Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2015, Article ID 769245, 16 pages
http://dx.doi.org/10.1155/2015/769245

Research Article

A Hybrid Backtracking Search Optimization Algorithm with Differential Evolution

Lijin Wang,[1,2] Yiwen Zhong,[1] Yilong Yin,[2,3] Wenting Zhao,[2] Binqing Wang,[2] and Yulong Xu[2]

[1] *College of Computer and Information Science, Fujian Agriculture and Forestry University, Fuzhou 350002, China*
[2] *School of Computer Science and Technology, Shandong University, Jinan 250101, China*
[3] *School of Computer Science and Technology, Shandong University of Finance and Economics, Jinan 250014, China*

Correspondence should be addressed to Lijin Wang; lijinwang@fafu.edu.cn and Yilong Yin; ylyin@sdu.edu.cn

Received 14 November 2014; Revised 10 February 2015; Accepted 19 February 2015

Academic Editor: Antonio Ruiz-Cortes

The backtracking search optimization algorithm (BSA) is a new nature-inspired method which possesses a memory to take advantage of experiences gained from previous generation to guide the population to the global optimum. BSA is capable of solving multimodal problems, but it slowly converges and poorly exploits solution. The differential evolution (DE) algorithm is a robust evolutionary algorithm and has a fast convergence speed in the case of exploitive mutation strategies that utilize the information of the best solution found so far. In this paper, we propose a hybrid backtracking search optimization algorithm with differential evolution, called HBD. In HBD, DE with exploitive strategy is used to accelerate the convergence by optimizing one worse individual according to its probability at each iteration process. A suit of 28 benchmark functions are employed to verify the performance of HBD, and the results show the improvement in effectiveness and efficiency of hybridization of BSA and DE.

1. Introduction

Optimization plays an important role in many fields, for example, decision science and physical system, and can be abstracted as the minimization or maximization of objective functions subject to constraints on their variables mathematically. Generally speaking, the optimization algorithms can be employed to find their solutions. The stochastic relaxation optimization algorithms, such as genetic algorithm (GA) [1], particle swarm optimization algorithm (PSO) [2, 3], ant colony algorithm (ACO) [4], and differential evolution (DE) [5], are one of the methods for solving solutions effectively and almost nature-inspired optimization techniques. For instance, DE, one of the most powerful stochastic optimization methods, employs the mutation, crossover, and selection operators at each generation to drive the population to global optimum. In DE, the mutation operator is one of core components and includes many differential mutation strategies which reveal different characteristics. For example, the strategies, which utilize the information of the best solution found so far, have fast convergence speed and favor exploitation. These strategies are classified as the exploitative strategies [6].

Inspired by the success of GA, PSO, ACO, and DE for solving optimization problems, new nature-inspired algorithms have been a hot topic in the development of the stochastic relaxation optimization techniques, such as artificial bee colony [7], cuckoo search [8], bat algorithm [9], firefly algorithm [10], social emotional optimization [11–13], harmony search [14], and biogeography based optimization [15]. A survey has pointed out that there are about 40 different nature-inspired algorithms [16].

The backtracking search optimization algorithm (BSA) [17] is a new stochastic method for solving real-valued numerical optimization problems. Similar to other evolutionary algorithms, BSA uses the mutation, crossover, and selection operators to generate trial solutions. When generating trial solutions, BSA employs a memory to store experiences gained from previous generation solutions. Taking advantage of historical information to guide the population to global optimum, BSA focuses on exploration and is capable of

```
Input: Mutant, N and D
output: T
Step 1.
Initiate map_{1:N,1:D} = 1
 Generate a, b drawn from uniformly distribution with the range between 0 and 1.
If a > b then
    For i from 1 to N do
        Generate a vector containing a random permutation of the integers U from 1 to D
        Generate c drawn from uniformly distribution with the range between 0 and 1.
        map_{i,U(1:c·D)} = 0
    End For
Else
     For i from 1 to N do
        Generate a random integer k from 1 to D
        map_{i,k} = 0
    End For
End If
Step 2.
For i from 1 to N do
    For j from 1 to D do
        If map_{i,j} = 1 then
            T_{i,j} = P_{i,j}
        Else
            T_{i,j} = Mutant_{i,j}
        End If
    End For
End For
```

Algorithm 1: Crossover operator.

solving multimodal optimization problems. However, utilizing experiences may make BSA converge slowly and prejudice exploitation on later iteration stage.

On the other hand, researches have paid more and more attention to combine different search optimization algorithms or machine learning methods to improve the performance for real-world optimization problems. Some good surveys about hybrid metaheuristics or machine learning methods can be found in the literatures [18–20]. In this paper, we also concentrate on a hybrid metaheuristic algorithm, called HBD, which combines BSA and DE. HBD employs DE with exploitative mutation strategy to improve convergence speed and to favor exploitation. Furthermore, in HBD, DE is invoked to optimize only one worse individual selected with the help of its probability at each iteration process. We use 28 benchmark functions to verify the performance of HBD, and the results show the improvement in effectiveness and efficiency of hybridization of BSA and DE. The major advantages of our approach are as follows. (i) DE with exploitive strategies helps HBD converge fast and favor exploitation. (ii) Since DE optimizes one individual, HBD expends only one more function evaluation at each iteration and will not increase the overall complexity of BSA. (iii) DE is embedded behind BSA, and therefore HBD does not destroy the structure of BSA, and it is still very simple.

The remainder of this paper is organized as follows. Section 2 describes BSA and DE. Section 3 presents the HBD algorithm. Section 4 reports the experimental results. Section 5 concludes this paper.

2. Preliminary

2.1. BSA. The backtracking search optimization algorithm is a new stochastic search technique developed recently [17]. BSA has a single control parameter and a simple structure that is effective and capable of solving different optimization problems. Furthermore, BSA is a population-based method and possesses a memory in which it stores a population from a randomly chosen previous generation for generating the search-direction matrix. In addition, BSA is a nature-inspired method employing three basic genetic operators: mutation, crossover, and selection.

BSA employs a random mutation strategy that used only one direction individual for each target individual, formulated as follows:

$$\text{Mutant} = P + F \oplus (\text{old}P - P), \tag{1}$$

where P is the current population, old$P$ is the historical population, and F is a coefficient which controls the amplitude of the search-direction matrix (old$P - P$).

BSA also uses a nonuniform and more complex crossover strategy. There are two steps in the crossover process. Firstly, a binary integer-values matrix (*map*) of size $N \times D$ (N and D are the population size and the problem dimensions) is generated to indicate the mutant individual to be manipulated by using the relevant individual. Secondly, the relevant dimensions of mutant individual are updated by using the relevant individual. This crossover process can be summarized as shown in Algorithm 1.

```
Initiate the population P and the historical population oldP randomly sampled from search space.
While (Stop Condition doesn't meet)
    Perform the first type selection: oldP = P in the case of a < b, where a and b are drawn from uniformly distribution with the
    range between 0 and 1.
    Permute arbitrary changes in position of oldP.
    Generate the mutant according to (1).
    Generate the population T based on Algorithm 1.
    Perform the second type selection: select the population with better fitness from T and P.
    Update the best solution.
End While
Output the best solution.
```

Algorithm 2: BSA.

BSA has two types of selection operators. The first type selection operator is employed to select the historical population for calculating search direction. The rule is that the historical population should be replaced with the current population when the random number is smaller than the other one. The second type of selection operator is greedy to determine the better individuals to go into the next generation.

According to the above descriptions, the pseudocode of BSA is summarized as shown in Algorithm 2.

2.2. DE. DE is a powerful evolutionary algorithm for global optimization over continuous space. When being used to solve optimization problems, it evolves a population of N candidate solutions with D-dimensional parameter vectors, noted as X. In DE, the population is initiated by uniform sampling within the prescribed minimum and maximum bounds.

After initialization, DE steps into the iteration process where the evolutionary operators, namely, mutation, crossover, and selection, are invoked in turn, respectively.

DE employs the mutation strategy to generate a mutant vector V. So far, there are several mutant strategies, and the most well-known and widely used strategies are listed as follows [21, 22]:

"DE/best/1":

$$V_i = X_{\text{best}} + F\left(X_{r_1} - X_{r_2}\right), \quad (2)$$

"DE/current-to-best/1":

$$V_i = X_i + F\left(X_{\text{best}} - X_i\right) + F\left(X_{r_1} - X_{r_2}\right), \quad (3)$$

"DE/best/2":

$$V_i = X_{\text{best}} + F\left(X_{r_1} - X_{r_2}\right) + F\left(X_{r_3} - X_{r_4}\right), \quad (4)$$

"DE/rand/1":

$$V_i = X_{r_1} + F\left(X_{r_2} - X_{r_3}\right), \quad (5)$$

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

$$V_i = X_i + F\left(X_{r_1} - X_i\right) + F\left(X_{r_2} - X_{r_3}\right), \quad (6)$$

"DE/rand/2":

$$V_i = X_{r_1} + F\left(X_{r_2} - X_{r_3}\right) + F\left(X_{r_4} - X_{r_5}\right), \quad (7)$$

where the indices r_1, r_2, r_3, r_4, and r_5 are uniformly random mutually different integers from 1 to N, X_{best} denotes the best individual obtained so far, and X_i and V_i are the ith vector of X and V, respectively.

The crossover operator is performed to generate a trial vector U_i according to each pair of X_i and V_i after the mutant vector V_i is generated. The most popular strategy is the binomial crossover described as follows:

$$u_{i,j} = \begin{cases} v_{i,j} & \text{if rand}\,(01) \le C_r \text{ OR } j = j_{\text{rand}} \\ x_{i,j} & \text{otherwise,} \end{cases} \quad (8)$$

where C_r is called the crossover rate, j_{rand} is randomly sampled from 1 to D, and $u_{i,j}$, $v_{i,j}$, and $x_{i,j}$ are the jth element of U_i, V_i, and X_i, respectively.

Finally, DE uses a greedy mechanism to select the better vector from each pair of X_i and U_i. This can be described as follows:

$$X_i = \begin{cases} U_i & \text{if fitness}\left(U_i\right) \le \text{fitness}\left(X_i\right) \\ X_i & \text{otherwise.} \end{cases} \quad (9)$$

3. HBD

In this section, we describe the HBD algorithm in detail. First, the motivations of this paper are given. Second, the framework of HBD is shown.

3.1. Motivations. BSA uses an external archive to store experiences gained from previous generation solutions and makes use of them to guide the population to global optimum. According to BSA, permuting arbitrary changes in position of historical population makes the individuals be chosen randomly in the mutation operator; therefore, the algorithm focuses on exploration and is capable of solving multimodal optimization problems. However, just due to random selection, by utilizing experiences, BSA may be led to converge slowly and to prejudice exploitation on later iteration stage. This motivates our approach which aims to accelerate the convergence speed and to enhance the exploitation of the search space to keep the balance between the exploration and exploitation capabilities of BSA.

On the other hand, some studies have investigated the exploration and exploitation ability of different DE mutation strategies and pointed out the mutation operators that incorporate the best individual (e.g., (2), (3), and (4)) favor exploitation because the mutant individuals are strongly attracted around the best individual [6, 23]. This motivates us to hybridize these exploitative mutation strategies to enhance the exploitation capability of BSA. In addition, this paper is also in light of some studies which have shown that it is an effective way to combine other optimization methods to improve the performance for real-world optimization problems [24–27].

