International Journal of Innovative Computing, Information and Control
Volume **9**, Number **3**, March **2013**

ICIC International ©2013 ISSN 1349-4198
pp. 1357–1372

GROUP-BASED DIFFERENTIAL EVOLUTION FOR NUMERICAL OPTIMIZATION PROBLEMS

Ming-Feng Han, Chin-Teng Lin, Jyh-Yeong Chang and Dong-Lin Li

Institute of Electrical Control Engineering
National Chiao Tung University
1001 University Road, Hsinchu 300, Taiwan
ming0901@gmail.com; { ctlin; jychang }@mail.nctu.edu.tw; lazybones000@yahoo.com.tw

Received January 2012; revised May 2012

Abstract. *This paper proposes a group-based differential evolution (GDE) algorithm for numerical optimization problems. The proposed GDE algorithm provides a new process using two mutation strategies to effectively enhance the search for the globally optimal solution. Initially, all individuals in the population are partitioned into an elite group and an inferior group based on their fitness value. In the elite group, individuals with a better fitness value employ the local mutation operation to search for better solutions near the current best individual. The inferior group, which is composed of individuals with worse fitness values, uses a global mutation operation to search for potential solutions and to increase the diversity of the population. Subsequently, the GDE algorithm employs crossover and selection operations to produce offspring for the next generation. This paper also proposes two parameter-tuning strategies for the robustness of the GDE algorithm in the evolution process. To validate the performance of the GDE algorithm, 13 well-known numerical benchmark functions were tested on low- and high-dimensional problems. The simulation results indicate that our approach is efficient.*

Keywords: Evolutionary algorithm (EA), Optimization, Differential evolution (DE), Adaptive strategy

1. **Introduction.**

Evolutionary algorithms (EAs) have become a popular optimization tool for global optimization problems [1-7]. The optimization process of EAs usually adopts stochastic search techniques that work with a set of individuals instead of a single individual and use certain evolution operators to naturally produce offspring for the next generation. These algorithms include genetic algorithms (GAs) [8], evolutionary programming (EP) [9], evolution strategies (ESs) [10], particle swarm optimization (PSO) [11] and differential evolution (DE) [12,13], which are well-known, effectual and classical search techniques.

In recent years, the DE algorithm has sparked the interest of researchers [14-27]. The DE algorithm, proposed by Storn and Price [12,13], is an efficient and effective global optimizer in the continuous search domain. It has been shown to perform better than genetic algorithms and particle swarm optimization with respect to several numerical benchmarks [12,13,21,28]. The DE algorithm employs the difference between two randomly selected individuals as the source of random variations for the mutation operation. Subsequently, crossover and selection operations are used for generating offspring. Many studies have applied the DE algorithm to difficult optimization problems and achieved better solutions [12,13]. However, a stagnation problem has been identified, in which the DE algorithm occasionally stops proceeding toward the global optimum [16,17]. The reason for the stagnation problem is the limitation of the mutation operation model. In the DE algorithm, the mutation operation model always favors the exploration ability or the exploitation ability, which easily results in a blind search over individual space or insufficient diversity in a population. To handle this problem, researchers have combined different learning methods to solve the stagnation problem. Rahnamayan *et al.* [25] combined an opposition-based learning method and the DE algorithm, calling the method opposition-based differential evolution (ODE). The ODE employs opposition-based optimization to choose the best solutions by simultaneously checking the fitness of the opposite solution in the current population. The ODE successfully increases the diversity of a population. A combination of one-step k-Means clustering and a multi-parent crossover operation in the DE algorithm was proposed by Cai *et al.* [14]. Their method enhances the performance of the DE algorithm and balances its exploration ability and its exploitation ability in the evolutionary process. Noman and Iba [22] proposed an adaptive local search (ALS) algorithm to increase the exploitation ability in the DE algorithm. The ALS algorithm uses a simple hill-climbing algorithm to adaptively determine the search length and effectively explore the neighborhood of each individual. Ali and Pant [18] applied a Cauchy mutation to improve the performance of the DE algorithm. The Cauchy mutation, which uses a Cauchy distribution, randomly forces solutions to move to another position. This method efficiently increases the probability of finding potential solutions in the DE algorithm. A combination of the fuzzy adaptive PSO algorithm and the DE algorithm, called the FAPSO-DE model, has been proposed, recently. They use two evolution processes to balance the exploration ability and the exploitation ability for economic dispatch problems.

Unlike the studies mentioned above, this paper proposes a new idea to solve the stagnation problem. This idea employs the inherent properties of the DE algorithm without depending on other learning algorithms. The idea combines two classical mutation strategies instead of a single mutation model. The DE/rand/bin approach has a powerful exploitation ability, and the DE/best/bin approach has an efficient exploration ability. This paper combines the two operations to tradeoff between the exploration ability and the exploitation ability in solving the stagnation problem.

In this paper, a group-based differential evolution (GDE) algorithm is proposed for numerical optimization problems. The GDE algorithm provides a new process using the DE/rand model and the DE/best model in the mutation operation. Initially, all individuals in a population are grouped into Group A (inferior group) and Group B (elite group) based on their fitness value. The Group A uses the DE/rand mutation model to globally search for potential solutions and maintain the diversity of the population. The Group B uses the DE/best mutation model to efficiently search the neighborhood of the current best solution. Subsequently, crossover and selection operations are employed for the next generation. Two adaptive strategies for automatically tuning parameters are also proposed in this paper. The contributions of this paper are summarized as follows.

(1) The proposed GDE algorithm employs the inherent properties of the DE algorithm to solve the stagnation problem. The GDE algorithm combines the two mutation operations to tradeoff between the exploration ability and the exploitation ability.
(2) Two adaptive strategies for automatically tuning parameters are proposed for automatically tuning parameters without the user's prior knowledge. An adaptive decreasing weight factor method is used for the Group A. Another strategy employs a self-adjusting method based on updating the success probability for the Group B.
(3) Thirteen well-known numerical benchmark functions are tested to validate the performance of the proposed GDE algorithm. The GDE algorithm shows significantly better performance than other EAs in statistical tests.

The remainder of the paper is organized as follows. Section 2 describes the basic procedure of differential evolution. The flow chart of GDE and adaptive parameter control strategy are elaborated in Section 3. Simulation results are presented in Section 4 to compare GDE with other evolutionary algorithms. Finally, concluding remarks are summarized in Section 5.

2. **Differential Evolution.**

This section introduces a complete DE algorithm. The process of the DE algorithm, like other EAs, produces offspring for the next generation by the mutation operation, the crossover operation and the selection operation. Figure 1(a) shows a standard flow chart of the DE algorithm.

