Contents lists available at ScienceDirect

Applied Soft Computing Journal

journal homepage: www.elsevier.com/locate/asoc

Artificial bee colony algorithm based on adaptive neighborhood search and Gaussian perturbation

Songyi Xiao [a], Hui Wang [a,*], Wenjun Wang [b,*], Zhikai Huang [a], Xinyu Zhou [c], Minyang Xu [a]

[a] School of Information Engineering, Nanchang Institute of Technology, Nanchang 330099, China
[b] School of Business Administration, Nanchang Institute of Technology, Nanchang 330099, China
[c] College of Computer and Information Engineering, Jiangxi Normal University, Nanchang 330022, China

ARTICLE INFO

Article history:
Received 14 July 2020
Received in revised form 9 November 2020
Accepted 25 November 2020
Available online 5 December 2020

Keywords:
Artificial bee colony
Adaptive neighborhood search
Gaussian perturbation
Evolutionary rate

ABSTRACT

Artificial bee colony (ABC) is a type of popular swarm intelligence optimization algorithm. It is widely concerned because of its easy implementation, few parameters and strong global search ability. However, there are some limitations for ABC, such as weak exploitation ability and slow convergence. In this paper, a novel ABC with adaptive neighborhood search and Gaussian perturbation (called ABCNG) is proposed to overcome these shortcomings. Firstly, an adaptive method is used to dynamically adjust the neighborhood size. Then, a modified global best solution guided search strategy is constructed based on the neighborhood structure. Finally, a new Gaussian perturbation with evolutionary rate is designed to evolve the unchanged solutions at each iteration. Performance of ABCNG is tested on two benchmark sets and compared with some excellent ABC variants. Results show ABCNG is more competitive than six other ABCs.

1. Introduction

Many problems in engineering areas can be converted into optimization problems. Traditional mathematical optimization methods such as linear programming and gradient descent have some defects in solving complex optimization problems. Compared with those traditional methods, intelligent optimization algorithms (IOAs) have more advantages in solving continuous/discrete, multimodal and non-differentiable optimization problems. In recent years, IOAs have gradually become popular optimization tools. There are some representative IOAs including evolutionary algorithm [1,2], differential evolution [3], particle swarm optimization [4–6], ant colony optimization [7], artificial bee colony [8,9], estimation of distribution algorithm [10], firefly algorithm [11,12], multitracker optimization algorithm [13], and cuckoo search [14].

As a member of IOAs, ABC has some typical advantages: good robustness, few parameters and strong exploration ability. It has been widely used in path planning [15], scheduling problem [16], power system [17], and other problems [18]. However, ABC and other IOAs have similar shortcomings. For complex problems, they easily fall into local minima and cannot maintain the population diversity. The search accuracy may not meet the requirements.

Several studies reported that neighborhood search could effectively improve the performance of IOAs [19–21]. Because the neighborhood of a solution may cover the global optimum or a sub-optimum. By searching the neighborhood of the current solution may help find a better one. However, it is a difficult task to set the neighborhood size. A larger neighborhood size can cover more solutions and enhance the exploration, while a smaller neighborhood size can contain less solutions and improve the exploitation. Though different neighborhood search strategies were introduced into ABC, their neighborhood sizes were fixed during the search [22]. In this paper, a new ABC with adaptive neighborhood search and Gaussian perturbation (ABCNG) is proposed. In ABCNG, an adaptive method is used to dynamically adjust the neighborhood size. Then, a modified global best solution guided search strategy is constructed based on the neighborhood structure. Moreover, a new Gaussian perturbation with evolutionary rate is designed to evolve the unchanged solutions at each iteration. Finally, ABCNG is compared with six other ABCs on two benchmark sets including classical, shifted and rotated test functions.

The rest of this work is organized as follows. Section 2 mainly introduces the standard ABC. And the next section briefly introduces some ABC research work. Our method is proposed in Section 4. Section 5 gives parameter discussions and performance validation. The work is concluded in Section 6.

* Corresponding authors.
E-mail addresses: speaknow@126.com (S. Xiao), huiwang@whu.edu.cn (H. Wang), wangwenjun881@126.com (W. Wang), huangzhik2001@163.com (Z. Huang), xyzhou@jxnu.edu.cn (X. Zhou), ncxmy2019@163.com (M. Xu).

https://doi.org/10.1016/j.asoc.2020.106955
1568-4946/© 2020 Elsevier B.V. All rights reserved.

