Gaussian Bare-Bones Differential Evolution

Hui Wang, Shahryar Rahnamayan, Hui Sun, and Mahamed G. H. Omran

***Abstract*—Differential evolution (DE) is a well-known algorithm for global optimization over continuous search spaces. However, choosing the optimal control parameters is a challenging task because they are problem oriented. In order to minimize the effects of the control parameters, a Gaussian bare-bones DE (GBDE) and its modified version (MGBDE) are proposed which are almost parameter free. To verify the performance of our approaches, 30 benchmark functions and two real-world problems are utilized. Conducted experiments indicate that the MGBDE performs significantly better than, or at least comparable to, several state-of-the-art DE variants and some existing bare-bones algorithms.**

***Index Terms*—Bare-bones particle swarm, differential evolution (DE), evolutionary optimization, global optimization, numerical optimization.**

I. Introduction

DIFFERENTIAL EVOLUTION (DE) [1], [2] is a population-based stochastic optimization algorithm, which has been applied successfully to solve many real-world and benchmark problems. Its performance, similar to that of other evolutionary algorithms (EAs), is greatly influenced by the settings of its control parameters, such as the mutation scale factor and crossover rate [3], [4]. Although the suggestions for parameter settings have been made in the literature [2], these values are not universally applicable to achieve the highest performance. The main reason is that those parameters are problem dependent and the fixed parameter settings are not a general solution.

In order to minimize the effects of DE control parameters, a number of improved DE variants haven been proposed in the past decade [4]–[10]. All of these algorithms are based on adaptive or self-adaptive mechanisms, in which the control parameters are automatically adjusted according to some updating rules. Although these strategies are helpful to improve the performance of DE and avoid choosing parameters based on trial and error process, many of them introduce additional complex adaptive operations which usually are not easy to implement.

Manuscript received September 10, 2011; revised May 25, 2012 and July 17, 2012; accepted August 7, 2012. Date of publication September 20, 2012; date of current version April 16, 2013. This work was supported in part by the Science and Technology Plan Projects of Jiangxi Provincial Education Department under Grants GJJ12641 and GJJ12633 and in part by the National Natural Science Foundation of China under Grants 61070008 and 61261039. This paper was recommended by Associate Editor T. Vasilakos.

H. Wang and H. Sun are with the School of Information Engineering, Nanchang Institute of Technology, Nanchang 330099, China (e-mail: huiwang@whu.edu.cn; sun_hui2006@yahoo.com.cn).

S. Rahnamayan is with the Department of Electrical and Computer Engineering, University of Ontario Institute of Technology, Oshawa, ON L1H 7K4, Canada (e-mail: shahryar.rahnamayan@uoit.ca).

M. G. H. Omran is with the Department of Computer Science, Gulf University for Science and Technology, Hawalli 32093, Kuwait (e-mail: omran.m@gust.edu.kw).

Digital Object Identifier 10.1109/TSMCB.2012.2213808

In this paper, a new bare-bones DE (BBDE) variant [Gaussian bare-bones DE (GBDE)] and its improved version [modified GBDE (MGBDE)] are proposed to minimize the effects of the control parameters. The proposed approaches are inspired by the concept of bare-bones particle swarm optimization (PSO) (BBPSO) [11]. Compared to the existing BBDE [12], our approaches are simpler yet more efficient. In order to verify the performance of the proposed approaches, a comprehensive set of experiments is conducted on 30 well-known numerical benchmark functions and two real-world optimization problems.

The rest of this paper is organized as follows. In Section II, the classical DE algorithm is briefly introduced. Section III presents an overview of DE related works. The proposed approaches are described in Section IV. Section V presents the experimental results and discussions. Finally, conclusion and future work are given in Section VI.

II. Classical DE

Like other EAs, DE is a population-based stochastic search algorithm. It starts with a population of N_p vectors representing the candidate solutions, where N_p indicates the population size. Let us assume that $X_{i,G} = [x_{i,1,G}, x_{i,2,G}, \ldots, x_{i,D,G}]$ is the ith candidate solution vector in generation G, where $(i = 1, 2, \ldots, N_p)$, D is the problem's dimension, and G is the generation index. For the classical DE, there are three following operations: mutation, crossover, and selection, which are described as follows.

A. Mutation

For each vector $X_{i,G}$ at generation G, this operation creates mutant vectors $V_{i,G}$ based on the current parent population. The following are five well-known variant mutation strategies.

1) DE/rand/1

$$V_{i,G} = X_{i_1,G} + F \cdot (X_{i_2,G} - X_{i_3,G}). \tag{1}$$

2) DE/best/1

$$V_{i,G} = X_{\text{best},t} + F \cdot (X_{i_1,G} - X_{i_2,G}). \tag{2}$$

3) DE/current-to-best/1

$$V_{i,G} = X_{i,G} + F \cdot (X_{\text{best},G} - X_{i,G}) + F \cdot (X_{i_1,G} - X_{i_2,G}). \tag{3}$$

4) DE/rand/2

$$V_{i,G} = X_{i_1,G} + F \cdot (X_{i_2,G} - X_{i_3,G}) + F \cdot (X_{i_4,G} - X_{i_5,G}). \tag{4}$$

5) DE/best/2

$$V_{i,G} = X_{\text{best},G} + F \cdot (X_{i_1,G} - X_{i_2,G}) + F \cdot (X_{i_3,G} - X_{i_4,G}). \quad (5)$$

The indices i_1, i_2, i_3, i_4, and i_5 are mutually different random indices chosen from the set $\{1, 2, \ldots, N_p\}$, and all are different from the base index i. The scale factor F is a real number that controls the difference vectors. $X_{\text{best},G}$ is the best vector in terms of fitness value at the current generation G.

B. Crossover

DE also employs a crossover operator to build trial vectors by recombining the current vector and the mutant one. The trial vector $U_{i,G} = (u_{i,1,G}, u_{i,2,G}, \ldots, u_{i,D,G})$ is defined as follows:

$$u_{i,j,G} = \begin{cases} v_{i,j,G}, & \text{if } \text{rand}_j(0,1) \le CR \vee j = j_{\text{rand}} \\ x_{i,j,G}, & \text{otherwise} \end{cases} \quad (6)$$

where $CR \in (0, 1)$ is the predefined crossover probability, $\text{rand}_j(0, 1)$ is a uniform random number within [0, 1] for the jth dimension, and $j_{\text{rand}} \in \{1, 2, \ldots, D\}$ is a random index.

C. Selection

After the crossover, a greedy selection mechanism is used to select the better one between the parent vector $X_{i,G}$ and the trial vector $U_{i,G}$ according to their fitness values $f(\cdot)$. Without loss of generality, this paper only considers minimization problems. If, and only if, the trial vector $U_{i,G}$ is better than the parent vector $X_{i,G}$, then $X_{i,G+1}$ is set to $U_{i,G}$; otherwise, we keep $X_{i,G+1}$ the same with $X_{i,G}$.

$$X_{i,G+1} = \begin{cases} U_{i,G}, & \text{if } f(U_{i,G}) \le f(X_{i,G}) \\ X_{i,G}, & \text{otherwise.} \end{cases} \quad (7)$$

III. A Brief Review of DE

Although DE has shown better performance in solving many benchmark and real-world problems, its performance highly depends on the selected mutation strategies and corresponding control parameters. To overcome these drawbacks, many improved DE variants have been proposed in the past decades. In this section, a brief overview of these enhanced approaches is presented.

Liu and Lampinen [5] proposed a fuzzy adaptive DE (FADE), which uses a fuzzy logic controller to set the probability of mutation and crossover. The presented experimental results show that FADE surpasses classical DE. Brest *et al.* [6] presented a self-adaptive DE (jDE), in which the control parameters are randomized. Experimental studies on jDE show that it achieves better results than classical DE, classical evolutionary programming [13], and fast evolutionary programming [13] on most test functions. For further improvement of jDE performance on high-dimensional optimization problems, several enhanced versions of jDE with population size reduction are proposed in [14] and [15].

Qin and Suganthan [7], [8] presented a self-adaptive DE (SaDE) for numerical optimization, which focuses on the adaptation of the CR parameter and mutation strategies of DE. The proposed SaDE does not use any particular learning strategy or any specific settings for the control parameters. SaDE uses its previous learning experiences to select the mutation strategies and parameter values adaptively. Yang *et al.* [16] introduced a neighborhood search strategy to DE (NSDE), which generates F using Gaussian and Cauchy distributed random numbers instead of predefining a constant F. Based on SaDE and NSDE, Yang *et al.* [17] proposed another version of DE self-adaptive differential evolution with neighborhood search (SaNSDE), which is inspired from NSDE to generate self-adaptive F and uses a weighted adaptation scheme for CR. The reported empirical results verified that SaNSDE outperforms SaDE and NSDE in terms of solution accuracy.

Noman and Iba [18] introduced an adaptive local search (i.e., hill-climbing strategy) to DE. Rahnamayan *et al.* [19] presented an opposition-based DE (ODE) to accelerate the convergence speed of DE. The proposed approach employs an opposition-based-computation concept to calculate the opposite candidate solutions of the current population. By simultaneously evaluating the current candidate solutions and their opposite, ODE can provide a higher chance of finding candidate solutions more closely to the global optimum. Experimental studies on ODE show that it is faster and more robust than classical DE and FADE. Inspired from ODE, Wang *et al.* [20] proposed an enhanced ODE generalized opposition-based DE (GODE) by using a generalized opposition-based learning [21]–[23]. Experimental results show that GODE performs better in terms of convergence rate and robustness. A parallel GODE version based on graphics processing unit can be found in [24].

Zhang and Sanderson [9] proposed a self-adaptive DE adaptive DE with optional external archive (JADE) which employs a new mutation strategy called "DE/current-to-$p$best." The presented results show that JADE outperforms PSO, DE, jDE, and SaDE on several benchmark functions. Gong *et al.* [10] proposed a simple strategy adaptation mechanism (SaM) which can be used for different mutation strategies. The simulation results confirm that SaM is helpful to improve the performance of DE variants. Das *et al.* [25] proposed an enhanced DE algorithm DE with global and local neighborhoods (DEGL) by using an improved "DE/target-to-best/1" strategy which employs two mutation strategies: local neighborhood and global neighborhood. It aims to achieve better balance between the exploration and exploitation abilities of DE. Experimental results show that DEGL is better than several state-of-the-art DE variants and evolutionary techniques. Inspired from the idea of DEGL, Wang *et al.* [26], [27] proposed a new DE variant with neighborhood search which focuses on searching the neighbors of vectors during the evolution. Ghosh *et al.* [28] described a simple yet effective adaptation technique for tuning both F and CR during the evolution. The adaptation strategy is based on the objective function value of individuals in the DE population. Simulation results show that the proposed approach obtains promising performance over 14 well-known benchmark functions and one real-life problem. Islam *et al.* [29] proposed a new adaptive DE variant which

employs an improved DE/current-to-best/1 mutation scheme and a fitness-induced parent selection scheme for the binomial crossover.

