Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2014, Article ID 294703, 15 pages
http://dx.doi.org/10.1155/2014/294703

Research Article
Composite Differential Search Algorithm

Bo Liu

School of Physical Education, Northeast Normal University, Changchun 130117, China

Correspondence should be addressed to Bo Liu; liubo19906@gmail.com

Received 18 May 2014; Accepted 30 July 2014; Published 21 August 2014

Academic Editor: Zhijun Liu

Differential search algorithm (DS) is a relatively new evolutionary algorithm inspired by the Brownian-like random-walk movement which is used by an organism to migrate. It has been verified to be more effective than ABC, JDE, JADE, SADE, EPSDE, GSA, PSO2011, and CMA-ES. In this paper, we propose four improved solution search algorithms, namely "DS/rand/1," "DS/rand/2," "DS/current to rand/1," and "DS/current to rand/2" to search the new space and enhance the convergence rate for the global optimization problem. In order to verify the performance of different solution search methods, 23 benchmark functions are employed. Experimental results indicate that the proposed algorithm performs better than, or at least comparable to, the original algorithm when considering the quality of the solution obtained. However, these schemes cannot still achieve the best solution for all functions. In order to further enhance the convergence rate and the diversity of the algorithm, a composite differential search algorithm (CDS) is proposed in this paper. This new algorithm combines three new proposed search schemes including "DS/rand/1," "DS/rand/2," and "DS/current to rand/1" with three control parameters using a random method to generate the offspring. Experiment results show that CDS has a faster convergence rate and better search ability based on the 23 benchmark functions.

1. Introduction

Optimization problems play an important role in both engineering design fields and the information theory. During the past decade, many researchers have developed different kinds of optimization computation algorithms to handle optimization problems, such as simulated annealing (SA), genetic algorithm (GA), differential evolution algorithm (DE), particle swarm optimization algorithm (PSO), ant colony optimization (ACO), biogeography based optimization (BBO), and differential search algorithm (DS) [1–7]. These algorithms have been adopted by researches so far and have been applied to solve many practical optimization problems such as pattern recognition, antenna design, and chaotic system [8–13].

Recently, differential search algorithm (DS) developed by Civicioglu [7] is a population-based heuristic evolutionary algorithm inspired by the Brownian-like random-walk movement which is used by an organism to migrate. This algorithm has been used to find the optimal solution in numerous practical navigational, geodetic, and astro-geodetic problems. In the paper [7], the statistical tests realized for the comparison of performances indicate that the problem-solving success of DS algorithm in global optimization problem is better than the success of the algorithms ABC [14], JDE [15], JADE [16], SADE [17], EPSDE [18], GSA [19], PSO2011 [20], and CMA-ES [21] used in this paper. However, there are still some limitations in this algorithm. It is good at exploring the search space and locating the region of global minimum, but it is slow at exploitation of the solution. Therefore, its convergence rate is also a problem in some cases. Accelerating the convergence rate and enhancing the exploitation ability of the algorithm have become two important problems and goals in the algorithm research. However, this field of study is still in its early days and a large number of future researches are necessary in order to develop the effective algorithm for optimization problems. Particularly, within our knowledge, there is almost no paper concerning an improved heuristic method for the DS algorithm.

In this paper, inspired by the mutation operation of the DE algorithm, we propose four improved solution search schemes to search the new space and enhance the convergence rate of the original algorithm. However, in some cases, these four improved solution search schemes are trapped in local optimal solutions and they cannot find the best solutions. In order to balance the exploration and exploitation of the original algorithm, this paper proposes a high-efficiency composite DS algorithm (CDS). The new algorithm combines three new proposed search schemes with three control parameters in a random method to generate the offspring. Experiments have been conducted on 23 benchmark functions chosen from previous literatures. Experimental results indicate that our approach is effective and efficient. Compared with different search schemes, CDS performs better, or at least comparably, in terms of the quality of the final solutions and the convergence rate.

The rest of this paper is organized as follows. In Section 2 we will review the basic DS. The proposed method is reviewed in Section 3, respectively. Benchmark problems and corresponding experimental results are given in Section 4. In the last section we conclude this paper and point out some future research directions.

2. Differential Search Algorithm

Differential search algorithm (DS) developed by Civicioglu [7] is one of the most superior evolutionary algorithms. The differential search algorithm is inspired by migration of living beings which constitute superorganisms during climate change of the year. In DS algorithm, the search space is simulated as the food areas and each point in the search space corresponds to an artificial-superorganism migration. The goal of this migration is to find the global optimal solution of the problem. During this process, the artificial-superorganism checks which randomly selected positions can be retained temporarily. If such a tested position is suitable to be retained for some time, the artificial-superorganism uses this migration model to settle at the discovered position and then continues its migration from this position on. Main steps of the DS algorithm are listed below.

The algorithm begins with a randomly initiated artificial-organism which utilizes NP $*$ D-dimension parameter vector within constrains by the prescribed minimum and maximum bounds as follows:

$$\begin{aligned}\vec{X}_{\min} &= \{x_{1,\min}, x_{2,\min}, \dots, x_{D,\min}\},\\ \vec{X}_{\max} &= \{x_{1,\max}, x_{2,\max}, \dots, x_{D,\max}\}.\end{aligned} \tag{1}$$

Therefore, we may generate the jth component of the ith vector as

$$x_{j,i,0} = x_{j,\min} + \text{rand}_{i,j}[0,1] \cdot \left(x_{j,\max} - x_{j,\min}\right), \tag{2}$$

where $\text{rand}_{i,j}[0,1]$ is a uniform distribution random number between 0 and 1. Consider $i = 1, \dots, \text{NP}$, and $j = 1, \dots, D$.

After initialization, *stopover* vectors $s_{i,G}$ at the areas are generated between the artificial-organisms that can be described by a Brownian-like random walk model. In order to calculate the *stopover* vectors, the algorithm creates a *stopover* vector corresponding to each population individual or target vector in the current population. The method for producing the *stopover* vectors can be described as follows:

$$s_{i,G} = X_{i,G} + \text{scale} \cdot \left(X_{r_1,G} - X_{i,G}\right), \tag{3}$$

where $r_1 \in [1, \dots, \text{NP}]$ are randomly chosen integers, and $r_1 \neq i$. Scale controlled the size of change in the positions of the individuals of the artificial-organisms. Note that the value of *scale* is generated by a gamma random number generator controlled by a uniform distribution random number between 0 and 1.

The search process of *stopover site* can be calculated by the individuals of the artificial organisms of the superorganism. This process can be described as follows:

$$s'_{i,j,G} = \begin{cases} s_{i,j,G} & \text{if } r_{i,j} = 0 \\ X_{i,j,G} & \text{if } r_{i,j} = 1, \end{cases} \tag{4}$$

where $j = [1, \dots, D]$; $r_{i,j}$ is an integer number either 1 or 0; $s'_{i,j,G}$ denotes the trail vector of the jth particle in the ith dimension at the Gth iteration.

Selection operation is used to choose the next population (i.e., $G = G + 1$) between the *stopover site* population and the artificial-organism population. The selection operation is described as

$$\begin{aligned} X_{i,G+1} &= s_{i,G}, \quad \text{if } f\left(s'_{i,G}\right) \le f\left(X_{i,G}\right),\\ &= X_{i,G}, \quad \text{if } f\left(s'_{i,G}\right) > f\left(X_{i,G}\right). \end{aligned} \tag{5}$$

The standard differential search algorithm can be described as in Procedure 1.

3. Improved Approach

3.1. Proposed IDS Algorithm. As we know, differential evolution is a simple yet efficient evolutionary algorithm, first introduced by Storn and Price [22]. Differential evolution algorithm has captured much attention and has been applied to solve many real-world problems. The crucial idea behind DE is a scheme for producing trial vectors according to the manipulation of target vector and difference vector. DE algorithm combines simple arithmetical operators with the classical operators of crossover, mutation, and selection to generate a new population. Among these operators, mutation part employs the mutation operation to produce a mutant vector with respect to each individual in the current population. Different kinds of strategies of DE have been proposed based on the target vector selected and the number of difference vectors used. In the standard DE algorithm, four differential mutation strategies can be used with one of two different crossover methods. They are listed in the following:

"DE/rand/1"

$$V_{i,G} = X_{r_1,G} + F \cdot \left(X_{r_2,G} - X_{r_3,G}\right); \tag{6}$$

"DE/rand/2"

$$V_{i,G} = X_{r_1,G} + F \cdot \left(X_{r_2,G} - X_{r_3,G}\right) + F \cdot \left(X_{r_4,G} - X_{r_5,G}\right); \tag{7}$$

"DE/current-to-rand/1"

$$V_{i,G} = X_{i,G} + F \cdot \left(X_{r_1,G} - X_{i,G}\right) + F \cdot \left(X_{r_2,G} - X_{r_3,G}\right); \tag{8}$$

