Hindawi
Mathematical Problems in Engineering
Volume 2018, Article ID 3102628, 16 pages
https://doi.org/10.1155/2018/3102628

Research Article

An Improved Artificial Bee Colony Algorithm Based on Factor Library and Dynamic Search Balance

Wenjie Yu,[1] **Xunbo Li**,[1] **Hanbin Cai**,[1] **Zhi Zeng**,[2] **and Xiang Li**[1]

[1]*School of Mechatronics Engineering, University of Electronic Science and Technology of China, Chengdu, Sichuan 611731, China*
[2]*Institute of Electronic and Information Engineering of UESTC in Guangdong, Guangdong, China*

Correspondence should be addressed to Wenjie Yu; wenjie.y@outlook.com

Received 29 July 2017; Revised 12 December 2017; Accepted 20 December 2017; Published 28 January 2018

Academic Editor: Jose J. Muñoz

The artificial bee colony (ABC) algorithm is a relatively new optimization technique for simulating the honey bee swarms foraging behavior. Due to its simplicity and effectiveness, it has attracted much attention in recent years. However, ABC search equation is good at global search but poor at local search. Some different search equations are developed to tackle this problem, while there is no particular algorithm to substantially attain the best solution for all optimization problems. Therefore, we proposed an improved ABC with a new search equation, which incorporates the global search factor based on the optimization problem dimension and the local search factor based on the factor library (FL). Furthermore, aimed at preventing the algorithm from falling into local optima, dynamic search balance strategy is proposed and applied to replace the scout bee procedure in ABC. Thus, a hybrid, fast, and enhanced algorithm, HFEABC, is presented. In order to verify its effectiveness, some comprehensive tests among HFEABC and ABC and its variants are conducted on 21 basic benchmark functions and 20 complicated functions from CEC 2017. The experimental results show HFEABC offers better compatibility for different problems than ABC and some of its variants. The HFEABC performance is very competitive.

1. Introduction

In the fields of science and engineering, a wide variety of actual problems can be converted to optimization problems and then be settled by optimization techniques. Unfortunately, many of these problems are often characterized as nonconvex, discontinuous or nondifferentiable; thus it is extremely hard to find optimal solutions. Over the last two decades, numerous algorithms have been developed to tackle such complex problems. Among these algorithms, many are inspired by swarm behaviors, such as ant colony optimization (ACO) [1, 2], particle swarm optimization (PSO) [3], artificial bee colony algorithm (ABC) [4], cuckoo search [5], and firefly algorithm [6]. These algorithms, belonging to Swarm Intelligence (SI) [7], have been studied so far and are well employed to solve various intricate computational problems, for instance, optimization of gait pattern tuning for humanoid robots [2], objective functions [8], and relay node deployment [9].

ABC algorithm inspired by the foraging behavior of honeybee swarms is a population based metaheuristic optimization algorithm [10]. Owing to its effectiveness and simplicity, the ABC has been diffusely employed to settle both continuous and discrete optimization problems since its introduction [11]. Thus, in this approach, we focus on ABC algorithm, while, in [12–14], it has been pointed out that ABC is prone to suffer poor intensification performance on complicated problems. In other words, it is easy to encounter the problem of poor convergence. The possible reason is that the search equation employed to produce new candidate solutions has good global search ability but poor local search ability [15], and thus it leads to the problem of slow convergence speed. The global search procedure is associated with the ability of independently exploring the global optimum, while the local search procedure is associated with the ability of applying the information existing to hunt for better solutions. The global search and local search are both extremely important mechanisms in ABC. Therefore, how to further balance

and accelerate the two processes is a challenging research topic.

In recent years, some modified or improved algorithms [16] based on the foraging behavior of honey bee swarm are raised, for instance, the *gbest*-guided ABC (GABC) [15], the best-so-far ABC (BSFABC) [17], Bee Swarm Optimization (BSO) [18], qABC [19], and GBABC [20]. These ABC variants show some better features compared to the original ABC. However, there is no particular algorithm to substantially attain the best solution for all optimization problems. Some algorithms only perform better than others on some special problems. Accordingly, it is very necessary to design a well improved algorithm.

In order to solve all the problems mentioned above, a hybrid, fast, and enhanced ABC method was proposed to further improve the performance of ABC. The improved and hybrid ABC is called HFEABC. It has the following differences with respect to ABC. Initially, global adjustable factor based on the dimension of the optimization problem and the best local search factor chosen from the factor library (FL) are introduced to modify the search process. This strategy is used to enhance the performance of ABC in terms of high convergence speed, high robustness, and being more compatible with different problems, while the improvement of ABC using merely the strategy based on FL may lead the algorithm to fall into local optimum to a degree. Therefore, a dynamic search balance strategy is proposed to strengthen the global search ability in order to further balance the exploration and exploitation of the algorithm. This strategy replaces the scout bee phase in original ABC so that one key parameter *limit* in ABC is discarded. The experimental results verify that our method shows a competitive performance.

The rest of this paper is organized as follows. Section 2 gives the main related work on ABC improvements. Section 3 describes the original ABC algorithm. In Section 4, the proposed approach is described in detail. Section 5 presents and discusses the experimental results. Finally, Section 6 provides a summary of this paper.

2. Related Works

In the last decade, many researchers contributed to the improvements on ABC due to its simplicity and efficiency. As a result, there is a lot of work on proposing ABC variants. After the comparative analysis of the literatures about improved and modified ABCs, we divide them into two categories. The first one takes the line on the improvements of the solution search equation, while the other one studies the effect of the hybridization of ABC with other search methods [21].

Taking the first category, we may cite some representative works focusing mainly on the improvements of the solution search equation. In [15], Zhu and Kwong inspired by the PSO algorithm proposes a *gbest*-guided ABC algorithm, namely, GABC, which considers improving the capability of the local search based on taking the knowledge of the global best solution into the solution search equation. The experimental results show that GABC outperforms the basic ABC on some used benchmark functions. In lieu of the original solution search equation, a Lévy mutation is utilized in [22] to produce new candidate solutions in the neighborhood of the global best solution of current population. Akay and Karaboga [16] probed the effects of two elements, frequency of the perturbation and magnitude of the perturbation, on the performance of ABC. Consequently, a modified ABC algorithm is proposed, introducing two new parameters controlling both factors. Banharnsakun et al. [17] proposed an improved ABC variant based on the best-so-far solutions. In this work, the best-so-far solution-predicated method is utilized by onlooker bees to search direction. In [23], Gao et al. carried out a comparison on the performance of two different ABC variants based on, respectively, the ABC/best/1 and ABC/best/2 search equations. The experimental outcomes exhibit that ABC/best/1 appreciably outperforms ABC/best/2. Further, in [24], Gao and Liu put forward a modified ABC (MABC) algorithm by adopting the same ABC/best/1 search method. But the MABC algorithm omits the probabilistic selection method and scout bee procedure. This is different from the original ABC. Das et al. [25] put forward an ABC variant based on fitness learning and proximity stimuli (FlABCps). This variant utilizes the Rechenberg's 1/5th mutation rule and the information of the top q% food sources to generate a new solution with more than one dimension updated. In [26], Bansal et al. considered utilizing a self-adaptive step size method to well adjust the parameters used in the solution update strategy. This improved ABC is called self-adaptive ABC (SAABC), and its parameter of limit is set as adaptive. In order to enhance the local search ability of ABC, Karaboga and Gorkemli [19] put forward a quick ABC (qABC) algorithm, in which the behavior of the onlookers is changed to just search the neighborhood food source. Li and Yin [27] proposed a self-adaptive modification rate to enhance the convergence rate of the ABC. Gao et al. [8] developed two new search equations in the employed bee phase and the onlookers phase, respectively, to balance the exploration and exploitation in the ABC. Further, in Zhou et al. [20], to balance the global search ability and the local search ability and remedy the "oscillation" phenomenon, the candidate solution is produced by utilizing the Gaussian distribution based on the global best solution. In Sharma et al. [28], Lévy Flight ABC (LFABC) is proposed, where the candidate solution is generated around the best solution by tuning the Lévy flight parameters, thereby tuning the step sizes, to enhance the local search capability.

The second category takes the line of hybridization. Kang et al. [29] proposed an improved ABC algorithm based on Rosenbrock, in which the Rosenbrock method is developed for multimodal optimization problems. In [30], the Lévy flight random walk was introduced into ABC to perform an additional local search. In order to find out more beneficial information from the search experiences for ABC, Gao et al. [21] employed the orthogonal experimental method to compose an orthogonal learning strategy. In the paper by Wang et al. [31], they utilize a pool of distinct solution search strategies coexisting throughout the search process and competing to produce offspring. In the paper by Aydin [32], he conducted a systematic experimental study to the

proposed modifications of a few ABC variants to evaluate their impact on algorithm performance, and based on these analyses, two new variants of ABC, using the best schemes tested in the experiment, are developed. To efficiently solve optimization problems with different characteristics, Kiran et al. [33] proposed the integration of multiple solution update rules with ABC, while Yurtkuran and Emel [34] used a random select strategy to select one solution search strategy from a variety of search strategies to balance the global search ability and the local search ability. Ma et al. [35] introduced a modified ABC algorithm, which utilizes the life cycle scheme to generate dynamical varying population and ensure proper balance between the global search and the local search.

3. The ABC Algorithm

The ABC algorithm is a population-predicated metaheuristic algorithm that simulates the foraging behavior of honey bee swarms. This technique is very easy to implement and effective. There are three groups of foraging bees in the ABC: employed bees, onlooker bees, and scout bees. The number of employed bees is the same as onlooker bees, being half of the colony. The responsibility of employed bees is to exploit the food sources and then onlooker bees decide whether to exploit the food sources or not according to the information shared by the employed bees. Scout bees try to find a new food source through random searching. One food source in ABC stands for a possible solution to the optimization problem. The amount of nectar on the food source represents the quality of this solution. The number of the food sources and the number of employed bees are the same. When the quality of a food source remains unchanged for a determinate time, the employed bee exploiting this food source will transform to a scout bee. And once the scout bee finds a new food source, it transforms back to an employed bee.

The following is the main procedure of the ABC.

(1) Initialization
(2) Assess the population
(3) Loop
(4) Employed bee process \par
(5) Onlooker bee process \par
(6) Scout bee process \par
(7) Memorize the best food source \par
(8) Until (the termination criteria is satisfied).