2. Artificial Bee Colony

In ABC [7], the main search task is performed by employed bees part and onlooker bees part. The former focuses on searching candidate solutions in a large region, and the latter is responsible for searching candidate solutions in a small region. Because bees determine all food sources in a population (called solutions), the number of food sources equals the number of employed bees and the number of onlooker bees. The search process of ABC is mainly composed of four parts: initialization part, employed bees part, onlooker bees part and scout bees part.

Initialization part: There are *SN* initial solutions randomly generated in this part and *SN* is the population size. Each initial solution X_i is produced as below.

$$x_{i,j} = x_{\min,j} + rand_{i,j} \cdot \left(x_{\max,j} - x_{\min,j}\right) \tag{1}$$

where i represents the index of solutions, j indicates the dimension index, $x_{i,j}$ is the jth dimension of the ith solution X_i, $x_{min,j}$ is the lower search bound, $x_{max,j}$ is the upper search bound, and $rand_{i,j}$ is a random number in the interval [0, 1].

Employed bees part: The main task of this part is to conduct a neighborhood search. New solutions are produced as below.

$$v_{i,j} = x_{i,j} + \phi_{i,j} \cdot \left(x_{i,j} - x_{k,j}\right) \tag{2}$$

where $X_k \neq X_i$, j is randomly selected from {1,2, ..., D}, and $\phi_{i,j}$ is a random weight factor in [−1, 1].

Onlooker bees part: This part aims to find more accurate solutions. In order to complete this task, onlooker bees need to select some excellent solutions for further searching. The selection method is defined as follows:

$$p_i = \frac{fit_i}{\sum_{i=1}^{SN} fit_i} \tag{3}$$

where p_i is the selection probability of X_i and fit_i is calculated as below [7].

$$fit_i = \begin{cases} 1/(1+f_i) \\ 1+abs(f_i) \end{cases} \tag{4}$$

where f_i is the objective function value of the ith solution X_i. When an excellent solution is selected, the onlooker bees use Eq. (2) to produce offspring. It is clear that a better solution has a higher selection probability.

Scout bees part: Scout bees do not exist at the initial search stage and they aim to help stagnant solutions escape from local minima. A stagnant solution means that it cannot be updated during a long cycle search. In ABC, stagnant solutions are re-initialized in the scout bees part according to Eq. (1).

3. Related work

Many excellent ABC algorithms have been proposed in recent years. A brief review of recent research on ABC is given in this section.

Lu et al. [23] used two search equations for onlooker bees and balanced those equations by Cauchy perturbation. Dong et al. [24] added Levy flight and differential perturbation strategy to ABC. Chen et al. [25] introduced Boltzmann selection, mutation operation and extremal optimization into ABC. Results proved the new algorithm could improve the solution accuracy and convergence speed. Banharnsakun et al. [9] proposed best so-far ABC by using the best-to-date selection mechanism. Li et al. [26] designed a novel ABC algorithm (called RABC) with improved gene recombination operator. Chen et al. [27] presented an adaptive differential artificial bee swarm algorithm (called sdABC). Through differential search strategy, more variables are updated based on the combination of variation and crossover. Jadon et al. [28] improved the position updating equation of the original ABC by using the global and local neighborhood of differential evolution. Kong et al. [29] constructed a combined search strategy based on random breadth-first search and random depth-first search. Song et al. [30] proposed an adaptive ABC algorithm with two search strategies. The change of the success rate of each search strategy can adjust the selection probability of the corresponding strategy. In [31], a redefined ABC was proposed. Experimental analysis showed it outperformed many competing algorithms on several practical problems and some basic test sets. Li et al. [32] designed an ABC algorithm combining a comprehensive search method. The comprehensive search method includes heuristic Gaussian search, neighborhood search and adaptive Gaussian perturbation.

