Dynamic Differential Evolution for Constrained Real-Parameter Optimization

Youyun Ao¹, Hongqin Chi²

¹School of Computer and Information, Anqing Teachers College, Anqing, China
Email: youyun.ao@gmail.com
²Department of Computer, Shanghai Normal University, Shanghai, China
Email: chihq@shnu.edu.cn

Abstract—Differential evolution (DE) has been shown to be a simple and effective evolutionary algorithm for global optimization both in benchmark test functions and many real-world applications. This paper introduces a dynamic differential evolution (D-DE) algorithm to solve constrained optimization problems. In D-DE, a novel mutation operator is firstly designed to prevent premature. Secondly, the scale factor \( F \) and the crossover probability \( CR \) are dynamic and adaptive to be beneficial for adjusting control parameters during the evolutionary process, especially, when done without any user interaction. Thirdly, D-DE uses orthogonal design method to generate initial population and reinitialize some solutions to replace some worse solutions during the search process. Finally, D-DE is validated on 6 benchmark functions and real-world applications. This paper introduces a dynamic multi-swarm particle swarm optimizer with a novel constraint-handling mechanism (DMS-PSO). DMS-PSO adopts a novel constraint-handling mechanism based on multi-swarm. Different from the existing constraints handling methods, sub-swarms are adaptively assigned to explore different constraints during the search process. Additionally, DMS-PSO introduces Sequential Quadratic Programming (SQP) method to improve local search ability. Finally, DMS-PSO is applied to the solution of constrained real-parameter optimization. Mezura-Montes et al. [8] proposed a modified differential evolution for constrained optimization (MDE). In order to increase the probability of each parent to generate a better offspring, MDE allows each solution to generate more than one offspring but using a different mutation operator which combines information of the current parent to find new search directions. Besides, MDE employs three selection criteria based on feasibility to deal with the constraints and adopts a diversity mechanism to maintain infeasible solutions located in promising areas of the search space. Takahama and Sakai [9] proposed a novel constrained optimization algorithm by the \( \varepsilon \) constrained differential evolution with gradient-based mutation and feasible elites (\( \varepsilon \)DE). Firstly, \( \varepsilon \)DE applies the \( \varepsilon \) constrained method to differential evolution. Secondly, to solve problems with
many equality constraints faster, which are very difficult problems for numerical optimization, dDE proposes gradient-based mutation and feasible elites preserving strategy. Finally, dDE is tested on 24 benchmark test functions provided by the CEC 2006 special session on constrained real-parameter optimization. Differential evolution (DE) [10], [11], a relatively new evolutionary technique, has been shown to be simple and powerful and has been widely applied to both benchmark test functions and real-world applications [12]. After analyzing the existing evolutionary algorithms, this paper introduces a new dynamic differential evolution (D-DE) algorithm for constrained real-parameter optimization efficiently.

The remainder of this paper is organized as follows. Section II briefly introduces the basic idea of DE. Section III describes in detail the D-DE algorithm. Section IV presents experimental settings adopted by D-DE, conventional DE and GA, respectively. Section VI provides an analysis of the results obtained from our empirical study. Finally, some conclusions and some possible paths for future research are provided in Section VII.

II. THE BASIC IDEA OF CONVENTIONAL DE

Let us assume that \( \vec{x}_0 = [x_{0,1}, x_{0,2}, \ldots, x_{0,N}] \) are solutions at generation 0, \( P^0 = \{x_0, x_1, \ldots, x_M\} \) is population, where \( M \) denotes the dimension of solution space, \( N \) is population size. In conventional DE, the child population \( P^{t+1} \) is generated through the following operators [10], [13]:

A. Mutation Operator

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

\[
\vec{v}_i^{t+1} = \vec{x}_i + F \times (\vec{x}_j - \vec{x}_k)
\]

Where \( 1 \leq i, j, k \leq N \) and \( i \neq j \neq k \). \( F \) is the scaling factor used to control amplification of the differential variation \( \vec{x}_j - \vec{x}_k \).

B. Crossover Operator

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

\[
\vec{u}_i^{t+1} = \begin{cases} 
\vec{v}_i^{t+1}, & \text{if } (\text{rand} \leq CR \| j = \text{rand}[1, D]) \\
\vec{x}_i, & \text{otherwise} 
\end{cases}
\]

Where \( \text{rand} \) is a uniformly distributed random number between 0 and 1, \( \text{rand}[1, D] \) is a randomly selected index from the set \( \{1, 2, \ldots, D\} \), the crossover probability \( CR \in [0,1] \) is used to control the diversity of individuals.

