over 30 runs.

Algorithms	Best	Median	Mean	Worst	Std	FFES
ADE	2994.4710662	2994.4710662	2994.4710662	2994.4710662	1.85e-012	120,000
DSS-MDE [8]	2994.471066	2994.471066	2994.471066	2994.471066	3.58e-012	30,000
Ray and Liew [6]	2994.744241	3001.758264	3001.7582264	3009.964736	4.0091423	54,456
Montes *et al.* [27]	2996.356689	NA	2996.367220	NA	8.2e-03	24,000
Akhtar *et al.* [9]	3008.08	NA	3012.12	3028	NA	19,154

Table 10. Comparison of best solutions found for speed reducer design.

Function	ADE	DSS-MDE [8]	Ray and Liew [6]	Montes *et al.* [27]	Akhtar *et al.*[9]
x_1	3.5	3.5	3.50000681	3.500010	3.506122
x_2	0.7	0.7	0.70000001	0.7	0.700006
x_3	17	17	17	17	17
x_4	7.3	7.3	7.32760205	7.300156	7.549126
x_5	7.715319911478	7.7153199115	7.71532175	7.800027	7.859330
x_6	3.350214666096	3.3502146661	3.35026702	3.350221	3.365576
x_7	5.286654464980	5.2866544650	5.28665450	5.286685	5.289773
$f(x)$	2994.4710662	2994.471066	2994.744241	2996.356689	3008.08
FFES	120,000	30,000	54,456	24,000	18,154

Table 11. Comparison of statistical results for himmelblau's nonlinear optimization problem.

Algorithms	Best	Median	Mean	Worst	Std	FFES
ADE	-31025.56024	-31025.56024	-31025.56024	-31025.56024	5.91e-010	90,000
COPSO [28]	-31025.56024	NA	-31025.56024	NA	0	200,000
HU-PSO [29]	-31025.56142	NA	-31025.56142	NA	0	200,000

Table 12. Comparison of best solutions found for himmelblau's nonlinear optimization problem.

Function	ADE	COPSO [28]	HU-PSO [29]	Colleo [21]	Homaifar *et al.* [30]
x_1	78.00000000000000	78	78.0	78.0495	78.0000
x_2	33.00000000000000	33	33.0	33.0070	33.0000
x_3	27.07099710517604	27.070997	27.070997	27.0810	29.9950
x_4	45.00000000000000	45	45.0	45.0000	45.0000
x_5	44.96924255010549	44.969242	44.96924255	44.9400	36.7760
$f(x)$	-31025.56024249794	-31025.56024	-31025.56142	-31020.859	-30665.609
FFES	90,000	200,000	200,000	NA	NA

For speed reducer design problem, the experimental results are given in Tables 9-10. According to Table 9, the best, median, mean, worst and standard derivation of values obtained by ADE and DSS-MDE [8] are superior to those obtained by Ray and Liew [6], Montes *et al.* [27] and Akhtar *et al.* [9] respectively, while the FFES (120,000) of ADE is the highest. Table 10 shows the detail of each best value obtained by ADE, DSS-MDE [8], Ray and Liew [6], Montes *et al.* [27] and Akhtar *et al.* [9] respectively. The best result obtained by ADE is

$$f(\vec{x}) = 2994.47106614682020,$$

corresponding to

$$\vec{x} = [x_1, x_2, x_3, x_4, x_5, x_6, x_7]$$
$$= [3.5, 0.7, 17, 7.3, 7.71531991147825, 3.35021466609645, 5.28665446498022]$$

and constraints

$$[g_1(\vec{x}), g_2(\vec{x}), g_3(\vec{x}), g_4(\vec{x}), g_5(\vec{x}), g_6(\vec{x}), g_7(\vec{x}), g_8(\vec{x}), g_9(\vec{x}), g_{10}(\vec{x}), g_{11}(\vec{x})]$$
$$=[-0.07391528039787, -0.19799852714195, -0.49917224810242, -0.90464390455607, -6.661338147750939e-016, 0, -0.70250000000000, -2.220446049250313e-016, -0.58333333333333, -0.05132575354183, -8.881784197001252e-016].$$