Gao et al. [33] proposed a new ABC with Parzen window. Firstly, multiple strategies were established. Then, Parzen window method was introduced into the greedy algorithm to evaluate the offspring. Finally, the convergence and diversity of the algorithm are balanced by two neighborhood mechanisms. In [34], a mixture of ABC and cultural algorithm was proposed. The hybrid algorithm uses previous generations of information to help find potential food sources. Gao et al. [35] proposed an ABC algorithm based on directional learning and elite learning. The former can lead the search to a promising direction and the latter can gradually improve the convergence rate without losing the population diversity. Cui et al. [36] divided the population into two subpopulations: convergent population (CP) and diverse population (DP). CP is commitment to finds outstanding or promising individuals, and DP focuses on maintaining population diversity. Yavuz et al. [37] proposed a new ABC algorithm integrating multiple methods (called SSEABC). It improves the algorithm performance by competing to determine the appropriate local search method and eliminating the inappropriate search equation to determine the appropriate search equation for the specific problem. According to chaotic mapping, Alatas et al. [38] proposed a new parameter adaptive ABC, in which random numbers required by ABC are generated by chaotic number generator. Seven new chaotic ABC algorithms were constructed and different chaotic maps were analyzed. Li and Yang [39] developed a new variant of ABC, in which the past successful experiences were retained by a memory mechanism. Zhang et al. [40] presented a modified ABC by embedding the grenade explosion method. Yang et al. [41] proposed an ABC algorithm with an adaptive coding learning method. Under the guidance of covariance matrix, the search equation is coded in both characteristic and natural coordinate systems. To improve grammatical evolution [42,43], a new ABC algorithm called grammatical ABC algorithm was proposed [43]. ABC was employed to write a program in the ABC algorithm.

4. Proposed approach

The search strategy/equation has a significant impact on ABC. To improve the performance ABC, different search strategies were proposed. However, merely changing the search strategy is not effective to tackle the shortcomings of ABC. Recently, neighborhood topology was introduced into ABC to obtain a balance between exploration ability and exploitation ability. In order to strengthen the exploitation ability of algorithm, the ring neighborhood topology was used in [22], and results showed a large improvement on ABC. In this paper, a new ABC algorithm (namely ABCNG) is proposed, which is composed of adaptive neighborhood search and Gaussian perturbation. In ABCNG, an adaptive method is used to dynamically adjust the neighborhood size. Then, a modified search strategy is designed based on the neighborhood. In addition, a new Gaussian perturbation with evolutionary rate is used to maintain population diversity and avoid local minima.

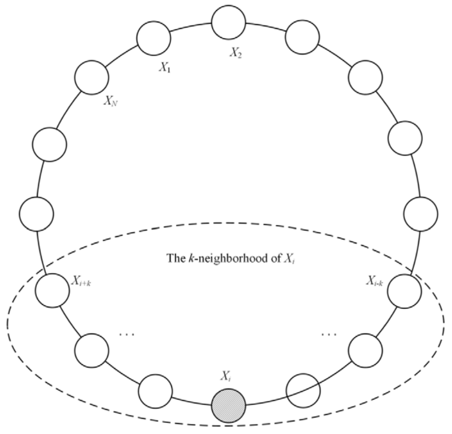


Fig. 1. The k-neighborhood of X_i.

4.1. Adaptive neighborhood search

Neighborhood search is an important operation for improving the performance of IOAs. Recently, a new neighborhood structure called k-neighborhood was successfully applied to different IOAs [21,43–45]. It concatenates all solutions to construct a ring topology. Fig. 1 illustrates the basic form of this new neighborhood. The individual with the last ordinal number in the population is associated with the individual with the first ordinal number. The k-neighborhood of X_i has $2k + 1$ solutions $\{X_{i-k}, \ldots, X_i, \ldots, X_{i+k}\}$, and X_i is located at the center of the neighborhood. In [44], Das et al. firstly used the k-neighborhood to design a mutation operator in differential evolution. In [45], Wang et al. introduced the k-neighborhood into PSO to construct a local search strategy. In order to reduce the computational complexity of firefly algorithm, Wang et al. [46] proposed a new neighborhood attraction model based on the k-neighborhood. Recently, the k-neighborhood was used to improve the selection method in ABC [22]. For the above studies, the neighborhood radius k was unchanged in the optimization process. The literature [46] pointed out that the parameter k could affect the search ability of the corresponding strategy. Choosing the best k is not an easy task and different search stages may require different k values.