C. Selection Operator

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

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

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

III. THE PROPOSED ALGORITHM DDE

A. Orthogonal Initial Population

Generally, the initial population \( P^0 = \{x_0^0, x_1^0, \ldots, x_N^0\} \) of evolutionary algorithm is randomly generated as follows:

\[
\forall i \leq N, \forall j \leq D : x_{i,j}^0 = L_j + r_j \times (U_j - L_j) \quad (5)
\]

Where \( N \) is the population size, \( D \) is the number of variables, \( r_j \) is a random number between 0 and 1, the \( j \)-th variable of \( x_{i,j}^0 \) 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 [14]:

For any given individual \( \vec{x} = [x_1, x_2, \ldots, x_D] \), the \( i \)-th decision variable \( x_i \) varies in the range \( [L_i, U_i] \). Here, each \( x_i \) is taken as each factor of orthogonal design. Let us assume that each factor holds \( Q \) levels, namely, quantize the domain \( [L_i, U_i] \) into \( Q \) levels \( \alpha_1, \alpha_2, \ldots, \alpha_Q \).

The \( j \)-th level of the \( i \)-th factor \( \alpha_{i,j} \) is defined as follows:

\[
\alpha_{i,j} = \begin{cases} 
L_i, & j = 1 \\
L_i + (j - 1)(\frac{U_i - L_i}{Q - 1}), & 2 \leq j \leq Q - 1 \\
U_i, & j = Q 
\end{cases}
\]

Thereafter, 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 generating the orthogonal array \( M = (b_{i,j})_{N \times D} \) is described as follows:

1: for \( i = 1 \) to \( N \) do \( i \leftarrow i++ \)
2: \( b_{1,j} = \text{int} ((i - 1)/Q) \mod Q \)
3: for \( j = 2 \) to \( D \) do \( j \leftarrow j++ \)
4: for \( i = 1 \) to \( N \) do \( i \leftarrow i++ \)
5: \( b_{i,j} = (b_{i,j-1} \cdot (j - 2) + b_{i,j}) \mod Q \)
6: Increment \( b_{i,j} \) by one for \( 1 \leq i \leq N, 1 \leq j \leq D \)

Figure 2. Generating orthogonal array \( M = (b_{i,j})_{N \times D} \).
Therefore, the initial population $P^0=\{x_{i,j}^0\}_{N \times D}$ can be generated by using the orthogonal array $M = (h_{i,j})_{N \times D}$, where the $j$th variable of individual $x_{i,j}^0$ is $x_{i,j}^0 = a_{j,h_{i,j}}$.

### B. Novel Mutation Scheme

According to the different variants of mutation, there are several different DE schemes often used, which are formulated as follows [10]:

- "DE/rand/1/bin": $v_{i}^{t+1} = x_{i}^{t} + F \times (x_{r1}^{t} - x_{r2}^{t})$  
  \begin{equation}
  (7)
  \end{equation}

- "DE/best/1/bin": $v_{i}^{t+1} = x_{\text{best}}^{t} + F \times (x_{i}^{t} - x_{r}^{t})$  
  \begin{equation}
  (8)
  \end{equation}

- "DE/current to best/2/bin": 
  $v_{i}^{t+1} = x_{\text{best}}^{t} + F \times (x_{i}^{t} - x_{r}^{t}) + F \times (x_{r}^{t} - x_{j}^{t})$  
  \begin{equation}
  (9)
  \end{equation}

- "DE/rand/2/bin": 
  $v_{i}^{t+1} = x_{i}^{t} + F \times (x_{r}^{t} - x_{j}^{t}) + F \times (x_{r}^{t} - x_{k}^{t})$  
  \begin{equation}
  (10)
  \end{equation}

- "DE/rand/2/bin": 
  $v_{i}^{t+1} = x_{\text{best}}^{t} + F \times (x_{i}^{t} - x_{r}^{t}) + F \times (x_{j}^{t} - x_{k}^{t})$  
  \begin{equation}
  (11)
  \end{equation}

Where $x_{\text{best}}^{t}$ is the best solution of the current population, and the control parameter $F$ is usually set to be a constant. Using $x_{\text{best}}^{t}$ can improve the convergence speed but also increase the probability of getting stuck in the local optimum. In order to overcome the limitations, this paper proposes a novel variant of mutation, which is defined as follows:

$\hat{v}_{i}^{t+1} = x_{\text{better}}^{t} + F \times \sum_{k=1}^{K/2} (x_{2k+1}^{t} - x_{2k}^{t})$  
\begin{equation}
(12)
\end{equation}

Where $n_1, n_2, ..., n_K \in \{1, 2, ..., N\} \setminus i$, they are $K$ mutually different and randomly chosen integers. The better solution $x_{\text{better}}^{t}$ is a random sample from top $N_s$ solutions after ranking the current population based on the feasibility rule described in the later. The scale factor $F$ is a dynamic control parameter and related to the generation number, which is defined as follows:

$F = F_{\text{min}} + (F_{\text{max}} - F_{\text{min}}) \times (1 - \frac{t}{T})^{F_s}$  
\begin{equation}
(13)
\end{equation}

Here $F_{\text{min}}$ and $F_{\text{max}}$ are the bottom and upper boundaries of $F$, and usually are set to 0.1 and 0.9 respectively. The exponent $F_s$ is a shape parameter determining the degree of dependency on the generation number and usually is set to 2 or 3. Two parameters $t$ and $T$ are the current generation number and the maximal generation number respectively.

### C. Dynamic Control Parameters

In conventional DE, the crossover probability $CR$ is a constant value between 0 and 1. In this study, a dynamic crossover probability $CR$ is defined as follows:

$CR = CR_{\text{min}} + (CR_{\text{max}} - CR_{\text{min}}) \times (1 - \frac{t}{T})^{CR_b}$  
\begin{equation}
(14)
\end{equation}

Here $CR_{\text{min}}$ and $CR_{\text{max}}$ are the bottom and upper boundaries of $CR$, and usually are set to 0.1 and 0.9 respectively. The exponent $CR_b$ is a shape parameter determining the degree of dependency on the generation number and usually is set to 2 or 3. Two parameters $t$ and $T$ are the current generation number and the maximal generation number respectively.

At the early stage, D-DE uses a bigger scale factor $F$ and a bigger crossover probability $CR$ to search the solution space to preserve the diversity of solutions and prevent premature; at the later stage, D-DE employs a smaller scale factor $F$ and a smaller crossover probability $CR$ to search the solution space to enhance the local search and prevent the better solutions found from being destroyed.

### D. Repair Rule

After crossover, if one or more of the variables in the new vector $u_{i}^{t+1}$ are outside their boundaries, the violated variable value $\bar{u}_{i}^{t+1}$ is either reflected back from the violated boundary or set to the corresponding boundary value using the repair rule as follows [15], [16]:

\begin{align}
\bar{u}_{i,j}^{t+1} &= \begin{cases} 
(L_j + u_{i,j}^{t+1}) / 2, & \text{if } (p \leq 1/3) \land (u_{i,j}^{t+1} < L_j) \\
L_j, & \text{if } (1/3 < p \leq 2/3) \land (u_{i,j}^{t+1} < L_j) \\
2L_j - u_{i,j}^{t+1}, & \text{if } (p > 2/3) \land (u_{i,j}^{t+1} < L_j) \\
(U_j + u_{i,j}^{t+1}) / 2, & \text{if } (p \leq 1/3) \land (u_{i,j}^{t+1} > U_j) \\
U_j, & \text{if } (1/3 < p \leq 2/3) \land (u_{i,j}^{t+1} > U_j) \\
2U_j - u_{i,j}^{t+1}, & \text{if } (p > 2/3) \land (u_{i,j}^{t+1} > U_j)
\end{cases}
\end{align}

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

### E. Constraint Handling Mechanism

In evolutionary algorithms for solving constrained optimization problems, the most common method to handle constraints is to use penalty functions. Usually, equality constraints are transformed into inequalities of the form [4]:

$|h_j(\bar{x})| - \varepsilon \leq 0, j = q + 1, q + 2, ..., m$  
\begin{equation}
(16)
\end{equation}

Here $\varepsilon$ is a tolerance allowed (a very small value) for the equality constraints.

In general, the constraint violation function of one individual $\bar{x}$ is transformed by $m$ equality and inequality constraints as follows [9], [17], [18]:

$G(\bar{x}) = \sum_{j=1}^{q} w_j \max(0, g_j(\bar{x})) + \sum_{j=q+1}^{m} \max(0, |h_j(\bar{x})| - \varepsilon)$  
\begin{equation}
(17)
\end{equation}

Here the exponent $\beta$ is a positive number and usually set to 1 or 2, and the coefficient $w_j$ is greater than zero. The function value $G(\bar{x})$ shows that the degree of constraints violation of individual $\bar{x}$. $\beta$ 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 parameters. When two solutions are compared at a time, the following criteria are always applied [3]:

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.