Initially, a population of *NP* *D*-dimensional parameter vectors that represent the candidate solutions (individuals) is generated by a uniformly random process. All individuals and the search spaceare constrained by the prescribed minimum $\boldsymbol{X}_{\min} = (x_{1,\min}, x_{2,\min}, \ldots, x_{D,\min})$ and maximum $\boldsymbol{X}_{\max} = (x_{1,\max}, x_{2,\max}, \ldots, x_{D,\max})$ parameter bounds. The following is a simple representation of the *i*-th individual in the current generation *Gen*:

$$\boldsymbol{X}_{i,Gen} = (x_{i,1,Gen}, x_{i,2,Gen}, x_{i,3,Gen}, \ldots, x_{i,D-1,Gen}, x_{i,D,Gen}). \tag{1}$$

After the initial population with *NP* individuals, the fitness evaluation process measures the quality of individuals to calculate the performance. The succeeding steps include the mutation operation, the crossover operation and the selection operation, which are explained in the following.

Mutation Operation

Each individual in the current generation is allowed to breed by mating with other randomly selected individuals from the population. This process randomly selects a parent pool of three individuals to produce an offspring. Specifically, for each individual $\mathbf{X}_{i,gen}$, $i = 1, 2, \ldots, NP$, where *gen* denotes the current generation and *NP* is population size, three random individuals, $\mathbf{X}_{r1,gen}$, $\mathbf{X}_{r2,gen}$ and $\mathbf{X}_{r3,gen}$, are selected from the population such that $r1$, $r2$ and $r3 \in \{1, 2, \ldots, NP\}$ and $i \neq r1 \neq r2 \neq r3$. This way, a parent pool of four individuals is formed to produce an offspring. The following are different mutation strategies frequently used in the literature:

$$\text{DE/rand/bin: } \mathbf{V}_{i,gen} = \mathbf{X}_{r1,gen} + F(\mathbf{X}_{r2,gen} - \mathbf{X}_{r3,gen}) \tag{2}$$

$$\text{DE/best/bin: } \mathbf{V}_{i,gen} = \mathbf{X}_{gbest,gen} + F(\mathbf{X}_{r2,gen} - \mathbf{X}_{r3,gen}) \tag{3}$$

$$\text{DE/target-to-best/bin: } \mathbf{V}_{i,gen} = \mathbf{X}_{r1,gen} + F(\mathbf{X}_{gbest,gen} - \mathbf{X}_{r1,gen}) + F(\mathbf{X}_{r2,gen} - \mathbf{X}_{r3,gen}) \tag{4}$$

where F is a scaling factor $\in [0, 1]$ and $\mathbf{X}_{gbest,gen}$ is the best-so-far individual.

Crossover Operation

After the mutation operation, the DE algorithm uses a crossover operation, often referred to as discrete recombination, in which the mutated individual $\mathbf{V}_{i,gen}$ is mated with $\mathbf{X}_{i,gen}$ and generates the offspring $\mathbf{U}_{i,gen}$. The elements of an individual $\mathbf{U}_{i,gen}$ are inherited from $\mathbf{X}_{i,gen}$ and $\mathbf{V}_{i,gen}$, which are determined by a parameter called the crossover probability ($CR \in [0, 1]$), as follows:

$$\mathbf{U}_{i,d,gen} = \begin{cases} \mathbf{V}_{i,d,gen}, & \text{if } rand(d) \leq CR \\ \mathbf{X}_{i,d,gen}, & \text{if } rand(d) > CR \end{cases} \tag{5}$$

where $d = 1, 2, \ldots, D$ denotes the dth element of individual vectors, D is the total number of elements in an individual vector and $rand(d) \in [0, 1]$ is the dth evaluation of a random-number generator.

Selection Operation

The DE algorithm applies a selection operation to determine whether the individual survives to the next generation. A knockout competition is played between each individual $\mathbf{X}_{i,gen}$ and its offspring $\mathbf{U}_{i,gen}$, and the winner is selected deterministically based on objective function values and is then promoted to the next generation. The selection operation is described as

$$\mathbf{X}_{i,gen+1} = \begin{cases} \mathbf{X}_{i,gen}, & \text{if fitness}(\mathbf{X}_{i,gen}) < \text{fitness}(\mathbf{U}_{i,gen}) \\ \mathbf{U}_{i,gen}, & \text{otherwise} \end{cases} \tag{6}$$

where $f(z)$ is the fitness value of individual z. After the selection operation, the population obtains a better fitness value or retains the same fitness value but never deteriorates.

3. **Group-Based Differential Evolution.**

This section describes a complete GDE learning process. This learning process groups the population into an elite group and an inferior group. The groups perform different tasks based on mutation operations to produce offspring for the next generation. Two adaptive parameter tuning strategies are also proposed in the GDE algorithm.

3.1. **The GDE algorithm.**

In the DE algorithm, the mutation operation, which leads to successful evolution performance, is a principal operator. In this paper, we propose a GDE algorithm with an exploration ability and an exploitation ability, thus combining two mutation strategies to solve practical problems. A flow chart of the GDE algorithm is shown in Figure 1(b).

In the first step of the GDE algorithm, a population of *NP D*-dimensional individuals is generated by a uniformly random process and evaluated with respect to the fitness value of all individuals. A sorting process arranges all individuals based on their fitness value as follows for minimum-objective problems: $\text{fitness}_1 < \text{fitness}_2 < \ldots < \text{fitness}_{NP-1} < \text{fitness}_{NP}$. According to fitness value, all individuals are partitioned into an inferior group and an elite group, called Group A and Group B, respectively. Group A, with includes the *NP/2* worst individuals, performs a global search to increase the diversity of the population and find a wide range of potential solutions. The other *NP/2* individuals in the Group B perform a local search to actively detect better solutions near the current best solution. The following represents a complete mutation operation for the Group A and the Group B.

$$\text{Group A: } \mathbf{V}_{i,gen} = \mathbf{X}_{i,gen} + F_a(\mathbf{X}_{r1,gen} - \mathbf{X}_{r2,gen}) \tag{7}$$

$$\text{Group B: } \mathbf{V}_{i,gen} = \mathbf{X}_{gbest,gen} + F_b(\mathbf{X}_{r3,gen} - \mathbf{X}_{r4,gen}) \tag{8}$$

where F_a and F_b are scaling factors; $\mathbf{X}_{r1,gen}$, $\mathbf{X}_{r2,gen}$, $\mathbf{X}_{r3,gen}$ and $\mathbf{X}_{r4,gen}$ are randomly selected from the population; $i \neq r1 \neq r2 \neq r3 \neq r4$; and the $\mathbf{X}_{gbest,gen}$ is the best-so-far individual in the population. Next, we perform the crossover operation and the selection operation to produce offspring. All steps are repeated until the terminal condition is reached.

3.2. **Adaptive parameter tuning strategy.**

Parameter control which can directly influence the convergence speed and search capability of an algorithm, is an important task in EAs [19-21,23,24,27,30-33]. In this section, we employ two adaptive approaches to control the parameters F and CR for the Group A (inferior) and the Group B (elite).