Chakraborty *et al.* [30] proposed an improved harmony search (HS) algorithm hybridized with DE. The reported simulation results show that the proposed approach differential harmony search achieves better solutions than the classical HS, global best HS, and DE. Mallipeddi and Suganthan [31] introduced an ensemble of mutation strategies and control parameters with DE (EPSDE), in which a pool of distinct mutation strategies along with a pool of values of each parameter coexists throughout the evolution process and competes to produce offspring. The experimental results showed that EPSDE outperforms SaDE, jDE, and JADE on a majority of test problems. Weber *et al.* [32] proposed a distributed DE which employs a scale factor inheritance mechanism for tuning F. In the proposed approach, the population is distributed over several allocated subpopulations based on a ring topology. Each subpopulation is characterized by its own scale factor value. Wang *et al.* [33] proposed an adaptive DE with variable population size to solve high-dimensional global optimization problems, in which the population size is dynamically adjusted according to the search status of the current population. Lin *et al.* [34] introduced a new mutation strategy, namely, the best of random, in which the best individual among several randomly chosen individuals is selected as the differential mutation base while the other worse individuals are donors for vector differences. Hence, higher diversity and fast evolution speed can be obtained. Epitropakis *et al.* [35] introduced a framework of proximity-based mutation operators which incorporates information of neighboring individuals to efficiently guide the evolution of the population toward the global optimum. Wang *et al.* [36] proposed a composite DE (CoDE) by using three trial vector generation strategies and three control parameter settings. Experimental results on the Congress on Evolutionary Computation (CEC) 2005 benchmark show that CoDE is very competitive.

Although DE has been successfully applied to many applications, there are few papers to study its convergence characteristic. In [37], Ghosh *et al.* take a first significant step toward the convergence analysis of a canonical DE (DE/rand/1/bin) algorithm. It has pointed out that the analysis is applicable to a class of continuous and real-valued objective functions that possesses a unique global optimum (but may have multiple local optima). Theoretical results have been substantiated with relevant computer simulations [37].

In this section, we only presented a brief overview of some recently proposed DE variants; a comprehensive survey can be found in [38] and [39].

IV. GBDE

A. BBPSO and BBDE

PSO is a swarm intelligence-based algorithm, which is inspired by the behavior of birds flocking and fish schooling [40]. In PSO, each particle is attracted by its personal best position $(pbest)$ and the global best position $(gbest)$ found so far. Some theoretical studies [41], [42] proved that each particle converges to the weighted average of $pbest$ and $gbest$

$$\lim_{G\to+\infty} X_{i,G} = \frac{c_1 \cdot pbest_{i,G} + c_2 \cdot gbest}{c_1 + c_2} \tag{8}$$

where c_1 and c_2 are two leaning factors in PSO.

Based on PSO's convergence behavior, Kennedy [11] developed a BBPSO. This new version of PSO eliminates the velocity term, and the position is updated as follows:

$$X_{i,G+1} = N\left(\frac{gbest + pbest_{i,G}}{2}, |gbest - pbest_i|\right) \tag{9}$$

where $X_{i,G+1}$ is the position of the ith particle in the population and N represents a Gaussian distribution with mean $(gbest + pbest_{i,G})/2$ and standard deviation $|gbest - pbest_{i,G}|$.

The particle positions are sampled by the aforementioned Gaussian distribution. The BBPSO facilitates initial exploration, due to large deviation. As the number of generation increases, the deviation approaches to zero, by focusing on the exploitation of the $pbest$ and $gbest$ [43].

Kennedy [11] proposed a modified BBPSO by using an alternative mechanism as follows:

$$X_{i,j,G+1} = \begin{cases} N\Big(\frac{gbest+pbest_{i,j,G}}{2}, \\ \qquad |gbest - pbest_{i,j,G}|\Big), & \text{if } \mathrm{rand}_j(0,1) > 0.5 \\ pbest_{i,j,G}, & \text{otherwise} \end{cases} \tag{10}$$

where $\mathrm{rand}_j(0,1)$ is a random value within [0, 1] for the jth dimension. For the alternative mechanism, there is a 50% chance that the search process is focusing on the previous best positions.

Inspired by the modified BBPSO and DE, Omran *et al.* [12] proposed a new and efficient DE variant, called BBDE, in which the vector is updated as follows:

$$x_{i,j,G} = \begin{cases} p_{i,j,G} + r_{2,j} \cdot (x_{i_1,j,G} - x_{i_2,j,G}), & \text{if } \mathrm{rand}_j(0,1) > CR \\ pbest_{i_3,G}, & \text{otherwise} \end{cases} \tag{11}$$

where i_1, i_2, and i_3 are three indices chosen from the set $\{1, 2, \ldots, N_p\}$ with $i_1 \neq i_2 \neq i$, $r_{2,j}$ is a random value within [0, 1] for the jth dimension, $\mathrm{rand}_j(0,1)$ is a random value within [0, 1] for the jth dimension, and $p_{i,j,G}$ is defined by

$$p_{i,j,G} = r_{1,j} \cdot pbest_{i,j,G} + (1 - r_{1j}) \cdot gbest_j \tag{12}$$

where $r_{1,j}$ is a random value within [0, 1] for the jth dimension.

B. Proposed Approach (GBDE)

In BBPSO, the new position is generated by a Gaussian distribution for sampling the search space based on the current vector and the best vector at the current generation. As a result, the new position vector will be centered around the weighted average of $pbest_i$ and $gbest$. At the initial evolutionary stages, the search process focuses on exploration due to the large deviation (initially, $pbest_i$ will be far from $gbest$). With increasing number of generations, the deviation becomes smaller, and the search process will focus on exploitation.

From the search behavior of BBPSO, the Gaussian sampling is a fine tuning procedure which starts during exploration and is continued to exploitation. This can be beneficial for the search of many evolutionary optimization algorithms. Based on this idea, a parameter-free DE algorithm, called GBDE, is proposed in this paper. In the GBDE, a Gaussian mutation strategy is defined by

$$V_{i,G} = N(\mu, \sigma) \tag{13}$$

where $N(\mu, \sigma)$ is a Gaussian random function with mean μ and standard deviation σ, where $\mu = (X_{\text{best},G} + X_{i,G})/2$ and $\sigma = |X_{\text{best},G} - X_{i,G}|$.

Similar to the classical DE, GBDE also employs the same crossover scheme as follows (without loss of generality, this paper only considers the binomial crossover scheme):

$$u_{i,j,G} = \begin{cases} v_{i,j,G}, & \text{if } \text{rand}_j(0,1) \leq CR \vee j = j_{\text{rand}} \\ x_{i,j,G}, & \text{otherwise} \end{cases} \tag{14}$$

where CR is the probability of crossover.

Like other DE variants, the parameter CR also greatly affects the performance of GBDE. When the crossover probability CR is a large value, the trial vector $U_{i,G}$ will have a large chance of inheriting the components of the mutant vector $V_{i,G}$ which is generated around the $X_{i,G}$ and $X_{\text{best},G}$. With increasing number of generations, $X_{i,G}$ gradually approaches to $X_{\text{best},G}$, and $U_{i,G}$ approaches to $X_{\text{best},G}$. It demonstrates that a large CR will accelerate the trial vector moving to the best vector. However, this may result in a premature convergence. When CR is a small value, the trial vector $U_{i,G}$ will inherit many components from its parent vector $X_{i,G}$. It means that $U_{i,G}$ is very similar to $X_{i,G}$. This will slow down the convergence speed. Therefore, the value of CR should not be fixed during the evolution.

In this paper, a simple self-adaptive strategy is proposed to dynamically update CR. In some well-known self-adaptive DE algorithms, such as SaDE [8] and JADE [9], the initial value of CR is independently generated by a normal distribution of mean 0.5 and standard deviation 0.1. After a predefined number of generations, the CR is updated according to the search experiences of successful crossover probabilities. By following this idea, a new self-adaptive CR strategy is proposed as follows:

$$CR_{i,G+1} = \begin{cases} CR_{i,G}, & \text{if } f(U_{i,G}) \leq f(X_{i,G}) \\ N(0.5, 0.1), & \text{otherwise} \end{cases} \tag{15}$$

where $N(0.5, 0.1)$ is a random value generated by a normal distribution of mean 0.5 and standard deviation 0.1.

When $f(U_{i,G}) \leq f(X_{i,G})$, it means that the current CR has a higher chance to improve the quality of candidate solutions during the next generation. Keeping the current CR may obtain better solutions. When the trial vector is worse than the current vector, it indicates that the current CR has a lower chance to generate better candidate solutions. Therefore, changing the current CR may be more suitable for the evolution process. By the suggestions of Qin *et al.* [8] and Zhang and Sanderson [9], a normal distribution of mean 0.5 and standard deviation 0.1 is used to generate a new CR.

Algorithm 1: The Pseudocode of GBDE

```
Randomly initialize the population at generation G = 0;
while the stopping criterion is not satisfied do
  for i = 1 to N_p do
    Generate a mutant vector V_{i,G} according to (13);
    Generate a trial vector U_{i,G} according to (14);
    Evaluate the trial vector U_{i,G};
    if f(U_{i,G}) ≤ f(X_{i,G}) then
      X_{i,G+1} = U_{i,G};
      if f(U_{i,G}) < f(X_{best,G}) then
        X_{best,G} = U_{i,G};
      end
    end
    else
      X_{i,G+1} = X_{i,G};
    end
    Update CR according to (15);
  end
  G = G + 1;
end
```

Compared to the classical DE, GBDE only modifies the mutation strategy and the CR parameter. Therefore, both GBDE and the classical DE have the same time complexity $O(G_{\max} \cdot N_p \cdot D)$, where $G_{\max}$ is the maximum number of generations. The main steps of GBDE are described in Algorithm 1.

Fig. 1 illustrates the search behaviors of GBDE for a 5-D Ackley's function. At the beginning of evolution, all vectors in the population almost cover the entire search space because of the large deviation. At this stage, the algorithm mainly focuses on exploration. As the number of generations increases, the deviation gradually decreases. Then, the search region covered by the population becomes smaller and smaller, and the vectors approach toward the best vector. The search behavior of GBDE gradually transforms from exploration to exploitation.

In GBDE, the mutation strategy uses a Gaussian sampling method. Because of its randomness characteristic, it may slow down the convergence speed of GBDE. It is known that the DE/best/1 has fast convergence rate in solving unimodal and simple multimodal problems due to the attraction toward the best vector. To balance the global search ability and convergence rate, a modified GBDE (called MGBDE) is proposed by the hybridization of GBDE and DE/best/1 as follows.