For Himmelblau's nonlinear optimization problem, the best, median, mean, worst and standard derivation of values is shown in Tables 11-12, it is clearly seen that ADE, COPSO [28], and HU-PSO [29] all can find one near-optimal solution after a single run. Additionally, ADE only requires 90,000 FFES, which is superior to other several algorithms, such as COPSO [28] 200,000 FFES and HU-PSO [29] 200,000 FFES. The best result obtained by ADE is

$$f(\vec{x}) = -3.1025.56024249794,$$

corresponding to

$$\vec{x} = [x_1, x_2, x_3, x_4, x_5]$$
$$= [78, 33, 27.07099710517604, 45, 44.96924255010549]$$

and constraints

$$[g_1(\vec{x}), g_2(\vec{x}), g_3(\vec{x}), g_4(\vec{x}), g_5(\vec{x}), g_6(\vec{x})]$$
$$=[0, -92, -9.59476568762383, -10.40523431237617, -5, 0].$$

In sum, compared with respect to several state-of-the-art algorithms, ADE can perform better on six benchmark test problems. It is clearly shown that ADE is feasible and effective to solve constrained optimization problems in engineering design. The reason is that ADE uses multi-parent mutation to generate a better offspring, and applies self-adaptive control parameter and effective repair rule etc.

5. Conclusions and Future Work

This paper proposes an adaptive differential evolution (ADE) algorithm for constrained optimization in Engineering Design. Firstly, ADE employs the orthogonal design method to generate the initial population to improve the diversity of solutions. Secondly, a multi-parent mutation scheme is developed to improve the capacity of exploration and the convergence speed of ADE. Thirdly, in order to improve the adaptive capacity of crossover operator, a new approach to adjusting the crossover rate is presented. In addition, ADE introduces a new repair rule and a constraint handling technique of the feasible-based rule is also applied when comparing two solutions at a time. Finally, ADE is tested on six constrained engineering design optimization problems taken from the specialized literature. Compared with respect to several state-of-the-art algorithms, the experimental results show that ADE is highly competitive and can obtain good results in terms of a test set of constrained optimization

problems in engineering design. However, there are still some things to do in the future. Firstly, we will further validate ADE in the case of higher dimensions. Secondly, we also will take some measures to improve the convergence speed during the evolutionary process. Additionally, testing some initial parameters of ADE is another future work.

6. References

[1] K. Deb, "An efficient constraint handling method for genetic algorithms," Computer Methods in Applied Mechanics and Engineering, Vol. 186, No. 2, pp. 311–338, 2000.

[2] E. Mezura-Montes and A. G. Palomeque-Qrtiz, "Parameter control in differential evolution for constrained optimization," 2009 IEEE Congress on Evolutionary Computation (CEC'2009), pp. 1375–1382, 2009.

[3] E. Mezura-Montes, C. A. Coello Coello, J. Velázquez-Reyes, and L. Muñoz-Dávila, "Multiple trial vectors in differential evolution for engineering design," Engineering Optimization, Vol. 39, No. 5, pp. 567–589, 2007.

[4] C. A. Coello Coello, "Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: A survey of the state of the art," Computer Methods in Applied Mechanics and Engineering, Vol. 191, No. 11-12, pp. 1245–1287, 2002.

[5] R. Landa-Becerra and C. A. Coello Coello, "Cultured differential evolution for constrained optimization," Computer Methods in Applied Mechanics and Engineering, Vol. 195, No. 33–36, pp. 4303–4322, 2006.

[6] T. Ray and K. M. Liew, "Society and civilization: An optimization algorithm based on the simulation of social behavior," IEEE Transactions on Evolutionary Computation, Vol. 7, No. 4, pp. 386–396, 2003.

[7] S. He, E. Prempain, and Q. H. Wu, "An improved particle swarm optimizer for mechanical design optimization problems," Engineering Optimization, Vol. 36, No. 5, pp. 585–605, 2004.

[8] M. Zhang, W. J. Luo, and X. F. Wang, "Differential evolution with dynamic stochastic selection for constrained optimization," Information Sciences, Vol. 178, pp. 3043–3074, 2008.

[9] S. Akhtar, K. Tai, and T. Ray, "A socio-behavioural simulation model for engineering design optimization," Engineering Optimization, Vol. 34, No. 4, pp. 341–354, 2002.

[10] Q. He and L. Wang, "An effective co-evolutionary particle swarm optimization for constrained engineering design problems," Engineering Applications of Artificial Intelligence, Vol. 20, No. 1, pp. 89–99, 2007.