The task of Group A is to globally search for potential offspring and increase the diversity of the population. A general strategy for parameter F often adopts a decreasing weight factor method [15,16] due to stable convergence. F, which depends on the generation, decreases linearly. In this paper, we propose an adaptive decreasing weight factor

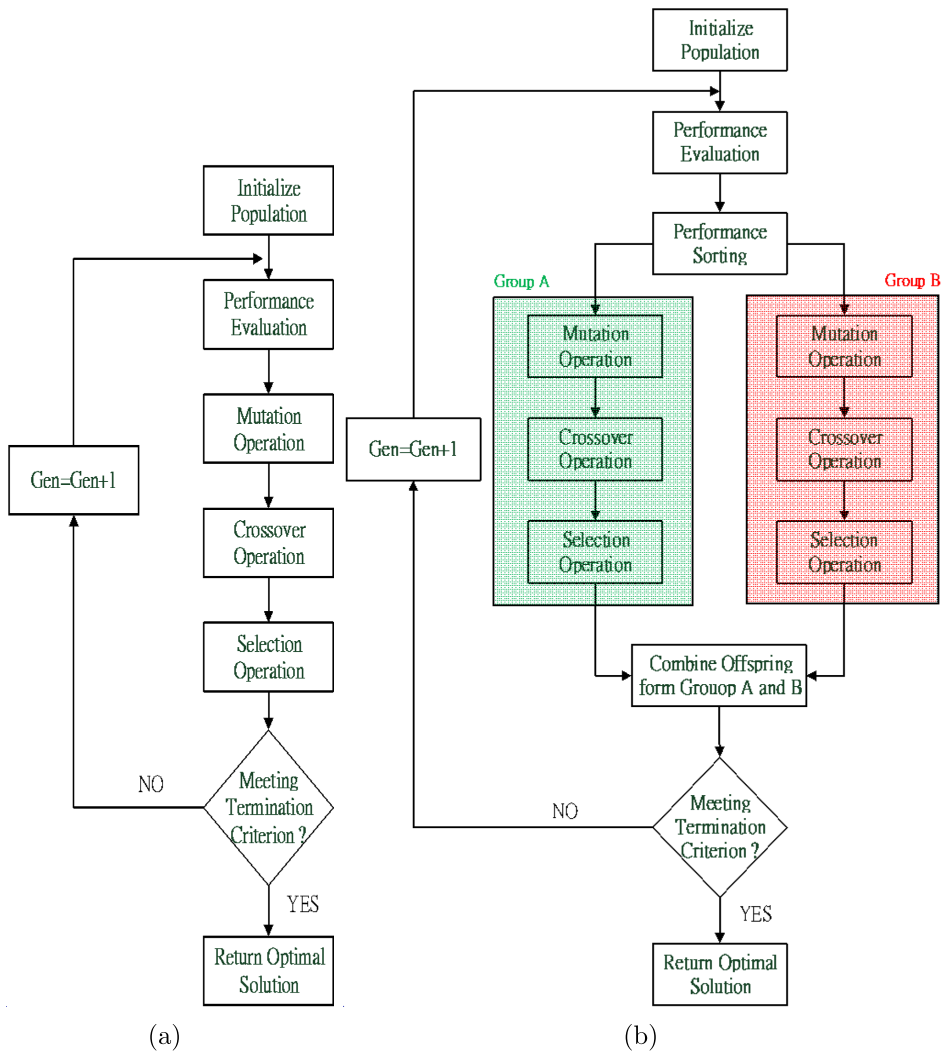

FIGURE 1. The flow chart of algorithms: (a) DE; (b) GDE. GEN is the generation counter.

method to control the parameter in Group A. F_a depends on the generation to decrease weight with flexibility as follows.

$$F_a = N\left(1 - \frac{Gen}{Gen_{\max}}, 0.1\right), \text{ for every } G_p, \tag{9}$$

where Gen is the current generation number, $Gen_{\max}$ is the maximum generation number, G_p is the pre-specified period of generation and $N(M, STD)$ is a normal distribution with mean M and standard deviation STD. In this paper, G_p is set to 20, and F_a is constrained between 0.1 and 1.

Unlike the task of Group A, the task of Group B, which features a stronger exploitation ability, is to find beneficial solutions near the current optimal solution. This characteristic always leads all individuals to a local minimum solution and stops at a certain point.

To handle this problem, an adaptive parameter-tuning strategy was proposed for F_b in Group B. This strategy provides a process to automatically adjust F_b based on global solution ($\mathbf{X}_{gbest,gen}$) updating success probability (GSP). If the GSP is greater than a pre-specified threshold (Th_{GSP}), F_b increases to prevent premature convergence; otherwise, the algorithm allows more individuals to move toward the $\mathbf{X}_{gbest,gen}$ position. This idea

Table 1. Benchmark functions. D is the dimension of the function.

Test Functions	D	Search Range
$f_1 = \sum_{i=1}^{D} (x_i)^2$	30 and 100	$[-100, 100]^D$
$f_2 = \sum_{i=1}^{D} \lvert x_i\rvert + \prod_{i=1}^{D} \lvert x_i\rvert$		$[-10, 10]^D$
$f_3 = \sum_{i=1}^{D} \left(\sum_{j=1}^{i} x_i \right)^2$		$[-100, 100]^D$
$f_4 = \max_i \lvert x_i\rvert$		$[-100, 100]^D$
$f_5 = \sum_{i=1}^{D} (x_i + 0.5)^2$		$[-100, 100]^D$
$f_6 = \sum_{i=1}^{D} i x_i^4 + rand\,[0, 1)$		$[-1.28, 1.28]^D$
$f_7 = \sum_{i=1}^{D} [100(x_{i+1} - x_i^2) + (x_i - 1)^2]$		$[-30, 30]^D$
$f_8 = \sum_{i=1}^{D} -x_i \sin \sqrt{\lvert x_i\rvert} + D \cdot 418.98288727243369$		$[-500, 500]^D$
$f_9 = \sum_{i=1}^{D} [x_i - 10 \cos(2\pi x_i) + 10]$		$[-5.12, 5.12]^D$
$f_{10} = -20 \exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D} \cos(2\pi x_i)\right) + 20 + e$		$[-32, 32]^D$
$f_{11} = \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$		$[-600, 600]^D$
$f_{12} = \frac{\pi}{D}\left\{ 10\sin^2(\pi y_1) + \sum_{i=1}^{D-1} (y_i - 1)^2 [1 + 10 \sin^2(\pi y_{i+1})] \right.$ $\left. + (y_D - 1)^2 \right\} + \sum_{i=1}^{D} u(x_i, 10, 100, 4)$ where $y_i = 1 + \frac{1}{4}(x_i + 1)$ and $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, & \text{if } x_i > a \\ k(-x_i - a)^m, & \text{if } x_i < -a \\ 0, & \text{otherwise} \end{cases}$		$[-50, 50]^D$
$f_{13} = \frac{1}{10}\left\{ \sin^2(3\pi x_1) + \sum_{i=1}^{D-1} (x_i - 1)^2 \left[1 + \sin^2(3\pi x_{i+1})\right] \right.$ $\left. + (x_D - 1)^2 [1 + \sin^2(2\pi x_D)] \right\} + \sum_{i=1}^{D} u(x_i, 10, 100, 4)$ where $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, & \text{if } x_i > a \\ k(-x_i - a)^m, & \text{if } x_i < -a \\ 0, & \text{otherwise} \end{cases}$		$[-50, 50]^D$