In the MGBDE, each vector is randomly assigned to a mutation strategy (either the Gaussian mutation or the DE/best/1 mutation) during the population initialization. The assigned mutation strategy for each vector does not change during the search process. Consider that M_i represents the mutation strategy of X_i. It can be expressed as follows:

$$M_i = \begin{cases} \text{DE/best/1/ (Eq. 2)} & \text{if } \text{rand}_i(0,1) \leq 0.5 \\ \text{Gaussian mutation (Eq. 13)} & \text{otherwise.} \end{cases} \tag{16}$$

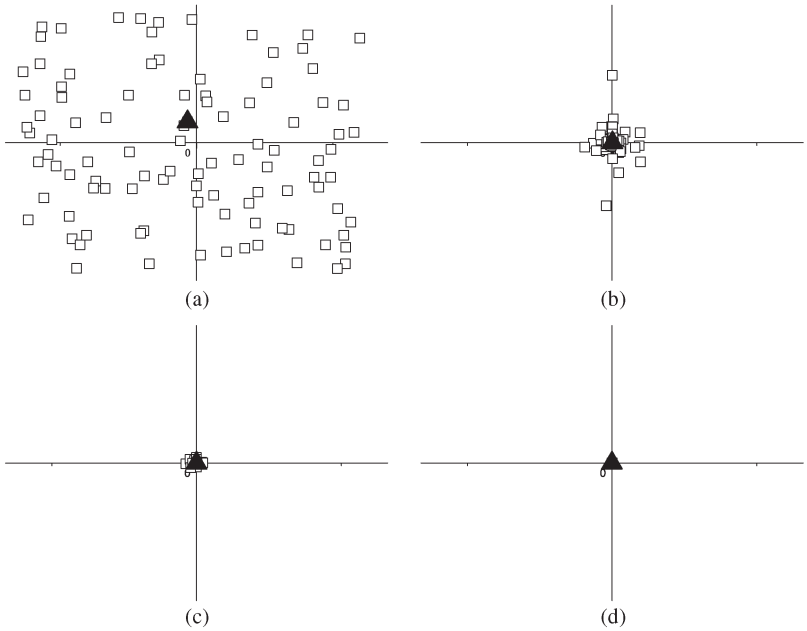

Fig. 1. Search behaviors of GBDE for a 5-D Ackley's function, where "□" represents the vectors in the current population at generation G and "▲" indicates the current best vector. (a) $G = 0$. (b) $G = 30$. (c) $G = 60$. (d) $G = 90$.

For each vector X_i, MGBDE first checks the M_i and selects a corresponding mutation strategy to generate mutant vectors. The rest of the steps of the MGBDE are the same as that with GBDE.

In the MGBDE, about 50% of the vectors in the population follow the behavior of DE/best/1, while the remaining 50% follow the GBDE. This will be helpful to balance the advantages of fast convergence rate (the attraction of DE/best/1) and exploration (the Gaussian sampling of GBDE) in MGBDE.

V. Experimental Studies on GBDE

A. Benchmark Functions

In order to verify the performance of our proposed approaches, namely, GBDE and MGBDE, a test bed of well-known 20 benchmark functions are used in the following experiments. These functions were early considered in [6], [13], and [25]. Among these problems, $f_1 - f_{13}$ are high-dimensional functions. $f_1 - f_5$ are unimodal functions. f_6 is a step function which has one minimum and is discontinuous. f_7 is a noisy quartic function. $f_8 - f_{13}$ are multimodal functions with many local minima. $f_{14} - f_{21}$ are low-dimensional multimodal functions with a few local minima. All the functions used in this paper have to be minimized. The descriptions of these benchmark functions are summarized in Table I.

B. Comparison of GBDE and MGBDE With Other DE Schemes

In this section, the proposed GBDE and MGBDE are compared with two other DE schemes including DE/rand/1 and DE/best/1. The current experiment includes three series of comparisons in terms of the following: 1) quality of the final solutions; 2) convergence speed and success rate (SR); and 3) scalability test. For common parameters, all the four contestant algorithms use the same settings, such as the same population size and stopping criterion. The specific parameter settings are listed as follows.

1) Population size: $ps = 100$ [6], [19].
2) $F = 0.5$, and $CR = 0.9$ [5], [6], [19].
3) Crossover scheme: Binomial (for all algorithms) [6], [19].
4) Stop criterion: There are two cases: 1) For the first and third experiments, each algorithm stops when the number of fitness evaluations (FEs) reaches the maximum FEs (MAX_FEs); for the first comparison, MAX_FEs $= 2.00E + 05$, and for the experiment, MAX_FEs $= 5000 \times D$, where D is the dimension of the problem; and 2) for the second experiment, each algorithm stops when the FEs reach the MAX_FEs or the best fitness value found so far is below the predefined threshold objective function value. In this case, MAX_FEs $= 1.00E + 06$, and the threshold value is given in Table III.
5) Number of runs: Each algorithm is run 50 times per function.

1) Comparison of Final Solution's Quality: The results achieved by DE/rand/1, DE/best/1/, GBDE, and MGBDE for $f_1 - f_{20}$ are summarized in Table II, where "Mean" indicates the mean best fitness value and "Std Dev" represents the corresponding standard deviation. The best results are shown in **boldface**. The comparison results among MGBDE and other algorithms are summarized as "$w/t/l$" in the last row of the table, which means that MGBDE wins in w functions, ties in t functions, and loses in l functions [10]. Fig. 2 shows the performance graphs of DE/rand/1, DE/best/1, GBDE, and MGBDE; due to the tight space limitation, some sample graphs are illustrated.

Based on the results, our approach achieves better results than other algorithms on the majority of test functions. MGBDE

TABLE I
TWENTY BENCHMARK FUNCTIONS USED IN OUR EXPERIMENTAL STUDY, WHERE D IS PROBLEM DIMENSION

Name	Function	D	Search range	Global optimum
Sphere	$f_1(x) = \sum_{i=1}^{D} x_i^2$	30	$[-100, 100]$	0
Schwefel 2.22	$f_2(x) = \sum_{i=1}^{D} \lvert x_i \rvert + \prod_{i=1}^{D} x_i$	30	$[-10, 10]$	0
Schwefel 1.2	$f_3(x) = \sum_{i=1}^{D} (\sum_{j=1}^{i} x_j)^2$	30	$[-100, 100]$	0
Schwefel 2.21	$f_4(x) = \max\{\lvert x_i \rvert, 1 \le i \le D\}$	30	$[-100, 100]$	0
Rosenbrock	$f_5(x) = \sum_{i=1}^{n} [100(x_{i+1} - x_i^2)^2 + (1 - x_i^2)^2]$	30	$[-30, 30]$	0
Step	$f_6(x) = \sum_{i=1}^{D} \lfloor x_i + 0.5 \rfloor$	30	$[-100, 100]$	0
Quartic with noise	$f_7(x) = \sum_{i=1}^{D} i \cdot x_i^4 + random[0, 1)$	30	$[-1.28, 1.28]$	0
Schwefel 2.26	$f_8(x) = \sum_{i=1}^{D} -x_i \cdot \sin(\sqrt{\lvert x_i \rvert})$	30	$[-500, 500]$	−12569.5
Rastrigin	$f_9(x) = \sum_{i=1}^{D} [x_i^2 - 10\cos 2\pi x_i + 10]$	30	$[-5.12, 5.12]$	0
Ackley	$f_{10}(x) = -20\exp(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}) - \exp(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)) + 20 + e$	30	$[-32, 32]$	0
Griewank	$f_{11}(x) = \frac{1}{4000}\sum_{i=1}^{D}(x_i)^2 - \prod_{i=1}^{D}\cos(\frac{x_i}{\sqrt{i}}) + 1$	30	$[-600, 600]$	0
Penalized 1	$f_{12}(x) = \frac{\pi}{D}\{\sum_{i=1}^{D-1}(y_i - 1)^2[1 + \sin(\pi y_{i+1})] + (y_D - 1)^2) + (10\sin^2(\pi y_1)\} + \sum_{i=1}^{D} u(x_i, 10, 100, 4),\ y_i = 1 + \frac{x_i+1}{4}$, $u(x_i, a, k, m) = \begin{cases} u(x_i, a, k, m), & x_i > a \\ 0, & -a \le x_i \le a \\ k(-x_i - a)^m, & x_i < -a \end{cases}$	30	$[-50, 50]$	0
Penalized 2	$f_{13}(x) = 0.1\{\sin^2(3\pi x_1) + \sum_{i=1}^{D-1}(x_i - 1)^2[1 + \sin^2(3\pi x_{i+1})] + (x_D - 1)^2[1 + \sin^2(2\pi x_D)]\} + \sum_{i=1}^{D} u(x_i, 5, 100, 4)$	30	$[-50, 50]$	0
Shekel's Foxholes	$f_{14}(x) = \left[\frac{1}{500} + \sum_{j=1}^{25}\frac{1}{j + \sum_{i=1}^{2}(x_i - a_{ij})^6}\right]^{-1}$	2	$[-65.536, 65.536]$	0.998004
Kowalik	$f_{15}(x) = \sum_{i=1}^{11}\left[a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4}\right]^2$	4	$[-5, 5]$	0.0003075
Six-hump Camel-back	$f_{16}(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	2	$[-5, 5]$	−1.0316285
Branin	$f_{17}(x) = (x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6)^2 + 10(1 - \frac{1}{8\pi})\cos x_1 + 10$	2	$[-5, 10]$	0.398
Goldstein-Price	$f_{18}(x) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2] \cdot [30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$	2	$[-2, 2]$	3
Shekel5	$f_{19}(x) = -\sum_{i=1}^{5}[(x_i - a_i)(x_i - a_i)^T + c_i]^{-1}$	6	$[-10, 10]$	−10.1532
Shekel7	$f_{20}(x) = -\sum_{i=1}^{7}[(x_i - a_i)(x_i - a_i)^T + c_i]^{-1}$	6	$[-10, 10]$	−10.4029
Shekel10	$f_{21}(x) = -\sum_{i=1}^{10}[(x_i - a_i)(x_i - a_i)^T + c_i]^{-1}$	6	$[-10, 10]$	−10.5364