[11] J. H. Wang and Z. Y. Yin, "A ranking selection-based particle swarm optimizer for engineering design optimization problems," Structural and Multidisciplinary Optimization, Vol. 37, No. 2, pp. 131–147, 2008.

[12] R. Storn and K. Price, "Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces," Journal of Global Optimization, Vol. 11, pp. 341–359, 1997.

[13] K. Price, R. Storn, and J. Lampinen, "Differential evolution: A practical approach to global optimization," Berlin: Springer-Verlag, 2005.

[14] Z. Y. Yang, K. Tang, and X. Yao, "Self-adaptive differential evolution with neighborhood search," 2008 Congress on Evolutionary Computation (CEC'2008), pp. 1110–1116, 2008.

[15] H. A. Abbass, R. Sarker, and C. Newton, "PDE: A Pareto-frontier differential evolution approach for multiobjective optimization problems," Proceedings of IEEE Congress on Evolutionary Computation, Vol. 2, pp. 971–978, 2001.

[16] Y. W. Leung and Y. P. Wang, "An orthogonal genetic algorithm with quantization for global numerical optimization," IEEE Transactions on Evolutionary Computation, Vol. 5, No. 1, pp. 40–53, 2001.

[17] S. Tsutsui, M. Yamamure, and T. Higuchi, "Multi-parent recombination with simplex crossover in real coded genetic algorithms," Proceedings of the Genetic and Evolutionary Computation Conference, pp. 657–664, 1999.

[18] J. Brest, V. Zumer, and M. S. Maucec, "Self-adaptive differential evolution algorithm in constrained real-parameter optimization," 2006 IEEE Congress on Evolutionary Computation (CEC'2006), pp. 919–926, 2006.

[19] Y. Wang and Z. X. Cai, "A hybrid multi-swarm particle swarm optimization to solve constrained optimization problems," Frontiers of Computer Science in China, Vol. 3, No. 1, pp. 38–52, 2009.

[20] L. C. Cagnina, S. C. Esquivel, and C. A. Coello Coello, "Solving engineering optimization problems with the simple constrained particle swarm optimizer," Informatica, Vol. 32, pp. 319–326, 2008.

[21] C. A. Coello Coello, "Use of a self-adaptive penalty approach for engineering optimization problems," Computers in Industry, Vol. 41, No. 2, pp. 113–127, 2000.

[22] T. P. Runarsson and X. Yao, "Stochastic ranking for constrained evolutionary optimization," IEEE Transactions on Evolutionary Computation, Vol. 4, No. 3, pp. 284–294, 2000.

[23] K. Hans Raj, R. S. Sharma, G. S. Mishra, A. Dua, and C. Patvardhan, "An evolutionary computational technique for constrained optimisation in engineering design," Journal of the Institution of Engineers India Part Me Mechanical Engineering Division, Vol. 86, pp. 121–128, 2005.

[24] T. Ray and P. Saini, "Engineering design optimization using a swarm with intelligent information sharing among individuals," Engineering Optimization, Vol. 33, No. 33, pp. 735–748, 2001.

[25] A. R. Hedar and M. Fukushima, "Derivative-free filter simulated annealing method for constrained continuous global optimization," Journal of Global Optimization, Vol. 35, No. 4, pp. 521–549, 2006.

[26] C. A. Coello, "Self-adaptive penalties for GA- based optimization," Proceedings of the Congress on Evolutionary Computation 1999 (CEC'99), Vol. 1, pp. 573–580, 1999.

[27] E. Mezura-Montes, C. A. Coello, and J. V. Reyes, "Increasing successful offspring and diversity in differential evolution for engineering design," Proceedings of the Seventh International Conference on Adaptive Computing in Design and Manufacture (ACDM 2006), pp. 131–139, 2006.

[28] A. E. Muñoz Zavala, A. Hernández Aguirre, E. R. Villa Diharce, and S. Botello Rionda, "Constrained optimization with an improved particle swarm optimization algorithm," International Journal of Intelligent Computing and Cybernetics, Vol. 1, No. 3, pp. 425–453, 2008.

[29] X. H. Hu, R. C. Eberhart, and Y. H. Shi, "Engineering optimization with particle swarm," Proceedings of the 2003 IEEE Swarm Intelligence Symposium, pp. 53–57, 2003.

[30] A. Homaifar, S. H. Y. Lai, and X. Qi, "Constrained optimization via genetic algorithms," Simulation, Vol. 62, No. 4, pp. 242–254, 1994.