Based on the above analysis, an adaptive neighborhood search strategy is proposed in this section. In our approach, the neighborhood radius k is not fixed and it can be adaptively adjusted in the search process. When we need to enhance the exploitation ability, the neighborhood radius k should be decreased. When we need to strengthen the exploration ability, the neighborhood radius k should be increased. For the current X_i, a new solution V_i is produced in the employed bee search or onlooker bee search. By monitoring the quality of the offspring V_i, the neighborhood radius k is dynamically updated as follows.

$$k = \begin{cases} k + 1, \text{ if } f(V_i) < f(X_i) \\ k - 1, \text{ otherwise} \end{cases} \tag{5}$$

where $f(V_i)$ and $f(X_i)$ represent the objective function values of V_i and X_i, respectively. The neighborhood radius k should satisfy the constraint $1 \le k \le (N\text{-}1)/2$. If k violates the constraint, it is modified as follows.

$$k = \begin{cases} 1, & \text{if } k < 1 \\ \dfrac{N-1}{2}, & \text{if } k > \dfrac{N-1}{2} \end{cases} \tag{6}$$

This paper only considers minimization problems. So, a better offspring V_i will increase the neighborhood radius k, and a worse offspring V_i will decrease the value of k. The offspring V_i is produced by the employed or onlooker bee search. If the quality of offspring V_i is better than X_i, it means this is a high-quality search. Increasing the neighborhood radius k can expand the search area and improve the exploration. If the quality of the offspring V_i is worse than X_i, it indicates a low search efficiency. Decreasing the neighborhood radius k can narrow the search range and enhance the exploitation.

According to the improved k-neighborhood structure, a modified GABC search strategy is designed as below.

$$v_{i,j} = x_{ni,j} + \phi_{i,j} \cdot (x_{ni,j} - x_{no,j}) + \varphi_{i,j} \cdot (Gbest_j - x_{ni,j}) \tag{7}$$

where X_{ni} is chosen from the dynamic neighborhood of X_i randomly ($i \neq ni$), and X_{no} is randomly selected from the outside of the neighborhood. The weight parameters $\phi_{i,j}$ and $\varphi_{i,j}$ satisfy the ranges $[-1, 1]$ and $[0, 1.5]$, respectively [47].

4.2. New Gaussian perturbation

In ABC, the onlooker bees aim to select elite solutions for further search. In the previous section, the adaptive neighborhood search is used in both the employed and the onlooker bees. Although this method can improve the search of ABC, the task of the onlooker bees is not fully completed. In this section, the local search ability is enhanced by using the search information of failed solutions. For each X_i, if its neighborhood search fails, then X_i is conducted on an enhanced search again.

Gaussian perturbation was successfully used in several intelligent optimization algorithms to achieve better performance [48–50]. However, the step size of Gaussian perturbation determines its search ability. A large step size has a great effect on jumping out of the local minima, and it can enhance the exploration. A small step size is helpful to find good solutions and strengthen the exploitation ability. To measure the improvement of solutions, two kinds of evolutionary rates (δ_i and δ_a) are defined as follows.

$$\delta_i = \frac{f_i(t) - f_i(t-1)}{f_i(t)} \tag{8}$$

$$\delta_a = \frac{\sum_{i=1}^{SN} \delta_i}{SN} \tag{9}$$

where $f_i(t)$ is the function value of X_i at the tth iteration. As seen, δ_i represents the evolutionary rate of X_i and δ_a indicates the average evolutionary rate of the population.

A new Gaussian perturbation search strategy based on evolutionary rate is constructed as follow.

$$v_{i,j} = x_{i,j} \cdot (1 + g\,(\delta_1, \delta_2)) \tag{10}$$

where $j = 1, 2, \ldots, D$, $g()$ represents a Gaussian distribution with mean δ_1 and standard deviation δ_2. The parameters δ_1 and δ_2 are selected from the evolutionary rates δ_i or δ_a.

Algorithm 1: Gaussian perturbation

```
Begin
  Generate $V_i$ according to Eq. (10) and evaluate $V_i$;
  $FEs$=$FEs$+1;
  if $V_i$ is better than $X_i$ then
    $X_i$=$V_i$ and set $trial_i$=0;
  else
    $trial_i$= $trial_i$+1;
  end if
End
```

Table 1
Results of ABC, ABCNG-*ii*, ABCNG-*ia*, ABCNG-*aa*, and ABCNG-*ai* for $D = 30$.