TABLE II
RESULTS OF MEAN BEST FITNESS VALUE AND STANDARD DEVIATION FOR ALL TEST FUNCTIONS

Functions	D	DE/rand/1		DE/best/1		GBDE		MGBDE	
		Mean	Std Dev	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev
f_1	30	8.34E–24	6.45E–24	**0.00E+00**	**0.00E+00**	1.57E–36	2.33E–36	8.79E–68	5.21E–69
f_2	30	1.05E–11	4.00E–11	**7.23E–47**	**1.25E–46**	3.67E–24	1.50E–24	8.50E–41	3.38E–41
f_3	30	4.34E–03	5.65E–03	**1.68E–50**	**2.64E–50**	1.20E+02	6.63E+01	6.10E–11	2.96E–11
f_4	30	4.47E–02	7.60E–02	**2.23E–06**	**1.77E–06**	2.40E–02	2.94E–02	1.31E–05	3.46E–06
f_5	30	8.20E+00	1.73E+00	**1.33E+00**	**2.30E+00**	2.25E+01	2.68E+00	1.69E+00	2.24E+00
f_6	30	**0.00E+00**	**0.00E+00**	1.51E+02	1.67E+02	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
f_7	30	8.42E–03	7.33E–04	1.35E–02	3.27E–03	9.43E–03	5.36E–03	**2.14E–03**	**1.08E–03**
f_8	30	–7426.7	7.52E+02	–2704.6	6.95E+02	**–12569.5**	**1.34E–12**	**–12569.5**	**1.09E–12**
f_9	30	1.81E+02	2.44E+01	8.42E+01	9.97E+00	6.96E+00	5.41E+00	**3.98E+00**	**2.98E+00**
f_{10}	30	3.02E–12	1.70E–12	4.07E+00	2.28E+00	**7.69E–15**	**0.00E+00**	**7.69E–15**	**0.00E+00**
f_{11}	30	**0.00E+00**	**0.00E+00**	2.54E–02	2.69E–02	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
f_{12}	30	7.82E–25	5.62E–25	2.48E+00	2.50E–01	**1.57E–32**	**0.00E+00**	**1.57E–32**	**0.00E+00**
f_{13}	30	2.75E–23	4.19E–23	4.57E–01	1.57E–01	**1.35E–32**	**0.00E+00**	**1.35E–32**	**0.00E+00**
f_{14}	2	**0.998004**	**0.00E+00**	**0.998004**	**0.00E+00**	**0.998004**	**0.00E+00**	**0.998004**	**0.00E+00**
f_{15}	2	**–1.03163**	**0.00E+00**	**–1.03163**	**0.00E+00**	**–1.03163**	**0.00E+00**	**–1.03163**	**0.00E+00**
f_{16}	2	**0.397887**	**0.00E+00**	0.460611	1.09E–01	**0.397887**	**0.00E+00**	**0.397887**	**0.00E+00**
f_{17}	2	**3.00E+00**	**0.00E+00**	**3.00E+00**	**0.00E+00**	**3.00E+00**	**0.00E+00**	**3.00E+00**	**0.00E+00**
f_{18}	6	**–10.1532**	**0.00E+00**	–5.17297	4.31E+00	**–10.1532**	**0.00E+00**	**–10.1532**	**0.00E+00**
f_{19}	6	**–10.4029**	**0.00E+00**	–7.8526	4.42E+00	**–10.4029**	**0.00E+00**	**–10.4029**	**0.00E+00**
f_{20}	6	**–10.5364**	**1.78E–15**	–7.98132	3.36E+00	**–10.5364**	**0.00E+00**	**–10.5364**	**0.00E+00**
$w/t/l$		11/9/0		12/3/5		7/13/0		––	

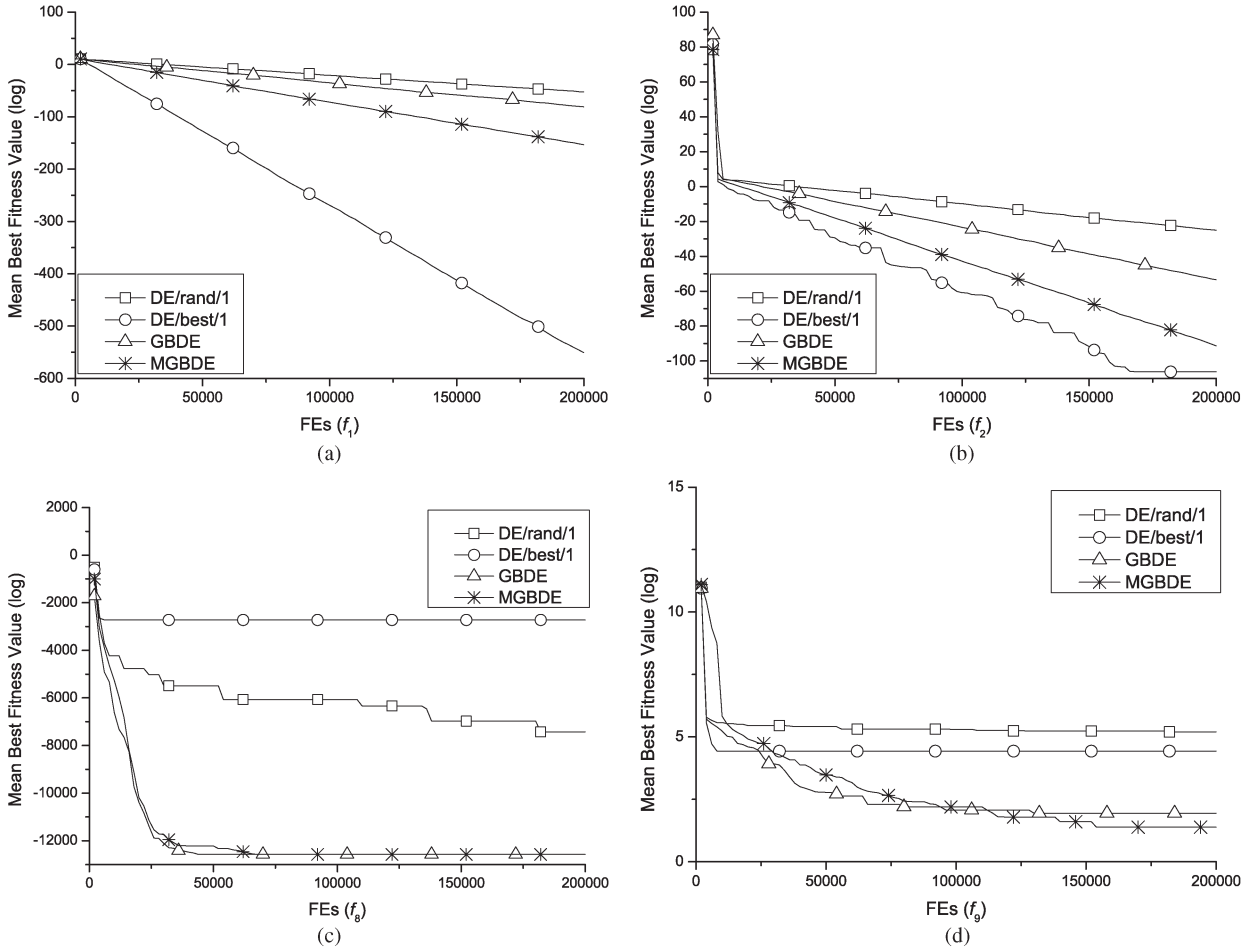

Fig. 2. Evolutionary processes of DE/rand/1, DE/best/1, GBDE, and MGBDE on some selected functions with $D = 30$. (a) Sphere (f_1). (b) Schwefel 2.22 (f_2). (c) Schwefel 2.26 (f_8). (d) Rastrigin (f_9).

outperforms DE/rand/1 on 11 functions. For the rest of the nine functions, both DE and MGBDE achieve the same results. Although DE/best/1 performs better than MGBDE on five functions, they are simple unimodal problems. For complex high-dimensional multimodal functions $(f_8 - f_{13})$, DE/best/1 could hardly achieve promising solutions. It demonstrates that DE/best/1 is only suitable for solving unimodal and simple multimodal problems. GBDE achieves promising solutions on all test functions except for f_3, f_5, and f_9. Although f_3 is a unimodal function, the rotated search space hinders the sampling of GBDE. It is pointed out in [44] that the Rosenbrock problem (f_5) is multimodal when $D > 3$. Moreover, its global optimum is inside a long narrow parabolic shaped flat valley. It is easy to find the valley but difficult to converge to the global optimum.

To accelerate the convergence rate of GBDE, the MGBDE is proposed by the hybridization of GBDE and DE/best/1. This balances the global search ability of GBDE and the fast convergence rate of DE/best/1 in MGBDE. The MGBDE achieves better results than GBDE on seven functions. For the rest of the 13 functions, both GBDE and MGBDE can find the global optimum. It demonstrates that the MGBDE improvement is caused by the proposed hybrid mutation strategies.

2) Comparison of the Convergence Speed and SR: In order to compare the convergence rate of different algorithms, we select a threshold value of the objective function for each test function. For functions with minima at zero, the threshold is set to 1e-10. For other functions, the threshold values are listed in the third column of Table III. The stopping criterion is that each algorithm is terminated when the best fitness value so far is below the predefined threshold value or the number of FEs reaches to its maximum value (1.00E+06). For each test function, each algorithm is run 50 times. The mean number of FEs required to converge to the threshold and successful rate are recorded.

Table III presents the results of the mean number of FEs (Mean FEs) and successful rate (SR), where "NA" represents that no runs of the corresponding algorithm converged below the predefined threshold before meeting the maximum number of FEs. The best results among the four algorithms are shown in **boldface**. As seen, DE/best/1 shows faster convergence speed on five unimodal and two simple multimodal functions, while it hardly converges to the threshold for complex multimodal functions. GBDE converges faster than DE/best/1 and DE/rand/1 on all multimodal functions with many local minima $(f_8 - f_{13})$. By the hybridization of GBDE and DE/best/1, MGBDE significantly reduces the number of FEs. Particularly for f_3 and f_5, GBDE fails to solve them, while MGBDE successfully converges to the threshold value. From the results of total average FEs, MGBDE costs the lowest FEs to reach the threshold. The acceleration rates between MGBDE and DE/rand/1,

TABLE III
RESULTS OF MEAN NUMBER OF FES AND SUCCESSFUL RATE UNDER PREDEFINED ACCURACY LEVEL (THRESHOLD)

Functions	D	Threshold	DE/rand/1		DE/best/1		GBDE		MGBDE	
			Mean FEs	SR	Mean FEs	SR	Mean FEs	SR	Mean FEs	SR
f_1	30	1.00E–10	1.10E+05	**100%**	**1.28E+04**	**100%**	7.67E+04	**100%**	3.90E+04	**100%**
f_2	30	1.00E–10	1.84E+05	**100%**	**3.53E+04**	**100%**	9.26E+04	**100%**	5.22E+04	**100%**
f_3	30	1.00E–10	4.17E+05	**100%**	**5.82E+04**	**100%**	NA	0%	3.08E+05	**100%**
f_4	30	1.00E–10	NA	0%	**3.33E+05**	**100%**	6.48E+05	**100%**	3.46E+05	**100%**
f_5	30	1.00E–10	4.12E+05	**100%**	**6.15E+04**	**100%**	NA	0%	3.92E+05	**100%**
f_6	30	1.00E–10	2.97E+04	**100%**	NA	0%	2.07E+04	**100%**	**1.23E+04**	**100%**
f_7	30	1.00E–02	9.87E+04	**100%**	1.41E+05	**100%**	1.28E+05	**100%**	**4.61E+04**	**100%**
f_8	30	-1.25E+04	4.29E+05	**100%**	NA	0%	**4.37E+04**	**100%**	6.80E+04	**100%**
f_9	30	1.00E–10	NA	0%	NA	0%	2.92E+05	**100%**	**2.78E+05**	**100%**
f_{10}	30	1.00E–10	1.72E+05	**100%**	NA	0%	1.17E+05	**100%**	**7.46E+04**	**100%**
f_{11}	30	1.00E–10	1.11E+05	**100%**	2.15E+05	84%	7.42E+04	**100%**	**3.64E+04**	**100%**
f_{12}	30	1.00E–10	1.01E+05	**100%**	NA	0%	6.83E+04	**100%**	**4.22E+04**	**100%**
f_{13}	30	1.00E–10	1.07E+05	**100%**	NA	0%	7.58E+04	**100%**	**3.92E+04**	**100%**
f_{14}	2	0.9981	3.90E+03	**100%**	1.25E+03	**100%**	1.20E+03	**100%**	**1.10E+03**	**100%**
f_{15}	2	–1.0316	3.10E+03	**100%**	**1.02E+03**	**100%**	1.15E+03	**100%**	1.60E+03	**100%**
f_{16}	2	0.398	2.05E+03	**100%**	NA	0%	**8.90E+02**	**100%**	1.80E+03	**100%**
f_{17}	2	3	6.10E+03	**100%**	**2.40E+03**	**100%**	4.10E+03	**100%**	7.60E+03	**100%**
f_{18}	6	–10.15	**5.90E+03**	**100%**	NA	0%	7.10E+03	**100%**	7.80E+03	**100%**
f_{19}	6	–10.4	4.80E+03	**100%**	NA	0%	5.60E+03	**100%**	**4.20E+03**	**100%**
f_{20}	6	–10.53	5.20E+03	**100%**	NA	0%	3.60E+03	**100%**	**3.90E+03**	**100%**
Total Average			2.10E+05	90%	4.93E+05	54.2%	1.83E+05	90%	**8.81E+04**	**100%**

DE/best/1, and GBDE are 2.38, 5.60, and 2.07, respectively. Both DE/rand/1 and GBDE achieve the same SR (90%) because they fail to solve two functions. DE/best/1 fails to solve nine functions and obtains the lowest SR. MGBDE successfully converges to the threshold value for all test functions and achieves the highest SR.