3.2. Framework of HBD. Generally speaking, there are many ways to hybridize BSA with DE. In this study, we propose another hybrid schema between BSA with DE. In this schema, HBD employs DE with exploitive strategy behind BSA at each iteration process to share the information between BSA and DE. However, more individuals are optimized by DE, and more function evaluations will be spent. In this case, HBD would gain the premature convergence, resulting in prejudicing exploration. Thus, to keep the exploration capability of HBD, DE is used to optimize only one worse individual according to its probability. In addition, (2) is used as default mutation strategy in HBD because (3) and (4) have stronger exploration capabilities by introducing more perturbation with the random individual [6] or a modification combining "DE/best/1" and "DE/rand/1" [28]. The performance influenced by different exploitative strategies will be discussed in Section 4.3.

In order to select one individual for DE, in this work, we assign a probability model for each individual according to its fitness. It can be formulated as follows:

$$p_i = \frac{r_i}{N}, \tag{10}$$

where N is the population size and r_i is the ranking value of each individual when the population is sorted from the worst fitness to the best one.

Note that the probability equation is similar to the selection probability in DE with ranking-based mutation operators [29]. In general, the worse individuals are more far away from the best individual than the better ones; thus, they will have higher probabilities to get around the best one. This selection strategy can be defined as follows:

$$I_s = i \quad \text{if rand}(0,1) > p_i \quad i = 1, \ldots, N, \tag{11}$$

where I_s is selected individual and optimized by DE.

It is worth pointing out that our previous work [30], called BSADE, splits the whole iteration process into two parts: the previous two-third and the latter one-third stages. BSA is used in the first stage, and DE is employed in the second stage. In this case, DE does not share the population information with BSA. Moreover, it is difficult to split the whole iteration process into two parts. Thus, the difference between HBD and BSADE is that HBD shares the population information between BSA and DE, while BSADE does not. The comparison can be found in Section 4.4.

According to the above descriptions, the pseudocode of HBD is described in Algorithm 3.

4. Experimental Verifications

In this section, to verify the performance of HBD, we carry out comprehensive experimental tests on a suit of 28 benchmark functions proposed in the CEC-2013 competition [31]. These 28 benchmark functions include 5 unimodal functions F_1–F_5, 15 basic multimodal functions F_6–F_{20}, and 8 composition functions F_{21}–F_{28}. More details about 28 functions can be found in [31].

To make a fair comparison, we use the same parameters for BSA and HBD, unless a change is mentioned. Each algorithm is performed 25 times for each function with the dimensions $D = 10$, 30, and 50, respectively. The population size of each algorithm N is D when $D = 30$ and $D = 50$, while it is 30 in the case of $D = 10$. The maximum function evaluations are $10000 \times D$. The mutation factor F and the crossover factor C_r are 0.8 and 0.9 for HBD, respectively. In addition, we use the boundary handling method given in [17].

To evaluate the performance of algorithms, we use *Error* as an evaluation indicator first. *Error*, which is the function error value for the solution X obtained by the algorithms, is defined as $f(X) - f(X^*)$, where X^* is the global optimum of function. In addition, the average and standard deviation of the best error values, presented as "$\text{AVG}_{\text{Er}} \pm \text{STD}_{\text{Er}}$," are used in the different tables. Second, the convergence graphs are employed to show the mean error values of the best solutions at iteration process over the total run. Third, a Wilcoxon signed-rank test at the 5% significance level ($\alpha = 0.05$) is used to show the significant differences between two algorithms. The "+" symbol shows that the null hypothesis is rejected at the 5% significant level and HBD outperforms BSA, the "−" symbol says that the null hypothesis is rejected at the 5% significant level and BSA exceeds HBD, and the "=" symbol reveals that the null hypothesis is accepted at the 5% significant level and HBD ties BSA. Additionally, we also give the total number of statistical significant cases at the bottom of each table.

4.1. The Effect of HBD. To show the effect of the proposed algorithm, Table 1 lists the average error values obtained by BSA and HBD for 30-dimentional benchmark functions. For unimodal functions F_1–F_5, HBD overall obtains better average error values than BSA does. For instance, HBD gains the global optimum on F_5 and brings solutions with high quality to F_2–F_4 in terms of average error values. HBD exhibits a little inferiority to BSA for F_1, but these two approaches are not significant. For 15 basic multimodal functions F_6–F_{20}, with the help of average error values, HBD brings superior solutions to 10 out of 15 functions, equal ones to 2 out of 15 functions, and inferior ones to 3 out of 15 functions. However, according to the results of Wilcoxon test,

Initiate the population P and the historical population $oldP$ randomly sampled from search space.
While (Stop Condition doesn't meet)
  Perform the first type selection: $oldP = P$ in the case of $a < b$, *where* a and b are drawn from uniformly distribution with the range between 0 and 1.
  Permute arbitrary changes in position of $oldP$.
  Generate the ***mutant*** according to (1).
  Generate the population **T** based on Algorithm 1.
  Perform the second type selection: select the population with better fitness from **T** and **P**.
  Update the best solution.
  //Invoke DE with exploitive strategy
  Select One Individual according to its probability: I_s.
  Optimize I with the help of DE, and get I_{DE}
  If (fitness(I_{DE}) <= fitness(I_s))
    $I_s = I_{\text{DE}}$
  End If
  Update the best solution.
End While
Output the best solution.

Algorithm 3: HBD.

Table 1: Error values obtained by BSA and HBD for 30-dimensional CEC-2013 benchmark functions.

	BSA	HBD		
	$\text{AVG}_{\text{Er}} \pm \text{STD}_{\text{Er}}$	$\text{AVG}_{\text{Er}} \pm \text{STD}_{\text{Er}}$		P value
F_1	$\boldsymbol{4.17E-30 \pm 1.40E-29}$	$1.36E-29 \pm 4.17E-29$	=	0.359375
F_2	$1.22E+06 \pm 5.54E+05$	$\boldsymbol{3.15E+05 \pm 1.52E+05}$	+	0.000014
F_3	$7.52E+06 \pm 8.54E+06$	$\boldsymbol{4.38E+06 \pm 8.11E+06}$	+	0.045010
F_4	$1.25E+04 \pm 3.47E+03$	$\boldsymbol{5.05E+03 \pm 2.23E+03}$	+	0.000016
F_5	$0.00E+00 \pm 0.00E+00$	$0.00E+00 \pm 0.00E+00$	=	1.000000
F_6	$3.04E+01 \pm 2.54E+01$	$\boldsymbol{1.12E+01 \pm 1.47E+01}$	+	0.001721
F_7	$7.39E+01 \pm 1.03E+01$	$\boldsymbol{5.03E+01 \pm 1.61E+01}$	+	0.000101
F_8	$\boldsymbol{2.09E+01 \pm 5.95E-02}$	$2.10E+01 \pm 3.17E-02$	=	0.142532
F_9	$2.70E+01 \pm 2.29E+00$	$\boldsymbol{2.12E+01 \pm 4.26E+00}$	+	0.000081
F_{10}	$1.78E-01 \pm 1.34E-01$	$\boldsymbol{9.15E-02 \pm 5.25E-02}$	+	0.009417
F_{11}	$\boldsymbol{7.96E-02 \pm 2.75E-01}$	$3.58E-01 \pm 6.34E-01$	=	0.062500
F_{12}	$8.41E+01 \pm 1.51E+01$	$\boldsymbol{8.09E+01 \pm 1.53E+01}$	=	0.396679
F_{13}	$1.44E+02 \pm 2.37E+01$	$\boldsymbol{1.29E+02 \pm 2.90E+01}$	=	0.061480
F_{14}	$3.70E+00 \pm 1.68E+00$	$\boldsymbol{2.83E+00 \pm 1.82E+00}$	+	0.034670
F_{15}	$3.73E+03 \pm 4.27E+02$	$\boldsymbol{3.50E+03 \pm 4.82E+02}$	=	0.097970
F_{16}	$1.31E+00 \pm 2.14E-01$	$1.31E+00 \pm 2.38E-01$	=	0.903627
F_{17}	$3.09E+01 \pm 1.97E-01$	$3.09E+01 \pm 1.79E-01$	=	0.840072
F_{18}	$1.20E+02 \pm 1.80E+01$	$\boldsymbol{9.46E+01 \pm 2.11E+01}$	+	0.000980
F_{19}	$\boldsymbol{1.16E+00 \pm 1.79E-01}$	$1.23E+00 \pm 2.28E-01$	=	0.312970
F_{20}	$1.15E+01 \pm 4.55E-01$	$\boldsymbol{1.11E+01 \pm 5.89E-01}$	=	0.057836
F_{21}	$\boldsymbol{2.78E+02 \pm 6.63E+01}$	$2.95E+02 \pm 7.95E+01$	=	0.431762
F_{22}	$\boldsymbol{4.45E+01 \pm 1.91E+01}$	$4.48E+01 \pm 1.35E+01$	=	0.443172
F_{23}	$4.46E+03 \pm 5.40E+02$	$\boldsymbol{4.16E+03 \pm 5.07E+02}$	=	0.103553
F_{24}	$2.32E+02 \pm 1.17E+01$	$\boldsymbol{2.28E+02 \pm 8.92E+00}$	=	0.157770
F_{25}	$2.87E+02 \pm 1.41E+01$	$\boldsymbol{2.80E+02 \pm 8.80E+00}$	=	0.087527
F_{26}	$2.00E+02 \pm 2.17E-02$	$2.00E+02 \pm 6.81E-03$	+	0.000020
F_{27}	$8.80E+02 \pm 1.49E+02$	$\boldsymbol{7.52E+02 \pm 1.33E+02}$	+	0.003822
F_{28}	$3.00E+02 \pm 1.65E-13$	$3.00E+02 \pm 1.32E-13$	+	0.016377
+/=/−				12/16/0

they are not significant for HBD and BSA for 3 functions in which HBD gains lower solution quality. For composition functions F_{21}–F_{28}, HBD and BSA draw a tie on F_{26} and F_{28} by the aid of average error values; however, HBD significantly outperforms BSA according to the results of Wilcoxon test. Moreover, according to average error values, HBD performs better than BSA in F_{23}, F_{24}, F_{25}, and F_{27} but worse than BSA in F_{21} and F_{22}. Nevertheless, two algorithms almost are not significant for these 8 composition functions in terms of the results of Wilcoxon test. Summarily, according to "+/=/−," HBD wins and ties BSA on 12 and 16 out of 28 benchmark functions, respectively.

In order to further show the convergence speed of HBD, the convergence curves of two algorithms for six selected benchmark functions are given in Figure 1.

It is observed that the selected functions can be divided into four groups, and overall the convergence performance of HBD is better than BSA. For example, for the first group of functions, for example, F_6 and F_{27} in which HBD has significantly better average error values than BSA, HBD converges faster than BSA in terms of the convergence curves seen in Figures 1(c) and 1(f). For F_{20} and F_{23} belong to the second group where HBD cannot bring the solutions with higher quality significantly, HBD still converges faster than BSA does. Third, for F_5 in which both of the two algorithms reach the global optimum, convergence performance of HBD is better compared to BSA. Additionally, HBD outperforms BSA according to the convergence curves seen in Figure 1(a), although the average error values optimized by HBD are inferior but not significant to BSA.

All in all, HBD overall outperforms BSA in terms of solution quality and convergence speed. This is because DE with exploitive mutation strategy enhances the exploitation capability of HBD, and it does not expend too much function evaluations.

4.2. Scalability of HBD. In this section, to analyze the performance of HBD affected by the problem dimensionality, a scalability study is investigated, respectively, on the 28 functions with 10-D and 50-D due to their definition up to 50-D [31]. The results are tabulated in Table 2.