F	ABC		ABCN-*ii*		ABCN-*ia*		ABCN-*aa*		ABCN-*ai*	
	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev
f_1	1.04×10^{-17}	1.20×10^{-17}	2.16×10^{-90}	2.60×10^{-89}	2.88×10^{-131}	7.18×10^{-130}	1.05×10^{-183}	0	**6.88×10^{-220}**	**0**
f_2	4.38×10^{-10}	4.72×10^{-10}	1.35×10^{-83}	3.15×10^{-82}	6.11×10^{-129}	1.43×10^{-127}	1.51×10^{-178}	0	**2.89×10^{-218}**	**0**
f_3	1.14×10^{-19}	9.89×10^{-20}	1.9×10^{-90}	2.95×10^{-89}	1.29×10^{-133}	2.59×10^{-132}	1.14×10^{-183}	0	**4.05×10^{-217}**	**0**
f_4	2.02×10^{-31}	5.30×10^{-31}	9.14×10^{-82}	2.05×10^{-80}	1.72×10^{-151}	4.98×10^{-150}	2.45×10^{-200}	0	**3.11×10^{-206}**	**0**
f_5	7.69×10^{-10}	3.04×10^{-10}	2.07×10^{-56}	2.92×10^{-55}	1.79×10^{-106}	4.23×10^{-105}	**8.05×10^{-144}**	**2.37×10^{-142}**	1.12×10^{-138}	2.91×10^{-137}
f_6	1.39×10^{1}	1.07×10^{1}	3.52×10^{-5}	4.04×10^{-4}	**1.65×10^{-83}**	**4.77×10^{-82}**	7.61×10^{-4}	1.88×10^{-3}	2.84×10^{-4}	7.55×10^{-4}
f_7	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
f_8	**7.18×10^{-66}**	**5.92×10^{-71}**	**7.18×10^{-66}**	**1.73×10^{-80}**	**7.18×10^{-66}**	**1.73×10^{-80}**	**7.18×10^{-66}**	**1.73×10^{-80}**	**7.18×10^{-66}**	**1.73×10^{-80}**
f_9	4.52×10^{-2}	1.09×10^{-2}	2.25×10^{-3}	7.50×10^{-3}	**3.29×10^{-5}**	**1.58×10^{-4}**	5.66×10^{-4}	1.62×10^{-3}	1.83×10^{-4}	1.21×10^{-3}
f_{10}	**5.45×10^{-2}**	**5.86×10^{-2}**	2.71×10^{1}	1.75×10^{0}	2.71×10^{1}	1.54×10^{0}	2.72×10^{1}	3.72×10^{-1}	2.72×10^{1}	8.15×10^{-1}
f_{11}	3.50×10^{-14}	1.35×10^{-13}	3.32×10^{-2}	9.78×10^{-1}	**0**	**0**	**0**	**0**	3.32×10^{-2}	9.78×10^{-1}
f_{12}	1.70×10^{-12}	4.36×10^{-12}	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
f_{13}	2.36×10^{-14}	5.62×10^{-14}	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
f_{14}	4.58×10^{-12}	1.59×10^{-12}	2.37×10^{1}	2.59×10^{2}	**1.76×10^{-12}**	**1.79×10^{-12}**	3.95×10^{0}	1.16×10^{2}	1.58×10^{1}	2.77×10^{2}
f_{15}	4.31×10^{-6}	1.85×10^{-6}	2.58×10^{-15}	9.53×10^{-15}	**4.44×10^{-16}**	**0**	**4.44×10^{-16}**	**0**	**4.44×10^{-16}**	**0**
f_{16}	1.03×10^{-18}	6.90×10^{-19}	**1.57×10^{-32}**	**4.50×10^{-47}**	**1.57×10^{-32}**	**4.50×10^{-47}**	**1.57×10^{-32}**	**4.50×10^{-47}**	**1.57×10^{-32}**	**4.50×10^{-47}**
f_{17}	4.88×10^{-18}	5.03×10^{-18}	**1.35×10^{-32}**	**3.00×10^{-47}**	**1.35×10^{-32}**	**3.00×10^{-47}**	**1.35×10^{-32}**	**3.00×10^{-47}**	3.66×10^{-04}	1.08×10^{-02}
f_{18}	2.35×10^{-6}	1.66×10^{-6}	7.69×10^{-52}	2.27×10^{-50}	2.99×10^{-105}	8.82×10^{-104}	2.78×10^{-17}	8.19×10^{-16}	**6.49×10^{-126}**	**1.92×10^{-124}**
f_{19}	4.46×10^{-14}	5.39×10^{-14}	**1.35×10^{-31}**	**3.60×10^{-46}**	**1.35×10^{-31}**	**3.60×10^{-46}**	3.82×10^{-08}	1.13×10^{-06}	**1.35×10^{-31}**	**3.60×10^{-46}**
f_{20}	2.06×10^{-02}	2.35×10^{-02}	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
f_{21}	**-7.83×10^{1}**	**0**	**-7.83×10^{1}**	**5.12×10^{-14}**	**-7.83×10^{1}**	**6.36×10^{-14}**	**-7.83×10^{1}**	**6.36×10^{-14}**	**-7.83×10^{1}**	**6.19×10^{-14}**
f_{22}	-2.80×10^{1}	2.73×10^{-1}	$-2.82 \times 10^{+1}$	1.24×10^{0}	**-2.83×10^{1}**	**1.32×10^{0}**	-2.82×10^{1}	1.34×10^{0}	-2.81×10^{1}	1.39×10^{0}