3) Scalability Test: The performance of most EAs (including DE) deteriorates quickly with the growth of the dimensionality of the problem. The main reason is that, in general, the complexity of the problem (search space) increases exponentially with its dimension. Here, we show a scalable test of GBDE and MGBDE for $D = 50$ and 100.

Table IV summarizes the comparison results of four DE variants for $D = 50$ and 100, where "Mean Error" represents mean function error values. From the results, it can be seen that both GBDE and MGBDE are not always affected by the growth of dimensionality. For f_5, f_7, and f_8, GBDE achieves similar performance when the dimension increases from 50 to 100. The performance of MGBDE deteriorates quickly with the growth of dimension for five functions ($f_3 - f_5$, f_8, and f_9). For the rest of the eight functions (f_1, f_2, f_6, f_7, and $f_{10} - f_{14}$), the growth of dimension does not affect the performance of MGBDE.

C. Comparison of GBDE and MGBDE With Some State-of-the-Art DE Variants and Similar Bare-Bones Algorithms

In this section, we compare GBDE and MGBDE with five other recently proposed DE variants and two BBPSO algorithms. The involved algorithms are listed as follows:

1) self-adapting DE (jDE) [6];
2) DE with neighborhood search (NSDE) [16];
3) ODE [19];
4) DEGL with self-adaptive weight factor (DEGL/SAW);
5) BBDE [12];
6) BBPSO [11];
7) modified BBPSO [11];
8) the proposed GBDE and MGBDE.

We have two series of experiments: 1) comparison of GBDE and MGBDE with jDE, NSDE, ODE, and DEGL/SAW and 2) comparison of GBDE and MGBDE with BBDE, BBPSO, and modified BBPSO. The former aims to check whether GBDE and MGBDE are better or worse than some state-of-the-art DE variants, while the latter focuses on investigating how good GBDE and MGBDE are within the similar context of bare-bones algorithms.

1) Comparison of GBDE and MGBDE With jDE, NSDE, ODE, and DEGL/SAW: This section presents the comparison of GBDE and MGBDE with some state-of-the-art DE variants, including jDE, NSDE, ODE, and DEGL/SAW. The parameter settings for GBDE and MGBDE are kept the same as before. For jDE, NSDE, and DEGL/SAW, the best set of parameters was employed from the relevant literature [25]. For ODE, we take $F = 0.5, CR = 0.9, N_p = 100$, and the rate of opposition-based jumping $J_r = 0.3$ according to the suggestions of Rahnamayan *et al.* [19]. For all algorithms, we use the same stopping criterion. Each algorithm runs on a function and stops when the number of FEs reaches the maximum value (MAX_FEs). In this comparison, MAX_FEs is set to 5.00e+05 [25] (see the third column in Table V).

The results of mean fitness values achieved by the six DE variants are presented in Table V, where "$w/t/l$" summarizes the competition results among MGBDE and other algorithms.

TABLE IV
MEAN FUNCTION ERROR VALUES FOR $D = 50$ AND $D = 100$

Functions	D = 50				D = 100			
	DE/rand/1 Mean Error	DE/best/1 Mean Error	GBDE Mean Error	MGBDE Mean Error	DE/rand/1 Mean Error	DE/best/1 Mean Error	GBDE Mean Error	MGBDE Mean Error
f_1	2.85E–17	**4.88E–141**	1.21E–18	7.34E–55	6.74E–20	**2.36E–66**	1.65E–08	6.76E–52
f_2	5.97E–09	1.31E–21	7.88E–13	**2.54E–28**	8.75E–11	7.93E+00	2.04E–07	**1.46E–31**
f_3	1.24E+02	**3.79E–16**	1.07E+04	3.11E+01	8.04E+03	**1.74E+01**	8.93E+04	2.15E+03
f_4	1.52E+01	1.64E+01	5.68E+00	**1.77E+00**	4.31E+01	4.61E+01	3.96E+01	**2.88E+01**
f_5	3.06E+01	**2.79E–21**	2.53E+01	9.25E–02	1.31E+02	**3.99E+00**	1.78E+02	3.22E+01
f_6	**0.00E+00**	4.84E+02	**0.00E+00**	**0.00E+00**	**0.00E+00**	6.21E+03	**0.00E+00**	**0.00E+00**
f_7	4.07E–02	1.15E+01	2.97E–02	**5.28E–03**	7.19E–02	1.09E+02	1.33E–02	**7.61E–03**
f_8	1.35E+04	1.60E+04	**0.00E+00**	**0.00E+00**	2.78E+04	3.27E+04	**0.00E+00**	1.58E+03
f_9	3.20E+02	1.68E+02	2.69E+01	**1.87E+01**	3.07E+02	4.39E+02	8.46E+01	**7.95E+01**
f_{10}	5.74E–10	1.05E+01	3.18E–10	**1.84E–14**	8.75E–05	1.31E+01	1.57E–05	**5.39E–14**
f_{11}	**0.00E+00**	1.99E–14	**0.00E+00**	**0.00E+00**	1.13E–08	2.32E–13	7.38E–09	**0.00E+00**
f_{12}	1.97E–18	2.49E–01	7.35E–18	**2.92E–21**	1.09E–06	7.49E–01	8.62E–07	**1.67E–21**
f_{13}	7.70E–18	1.22E+00	1.50E–15	**4.84E–31**	9.91E–06	2.75E–01	8.69E–06	**1.52E–31**

TABLE V
RESULTS OF MEAN BEST FITNESS VALUES FOR THE SIX DE VARIANTS

Functions	D	MAX_FEs	jDE Mean	NSDE Mean	ODE Mean	DEGL/SAW Mean	GBDE Mean	MGBDE Mean
f_1	25	5.00e+05	4.04E–35	9.55E–35	1.64E–38	8.78E–37	2.02E–83	**6.31E–128**
f_2	25	5.00E+05	8.34E–26	8.94E–30	3.51E–11	4.95E–36	8.10E–56	**5.86E–72**
f_3	25	5.00E+05	4.28E–14	3.06E–09	7.18E–04	**1.21E–26**	9.21E–01	9.50E–24
f_4	25	5.00E+05	3.02E–14	2.09E–11	**1.28E–27**	4.99E–15	2.40E–09	2.67E–24
f_5	25	5.00E+05	**5.65E–26**	2.65E–25	1.09E–04	6.89E–25	1.96E+01	2.83E–11
f_6	25	5.00E+05	1.67E–36	4.04E–28	**0.00E+00**	9.56E–48	**0.00E+00**	**0.00E+00**
f_7	25	5.00E+05	3.76E–02	4.35E–03	1.31E–03	**1.05E–07**	2.49E–03	4.56E–04
f_8	25	5.00E+05	**–10475**	**–10475**	–5982.1	**–10475**	**–10475**	**–10475**
f_9	25	5.00E+05	6.73E–24	4.84E–21	6.53E+01	**5.85E–25**	2.23E–13	4.97E–17
f_{10}	25	5.00E+05	7.83E–15	5.97E–10	4.14E–15	**5.98E–23**	4.14E–15	4.14E–15
f_{11}	25	5.00E+05	1.83E–28	7.93E–26	0.00E+00	2.99E–36	**0.00E+00**	**0.00E+00**
f_{12}	25	5.00E+05	9.37E–24	5.85E–21	**1.88E–32**	7.21E–27	**1.88E–32**	**1.88E–32**
$w/t/l$			9/1/2	9/1/2	8/3/1	6/1/5	7/5/0	––

TABLE VI
AVERAGE RANKINGS ACHIEVED BY FRIEDMAN TEST

Algorithms	Average Rankings
MGBDE	**4.67**
DEGL/SAW	4.33
GBDE	3.41
ODE	3.33
jDE	2.91
NSDE	2.33

TABLE VII
WILCOXON TEST BETWEEN MGBDE AND OTHER FIVE DE VARIANTS

MGBDE vs.	p-values
DEGL/SAW	4.24E–01
GBDE	**1.80E–02**
ODE	**2.51E–02**
jDE	1.31E–01
NSDE	9.12E-02

The results of jDE, NSDE, and DEGL/SAW are taken from [25, Tbl. VI]. Among the six algorithms, the best results are shown in **boldface**. Here, we only list the results for functions $f_1 - f_{12}$. The main reason is that all the six DE variants obtain the same results for functions $f_{14} - f_{20}$. For f_{13}, the literature [25] uses a different function.

From the results, it can be seen that our approaches, GBDE and MGBDE, achieve better results than jDE, NSDE, and ODE on the majority of test functions. MGBDE outperforms jDE and NSDE on nine functions, while jDE only preforms better than MGBDE on f_5. ODE achieves better results than MGBDE on f_4, while MGBDE outperforms ODE on eight functions. Both MGBDE and DEGL/SAW almost have the same performance on f_8. DEGL/SAW wins on five functions, while MGBDE wins on six. MGBDE outperforms GBDE on seven functions, which mainly focuses on unimodal functions. For multimodal functions, both GBDE and MGBDE obtain the same performance except for f_9. These improvements are caused by the introduction of the DE/best/1 strategy.