TABLE 2. Experimental results of GDE, DE/rand/bin, DE/best/bin and DE/target-to-best/bin algorithms for low-dimensional problems ($D = 30$)

Function	Gen.	GDE	DE/rand/bin	DE/best/bin	DE/target-to -best/bin
		Mean (Best, Worst)			
f1	1500	1.83E–42 (9.61E–59, 9.15E–41)	2.53E–13 (5.37E–14, 1.16E–12)	4.51E–14 (2.30E–15, 1.56E–13)	4.84E–16 (7.17E–17, 1.76E–15)
f2	2000	4.02E–30 (3.86E–41, 1.37E–28)	2.93E–09 (5.42E–10, 8.45E–09)	7.82E–11 (1.75E–11, 3.00E–10)	2.11E–11 (3.84E–12, 6.81E–11)
f3	5000	1.13E–25 (9.22E–38, 5.53E–24)	3.78E–10 (3.72E–11, 1.93E–09)	3.77E–11 (3.43E–13, 7.58E–10)	3.18E–14 (1.96E–16, 1.60E–13)
f4	5000	6.67E–11 (2.43E–14, 2.59E–09)	2.17 E–02 (4.15E–13, 5.25E–01)	1.93E–09 (2.48E–11, 1.95E–08)	8.34E–11 (4.04E–14, 6.83E–10)
f5	1500	0.0E+00 (0.0E+00, 0.0E+00)	2.98E–13 (6.03E–14, 8.50E–13)	3.97E–14 (4.03E–15, 1.82E–13)	5.55E–16 (3.87E–17, 5.20E–15)
f6	3000	2.08E–03 (6.02E–04, 9.43E–03)	1.74E–01 (3.60E–03, 7.77E–01)	7.12E–03 (3.00E–03, 1.23E–02)	5.79E–03 (2.16E–03, 1.14E–02)
f7	3000	3.73E–07 (1.27E–19, 1.12E–05)	1.17E+00 (1.67E–05, 3.06E+00)	7.97E–01 (1.83E–11, 3.98E+00)	5.58E–01 (1.04E–13, 3.98E+00)
f8	1500	2.52E+00 (1.18E+02, 8.58E–04)	6.80E+03 (4.71E+03, 7.27E+03)	2.94E+03 (1.78E+03, 4.88E+03)	3.12E+03 (9.49E+02, 6.89E+03)
f9	1500	5.68E–13 (0.0E+00, 6.86 E–12)	7.62E+01 (7.88E+00, 1.67 E+02)	4.55E+01 (2.28E+01, 7.36E+01)	1.71E+02 (1.33E+02, 2.13E+02)
f10	1500	9.69E–15 (7.99E–15, 3.28E–14)	1.68E–07 (7.25E–08, 3.31E–07)	5.59E–08 (2.08E–08, 2.16E–07)	6.64E–09 (2.47E–09, 1.67E–08)
f11	1500	0.0E+00 (0.0E+00, 0.0E+00)	1.08E–12 (5.87E–14, 1.38E–11)	8.31E–03 (6.32E–15, 5.65E–02)	5.86E–03 (0.0E+00, 2.21E–02)
f12	1500	1.50E–32 (1.34E–32, 4.06E–32)	3.81E–14 (1.66E–15, 2.84E–13)	1.03E–01 (8.03E–16, 2.06E+00)	2.69E–02 (3.60E–18, 5.19E–01)
f13	1500	1.70E–32 (1.57E–32, 6.8E–32)	3.17E–13 (2.82E–14, 1.76E–12)	2.63E–03 (2.49E–15, 1.09E–02)	1.08E–08 (2.86E–17, 5.41E–07)

is fully captured by the following equations.

$$F_b = F_b - [(rand \cdot (Th_{GSP} - GSP)], \tag{10}$$

$$GSP = \frac{\mathbf{X}_{gbest,gen} \text{ Updating Time}}{\text{Size of Group B}}, \tag{11}$$

where *rand* is a random number between 0 and 1. The average *GSP* was computed to adjust the F_b at every period G_p. In this study, Th_{GSP} was set to 0.2 and F_b was constrained between 0.1 and 1.

4. **Simulation Results.**

To verify the performance of the proposed algorithm, a set of 13 classical benchmark test functions [9,35,36] is used in this simulation. The analytical forms of these functions are shown in Table 1, where D denotes the dimensionality of the problem. Based on their properties, the functions can be divided into two problems: a unimodal function problem and a multimodal function problem. All of these functions have an optimal value at zero.

The GDE algorithm is compared with three classical DE algorithms, including the DE/rand/bin, the DE/best/bin and the DE/target-to-best/bin algorithms. In all simulations, we set the parameters of the GDE algorithm to be fixed, initial F_a = initial F_b = 0.9, initial CR_a = initial CR_b = 0.5. The parameters settings for three classical DE algorithms are recommended as follows. DE/rand/bin: $F = 0.5$ and $CR = 0.9$ [13,20,30]; DE/best/bin: $F = 0.8$ and $CR = 0.9$ [16]; DE/target-to-best/bin: $F = 0.8$ and $CR = 0.9$ [18].

Many researchers have used the same parameter setting to solve their problems. In this simulation, we set the population size *NP* to be 100 and 400 for $D = 30$ and $D = 100$, respectively. All results reported in this section are obtained based on 50 independent runs.

4.1. **Results for low-dimensional problems.**

In this simulation, the GDE, DE/rand/bin, DE/best/bin and DE/target-to-best/bin algorithms were applied to low-dimensional problems with respect to 13 benchmark test functions. Table 2 shows the detailed performance of the GDE, DE/rand/bin, DE/best/bin and DE/target-to-best/bin algorithms, including the mean, best and worst performances over 50 independent runs. This table indicates that the GDE algorithm clearly exhibits better performance than the DE/rand/bin, DE/best/bin and DE/target-to-best/bin algorithms with respect to 13 benchmark test functions. In particular, the GDE algorithm searched the global optimal solution at zero on Function 5 and Function 11.

The learning curves of the GDE, DE/rand/bin, DE/best/bin and DE/target-to-best/bin algorithms with respect to the 13 test functions for low-dimensional ($D = 30$) problems are shown in Figure 2. This figure shows that the GDE algorithm exhibits speedier convergence than the DE/rand/bin, DE/best/bin and DE/target-to-best/bin algorithms with respect to the 13 benchmark test functions. An interesting case is shown in Figure 3(h) and Figure 3(i). The DE/rand/bin, DE/best/bin and DE/target-to-best/bin algorithms were stopped at locally optimal solutions on Function 9 and Function 10. The GDE algorithm maintained continued convergence to find the optimal solutions. It is shown that the proposed GDE algorithm successfully overcomes the stagnation problem for low-dimensional problems.