In the case of $D = 10$, according to average error values shown in Table 2, HBD exhibits superiority in the majority of functions while inferiority in a handful of ones. Additionally, in terms of the total of "+/=/−," HBD wins and ties BSA in 9 and 19 out of 28 functions, respectively.

When $D = 50$, HBD still can bring solutions with higher quality than BSA does in most of benchmark functions. Moreover, HBD outperforms and ties BSA in 13 and 15 out of 28 functions, respectively.

In summary, it suggests that the advantage of HBD over BSA is stable when the dimensionality of problems increases.

4.3. The Effect of Mutation Strategy. In HBD, the "DE/best/1" mutation strategy is used to enhance the exploitation capability of HBD in default. To show the performance of HBD influenced by other exploitive mutation strategies, the experiments are carried on benchmark functions and the results are listed in Table 3 where cHBD and bHBD mean that HBD uses "DE/current-to-best/1" and "DE/best/2," respectively. The results obtained by cHBD and bHBD, which are highly accurate compared to those obtained by HBD, are marked in bold.

From Table 3, in terms of the average error values, bHBD shows the higher accuracy compared to HBD for a few functions since "DE/best/2" usually exhibits better exploration than "DE/best/1" because of one more difference of randomly selected individuals in the former [23]. cHBD also gains higher accuracy of solutions than HBD does for a handful of functions because "DE/current-best/1," a modification combining "DE/best/1" and "DE/rand/1" [28], shows better exploration than "DE/best/1." In other words, for a few functions, "DE/best/2" and "DE/current-best/1" can balance the exploration and exploitation capabilities of HBD better. For example, bHBD and cHBD bring the solutions with higher quality to F_1, F_3, F_8, F_{10}, F_{11}, and F_{16}; in particular, they reach the global optimum. However, for most of the functions, HBD with "DE/best/1" performs better than cHBD and bHBD.

Additionally, Table 4 reports the results of the multiple-problem Wilcoxon test which was done similarly in [29, 32] between HBD and its variants for all functions. We can see from Table 4 that HBD is significantly better than bHBD and HBD gets higher R^+ value than R^- value although two values are not significant. Therefore, HBD uses "DE/best/1" in the tradeoff.

4.4. The Effect of Hybrid Schema. In this section, we analyze the performance of HBD affected by the hybrid schema. Firstly, to show the effect of more than one individual optimized by DE, the algorithm, called aHBD which uses DE to optimize the whole population, is used to compare with HBD. Secondly, we add a probability P_c on aHBD to control the use of DE and propose paHBD. In paHBD, if the random number r drawn from uniform distribution between 0 and 1 is less than P_c, then DE is invoked. The P_c is defined as follows:

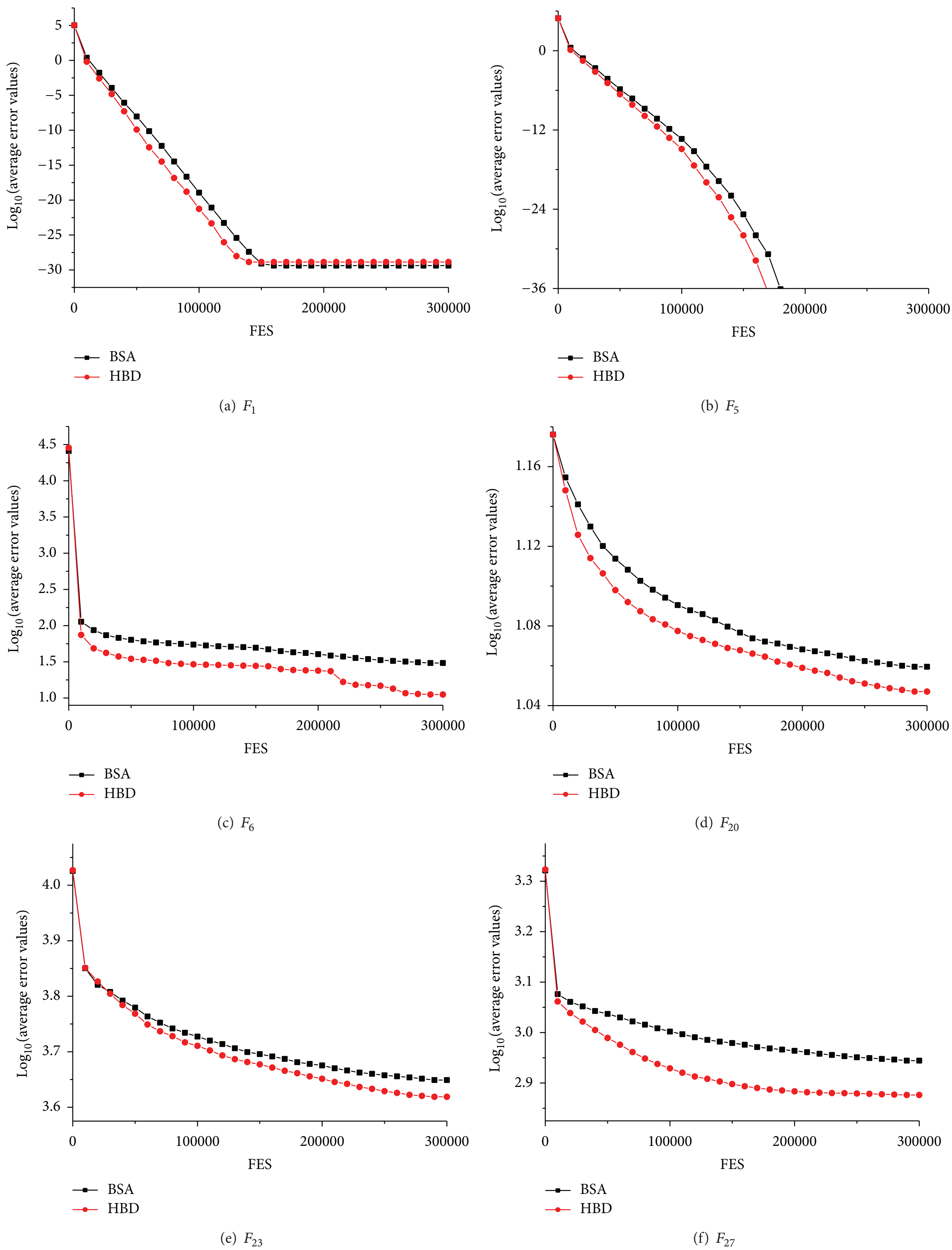
$$P_c = \frac{fes}{mfes}, \tag{12}$$

where fes is the number of function evaluations which had been spent and $mfes$ is the maximum number of function evaluations. Additionally, BSADE is compared with HBD to show their differences.

Table 5 lists the error values obtained by aHBD, paHBD, BSADE, and HBD for 28 functions at $D = 30$. It can be observed that HBD wins, ties, and loses aHBD in 10, 12, and 6 out of 28 functions in terms of "+/=/−," respectively. It says that optimizing more individuals using DE costs more function evaluations when DE is embedded behind BSA directly, resulting in reducing the iteration process cycles and then getting poor performance for most functions. Regarding BSADE, since BSA and DE are invoked in different stages where they cannot exchange the population information, it is clear that this schema cannot balance the exploitation and exploration well. Thus, compared with BSADE, HBD brings solutions with higher accuracy for most functions. Moreover, HBD wins, ties, and loses BSADE in 8, 17, and 3 out of 28

Table 2: Error values obtained by BSA and HBD for 10- and 50-dimensional CEC-2013 benchmark functions.

	$D = 10$ BSA $\text{AVG}_{\text{Er}} \pm \text{STD}_{\text{Er}}$	HBD $\text{AVG}_{\text{Er}} \pm \text{STD}_{\text{Er}}$		P value	$D = 50$ BSA $\text{AVG}_{\text{Er}} \pm \text{STD}_{\text{Er}}$	HBD $\text{AVG}_{\text{Er}} \pm \text{STD}_{\text{Er}}$		P value
F_1	0.00E + 00 ± 0.00E + 00	0.00E + 00 ± 0.00E + 00	=	1.000000	1.69E − 29 ± 3.04E − 29	**1.48E − 29 ± 4.26E − 29**	=	0.582031
F_2	8.59E + 04 ± 2.30E + 05	**4.75E + 03 ± 5.65E + 03**	+	0.000065	2.77E + 06 ± 8.09E + 05	**1.11E + 06 ± 4.38E + 05**	+	0.000014
F_3	5.46E + 04 ± 1.36E + 05	**1.44E + 02 ± 6.95E + 02**	+	0.000012	4.95E + 07 ± 4.00E + 07	**1.25E + 07 ± 1.20E + 07**	+	0.000140
F_4	5.04E + 02 ± 3.66E + 02	**5.67E + 01 ± 1.10E + 02**	+	0.000014	3.22E + 04 ± 5.16E + 03	**2.04E + 04 ± 4.88E + 03**	+	0.000023
F_5	0.00E + 00 ± 0.00E + 00	0.00E + 00 ± 0.00E + 00	=	1.000000	3.06E − 38 ± 1.53E − 37	**0.00E + 00 ± 0.00E + 00**	=	0.250000
F_6	**1.60E + 00 ± 3.25E + 00**	2.75E + 00 ± 4.50E + 00	=	0.220852	5.42E + 01 ± 1.87E + 01	**4.42E + 01 ± 7.78E − 01**	+	0.000046
F_7	6.96E + 00 ± 6.85E + 00	**1.73E + 00 ± 2.49E + 00**	+	0.045010	8.48E + 01 ± 1.02E + 01	**8.38E + 01 ± 1.06E + 01**	=	0.756995
F_8	2.04E + 01 ± 7.79E − 02	2.04E + 01 ± 5.55E − 02	=	0.989266	2.11E + 01 ± 3.33E − 02	2.11E + 01 ± 2.32E − 02	=	0.562928
F_9	3.54E + 00 ± 8.43E − 01	**2.47E + 00 ± 1.29E + 00**	+	0.006313	**5.44E + 01 ± 3.73E + 00**	5.50E + 01 ± 3.85E + 00	=	0.657069
F_{10}	8.49E − 02 ± 3.51E − 02	**7.75E − 02 ± 5.50E − 02**	=	0.989266	3.99E − 01 ± 2.11E − 01	**1.14E − 01 ± 6.51E − 02**	+	0.000018
F_{11}	0.00E + 00 ± 0.00E + 00	0.00E + 00 ± 0.00E + 00	=	1.000000	7.96E − 02 ± 2.75E − 01	**0.00E + 00 ± 0.00E + 00**	=	0.062500
F_{12}	1.10E + 01 ± 3.31E + 00	**1.09E + 01 ± 3.48E + 00**	=	0.599802	1.92E + 02 ± 2.87E + 01	**1.68E + 02 ± 2.65E + 01**	+	0.007423
F_{13}	1.68E + 01 ± 7.18E + 00	**1.60E + 01 ± 7.27E + 00**	=	0.736617	3.25E + 02 ± 3.81E + 01	**2.88E + 02 ± 4.17E + 01**	+	0.001569
F_{14}	1.31E − 01 ± 5.44E − 02	**1.28E − 01 ± 8.94E − 02**	+	0.000296	2.17E + 01 ± 5.08E + 00	**2.12E + 01 ± 4.56E + 00**	=	0.287862
F_{15}	6.10E + 02 ± 1.78E + 02	**4.67E + 02 ± 1.74E + 02**	+	0.022988	8.25E + 03 ± 4.91E + 02	**8.13E + 03 ± 4.40E + 02**	=	0.142532
F_{16}	6.54E − 01 ± 1.92E − 01	**6.23E − 01 ± 2.50E − 01**	=	0.989266	1.88E + 00 ± 2.67E − 01	**1.82E + 00 ± 2.86E − 01**	=	0.427339
F_{17}	**7.77E + 00 ± 2.88E + 00**	8.37E + 00 ± 2.87E + 00	=	0.427339	**5.44E + 01 ± 6.11E − 01**	5.45E + 01 ± 6.85E − 01	=	0.924971
F_{18}	2.67E + 01 ± 4.85E + 00	**2.05E + 01 ± 4.06E + 00**	+	0.000665	2.65E + 02 ± 2.44E + 01	**2.45E + 02 ± 3.23E + 01**	+	0.039554
F_{19}	2.45E − 01 ± 9.51E − 02	**2.15E − 01 ± 1.19E − 01**	=	0.657069	**2.67E + 00 ± 3.30E − 01**	2.80E + 00 ± 2.59E − 01	=	0.312970
F_{20}	2.92E + 00 ± 3.60E − 01	**2.57E + 00 ± 4.67E − 01**	+	0.019941	2.09E + 01 ± 7.34E − 01	**2.08E + 01 ± 4.44E − 01**	=	0.819095
F_{21}	3.16E + 02 ± 1.32E + 02	**3.04E + 02 ± 1.21E + 02**	=	0.444824	**6.94E + 02 ± 4.53E + 02**	8.72E + 02 ± 3.27E + 02	=	0.675764
F_{22}	1.49E + 01 ± 4.29E + 00	**1.42E + 01 ± 4.11E + 00**	=	0.544910	**5.96E + 01 ± 1.39E + 01**	6.10E + 01 ± 9.87E + 00	=	0.544910
F_{23}	8.60E + 02 ± 1.53E + 02	**8.06E + 02 ± 2.04E + 02**	=	0.818641	9.53E + 03 ± 6.62E + 02	**9.41E + 03 ± 7.87E + 02**	=	0.599802
F_{24}	**1.50E + 02 ± 3.69E + 01**	1.56E + 02 ± 4.44E + 01	=	0.716423	2.70E + 02 ± 1.04E + 01	**2.58E + 02 ± 1.09E + 01**	+	0.001569
F_{25}	1.93E + 02 ± 2.44E + 01	**1.84E + 02 ± 3.39E + 01**	=	0.241820	3.79E + 02 ± 1.40E + 01	**3.65E + 02 ± 2.10E + 01**	+	0.013817
F_{26}	1.13E + 02 ± 5.23E + 00	**1.11E + 02 ± 1.87E + 01**	=	0.109386	2.00E + 02 ± 7.33E − 02	2.00E + 02 ± 4.44E − 02	+	0.000012
F_{27}	**3.13E + 02 ± 2.39E + 01**	3.32E + 02 ± 4.73E + 01	=	0.676637	1.53E + 03 ± 1.89E + 02	**1.39E + 03 ± 2.01E + 02**	+	0.028314
F_{28}	2.20E + 02 ± 1.00E + 02	**2.04E + 02 ± 1.02E + 02**	=	0.802856	4.00E + 02 ± 1.33E − 13	4.00E + 02 ± 1.43E − 13	+	0.000001
+/=/−				9/19/0				13/15/0