In order to compare the performance of multiple algorithms on the test suite, we conducted the Friedman test [45]. Table VI shows the average ranking of the six DE algorithms. The highest ranking is shown in **boldface**. As seen, the

TABLE VIII
RESULTS OF MEAN BEST FITNESS VALUES FOR THE BARE-BONES ALGORITHMS

Functions	D	MAX_FEs	BBPSO Mean	Modified BBPSO Mean	BBDE Mean	GBDE Mean	MGBDE Mean
f_1	30	5.00E+04	**0.00E+00**	**0.00E+00**	**0.00E+00**	1.03E–13	2.19E–24
f_2	30	5.00E+04	4.33E+00	**0.00E+00**	**0.00E+00**	6.99E–10	3.46E–16
f_3	30	5.00E+04	7.23E+03	6.88E+03	5.65E+01	2.08E+03	**5.39E+01**
f_5	30	5.00E+04	1.56E+04	7.71E+01	4.79E+01	1.89E+01	**1.76E+01**
f_6	30	5.00E+04	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
f_8	30	5.00E+04	–9091.02	–10471.8	–11649	**–12273.4**	–12087.3
f_9	30	5.00E+04	8.76E+01	1.35E+01	3.74E+01	2.42E+01	**1.29E+01**
f_{10}	30	5.00E+04	2.26E+00	**0.00E+00**	**0.00E+00**	4.67E–07	2.13E–12
f_{11}	30	5.00E+04	1.12E–01	8.75E–04	6.57E–04	9.20E–08	**2.26E–13**
f_{15}	30	5.00E+04	**–1.03163**	**–1.03163**	**–1.03163**	**–1.03163**	**–1.03163**
$w/t/l$			7/2/1	5/2/3	5/2/3	7/2/1	−−

performance of the six algorithms can be sorted by average ranking into the following order: MGBDE, DEGL/SAW, GBDE, ODE, jDE, and NSDE. The best average ranking was obtained by the MGBDE algorithm, which outperforms the other five algorithms.

Aside from the Friedman test, we also conduct Wilcoxon's test to recognize significant differences between the behavior of two algorithms [46], [47]. Table VII shows the p-values of applying Wilcoxon's test among MGBDE and other five DE algorithms. The p-values below 0.05 (the significant level) are shown in **boldface**. From the results, it can be seen that MGBDE is significantly better than ODE and GBDE. Although MGBDE is not significantly better than the rest of the three algorithms, it outperforms them according to the results of average ranking.

2) Comparison of GBDE and MGBDE With Similar Bare-Bones Algorithms: This section compares GBDE and MGBDE with similar bare-bones algorithms, including BBPSO, modified BBPSO, BBDE, GBDE, and MGBDE. The parameter settings are described as follows. For all algorithms, we take $N_p = 50$ and MAX_FEs $= 5.00E + 04$ which were employed in the literature [12]. For BBDE, $CR = 0.9$. Each algorithm runs on a function and stops when the number of FEs reaches the MAX_FEs.

The results of mean fitness values achieved by the five bare-bones algorithms are given in Table VIII, where "$w/t/l$" summarizes the competition results among MGBDE and other four algorithms. The results of BBPSO, modified BBPSO, and BBDE are taken from [12, Tbls. I and II]. Among the five algorithms, the best results are shown in **boldface**.

As seen, MGBDE outperforms BBPSO on seven functions, while BBPSO only achieves better results on the sphere function. The modified BBPSO improves the performance of BBPSO on some functions, and it achieves better results than MGBDE on three functions, while MGBDE outperforms it on five functions. BBDE performs better than MGBDE on three functions, while MGBDE wins on five. Compared to the last experiments, we use smaller population size and less MAX_FEs in this comparison, but MGBDE still outperforms GBDE on most functions. From the whole comparison, the advantages of BBPSO, modified BBPSO, and BBDE mainly focus on unimodal functions, such as $f_1 - f_3$, while GBDE and MGBDE show better performance on multimodal functions except for f_{10} and f_{15}.

TABLE IX
AVERAGE RANKINGS ACHIEVED BY FRIEDMAN TEST

Algorithms	Average Rankings
MGBDE	**3.80**
BBDE	3.40
Modified BBPSO	3.10
GBDE	3.00
BBPSO	1.70

TABLE X
WILCOXON TEST BETWEEN MGBDE AND OTHER BARE-BONES ALGORITHMS

MGBDE vs.	p-values
BBDE	9.29E–02
Modified BBPSO	9.29E–02
GBDE	1.23E–01
BBPSO	**1.73E–02**

Table IX shows the average ranking of the six DE algorithms. The highest ranking is shown in **boldface**. As seen, the performance of the five bare-bones algorithms can be sorted by average ranking into the following order: MGBDE, BBDE, modified BBPSO, GBDE, and BBPSO. The best average ranking was obtained by the MGBDE algorithm, which outperforms the other four algorithms.

Table X shows the p-values of applying Wilcoxon's test between MGBDE and the other four bare-bones algorithms. The p-values below 0.05 (the significant level) are shown in **boldface**. As seen, MGBDE is significantly better than BBPSO. Although MGBDE is not significantly better than the rest of the three algorithms, it outperforms them according to the results of average ranking.

D. Test on CEC 2005 Shifted and Rotated Benchmarks

To further verify the performance of GBDE and MGBDE, a set of ten CEC 2005 shifted and rotated benchmark functions are used [48]. Simple descriptions of these functions are listed in Table XI. More detailed definitions of them can be found in [49].

TABLE XI
TEN CEC 2005 BENCHMARK FUNCTIONS USED IN THE EXPERIMENTS, WHERE D IS THE DIMENSION AND $f(x^o)$ IS THE GLOBAL OPTIMUM

F	Name	D	$f(x^o)$
F_1	Shifted Sphere Function	30	–450
F_2	Shifted Schwefel's Problem 1.2	30	–450
F_3	Shifted Rotated High Conditioned Elliptic Function	30	–450
F_4	Shifted Schwefel's Problem 1.2 with Noise	30	–450
F_5	Schwefel's Problem 2.6	30	–310
F_6	Shifted Rosenbrock's Function	30	390
F_7	Shifted Rotated Griewank's Function	30	–180
F_8	Shifted Rotated Ackley's Function	30	–140
F_9	Shifted Rastrigin's Function	30	–330
F_{10}	Shifted Rotated Rastrigin's Function	30	–330

In this section, we compare GBDE and MGBDE with three other recently proposed DE variants. The involved algorithms are listed as follows:

1) self-adapting DE (jDE) [6];
2) self-adaptive DE (SaDE) [8];
3) DE with ensemble of parameters and mutation strategies (EPSDE) [50];
4) the proposed GBDE and MGBDE.

For jDE, SaDE, and EPSDE, we use the same parameter settings for these three methods as in their original papers. For EPSDE, there are two versions published in [31] and [50], respectively. In this paper, the journal version of EPSDE [50] is utilized. For GBDE and MGBDE, the population size N_p is set to 100. By the suggestions of Suganthan *et al.* [49], the number of MAX_FEs is set to 3.00E+05 for all algorithms. For each test function, each algorithm is run 25 times. Throughout the experiments, the mean and standard deviation of the function error value $(f(x) - f(x^o))$ are reported, where x is the best solution found by the algorithm in a run and x^o is the global optimum of the test function.

Table XII presents the results of the five DE variants, where "Mean Error" represents mean function error values, "Std Dev" indicates the standard deviation, and "$w/t/l$" summarizes the competition results among MGBDE and other algorithms. For each test function, the best results among the five algorithms are shown in **boldface**.

From the results, all the five algorithms could converge to the global optimum on F_1. For F_8, they fall into local minima. Compared to jDE and EPSDE, MGBDE performs better on five functions, while jDE and EPSDE outperform MGBDE on three functions. SaDE achieves better results than MGBDE on F_9, while MGBDE performs better for the rest of the seven functions. Compared to GBDE, MGBDE significantly improves the quality of solutions on the majority of test functions. Particularly for F_2 and F_6, GBDE suffers from premature convergence, while MGBDE could achieve promising solutions.

E. Comparative Study on Two Real-World Optimization Problems

In this section, we investigate the performance of our approaches over two real-world problems, viz., the frequency modulated (FM) sound synthesis problem and the spread spectrum radar polyphase code design problem [51].

1) FM Sound Synthesis Problem: FM sound synthesis plays an important role in several modern music systems. It provides a simple and efficient method creating complex sound timbres. This section applies the proposed DE-based algorithms to optimize the parameters of an FM synthesizer. Some related works on solving this problem using evolutionary methods can be found in [25], [52], and [53]. The details of the problems are described as follows.

The FM sound synthesis aims to optimize the parameter of an FM synthesizer with a D-dimensional vector X. In this paper, we only consider the case of $D = 6$ by the suggestions of the literature [25], [53]. The objective of this problem is to optimize a six-dimensional vector $X = \{a_1, w_1, a_2, w_2, a_3, w_3\}$ of the sound wave given in (17). The problem is to generate a sound [generated by (17)] similar to the object sound [generated by (18)]. The formulas for the estimated sound wave and the target sound are given as follows [53]:

$$y(t) = a_1 \sin\left(w_1 t\theta + a_2 \sin\left(w_2 t\theta + a_3 \sin(w_3 t\theta)\right)\right) \quad (17)$$

$$y_0(t) = 1.0 \sin\left(0.5 t\theta - 1.5 \sin\left(4.8 t\theta + 2.0 \sin(4.9 t\theta)\right)\right) \quad (18)$$

where $\theta = 2\pi/100$ and $X_i \in [-6.5, 6.35]$.

The goal of this problem is to minimize the sum of squared errors between the estimated sound and the target sound, as given by (18). This problem is a highly complex multimodal one having strong epistasis, with minimum value $f(X) = 0$ [53]

$$f(X) = \sum_{t=0}^{100} \left(y(t) - y_0(t)\right)^2. \quad (19)$$

In the experiment, GBDE and MGBDE as well as the other four DE variants are applied to solve this problem. For all algorithms, the MAX_FEs is set to 1.00E+05 according to the suggestions of the literature [25]. For the rest of the parameter settings, we keep them the same as described in Section V-C. Table XIII reports the results of the six DE variants over the FM synthesizer problem. Results for jDE, NSDE, and DEGL/SAW are taken from [25, Tbl. XV]. As seen, MGBDE achieves better results than the other five algorithms in terms of the quality of the final solutions.