Table 2
Results of ABC, ABCNG-*ii*, ABCNG-*ia*, ABCNG-*aa*, and ABCNG-*ai* for *D*=50.

F	ABC		ABCN-*ii*		ABCN-*ia*		ABCN-*aa*		ABCN-*ai*	
	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev
f_1	1.03×10^{-16}	2.46×10^{-16}	5.54×10^{-95}	6.97×10^{-94}	8.68×10^{-180}	0	5.38×10^{-257}	0	$\mathbf{4.35 \times 10^{-298}}$	**0**
f_2	8.93×10^{-16}	2.30×10^{-16}	3.67×10^{-87}	9.20 × 10–86	2.05×10^{-178}	0	2.79×10^{-249}	0	$\mathbf{1.83 \times 10^{-293}}$	**0**
f_3	1.06×10^{-15}	1.89×10^{-16}	7.75×10^{-96}	1.32×10^{-94}	7.94×10^{-180}	0	6.81×10^{-256}	0	$\mathbf{1.63 \times 10^{-297}}$	**0**
f_4	1.90×10^{-17}	1.25×10^{-17}	3.28×10^{-79}	9.63×10^{-78}	4.25×10^{-201}	0	3.23×10^{-262}	0	$\mathbf{4.40 \times 10^{-260}}$	**0**
f_5	2.40×10^{-15}	2.20×10^{-16}	2.24×10^{-58}	4.10×10^{-57}	1.67×10^{-147}	4.61×10^{-146}	1.25×10^{-52}	3.70×10^{-51}	$\mathbf{4.62 \times 10^{-150}}$	$\mathbf{1.36 \times 10^{-148}}$
f_6	6.51×10^{0}	1.08×10^{0}	9.44×10^{-2}	4.60×10^{-1}	$\mathbf{1.35 \times 10^{-48}}$	$\mathbf{2.77 \times 10^{-47}}$	8.68×10^{-2}	1.07×10^{-1}	4.89×10^{-2}	5.58×10^{-2}
f_7	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
f_8	$\mathbf{2.67 \times 10^{-109}}$	1.43×10^{-116}	$\mathbf{2.67 \times 10^{-109}}$	$\mathbf{2.59 \times 10^{-124}}$	$\mathbf{2.67 \times 10^{-109}}$	$\mathbf{2.59 \times 10^{-124}}$	$\mathbf{2.67 \times 10^{-109}}$	$\mathbf{2.59 \times 10^{-124}}$	$\mathbf{2.67 \times 10^{-109}}$	$\mathbf{2.59 \times 10^{-124}}$
f_9	3.93×10^{2}	7.61×10^{-3}	2.82×10^{-3}	8.71×10^{-3}	$\mathbf{2.64 \times 10^{-4}}$	$\mathbf{8.97 \times 10^{-4}}$	5.40×10^{-4}	2.39×10^{-3}	2.81×10^{-4}	1.83×10^{-3}
f_{10}	4.67×10^{0}	1.56×10^{1}	4.70×10^{1}	1.04×10^{0}	4.70×10^{1}	5.92×10^{-1}	4.70×10^{1}	4.30×10^{-1}	4.70×10^{1}	7.40×10^{-1}
f_{11}	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
f_{12}	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
f_{13}	3.55×10^{-16}	4.55×10^{-16}	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