(a) F_1

(b) F_5

(c) F_6

(d) F_{20}

(e) F_{23}

(f) F_{27}

Figure 1: The convergence curves of BSA and HBD for selected benchmark functions.

Table 3: Error values obtained by cHBD, HBD, and bHBD for CEC-2013 benchmark functions at $D = 30$.

	cHBD			HBD			bHBD
	$AVG_{Er} \pm STD_{Er}$		P value	$AVG_{Er} \pm STD_{Er}$		P value	$AVG_{Er} \pm STD_{Er}$
F_1	**1.01E − 30 ± 3.49E − 30**	=	0.125000	1.36E − 29 ± 4.17E − 29	=	0.328125	**6.31E − 30 ± 1.67E − 29**
F_2	**2.97E + 05 ± 1.51E + 05**	=	0.676637	3.15E + 05 ± 1.52E + 05	+	0.000240	7.07E + 05 ± 3.84E + 05
F_3	**3.77E + 06 ± 7.51E + 06**	=	0.618641	4.38E + 06 ± 8.11E + 06	=	0.637733	**4.25E + 06 ± 7.27E + 06**
F_4	**4.72E + 03 ± 1.90E + 03**	=	0.736617	5.05E + 03 ± 2.23E + 03	+	0.006848	6.55E + 03 ± 2.23E + 03
F_5	0.00E + 00 ± 0.00E + 00	=	1.000000	0.00E + 00 ± 0.00E + 00	=	1.000000	0.00E + 00 ± 0.00E + 00
F_6	1.12E + 01 ± 1.34E + 01	=	0.326049	1.12E + 01 ± 1.47E + 01	=	0.562928	**8.98E + 00 ± 5.93E + 00**
F_7	5.69E + 01 ± 1.50E + 01	=	0.142532	5.03E + 01 ± 1.61E + 01	+	0.003822	6.58E + 01 ± 1.69E + 01
F_8	**2.09E + 01 ± 4.71E − 02**	=	0.264150	2.10E + 01 ± 3.17E − 02	=	0.287862	**2.09E + 01 ± 4.67E − 02**
F_9	2.61E + 01 ± 3.12E + 00	+	0.000126	2.12E + 01 ± 4.26E + 00	+	0.000446	2.65E + 01 ± 3.29E + 00
F_{10}	**8.18E − 02 ± 5.50E − 02**	=	0.509755	9.15E − 02 ± 5.25E − 02	=	0.300241	**8.06E − 02 ± 4.77E − 02**
F_{11}	**0.00E + 00 ± 0.00E + 00**	−	0.015625	3.58E − 01 ± 6.34E − 01	−	0.015625	**0.00E + 00 ± 0.00E + 00**
F_{12}	8.19E + 01 ± 1.61E + 01	=	0.427339	8.09E + 01 ± 1.53E + 01	=	0.065311	9.01E + 01 ± 1.78E + 01
F_{13}	**1.26E + 02 ± 2.58E + 01**	=	0.798248	1.29E + 02 ± 2.90E + 01	=	0.174210	1.39E + 02 ± 2.67E + 01
F_{14}	4.01E + 00 ± 1.94E + 00	+	0.032428	2.83E + 00 ± 1.82E + 00	=	0.082653	3.88E + 00 ± 2.08E + 00
F_{15}	3.80E + 03 ± 4.02E + 02	+	0.017253	3.50E + 03 ± 4.82E + 02	+	0.014889	3.81E + 03 ± 2.84E + 02
F_{16}	**1.29E + 00 ± 2.45E − 01**	=	0.967806	1.31E + 00 ± 2.38E − 01	=	0.924971	**1.30E + 00 ± 2.09E − 01**
F_{17}	3.10E + 01 ± 1.75E − 01	=	0.165837	3.09E + 01 ± 1.79E − 01	=	0.989266	3.09E + 01 ± 1.55E − 01
F_{18}	1.17E + 02 ± 1.58E + 01	+	0.000665	9.46E + 01 ± 2.11E + 01	+	0.021418	1.10E + 02 ± 1.99E + 01
F_{19}	1.27E + 00 ± 2.22E − 01	=	0.427339	1.23E + 00 ± 2.28E − 01	=	0.924971	**1.22E + 00 ± 2.55E − 01**
F_{20}	1.12E + 01 ± 6.80E − 01	=	0.443172	1.11E + 01 ± 5.89E − 01	=	0.121828	1.14E + 01 ± 4.65E − 01
F_{21}	3.18E + 02 ± 8.04E + 01	=	0.300009	2.95E + 02 ± 7.95E + 01	+	0.040267	3.49E + 02 ± 8.29E + 01
F_{22}	5.28E + 01 ± 2.67E + 01	=	0.367385	4.48E + 01 ± 1.35E + 01	=	0.696425	4.58E + 01 ± 2.05E + 01
F_{23}	4.37E + 03 ± 4.03E + 02	=	0.121828	4.16E + 03 ± 5.07E + 02	+	0.039554	4.48E + 03 ± 4.85E + 02
F_{24}	2.28E + 02 ± 7.11E + 00	=	0.861162	2.28E + 02 ± 8.92E + 00	=	0.946369	2.28E + 02 ± 8.83E + 00
F_{25}	2.85E + 02 ± 2.02E + 01	=	0.054374	2.80E + 02 ± 8.80E + 00	=	0.275832	2.85E + 02 ± 1.03E + 01
F_{26}	2.00E + 02 ± 8.66E − 03	=	0.637733	2.00E + 02 ± 6.81E − 03	+	0.022988	2.00E + 02 ± 9.50E − 03
F_{27}	7.68E + 02 ± 1.45E + 02	=	0.736617	7.52E + 02 ± 1.33E + 02	=	0.092631	8.36E + 02 ± 1.67E + 02
F_{28}	**2.94E + 02 ± 2.96E + 01**	=	0.161513	3.00E + 02 ± 1.32E − 13	=	0.808365	3.00E + 02 ± 1.36E − 13
+/=/−			**4/23/1**			**9/18/1**	

Table 4: Results of the multiple-problem Wilcoxon test for HBD, cHBD, and bHBD for F_1–F_{28} at $D = 30$.

Algorithm	R^+	R^-	P-value	$\alpha = 0.05$	$\alpha = 0.1$
HBD versus cHBD	235	143	0.269095	=	=
HBD versus bHBD	276	75	0.010695	+	+

functions with the help of "+/=/−," respectively. However, paHBD uses the probability to control the use of DE. In this case, it can decrease the cost of function evaluation at early evolution stage. Thus, paHBD is almost similar to HBD according to "+/=/−."

In addition, we also perform the multiple-problem Wilcoxon test for HBD, aHBD, paHBD, and BSADE for 28 functions and list the results in Table 6.

It can be found from Table 6 that HBD is not significant to aHBD, paHBD, and BSADE. But HBD gets higher R^+ values than R^- values, compared with aHBD and BSADE, respectively. But HBD obtains slightly lower R^- value than R^+ value in comparison with paHBD. This is because HBD brings weakly lower accurate solutions on F_2, F_3, F_4, and F_{15}, resulting in higher ranking. Nevertheless, it indicates that the hybrid schema used in HBD is a reasonable choice.