4.2. **Results for high-dimensional problems.**

To verify the ability of the proposed algorithm to address high-dimensional problems, the GDE, DE/rand/bin, DE/best/bin and DE/target-to-best/bin algorithms were applied to the 13 benchmark test functions. Table 3 shows the detailed performance of the GDE, DE/rand/bin, DE/best/bin and

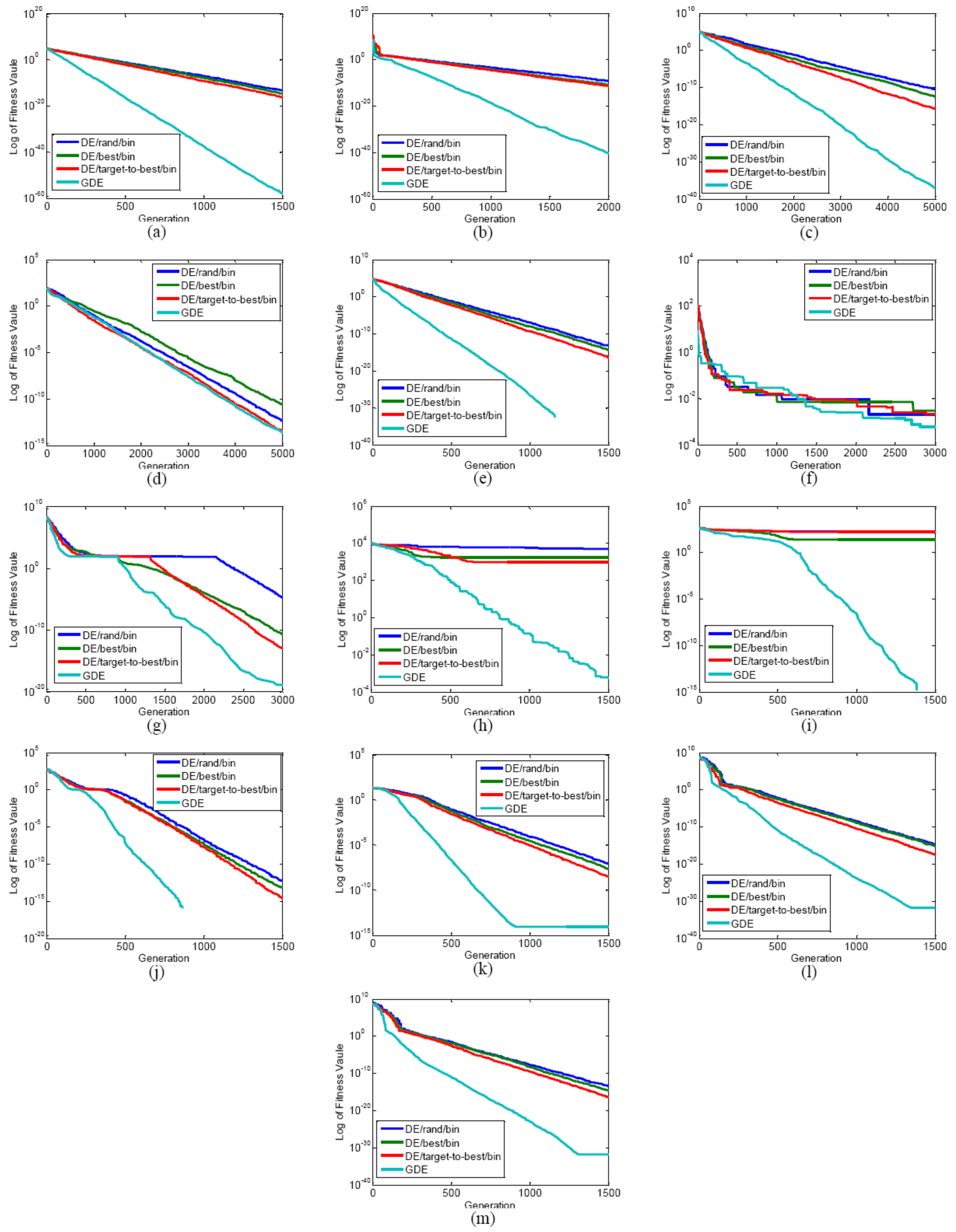

Figure 2. The best learning curves of GDE, DE/rand, DE/best and DE/target-to-best with respect to the 13 benchmark test functions for low-dimensional problems. (a) Function 1; (b) Function 2; (c) Function 3; (d) Function 4; (e) Function 5; (f) Function 6; (g) Function 7; (h) Function 8; (i) Function 9; (j) Function 10; (k) Function 11; (l) Function 12; (m) Function 13.