"DE/current-to-rand/2"

$$\begin{aligned} V_{i,G} = X_{i,G} &+ F \cdot \left(X_{r_1,G} - X_{i,G}\right)\\ &+ F \cdot \left(X_{r_2,G} - X_{r_3,G} + X_{r_4,G} - X_{r_5,G}\right), \end{aligned} \tag{9}$$

```
(1) begin
(2)   Set the generation counter $G = 0$; and randomly initialize a population of
      NP $*$ $D$ individuals $X_i$. Initialize the parameter $p1$, $p2$
(3)   Evaluate the fitness for each individual in $P$.
(4)   while stopping criteria is not satisfied do
(5)         scale = rand$g$(2 $*$ rand) $*$ (rand-rand)
(6)      for $i = 1$ to NP do
(7)           select randomly $a \neq i$
(8)           $s_i = x_i + \text{scale} \times (x_a - x_i)$
(9)      end
(10)     $r$ = rand (NP, $D$);
(11)     If rand < rand then
(12)             If rand < $p1$ then
(13)             for $i = 1$ to NP do
(14)                 $r(i,:) = r(i,:) <$ rand
(15)             end
(16)        else
(17)           for $i = 1$ to NP do
(18)                 $r(i, \text{randi}(D)) = 0$
(19)             end
(20)           end
(21)        else
(22)           for $i = 1$ to NP do
(23)                 $d = \text{randi}(D, 1, \lceil p2 \cdot \text{rand} \rceil)$
(24)                  for $j = 1$ to size $(d, 2)$ do
(25)                       $r(i, d(j)) = 0$
(26)                  end
(27)           end
(28)        end
(29)        $r = r > 0$;
(30)        $s(r) = X(r)$;
(31)     for $i = 1$ to NP do
(32)           Evaluate the offspring $s_i$
(33)           If $s_i$ is better than $X_i$ then
(34)                 $X_i = s_i$
(35)           end if
(36)     end for
(37)     Memorize the best solution achieved so far
(38)  end while
(39) end
```

Procedure 1: Algorithm description of Differential search algorithm.

where $r_1, r_2, r_3, r_4, r_5 \in [1, \ldots, \text{NP}]$ are randomly chosen integers and $r_1 \neq r_2 \neq r_3 \neq r_4 \neq r_5 \neq i$. F is the scaling factor controlling the amplification of the differential evolution. $X_{\text{best},G}$ is the best individual vector with the best value in the population at generation G.

Based on DE algorithm and the property of DS, we propose the following four novel search mechanisms to improve DS:

"DS/rand/1"

$$s_{i,G} = X_{r_1,G} + \text{scale} \cdot \left(X_{r_2,G} - X_{i,G}\right); \tag{10}$$

"DS/rand/2"

$$s_{i,G} = X_{r_1,G} + \text{scale} \cdot \left(X_{r_2,G} - X_{i,G}\right) + \text{scale} \cdot \left(X_{r_4,G} - X_{r_5,G}\right); \tag{11}$$

"DS/current-to-rand/1"

$$s_{i,G} = X_{i,G} + \text{rand} \cdot \left(X_{r_1,G} - X_{i,G}\right) + \text{scale} \cdot \left(X_{r_2,G} - X_{r_3,G}\right); \tag{12}$$

"DS/current-to-rand/2"

$$\begin{aligned} s_{i,G} = X_{i,G} &+ \text{rand} \cdot \left(X_{r_1,G} - X_{i,G}\right) \\ &+ \text{scale} \cdot \left(X_{r_2,G} - X_{r_3,G} + X_{r_4,G} - X_{r_5,G}\right), \end{aligned} \tag{13}$$

where *scale* controls the size of change in the positions of the individuals of the artificial-organisms.

Similar to DE, four mutation schemes are proposed in this paper. The search methods "DS/rand/1" and "DS/rand/2" are two strategies which bear stronger exploration capabilities that can effectively maintain population diversity. Compared with other strategies, the search schemes "DS/current to rand/1" and "DS/current to rand/2" benefit from their fast convergence by guiding the evolutionary search with the random target. However, these two new strategies may lose their diversity and their global exploration abilities. Compared with the "DS/original/1," we can find the advantages of these four strategies. "DS/rand/1" and "DS/rand/2" are random enough for exploration. "DS/current to rand/1" and "DS/current to rand/2" can guide the search to a random

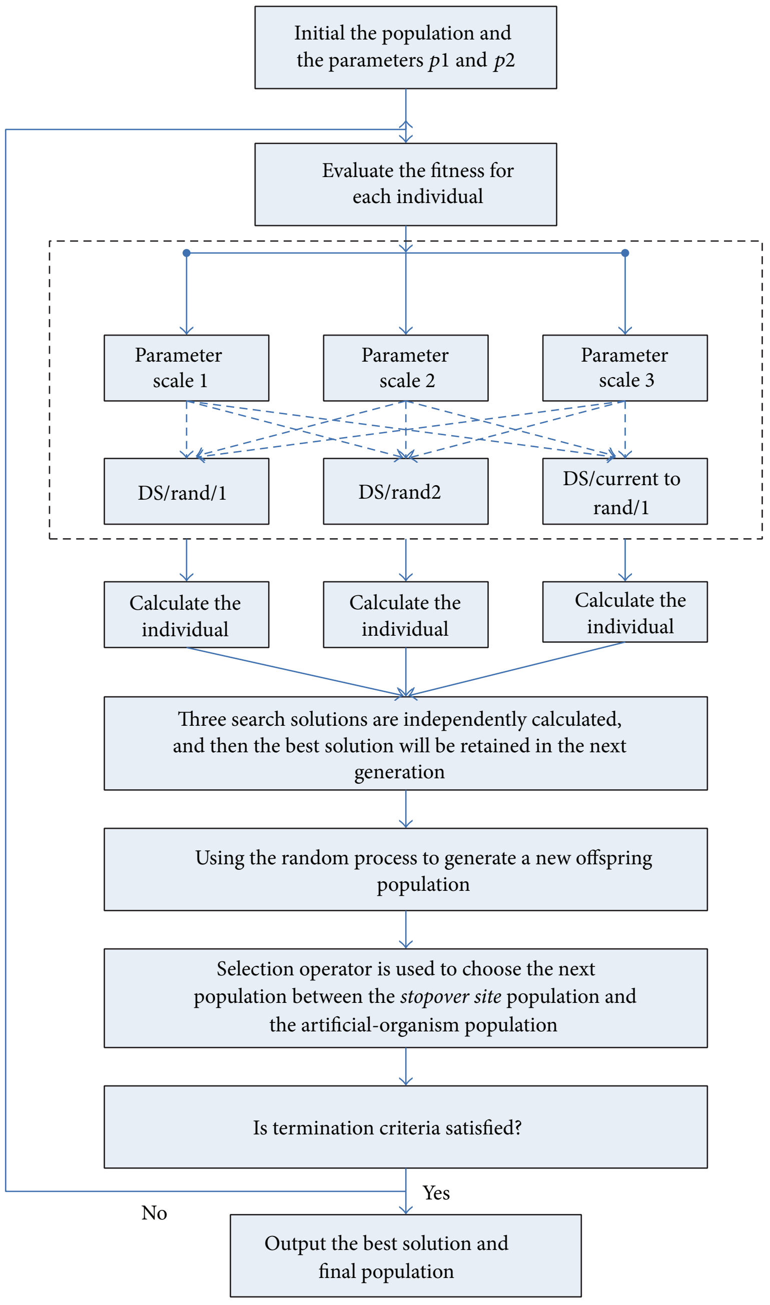
Initial the population and the parameters $p1$ and $p2$
Evaluate the fitness for each individual
Parameter scale 1
Parameter scale 2
Parameter scale 3
DS/rand/1
DS/rand2
DS/current to rand/1
Calculate the individual
Calculate the individual
Calculate the individual
Three search solutions are independently calculated, and then the best solution will be retained in the next generation
Using the random process to generate a new offspring population
Selection operator is used to choose the next population between the *stopover site* population and the artificial-organism population
Is termination criteria satisfied?
Yes
No
Output the best solution and final population

Figure 1: Flowchart of the CDS algorithm.

direction. In the experiment section, we will use different functions to test these five schemes so that we can show the effective and efficient of these strategies.

3.2. Composite DS. For successful application to optimization problems, a population-based optimization should not only find the global optimization solution but also have a faster convergence speed. Based on the experiment results of these five search schemes, we find the effectiveness of differential search algorithm in solving global numerical problem that depends on selected search schemes and its parameters. However, some different problems need different search schemes and different parameter values according to their problems. From the experiment results in Section 4, we can find that five search schemes show different advantages in various directions such as diversity and convergence rate and so on.

In order to obtain these goals and combine the advantages of these different schemes, a composite differential search algorithm (CDS) is proposed in this paper, which is used to randomly combine several search schemes and some relative parameters to produce the new offspring. The flowchart of the CDS algorithm is shown in Figure 1. In this paper, we use three search schemes and three control parameters to consist

Table 1: Benchmark functions based on our experimental study for high-dimensional.