In the procedure of initialization, the ABC produces a randomly distributed population of SN solutions (food sources), where SN is half of the colony size and also represents the number of employed or onlooker bees. Let $x_i = \{x_{i1}, x_{i2}, \ldots, x_{iD}\}$ represent the ith food source, where D is the problem size. Each food source is produced within the limited range of jth index by

$$x_{ij} = x_j^{\min} + \lambda \left(x_j^{\max} - x_j^{\min}\right), \tag{1}$$

where $i = 1, 2, \ldots, \mathrm{SN}$ and $j = 1, 2, \ldots, D$, $x_j^{\min}$ and $x_j^{\max}$ are the lower and upper bounds for the index j, respectively, and λ is a random real number within the range $[0, 1]$.

In the procedure of employed bees, the candidate solution v_i is produced by performing a local search around a neighboring food source. The equation is as follows:

$$v_{ij} = x_{ij} + \phi_{ij}\left(x_{ij} - x_{kj}\right), \tag{2}$$

where j is a randomly selected dimension such that $j \in \{1, 2, \ldots, D\}$ and k is a randomly chosen food source such that $k \in \{1, 2, \ldots, \mathrm{SN}\}$ and $k \neq i$. ϕ_{ij} is produced randomly in the range $[-1, 1]$. Then comparing the fitness of x_i and v_i, the one with greater fitness is kept and the employed bees will come back to hive to share the information on new food sources with the onlookers. The fitness is calculated as follows:

$$\mathrm{fit}_i = \begin{cases} \dfrac{1}{1 + f_i}, & f_i \geq 0 \\ 1 + \mathrm{abs}\left(f_i\right), & f_i < 0, \end{cases} \tag{3}$$

where fit_i represents the fitness of solution i, f_i is the result of objective function, and $i \in \{1, 2, \ldots, \mathrm{SN}\}$.

In the procedure of onlooker bees, the food source is chosen relying on the probability value p, and the calculation method of p might be given as follows:

$$p_i = 0.9 \times \frac{\mathrm{fit}_i}{\max\left(\mathrm{fit}_i\right)} + 0.1. \tag{4}$$

Via utilizing this scheme, the food sources with greater fitness value are more possible to be chosen by onlookers for update. Once the onlooker selects the food source, it produces a candidate solution utilizing (2). Then, the selection procedure in the employed bee phase is conducted on x_i and v_i. The one with greater fitness value is kept.

4. Proposed Algorithm: HFEABC

In this section, the algorithm proposed is introduced in detail. First, it is presented that a new search equation combines a dynamic adaptive global search part based on the dimension of the problem and the local search part based on a factor library. Second, in order to prevent the algorithm from falling into local optima, the dynamic search balance strategy is proposed instead of scout bee phase in original ABC.

4.1. A New Search Equation Based on Factor Library. As we all know, for any metaheuristic algorithm, the balance between the global search and the local search is one of the most critical mechanisms. The global search means the capability of searching for global optimal solution in entire solution space. And the local search means the capability of employing the information of previous solutions to find better solutions. However, the procedures of the global search and the local search against each other must be well adjusted to achieve desired optimization results. In ABC, because a new solution is produced utilizing the knowledge of the previous food source with the guidance of the term $\phi_{ij}(x_{ij} - x_{kj})$ in (2) and ϕ_{ij} is a random number within $[-1, 1]$, there is no guarantee that a better individual influences the candidate solution. This search equation has a good global search ability but it ignores the local search ability. This may lead to a poor convergence

speed and intensification performance. To overcome such issues, in literature [15], they proposed GABC with a new searching strategy:

$$v_{ij} = x_{ij} + \phi_{ij}\left(x_{ij} - x_{kj}\right) + \varphi_{ij}\left(y_j - x_{ij}\right), \quad (5)$$

where k, j, and ϕ_{ij} are yielded in the same manner as in (2), φ_{ij} is a uniform random number within $[0, C]$, where C is a nonnegative constant, and y_j is the jth element of the global best solution. This new search equation somehow improves the local search ability without harming the global search ability and the experimental results show that it is superior to the original one on some test functions. However, the scheme of employing *gbest* still brings some inefficiency to the search capability of the algorithm and decelerates convergence speed [21]. In order to analyze this in detail, we rewrite (5) as

$$\begin{aligned} v_{ij} &= x_{ij} + v_{ij}^{(1)} + v_{ij}^{(2)} \\ v_{ij}^{(1)} &= \phi_{ij}\left(x_{ij} - x_{kj}\right) \\ v_{ij}^{(2)} &= \varphi_{ij}\left(y_j - x_{ij}\right). \end{aligned} \quad (6)$$

In (6), it is clear that $v_{ij}^{(1)}$ stands for exploration and $v_{ij}^{(2)}$ stands for exploitation. It is important to note that this algorithm assumes a random local search ability throughout the optimization process. However, according to general optimization procedure, global search is more important than local search in early stage in order to avoid trapping in local optimum. In late stage of optimization process, local search becomes more important than global search, because better local search ability means higher speed of convergence and leads to more accurate result. In addition, although a few different search equations are proposed in [17, 36–39], each algorithm only supplies a better solution for some specific problems than others. Therefore, it is very necessary to seek for a well improved method to be compatible for different problems. To this end, we consider redesigning this search equation to get a good balance between exploration and exploitation with higher speed of convergence and moreover it will be suited for different problems.

First, we consider reducing the ability of global search ability to accelerate the speed of convergence to a degree; therefore we modify $v_{ij}^{(1)}$ as

$$v_{ij}^{(1)} = \phi_{ij} \cdot f_g \cdot \left(x_{ij} - x_{kj}\right), \quad (7)$$

where f_g is the parameter to control the global search ability; it is defined as

$$f_g = (1 - l)^{1/D}, \quad (8)$$

where D is the size of the problem and l is a linear parameter, being defined as

$$l = \frac{Curcycle}{\text{Maxcycle}}, \quad (9)$$

where *Curcycle* is the current number of iterations and Maxcycle is the maximum number of iterations.

It can be noted that the global search ability will decrease as l increases, but the higher the dimension of the problem, the stronger the global search ability of the algorithm.

Second, with the objective of strengthening robustness and further improving the convergence speed of the algorithm, $v_{ij}^{(2)}$ is developed as

$$v_{ij}^{(2)} = C \cdot f_{\text{best}}^{l} \cdot \left(y_j - x_{ij}\right), \quad (10)$$

where C is a positive constant as the description of [15] and f_{best}^{l} is the value of the best factor chosen from the FL. It is noticed that the most appropriate factor for different problem is different. Therefore, we propose the concept of FL. FL includes the following factors: $\log_{10}(l+1)/\log_{10}(2)$, $l^{0.5}$, $l^{0.75}$, $\ln(l+1)/\ln(2)$, $2^l - 1$, $((e^2+1)/(e^2-1)) \cdot ((e^{2l}-1)/(e^{2l}+1))$, and $\ln(l+\sqrt{l^2+1})/\ln(1+\sqrt{2})$. These factors contained in FL are derived by experimenter and are detailed in Section 5.3. Finally, the search equation is given as

$$v_{ij} = x_{ij} + \phi_{ij} \cdot f_g \cdot \left(x_{ij} - x_{kj}\right) + C \cdot f_{\text{best}}^{l} \cdot \left(y_j - x_{ij}\right). \quad (11)$$

4.2. Dynamic Search Balance Strategy. In real world, problems are complex. It cannot be figured out when an algorithm will find the best solutions to the problems. Aimed at speeding the convergence of the algorithm and obtaining a more accurate result, in (11), the global search ability is reduced somehow while the local search ability is strengthened. However, this may lead to the phenomenon of prematurity. To improve this situation, the dynamic search balance strategy is proposed, which can provide good global search ability. This strategy replaces the scout bee procedure in ABC. In this strategy, the position of global solution is monitored. If *GlobalMin* is updated, set *GlobalSearch* to false and (11) will be used to generate the candidate solution. If *GlobalMin* is not updated, then the swarm of bees (the onlooker bees and the employed bees) chooses the search equation proposed in ABC, which has a strong global search ability [15]. Then, all the solutions are sorted. One solution will be selected with a certain probability and processed with GOBL strategy [20]. The main idea behind GOBL is that when a candidate solution S to a given problem is evaluated, simultaneously computing its opposite solution $\overline{S}$ can provide a higher chance for $\overline{S}$ to be closer to the global optimum than a random generated solution. It is beneficial to preserve the search experiences for the efficiency of the algorithm. One more important point in our approach is that the key parameter *limit* in ABC is eliminated. The process of search equation chosen is given in Algorithm 1.

4.3. Pseudocode of HFEABC. Compared with the original ABC, HFEABC makes three modifications. First, in the employed and onlooker bee procedures, the original search equation is displaced by the new designed one. Second, a dynamic search balance strategy is employed to enhance the global search ability of this algorithm. Third, the scout bee procedure in the original ABC is replaced by dynamic search balance strategy. The pseudocode of HFEABC is depicted in Algorithm 2.

Input a new solution set P_{new}.
(1) **if** the *GlobalMin* is updated **then**
(2) *GlobalSearch* = false
(3) **else**
(4) *GlobalSearch* = true
(5) Sort all the solutions according to the fitness values in ascending order. Each solution gets its rank order k.
(6) Calculate a selection range $[k(k-1)/\text{PS}(\text{PS}+1), k(k+1)/\text{PS}(\text{PS}+1))$ for each solution s_k.
(7) Generate a random real number r.
(8) **if** r in the selection range of any solution s_a **then**
(9) Do GOBL on solution s_a.
(10) **end if**
(11) **end if**

Algorithm 1: Dynamic search balance.