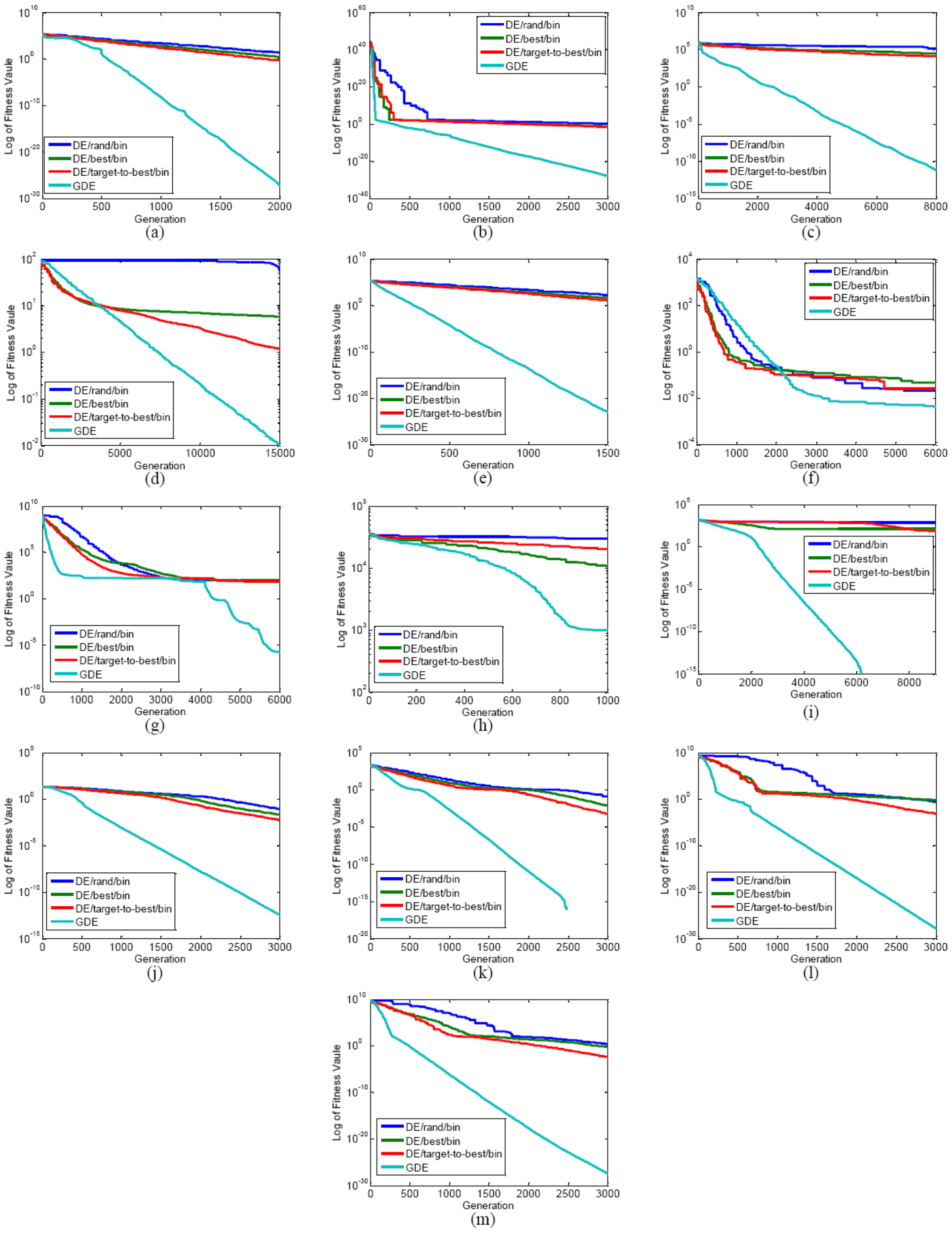

Figure 3. The best learning curves of GDE, DE/rand, DE/best and DE/target-to-best with respect to the 13 benchmark test functions for high-dimensional problems. (a) Function 1; (b) Function 2; (c) Function 3; (d) Function 4; (e) Function 5; (f) Function 6; (g) Function 7; (h) Function 8; (i) Function 9; (j) Function 10; (k) Function 11; (l) Function 12; (m) Function 13.

Table 3. Experimental results of GDE, DE/rand/bin, DE/best/bin and DE/target-to-best/bin algorithms for high-dimensional problems ($D = 100$)

Function	Gen.	GDE	DE/rand/bin	DE/best/bin	DE/target-to -best/bin
		Mean (Best, Worst)			
f1	2000	4.95E–21 (8.68E–28, 9.07E–20)	3.71E+01 (2.14E+01, 5.22E+01)	5.25E+00 (2.31E+00, 1.11E+01)	1.13E+00 (5.33E–01, 2.60E+00)
f2	3000	9.81E–23 (1.60E–28, 3.66E–21)	2.46E+00 (1.58E+00, 3.82E+00)	1.41E–01 (7.18E–02, 2.29E–01)	7.27E–02 (2.87E–02, 1.41E–01)
f3	8000	2.74E–10 (7.24E–12, 4.08E–09)	2.23E+05 (1.47E+05, 3.13E+05)	4.91E+04 (2.97E+04, 7.34E+04)	3.04E+04 (1.39E+04, 4.65E+04)
f4	15000	1.23E–02 (1.00E–02, 1.55E–23)	9.19E+01 (5.68E+01, 9.54E+01)	1.08E+01 (5.86E+00, 15.9E+00)	2.40E+00 (1.19E+00, 4.25E+00)
f5	1500	5.27E–22 (1.31E–23, 5.44E–21)	3.70E+02 (2.03E+02, 5.19E+02)	6.93E+01 (4.00E+01, 1.06E+02)	2.08E+01 (1.32E+01, 3.23E+01)
f6	6000	6.15E–03 (4.40E–03, 8.26E–03)	2.98E–02 (2.21E–02, 3.49E–02)	7.27E–02 (4.67E–02, 1.10E–01)	4.24E–02 (2.66E–02, 610E–02)
f7	6000	6.70 E+00 (1.71E–06, 3.72E+01)	9.11E+01 (9.05E+01, 9.23E+01)	1.52E+02 (8.40E+01, 2.99E+02)	1.06E+02 (7.69E+01, 1.49E+02)
f8	1000	2.66E+03 (9.82E+02, 3.79E+03)	3.14E+04 (2.94E+04, 3.24E+04)	1.36E+04 (1.05E+04, 1.81E+04)	2.86E+04 (2.03E+04, 3.17E+04)
f9	9000	0.0E+00 (0.0E+00, 0.0E+00)	8.08E+02 (7.53E+02, 8.47E+02)	1.74E+02 (1.26E+02, 2.42E+02)	4.20E+02 (7.3E+01, 7.99E+02)
f10	3000	4.41E–13 (3.45E–13, 5.65E–13)	1.54E–01 (9.20E–02, 2.09E–01)	2.68E–01 (2.08E–02, 1.32E+00)	9.45E–03 (5.51E–03, 1.30E–02)
f11	3000	0.0E+00 (0.0E+00, 0.0E+00)	2.48E–01 (1.47E–01, 3.71E–01)	1.72E–02 (7.14E–03, 3.53E–02)	2.32E–03 (5.98E–04, 1.44 E–02)
f12	3000	2.43E–24 (1.80E–28, 2.54E–23)	2.35E+00 (3.11E–01, 1.05E+01)	2.81E+00 (6.42E–01, 6.66E+00)	2.47E–01 (6.86E–04, 1.34E+00)
f13	3000	7.65E–25 (3.38E–28, 4.18E–24)	8.82E+00 (2.32E+00, 2.47E+01)	7.49E+00 (5.72E–01, 3.58E+01)	1.91E–01 (4.17E–03, 3.84E+00)

DE/target-to-best/bin algorithms, including the mean, best and worst performances over 50 independent runs. Clearly, it is difficult to find optimal solutions using these algorithms because of the high dimensionality of the problem. According to Table 3, the GDE algorithm showed better performance than the DE/rand/bin, DE/best/bin and DE/target-to-best/bin algorithms with respect to the 13 benchmark test functions. Note that the GDE algorithm efficiently searches for a global optimal solution at zero for Function 9 and Function 11.

The learning curve of the GDE, DE/rand/bin, DE/best/bin and DE/target-to-best/bin algorithms with respect to the 13 benchmark test functions for high-dimensional ($D = 100$) problems is shown in Figure 3. In this figure, the GDE algorithm also presents speedier convergent curves than other algorithms for high-dimensional functions. The stagnation situation also occurs when the DE/rand/bin, DE/best/bin and DE/target-to-best/bin algorithms are executed, as shown in Figure 3(b), Figure 3(c), Figure 3(e), Figure 3(g) and Figure 3(i). The GDE algorithm continuously maintained a convergent curve on Function 2, Function 3, Function 5, Function 7 and Function 9. In this paper, the simulation results show that the proposed GDE algorithm clearly achieves better performance and successfully overcomes the stagnation situation for low- and high-dimensional problems.

4.3. **Statistical comparison.** To understand the significant difference between the GDE and other algorithms over multiple test functions, we performed a statistical procedure based on the Friedman test. [40,41] with the corresponding post-hoc tests. We set the GDE algorithm as the control algorithm to compare with other algorithms. The performance of the algorithm is significantly different if the corresponding average ranks differ by at least the critical difference (CD)

$$CD = q_{0.05}\sqrt{\frac{j(j+1)}{6T}}, \tag{12}$$

where j is the number of algorithms, T is the number of test functions, and the critical value $q_{0.05} = 2.569$ can be found in [41].