f_{14}	$\mathbf{2.71 \times 10^{-10}}$	$\mathbf{1.13 \times 10^{-9}}$	2.37×10^{1}	2.59×10^{2}	2.09×10^{0}	1.62×10^{0}	3.95×10^{1}	3.49×10^{2}	1.18×10^{1}	2.57×10^{2}
f_{15}	5.73×10^{-14}	1.00×10^{-14}	3.17×10^{-15}	8.23×10^{-15}	$\mathbf{4.44 \times 10^{-16}}$	**0**	$\mathbf{4.44 \times 10^{-16}}$	**0**	$\mathbf{4.44 \times 10^{-16}}$	**0**
f_{16}	1.00×10^{-15}	3.26×10^{-16}	$\mathbf{9.42 \times 10^{-33}}$	$\mathbf{1.50 \times 10^{-47}}$	$\mathbf{9.42 \times 10^{-33}}$	$\mathbf{1.50 \times 10^{-47}}$	$\mathbf{9.42 \times 10^{-33}}$	$\mathbf{1.50 \times 10^{-47}}$	$\mathbf{9.42 \times 10^{-33}}$	$\mathbf{1.50 \times 10^{-47}}$
f_{17}	1.02×10^{-15}	1.37×10^{-16}	$\mathbf{1.35 \times 10^{-32}}$	$\mathbf{3.00 \times 10^{-47}}$	$\mathbf{1.35 \times 10^{-32}}$	$\mathbf{3.00 \times 10^{-47}}$	$\mathbf{1.35 \times 10^{-32}}$	$\mathbf{3.00 \times 10^{-47}}$	$\mathbf{1.35 \times 10^{-32}}$	$\mathbf{3.00 \times 10^{-47}}$
f_{18}	1.39×10^{-7}	6.03×10^{-7}	1.03×10^{-54}	2.09×10^{-53}	$\mathbf{2.21 \times 10^{-136}}$	$\mathbf{6.51 \times 10^{-135}}$	6.19×10^{-43}	1.83×10^{-41}	2.04×10^{-17}	6.00×10^{-16}
f_{19}	8.71×10^{-16}	2.09×10^{-16}	$\mathbf{1.35 \times 10^{-31}}$	$\mathbf{3.60 \times 10^{-46}}$	$\mathbf{1.35 \times 10^{-31}}$	$\mathbf{3.60 \times 10^{-46}}$	$\mathbf{1.35 \times 10^{-31}}$	$\mathbf{3.60 \times 10^{-46}}$	$\mathbf{1.35 \times 10^{-31}}$	$\mathbf{3.60 \times 10^{-46}}$
f_{20}	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
f_{21}	$\mathbf{-7.83 \times 10^{1}}$	$\mathbf{1.42 \times 10^{-14}}$	$\mathbf{-7.83 \times 10^{1}}$	$\mathbf{6.19 \times 10^{-14}}$	$\mathbf{-7.83 \times 10^{1}}$	$\mathbf{6.82 \times 10^{-14}}$	$\mathbf{-7.83 \times 10^{1}}$	$\mathbf{6.03 \times 10^{-14}}$	$\mathbf{-7.83 \times 10^{1}}$	$\mathbf{6.19 \times 10^{-14}}$
f_{22}	-4.55×10^{1}	6.61×10^{-1}	-4.52×10^{1}	1.95×10^{0}	$\mathbf{-4.59 \times 10^{1}}$	$\mathbf{2.09 \times 10^{0}}$	-4.53×10^{1}	2.00×10^{0}	-4.53×10^{1}	1.91×10^{0}

Table 3
Results of ABC, ABCNG-*ii*, ABCNG-*ia*, ABCNG-*aa*, and ABCNG-*ai* for *D*=100.