4.5. The Effect of Probability Model. In HBD, a linear model seen (10) is used to select one individual to optimize. It is worth pointing out that other models, for example, nonlinear, can also be adopted in our algorithm. In this section, we do not seek the optimal probability model but only analyze the performance influenced by different models. Thus, two models, as similarly used in [29, 33], are employed to study the performance affected by other models. They are the quadratic model and the sinusoidal model, formulated as seen in (13) and (14), respectively. The average error values and the results of the multiple-problem Wilcoxon test are reported in Tables 7 and 8, respectively, where qHBD is HBD

Table 5: Error values obtained by aHBD, HBD, and paHBD for CEC-2013 benchmark functions at $D = 30$.

	aHBD			paHBD			BSADE			HBD
	$AVG_{Er} \pm STD_{Er}$		P value	$AVG_{Er} \pm STD_{Er}$		P value	$AVG_{Er} \pm STD_{Er}$		P value	$AVG_{Er} \pm STD_{Er}$
F_1	1.29E − 28 ± 1.18E − 28	+	0.000292	5.53E − 29 ± 8.39E − 29	+	0.030946	**1.11E − 29 ± 4.11E − 29**	=	0.666016	1.36E − 29 ± 4.17E − 29
F_2	**5.01E + 04 ± 2.88E + 04**	−	0.000012	**5.64E + 04 ± 3.39E + 04**	−	0.000012	**6.80E + 04 ± 3.33E + 04**	−	0.000012	3.15E + 05 ± 1.52E + 05
F_3	1.13E + 07 ± 3.20E + 07	=	0.924971	**1.74E + 06 ± 2.61E + 06**	=	0.201222	5.21E + 06 ± 1.24E + 07	=	0.696425	4.38E + 06 ± 8.11E + 06
F_4	**5.61E + 01 ± 5.41E + 01**	−	0.000012	**2.40E + 02 ± 2.17E + 02**	−	0.000012	**4.02E + 02 ± 2.24E + 02**	−	0.000012	5.05E + 03 ± 2.23E + 03
F_5	4.26E − 16 ± 1.48E − 15	+	0.007813	**2.84E − 16 ± 1.42E − 15**	=	0.062500	0.00E + 00 ± 0.00E + 00	=	1.000000	0.00E + 00 ± 0.00E + 00
F_6	**5.01E + 00 ± 5.53E + 00**	−	0.017253	**4.67E + 00 ± 3.66E + 00**	−	0.003822	1.18E + 01 ± 1.25E + 01	=	0.396679	1.12E + 01 ± 1.47E + 01
F_7	**4.31E + 01 ± 1.62E + 01**	=	0.097970	**4.74E + 01 ± 1.50E + 01**	=	0.492633	5.87E + 01 ± 1.63E + 01	=	0.097970	5.03E + 01 ± 1.61E + 01
F_8	**2.09E + 01 ± 4.16E − 02**	=	0.300241	**2.09E + 01 ± 6.05E − 02**	=	0.509755	**2.09E + 01 ± 4.74E − 02**	=	0.210872	2.10E + 01 ± 3.17E − 02
F_9	2.18E + 01 ± 3.56E + 00	=	0.861162	2.18E + 01 ± 3.42E + 00	=	0.637733	2.60E + 01 ± 2.82E + 00	+	0.000296	2.12E + 01 ± 4.26E + 00
F_{10}	**6.25E − 02 ± 3.76E − 02**	−	0.014889	**6.11E − 02 ± 3.92E − 02**	−	0.022988	2.04E − 01 ± 1.19E − 01	+	0.000126	9.15E − 02 ± 5.25E − 02
F_{11}	3.03E + 01 ± 1.53E + 01	+	0.000012	3.06E + 00 ± 2.05E + 00	+	0.000083	**7.96E − 02 ± 2.75E − 01**	=	0.062500	3.58E − 01 ± 6.34E − 01
F_{12}	**7.43E + 01 ± 1.98E + 01**	=	0.264150	8.12E + 01 ± 1.39E + 01	=	0.967806	8.64E + 01 ± 2.23E + 01	=	0.241820	8.09E + 01 ± 1.53E + 01
F_{13}	1.38E + 02 ± 2.88E + 01	=	0.509755	1.35E + 02 ± 2.66E + 01	=	0.367385	1.43E + 02 ± 1.76E + 01	=	0.054374	1.29E + 02 ± 2.90E + 01
F_{14}	5.85E + 01 ± 5.92E + 01	+	0.000012	7.23E + 00 ± 4.53E + 00	+	0.000266	7.33E + 00 ± 4.72E + 00	+	0.000808	2.83E + 00 ± 1.82E + 00
F_{15}	**3.45E + 03 ± 4.72E + 02**	=	0.339479	**3.49E + 03 ± 5.12E + 02**	=	0.861162	**3.39E + 03 ± 3.92E + 02**	=	0.381860	3.50E + 03 ± 4.82E + 02
F_{16}	1.49E + 00 ± 3.48E − 01	+	0.008041	**1.14E + 00 ± 5.92E − 01**	=	0.676637	**1.23E + 00 ± 5.19E − 01**	=	0.882352	1.31E + 00 ± 2.38E − 01
F_{17}	3.92E + 01 ± 9.55E + 00	+	0.000012	3.17E + 01 ± 7.18E − 01	+	0.000023	3.17E + 01 ± 5.55E − 01	+	0.000025	3.09E + 01 ± 1.79E − 01
F_{18}	1.05E + 02 ± 2.28E + 01	=	0.121828	**8.78E + 01 ± 2.07E + 01**	=	0.300241	**8.84E + 01 ± 1.83E + 01**	=	0.411840	9.46E + 01 ± 2.11E + 01
F_{19}	2.10E + 00 ± 1.09E + 00	+	0.000036	1.44E + 00 ± 2.94E − 01	+	0.011000	1.47E + 00 ± 2.13E − 01	+	0.000602	1.23E + 00 ± 2.28E − 01
F_{20}	**1.07E + 01 ± 5.66E − 01**	−	0.004530	1.13E + 01 ± 4.56E − 01	=	0.562928	1.13E + 01 ± 6.33E − 01	=	0.427339	1.11E + 01 ± 5.89E − 01
F_{21}	3.26E + 02 ± 9.08E + 01	=	0.121477	3.41E + 02 ± 9.20E + 01	=	0.061407	**2.81E + 02 ± 6.67E + 01**	=	0.894867	2.95E + 02 ± 7.95E + 01
F_{22}	2.47E + 02 ± 2.57E + 02	+	0.000014	5.78E + 01 ± 3.61E + 01	=	0.492633	6.63E + 01 ± 2.99E + 01	+	0.002947	4.48E + 01 ± 1.35E + 01
F_{23}	**3.86E + 03 ± 5.21E + 02**	=	0.252813	4.21E + 03 ± 5.58E + 02	=	0.339479	4.19E + 03 ± 4.20E + 02	=	0.819095	4.16E + 03 ± 5.07E + 02
F_{24}	2.36E + 02 ± 8.94E + 00	+	0.008705	**2.27E + 02 ± 9.76E + 00**	=	0.562928	2.35E + 02 ± 9.21E + 00	+	0.009417	2.28E + 02 ± 8.92E + 00
F_{25}	**2.75E + 02 ± 1.32E + 01**	=	0.103553	2.80E + 02 ± 1.02E + 01	=	0.756995	2.86E + 02 ± 1.34E + 01	=	0.069337	2.80E + 02 ± 8.80E + 00
F_{26}	2.11E + 02 ± 3.75E + 01	+	0.002470	2.00E + 02 ± 3.02E − 03	−	0.000016	2.00E + 02 ± 1.85E − 03	−	0.000012	2.00E + 02 ± 6.81E − 03
F_{27}	**7.42E + 02 ± 1.07E + 02**	=	0.882352	7.58E + 02 ± 1.52E + 02	=	0.989266	9.19E + 02 ± 1.49E + 02	+	0.000891	7.52E + 02 ± 1.33E + 02
F_{28}	3.00E + 02 ± 1.90E − 13	+	0.016377	3.00E + 02 ± 1.95E − 13	+	0.038947	3.00E + 02 ± 1.65E − 13	=	1.000000	3.00E + 02 ± 1.32E − 13
+/=/−			**10/12/6**			**6/17/5**	**8/17/3**			

Table 6: Results of the multiple-problem Wilcoxon test for HBD, aHBD, paHBD, and BSADE for F_1–F_{28} at $D = 30$.

Algorithm	R^+	R^-	P value	$\alpha = 0.05$	$\alpha = 0.1$
HBD versus aHBD	226	180	0.600457	=	=
HBD versus paHBD	197	209	0.891321	=	=
HBD versus BSADE	262.5	143.5	0.175450	=	=

Table 7: Error values obtained by qHBD, HBD, and sHBD for CEC-2013 benchmark functions at $D = 30$.

	qHBD			HBD			sHBD
	$\text{AVG}_{\text{Er}} \pm \text{STD}_{\text{Er}}$		P value	$\text{AVG}_{\text{Er}} \pm \text{STD}_{\text{Er}}$		P value	$\text{AVG}_{\text{Er}} \pm \text{STD}_{\text{Er}}$
F_1	$\mathbf{5.05E-31 \pm 2.52E-30}$	=	0.078125	$1.36E-29 \pm 4.17E-29$	=	0.429688	$\mathbf{6.18E-30 \pm 2.22E-29}$
F_2	$3.56E+05 \pm 2.89E+05$	=	0.798248	$3.15E+05 \pm 1.52E+05$	=	0.736617	$\mathbf{2.87E+05 \pm 1.36E+05}$
F_3	$4.94E+06 \pm 7.32E+06$	=	0.618641	$4.38E+06 \pm 8.11E+06$	=	0.756995	$3.57E+06 \pm 6.05E+06$
F_4	$\mathbf{4.69E+03 \pm 1.64E+03}$	=	0.736617	$5.05E+03 \pm 2.23E+03$	=	0.903627	$5.20E+03 \pm 2.96E+03$
F_5	$2.84E-16 \pm 1.42E-15$	=	1.000000	$0.00E+00 \pm 0.00E+00$	=	1.000000	$0.00E+00 \pm 0.00E+00$
F_6	$\mathbf{1.05E+01 \pm 1.31E+01}$	=	0.381860	$1.12E+01 \pm 1.47E+01$	=	0.861162	$\mathbf{9.06E+00 \pm 6.44E+00}$
F_7	$5.68E+01 \pm 1.66E+01$	=	0.191898	$5.03E+01 \pm 1.61E+01$	=	0.756995	$5.19E+01 \pm 1.93E+01$
F_8	$\mathbf{2.09E+01 \pm 6.58E-02}$	=	0.509755	$2.10E+01 \pm 3.17E-02$	=	0.165837	$\mathbf{2.09E+01 \pm 5.52E-02}$
F_9	$2.38E+01 \pm 3.78E+00$	+	0.039554	$2.12E+01 \pm 4.26E+00$	=	0.231167	$2.24E+01 \pm 3.57E+00$
F_{10}	$9.15E-02 \pm 6.35E-02$	=	0.264150	$9.15E-02 \pm 5.25E-02$	=	0.527183	$9.80E-02 \pm 6.25E-02$
F_{11}	$4.38E-01 \pm 9.56E-01$	=	0.986328	$3.58E-01 \pm 6.34E-01$	=	0.366699	$\mathbf{1.99E-01 \pm 4.06E-01}$
F_{12}	$\mathbf{7.44E+01 \pm 1.79E+01}$	=	0.073565	$8.09E+01 \pm 1.53E+01$	=	0.374558	$\mathbf{7.83E+01 \pm 1.32E+01}$
F_{13}	$\mathbf{1.26E+02 \pm 2.53E+01}$	=	0.989266	$1.29E+02 \pm 2.90E+01$	=	0.411840	$\mathbf{1.26E+02 \pm 2.94E+01}$
F_{14}	$3.14E+00 \pm 1.74E+00$	=	0.618641	$2.83E+00 \pm 1.82E+00$	=	0.051087	$3.66E+00 \pm 1.60E+00$
F_{15}	$3.60E+03 \pm 4.07E+02$	=	0.903627	$3.50E+03 \pm 4.82E+02$	=	0.676637	$\mathbf{3.49E+03 \pm 3.98E+02}$
F_{16}	$\mathbf{1.29E+00 \pm 2.55E-01}$	=	0.967806	$1.31E+00 \pm 2.38E-01$	=	0.696425	$\mathbf{1.28E+00 \pm 2.15E-01}$
F_{17}	$3.10E+01 \pm 1.86E-01$	=	0.492633	$3.09E+01 \pm 1.79E-01$	=	0.946369	$3.09E+01 \pm 2.11E-01$
F_{18}	$9.93E+01 \pm 1.75E+01$	=	0.396679	$9.46E+01 \pm 2.11E+01$	=	0.840072	$9.52E+01 \pm 1.75E+01$
F_{19}	$\mathbf{1.20E+00 \pm 2.23E-01}$	=	0.736617	$1.23E+00 \pm 2.28E-01$	=	0.210872	$1.26E+00 \pm 2.04E-01$
F_{20}	$\mathbf{1.09E+01 \pm 7.81E-01}$	=	0.165837	$1.11E+01 \pm 5.89E-01$	=	0.191898	$1.14E+01 \pm 5.24E-01$
F_{21}	$3.10E+02 \pm 8.68E+01$	=	0.480701	$2.95E+02 \pm 7.95E+01$	=	0.165492	$3.28E+02 \pm 8.02E+01$
F_{22}	$\mathbf{4.19E+01 \pm 1.90E+01}$	=	0.443172	$4.48E+01 \pm 1.35E+01$	=	0.287862	$\mathbf{4.24E+01 \pm 1.86E+01}$
F_{23}	$4.16E+03 \pm 4.19E+02$	=	0.924971	$4.16E+03 \pm 5.07E+02$	=	0.840072	$\mathbf{4.12E+03 \pm 3.91E+02}$
F_{24}	$2.31E+02 \pm 1.02E+01$	=	0.637733	$2.28E+02 \pm 8.92E+00$	=	0.777543	$2.28E+02 \pm 9.52E+00$
F_{25}	$\mathbf{2.76E+02 \pm 9.42E+00}$	−	0.042207	$2.80E+02 \pm 8.80E+00$	=	0.157770	$\mathbf{2.75E+02 \pm 1.43E+01}$
F_{26}	$2.00E+02 \pm 6.05E-03$	=	0.777543	$2.00E+02 \pm 6.81E-03$	=	0.840072	$2.00E+02 \pm 6.31E-03$
F_{27}	$7.82E+02 \pm 1.48E+02$	=	0.287862	$7.52E+02 \pm 1.33E+02$	=	0.459336	$7.90E+02 \pm 1.21E+02$
F_{28}	$3.00E+02 \pm 1.14E-13$	=	1.000000	$3.00E+02 \pm 1.32E-13$	−	0.025347	$3.00E+02 \pm 8.76E-14$
+/=/−			**1/26/1**			**0/27/1**	