Test function	Range	Optimum
$f_{01} = \sum_{i=1}^{n} x_i^2$	$[-100, 100]$	0
$f_{02} = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	$[-10, 10]$	0
$f_{03} = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j \right)^2$	$[-100, 100]$	0
$f_{04} = \max_i \{\lvert x_i \rvert, 1 \le i \le D\}$	$[-100, 100]$	0
$f_{05} = \sum_{i=1}^{D-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	$[-30, 30]$	0
$f_{06} = \sum_{i=1}^{D} (\lfloor x_i + 0.5 \rfloor)^2$	$[-100, 100]$	0
$f_{07} = \sum_{i=1}^{D} i x_i^4 + \text{random}[0, 1)$	$[-1.28, 1.28]$	0
$f_{08} = \sum_{i=1}^{D} -x_i \sin\left(\sqrt{\lvert x_i \rvert}\right)$	$[-500, 500]$	$-418.9829 * n$
$f_{09} = \sum_{i=1}^{D} [x_i^2 - 10\cos(2\pi x_i) + 10]$	$[-5.12, 5.12]$	0
$f_{10} = -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos 2\pi x_i\right) + 20 + e$	$[-32, 32]$	0
$f_{11} = \frac{1}{400}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$[-600, 600]$	0
$f_{12} = \frac{\pi}{D}\left\{10\sin^2(\pi y_i) + \sum_{i=1}^{D-1}(y_i - 1)^2\left[1 + 10\sin^2(\pi y_i + 1)\right] + (yD - 1)^2 + \sum_{i=1}^{D} u(x_i, 10, 100, 4)\right\}$ $y_i = 1 + \frac{x_i + 1}{4}$ $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m & x_i > a \\ 0 & -a < x_i < a \\ k(-x_i - a)^m & x_i < -a \end{cases}$	$[-50, 50]$	0
$f_{13} = 0.1\left\{10\sin^2(\pi y_i) + \sum_{i=1}^{D-1}(y_i - 1)^2\left[1 + 10\sin^2(\pi y_i + 1)\right] + (yD - 1)^2\right\} + \sum_{i=1}^{D} u(x_i, 10, 100, 4)$	$[-50, 50]$	0

a pool to improve the global search ability and enhance the convergence rate. In the algorithm, three search solutions are independently calculated and then the best solution will be retained in the next generation. The chosen three search schemes are describe as follows:

"DS/rand/1,"

"DS/rand/2,"

"DE/current-to-rand/1."

The values of *scale* are

$$\text{scale1} = \text{rand}g\,(2 * \text{rand}) * (\text{rand} - \text{rand})$$
$$\text{scale2} = \text{rand}g\,(3 * \text{rand}) * (\text{rand} - \text{rand}) \qquad (14)$$
$$\text{scale3} = \text{rand}g\,(4 * \text{rand}) * (\text{rand} - \text{rand}).$$

4. Experimental Results

To evaluate the performance of our algorithm, we applied it to 23 standards benchmark functions in [23]. These functions have been widely used in the literature. Since we do not make any modification of these functions, they are given in Table 1. The first seven functions are unimodal functions. The f_{06} is the step function which has one minimum and is discontinuous. Function f_{07} is a noisy quadratic function. The following seven functions are multimodal test functions. For these functions, the number of local minima increases exotically with the problem dimensions. Then, ten multimodal test functions with fixed dimension which have only a few local search minima are used in our experimental study. Tables 1 and 2 have shown the details of these functions. So far, these problems have been widely used as benchmarks for study with different methods by many researchers.

The algorithm is coded in MATLAB 7.9 and experiments are made on a Pentium 3.0 GHz processor with 4.0 GB of memory.

In this experiment, we set the number of particles to be 100, and we set the p1 and p2 to be 0.3 $*$ rand. In this strategy, all vectors for the update rule are selected from the population at random and, then, it has no bias to any special search directions and it chooses new search

TABLE 2: Benchmark functions based on our experimental study for fixed function.

Test function	D	Range	Optimum
$f_{14} = \left[\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2}(x_i - a_{ij})^6}\right]^{-1}$	2	$[-65.53, 65.53]$	0.998004
$f_{15} = \sum_{i=1}^{11} \left[a_i - \frac{x_1(b_i^2 + b_i x_i)}{b_i^2 + b_1 x_3 + x_4}\right]^2$	4	$[-5, 5]$	0.0003075
$f_{16} = 4x_1^2 - 2.1x_i^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	2	$[-5, 5]$	−1.0316285
$f_{17} = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos x_1 + 10$	2	$[-5, 10] * [0, 15]$	0.398
$f_{18} = \left[1 + (x_1 + x_2 + 1)^2 (19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)\right] \times \left[30 + (2x_1 - 3x_2)^2 (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)\right]$	2	$[-5, 5]$	3
$f_{19} = -\sum_{i=1}^{4} c_i \exp\left(-\sum_{j=1}^{3} a_{ij}(x_j - p_{ij})^2\right)$	3	$[0, 1]$	−3.86
$f_{20} = -\sum_{i=1}^{4} c_i \exp\left(-\sum_{j=1}^{6} a_{ij}(x_j - p_{ij})^2\right)$	6	$[0, 1]$	−3.32
$f_{21} = -\sum_{i=1}^{5} \left[(X - a_i)(X - a_i)^T + c_i\right]^{-1}$	4	$[0, 10]$	−10.1532
$f_{22} = -\sum_{i=1}^{7} \left[(X - a_i)(X - a_i)^T + c_i\right]^{-1}$	4	$[0, 10]$	−10.4029
$f_{23} = -\sum_{i=1}^{10} \left[(X - a_i)(X - a_i)^T + c_i\right]^{-1}$	4	$[0, 10]$	−10.5364

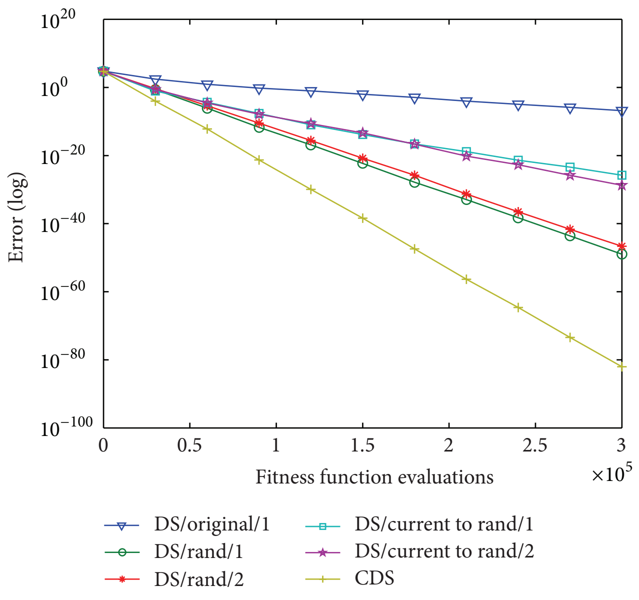

FIGURE 2: Comparison of performance of six algorithms for minimization of f_{01} with dimension 30.

directions in a random manner. The maximum number of fitness function evaluations is *100000*, *300000*, and *500000* for f_1–f_{13} with *10*, *30*, and *50* dimensions, respectively, and is *10000* for f_{14}–f_{23}. For all test functions, the algorithms carry out 30 independent runs each starting from a random population with different random seeds.

4.1. Comparison of Different Search Schemes. To investigate the performance of the different search schemes employed on the effectiveness of the differential search algorithm, five search schemes are proposed in the original DS. Four schemes, namely, DS/original/1, DS/rand/1, DS/rand/2, DS/current to rand/1, and DS/current to rand/2 are used in our experiments. These functions were studied at $D = 10$, $D = 30$, and $D = 50$. Some representative convergence graphs are shown in Figures 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, and 13. As can be seen in Table 3, for the 10D problem, it is interesting to note that DS/rand/1 outperforms DS/original/1 on thirteen functions (f_{01}–f_{13}). The DS/rand/1 can find the global optimization value on 6 functions (f_{06}, f_{08}, f_{09}, f_{10}, f_{11}, f_{12}, and f_{13}). On three functions (f_{01}, f_{02}, and f_{07}), the DS/rand/1 can find the nearest global optimization solution. For the rest of the problems, the DS/rand/1 cannot find the best solutions within the maximum function evaluation. Compared with the DS/original/1, DS/rand/2 can give a better solution on all functions. This scheme also can find the global optimization value on six functions (f_{06}, f_{08}, f_{09}, f_{10}, f_{11}, f_{12}, and f_{13}). But this method cannot beat the DS/rand/1. For f_{05}, the DS/rand/2 search scheme can provide a better solution than the DS/rand/1 method. For f_{01}, f_{02}, f_{03}, f_{04}, and f_{06}, the DS/rand/1 search scheme can own better search performance than DS/rand/2. For the DS/current to rand/1 and the DS/current to rand/2 schemes, these two schemes outperform other search schemes. The experiment results are shown in Table 4 for 30D; as can be seen in Table 4, DS/rand/1 can provide the highest accuracy on functions f_{01}, f_{02}, f_{04}, and f_{05}. For f_{03}, the DS/current to rand/2 can obtain better solutions. For f_{06}, all search schemes can find the optimal solution. The search method DS/current to rand/1 can perform better on function f_{07}. For multimodal functions

Table 3: Comparisons of different search schemes for 10*D*.