F. Algorithm Framework

The general framework of the D-DE algorithm is outlined as follows:

1. Generate orthogonal initial population \( P^0 = \{x_1^0, x_2^0, ..., x_N^0\} \).
2. Initialize parameters, and let \( t = 0 \).
3. Evaluate \( P^0 \), and rank \( P^0 \) based on feasibility rule.
4. \textbf{repeat}
5. \textbf{for} each individual \( i \) in the population \( P^t \) \textbf{do}
6. Generate \( K \) random integers \( r_1, r_2, ..., r_K \in \{1, 2, ..., N\} \) \textbf{end for}
7. \textbf{for} each parameter \( j \in \{1, 2, ..., N/K\} \) \textbf{do}
8. \( \tilde{x}^{(i),j} = \text{rand} \ni \tilde{x}^{(j),i} \iff (\text{rand} \leq \|j\| j = \text{rand}[1, D]) \)
9. \textbf{end for}
10. Apply repair rule to repair \( \tilde{x}^{(i),j} \) if required, and evaluate \( \tilde{x}^{(i),j} \).
11. Replace \( x_i \) with the child \( \tilde{x}^{(i),j} \) in the population \( P^{t+1} \), if \( \tilde{x}^{(i),j} \) is better, otherwise \( x_i \) is retained.
12. \textbf{end for}
13. \textbf{end for}
14. Randomly select a sample \( x_{\text{better}} \) from top \( N_b \) individuals of the current population \( P^t \).
15. \textbf{repeat}
16. \textbf{for} each individual \( i \) in \( \{1, 2, ..., N\} \) \textbf{do}
17. \textbf{end for}
18. \textbf{end for}
19. The optimum solution is \( \tilde{x}^* \).

IV. TEST FUNCTION SUITE

In order to validate D-DE, we employ 6 benchmark test problems \( g04 \), \( g06 \), \( g08 \), \( g11 \), \( g12 \), which are provided by the CEC 2006 special session on constrained real-parameter optimization [4], and which are described in the following.

A. Test function \( g02 \)

Minimize \( f(\tilde{x}) = -\frac{\sum_{i=1}^{n} \cos(x_i) - 2 \prod_{i=1}^{n} \cos^2(x_i)}{\sqrt[n]{\sum_{i=1}^{n} x_i^2}} \)

Subject to \( g_1(\tilde{x}) = 0.75 - \sum_{i=1}^{n} x_i \leq 0 \)

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

Where \( n = 20 \) and \( 0 < x_i \leq 10 \). The global minimum \( \tilde{x}^* = (3.16246061572185, 3.12833142812967, 3.09479212988791, 3.06145059523469, 3.02792915885555, 3.019982606707130, 3.02903267044826762241835, 3.04424700958760, 3.0440 38285956317) \), the best is \( f(\tilde{x}^*) = -0.8036190412559 \), constraint \( g_1 \) is close to being active.

B. Test function \( g04 \)

Minimize \( f(\tilde{x}) = 5.3578547x_1^2 + 0.835689x_1x_3 + 37.293239x_4 - 40792.141 \)

Subject to \( g_1(\tilde{x}) = 85.334407 + 0.0056858x_2x_3 + 0.0006262x_1x_4 - 0.0022053x_3x_5 - 92 \leq 0 \)

\( g_2(\tilde{x}) = -85.334407 - 0.0056858x_2x_5 - 0.0006262x_1x_4 + 0.0022053x_3x_5 \leq 0 \)

\( g_3(\tilde{x}) = 80.51249 + 0.0071317x_2x_5 + 0.0029955x_2x_3 + 0.0021813x_2^2 - 110 \leq 0 \)

\( g_4(\tilde{x}) = -80.51249 - 0.0071317x_2x_5 - 0.0029955x_2x_3 - 0.0021813x_2^2 + 90 \leq 0 \)

\( g_5(\tilde{x}) = 9.300961 - 0.0047026x_2x_5 + 0.0021547x_1x_3 + 0.0019085x_3x_4 - 25 \leq 0 \)

Where \( 78 \leq x_1 \leq 102, \quad 33 \leq x_2 \leq 45 \quad 27 \leq x_3 \leq 45 \quad 1 \leq 3, 4, 5 \). The optimum solution is \( \tilde{x}^* = (78.33, 29.9952560256815985, 45.367751819057882073) \), where \( f(\tilde{x}^*) = -3.066553867187332e+004 \). Both constraints are active (\( g_1 \) and \( g_6 \)).