(1) Parameter initialization (PS = 2SN, Maxcycle = Maximum number of iterations, *GlobalSearch* = false)
(2) Produce initial population by Eq. (1).
(3) Assess initial population // Calculate $f(x)$ and memorize the global best.
(4) Set *Curcycle* = 1.
(5) **for** each food source i do, set $trial_i = 0$ **end for**
(6) **do while** *Curcycle* ≤ Maxcycle
(7) /∗EMPLOYED BEE PROCEDURE∗/
(8) **for** each i **do**
(9) **if** *GlobalSearch* == false **then**
(10) Produce v_i by Eq. (11).
(11) **else**
(12) Produce v_i by Eq. (2).
(13) **end if**
(14) Evaluate v_i, if it is better, replace the original one.
(15) **end for**
(16) /∗ONLOOKER BEE PROCEDURE∗/
(17) Calculate probability values p_i for all i by Eq. (4).
(18) Set $t = 0$, $i = 1$
(19) **while** $t <$ PS **do**
(20) Produce random number $r \in [0, 1]$
(21) **if** $r < p_i$ **then**
(22) $t = t + 1$
(23) **if** *GlobalSearch* == false **then**
(24) Produce v_i by Eq. (11).
(25) **else**
(26) Produce v_i by Eq. (2).
(27) **end if**
(28) Evaluate v_i, if it is better, replace the original one.
(29) **end if**
(30) Set $i = i + 1$.
(31) **if** $i >$ PS **then** set $i = 1$ **end if**
(32) **end while**
(33) Do dynamic search balance as depicted in Algorithm 1.
(34) Memorize the best *GlobalMin*.
(35) Set *Curcycle* = *Curcycle* + 1.
(36) **end while**

Algorithm 2: HFEABC algorithm.

5. Experiments and Discussion

In this section, the performance of HFEABC (H) is compared with other famous ABC variants, which involves the standard ABC (A1) [4], AABCLS (A2) [38], BSFABC (A3) [17], GABC (A4) [15], GABCS (A5) [40], GBABC (A6) [20], HGABC (A7) [41], IABC (A8) [36], and LABCSS (A9) [37]. The functions used to test these algorithms are described in Section 5.1. The parameters' configuration is given in Section 5.2. Section 5.3 analyzes the best factors chosen to process in this approach. The numerical comparison results are discussed in Sections 5.4 and 5.5.

5.1. Benchmark Functions. The performance of the proposed scheme is verified through minimizing 21 basic benchmark functions and the hybrid and composite functions from CEC 2017 [42] in this subsection.

The basic benchmark functions contain a set of 19 scalable benchmark functions of dimensions $D = 30$ or 60 and a set of 2 functions of dimensions $D = 100$ and 200, as shown in Table 1. In this table, the first column represents the function names. Functions' mathematical expressions are given in the second column, in which the vector X is $\{x_1, x_2, \ldots x_i, \ldots, x_D\}$. The third column gives the search range of the vector X, in which each variable has the same search range. These benchmark functions are widely employed to test the performance of global optimization algorithms [15, 43–45]. Among them, functions f_2, f_4, f_{10}, f_{11}, f_{13}, f_{14}, f_{15}, and f_{19} are taken from CEC2017 [46]. For simplicity, based on different dimensions, these functions are divided into the low- and high-dimensional functions. For instance, the low-dimensional functions include 30-dimensional functions f_1–f_{19} and 100-dimensional functions f_{20}-f_{21}. The high-dimensional functions include 60-dimensional functions f_1–f_{19} and 200-dimensional functions f_{20}-f_{21}. In addition, the functions in Table 1 are classified into unimodal and multimodal functions. Unimodal functions have one local minimum as the global optimum. These functions are generally used to test the intensification ability of algorithms. Multimodal functions have one or more local optimums which may be the global optimum. f_1–f_6 and f_8 are continuous unimodal functions. f_7 is a discontinuous step function, and f_9 is a noisy quartic function. f_{10}–f_{21} are multimodal and the number of their local minima increases exponentially with the problem dimension. Moreover, f_{14} is a bound constrained function.

In addition, the hybrid and composite functions from CEC 2017 are included in this experiment to further testify the proposed algorithm. These functions are also tested in low dimension $D = 30$ and high dimension $D = 60$, respectively.

5.2. Parameter Settings. For an equitable comparison among ABCs, they are tested by using the same configurations of the parameters; that is, the number of food sources SN is set to 20. The other parameters of test algorithms are set to their original values given in their corresponding papers. The parameter C is set to 1.5 as GABC proposed. All experiments were run for 2,500 and 5,000 iterations, respectively, in the case of D = low and high or until the function error dropped below $e - 20$ (values less than $e - 20$ were reported as 0). Each of the experiments was repeated 30 times independently.

5.3. The Analysis of the Best Factors Selected for Search Equation. Inspired by [36], we consider taking more different factors to bring ABC better performance for different problems. Therefore, we assembled different elementary functions into different factors. All tested factors in this experiment include $\log_{10}(l+1)/\log_{10}(2)$(logl), $l^{0.01}$(pdot01), $l^{0.1}$(pdot1), $l^{0.2}$(pdot2), $l^{0.25}$(pdot25), $l^{0.5}$(pdot5), $l^{0.75}$(pdot75), l(p1dot0), $l^{1.5}$(p1dot5), l^2(p2dot0), l^3(p3dot0), l^5(p5dot0), l^{10}(p10dot0), $\ln(l+1)/\ln(2)$(lnl), $2^l - 1$(exp 2), $((3^l - 1)/2)$(exp 3), $((4^l - 1)/3)$(exp 4), $((e^l - 1)/(e - 1))$(expe), $(\sin(l)/\sin(1))$(sl), $((e^l - e^{-l})/(e - e^{-1}))$(shxl), $((\cos(l) - 1)/(\cos(1) - 1))$(cl), $\arctan(l)/\arctan(1)$(arl), $((e^l+e^{-l}-2)/(e+e^{-l}-2))$(chxl), $(((e^2+1)/(e^2-1))\cdot((e^{2l}-1)/(e^{2l}+1)))$(thxl), and $(\ln(l+\sqrt{l^2+1})/\ln(1+\sqrt{2}))$(arshxl). It should be noted that the text in parentheses is the name of the factor. In each iteration, the HFEABC algorithm just takes one factor. In order to select the best factors for HFEABC, the Friedman test is conducted to obtain average rankings according to the suggestion in [47, 48]. Table 2 shows the average rankings (AR) of HFEABC with different single factor in all iterations to 21 basic benchmark functions mentioned in Section 5.1. In the procedure of factors chosen, it is found that more factors are included in this algorithm, with better performance but higher computational complexity and longer calculation time, while including too few factors into the algorithm means smaller computational complexity and shorter calculation time but poor performance. Thus, to get a balance, the factors with the average rankings smaller than 10 are selected into the Factor Library of HFEABC.

After combining some different factors into the HFEABC, this algorithm is tested on basic benchmark functions to observe the effect of the best factor chosen. The results are depicted in Figure 1. It can be seen from Figure 1 that the choice of factors for different problems is different and the combination of factors in high-dimensional functions and low-dimensional functions tests shows similarities.

5.4. Numerical Results and Comparisons on Basic Benchmark Functions

(1) The Comparison of HFEABC and ABC Variants. This subsection presents a comparative study of HFEABC with ABC and its variants at both $D = 30$ and 60. The values shown in Tables 3 and 4 are medians of the proposed algorithm and other algorithms in this approach. The best values are emphasized in boldface. For the low-dimensional functions testing results shown in Table 3, it is clear that HFEABC performs significantly better than the other algorithms on the majority of test functions. To be specific, it is noted that only the optimization results of A3 (BSFABC) on f_4, f_8, and f_{13} are a little better than HFEABC, but the difference between A3 (BSFABC) and H (HFEABC) on f_4, f_8, and f_{13} is not significant according to comparison results of Wilcoxon test. For high-dimensional functions testing results listed in Table 4, the HFEABC algorithm outperforms the other

Table 1: Benchmark functions used in experiments.

Function name	Function mathematical expressions	Search range
Shpere	$f_1(X) = \sum_{i=1}^{D} x_i^2$	$[-100, 100]^D$
Elliptic	$f_2(X) = \sum_{i=1}^{D} \left(10^6\right)^{(i-1)/(D-1)} x^2$	$[-100, 100]^D$
SumSquare	$f_3(X) = \sum_{i=1}^{D} i x_i^2$	$[-1, 1]^D$
SumPower	$f_4(X) = \sum_{i=1}^{D} \lvert x_i \rvert^{(i+1)}$	$[-10, 10]^D$
Schwefel 2.22	$f_5(X) = \sum_{i=1}^{D} \lvert x_i \rvert + \prod_{i=1}^{D} \lvert x_i \rvert$	$[-10, 10]^D$
Schwefel 2.21	$f_6(X) = \max\{\lvert x_i \rvert, 1 \le i \le D\}$	$[-100, 100]^D$
Step	$f_7(X) = \sum_{i=1}^{D} (\lfloor x_i + 0.5 \rfloor)^2$	$[-100, 100]^D$
Exponential	$f_8(X) = \exp\left(0.5 * \sum_{i=1}^{D} x_i\right) - 1$	$[-1.28, 1.28]^D$
Quartic	$f_9(X) = \sum_{i=1}^{D} i x_i^4 + \text{random}[0, 1)$	$[-1.28, 1.28]^D$
Rosenbrock	$f_{10}(X) = \sum_{i=1}^{D} \left[100\left(x_{i+1} - x_i^2\right)^2 + \left(x_i - 1\right)^2\right]$	$[-5, 10]^D$
Rastrigin	$f_{11}(X) = \sum_{i=1}^{D} \left[x_i - 10\cos\left(2\pi x_i\right) + 10\right]$	$[-5.12, 5.12]^D$
NCRastrigin	$f_{12}(X) = \sum_{i=1}^{D} \left[y_i - 10\cos\left(2\pi y_i\right) + 10\right],\ y_i = \begin{cases} x_i & \lvert x_i \rvert < \frac{1}{2} \\ \frac{\text{round}\left(2x_i\right)}{2} & \lvert x_i \rvert \ge \frac{1}{2} \end{cases}$	$[-5.12, 5.12]^D$
Griewank	$f_{13}(X) = \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$[-600, 600]^D$
Schwefel 2.26	$f_{14}(X) = 418.98288727243369 * D - \sum_{i=1}^{D} x_i \sin\left(\sqrt{\lvert x_i \rvert}\right)$	$[-500, 500]^D$
Ackley	$f_{15}(X) = 20 + e - 20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{D} x_i^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D} \cos\left(2\pi x_i\right)\right)$	$[-32, 32]^D$
Penalized 1	$f_{16}(X) = \frac{\pi}{D}\left\{10\sin^2\left(\pi y_1\right) + \sum_{i=1}^{D-1} \left(y_i - 1\right)^2 \left[1 + 10\sin^2\left(\pi y_{i+1}\right)\right] + \left(y_D - 1\right)^2\right\} + \sum_{i=1}^{D} u\left(x_i, 10, 100, 4\right)$ $y_i = 1 + \frac{1}{4}(x_i + 1)$ $u_{x_i,a,k,m} = \begin{cases} k\left(x_i - a\right)^m & x_i > a \\ 0 & a \le x_i \le a \\ k\left(-x_i - a\right)^m & x_i < -a \end{cases}$	$[-50, 50]^D$
Penalized 2	$f_{17}(X) = \frac{1}{10}\left\{\sin^2\left(\pi x_1\right) + \sum_{i=1}^{D-1} \left(x_i - 1\right)^2 \left[1 + \sin^2\left(3\pi x_{i+1}\right)\right] + \left(x_D - 1\right)^2 \left[1 + \sin^2\left(2\pi x_{i+1}\right)\right]\right\}$ $+\sum_{i=1}^{D} u\left(x_i, 5, 100, 4\right)$	$[-50, 50]^D$
Alpine	$f_{18}(X) = \sum_{i=1}^{D} \lvert x_i \sin\left(x_i\right) + 0.1 x_i \rvert$	$[-10, 10]^D$
Levy	$f_{19}(X) = \sum_{i=1}^{D-1} \left(x_i - 1\right)^2 \left[1 + \sin^2\left(3\pi x_{i+1}\right)\right] + \sin^2\left(3\pi x_1\right) + \lvert x_D - 1 \rvert \left[1 + \sin^2\left(3\pi x_D\right)\right]$	$[-10, 10]^D$
Himmelblau	$f_{20}(X) = \frac{1}{D}\sum_{i=1}^{D} \left(x_i^4 - 16x_i^2 + 5x_i\right)$	$[-5, 5]^D$
Michalewicz	$f_{21}(X) = -\sum_{i=1}^{D} \sin\left(x_i\right) \sin^{20}\left(\frac{i \times x_i^2}{\pi}\right)$	$[0, \pi]^D$