F	DS/original/1 Mean (std)	DS/rand/1 Mean (std)	DS/rand/2 Mean (std)	DS/current to rand/1 Mean (std)	DS/current to rand/2 Mean (std)
f_{01}	2.3558*e* − 010 (2.8234*e* − 010)	**4.1354e − 046 (7.6284e − 046)**	4.9124*e* − 043 (6.9672*e* − 043)	2.8575*e* − 037 (7.4424*e* − 037)	8.4320*e* − 036 (8.0751*e* − 036)
f_{02}	1.0358*e* − 006 (9.0345*e* − 007)	**2.7832e − 026 (2.5461e − 026)**	2.6303*e* − 024 (2.1985*e* − 024)	6.5910*e* − 021 (7.0243*e* − 021)	9.4820*e* − 020 (6.9772*e* − 020)
f_{03}	1.6872 (1.3868)	0.2733 (0.2685)	0.4707 (0.2919)	0.0141 (0.0199)	**0.0128 (0.0137)**
f_{04}	0.2350 (0.0706)	3.2952*e* − 005 (1.6684*e* − 005)	5.5161*e* − 005 (3.0459*e* − 005)	**1.0130e − 005 (1.0257e − 005)**	1.9844*e* − 005 (1.7062*e* − 005)
f_{05}	0.2244 (0.3258)	0.1781 (0.2966)	0.1137 (0.3661)	**0.0558 (0.0900)**	0.0889 (0.1523)
f_{06}	**0 (0)**	**0 (0)**	**0 (0)**	**0 (0)**	**0 (0)**
f_{07}	0.0052 (0.0026)	0.0022 (8.5176*e* − 004)	0.0026 (7.5113*e* − 004)	**0.0016 (8.2560e − 004)**	0.0019 (7.0409*e* − 004)
f_{08}	−4.1898*e* + 03 (0.0026)	**−4.1898e + 03 (1.8828e − 012)**	**−4.1898e + 03 (1.8828e − 012)**	**−4.1898e + 03 (1.7862e − 012)**	**−4.1898e + 03 (1.7359e − 012)**
f_{09}	0.3075 (0.4777)	**0 (0)**	**0 (0)**	**0 (0)**	**0 (0)**
f_{10}	3.5531*e* − 005 (2.6080*e* − 005)	**4.4409e − 015 (0)**	**4.4409e − 015 (0)**	**4.4409e − 015 (0)**	**4.4409e − 015 (0)**
f_{11}	0.0032 (0.0048)	**0 (0)**	**0 (0)**	1.6701*e* − 009 (4.2692*e* − 009)	7.3047*e* − 008 (1.0674*e* − 007)
f_{12}	4.2526*e* − 011 (5.4059*e* − 011)	**4.7116e − 032 (1.1332e − 047)**	**4.7116e − 032 (1.1332e − 047)**	**4.7116e − 032 (1.1332e − 047)**	**4.7116e − 032 (1.1332e − 047)**
f_{13}	7.9206*e* − 011 (6.3888*e* − 011)	**1.3498e − 032 (2.8330e − 048)**	**1.3498e − 032 (2.8330e − 048)**	**1.3498e − 032 (2.8330e − 048)**	**1.3498e − 032 (2.8330e − 048)**

Table 4: Comparisons of different schemes for 30*D*.

F	DS/original/1 Mean (std)	DS/rand/1 Mean (std)	DS/rand/2 Mean (std)	DS/current to rand/1 Mean (std)	DS/current to rand/2 Mean (std)
f_{01}	2.2615*e* − 007 (2.7476*e* − 007)	**4.8642e − 050 (5.8982e − 050)**	2.7065*e* − 046 (3.3764*e* − 046)	8.9539*e* − 022 (2.3306*e* − 021)	1.9725*e* − 029 (4.8562*e* − 029)
f_{02}	5.4923*e* − 006 (3.0819*e* − 006)	**2.9127e − 030 (2.0084e − 030)**	7.0042*e* − 028 (7.3930*e* − 028)	4.7959*e* − 018 (1.1047*e* − 017)	2.2565*e* − 020 (3.0711*e* − 020)
f_{03}	43.3209 (13.0610)	931.1284 (356.2153)	1.3430*e* + 003 (566.5445)	16.3851 (7.4977)	**15.8489 (8.3607)**
f_{04}	1.6862 (0.6287)	**0.0117 (0.0100)**	0.0142 (0.0065)	1.0787 (0.4760)	0.4105 (0.1242)
f_{05}	17.9165 (16.1629)	**7.3142 (18.4437)**	12.3902 (26.8529)	31.6675 (25.9994)	40.8267 (34.9220)
f_{06}	**0 (0)**	**0 (0)**	**0 (0)**	**0 (0)**	**0 (0)**
f_{07}	0.0104 (0.0018)	0.0080 (0.0026)	0.0087 (0.0020)	**0.0031 (8.7208e − 004)**	0.0035 (0.0011)
f_{08}	−1.2546*e* + 004 (49.1418)	**−1.2569e + 04 (1.8828e − 012)**	**−1.2569e + 04 (1.8828e − 012)**	**−1.2569e + 04 (3.5724e − 12)**	−1.2569*e* + 04 (1.0397*e* − 10)
f_{09}	5.0455 (1.7970)	**0 (0)**	**0 (0)**	**0 (0)**	**0 (0)**
f_{10}	6.0497*e* − 005 (3.0925*e* − 05)	**7.9936e − 015 (0)**	**7.9936e − 015 (0)**	5.0772*e* − 012 (8.7825*e* − 12)	1.7941*e* − 014 (4.0729*e* − 15)
f_{11}	1.7329*e* − 007 (2.2580*e* − 007)	**0 (0)**	**0 (0)**	9.8994*e* − 013 (3.8340*e* − 012)	**0 (0)**
f_{12}	5.8453*e* − 010 (1.6621*e* − 009)	**1.5705e − 032 (2.8330e − 048)**	**1.5705e − 032 (2.8330e − 048)**	1.2823*e* − 025 (2.4757*e* − 025)	1.0572*e* − 031 (2.7593*e* − 031)
f_{13}	8.7037*e* − 009 (1.1884*e* − 008)	**1.3498e − 032 (2.8330e − 048)**	**1.3498e − 032 (2.8330e − 048)**	1.1804*e* − 024 (1.5019*e* − 024)	8.1287*e* − 031 (2.4528*e* − 030)

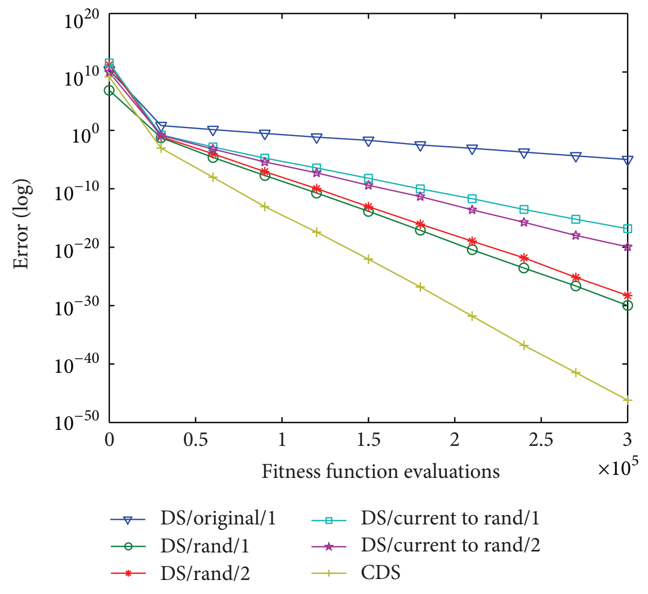

FIGURE 3: Comparison of performance of six algorithms for minimization of f_{02} with dimension 30.

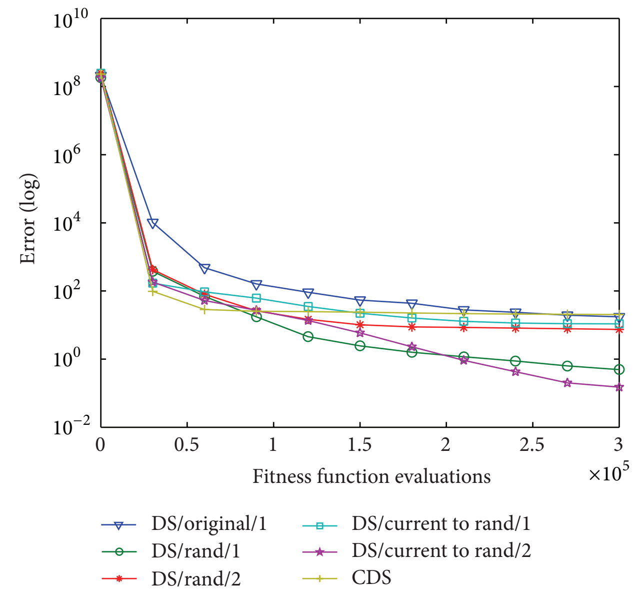

FIGURE 5: Comparison of performance of six algorithms for minimization of f_{05} with dimension 30.

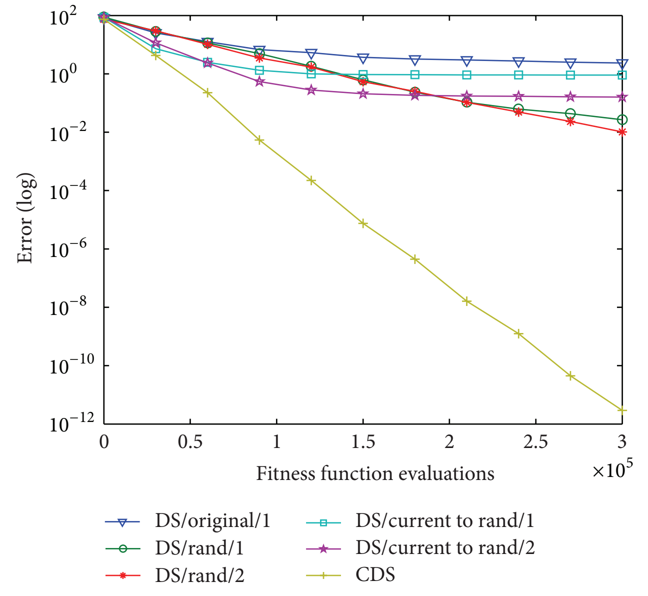

FIGURE 4: Comparison of performance of six algorithms for minimization of f_{04} with dimension 30.