Table 8: Results of the multiple-problem Wilcoxon test for HBD, qHBD, and sHBD for F_1–F_{28} at $D = 30$.

Algorithm	R^+	R^-	P value	$\alpha = 0.05$	$\alpha = 0.1$
HBD versus qHBD	238	140	0.239106	=	=
HBD versus sHBD	143	208	0.409125	=	=

with the quadratic model and sHBD means HBD with the sinusoidal one. Consider

$$p_i = \left(\frac{r_i}{N}\right)^2 \tag{13}$$

$$p_i = 0.5\left(1.0 - \cos\left(\frac{r_i}{N}\pi\right)\right). \tag{14}$$

From Table 7, we can find that qHBD can bring higher solutions to 11 out of 28 functions compared with HBD, although the results they obtain are not significant in terms of "+/=/−." In addition, qHBD gets lower R^- values than R^+ values HBD gained, though they are not significant at the 5% and 10% significance level. It says that the linear model is a reasonable choice compared with the quadratic model. However, it is not the optimal one compared with

Table 9: Fitness obtained by HBD and 6 non-BSAs for CEC-2005 functions at $D = 10$.

		PSO2011	CMAES	ABC	JDE	CLPSO	SADE	HBD
F_1	Mean	−450.0000000000000000	−450.0000000000000000	−450.0000000000000000	−450.0000000000000000	−450.0000000000000000	−450.0000000000000000	−450.0000000000000000
	Std.	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000	0.0000000000000000
F_2	Mean	−450.0000000000000000	−450.0000000000000000	−449.9999999999220000	−450.0000000000000000	−418.8551838547760000	−450.0000000000000000	−450.0000000000000000
	Std.	0.0000000000000350	0.0000000000000000	0.0000000002052730	0.0000000000000615	51.0880511039985000	0.0000000000000000	0.0000000000000000
F_3	Mean	−44.5873911956554000	−450.0000000000000000	387131.2441213970000000	−197.9999999999850000	62142.0000000000000000	245.0483283713550000	−449.9999999999980000
	Std.	458.5794120016290000	0.0000000000000000	166951.7336592640000000	391.5169437474990000	34796.1785167236000000	790.6056596723160000	0.0000000000012208
F_4	Mean	−450.0000000000000000	77982.4567046980000000	140.4509447125110000	−414.0000000000000000	−178.8320689185280000	−450.0000000000000000	−450.0000000000000000
	Std.	0.0000000000000460	131376.7365456010000000	217.2646715063190000	55.9309919639279000	394.8667499339530000	0.0000000000000000	0.0000000000000000
F_5	Mean	−310.0000000000000000	−310.0000000000000000	−291.5327549384120000	−271.0000000000000000	333.4108259915760000	−309.9999999999960000	−309.9999999999990000
	Std.	0.0000000000000000	0.0000000000000000	17.6942171217937000	60.5919079609218000	512.6920837704510000	0.0000000000133965	0.0000000000017057
F_6	Mean	393.4959999056240000	390.5315438816460000	391.2531452421960000	231.3986579112350000	405.5233436479650000	390.2657719408230000	390.2657719408230000
	Std.	16.0224965900462000	1.3783433976373800	3.7254660805238600	247.2968415284400000	10.7480096852869000	1.0114275384776600	1.0114275384776500
F_7	Mean	1091.0644335162500000	1087.2645466786700000	1087.0459486286000000	1141.0459486286000000	1087.0459486286000000	1087.0459486286000000	−179.9480783793820000
	Std.	3.4976948942723200	0.5365230018001780	0.0000000000005585	83.8964879458918000	0.0000000000004264	0.0000000000004814	0.0353036104861447
F_8	Mean	−119.8190232990920000	−119.9261073509850000	−119.7446063439080000	−119.4459380180300000	−119.9300269839980000	−119.7727713703720000	−119.8110293720060000
	Std.	0.0720107560874199	0.1554021446157740	0.0623866434489108	0.0927418223065644	0.0417913553101429	0.1248514853682450	0.0915374510176448
F_9	Mean	−324.6046006320200000	−306.5782069681560000	−330.0000000000000000	−329.8673387923880000	−329.4361898676470000	−329.9668346980970000	−329.9668346980960000
	Std.	2.5082306041521000	21.9475396048756000	0.0000000000000000	0.3440030182812760	0.6229063711904190	0.1816538397880230	0.1816538397880230
F_{10}	Mean	−324.3311322538170000	−314.7871102989330000	−306.7949047862760000	−319.6763749798700000	−321.7278926895280000	−322.9689591871600000	−320.3489045380250000
	Std.	3.0072222933667300	8.3115989308305500	5.1787864195870400	4.9173541245304800	1.8971778613701300	2.8254645254663600	4.1899978130687500
F_{11}	Mean	92.5640111212146000	90.7642785704506000	94.8428485804138000	93.2972315784963000	94.6109567642977000	91.6859083842723000	92.0330962077418000
	Std.	1.5827416781636900	26.4613831425879000	0.6869412813090850	1.8766951726453600	0.6689129174038950	0.9033073777915270	1.4570152623440700
F_{12}	Mean	18611.3142254809000000	−70.0486708747625000	−337.3273080760500000	400.3240208136310000	−447.8870804905020000	−394.5206365378250000	−453.7580906206240000
	Std.	12508.7866126316000000	637.4585182420270000	56.5730759032367000	688.3344299264300000	11.8934815947019000	128.6353424718180000	7.9399805117226300
F_{13}	Mean	−129.2373581503910000	−128.7850616923410000	−129.8343428775830000	−129.6294851450880000	−129.8382867796110000	−129.7129164862680000	−129.7795909188150000
	Std.	0.5986210944493790	0.6157633658946230	0.0408016481905455	0.1054759371085400	0.0372256921835666	0.0875456568200232	0.0974559125367515
F_{14}	Mean	−298.2835926212850000	−295.1290938304830000	−296.9323391084610000	−296.8839733969750000	−297.5119726691150000	−297.8403738182600000	−297.1054029448800000
	Std.	0.5587676271753680	0.1634039984609270	0.2251930667702880	0.4330673614598290	0.3440115280624180	0.4536801689800720	0.3062178045203870
F_{15}	Mean	417.4613663019860000	492.5045364088000000	120.0000000000000000	326.6601114362900000	131.3550392249760000	234.2689845349590000	134.6767426915020000
	Std.	153.9215808771580000	181.5709657779580000	0.0000000000000188	174.6877238188330000	26.1407360548431000	150.7595974059750000	23.3648038768225000
F_{16}	Mean	221.4232628350220000	455.4454684594550000	258.8582688922670000	231.1806131539990000	231.5547154800990000	222.0256674919140000	231.8426524963750000
	Std.	12.2450207482898000	254.3583511786970000	11.8823213189685000	13.5473380962764000	11.5441451076421000	6.1841489800660300	10.3007095087283000
F_{17}	Mean	217.3338617866620000	681.0349114021570000	265.0370119084380000	228.7309024901770000	240.3635189964930000	221.1801916743850000	230.6398805937190000
	Std.	20.6685850658838000	488.0618274343640000	12.4033917090208000	12.3682716268631000	14.8435137485293000	5.7037006844690500	10.7176191104135000
F_{18}	Mean	668.9850326105730000	926.9488078829420000	513.8925774904480000	743.9859973770210000	892.4391527217660000	845.4504613493740000	626.6666666666660000
	Std.	275.8071370273340000	174.1027182659660000	31.0124861524005000	175.6497294240330000	79.1422224454971000	120.8505129523180000	245.0662589267800000
F_{19}	Mean	708.2979222913040000	831.2324139697050000	500.5478931040730000	776.5150806087790000	863.8929608090610000	809.7183195902260000	673.6943739862640000
	Std.	256.2419561521300000	289.7296413284470000	31.2240894705539000	160.7307526692470000	96.5618989087194000	147.3158109824600000	240.7952379798540000
F_{20}	Mean	711.2970397614200000	876.9306161887680000	483.2984167460740000	761.2954767038960000	844.6391674419360000	810.5227124472170000	568.3524718683710000
	Std.	258.9317052508320000	289.7296413284470000	99.3976740616107000	163.4084080636550000	113.6848457105400000	104.7139423525340000	264.1320553106420000
F_{21}	Mean	1117.8857079625100000	1258.1065536572400000	659.5351969346130000	959.3735119754180000	911.4640642691360000	990.8546718748010000	887.0165496528500000
	Std.	311.0011859260640000	359.7382897536570000	98.5410511961986000	240.5568407069990000	238.3180009803040000	235.1014092849970000	130.7951954796500000
F_{22}	Mean	1094.8305116977000000	$-7.159E+49$	915.4958100611630000	1133.7536009808600000	1075.5292326436900000	1094.6823697304900000	1033.4073460688400000
	Std.	121.3539576317800000	$4.387E+50$	242.1993331983530000	42.1171260000361000	166.9355145236330000	87.9884000140656000	170.5780088721150000
F_{23}	Mean	1304.3661550124000000	1159.9280867973000000	830.2290165794410000	1167.9040488743800000	1070.4327462836400000	1105.2511774948600000	985.5892101103390000
	Std.	262.1065863453340000	742.1215416320490000	60.2286903507069000	236.7325108248320000	203.0676627074300000	190.6172874229610000	140.3217672954560000
F_{24}	Mean	500.0000000000000000	653.3355378428050000	460.0000000000000000	510.0000000000000000	493.3333333333400000	490.0000000000000000	460.0000000000000000
	Std.	103.7237710925280000	302.5312999719650000	0.0000000000016493	113.7147065368360000	137.2973951415090000	91.5385729888094000	0.0000000000000000
F_{25}	Mean	1107.9038127876700000	1401.6553278264300000	930.4565414149210000	1072.9924659809200000	1258.5157766524700000	1074.3695435628600000	632.0799172389040000
	Std.	127.9566489362040000	253.2428066220210000	87.9959072391079000	2.2606058314671500	241.4024507676890000	2.8314182838917800	3.5935258481745500

Table 10: Results of the multiple-problem Wilcoxon test for seven algorithms for CEC2005 functions at $D = 10$.