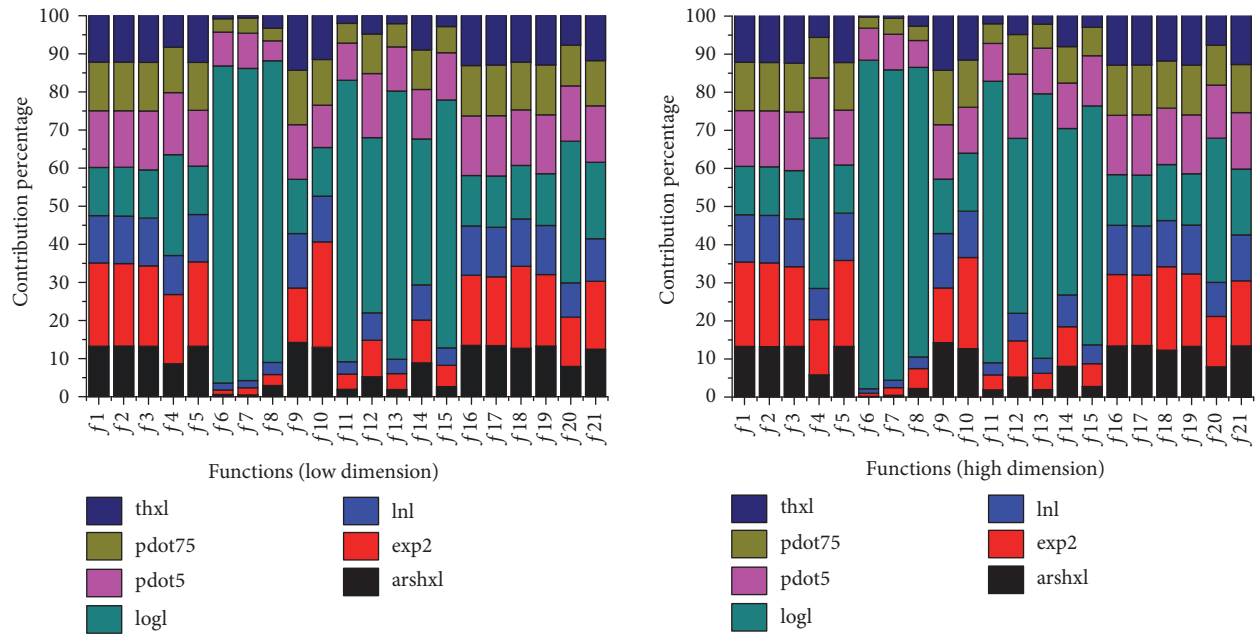

Figure 1: The ratio of being selected of the updated rules in the proposed HFEABC algorithm.

Table 2: The average rankings of factors.

Factor	AR
logl	9.16
pdot5	9.48
pdot75	9.61
lnl	9.74
exp2	9.83
thxl	9.88
arshxl	9.94
arl	10.1
p1dot0	10.17
expe	10.17
exp3	11.10
sl	11.30
shxl	11.60
exp4	12.47
p1dot5	12.49
pdot25	14.11
cl	14.40
p2dot0	15.00
chxl	15.52
pdot2	16.10
p3dot0	16.32
pdot1	18.01
pdot01	19.43
p5dot0	19.43
p10dot0	19.68

algorithms on all test functions excluding A3 (BSFABC) on f_4 and f_{13}, A5 (GABCS) on f_6, and A6 (GBABC) on f_{13}. According to the Friedman test, HFEABC ranks first on these functions for both low- and high-dimensional functions. It can be seen from the rank order at the bottom of Tables 3 and 4.

Moreover, to compare the significance between HFEABC and other ABC variants, the paired Wilcoxon signed-rank test on median results is applied. The Wilcoxon signed-rank test is a nonparametric statistical hypothesis test, which can be used as an alternative to the paired t-test when the results cannot be supposed to be normally distributed [47, 48]. The comparison results are shown in Table 5. In this table, "+", "−," and "=" indicate that our approach is, respectively, better than, worse than, and similar to that of its competitor according to the Wilcoxon signed-ranked test at $\alpha = 0.05$. The competition results are summarized as "Win/Loss/Draw," which indicates our approach wins on Win functions, loses on Loss functions, and ties on Draw functions, compared with its competitor. In Table 5, it is clear that the performance of HFEABC is significantly better than the ABC variants on low- and high-dimensional functions; moreover the Wilcoxon test finds significant difference between HFEABC and ABC variants on low- and high-dimensional functions. Therefore, it can be said that HFEABC is the best performing algorithm compared to ABC and its variants in terms of medians test.

(2) The Comparisons of Robustness Utilizing the Analysis of Variance Test. In order to further investigate the efficacy and robustness of the proposed HFEABC, the analysis of variance (ANOVA) test is also employed to determine the statistical characteristics of each tested algorithm. The box plots depicted in Figure 2 demonstrate the statistical performance representation of all algorithms on some basic benchmark functions test. Looking at these box plots, the general features

Table 3: Comparison results of ABC variants with HFEABC on 30-dimensional functions (for functions Himmelblau and Michalewicz, $D = 100$).

Func.	A1	A2	A3	A4	A5	A6	A7	A8	A9	H
f_1	7.42$e-16$	5.28$e-16$	4.52$e-16$	5.18$e-16$	7.80$e-07$	7.74$e-16$	5.28$e-16$	7.23$e-16$	1.19$e-15$	**2.97e − 16**
f_2	7.54$e-16$	6.91$e-16$	4.80$e-16$	4.92$e-16$	8.15$e+00$	1.19$e-15$	5.14$e-16$	7.51$e-16$	1.21$e-15$	**2.08e − 16**
f_3	7.64$e-16$	5.26$e-16$	4.17$e-16$	5.20$e-16$	5.89$e-09$	2.38$e-15$	5.41$e-16$	7.03$e-16$	8.91$e-16$	**3.14e − 16**
f_4	4.38$e-16$	2.37$e-14$	**4.91e − 19**	3.11$e-17$	5.10$e-05$	7.88$e-14$	1.53$e-17$	5.95$e-17$	2.27$e-15$	2.28$e-18$
f_5	1.64$e-15$	1.56$e-15$	6.69$e-15$	1.58$e-15$	3.56$e-05$	2.60$e-08$	1.40$e-15$	1.74$e-15$	7.17$e-12$	**9.00e − 16**
f_6	27.084	21.302	9.498	8.925	0.552	2.786	23.519	14.462	34.882	**0.106**
f_7	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
f_8	−0.9999	−0.9999	**−1.00e + 00**	−0.9999	−0.9999	−0.9999	−0.9999	−0.9999	−0.9999	−0.9999
f_9	0.187	0.125	2.86$e-02$	8.28$e-02$	0.383	0.132	7.10$e-02$	0.131	0.144	**2.48e − 02**
f_{10}	1.31$e+00$	7.36$e+00$	4.27$e+00$	4.18$e+00$	8.46$e+01$	2.61$e+01$	1.05$e+00$	1.57$e+00$	5.86$e+00$	**2.89e − 02**
f_{11}	2.31$e-14$	6.48$e-07$	1.08$e+00$	**0.00e + 00**	1.10$e-05$	3.96$e-03$	0.00$e+00$	0.00$e+00$	1.51$e-08$	**0.00e + 00**
f_{12}	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
f_{13}	8.88$e-16$	4.44$e-16$	**3.33e − 16**	5.55$e-16$	2.74$e-07$	4.44$e-16$	5.55$e-16$	5.55$e-16$	8.88$e-16$	4.44$e-16$
f_{14}	118.44	236.99	496.37	236.88	174.31	502.18	473.75	118.44	357.53	**3.82e − 04**
f_{15}	7.11$e-14$	4.09$e-14$	9.53$e-11$	4.26$e-14$	1.02$e-05$	4.71$e-09$	**3.20e − 14**	5.33$e-14$	4.81$e-11$	**3.20e − 14**
f_{16}	7.67$e-16$	5.45$e-16$	8.64$e-16$	4.84$e-16$	5.01$e-03$	2.34$e-09$	5.42$e-16$	7.02$e-16$	7.74$e-16$	**1.83e − 16**
f_{17}	7.74$e-16$	4.90$e-16$	7.58$e-16$	5.28$e-16$	1.70$e-02$	2.02$e-14$	5.11$e-16$	7.24$e-16$	9.45$e-16$	**1.88e − 16**
f_{18}	6.34$e-06$	4.73$e-06$	6.04$e-04$	3.51$e-08$	3.53$e-03$	8.18$e-06$	4.35$e-07$	1.59$e-05$	5.54$e-05$	**5.19e − 16**
f_{19}	6.87$e-14$	1.15$e-14$	1.96$e-15$	5.30$e-16$	3.24$e-02$	4.72$e-11$	5.17$e-16$	9.34$e-16$	3.55$e-14$	**1.75e − 16**
f_{20}	−78.048	−78.050	−76.325	−78.050	−74.020	−74.932	−78.050	−78.183	−77.550	**−7.83e + 01**
f_{21}	−91.927	−97.303	−82.105	−93.865	−77.963	−79.562	−93.882	−91.751	−87.802	**−9.95e + 01**
Rank	7	4	6	3	10	9	2	5	8	1

Table 4: Comparison results of ABC variants with HFEABC on 60-dimensional functions (for functions Himmelblau and Michalewicz, $D = 200$).