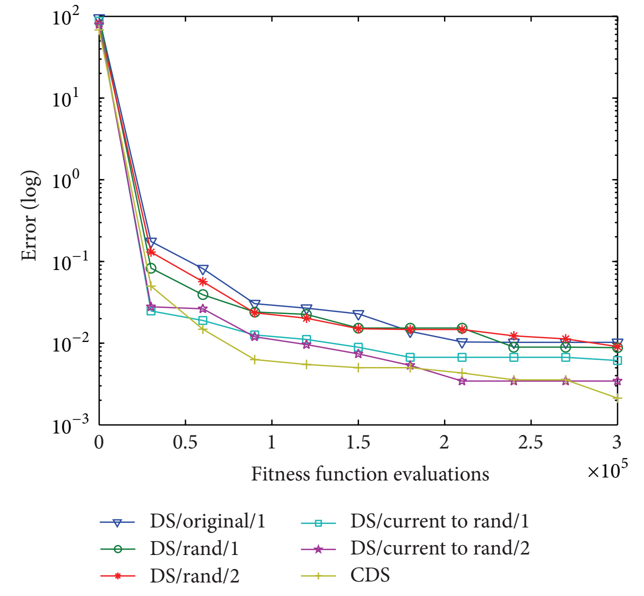

FIGURE 6: Comparison of performance of six algorithms for minimization of f_{07} with dimension 30.

f_{08}–f_{13}, the DS/rand/1 and DS/rand/2 can also find the optimal solution on these complex functions. DS/current to rand/1 and DS/current to rand/2 can provide closer to optimal solution on multimodal optimization functions,; however, they perform a little worse than DS/rand/1 and DS/rand/2. For 50D problems, the experiment results are shown in Table 5; as is shown in Table 5, while solving the unimodial optimization problem, DS/rand/1 can give a better solution than other schemes for functions f_{01}, f_{02}, and f_{05}. For f_{03} and f_{07}, DS/current to rand/2 outperforms the other algorithms, but they are a little far from the global optimums. For the f_{04}, DS/rand/2 has a better solution. For multimodal functions f_{08}–f_{13} with many local minima, the final results are more important because this function can reflect the algorithm's ability to escape from poor local optima and obtain the near-global optimum. The DS/rand/1 and DS/rand/2 provide better solutions than other algorithms except for f_{09}. As can be seen in Tables 3–5, the results show that DS/rand/1 and DS/rand/2 perform much better in most cases than other schemes.

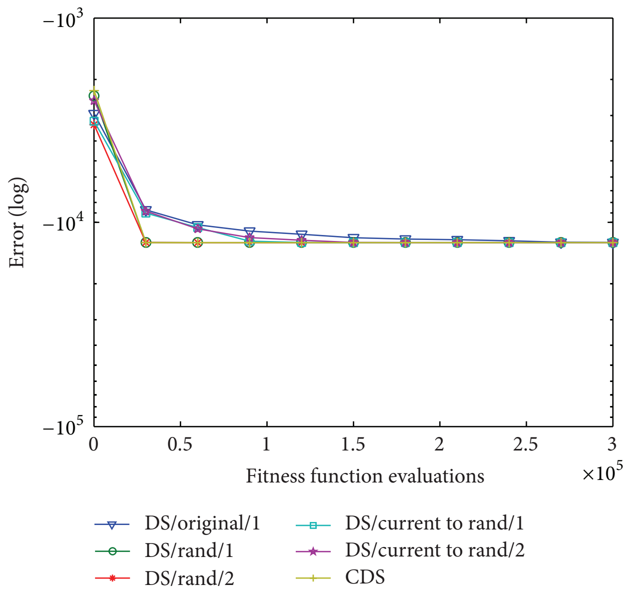

Figure 7: Comparison of performance of six algorithms for minimization of f_{08} with dimension 30.

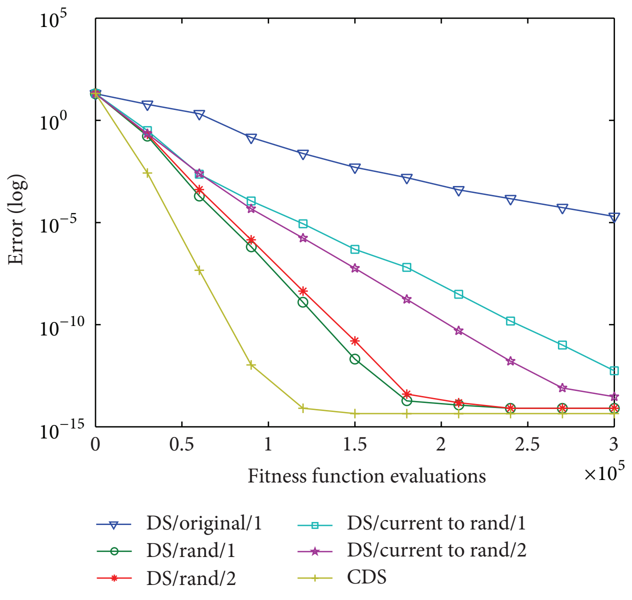

Figure 9: Comparison of performance of six algorithms for minimization of f_{10} with dimension 30.

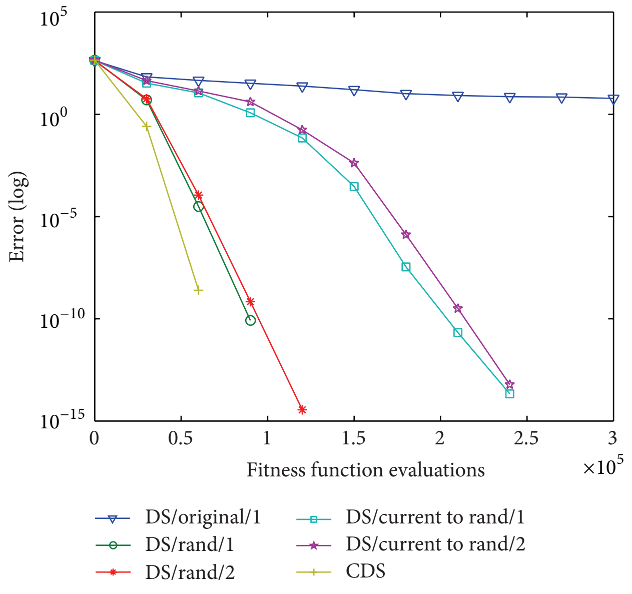

Figure 8: Comparison of performance of six algorithms for minimization of f_{09} with dimension 30.

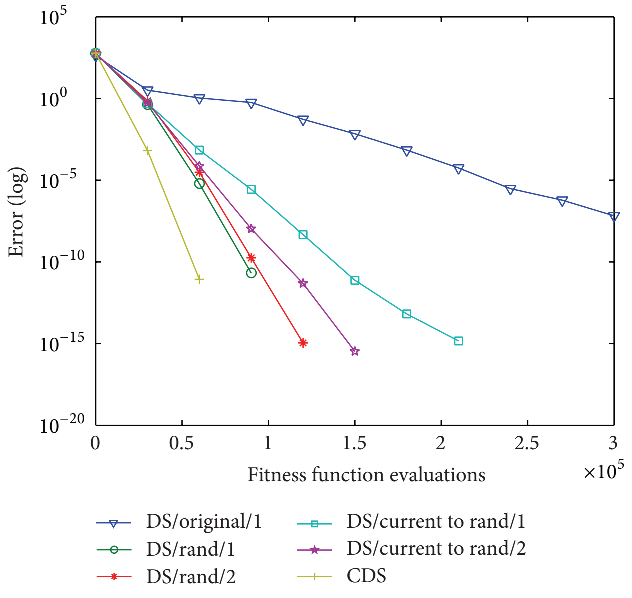

Figure 10: Comparison of performance of six algorithms for minimization of f_{11} with dimension 30.

4.2. Sensitivities to Population Size. Performance of DS is always sensitive to the selected population size. If the population is too small, the diversity of possible movements is poor and then the algorithm may be easily trapped in a local optimum. On the other hand, if the population size is too large, DS exhausts the fitness evaluations very quickly without being able to locate the optimum. Therefore, the choice of the best population size of DS is always critical for different problems.

To investigate the sensitivity of the proposed algorithm to variations of population size, some experiments are repeated for NP = 50 and NP = 150. The experimental results are given in Tables 6 and 7 for five search schemes at dimension $D = 30$. For NP = 50, the performances of DS/rand/1 and DS/rand/2 are significantly superior to that of other algorithms according to the experimental results shown in Table 6, since the DS/rand/1 and DS/rand/2 are better than other algorithms except for the f_{03} and f_{07}. For f_{12} and f_{13}, all algorithms can locate the near-global optimum over all 50 runs. When the population increases to NP = 100, DS/rand/1 and DS/rand/2 can obtain values higher than NP = 50. We can find that DS/rand/1 and DS/rand/2 are faster

Table 5: Comparisons of different schemes for $50D$.