Table 4 presents the complete results of the Friedman test. In this simulation, $j = 4$, $T = 13$ and $CD = 0.13$. All differences were greater than the critical difference, which means that the GDE algorithm is significantly better than the DE/rand/bin, DE/best/bin and DE/target-to-best/bin algorithms in this case.

4.4. **Comparison with other algorithms.** Further results regarding the comparison of the GDE algorithm with other evolutionary algorithms is presented in this section. These algorithms include jDE [30], SaDE [24], ALEP [37], BestLevy [37], NSDE [38] and RTEP [39]. Table 5 shows the comparative results with respect to 30-dimensional problems. In

Table 4. Results of the Friedman test for statistical comparison

$D = 30$		
Algorithm	Difference in Rank	Critical Difference (CD)
DE/rand/bin	$(3.54 - 1) = 2.54$	1.30
DE/best/bin	$(3.15 - 1) = 2.15$	
DE/target-to-best/bin	$(2.31 - 1) = 1.31$	
$D = 100$		
DE/rand/bin	$(3.69 - 1) = 2.69$	1.30
DE/best/bin	$(3.07 - 1) = 2.07$	
DE/target-to-best/bin	$(2.23 - 1) = 1.23$	

Table 5. Comparison with other evolutionary algorithms ($D = 30$), including GDE, jDE [30], SaDE [24], ALEP [37], BestLevy [37], NSDE [38] and RTEP [39]

Function	GDE	jDE [30]	SaDE [24]	ALEP [37]	BestLevy [37]	NSDE [38]	RTEP [39]
	Mean						
f1	**1.83E–42**	1.10E–28	4.50E–20	6.32E–04	6.59E–04	7.10E–17	7.50E–18
f2	**4.02E–30**	1.50E–23	1.90E–14	–	–	–	–
f3	**1.13E–25**	9.00E–02	–	4.18E–02	3.06E+01	7.90E–16	2.40E–15
f4	6.67E–11	**1.40E–15**	7.40E–11	–	–	–	–
f5	**0.0E+00**	**0.00E+00**	**0.00E+00**	–	–	–	–
f6	**2.08E–03**	3.30E–03	4.80E–03	–	–	–	–
f7	3.73E–07	3.10E–15	–	4.34E+01	5.77E+01	**5.90E–28**	1.10E+00
f8	2.52E+00	–	–	1.10E+03	6.70E+02	–	**2.90E–07**
f9	5.68E–13	**1.50E–15**	–	5.85E+00	1.30E+01	–	2.50E–14
f10	9.69E–15	**7.70E–15**	–	1.90E–02	3.10E–02	1.69E–09	2.00E–10
f11	**0.0E+00**	**0.00E+00**	–	2.4E–02	1.80E–02	5.80E–16	2.70E–25
f12	**1.50E–32**	6.60E–30	6.10E–05	6.00E–06	3.00E–05	5.40E–16	3.20E–13
f13	**1.70E–32**	5.00E–29	1.70E–19	9.80E–05	2.60E–04	6.40E–17	7.10E–08

Table 6. Simulation results of the GDE with various initial F and CR values

Function	GDE					
	$F = 0.1$ $CR = 0.1$	$F = 0.1$ $CR = 0.5$	$F = 0.5$ $CR = 0.5$	$F = 0.5$ $CR = 0.9$	$F = 0.9$ $CR = 0.5$	$F = 0.9$ $CR = 0.9$
f1	4.38E–43	7.12E–43	5.21E–42	2.53E–42	1.83E–42	2.13E–42
f3	1.01E–25	1.33E–25	3.83E–25	1.77E–25	1.13E–25	1.40E–25
f5	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
f7	1.24E–05	2.95E–05	7.15E–07	2.11E–08	3.73E–07	5.51E–07
f9	8.69E–10	8.69E–10	6.90E–13	5.91E–13	5.68E–13	5.95E–13
f11	7.79E–54	1.61E–54	0.0E+00	0.0E+00	0.0E+00	0.0E+00
f13	3.90E–29	1.27E–29	3.45E–32	3.21E–32	1.70E–32	2.53E–32

unimodal function problems, the GDE algorithm showed the best results for five of six functions. In multimodal function problems, the GDE algorithm showed the best results for three of seven functions and provided a result near the best solution for Function 10. The overall results show that the GDE algorithm is a more effective algorithm than other competitive algorithms.

4.5. The sensitivity of initial F and CR.

To understand the sensitivity of parameters, this paper presents the results of further simulations with various initial F_a, F_b, CR_a and CR_b values. The proposed GDE algorithm was applied to low-dimensional problems. In this simulation, the population size and the maximum generation were set to100 and 1500. For the redundancy of parameters, we set $F = F_a = F_b$ and $CR = CR_a = CR_b$. All results reported in this section are obtained based on 50 independent runs for various initial F and CR values. Table 6 presents the results of the GDE algorithm based on various initial F and CR values. The results show that the overall performance is not significantly different when the GDE algorithm uses various initial F and CR values. Therefore, the proposed GDE algorithm offers an advantage that is robust with respect to the initial F and CR values.

5. **Conclusions.**

This paper has proposed a GDE algorithm for numerical optimization problems. The GDE algorithm is a generalized DE model that combines exploitation ability and exploration ability. In addition, two adaptive parameter-tuning strategies are used to adjust the scaling factors F and CR in the GDE algorithm. The simulation results demonstrate that the proposed GDE algorithm successfully handles the stagnation problem and effectively searches for the global optimal solution in benchmark function optimization problems. The statistical test reveals the significant differences between the GDE algorithm and other EAs.

Two advanced topics regarding the proposed GDE algorithm should be addressed in future research. First, the GDE algorithm may adopt other further evolutionary learning strategies to improve its performance. For example, a symbiotic learning method has been to used to enhance the performance of the PSO algorithm. The basic idea of symbiotic learning methods is that an individual is composed of multiple elements, which are randomly selected from a subpopulation. Every subpopulation performs an evolution process to produce new elements. This method increases the possibility of finding potential solutions. Second, in addition to the simulations performed in this paper, the proposed model can be applied to solve neuro-fuzzy system optimization problems.

Acknowledgment. This work was supported in part by the Aiming for the Top University Plan of National Chiao Tung University, the Ministry of Education, Taiwan, under Contract 100W9633 and 101W963. This work was also supported in part by the UST-UCSD International Center of Excellence in Advanced Bioengineering sponsored by the Taiwan National Science Council I-RiCE Program under Grant Number: NSC-100-2911-I-009-101 and NSC-101-2911-I-009-101.

REFERENCES

[1] C. A. C. Carlos, 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, pp.1245-1287, 2002.

[2] P. Fei, T. Ke, C. Guoliang and Y. Xin, Population-based algorithm portfolios for numerical optimization, *IEEE Transactions on Evolutionary Computation*, vol.14, pp.782-800, 2010.

[3] D. B. Fogel, *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*, IEEE Press, New York, 1995.

[4] D.-W. Gong, N.-N. Qin and X.-Y. Sun, Evolutionary algorithms for optimization problems with uncertainties and hybrid indices, *Information Sciences*, vol.181, pp.4124-4138, 2011.