2) Spread Spectrum Radar Polyphase Code Design Problem: The spread spectrum radar polyphase code design problem is known for designing radar systems [25], [51]. It can be formally defined as follows:

$$\text{Global } \min f(X) = \max\left\{\varphi_1(X), \ldots, \varphi_{2m}(X)\right\} \quad (20)$$

where $X = \{(x_1, \ldots, x_D) \in R^D | 0 \le x_j \le 2\pi\}$, $m = 2D - 1$, and

$$\varphi_{2i-1}(X) = \sum_{j=i}^{D} \cos\left(\sum_{k=|2i-j-1|-1}^{j} x_k\right), \quad i = 1, 2, \ldots, D$$

TABLE XII
RESULTS FOR THE TEN CEC 2005 BENCHMARK FUNCTIONS

Functions	jDE		SaDE		EPSDE		GBDE		MGBDE	
	Mean Error	Std Dev	Mean Error	Std Dev	Mean Error	Std Dev	Mean Error	Std Dev	Mean Error	Std Dev
F_1	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
F_2	1.11E–06	1.96E–06	8.26E–06	1.65E–05	**4.23E–26**	**4.07E–26**	1.49E+01	1.18E+01	2.96E–08	5.63E–08
F_3	1.98E+05	1.10E+05	4.27E+05	2.08E+05	8.74E+05	3.28E+06	7.65E+06	3.39E+05	**1.53E+05**	**1.09E+05**
F_4	**4.40E–02**	**1.26E–01**	1.77E+02	2.67E+02	3.49E+02	2.23E+03	2.91E+02	4.68E+01	8.68E+00	2.52E+00
F_5	**5.11E+02**	**4.40E+02**	3.25E+03	5.90E+02	1.40E+03	7.12E+02	2.45E+03	8.91E+02	1.89E+03	6.77E+02
F_6	2.35E+01	2.50E+01	5.31E+01	3.25E+01	6.38E–01	1.49E+00	2.26E+01	1.34E+01	**5.11E–08**	**3.82E–08**
F_7	1.18E–02	7.78E–03	1.57E–02	1.38E–02	1.77E–02	1.34E–02	8.12E–03	3.96E–03	**2.15E–03**	**2.59E–03**
F_8	**2.09E+01**	**4.86E–02**	**2.09E+01**	**4.33E–01**	**2.09E+01**	**5.81E–02**	**2.09E+01**	**5.39E–02**	**2.09E+01**	**6.27E–02**
F_9	**0.00E+00**	**0.00E+00**	2.39E–01	4.33E–01	3.98E–02	1.99E–01	3.98E+00	1.31E+00	1.98E+00	1.57E+00
F_{10}	5.54E+01	8.46E+00	4.72E+01	1.01E+01	5.36E+01	3.03E+01	3.00E+00	1.14E+00	**1.99E+00**	**1.03E+00**
$w/t/l$	5/2/3		7/2/1		5/2/3		8/2/0		– –	

TABLE XIII
RESULTS OF MEAN AND STANDARD DEVIATION ACHIEVED BY SIX ALGORITHMS FOR THE FM SYNTHESIS PROBLEM

Algorithm	Mean	Std Dev
jDE	7.84E–02	5.83E–03
NSDE	9.46E–03	6.92E–01
ODE	1.58E–22	3.24E–23
DEGL/SAW	4.82E–09	6.26E–08
GBDE	2.52E–13	3.68E–12
MGBDE	**2.31E–28**	**7.29E–28**

TABLE XIV
RESULTS OF MEAN AND STANDARD DEVIATION ACHIEVED BY SIX ALGORITHMS FOR THE SPREAD SPECTRUM RADAR POLYPHASE CODE DESIGN PROBLEM

Algorithm	$D = 19$		$D = 20$	
	Mean	Std Dev	Mean	Std Dev
jDE	7.59E–01	3.88E–05	8.34E–01	6.53E–01
NSDE	7.61E–01	4.72E–03	8.43E–01	3.44E–02
ODE	7.81E–01	2.53E–01	8.62E–01	2.03E–01
DEGL/SAW	7.44E–01	5.84E–01	**8.03E–01**	**2.73E–03**
GBDE	7.55E–01	2.29E–01	8.38E–01	1.95E–01
MGBDE	**7.41E–01**	**2.16E–01**	8.19E–01	2.33E–01

$$\varphi_{2i}(X) = 0.5 + \sum_{j=i+1}^{D} \cos\left(\sum_{k=|2i-j-1|-1}^{j} x_k)\right), \quad i = 1, 2, \ldots, D-1$$
$$\varphi_{m+i}(X) = -\varphi_i(X). \tag{21}$$

By the suggestions of the literature [25], this paper considers two instances of the radar polyphase code design problem for $D = 19$ and 20. For all algorithms, the MAX_FEs is set to 5.00E+06 [25]. For the rest of the parameters, we keep the same values as in the FM synthesis problem. Table XIV presents the results of the six DE variants over the FM synthesizer problem. The results for jDE, NSDE, and DEGL/SAW are taken from [25, Tbl. XIV]. As seen, MGBDE achieves better results than the other five algorithms for $D = 19$. For $D = 20$, DEGL/SAW obtains the best result, and MGBDE is the second best.

This paper only presents a comparative study on two real-world optimization problems. More applications will be investigated in the future work [51], [54], [55].

VI. CONCLUSION REMARKS AND FUTURE WORK

DE is a population-based random optimization algorithm, which has shown better performance on many real-world and benchmark optimization problems. There are two important control parameters, F and CR, in the original DE algorithm and many DE variants. However, it is difficult to determine the optimal control parameters because they are problem dependent. Although a few improved DE variants have been proposed, they mainly focuses on adaptive or self-adaptive mechanisms. To minimize the effects of the control parameters, this paper presents two versions of improved BBDE variants, namely, GBDE and MGBDE, in which the parameter F is eliminated and CR is dynamically adjusted during the evolution. Experiments are conducted on 30 benchmark functions and two real-world problems. The experimental results can be summarized as follows.

1) Based on the conducted comparative studies on DE/rand/1, DE/best/1, GBDE, and MGBDE, our proposed MGBDE and GBDE perform better than the other two basic DE schemes. GBDE achieves better results than DE/rand/1 and DE/best/1 on high-dimensional multimodal problems, while it shows worse performance than DE/best/1 on unimodal and simple multimodal functions. To take advantage of DE/best/1, a hybrid algorithm, called MGBDE, is proposed by the hybridization of GBDE and DE/best/1. The results show that MGBDE achieves better performance than GBDE on many unimodal and multimodal functions.
2) Compared to other popular DE variants and similar bare-bones algorithms, both GBDE and MGBDE outperform other algorithms on the majority of multimodal functions, while they obtain worse performance on many unimodal and simple multimodal functions. It demonstrates that the proposed approaches are beneficial for solving multimodal problems. The possible reason is that the Gaussian sampling method can keep up the population diversity and improve the exploration.

GBDE is helpful to solve multimodal functions, while DE/best/1 is beneficial for solving unimodal functions. It is difficult to balance these two DE mutation strategies in MGBDE. In this paper, a simple alternative method is used to determine the mutation strategies. Other alternative methods will be tried in the future work.

REFERENCES

[1] R. Storn and K. Price, "Differential Evolution—A Simple and Efficient Adaptive Scheme for Global Optimization Over Continuous Spaces," Int. Comput. Sci. Inst., Berkeley, CA, TR-95-012, 1995.

[2] R. Storn and K. Price, "Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces," *J Global Optim.*, vol. 11, no. 4, pp. 341–359, Dec. 1997.

[3] R. Kennedy, S. D. Mjuller, and P. Koumoutsakos, "A parameter study for differential evolution," in *Proc. Advances Intell. Syst., Fuzzy Syst., Evol. Comput.*, Crete, Greece, 2002, pp. 293–298.

[4] J. Zhang and A. C. Sanderson, "An approximate Gaussian model of differential evolution with spherical fitness functions," in *Proc. IEEE Congr. Evol. Comput.*, Singapore, 2007, pp. 2220–2228.

[5] J. Liu and J. Lampinen, "A fuzzy adaptive differential evolution algorithm," *Soft Comput.–A Fusion Found., Methodol. Appl.*, vol. 9, no. 6, pp. 448–462, Jun. 2005.

[6] J. Brest, S. Greiner, B. Bošković, M. Mernik, and V. Žumer, "Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems," *IEEE Trans. Evol. Comput.*, vol. 10, no. 6, pp. 646–657, Dec. 2006.

[7] A. K. Qin and P. N. Suganthan, "Self-adaptive differential evolution algorithm for numerical optimization," in *Proc. IEEE Congr. Evol. Comput.*, 2005, vol. 2, pp. 1785–1791.

[8] A. K. Qin, V. L. Huang, and P. N. Suganthan, "Differential evolution algorithm with strategy adaption for global numerical optimization," *IEEE Trans. Evol. Comput.*, vol. 13, no. 2, pp. 398–417, Apr. 2009.

[9] J. Zhang and A. C. Sanderson, "JADE: Adaptive differential evolution with optional external archive," *IEEE Trans. Evol. Comput.*, vol. 13, no. 5, pp. 945–958, Oct. 2009.

[10] W. Gong, Z. Cai, C. X. Ling, and H. Li, "Enhanced differential evolution with adaptive strategies for numerical optimization," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 41, no. 2, pp. 397–413, Apr. 2011.

[11] J. Kennedy, "Bare bones particle swarms," in *Proc. IEEE Swarm Intell. Symp.*, 2003, pp. 80–87.

[12] M. G. H. Omran, A. Engelbrecht, and A. Salman, "Bare bones differential evolution," *Eur. J. Oper. Res.*, vol. 196, no. 1, pp. 128–139, Jul. 2009.

[13] X. Yao, Y. Liu, and G. Lin, "Evolutionary programming made faster," *IEEE Trans. Evol. Comput.*, vol. 3, no. 2, pp. 82–102, Jul. 1999.

[14] J. Brest, A. Zamuda, B. Bošković, M. S. Maučec, and V. Žumer, "High-dimensional real-parameter optimization using self-adaptive differential evolution algorithm with population size reduction," in *Proc. IEEE Congr. Evol. Comput.*, 2008, pp. 2032–2039.

[15] J. Brest and M. S. Maučec, "Self-adaptive differential evolution algorithm using population size reduction and three strategies," *Soft Comput.–A Fusion Found., Methodol. Appl.*, vol. 15, no. 11, pp. 2157–2174, Nov. 2011.

[16] Z. Yang, J. He, and X. Yao, "Making a difference to differential evolution," Advance in Metaheuristics for Hard Optimization, pp. 397–414, Berlin, Germany: Springer-Verlag 2008.

[17] Z. Yang, K. Tang, and X. Yao, "Self-adaptive differential evolution with neighborhood search," in *Proc. IEEE Congr. Evol. Comput.*, 2008, pp. 1110–1116.

[18] N. Noman and H. Iba, "Accelerating differential evolution using an adaptive local search," *IEEE Trans. Evol. Comput.*, vol. 12, no. 1, pp. 107–125, Feb. 2008.

[19] S. Rahnamayan, H. R. Tizhoosh, and M. M. A. Salama, "Opposition-based differential evolution," *IEEE Trans. Evol. Comput.*, vol. 12, no. 1, pp. 64–79, Feb. 2008.

[20] H. Wang, S. Rahnamayan, and S. Zeng, "Generalised opposition-based differential evolution: An experimental study," *Int. J. Comput. Appl. Technol.*, vol. 43, no. 4, pp. 311–319, Jan. 2012.

[21] H. Wang, Z. Wu, S. Rahnamayan, and L. Kang, "A scalability test for accelerated DE using generalized opposition-based learning," in *Proc. Intell. Syst. Des. Appl.*, 2009, pp. 1090–1095.

[22] H. Wang, Z. Wu, S. Rahnamayan, Y. Liu, and M. Ventresca, "Enhancing particle swarm optimization using generalized opposition-based learning," *Inf. Sci.*, vol. 181, no. 20, pp. 4699–4714, Oct. 2011.