C. Test function \( g06 \)

Minimize \( f(\tilde{x}) = (x_1 - 10)^2 + (x_2 - 20)^2 \)

Subject to \( g_1(\tilde{x}) = -(x_1 - 5)^2 - (x_2 - 5)^2 + 100 \leq 0 \)

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

Where \( 13 \leq x_1 \leq 100 \) and \( 0 \leq x_2 \leq 100 \). The optimum solution is \( \tilde{x}^* = (14.09500000000000064, 0.84 29607892154795668) \) where \( f(\tilde{x}^*) = -6961.81387558015 \). Both constraints are active.

D. Test function \( g08 \)

Minimize \( f(\tilde{x}) = -\frac{\sin^3(2\pi x_1) \sin(2\pi x_2)}{x_1 + x_2} \)

Subject to \( g_1(\tilde{x}) = x_2 - x_1 + 1 \leq 0 \)

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

Where \( 0 \leq x_1 \leq 10 \) and \( 0 \leq x_2 \leq 10 \). The optimum solution is \( \tilde{x}^* = (1.22797135260752599, 4.24537336612 274885) \) where \( f(\tilde{x}^*) = -0.0958250 414180359 \).
where \( f(\hat{x}) = 0.7499 \).

**F. Test function g12**

Minimize

\[
\min f(\hat{x}) = -100 - (x_1 - 5)^2 - (x_2 - 5)^2 - (x_3 - 5)^2 / 100
\]

Subject to \( g(\hat{x}) = (x_1 - p)^2 + (x_2 - q)^2 + (x_3 - p)^2 - 0.0625 \leq 0 \)

Where \( 0 \leq x_i \leq 10 \) \( i = 1, 2, 3 \) and \( p, q, r = 1, 2, \ldots, 9 \). The feasible region of the search space consists of \( 9^3 \) disjointed spheres. A point \((x_1, x_2, x_3)\) is feasible if and only if there exists \( p, q, r \) such that the above inequality holds. The optimum solution is \( \hat{x}^* = (5, 5, 5) \) where \( f(\hat{x}^*) = -1 \). The solution lies within the feasible region.

**V. EXPERIMENTAL SETTINGS**

**A. Parameter Settings of D-DE**

In our experimental study, the parameter values used in D-DE are set as follows: the population size \( N = 50 \), the maximal generation number \( T = 50000 \), the level number \( Q = \sqrt{N} \), the number of top solutions \( N_p = 0.1 \times N = 5 \), the number of replaced solutions \( N_h = 0.1 \times N = 5 \), the minimal and maximal values of scale factor \( F \) are set to \( F_{\text{min}} = 0.1 \) and \( F_{\text{max}} = 0.9 \) respectively, \( K = 6 \), the minimal and maximal values of crossover probability \( CR \) are set to \( CR_{\text{min}} = 0.1 \) and \( CR_{\text{max}} = 0.9 \) respectively, the shape parameter values \( F_a = 3 \) and \( F_b = 3 \) respectively, the exponent \( \beta = 2 \), the tolerant value \( \varepsilon = 0.0001 \). The number of function evaluations (FES) is equal to \( (1 + N_p) \times N \times T = 275000 \). The achieved solution at the end of \( N \times T \) FES is used to measure the performance of D-DE. D-DE is independently run 30 times on each test function.

**B. Parameter Settings of Conventional DE**

In our experimental study, the parameter values adopted by the conventional DE are set as follows: the population size \( N = 50 \), the maximal generation number \( T = 55000 \), the crossover probability \( CR = 0.9 \), the scale factor \( F = 0.6 \), the tolerant value \( \varepsilon = 0.0001 \). The number of function evaluations (FES) is equal to \( N \times T = 275000 \). The achieved solution at the end of \( N \times T \) FES is used to measure the performance of the conventional DE. The DE employs the repair rule and constraint handling mechanism described in Section III. The parameter values employed by the real-coded GA are set as follows: the population size \( N = 50 \), the maximal generation number \( T = 55000 \), the crossover probability \( P_c = 0.9 \) and a mutation probability \( P_m = 1/n \) (where \( n \) is the number of decision variables for real-coded GA), the distribution indexes for crossover and mutation operators as \( \eta_c = 20 \) and \( \eta_m = 20 \); the tolerant value \( \varepsilon = 0.0001 \). The number of function evaluations (FES) is equal to \( N \times T = 275000 \). The obtained solution at the end of \( N \times T \) FES is used to measure the performance of the conventional GA. The GA is independently run 30 times on each test function.