Func.	A1	A2	A3	A4	A5	A6	A7	A8	A9	H
f_1	1.59$e-15$	1.15$e-15$	1.18$e-15$	1.20$e-15$	4.13$e-10$	1.73$e-15$	1.39$e-15$	1.61$e-15$	2.02$e-15$	**7.26e − 16**
f_2	1.86$e-15$	1.12$e-15$	1.19$e-15$	1.17$e-15$	1.47$e+01$	2.56$e-15$	1.28$e-15$	1.44$e-15$	2.06$e-15$	**5.12e − 16**
f_3	1.65$e-15$	1.12$e-15$	1.11$e-15$	1.19$e-15$	1.41$e-14$	1.87$e-14$	1.39$e-15$	1.50$e-15$	1.84$e-15$	**7.59e − 16**
f_4	2.86$e-16$	4.34$e-15$	**9.51e − 19**	3.28$e-17$	1.75$e-04$	7.65$e-14$	2.48$e-17$	7.45$e-17$	7.10$e-16$	5.88$e-18$
f_5	3.43$e-15$	2.90$e-15$	8.80$e-14$	3.18$e-15$	1.68$e-07$	7.40$e-08$	3.15$e-15$	3.40$e-15$	2.99$e-11$	**1.85e − 15**
f_6	48.23	40.09	26.53	26.26	3.26	22.59	44.98	31.69	62.28	**4.01**
f_7	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
f_8	−0.9999	**−1.00e + 00**	**−1.00e + 00**	−0.9999	−0.9999	−0.9999	**−1.00e + 00**	−0.9999	−0.9999	**−1.00e + 00**
f_9	0.371	0.277	0.129	0.160	0.176	0.380	0.159	0.284	0.295	**0.066**
f_{10}	1.74	4.96	10.22	7.60	115.16	57.87	4.50	5.38	11.68	**9.61e − 02**
f_{11}	1.66$e-12$	1.61$e-07$	3.9849	5.95$e-14$	0.00$e+00$	1.4603	0.9950	1.78$e-15$	0.9950	**0.00e + 00**
f_{12}	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
f_{13}	1.89$e-15$	1.22$e-15$	**9.44e − 16**	1.11$e-15$	4.47$e-11$	5.55$e-16$	1.44$e-15$	1.44$e-15$	1.89$e-15$	9.99$e-16$
f_{14}	161.32	538.25	1198.29	592.24	332.88	1185.60	819.20	359.78	974.02	**7.64e − 04**
f_{15}	1.72$e-13$	9.24$e-14$	6.46$e-10$	9.24$e-14$	5.99$e-08$	8.04$e-09$	6.93$e-14$	1.12$e-13$	9.05$e-11$	**6.75e − 14**
f_{16}	1.63$e-15$	1.20$e-15$	1.63$e-15$	1.20$e-15$	2.59$e-03$	1.39$e-03$	1.39$e-15$	1.61$e-15$	1.61$e-15$	**4.35e − 16**
f_{17}	1.63$e-15$	1.06$e-15$	1.58$e-15$	1.20$e-15$	5.84$e-03$	8.98$e-14$	1.40$e-15$	1.58$e-15$	1.65$e-15$	**4.67e − 16**
f_{18}	9.67$e-05$	4.64$e-06$	1.84$e-02$	5.28$e-06$	9.22$e-04$	4.60$e-05$	1.70$e-05$	1.57$e-04$	1.12$e-04$	**1.22e − 15**
f_{19}	2.59$e-14$	1.40$e-13$	2.41$e-15$	1.19$e-15$	2.41$e-02$	6.43$e-11$	1.18$e-15$	1.71$e-15$	2.67$e-14$	**4.09e − 16**
f_{20}	−77.985	−78.050	−76.047	−78.050	−73.676	−74.611	−78.050	−78.109	−77.194	**−78.332**
f_{21}	−182.913	−192.964	−159.554	−186.562	−154.778	−154.897	−186.849	−182.445	−174.026	**−199.311**
Rank	7	3	6	2	10	9	4	5	8	1

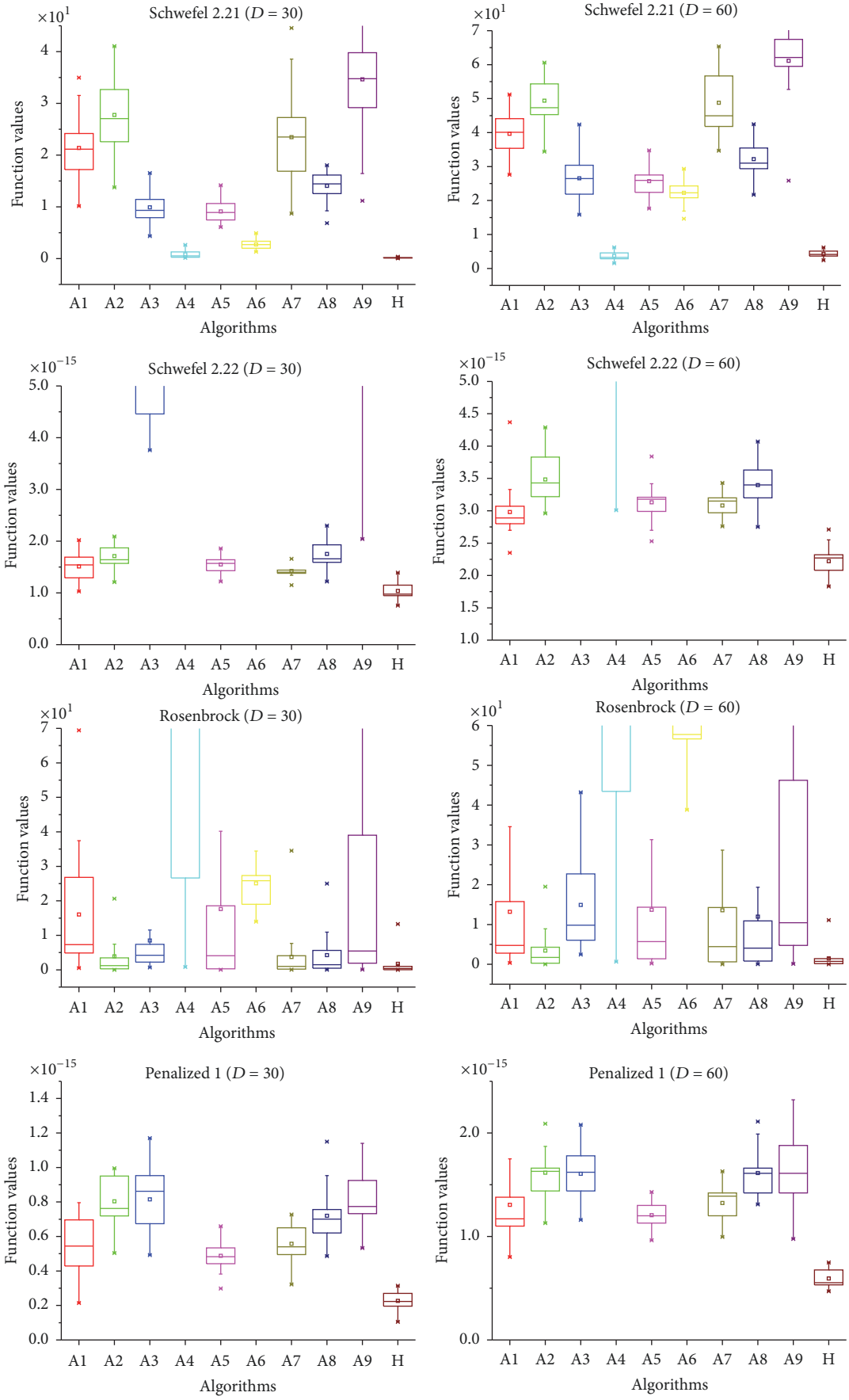

Figure 2: Continued.

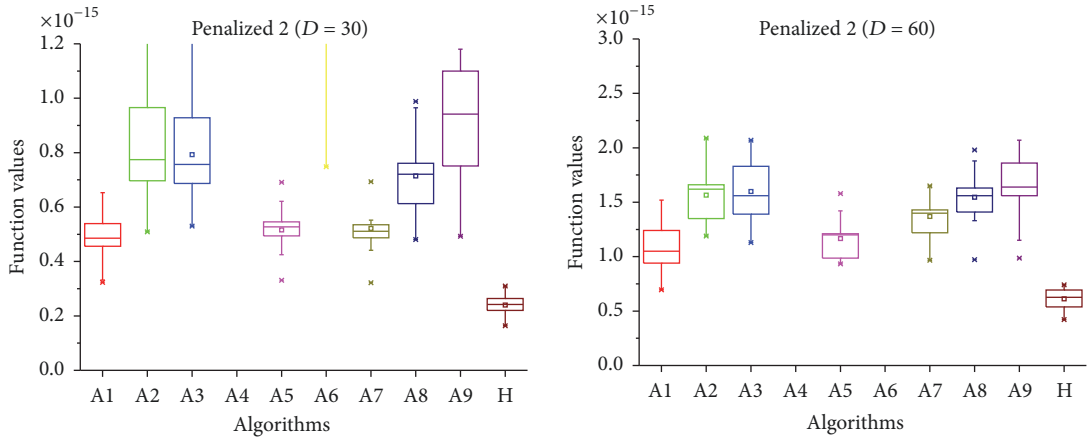

Figure 2: ANOVA results of all algorithms on some functions.

Table 5: Comparison results of ABC variants with HFEABC using pairwise Wilcoxon signed ranks test.