F	DS/original/1	DS/rand/1	DS/rand/2	DS/current to rand/1	DS/current to rand/2
	Mean (std)	Mean (std)	Mean (std)	Mean (std)	Mean (std)
f_{01}	$1.6310e-005$ $(1.2662e-005)$	**1.2837e − 054 (1.7623e − 054)**	$2.2293e-050$ $(3.4395e-050)$	$4.0249e-016$ $(8.6977e-016)$	$4.9414e-024$ $(5.9120e-024)$
f_{02}	$3.0789e-005$ $(1.1049e-005)$	**1.1757e − 033 (1.1452e − 033)**	$5.6476e-031$ $(5.2712e-031)$	$5.3804e-016$ $(5.1135e-016)$	$9.7757e-019$ $(9.9765e-019)$
f_{03}	166.2852 (47.9112)	$5.0333e+03$ $(1.7113e+03)$	$8.0260e+03$ $(2.3708e+03)$	103.8671 (32.2863)	**92.9566 (25.0034)**
f_{04}	3.7643 (0.5088)	1.5477 (0.8361)	**0.9776 (1.2725)**	3.6342 (0.8480)	2.7460 (0.5450)
f_{05}	109.8978 (41.1906)	**45.0618 (37.5075)**	53.0212 (35.5623)	86.9502 (36.1520)	108.7941 (40.4535)
f_{06}	**0 (0)**	**0 (0)**	**0 (0)**	**0 (0)**	**0 (0)**
f_{07}	0.0175 (0.0042)	0.0124 (0.0026)	0.0125 (0.0024)	0.0068 (0.0022)	**0.0068 (0.0018)**
f_{08}	$-2.0696e+04$ (130.9369)	**−2.0949e + 04 (7.5313e − 12)**	**−2.0949e + 04 (7.5313e − 012)**	$-2.0949e+04$ $(7.1137e-010)$	$-2.0949e+04$ $(8.5767e-009)$
f_{09}	10.7632 (1.7425)	0.0663 (0.2569)	0.0663 (0.2569)	$3.4343e-015$ $(7.2457e-015)$	**3.5527e − 016 (1.3760e − 015)**
f_{10}	$2.6408e-004$ $(1.0382e-004)$	**8.2305e − 015 (9.1731e − 16)**	**7.9936e − 015 (0)**	$8.5383e-010$ $(6.0168e-010)$	$3.2371e-013$ $(2.5385e-013)$
f_{11}	$9.6302e-006$ $(5.9119e-006)$	**0 (0)**	**0 (0)**	0.0023 (0.0040)	$6.5715e-004$ (0.0025)
f_{12}	$9.1415e-009$ $(7.6863e-009)$	**9.4233e − 033 (1.4165e − 048)**	**9.4233e − 033 (1.4165e − 048)**	$5.0418e-020$ $(8.3674e-020)$	$1.0849e-026$ $(2.4721e-026)$
f_{13}	$4.1252e-007$ $(3.0875e-007)$	**1.3498e − 032 (2.8330e − 048)**	**1.3498e − 032 (2.8330e − 048)**	$5.4875e-017$ $(1.9201e-016)$	$9.2927e-025$ $(1.9463e-024)$

Table 6: Comparisons of different schemes with population size 50.

F	DS/original/1	DS/rand/1	DS/rand/2	DS/current to rand/1	DS/current to rand/2
	Mean (std)	Mean (std)	Mean (std)	Mean (std)	Mean (std)
f_{01}	$2.0914e-015$ $(2.6079e-015)$	**7.7011e − 102 (9.2672e − 102)**	$2.7581e-095$ $(6.0508e-095)$	$3.7710e-043$ $(1.3868e-042)$	$1.1331e-057$ $(4.1221e-057)$
f_{02}	$4.7037e-012$ $(3.7353e-012)$	**7.3457e − 061 (5.8919e − 061)**	$1.9472e-056$ $(1.9189e-056)$	$6.5250e-035$ $(1.3700e-034)$	$7.1128e-040$ $(8.8854e-040)$
f_{03}	14.8940 (7.6726)	33.0264 (16.4273)	59.7994 (37.5156)	11.0305 (4.7774)	**3.5511 (2.3166)**
f_{04}	2.4420 (0.5466)	3.5091 (2.3410)	**0.2884 (0.3909)**	3.3452 (1.3393)	1.5449 (0.5293)
f_{05}	14.7717 (23.5488)	**8.6311 (19.1644)**	18.3336 (29.3195)	43.8933 (33.3687)	52.1674 (29.2545)
f_{06}	**0 (0)**	**0 (0)**	**0 (0)**	**0 (0)**	**0 (0)**
f_{07}	0.0099 (0.0032)	0.0047 $(9.4232e-004)$	0.0060 (0.0016)	**0.0032 (0.0013)**	0.0034 (0.0012)
f_{08}	$-1.2569e+04$ $(5.0912e-009)$	$-1.2530e+04$ (57.7920)	**−1.2569e + 04 (1.8828e − 012)**	$-1.2569e+04$ $(2.5261e-012)$	$-1.2569e+04$ $(2.2278e-012)$
f_{09}	0.1329 (0.3501)	0.5970 (1.0503)	0.1990 (0.5578)	**0 (0)**	**0 (0)**
f_{10}	$2.6201e-009$ $(1.6822e-009)$	**7.2831e − 015 (1.4710e − 015)**	$7.9936e-015$ (0)	0.0621 (0.2405)	$1.1309e-014$ $(3.1396e-015)$
f_{11}	$4.9590e-016$ $(8.4121e-016)$	**0 (0)**	**0 (0)**	0.0029 (0.0114)	**0 (0)**
f_{12}	$1.0090e-018$ $(1.2563e-018)$	**1.5705e − 032 (2.8330e − 048)**	**1.5705e − 032 (2.8330e − 048)**	**1.5705e − 032 (2.8330e − 048)**	**1.5705e − 032 (2.8330e − 048)**
f_{13}	$2.8327e-017$ $(4.0493e-017)$	**1.3498e − 032 (2.8330e − 048)**	**1.3498e − 032 (2.8330e − 048)**	**2.6243e − 032 (4.9023e − 032)**	**1.3498e − 032 (2.8330e − 048)**

Table 7: Comparisons of different schemes with population size 150.

F	DS/original/1	DS/rand/1	DS/rand/2	DS/current to rand/1	DS/current to rand/2
	Mean (std)	Mean (std)	Mean (std)	Mean (std)	Mean (std)
f_{01}	1.5722e − 004 (1.2224e − 004)	**3.0028e − 032 (2.9420e − 032)**	8.9889e − 030 (7.0452e − 030)	1.8911e − 016 (2.2287e − 016)	1.0279e − 019 (2.7982e − 019)
f_{02}	5.8298e − 004 (3.7871e − 004)	**7.1419e − 020 (3.3572e − 020)**	2.7142e − 018 (1.0027e − 018)	3.3349e − 012 (5.4473e − 012)	8.9033e − 014 (9.4785e − 014)
f_{03}	64.9635 (15.6465)	2.8743e + 003 (894.1078)	4.2360e + 03 (1.0469e + 003)	21.7178 (8.9633)	**18.7298 (5.6650)**
f_{04}	1.7494 (0.3367)	**0.1340 (0.0298)**	0.1697 (0.0380)	0.5319 (0.2686)	0.2483 (0.1518)
f_{05}	23.9970 (8.9851)	18.6080 (27.1287)	**9.8861 (14.1526)**	50.9432 (31.0728)	33.1652 (26.2450)
f_{06}	**0 (0)**	**0 (0)**	**0 (0)**	**0 (0)**	**0 (0)**
f_{07}	0.0135 (0.0046)	0.0112 (0.0024)	0.0118 (0.0019)	**0.0034 (8.6636e − 004)**	0.0044 (0.0015)
f_{08}	−1.2209e + 004 (193.5436)	**−1.2569e + 04 (1.8828e − 012)**	**−1.2569e + 04 (1.8828e − 012)**	−1.2569e + 04 (9.3662e − 004)	−1.2569e + 04 (0.0354)
f_{09}	9.4868 (1.2681)	**0 (0)**	**0 (0)**	3.7046e − 011 (9.9032e − 011)	7.1731e − 008 (2.5004e − 007)
f_{10}	0.0017 (6.3671e − 004)	**8.7041e − 015 (1.4710e − 015)**	1.4388e − 014 (3.3435e − 015)	3.0953e − 009 (3.3465e − 009)	3.3529e − 011 (2.1616e − 011)
f_{11}	8.5273e − 005 (5.6899e − 005)	**0 (0)**	0 (0)	2.0576e − 015 (7.1610e − 015)	7.5495e − 016 (2.9239e − 015)
f_{12}	1.4976e − 007 (1.2099e − 007)	**1.5705e − 032 (2.8330e − 048)**	1.3626e − 031 (1.3531e − 031)	2.9735e − 018 (4.6641e − 018)	8.8589e − 023 (1.3653e − 022)
f_{13}	4.7327e − 006 (3.8476e − 006)	**1.6374e − 032 (3.5683e − 033)**	8.2275e − 031 (1.0420e − 030)	1.5548e − 016 (3.7989e − 016)	2.1502e − 021 (3.5186e − 021)

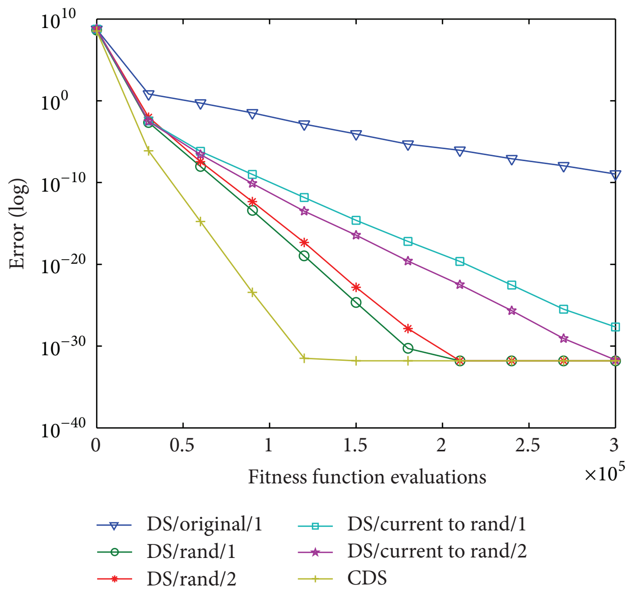

Figure 11: Comparison of performance of six algorithms for minimization of f_{12} with dimension 30.