Algorithm	R^+	R^-	P value	$\alpha = 0.05$	$\alpha = 0.1$
HBD versus SPSO2011	**261.95**	63.05	0.007453	=	+
HBD versus CMAES	**275.90**	49.10	0.002279	+	+
HBD versus ABC	**179.93**	145.07	0.639106	=	=
HBD versus JDE	**276.00**	49.00	0.002259	+	+
HBD versus CLPSO	**291.00**	34.00	0.000545	+	+
HBD versus SADE	**234.50**	90.50	0.052709	=	+

Table 11: Average ranking of seven algorithms by the Friedman test for CEC2005 functions at $D = 10$.

Algorithm	SPSO2011	CMAES	ABC	JDE	CLPSO	SADE	HBD
Ranking	4.06	5.24	3.58	4.44	4.60	3.42	**2.66**

the sinusoidal model. For instance, sHBD wins, ties, and loses HBD in 1, 27, and 0 out of 28 functions according to "+/=/−." Moreover, sHBD has higher R^- values than R^+ values HBD does though they are not significant at the 5% and 10% significance level.

4.6. Compared with Other Algorithms. Firstly, HBD is compared with 6 non-BSA approaches in [17], namely, PSO2011 [34], CMAES [35, 36], ABC [7], JDE [37], CLPSO [38], and SADE [39]. Moreover, to compare fair and conveniently, we use the 25 functions and the parameters which are employed and suggested in [17]. More details about these 25 functions can be found in CEC-2005 competition [40]. Table 9 lists the minimal fitness and average fitness of 7 approaches, where the results of 6 non-BSA algorithms are adopted from [17] directly. In addition, the results of multiple-problem Wilcoxon test and Friedman test similarly done in [29] for the seven algorithms are listed in Tables 10 and 11, respectively.

From Table 9, we find that each algorithm does well in some functions according to its average error value. For instance, PSO2011, CMAES, ABC, JDE, CLPSO, SADE, and HBD perform better in 8, 5, 9, 3, 3, 3, and 7 out of 25 functions, respectively. However, Table 10 shows that HBD gets higher R^+ values than R^- values in all cases. This suggests that HBD is better than the other 6 algorithms. Moreover, for Wilcoxon test at $\alpha = 0.05$ and $\alpha = 0.01$ in three cases, there are significant differences for CEC2005 functions. Furthermore, with respect to the average rankings of different algorithms by the Friedman test, it can be seen clearly from Table 11 that HBD offers the best overall performance, while SADE is the second best, followed by ABC, PSO2011, CLPSO, JDE, and CMAES.

Secondly, to appreciate the actual performance of the proposed algorithm, HBD is in comparison with the other five algorithms identified as NBIPOP-aCMA [41], fk-PSO [42], SPSO2011 [43], SPSOABC [44], and PVADE [45], which were presented during the CEC-2013 Special Session & Competition on Real-Parameter Single Objective Optimization.

Table 12 lists the average error values which are dealt with from [46], and the average rankings of the six algorithms by the Friedman test for CEC-2013 functions at $D = 30$ are given in Table 13. Since NBIPOP-aCMA is one of top three performing algorithms for CEC-2013 functions [47], seen from Table 12, it shows the promising performance in almost all of functions. Other algorithms bring solutions with higher accuracy in a handful of functions. For example, fk-PSO, SPSO2011, SPSOABC, PVADE, and HBD yield the better performance on 3, 2, 6, 4, and 5 out of 28 functions in terms of the average error values. However, according to the average rankings of different algorithms by the Friedman test in Table 13, we can find that NBIPOP-aCMA is the best, and HBD offers the second best overall performance, followed by SPSOABC, fk-PSO, PVADE, and PSO2011.

5. Conclusion

In this paper, we presented a hybrid BSA, called HBD, which combined BSA and DE with exploitive mutation strategy. At each iteration process, DE was embedded behind the BSA algorithm to optimize one individual which was selected according to its probability in order to enhance the convergence of BSA and to bring solutions with higher quality.

Comprehensive experiments have been carried out in 28 benchmark functions proposed in CEC-2013 competition. The experimental results reveal that the hybridization of BSA and DE provides the high effectiveness and efficiency in most of functions, contributing to solutions with higher accuracy, faster convergence speed, and more stable scalability. HBD was also compared with other evolutionary algorithms and has shown its promising performance.

There are several interesting directions for future work. Experimentally, the linear probability model used to select one individual to optimize is a reasonable but not optimal one; thus, firstly, the comprehensive tests will be performed on various probability models in HBD. Secondly, although experimental results have shown that HBD owns the stable scalability, we plan to investigate HBD for large-scale optimization problems. Last but not least, we plan to apply HBD to some real-world optimization problems for further examinations.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Table 12: Error values obtained by HBD and 5 compared algorithms for CEC-2013 benchmark functions at $D = 30$.

	NBIPOP-aCMA	fk-PSO	SPSO2011	SPSOABC	PVADE	HBD
	$AVG_{Er} \pm STD_{Er}$	$AVG_{Er} \pm STD_{Er}$	$AVG_{Er} \pm STD_{Er}$	$AVG_{Er} \pm STD_{Er}$	$AVG_{Er} \pm STD_{Er}$	$AVG_{Er} \pm STD_{Er}$
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**	1.36E−29 ± 4.17E−29
F_2	**0.00E+00 ± 0.00E+00**	1.59E+06 ± 8.03E+05	3.38E+05 ± 1.67E+05	8.78E+05 ± 1.69E+06	2.12E+06 ± 1.56E+06	3.15E+05 ± 1.52E+05
F_3	**0.00E+00 ± 0.00E+00**	2.40E+08 ± 3.71E+08	2.88E+08 ± 5.24E+08	5.16E+07 ± 8.00E+07	1.65E+03 ± 2.83E+03	4.38E+06 ± 8.11E+06
F_4	**0.00E+00 ± 0.00E+00**	4.78E+02 ± 1.96E+02	3.86E+04 ± 6.70E+03	6.02E+03 ± 2.30E+03	1.70E+04 ± 2.85E+03	5.05E+03 ± 2.23E+03
F_5	**0.00E+00 ± 0.00E+00**	**0.00E+00 ± 0.00E+00**	5.42E−04 ± 4.91E−05	**0.00E+00 ± 0.00E+00**	1.40E−07 ± 1.86E−07	**0.00E+00 ± 0.00E+00**
F_6	**0.00E+00 ± 0.00E+00**	2.99E+01 ± 1.76E+01	3.79E+01 ± 2.83E+01	1.09E+01 ± 1.09E+01	8.29E+00 ± 5.82E+00	1.12E+01 ± 1.47E+01
F_7	2.31E+00 ± 6.05E+00	6.39E+01 ± 3.09E+01	8.79E+01 ± 2.11E+01	5.12E+01 ± 2.04E+01	**1.29E+00 ± 1.22E+00**	5.03E+01 ± 1.61E+01
F_8	**2.09E+01 ± 4.80E−02**	**2.09E+01 ± 6.28E−02**	**2.09E+01 ± 5.89E−02**	**2.09E+01 ± 4.92E−02**	**2.09E+01 ± 4.82E−02**	2.10E+01 ± 3.17E−02
F_9	**3.30E+00 ± 1.38E+00**	1.85E+01 ± 2.69E+00	2.88E+01 ± 4.43E+00	2.95E+01 ± 2.62E+00	6.30E+00 ± 3.27E+00	2.12E+01 ± 4.26E+00
F_{10}	**0.00E+00 ± 0.00E+00**	2.29E−01 ± 1.32E−01	3.40E−01 ± 1.48E−01	1.32E−01 ± 6.23E−02	2.16E−02 ± 1.36E−02	9.15E−02 ± 5.25E−02
F_{11}	3.04E+00 ± 1.41E+00	2.36E+01 ± 8.76E+00	1.05E+02 ± 2.74E+01	**0.00E+00 ± 0.00E+00**	5.84E+01 ± 1.11E+01	3.58E−01 ± 6.34E−01
F_{12}	**2.91E+00 ± 1.38E+00**	5.64E+01 ± 1.51E+01	1.04E+02 ± 3.54E+01	6.44E+01 ± 1.48E+01	1.15E+02 ± 1.14E+01	8.09E+01 ± 1.53E+01
F_{13}	**2.78E+00 ± 1.45E+00**	1.23E+02 ± 2.19E+01	1.94E+02 ± 3.86E+01	1.15E+02 ± 2.24E+01	1.31E+02 ± 1.24E+01	1.29E+02 ± 2.90E+01
F_{14}	8.10E+02 ± 3.60E+02	7.04E+02 ± 2.38E+02	3.99E+03 ± 6.19E+02	1.55E+01 ± 6.13E+00	3.20E+03 ± 4.38E+02	**2.83E+00 ± 1.82E+00**
F_{15}	**7.65E+02 ± 2.95E+02**	3.42E+03 ± 5.16E+02	3.81E+03 ± 6.94E+02	3.55E+03 ± 3.04E+02	5.16E+03 ± 3.19E+02	3.50E+03 ± 4.82E+02
F_{16}	**4.40E−01 ± 9.26E−01**	8.48E−01 ± 2.20E−01	1.31E+00 ± 3.59E−01	1.03E+00 ± 2.01E−01	2.39E+00 ± 2.66E−01	1.31E+00 ± 2.38E−01
F_{17}	3.44E+01 ± 1.87E+00	5.26E+01 ± 7.11E+00	1.16E+02 ± 2.02E+01	**3.09E+01 ± 1.23E−01**	1.02E+02 ± 1.17E+01	**3.09E+01 ± 1.79E−01**
F_{18}	**6.23E+01 ± 4.56E+01**	6.81E+01 ± 9.68E+00	1.21E+02 ± 2.46E+01	9.01E+01 ± 8.95E+00	1.82E+02 ± 1.20E+01	9.46E+01 ± 2.11E+01
F_{19}	2.23E+00 ± 3.41E−01	3.12E+00 ± 9.83E−01	9.51E+00 ± 4.42E+00	1.71E+00 ± 4.68E−01	5.40E+00 ± 8.10E−01	**1.23E+00 ± 2.28E−01**
F_{20}	1.29E+01 ± 5.98E−01	1.20E+01 ± 9.26E−01	1.35E+01 ± 1.11E+00	**1.11E+01 ± 7.60E−01**	1.13E+01 ± 3.28E−01	**1.11E+01 ± 5.89E−01**
F_{21}	**1.92E+02 ± 2.72E+01**	3.11E+02 ± 7.92E+01	3.09E+02 ± 6.80E+01	3.18E+02 ± 7.53E+01	3.19E+02 ± 6.26E+01	2.95E+02 ± 7.95E+01
F_{22}	8.38E+02 ± 4.60E+02	8.59E+02 ± 3.10E+02	4.30E+03 ± 7.67E+02	8.41E+01 ± 3.90E+01	2.50E+03 ± 3.86E+02	**4.48E+01 ± 1.35E+01**
F_{23}	**6.67E+02 ± 2.90E+02**	3.57E+03 ± 5.90E+02	4.83E+03 ± 8.23E+02	4.18E+03 ± 5.62E+02	5.81E+03 ± 5.04E+02	4.16E+03 ± 5.07E+02
F_{24}	**1.62E+02 ± 3.00E+01**	2.48E+02 ± 8.11E+00	2.67E+02 ± 1.25E+01	2.51E+02 ± 1.43E+01	2.02E+02 ± 1.40E+00	2.28E+02 ± 8.92E+00
F_{25}	**2.20E+02 ± 1.11E+01**	2.49E+02 ± 7.82E+00	2.99E+02 ± 1.05E+01	2.75E+02 ± 9.76E+00	2.30E+02 ± 2.08E+01	2.80E+02 ± 8.80E+00
F_{26}	**1.58E+02 ± 3.00E+01**	2.95E+02 ± 7.06E+01	2.86E+02 ± 8.24E+01	2.60E+02 ± 7.62E+01	2.18E+02 ± 4.01E+01	2.00E+02 ± 6.81E−03
F_{27}	4.69E+02 ± 7.38E+01	7.76E+02 ± 7.11E+01	1.00E+03 ± 1.12E+02	9.10E+02 ± 1.62E+02	**3.26E+02 ± 1.14E+01**	7.52E+02 ± 1.33E+02
F_{28}	**2.69E+02 ± 7.35E+01**	4.01E+02 ± 3.48E+02	4.01E+02 ± 4.76E+02	3.33E+02 ± 2.32E+02	3.00E+02 ± 2.24E−02	3.00E+02 ± 1.32E−13