Algorithms	$D = 30$ (For f_{20} and f_{21}, $D = 100$)				$D = 60$ (For f_{20} and f_{21}, $D = 200$)			
HFEABC (H) versus	Wilcoxon t.	Win	Loss	Draw	Wilcoxon t.	Win	Loss	Draw
ABC (A1)	+	18	0	3	+	19	0	2
AABCLS (A2)	+	17	0	4	+	18	0	3
BSFABC (A3)	+	16	3	2	+	16	2	3
GABC (A4)	+	17	0	4	+	19	0	2
GABCS (A5)	+	19	0	2	+	17	1	3
GBABC (A6)	+	17	0	4	+	18	1	2
HGABC (A7)	+	16	0	5	+	18	0	3
IABC (A8)	+	17	0	4	+	19	0	2
LABCSS (A9)	+	18	0	3	+	19	0	2

of the data distribution can be noticed and it is clearly visible and proved that HFEABC achieved good variance distribution of compromise solutions on the majority of the benchmark functions. Note that the A7 (HGABC) algorithm also displayed its robustness on few particular functions, but its accuracy is poor.

In addition to the solution accuracy and robustness, the convergence graphs are also important tools for comparing the useful approximation abilities of different algorithms. To compare the convergence speed of the HFEABC and other ABC variants algorithms, the convergence histories of the means of 30 runs from all these algorithms for several representative functions are depicted in Figure 3, in which the horizontal axis represents the number of iterations and the vertical axis shows the mean value of the objective function values from 30 runs. It can be seen that the HFEABC has a faster convergence speed than the other algorithms on the majority of benchmark functions. To be specific, only GABCS's convergence speed on Schwefel 2.21 ($D = 30$ and $D = 60$) is faster than HFEABC. This is because GABCS sacrifices its accuracy for faster convergence speed using an enhanced local search strategy in employed bee procedure, onlooker bee procedure, and scout bee procedure and the dynamic search balance strategy used in HFEABC reduces its convergence speed to a degree. However, the accuracy and robustness of GABCS are poorer than HFEABC.

5.5. Numerical Results and Comparisons on Some Functions from CEC 2017. In addition to testing on the basic benchmark functions, some more complicated functions from CEC 2017 are used to testify the proposed algorithm. The selected functions contain hybrid functions 1~10 (function names are abbreviated as hf1~hf10) and composition functions 1~10 (function names are abbreviated as cf1~cf10). The comparison results of HFEABC with ABC and its variant at low dimension and high dimension are given in Tables 6 and 7. The values in these tables are medians obtained by these algorithms. The best values are emphasized in boldface.

In Table 6 (functions with low dimension), HFEABC (H) shows better performance than other algorithms on most hybrid functions, while for composition functions, the results obtained by GBABC (A6) are slightly better than HFEABC (H) on composition functions and they are better than other algorithms. In Table 7 (functions with high dimension), HFEABC (H) is still better than other algorithms on hybrid functions, while its performance on composition functions is improved to be nearly as good as GBABC (A6) on composition functions and they are better than other

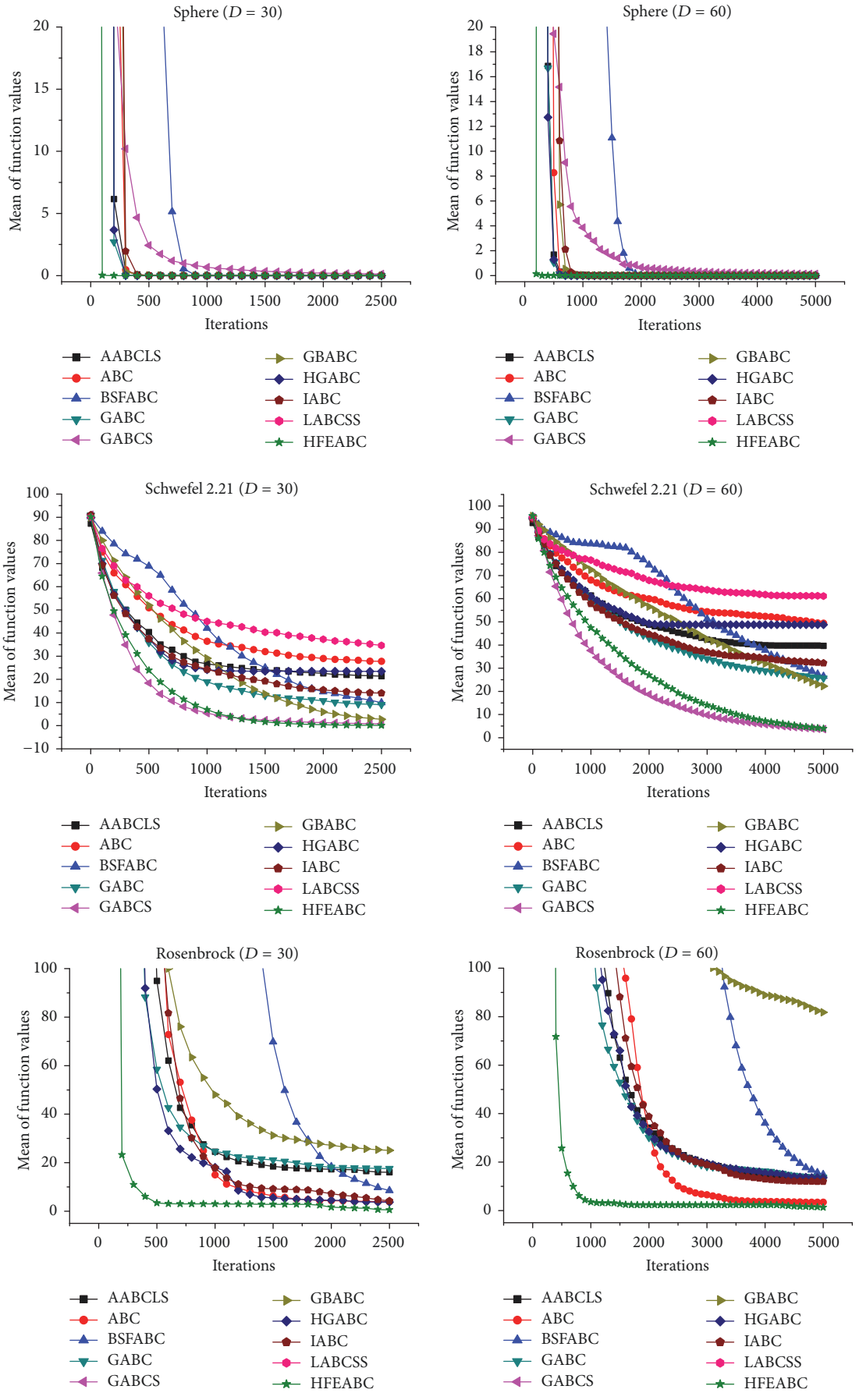

Figure 3: Continued.

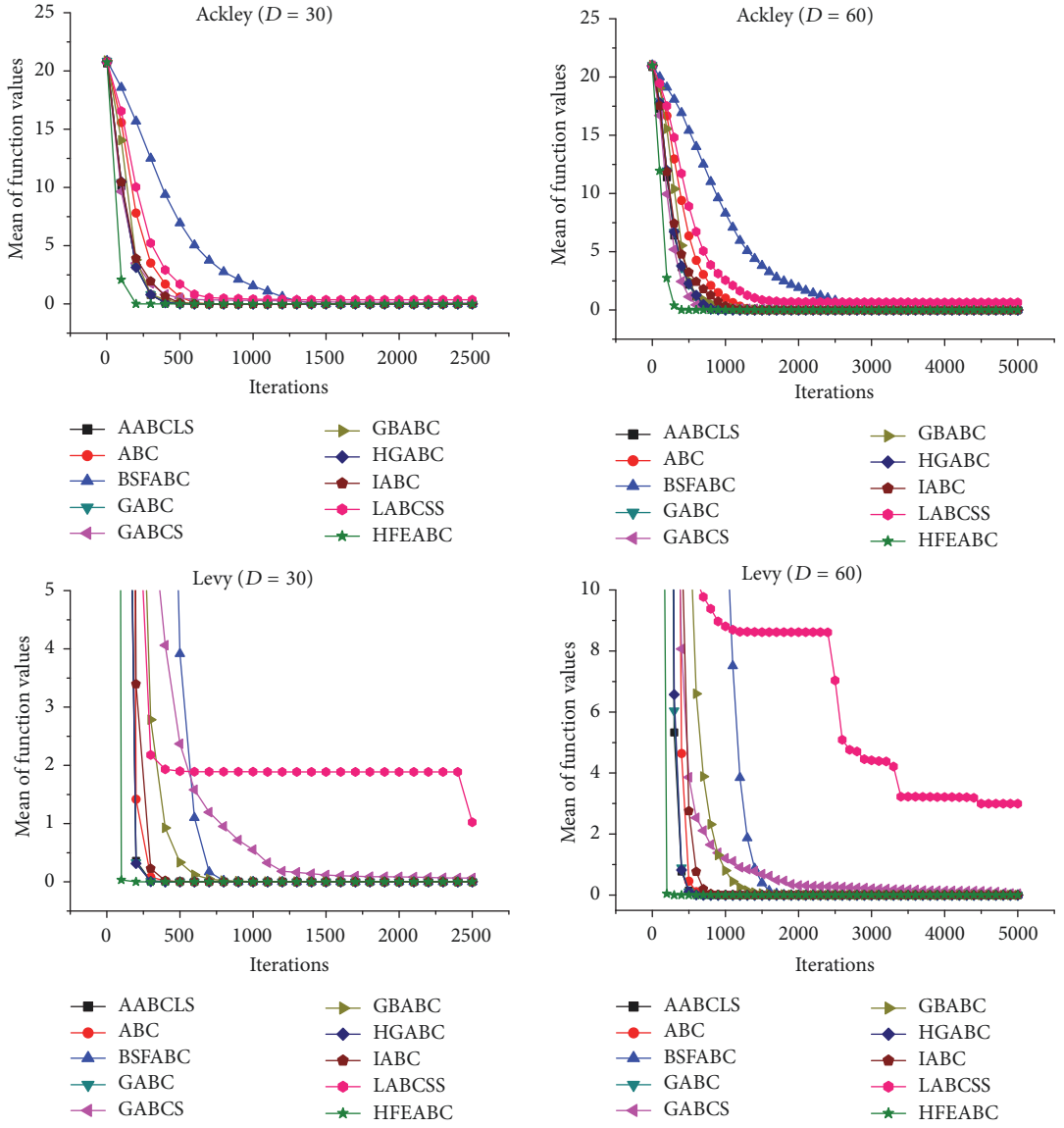

Figure 3: Mean convergence results of all algorithms on some functions with D = 30 and D = 60.

algorithms. It can be concluded that the proposed algorithm's comprehensive performance is very competitive for most complicated functions, especially for the functions with high dimension. This verifies that the proposed algorithm could provide competitive optimization performance on different problems.