[23] H. Wang, Z. Wu, and S. Rahnamayan, "Enhanced opposition-based differential evolution for solving high-dimensional continuous optimization problems," *Soft Comput.–A Fusion Found., Methodol. Appl.*, vol. 15, no. 11, pp. 2127–2140, Nov. 2011.

[24] H. Wang, S. Rahnamayan, and Z. Wu, "Parallel differential evolution with self-adapting control parameters and generalized opposition-based learning for solving high-dimensional optimization problems," *J. Parallel Distrib. Comput.*, 2012. doi:10.1016/j.jpdc.2012.02.019, to be published.

[25] S. Das, A. Konar, U. K. Chakraborty, and A. Abraham, "Differential evolution using a neighborhood based mutation operator," *IEEE Trans. Evol. Comput.*, vol. 13, no. 3, pp. 526–553, Jun. 2009.

[26] H. Wang, Z. Wu, and S. Rahnamayan, "Differential evolution enhanced by neighborhood search," in *Proc. IEEE Congr. Evol. Comput.*, 2010, pp. 1–8.

[27] H. Wang, Z. Wu, S. Rahnamayan, and D. Jiang, "Sequential differential evolution enhanced by neighborhood search for large scale global optimization," in *Proc. IEEE Congr. Evol. Comput.*, 2010, pp. 4056–4062.

[28] A. Ghosh, S. Das, A. Chowdhury, and R. Giri, "An improved differential evolution algorithm with fitness-based adaptation of the control parameters," *Inf. Sci.*, vol. 181, no. 18, pp. 3749–3765, Sep. 2011.

[29] S. M. Islam, S. Das, S. Ghosh, S. Roy, and P. N. Suganthan, "An adaptive differential evolution algorithm with novel mutation and crossover strategies for global numerical optimization," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 42, no. 2, pp. 482–500, Apr. 2012.

[30] P. Chakraborty, G. G. Roy, S. Das, D. Jain, and A. Abraham, "An improved harmony search algorithm with differential mutation operator," *Fundam. Inf.*, vol. 95, no. 4, pp. 401–426, Dec. 2009.

[31] R. Mallipeddi and P. N. Suganthan, "Differential evolution algorithm with ensemble of parameters and mutation and crossover strategies," in *Proc. Swarm Evol. Memetic Comput. Conf.*, 2010, vol. LNCS 6466, pp. 71–78.

[32] M. Weber, V. Tirronen, and F. Neri, "Scale factor inheritance mechanism in distributed differential evolution," *Soft Comput.–A Fusion Found., Methodol. Appl.*, vol. 14, no. 11, pp. 1187–1207, Sep. 2010.

[33] H. Wang, S. Rahnamayan, and Z. Wu, "Adaptive differential evolution with variable population size for solving high-dimensional problems," in *Proc. IEEE Congr. Evol. Comput.*, 2011, pp. 2626–2632.

[34] C. Lin, A. Qin, and Q. Feng, "A new differential mutation base generator for differential evolution," *J. Global Optim.*, vol. 49, no. 1, pp. 69–90, Jan. 2011.

[35] M. G. Epitropakis, D. K. Tasoulis, N. G. Pavlidis, V. P. Plagianakos, and M. N. Vrahatis, "Enhancing differential evolution utilizing proximity-based mutation operators," *IEEE Trans. Evol. Comput.*, vol. 15, no. 1, pp. 99–119, Feb. 2011.

[36] Y. Wang, Z. Cai, and Q. Zhang, "Differential evolution with composite trial vector generation strategies and control parameters," *IEEE Trans. Evol. Comput.*, vol. 15, no. 1, pp. 55–66, Feb. 2011.

[37] S. Ghosh, S. Das, A. V. Vasilakos, and K. Suresh, "On convergence of differential evolution over a class of continuous functions with unique global optimum," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 42, no. 1, pp. 107–124, Feb. 2012.

[38] S. Das and P. N. Suganthan, "Differential evolution: A survey of the state-of-the-art," *IEEE Trans. Evol. Comput.*, vol. 15, no. 1, pp. 4–31, Feb. 2011.

[39] F. Neri and V. Tirronen, "Recent advances in differential evolution: A survey and experimental analysis," *Artif. Intell. Rev.*, vol. 33, no. 1/2, pp. 61–106, Feb. 2010.

[40] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proc. IEEE Int. Conf. Neural Netw.*, 1995, pp. 1942–1948.

[41] M. Clerc and J. Kennedy, "The particle swarm-explosion, stability, and convergence in a multidimensional complex space," *IEEE Trans. Evol. Comput.*, vol. 6, no. 1, pp. 58–73, Feb. 2002.

[42] F. van den Bergh and A. Engelbrecht, "A study of particle swarm optimization particle trajectories," *Inf. Sci.*, vol. 176, no. 8, pp. 937–971, Apr. 2006.

[43] A. Engelbrecht, "Heterogeneous particle swarm optimization," in *Proc. Int. Conf. Swarm Intell.*, 2010, pp. 191–202.

[44] Y. W. Shang and Y. H. Qiu, "A note on the extended Rosenbrock function," *Evol. Comput.*, vol. 14, no. 1, pp. 119–126, 2006.

[45] S. García, A. Fernández, J. Luengo, and F. Herrera, "Advanced non-parametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power," *Inf. Sci.*, vol. 180, no. 10, pp. 2044–2064, May 2010.

[46] S. García, D. Molina, M. Lozano, and F. Herrera, "A study on the use of non-parametric tests for analyzing the evolutionary algorithms behavior: A case study on the CEC 2005 special session on real parameter optimization," *J. Heuristics*, vol. 15, no. 6, pp. 617–644, Dec. 2009.

[47] J. Derrac, S. García, D. Molina, and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms," *Swarm Evol. Comput.*, vol. 1, no. 1, pp. 3–18, Mar. 2011.
[48] A. Ghosh, A. Chowdhury, S. Sinha, A. V. Vasilakos, and S. Das, "A genetic Lbest particle swarm optimizer with dynamically varying subswarm topology," in *Proc. IEEE Congr. Evol. Comput.*, 2012, pp. 1–7.
[49] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y. P. Chen, A. Auger, and S. Tiwari, "Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real-Parameter Optimization," Nanyang Technol. Univ., Singapore, Tech. Rep., 2005.
[50] R. Mallipeddi, P. N. Suganthan, Q. K. Pan, and M. F. Tasgetiren, "Differential evolution algorithm with ensemble of parameters and mutation strategies," *Appl. Soft Comput.*, vol. 11, no. 2, pp. 1679–1696, Mar. 2011.
[51] S. Das and P. N. Suganthan, "Problem Definitions and Evaluation Criteria for CEC 2011 Competition on Testing Evolutionary Algorithms on Real World Optimization Problems," Nanyang Technol. Univ., Singapore, Tech. Rep., 2010.
[52] A. Horner, J. Beauchamp, and L. Haken, "Machine tongues XVI: Genetic algorithms and their application to FM matching synthesis," *Comput. Music J.*, vol. 17, no. 4, pp. 17–29, 1993.
[53] F. Herrera and M. Lozano, "Gradual distributed real-coded genetic algorithms," *IEEE Trans. Evol. Comput.*, vol. 4, no. 1, pp. 43–63, Apr. 2000.
[54] S. Sengupta, S. Das, M. Nasir, A. V. Vasilakos, and W. Pedrycz, "An evolutionary multi-objective sleep scheduling scheme for differentiated coverage in wireless sensor networks," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, 2012. doi:10.1109/TSMCC.2012.2196996, to be published.
[55] S. Sengupta, S. Das, M. Nasir, A. V. Vasilakos, and W. Pedrycz, "Energy-efficient differentiated coverage of dynamic objects using an improved evolutionary multi-objective optimization algorithm with fuzzy-dominance," in *Proc. IEEE Congr. Evol. Comput.*, 2012, pp. 1–8.

Hui Wang received the B.Sc. and M.Sc. degrees in computer science from the China University of Geosciences, Wuhan, China, in 2005 and 2008, respectively, and the Ph.D. degree in computer software and theory from Wuhan University, Wuhan, in 2011.

He is currently a Lecturer with the Nanchang Institute of Technology, Nanchang, China. He has published more than 30 international journal/conference papers. He serves as a reviewer for more than ten international journals. His research interest includes evolutionary optimization, large-scale global optimization, and parallel computing.

Shahryar Rahnamayan received the B.Sc. and M.Sc. degrees (both with honors) in software engineering from Shahid Beheshti University, Tehran, Iran, in 1998 and 2001, respectively. In 2007, he received the Ph.D. degree in the field of evolutionary computation from the University of Waterloo, Waterloo, Canada. The opposition-based differential evolution was proposed in his Ph.D. thesis.

Since August 2007, he has been a Chief Research Manager at OMISA Inc. (Omni-Modality Intelligent Segmentation Assistant), a company which develops innovative software for medical image segmentation. Before joining the Faculty of Engineering and Applied Science, University of Ontario Institute of Technology, Oshawa, Canada, in 2008 as a core faculty member, he was a postdoctoral fellow at Simon Fraser University, Vancouver, Canada. His research includes metaheuristics, evolutionary computation, large-scale optimization, image processing, and computer vision. He has received several prestigious awards such as the Ontario Graduate Scholarship, President's Graduate Scholarship, Natural Sciences and Engineering Research Council of Canada (NSERC)'s Japan Society for the Promotion of Science Fellowship, NSERC's Industrial R&D Fellowship, NSERC's Visiting Fellowship in Canadian Government Laboratories, and the Canadian Institute of Health Research Fellowship for two times. In 2009, he received the NSERC Discovery Grant and several other industrial grants. He is also a very active reviewer for more than 15 international journals, including the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, Elsevier Applied Soft Computing, Intelligent and Fuzzy Systems, Information Science and Engineering, IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS, Elsevier Information Sciences, Elsevier Artificial Intelligence in Medicine, ACM Transactions on Autonomous and Adaptive Systems, Springer-Verlag Neural Computing and Applications, and Elsevier Information Processing Letters. He has published more than 70 papers so far.

Hui Sun received the B.Sc. degree in mathematical mechanics from the Nanchang University, Nanchang, China in 1982, the M.Sc. degree in mechanics from the Tsinghua University, Beijing, China in 1988, and the Ph.D. degree in metal plastic working from the Nanchang University in 2002.

He is currently a Professor with Nanchang Institute of Technology, Nanchang, China. His research interests include intelligent algorithms, rough sets and granular computing, variational inequality principle, and variational inequalities.

Mahamed G. H. Omran received the B.Sc. and M.Sc. degrees (with honors) in computer engineering from Kuwait University, Safat, Kuwait, in 1998 and 2000, respectively. He completed his Ph.D. degree in the Department of Computer Science, University of Pretoria, Pretoria, South Africa, in April 2005.

His current research interest is in the area of metaheuristics and operational research. In addition, he is interested in the area of data clustering, unsupervised image classification, and color image quantization. He has more than 40 publications in these fields.