**VI. EXPERIMENTAL RESULTS AND DISCUSSIONS**

**A. Comparison with Respect to Conventional DE and GA on 6 Benchmark Test Problems**

Using the above experimental settings, the best, mean and worst results obtained by D-DE, DE, and GA are given in Tables I-III. As shown in Table I, D-DE, DE and GA all can find the best solution for each test problems \( g08 \), \( g11 \) and \( g12 \), respectively. For test problems \( g04 \) and \( g06 \), D-DE and DE can find the best solution and is better than that of GA. For test problem \( g02 \), the best result obtained by DE is slightly better than that of D-DE, and obviously better than that of GA. Additionally, according to Tables I-III, we can find that the worst result obtained by D-DE is better than or not worse than that of DE and GA for all test problems \( g02, g04, g06, g08, g11 \) and \( g12 \), while the mean result obtained by DE is better than or not worse than that of GA for test problems \( g02, g04, g06, g08 \) and \( g12 \) except for test problem \( g11 \) where the mean result obtained by GA is better than that of DE. Table II shows that the worst result obtained by D-DE is better than or not worse than that of DE and GA for all test problems \( g02, g04, g06, g08 \) and \( g12 \) except for test problem \( g11 \) where the worst result obtained by GA is better than that of DE. Additionally, according to Tables I-III, we can find that D-DE can almost find the optimum solution for each test problems \( g02, g04, g06, g08, g11 \) and \( g12 \) in one single run, and that D-DE is robust and can outperform DE and GA on a set of test problems.

<table>
<thead>
<tr>
<th>Function</th>
<th>Optimal</th>
<th>D-DE</th>
<th>DE</th>
<th>GA</th>
</tr>
</thead>
<tbody>
<tr>
<td>g02</td>
<td>-0.803619</td>
<td>-0.80356676</td>
<td>-0.8036192</td>
<td>-0.80254846</td>
</tr>
<tr>
<td>g04</td>
<td>-0.0865533</td>
<td>-0.08653586</td>
<td>-0.08655386</td>
<td>-0.086486146</td>
</tr>
<tr>
<td>g06</td>
<td>-0.9611814</td>
<td>-0.961181388</td>
<td>-0.961181388</td>
<td>-0.96524296</td>
</tr>
<tr>
<td>g08</td>
<td>0.095825</td>
<td>0.09582504</td>
<td>0.09582504</td>
<td>0.09582504</td>
</tr>
<tr>
<td>g11</td>
<td>0.7499</td>
<td>0.749900000</td>
<td>0.749900000</td>
<td>0.749900232</td>
</tr>
<tr>
<td>g12</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1000000000</td>
</tr>
</tbody>
</table>
TABLE II.
THE MEAN RESULTS OBTAINED BY D-DE WITH RESPECT TO THOSE OBTAINED BY DE, GA ON 6 BENCHMARK PROBLEMS.

<table>
<thead>
<tr>
<th>Function</th>
<th>Optimal</th>
<th>D-DE</th>
<th>DE</th>
<th>GA</th>
</tr>
</thead>
<tbody>
<tr>
<td>g02</td>
<td>-0.803619</td>
<td>-0.80150890</td>
<td>-0.79516035</td>
<td>-0.78941723</td>
</tr>
<tr>
<td>g04</td>
<td>-30665.539</td>
<td>-30665.53867</td>
<td>-30665.53867</td>
<td>-30661.64684</td>
</tr>
<tr>
<td>g06</td>
<td>-6961.814</td>
<td>-6961.81388</td>
<td>-6961.81388</td>
<td>-6923.57354</td>
</tr>
<tr>
<td>g08</td>
<td>-0.095825</td>
<td>-0.09582504</td>
<td>-0.09582504</td>
<td>-0.09582504</td>
</tr>
<tr>
<td>g11</td>
<td>0.7499</td>
<td>0.749900000</td>
<td>0.897159852</td>
<td>0.750805569</td>
</tr>
<tr>
<td>g12</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1.0000000</td>
</tr>
</tbody>
</table>

TABLE III.
THE WORST RESULTS OBTAINED BY D-DE WITH RESPECT TO THOSE OBTAINED BY DE, GA ON 6 BENCHMARK PROBLEMS.