Table 13: Average ranking of six algorithms by the Friedman test for CEC2013 functions at $D = 30$.

Algorithm	NBIPOP-aCMA	fk-PSO	SPSO2011	SPSOABC	PVADE	HBD
Ranking	**1.80**	3.61	5.29	3.34	3.95	***3.02***

Acknowledgments

The authors are very grateful to the editor and the anonymous reviewers for their constructive comments and suggestions to this paper. This work was supported by the NSFC Joint Fund with Guangdong of China under Key Project U1201258, the Shandong Natural Science Funds for Distinguished Young Scholar under Grant no. JQ201316, and the Natural Science Foundation of Fujian Province of China under Grant no. 2013J01216.

References

[1] J. H. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, Mich, USA, 1975.

[2] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proceedings of the 6th International Symposium on Micro Machine and Human Science*, vol. 1, pp. 39–43, New York, NY, USA, October 1995.

[3] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, vol. 4, pp. 1942–1948, IEEE, Perth, Australia, December 1995.

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

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

[6] Y. Cai and J. Wang, "Differential evolution with neighborhood and direction information for numerical optimization," *IEEE Transactions on Cybernetics*, vol. 43, no. 6, pp. 2202–2215, 2013.

[7] D. Karaboga, "An idea based on honey bee swarm for numerical optimization," Tech. Rep. TR-06, Erciyes University, Engineering Faculty, Computer Engineering Department, 2005.

[8] X.-S. Yang and S. Deb, "Cuckoo search via Lévy flights," in *Proceedings of the World Congress on Nature & Biologically Inspired Computing (NABIC '09)*, pp. 210–214, IEEE, December 2009.

[9] X. S. Yang, "A new metaheuristic bat-inspired algorithm," in *Nature Inspired Cooperative Strategies for Optimization (NICSO 2010)*, vol. 284 of *Studies in Computational Intelligence*, pp. 65–74, Springer, Berlin, Germany, 2010.

[10] X. S. Yang, *Nature-Inspired Metaheuristic Algorithms*, Luniver Press, Frome, UK, 2nd edition, 2010.

[11] Y. Chen, Z. Cui, and J. Zeng, "Structural optimization of Lennard-Jones clusters by hybrid social cognitive optimization algorithm," in *Proceedings of the 9th IEEE International Conference on Cognitive Informatics*, pp. 204–208, IEEE, July 2010.

[12] Z. Cui and X. Cai, "Optimal coverage configuration with social emotional optimisation algorithm in wireless sensor networks," *International Journal of Wireless and Mobile Computing*, vol. 5, no. 1, pp. 43–47, 2011.

[13] Z. Cui, Z. Shi, and J. Zeng, "Using social emotional optimization algorithm to direct orbits of chaotic systems," in *Swarm, Evolutionary, and Memetic Computing*, pp. 389–395, Springer, 2010.

[14] Z. W. Geem, J. H. Kim, and G. V. Loganathan, "A new heuristic optimization algorithm: harmony search," *Simulation*, vol. 76, no. 2, pp. 60–68, 2001.

[15] D. Simon, "Biogeography-based optimization," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 6, pp. 702–713, 2008.

[16] I. J. Fister, X. S. Yang, J. Brest, and D. Fister, "A brief review of nature-inspired algorithms for optimization," *Electrotechnical Review*, vol. 80, no. 3, 2013.

[17] P. Civicioglu, "Backtracking search optimization algorithm for numerical optimization problems," *Applied Mathematics and Computation*, vol. 219, no. 15, pp. 8121–8144, 2013.

[18] C. Blum, J. Puchinger, G. R. Raidl, and A. Roli, "Hybrid metaheuristics in combinatorial optimization: a survey," *Applied Soft Computing Journal*, vol. 11, no. 6, pp. 4135–4151, 2011.

[19] M. Lozano and C. García-Martínez, "Hybrid metaheuristics with evolutionary algorithms specializing in intensification and diversification: overview and progress report," *Computers & Operations Research*, vol. 37, no. 3, pp. 481–497, 2010.

[20] J. Zhang, Z.-H. Zhang, Y. Lin et al., "Evolutionary computation meets machine learning: a survey," *IEEE Computational Intelligence Magazine*, vol. 6, no. 4, pp. 68–75, 2011.

[21] K. Price, R. M. Storn, and J. A. Lampinen, *Differential Evolution: A Practical Approach to Global Optimization*, Springer, 2006.

[22] R. Storn and K. Price, Differential Evolution, 2013, http://www1.icsi.berkeley.edu/~storn/code.html.

[23] 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 Transactions on Evolutionary Computation*, vol. 15, no. 1, pp. 99–119, 2011.

[24] Z. Beheshti, S. M. Shamsuddin, and S. Sulaiman, "Fusion global-local-topology particle swarm optimization for global optimization problems," *Mathematical Problems in Engineering*, vol. 2014, Article ID 907386, 19 pages, 2014.

[25] W. Y. Gong, Z. H. Cai, and C. X. Ling, "DE/BBO: a hybrid differential evolution with biogeography-based optimization for global numerical optimization," *Soft Computing*, vol. 15, no. 4, pp. 645–665, 2011.

[26] F. Wang, X.-S. He, L. G. Luo, and Y. Wang, "Hybrid optimization algorithm of PSO and Cuckoo Search," in *Proceedings of the 2nd International Conference on Artificial Intelligence, Management Science and Electronic Commerce (AIMSEC '11)*, pp. 1172–1175, August 2011.

[27] Z.-H. Zhan, J. Zhang, Y. Li, and Y.-H. Shi, "Orthogonal learning particle swarm optimization," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 6, pp. 832–847, 2011.

[28] M. G. Epitropakis, V. P. Plagianakos, and M. N. Vrahatis, "Balancing the exploration and exploitation capabilities of the differential evolution algorithm," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '08)*, pp. 2686–2693, June 2008.

[29] W. Y. Gong and Z. H. Cai, "Differential evolution with ranking-based mutation operators," *IEEE Transactions on Cybernetics*, vol. 43, no. 6, pp. 2066–2081, 2013.

[30] W. T. Zhao, L. J. Wang, Y. L. Yin, B. Wang, Y. Wei, and Y. Yin, "An improved backtracking search algorithm for constrained optimization problems," in *Knowledge Science, Engineering and Management: 7th International Conference, KSEM 2014, Sibiu, Romania, October 16-18, 2014. Proceedings*, vol. 8793 of *Lecture Notes in Computer Science*, pp. 222–233, Springer, 2014.

[31] J. J. Liang, B. Y. Qu, P. N. Suganthan et al., "Problem definitions and evaluation criteria for the cec 2013 special session on real-parameter optimization," Tech. Rep., Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou, China; Nanyang Technological University, Singapore, 2013.

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

[33] H. Ma, "An analysis of the equilibrium of migration models for biogeography-based optimization," *Information Sciences*, vol. 180, no. 18, pp. 3444–3464, 2010.

[34] M. Clerk, 2014, http://clerc.maurice.free.fr/pso/.

[35] N. Hansen and A. Ostermeier, "Completely derandomized self-adaptation in evolution strategies," *Evolutionary Computation*, vol. 9, no. 2, pp. 159–195, 2001.

[36] C. Igel, N. Hansen, and S. Roth, "Covariance matrix adaptation for multi-objective optimization," *Evolutionary Computation*, vol. 15, no. 1, pp. 1–28, 2007.

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

[38] J. J. Liang, A. K. Qin, P. N. Suganthan, and S. Baskar, "Comprehensive learning particle swarm optimizer for global optimization of multimodal functions," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 3, pp. 281–295, 2006.

[39] A. K. Qin and P. N. Suganthan, "Self-adaptive differential evolution algorithm for numerical optimization," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '05)*, vol. 2, pp. 1785–1791, IEEE, September 2005.

[40] P. N. Suganthan, N. Hansen, J. J. Liang et al., "Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization," KanGAL Report 2005005, Nanyang Technological University, Singapore, 2005.

[41] I. Loshchilov, "CMA-ES with restarts for solving CEC 2013 benchmark problems," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '13)*, pp. 369–376, Cancún, Mexico, June 2013.

[42] F. V. Nepomuceno and A. P. Engelbrecht, "A self-adaptive heterogeneous PSO for real-parameter optimization," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '13)*, pp. 361–368, June 2013.

[43] M. Zambrano-Bigiarini, M. Clerc, and R. Rojas, "Standard particle swarm optimization 2011 at CEC-2013: a baseline for future PSO improvements," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '13)*, pp. 2337–2344, June 2013.

[44] M. El-Abd, "Testing a Particle Swarm Optimization and Artificial Bee Colony Hybrid algorithm on the CEC13 benchmarks," in *Proceedings of the IEEE Congress on Evolutionary Computation*, pp. 2215–2220, June 2013.

[45] L. Dos Santos Coelho, H. V. H. Ayala, and R. Z. Freire, "Population's variance-based Adaptive differential evolution for real parameter optimization," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '13)*, pp. 1672–1677, June 2013.

[46] P. N. Suganthan, 2015, http://www.ntu.edu.sg/home/EPNSugan/.

[47] I. Loshchilov, T. Stuetzle, and T. J. Liao, "Ranking results of CEC'13 special session & competition on real-parameter single objective optimization," [OL], 2015, http://www.ntu.edu.sg/home/EPNSugan/index_files/CEC2013/results_21.pdf.