6. Conclusion

This approach proposed HFEABC to improve the performance of ABC. In HFEABC, the search equation is enhanced with the global search factor based on dimension of the problem and the local search factor based on the factor library. The choice of factors in this algorithm is different for different problems and the combination of factors in low-dimensional functions and high-dimensional functions tests shows similarities. Furthermore, to prevent the algorithm from the prematurity, dynamic search balance is employed and replaces the scout bee procedure in ABC. The proposed algorithm is very effective as compared to ABC and other novel ABC variants. In basic benchmark functions tests, HFEABC outperforms ABC and its variants in terms of compatibility with different problems, robustness, and convergence speed. In some CEC 2017 functions tests, HFEABC shows better comprehensive performance. All in all, the performance of the proposed algorithm is very competitive. In addition, introducing the strategy of dynamic search balance eliminates the parameter *limit* needed in the original ABC. Future research will be along the line of implementing the HFEABC to solve more other complex engineering problems. More effective factors will be tested and the factor chosen mechanism will be studied in depth.

Table 6: Comparison results of ABC variants with HFEABC on 20 functions from CEC 2017 ($D = 30$).

Func.	A1	A2	A3	A4	A5	A6	A7	A8	A9	H
hf1	3.39e + 00	1.67e + 01	1.09e + 02	2.13e + 01	1.19e + 02	3.59e + 01	1.47e + 01	5.37e + 00	4.43e + 01	**4.31e − 02**
hf2	1.15e − 04	1.15e − 04	1.15e − 04	1.15e − 04	1.22e + 04	1.15e − 04	1.15e − 04	1.15e − 04	1.15e − 04	**1.15e − 04**
hf3	5.38e + 00	9.63e + 00	8.53e + 00	2.14e + 00	9.51e + 03	4.03e + 00	4.20e + 00	5.63e + 00	2.32e + 01	**1.50e − 02**
hf4	2.00e + 01	2.00e + 01	2.00e + 01	2.00e + 01	5.16e + 01	3.09e − 07	2.00e + 01	2.00e + 01	2.00e + 01	**4.34e − 06**
hf5	4.17e + 00	2.22e + 01	5.93e + 00	2.44e + 00	5.02e + 03	4.43e + 00	4.65e + 00	5.13e + 00	1.50e + 01	**4.52e − 01**
hf6	5.33e + 00	1.59e + 01	7.93e + 00	4.83e + 00	3.90e + 01	6.22e + 00	3.00e + 00	2.06e + 00	8.55e + 00	**5.24e − 01**
hf7	1.89e + 01	1.89e + 01	1.89e + 01	1.89e + 01	2.35e + 01	−1.11e + 00	1.89e + 01	1.89e + 01	1.89e + 01	**−1.11e + 00**
hf8	2.03e + 01	2.02e + 01	2.03e + 01	2.02e + 01	3.08e + 02	4.41e − 01	2.02e + 01	2.02e + 01	2.03e + 01	**1.22e − 01**
hf9	8.30e − 02	1.07e − 01	4.00e − 01	6.02e − 02	6.95e + 03	**2.16e − 03**	6.04e − 02	6.43e − 02	1.27e − 01	1.08e − 02
hf10	1.90e + 01	1.91e + 01	1.90e + 01	1.90e + 01	1.89e + 01	−8.78e − 01	1.91e + 01	1.90e + 01	1.91e + 01	**−1.02e + 00**
cf1	4.52e + 02	5.37e + 02	4.87e + 02	5.99e + 02	7.25e + 02	**3.29e + 02**	4.65e + 02	5.02e + 02	6.75e + 02	3.32e + 02
cf2	5.89e + 04	5.90e + 04	8.05e + 04	6.20e + 04	8.38e + 04	**5.76e + 04**	8.13e + 04	5.98e + 04	6.14e + 04	6.04e + 04
cf3	2.25e + 10	2.04e + 10	3.77e + 10	2.09e + 10	3.28e + 10	**1.93e + 10**	3.52e + 10	2.11e + 10	2.02e + 10	2.03e + 10
cf4	6.00e + 02	6.00e + 02	6.00e + 02	**6.00e + 02**	6.00e + 02	6.00e + 02	**6.00e + 02**	6.00e + 02	6.00e + 02	**6.00e + 02**
cf5	1.61e + 03	1.83e + 03	1.67e + 03	1.54e + 03	2.41e + 03	**1.03e + 03**	1.63e + 03	1.69e + 03	2.16e + 03	1.34e + 03
cf6	7.28e + 14	6.18e + 14	8.54e + 14	7.42e + 14	8.52e + 14	**7.18e + 14**	8.58e + 14	7.37e + 14	7.33e + 14	7.22e + 14
cf7	1.39e + 06	1.39e + 06	1.79e + 06	**1.33e + 06**	1.77e + 06	1.39e + 06	1.80e + 06	1.39e + 06	1.43e + 06	1.39e + 06
cf8	1.55e + 03	1.62e + 03	1.57e + 03	1.64e + 03	1.73e + 03	**1.53e + 03**	1.60e + 03	1.61e + 03	1.66e + 03	1.54e + 03
cf9	1.46e + 03	2.09e + 03	1.55e + 03	1.29e + 03	1.46e + 04	**3.41e + 02**	1.25e + 03	1.07e + 03	2.47e + 03	3.64e + 02
cf10	4.55e + 02	5.43e + 02	5.62e + 02	3.97e + 02	1.92e + 04	**3.09e + 02**	4.14e + 02	4.08e + 02	4.70e + 02	3.20e + 02

Table 7: Comparison results of ABC variants with HFEABC on 20 functions from CEC 2017 ($D = 60$).

Func.	A1	A2	A3	A4	A5	A6	A7	A8	A9	H
hf1	1.30e + 01	2.93e + 01	1.10e + 03	2.42e + 01	4.69e + 02	4.60e + 02	4.47e + 01	1.02e + 01	7.28e + 01	**4.63e − 01**
hf2	2.29e − 04	2.29e − 04	2.29e − 04	2.29e − 04	2.29e − 04	2.29e − 04	2.29e − 04	2.29e − 04	2.29e − 04	**2.29e − 04**
hf3	4.74e + 00	1.40e + 01	1.35e + 01	1.09e + 01	6.05e + 03	1.05e + 01	1.74e + 01	1.03e + 01	1.54e + 01	**3.43e − 02**
hf4	2.00e + 01	2.03e + 01	2.19e + 01	2.00e + 01	6.84e + 01	8.75e − 04	2.00e + 01	2.00e + 01	2.00e + 01	**1.46e − 03**
hf5	4.00e + 00	1.70e + 01	1.72e + 01	6.48e + 00	2.00e + 04	1.27e + 01	5.36e + 00	1.62e + 01	1.40e + 01	**6.40e − 01**
hf6	5.66e + 00	2.46e + 01	1.32e + 01	6.47e + 00	7.61e + 01	1.49e + 01	1.44e + 01	6.95e + 00	1.09e + 01	**1.14e + 00**
hf7	1.97e + 01	1.98e + 01	2.10e + 01	1.97e + 01	2.09e + 01	−2.73e − 01	1.97e + 01	1.97e + 01	1.97e + 01	**−2.78e − 01**
hf8	2.03e + 01	2.03e + 01	2.08e + 01	2.03e + 01	7.34e + 02	4.40e − 01	2.03e + 01	2.03e + 01	2.03e + 01	**3.90e − 01**
hf9	2.43e − 01	3.64e − 01	1.85e + 00	1.96e − 01	3.33e + 03	**4.54e − 03**	2.06e − 01	2.60e − 01	3.15e − 01	1.11e − 01
hf10	1.99e + 01	1.99e + 01	2.03e + 01	1.99e + 01	2.04e + 01	2.70e − 02	1.99e + 01	1.99e + 01	2.00e + 01	**−1.42e − 01**
cf1	5.85e + 02	8.41e + 02	5.60e + 02	5.89e + 02	1.19e + 03	3.90e + 02	7.11e + 02	6.98e + 02	7.50e + 02	**3.64e + 02**
cf2	1.48e + 05	1.46e + 05	1.84e + 05	1.48e + 05	1.82e + 05	**1.44e + 05**	1.79e + 05	1.49e + 05	1.48e + 05	1.46e + 05
cf3	6.05e + 10	6.12e + 10	8.60e + 10	6.44e + 10	8.76e + 10	6.09e + 10	8.63e + 10	6.43e + 10	6.22e + 10	**5.93e + 10**
cf4	6.00e + 02	6.00e + 02	6.14e + 02	6.00e + 02	6.00e + 02	6.00e + 02	**6.00e + 02**	6.00e + 02	6.00e + 02	**6.00e + 02**
cf5	1.98e + 03	2.27e + 03	2.13e + 03	2.33e + 03	3.01e + 03	**1.07e + 03**	2.33e + 03	2.32e + 03	2.56e + 03	1.64e + 03
cf6	1.65e + 15	**1.41e + 15**	1.81e + 15	1.66e + 15	1.82e + 15	1.65e + 15	1.82e + 15	1.64e + 15	1.65e + 15	1.63e + 15
cf7	3.42e + 06	3.28e + 06	4.01e + 06	3.34e + 06	3.94e + 06	**3.32e + 06**	4.06e + 06	3.38e + 06	3.44e + 06	3.36e + 06
cf8	1.55e + 03	1.62e + 03	1.59e + 03	1.59e + 03	1.73e + 03	1.57e + 03	1.64e + 03	1.57e + 03	1.66e + 03	**1.54e + 03**
cf9	1.27e + 03	1.40e + 03	1.76e + 03	3.56e + 03	1.09e + 04	**3.91e + 02**	2.00e + 03	1.41e + 03	1.99e + 03	6.32e + 02
cf10	5.39e + 02	7.48e + 02	7.25e + 02	6.29e + 02	5.64e + 03	**3.18e + 02**	5.41e + 02	4.37e + 02	4.56e + 02	4.12e + 02

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] L. Zuo, L. Shu, S. Dong, C. Zhu, and T. Hara, "A multi-objective optimization scheduling method based on the ant colony algorithm in cloud computing," *IEEE Access*, vol. 3, pp. 2687–2699, 2015.