<table>
<thead>
<tr>
<th>Function</th>
<th>Optimal</th>
<th>D-DE</th>
<th>DE</th>
<th>GA</th>
</tr>
</thead>
<tbody>
<tr>
<td>g02</td>
<td>-0.803619</td>
<td>-0.79253455</td>
<td>-0.77041284</td>
<td>-0.75131833</td>
</tr>
<tr>
<td>g04</td>
<td>-30665.539</td>
<td>-30665.53867</td>
<td>-30665.53867</td>
<td>-30655.21039</td>
</tr>
<tr>
<td>g06</td>
<td>-6961.814</td>
<td>-6961.81388</td>
<td>-6887.72373</td>
<td>-6887.72373</td>
</tr>
<tr>
<td>g08</td>
<td>-0.095825</td>
<td>-0.09582504</td>
<td>-0.09582504</td>
<td>-0.09582504</td>
</tr>
<tr>
<td>g11</td>
<td>0.7499</td>
<td>0.749900000</td>
<td>1.000000000</td>
<td>0.75040572</td>
</tr>
<tr>
<td>g12</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1.0000000</td>
</tr>
</tbody>
</table>

B. Comparison with Respect to Some State-of-the-art Approaches on 6 Benchmark Test Problems

In this section, we present the experimental results in detail and compare D-DE with respect to state-of-the-art algorithms. The experimental results are given in Table IV. 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.

According to the summary of statistical results of test problems g04, g06, g08, g11, g12 given in Table IV, it is clearly seen that D-DE, A-DDE [19], COPSO [6] and SRES [18] all can find the optimum or near-optimum, when D-DE uses 275,000 FES, A-DDE 180,000 FES, COPSO 350,000 FES and SRES 500,000 FES, respectively. For test problem g02, the mean, worst and standard derivation of values obtained by D-DE are the best when compared with other algorithms, while the median value obtained by D-DE is better than that of A-DDE and SRES, and is slightly worse than that of COPSO. Besides, for test problem g02, the best value obtained by D-DE is slightly worse than that of the other algorithms. As shown in Table V, the best, median, mean, worst and standard derivation of values obtained by D-DE when set to 550,000 FES are obviously better than those when set to 275,000 FES. The best, median values obtained by D-DE when set to 550,000 FES are almost convergent to the optimum or near-optimum. Therefore, for test problem g02, D-DE is not still convergent to the optimum when set to 275,000 FES. In conclusion, the performance of D-DE is stable and better than or not worse than some state-of-the-art evolutionary algorithms on a set of test problems.

TABLE IV.
COMPARISON D-DE WITH RESPECT TO ALGORITHMS A-DDE [19], COPSO [6], SRES [18] ON 6 BENCHMARK TEST FUNCTIONS

<table>
<thead>
<tr>
<th>Function</th>
<th>Optimal</th>
<th>Method</th>
<th>Best</th>
<th>Median</th>
<th>Mean</th>
<th>Worst</th>
<th>Std</th>
<th>FES</th>
</tr>
</thead>
<tbody>
<tr>
<td>g02</td>
<td>-0.803619</td>
<td>D-DE</td>
<td>-0.80356676178</td>
<td>-0.801508902296</td>
<td>-0.801508902296</td>
<td>-0.79253454688</td>
<td>4.00E-03</td>
<td>275,000</td>
</tr>
<tr>
<td>g04</td>
<td>-30665.539</td>
<td>A-DDE</td>
<td>-0.79253454688</td>
<td>-0.79253454688</td>
<td>-0.79253454688</td>
<td>-0.79253454688</td>
<td>-0.79253454688</td>
<td>275,000</td>
</tr>
<tr>
<td>g06</td>
<td>-6961.814</td>
<td>COPSO</td>
<td>-0.77041284</td>
<td>-0.7710900</td>
<td>-0.7710900</td>
<td>-0.786566</td>
<td>4.00E-03</td>
<td>350,000</td>
</tr>
<tr>
<td>g08</td>
<td>-0.095825</td>
<td>SRES</td>
<td>-0.757159582</td>
<td>-0.757159582</td>
<td>-0.757159582</td>
<td>-0.757159582</td>
<td>-0.757159582</td>
<td>3.00E-03</td>
</tr>
<tr>
<td>g11</td>
<td>0.7499</td>
<td>D-DE</td>
<td>-1.000000</td>
<td>-1.000000</td>
<td>-1.000000</td>
<td>-1.000000</td>
<td>-1.000000</td>
<td>4.00E-03</td>
</tr>
<tr>
<td>g12</td>
<td>-1</td>
<td>A-DDE</td>
<td>-1.000000</td>
<td>-1.000000</td>
<td>-1.000000</td>
<td>-1.000000</td>
<td>-1.000000</td>
<td>4.00E-03</td>
</tr>
</tbody>
</table>