[5] M.-F. Han, L.-D. Liao, Y.-H. Liu, W.-R. Wang, C.-T. Lin and B.-S. Lin, Performance optimized of the novel dry EEG electrodes by using the non-dominated sorting genetic algorithms (NSGA-II), *TENCON 2010-2010 IEEE Region 10 Conference*, pp.1710-1715, 2010.

[6] M. N. Omidvar, X. Li, Z. Yang and Y. Xin, Cooperative co-evolution for large scale optimization through more frequent random grouping, *IEEE Congress on Evolutionary Computation*, pp.1-8, 2010.

[7] A. Zhou, B.-Y. Qu, H. Li, S.-Z. Zhao, P. N. Suganthan and Q. Zhang, Multiobjective evolutionary algorithms: A survey of the state of the art, *Swarm and Evolutionary Computation*, vol.1, pp.32-49, 2011.

[8] J. H. Holland, *Adaptation in Natural and Artificial Systems*, MIT Press, Univ. Michigan, 1975.

[9] X. Yao, Y. Liu and G. Lin, Evolutionary programming made faster, *IEEE Transactions on Evolutionary Computation*, vol.3, pp.82-102, 1999.

[10] T. T. Bäck and H.-P. Schwefel, Evolution strategies: A comprehensive introduction, *Natural Computing*, pp.3-52, 2002.

[11] J. Kennedy and R. Eberhart, Particle swarm optimization, *The IEEE Int. Neural Netw.*, 1995.

[12] K. Price, R. Storn and J. Lampinen, *Differential Evolution: A Practical Approach to Global Optimization*, Springer-Verlag, Berlin, 2005.

[13] 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.

[14] Z. Cai, W. Gong, C. X. Ling and H. Zhang, A clustering-based differential evolution for global optimization, *Applied Soft Computing*, vol.11, pp.1363-1379, 2011.

[15] C.-H. Chen, C.-J. Lin and C.-T. Lin, Nonlinear system control using adaptive neural fuzzy networks based on a modified differential evolution, *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol.39, pp.459-473, 2009.

[16] S. Das, A. Abraham, U. K. Chakraborty and A. Konar, Differential evolution using a neighborhood-based mutation operator, *IEEE Transactions on Evolutionary Computation*, vol.13, pp.526-553, 2009.

[17] S. Das and P. N. Suganthan, Differential evolution: A survey of the state-of-the-art, *IEEE Transactions on Evolutionary Computation*, vol.15, pp.4-31, 2011.

[18] M. Ali and M. Pant, Improving the performance of differential evolution algorithm using Cauchy mutation, *Soft Computing*, vol.15, pp.991-1007, 2011.

[19] J. Zhang and A. C. Sanderson, JADE: Self-adaptive differential evolution with fast and reliable convergence performance, *IEEE Congress on Evolutionary Computation*, pp.2251-2258, 2007.

[20] J. Zhang and A. C. Sanderson, JADE: Adaptive differential evolution with optional external archive, *IEEE Transactions on Evolutionary Computation*, vol.13, pp.945-958, 2009.

[21] E. Mezura-Montes, M. E. Miranda-Varela and R. del Carmen Gmez-Ramn, Differential evolution in constrained numerical optimization: An empirical study, *Information Sciences*, vol.180, pp.4223-4262, 2010.

[22] N. Noman and H. Iba, Accelerating differential evolution using an adaptive local search, *IEEE Transactions on Evolutionary Computation*, vol.12, pp.107-125, 2008.

[23] 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, pp.398-417, 2009.

[24] A. K. Qin and P. N. Suganthan, Self-adaptive differential evolution algorithm for numerical optimization, *The 2005 IEEE Congress on Evolutionary Computation*, vol.2, pp.1785-1791, 2005.

[25] S. Rahnamayan, H. R. Tizhoosh and M. M. A. Salama, Opposition-based differential evolution, *IEEE Transactions on Evolutionary Computation*, vol.12, pp.64-79, 2008.

[26] M.-T. Su, C.-H. Chen, C.-J. Lin and C.-T. Lin, A rule-based symbiotic modified differential evolution for self-organizing neuro-fuzzy systems, *Applied Soft Computing*, vol.11, pp.4847-4858, 2011.

[27] W. Gong, Z. Cai, C. X. Ling and L. Hui, Enhanced differential evolution with adaptive strategies for numerical optimization, *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol.41, pp.397-413, 2011.

[28] J. Vesterstrom and R. Thomsen, A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems, *Congress on Evolutionary Computation*, vol.2, pp.1980-1987, 2004.

[29] C. T. Lin, M. F. Han, Y. Y. Lin, J. Y. Chang and L. W. Ko, Differential evolution based optimization of locally recurrent neuro-fuzzy system for dynamic system identification, *The 17th National Conference on Fuzzy Theory and Its Applications*, 2010.

[30] J. Brest, S. Greiner, B. Boskovic, 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, pp.646-657, 2006.

[31] C.-T. Lin, M.-F. Han, Y.-Y. Lin, S.-H. Liao and J.-Y. Chang, Neuro-fuzzy system design using differential evolution with local information, *2011 IEEE International Conference on Fuzzy Systems (FUZZ)*, pp.1003-1006, 2011.

[32] T. Josef, Adaptation in differential evolution: A numerical comparison, *Applied Soft Computing*, vol.9, pp.1149-1155, 2009.

[33] J. Liu and L. Jouni, A fuzzy adaptive differential evolution algorithm, *TENCON'02, Proc. of 2002 IEEE Region 10 Conference on Computers, Communications, Control and Power Engineering*, vol.1, pp.606-611, 2002.

[34] F. Xue, A. C. Sanderson, P. P. Bonissone and R. J. Graves, Fuzzy logic controlled multiobjective differential evolution, *The IEEE Int. Conf. Fuzzy Syst.*, 2005.

[35] Y.-W. Shang and Y.-H. Qiu, A note on the extended rosenbrock function, *Evolutionary Computation*, vol.14, pp.119-126, 2006.

[36] X. Yao, Y. Liu, K.-H. Liang and G. Lin, Fast evolutionary algorithms, *The Advances Evol. Computing: Theory Applicat.*, New York, 2003.

[37] C. Lee and X. Yao, Evolutionary programming using mutations based on the Lévy probability distribution, *IEEE Trans. Evol. Comput.*, vol.8, pp.1-13, 2004.

[38] Z. Yang, J. He and X. Yao, Making a difference to differential evolution, *Advances Metaheuristics Hard Optimization*, pp.397-414, 2007.

[39] M. S. Alam, M. M. Islam, F. X. Yao and K. Murase, Recurring two-stage evolutionary programming: A novel approach for numeric optimization, *IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics*, vol.41, pp.1352-1365, 2011.

[40] J. Demar, Statistical comparisons of classifiers over multiple data sets, *Journal of Machine Learning Research*, pp.1-30, 2006.

[41] S. García and F. Herrera, An extension on statistical comparisons of classifiers over multiple data sets for all pairwise comparisons, *Journal of Machine Learning Research*, pp.2677-2694, 2008.