[2] T.-H. S. Li, P.-H. Kuo, Y.-F. Ho, M.-C. Kao, and L.-H. Tai, "A biped gait learning algorithm for humanoid robots based on environmental impact assessed artificial bee colony," *IEEE Access*, vol. 3, pp. 13–26, 2015.

[3] J. Kennedy, "Particle swarm optimization," in *Encyclopedia of machine learnin*, pp. 760–766, Spriner, 2011.

[4] D. Karaboga and B. Basturk, "A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm," *Journal of Global Optimization*, vol. 39, no. 3, pp. 459–471, 2007.

[5] M. Jiang, J. Luo, D. Jiang, J. Xiong, H. Song, and J. Shen, "A Cuckoo Search-Support Vector Machine Model for Predicting Dynamic Measurement Errors of Sensors," *IEEE Access*, vol. 4, pp. 5030–5037, 2016.

[6] I. Fister, I. Fister Jr., X.-S. Yang, and J. Brest, "A comprehensive review of firefly algorithms," *Swarm and Evolutionary Computation*, vol. 13, no. 1, pp. 34–46, 2013.

[7] A. P. Engelbrecht, *Fundamentals of computational swarm intelligence*, John Wiley & Sons, 2006.

[8] W.-f. Gao, S.-y. Liu, and L.-l. Huang, "Enhancing artificial bee colony algorithm using more information-based search equations," *Information Sciences*, vol. 270, pp. 112–133, 2014.

[9] W. Yu, X. Li, H. Yang, and B. Huang, "A Multi-Objective Metaheuristics Study on Solving Constrained Relay Node Deployment Problem in WSNS," *Intelligent Automation and Soft Computing*, pp. 1–10, 2017.

[10] D. Karaboga, "An idea based on honey bee swarm for numerical optimization," Technical report-tr06, Erciyes university, engineering faculty, computer engineering department, Report2005.

[11] D. Karaboga, B. Gorkemli, C. Ozturk, and N. Karaboga, "A comprehensive survey: artificial bee colony (ABC) algorithm and applications," *Artificial Intelligence Review*, vol. 42, pp. 21–57, 2014.

[12] B. Alatas, "Chaotic bee colony algorithms for global numerical optimization," *Expert Systems with Applications*, vol. 37, no. 8, pp. 5682–5687, 2010.

[13] W. Gao and S. Liu, "Improved artificial bee colony algorithm for global optimization," *Information Processing Letters*, vol. 111, no. 17, pp. 871–882, 2011.

[14] A. Banharnsakun, B. Sirinaovakul, and T. Achalakul, "Job shop scheduling with the Best-so-far ABC," *Engineering Applications of Artificial Intelligence*, vol. 25, no. 3, pp. 583–593, 2012.

[15] G. Zhu and S. Kwong, "Gbest-guided artificial bee colony algorithm for numerical function optimization," *Applied Mathematics and Computation*, vol. 217, no. 7, pp. 3166–3173, 2010.

[16] B. Akay and D. Karaboga, "A modified Artificial Bee Colony algorithm for real-parameter optimization," *Information Sciences*, vol. 192, pp. 120–142, 2012.

[17] A. Banharnsakun, T. Achalakul, and B. Sirinaovakul, "The best-so-far selection in artificial bee colony algorithm," *Applied Soft Computing*, vol. 11, no. 2, pp. 2888–2901, 2010.

[18] R. Akbari, A. Mohammadi, and K. Ziarati, "A novel bee swarm optimization algorithm for numerical function optimization," *Communications in Nonlinear Science and Numerical Simulation*, vol. 15, no. 10, pp. 3142–3155, 2010.

[19] D. Karaboga and B. Gorkemli, "A quick artificial bee colony (qABC) algorithm and its performance on optimization problems," *Applied Soft Computing*, vol. 23, pp. 227–238, 2014.

[20] X. Zhou, Z. Wu, H. Wang, and S. Rahnamayan, "Gaussian bare-bones artificial bee colony algorithm," *Soft Computing*, vol. 20, no. 3, pp. 907–924, 2016.

[21] W.-F. Gao, S.-Y. Liu, and L.-L. Huang, "A novel artificial bee colony algorithm based on modified search equation and orthogonal learning," *IEEE Transactions on Cybernetics*, vol. 43, no. 3, pp. 1011–1024, 2013.

[22] A. Rajasekhar, A. Abraham, and M. Pant, "Design of fractional order PID controller using sobol mutated artificial bee colony alogrithm," in *Proceedings of the 2011 11th International Conference on Hybrid Intelligent Systems, HIS 2011*, pp. 151–156, Malaysia, December 2011.

[23] W. Gao, S. Liu, and L. Huang, "A global best artificial bee colony algorithm for global optimization," *Journal of Computational and Applied Mathematics*, vol. 236, no. 11, pp. 2741–2753, 2012.

[24] W. F. Gao and S. Y. Liu, "A modified artificial bee colony algorithm," *Computers & Operations Research*, vol. 39, no. 3, pp. 687–697, 2012.

[25] S. Das, S. Biswas, and S. Kundu, "Synergizing fitness learning with proximity-based food source selection in artificial bee colony algorithm for numerical optimization," *Applied Soft Computing*, vol. 13, no. 12, pp. 4676–4694, 2013.

[26] J. C. Bansal, H. Sharma, K. V. Arya, K. Deep, and M. Pant, "Self-adaptive artificial bee colony," *Optimization. A Journal of Mathematical Programming and Operations Research*, vol. 63, no. 10, pp. 1513–1532, 2014.

[27] X. Li and M. Yin, "Self-adaptive constrained artificial bee colony for constrained numerical optimization," *Neural Computing and Applications*, vol. 24, no. 3-4, pp. 723–734, 2014.

[28] H. Sharma, J. C. Bansal, K. V. Arya, and X.-S. Yang, "Lévy flight artificial bee colony algorithm," *International Journal of Systems Science*, vol. 47, no. 11, pp. 2652–2670, 2016.

[29] F. Kang, J. Li, and Q. Xu, "Structural inverse analysis by hybrid simplex artificial bee colony algorithms," *Computers & Structures*, vol. 87, no. 13-14, pp. 861–870, 2009.

[30] T. K. Sharma and M. Pant, "Enhancing the food locations in an artificial bee colony algorithm," *Soft Computing*, vol. 17, no. 10, pp. 1939–1965, 2013.

[31] H. Wang, Z. Wu, S. Rahnamayan, H. Sun, Y. Liu, and J.-s. Pan, "Multi-strategy ensemble artificial bee colony algorithm," *Information Sciences*, vol. 279, pp. 587–603, 2014.

[32] D. Aydin, "Composite artificial bee colony algorithms: From component-based analysis to high-performing algorithms," *Applied Soft Computing*, vol. 32, pp. 266–285, 2015.

[33] M. S. Kiran, H. Hakli, M. Gunduz, and H. Uguz, "Artificial bee colony algorithm with variable search strategy for continuous optimization," *Information Sciences*, vol. 300, pp. 140–157, 2015.

[34] A. Yurtkuran and E. Emel, "An Enhanced Artificial Bee Colony Algorithm with Solution Acceptance Rule and Probabilistic Multisearch," *Computational Intelligence and Neuroscience*, vol. 2016, Article ID 8085953, 2016.

[35] L. B. Ma, Y. L. Zhu, D. Y. Zhang, and B. Niu, "A hybrid approach to artificial bee colony algorithm," *Neural Computing & Applications*, vol. 27, pp. 387–409, 2016.

[36] G. Li, P. Niu, and X. Xiao, "Development and investigation of efficient artificial bee colony algorithm for numerical function optimization," *Applied Soft Computing*, vol. 12, no. 1, pp. 320–332, 2012.

[37] S. Saxena, K. Sharma, S. Shiwani, and H. Sharma, "Lbest artificial bee colony using structured swarm," in *Proceedings of the 2014 4th IEEE International Advance Computing Conference, IACC 2014*, pp. 1354–1360, India, February 2014.

[38] S. S. Jadon, J. C. Bansal, R. Tiwari, and H. Sharma, "Accelerating Artificial Bee Colony algorithm with adaptive local search," *Memetic Computing*, vol. 7, no. 3, pp. 215–230, 2015.

[39] A. n. Yurtkuran and E. Emel, "An adaptive artificial bee colony algorithm for global optimization," *Applied Mathematics and Computation*, vol. 271, pp. 1004–1023, 2015.

[40] P. Guo, W. Cheng, and J. Liang, "Global artificial bee colony search algorithm for numerical function optimization," in *Proceedings of the 2011 7th International Conference on Natural Computation, ICNC 2011*, pp. 1280–1283, China, July 2011.

[41] H. Shah, T. Herawan, R. Naseem, and R. Ghazali, "Hybrid Guided Artificial Bee Colony Algorithm for Numerical Function Optimization," in *Proceedings of Advances in Swarm Intelligence: 5th International Conference, ICSI 2014, Hefei, China*, pp. 197–206, Springer International Publishing, 2014.

[42] G. Wu, R. Mallipeddi, and P. N. Suganthan, "Problem Definitions and Evaluation Criteria for the CEC 2017 Competition and Special Session on Constrained Single Objective Real-Parameter Optimization," Technical Report, 2017.

[43] Y. Leung and Y. Wang, "An orthogonal genetic algorithm with quantization for global numerical optimization," *IEEE Transactions on Evolutionary Computation*, vol. 5, no. 1, pp. 41–53, 2001.

[44] Y. Wang and C. Dang, "An evolutionary algorithm for global optimization based on level-set evolution and latin squares," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 5, pp. 579–595, 2007.

[45] D. Karaboga and B. Akay, "A comparative study of artificial Bee colony algorithm," *Applied Mathematics and Computation*, vol. 214, no. 1, pp. 108–132, 2009.

[46] N. H. Awad, M. Z. Ali, B. Y. Q. J. J. Liang, B. Y. Qu, and P. N. Suganthan, "Problem Definitions and Evaluation Criteria for the CEC 2017 Special Session and Competition on Single Objective Bound Constrained Real-Parameter Numerical Optimization," Tech. Rep., Nanyang Technological University, Singapore, 2016.

[47] 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.

[48] S. Garcia, A. Fernandez, J. Luengo, and F. Herrera, "Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: experimental analysis of power," *Information Sciences*, vol. 180, no. 10, pp. 2044–2064, 2010.