TABLE V.
EXPERIMENTAL RESULTS OBTAINED BY D-DE WHEN FES=275,000, FES=550,000 FOR TEST FUNCTION g02 OVER 30 RUNS

<table>
<thead>
<tr>
<th>FES</th>
<th>Best</th>
<th>Median</th>
<th>Mean</th>
<th>Worst</th>
<th>Std</th>
</tr>
</thead>
<tbody>
<tr>
<td>275,000</td>
<td>-0.80356676178</td>
<td>-0.801508902296</td>
<td>-0.79253454688</td>
<td>4.00E-03</td>
<td></td>
</tr>
<tr>
<td>550,000</td>
<td>-0.80361009279</td>
<td>-0.80360588090</td>
<td>-0.802935868229</td>
<td>-0.79259834414</td>
<td>2.55E-03</td>
</tr>
</tbody>
</table>

C. Convergence Graphs Obtained by D-DE for 6 Benchmark Test Problems

© 2010 ACADEMY PUBLISHER
In order to provide a more intuitive comprehension, we present the convergence graphs obtained by D-DE for test problems \( g_{02}, g_{04}, g_{06}, g_{08}, g_{11} \) and \( g_{12} \). Figures 4-9 depict the convergence graphs for test problems \( g_{02}, g_{04}, g_{06}, g_{08}, g_{11} \) and \( g_{12} \), respectively. It is clearly seen that D-DE has a trend to find the optimum solution for test problem \( g_{02} \) within 300,000 FES, that D-DE can find the optimum solution for each test problem \( g_{06}, g_{08}, g_{11}, g_{12} \) within 50,000 FES, and that D-DE can obtain the optimum solution for test problem \( g_{04} \) within 100,000 FES.

Figure 4. Convergence graph for \( g_{02} \).

Figure 5. Convergence graph for \( g_{04} \).

Figure 6. Convergence graph for \( g_{06} \).

Figure 7. Convergence graph for \( g_{08} \).

Figure 8. Convergence graph for \( g_{11} \).

Figure 9. Convergence graph for \( g_{12} \).
VII. CONCLUSIONS AND FUTURE WORK

In this study, we present a dynamic differential evolution algorithm (D-DE) for solving constrained real-parameter optimization problems. In this model of D-DE, there exist at least three important contributions as follows:

1) The first contribution is the novel mutation scheme, which can improve the convergence speed, prevent premature and preserve the diversity of solutions.

2) The second contribution is two important control parameters (i.e., the scale factor $F$ and the crossover probability $CR$), which are dynamic and beneficial for adjusting control parameters during the evolutionary and search process, especially, when done without any user interaction.

3) The third contribution is that D-DE can prevent premature and enhance the search performance mainly due to replacing some relatively worse solutions with reinitialized solutions during the evolutionary process.

In addition, D-DE employs orthogonal design method to generate initial population to improve the diversity of solutions and introduces a constraint handling technique based on the feasibility rule and the sum of constraints violation.

Finally, D-DE is tested on 6 benchmark test functions provided by the CEC 2006 special session on constrained real-parameter optimization. Through comparing D-DE with respect to state-of-the-art evolutionary algorithms, the experimental results show that D-DE is highly competitive and can obtain good results in terms of a test set of constrained real-parameter optimization problems. However, in the future, there are still many aspects to do. Firstly, in order to further validate D-DE, we are considering of the possibility of testing more benchmark test functions (especially, highly dimensional problems) and real-world constrained optimization problems. Secondly, for some test functions, there exists the phenomenon of slow evolutionary at the later stage. In order to overcome the limitation, we will incorporate some local search techniques into D-DE to improve the convergence speed. Additionally, improving constraint handling technique is another future work.

REFERENCES


Youyun Ao was born in 1973. He received his B.S. degree in Computer Science from Jiangxi Normal University, Nanchang, China in 1999. He received his M.S. degree in Computer Software and Theory from Shanghai Normal University, Shanghai, China in 2006. He is currently a lecturer in Computer Science at Anqing Teachers College, Anqing, Anhui, China. His research interests include evolutionary computation, intelligent information processing, etc.