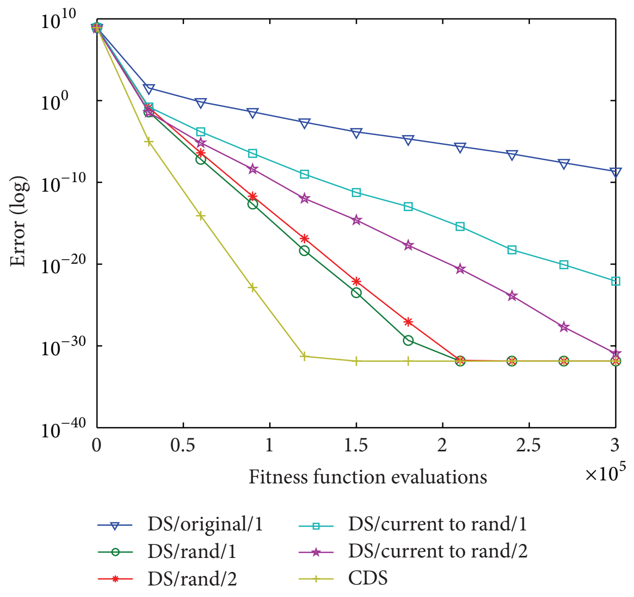

Figure 12: Comparison of performance of six algorithms for minimization of f_{13} with dimension 30.

than DS/current to rand/1 and DS/current to rand/2 on these functions. For NP = 150 in Table 7, DS/rand/1 and DS/rand/2 are able to obtain a significantly better performance than other schemes on 11 functions.

4.3. Comparison of CDS with Enhanced Differential Search Algorithm.

The performances of CDS are compared with those of original DS and DS with "DS/rand/1". In CDS, the population size is 40. The maximum number of fitness

Table 8: Comparisons of CDS with DS/original/1 and DS/rand/1.

F	D	DS/original/1 Mean (std)	DS/rand/1 Mean (std)	CDS Mean (std)
f_{01}	10	$2.3558e-010$ $(2.8234e-010)$	$4.1354e-046$ $(7.6284e-046)$	**1.6712e − 067 (3.4629e − 067)**
	30	$2.2615e-007$ $(2.7476e-007)$	$4.8642e-050$ $(5.8982e-050)$	**1.6645e − 082 (3.9759e − 082)**
	50	$1.6310e-005$ $(1.2662e-005)$	$1.2837e-054$ $(1.7623e-054)$	**4.0148e − 098 (7.8156e − 098)**
f_{02}	10	$1.0358e-006$ $(9.0345e-007)$	$2.7832e-026$ $(2.5461e-026)$	**1.8086e − 038 (1.7428e − 038)**
	30	$5.4923e-006$ $(3.0819e-006)$	$2.9127e-030$ $(2.0084e-030)$	**2.0469e − 047 (1.2285e − 047)**
	50	$3.0789e-005$ $(1.1049e-005)$	$1.1757e-033$ $(1.1452e-033)$	**2.2422e − 055 (2.1011e − 055)**
f_{03}	10	1.6872 (1.3868)	0.2733 (0.2685)	**6.0309e − 006 (1.1497e − 005)**
	30	43.3209 (13.0610)	931.1284 (356.2153)	**2.8232 (2.0270)**
	50	166.2852 (47.9112)	$5.0333e+03$ $(1.7113e+03)$	**56.4185 (52.1054)**
f_{04}	10	0.2350 (0.0706)	$3.2952e-005$ $(1.6684e-005)$	**2.0296e − 013 (2.1747e − 013)**
	30	1.6862 (0.6287)	0.0117 (0.0100)	**3.3495e − 011 (5.1641e − 011)**
	50	3.7643 (0.5088)	1.5477 (0.8361)	**0.0829 (0.0980)**
f_{05}	10	0.2244 (0.3258)	0.1781 (0.2966)	**0.0430 (0.1053)**
	30	17.9165 (16.1629)	7.3142 (18.4437)	**18.5197 (5.3463)**
	50	109.8978 (41.1906)	45.0618 (37.5075)	**65.8609 (25.7663)**
f_{06}	10	**0 (0)**	0 (0)	**0 (0)**
	30	**0 (0)**	0 (0)	**0 (0)**
	50	**0 (0)**	0 (0)	**0 (0)**
f_{07}	10	0.0052 (0.0026)	0.0022 $(8.5176e-004)$	**0.0015 (3.2425e − 004)**
	30	0.0104 (0.0018)	0.0080 (0.0026)	**0.0031 (9.0754e − 004)**
	50	0.0175 (0.0042)	0.0124 (0.0026)	**0.0039 (0.0011)**
f_{08}	10	$-4.1898e+03$ (0.0026)	$-4.1898e+03$ $(1.8828e-012)$	**−4.1898e + 03 (9.5869e − 013)**
	30	$-1.2546e+004$ (49.1418)	$-1.2569e+04$ $(1.8828e-012)$	**−1.2569e + 004 (1.9174e − 012)**
	50	$-2.0696e+04$ (130.9369)	$-2.0949e+04$ $(7.5313e-12)$	**−2.0949e + 004 (3.8348e − 012)**
f_{09}	10	0.3075 (0.4777)	0 (0)	**0 (0)**
	30	5.0455 (1.7970)	0 (0)	**0 (0)**
	50	10.7632 (1.7425)	0.0663 (0.2569)	**0 (0)**
f_{10}	10	$3.5531e-005$ (2.6080e-005)	$4.4409e-015$ (0)	**3.0198e − 015 (1.8346e − 015)**
	30	$6.0497e-005$ $(3.0925e-005)$	$7.9936e-015$ (0)	**4.4409e − 015 (0)**
	50	$2.6408e-004$ $(1.0382e-004)$	$8.2305e-015$ $(9.1731e-16)$	**4.7962e − 015 (1.1235e − 015)**

Table 8: Continued.

F	D	DS/original/1 Mean (std)	DS/rand/1 Mean (std)	CDS Mean (std)
f_{11}	10	0.0032 (0.0048)	0 (0)	**0 (0)**
	30	1.7329e − 007 (2.2580e − 007)	0 (0)	**0 (0)**
	50	9.6302e − 006 (5.9119e − 006)	0 (0)	**0 (0)**
f_{12}	10	4.2526e − 011 (5.4059e − 011)	4.7116e − 032 (1.1332e − 047)	**4.7116e − 032 (0)**
	30	5.8453e − 010 (1.6621e − 009)	1.5705e − 032 (2.8330e − 048)	**1.5705e − 032 (2.8330e − 048)**
	50	9.1415e − 009 (7.6863e − 009)	9.4233e − 033 (1.4165e − 048)	**9.4233e − 033 (1.4165e − 048)**
f_{13}	10	7.9206e − 011 (6.3888e − 011)	1.3498e − 032 (2.8330e − 048)	**1.3498e − 032 (2.8330e − 048)**
	30	8.7037e − 009 (1.1884e − 008)	1.3498e − 032 (2.8330e − 048)	**1.3498e − 032 (2.8330e − 048)**
	50	4.1252e − 007 (3.0875e − 007)	1.3498e − 032 (2.8330e − 048)	**1.3498e − 032 (2.8330e − 048)**

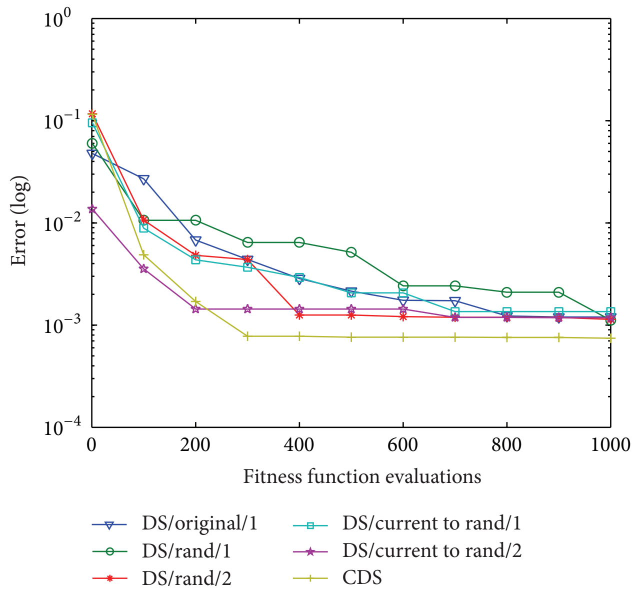

Figure 13: Comparison of performance of six algorithms for minimization of f_{15} with fixed dimension.

function evaluations is *100000*, *300000*, and *500000* for f_1–f_{13} with *10*, *30*, and *50* dimensions, respectively. The parameters $p1$ and $p2$ are set to be $0.3 * \text{rand}$. CDS can inherit the bright sides of the three search schemes. The mean and standard deviation results of CDS with other algorithms are shown in Table 8. As can be seen in Table 8, the results of CDS can obtain much better results than original DS and DS with "DS/rand/1" for all benchmarks with 10D. There is no dispute that a more precise exploitation can enhance the performance of the algorithms. For 30D problem, CDS owns a very fast convergence rate and can give a better solution than original DS and DS with "DS/rand/1" for 12 functions, except for f_{05}. For the function f_{05}, the "DS/rand/1" can provide better solutions than original DS and CDS. For 50D problems, both CDS and "DS/rand/1" could search the optimal solution on some functions (f_{06}, f_{08}, f_{09}, f_{11}, f_{12}, and f_{13}). In addition, CDS have much better performances than DS and DS with "DS/rand/1" on the functions f_{01}, f_{02}, f_{03}, f_{04}, and f_{07}. However, for the f_{05}, the "DS/rand/1" can obtain a better solution than other algorithms. Therefore, it is concluded that CDS is more effective than DS and "DS/rand/1" for high-dimensional classical benchmark functions. In particular, CDS exhibits an overall higher convergence speed and better robustness than the two competitors under some conditions. We also can conclude that the combination operator of these methods has the ability to accelerate DS, especially for the higher dimensionality. In addition, the graphs of Figures 2–12 show that CDS has improved the convergence characteristics of the original algorithm, regardless to dimension.

4.4. Comparison of CDS with Enhanced Differential Search Algorithm in Fixed Dimension. In this section, we will compare our algorithm with enhanced differential search algorithm for fixed functions. The experimental results are listed in Table 9. As can be seen in Table 9, for f_{14}, f_{16}, and f_{17}, with only a few local minima, the dimension of the function is also small. In this case, it is hard to judge the performances of individual algorithms. All algorithms were able to find optimal solutions for these two functions. For f_{15}, f_{18}, f_{19}, and f_{20}, the CDS can provide better solutions than DS and "DS/rand/1". For f_{21}–f_{23}, the CDS can provide all the better solution. The algorithm performs superiorly better than DS and DS with "DS/rand/1". The graphs of Figure 13 show that the convergence progresses of different search schemes and CDS for f_{15}.

Table 9: The mean and convergence iteration of the functions in Table 2.

F	D	DS/original/1 Mean (std)	DS/rand/1 Mean (std)	CDS Mean (std)
f_{14}	2	0.9980 (3.0790e − 009)	**0.9980 (9.0649e − 017)**	0.9980 (1.2820e − 016)
f_{15}	4	0.0013 (4.5585e − 004)	0.0012 (2.5461e − 026)	**8.7526e − 004 (1.8474e − 004)**
f_{16}	2	−1.0316 (6.5526e − 008)	−1.0316 (5.2049e − 010)	**−1.0316 (5.9300e − 015)**
f_{17}	2	0.3979 (6.2820e − 007)	0.3979 (0)	**0.3979 (0)**
f_{18}	2	3.0006 (7.2153e − 004)	3 (1.0977e − 004)	**3 (2.8723e − 008)**
f_{19}	3	−3.8628 (8.4380e − 006)	−3.8628 (3.7942e − 010)	**−3.8628 (1.1014e − 013)**
f_{20}	6	−3.3152 (0.0052)	−3.3220 (1.0041e − 005)	**−3.3220 (2.4649e − 006)**
f_{21}	4	−10.0655 (0.0847)	−10.0114 (0.2982)	**−10.1409 (0.0200)**
f_{22}	4	−10.2986 (0.1112)	−10.3320 (0.0919)	**−10.3980 (0.0063)**
f_{23}	4	−10.3675 (0.0994)	−10.4213 (0.2538)	**−10.5233 (0.0183)**

5. Conclusions

In this paper, we propose four different search schemes. Although these new schemes could not find better solutions than the original algorithm for only a few functions, these new schemes could have a faster convergence rate and better diversity than the original algorithm. In order to further enhance the exploitation of the algorithm, we combined three new schemes with three control parameters in a random method to consist a new algorithm (CDS). To verify the performance of CDS, 23 benchmark functions chosen from literature are employed. The results show that the proposed CDS algorithm clearly outperforms the basic DS and the new proposed schemes. In this paper, we only consider the global optimization. The algorithm can be extended to solve other problems such as constrained optimization problems.

Conflict of Interests

The author declares that there is no conflict of interests regarding the publication of this paper.

References

[1] B. Suman, "Study of simulated annealing based algorithms for multiobjective optimization of a constrained problem," *Computers and Chemical Engineering*, vol. 28, no. 9, pp. 1849–1871, 2004.

[2] J. Horn, N. Nafpliotis, and D. E. Goldberg, "A niched Pareto genetic algorithm for multiobjective optimization," in *Proceedings of the 1st IEEE Conference on Evolutionary Computation*, pp. 82–87, June 1994.

[3] X. Li and M. Yin, "Modified differential evolution with self-adaptive parameters method," *Journal of Combinatorial Optimization*, 2014.

[4] M. Clerc and J. Kennedy, "The particle swarm-explosion, stability, and convergence in a multidimensional complex space," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 1, pp. 58–73, 2002.

[5] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: optimization by a colony of cooperating agents," *IEEE Transactions on Systems, Man, and Cybernetics B*, vol. 26, no. 1, pp. 29–41, 1996.

[6] X. Li and M. Yin, "Multiobjective binary biogeography based optimization based feature selection for gene expression data," *IEEE Transactions on NanoBioscience*, vol. 12, no. 4, pp. 343–353, 2013.

[7] P. Civicioglu, "Transforming geocentric cartesian coordinates to geodetic coordinates by using differential search algorithm," *Computers and Geosciences*, vol. 46, pp. 229–247, 2012.

[8] X. Li and M. Yin, "Multi-operator based biogeography based optimization with mutation for global numerical optimization," *Computers & Mathematics with Applications*, vol. 64, no. 9, pp. 2833–2844, 2012.

[9] X. Li, J. Y. Wang, J. P. Zhou, and M. Yin, "A perturb biogeography based optimization with mutation for global numerical optimization," *Applied Mathematics and Computation*, vol. 218, no. 2, pp. 598–609, 2011.

[10] X. Li and M. Yin, "Application of differential evolution algorithm on self-potential data," *PLoS ONE*, vol. 7, no. 12, Article ID e51199, 2012.

[11] X. Li and M. Yin, "An opposition-based differential evolution algorithm for permutation flow shop scheduling based on diversity measure," *Advances in Engineering Software*, vol. 55, pp. 10–31, 2013.

[12] X. Li, J. N. Wang, and M. Yin, "Enhancing the performance of cuckoo search algorithm using orthogonal learning method," *Neural Computing and Applications*, vol. 24, no. 6, pp. 1233–1247, 2014.

[13] X. Li, J. Zhang, and M. Yin, "Animal migration optimization: an optimization algorithm inspired by animal migration behavior," *Neural Computing and Applications*, pp. 1–11, 2013.

[14] D. Karaboga and B. Basturk, "On the performance of artificial bee colony (ABC) algorithm," *Applied Soft Computing Journal*, vol. 8, no. 1, pp. 687–697, 2008.

[15] J. Brest, S. Greiner, B. Bošković, M. Mernik, and V. Zumer, "Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 6, pp. 646–657, 2006.

[16] J. Q. Zhang and A. C. Sanderson, "JADE: adaptive differential evolution with optional external archive," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 5, pp. 945–958, 2009.

[17] A. K. Qin, V. L. Huang, and P. N. Suganthan, "Differential evolution algorithm with strategy adaptation for global numerical optimization," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 2, pp. 398–417, 2009.

[18] R. Mallipeddi, P. N. Suganthan, Q. K. Pan, and M. F. Tasgetiren, "Differential evolution algorithm with ensemble of parameters and mutation strategies," *Applied Soft Computing Journal*, vol. 11, no. 2, pp. 1679–1696, 2011.

[19] E. Rashedi, H. Nezamabadi-pour, and S. Saryazdi, "GSA: a Gravitational Search Algorithm," *Information Sciences*, vol. 179, no. 13, pp. 2232–2248, 2009.

[20] M. G. H. Omran and M. Clerc, 2011, http://www.particleswarm.info/.

[21] N. Hansen, A. S. N. Niederberger, L. Guzzella, and P. Koumoutsakos, "A method for handling uncertainty in evolutionary optimization with an application to feedback control of combustion," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 1, pp. 180–197, 2009.

[22] R. Storn and K. Price, "Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[23] X. Yao, Y. Liu, and G. Lin, "Evolutionary programming made faster," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 2, pp. 82–102, 1999.