F	ABC		ABCN-*ii*		ABCN-*ia*		ABCN-*aa*		ABCN-*ai*	
	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev
f_1	4.78×10^{-16}	4.79×10^{-16}	8.10×10^{-104}	7.61×10^{-103}	4.52×10^{-314}	0	**0**	**0**	**0**	**0**
f_2	5.30×10^{-09}	4.89×10^{-09}	8.07×10^{-93}	1.65×10^{-91}	3.15×10^{-299}	0	**0**	**0**	**0**	**0**
f_3	1.03×10^{-16}	8.69×10^{-17}	1.47×10^{-102}	2.51×10^{-101}	2.02×10^{-312}	0	**0**	**0**	**0**	**0**
f_4	1.17×10^{-31}	4.89×10^{-31}	4.01×10^{-79}	8.18×10^{-78}	5.92×10^{-319}	0	**0**	**0**	**0**	**0**
f_5	1.27×10^{-09}	4.11×10^{-10}	6.79×10^{-63}	9.40×10^{-62}	$\mathbf{1.46 \times 10^{-266}}$	**0**	9.58×10^{-42}	2.55×10^{-40}	1.62×10^{-46}	2.36×10^{-45}
f_6	$2.85 \times 10^{+01}$	$1.31 \times 10^{+00}$	$6.26 \times 10^{+00}$	$1.01 \times 10^{+01}$	$\mathbf{1.71 \times 10^{-38}}$	$\mathbf{5.04 \times 10^{-37}}$	$3.03 \times 10^{+00}$	$2.08 \times 10^{+00}$	$2.29 \times 10^{+00}$	$1.26 \times 10^{+00}$
f_7	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
f_8	$\mathbf{7.12 \times 10^{-218}}$	**0**	$\mathbf{7.12 \times 10^{-218}}$	**0**	$\mathbf{7.12 \times 10^{-218}}$	**0**	$\mathbf{7.12 \times 10^{-218}}$	**0**	$\mathbf{7.12 \times 10^{-218}}$	**0**
f_9	2.89×10^{-1}	5.49×10^{-2}	4.04×10^{-3}	1.19×10^{-2}	$\mathbf{1.55 \times 10^{-4}}$	$\mathbf{5.74 \times 10^{-4}}$	7.25×10^{-4}	2.65×10^{-3}	2.21×10^{-4}	3.12×10^{-3}
f_{10}	$\mathbf{1.82 \times 10^{-1}}$	$\mathbf{2.04 \times 10^{-1}}$	9.65×10^{1}	1.75×10^{0}	9.65×10^{1}	8.53×10^{-1}	9.65×10^{1}	3.07×10^{-1}	9.65×10^{1}	5.40×10^{-1}
f_{11}	2.28×10^{-8}	1.14×10^{-7}	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
f_{12}	1.69×10^{-1}	3.97×10^{-1}	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
f_{13}	9.77×10^{-17}	1.90×10^{-16}	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
f_{14}	$\mathbf{5.79 \times 10^{-11}}$	$\mathbf{7.42 \times 10^{-12}}$	3.95×10^{1}	3.87×10^{2}	4.19×10^{1}	3.21×10^{2}	6.32×10^{1}	4.01×10^{2}	5.53×10^{1}	3.64×10^{2}
f_{15}	1.42×10^{-8}	4.33×10^{-9}	3.64×10^{-15}	5.84×10^{-15}	$\mathbf{4.44 \times 10^{-16}}$	**0**	$\mathbf{4.44 \times 10^{-16}}$	**0**	1.32×10^{0}	2.72×10^{1}
f_{16}	1.48×10^{-17}	1.08×10^{-17}	$\mathbf{4.71 \times 10^{-33}}$	$\mathbf{7.50 \times 10^{-48}}$	$\mathbf{4.71 \times 10^{-33}}$	$\mathbf{7.50 \times 10^{-48}}$	$\mathbf{4.71 \times 10^{-33}}$	$\mathbf{7.50 \times 10^{-48}}$	$\mathbf{4.71 \times 10^{-33}}$	$\mathbf{7.50 \times 10^{-48}}$
f_{17}	3.27×10^{-16}	4.05×10^{-16}	$\mathbf{1.35 \times 10^{-32}}$	$\mathbf{3.00 \times 10^{-47}}$	$\mathbf{1.35 \times 10^{-32}}$	$\mathbf{3.00 \times 10^{-47}}$	5.08×10^{-3}	1.04×10^{-1}	6.86×10^{-3}	1.33×10^{-1}
f_{18}	3.96×10^{-4}	4.66×10^{-4}	2.04×10^{-17}	6.00×10^{-16}	$\mathbf{3.16 \times 10^{-224}}$	**0**	1.26×10^{-16}	2.00×10^{-15}	4.07×10^{-17}	8.34×10^{-16}
f_{19}	4.47×10^{-14}	5.35×10^{-14}	$\mathbf{1.35 \times 10^{-31}}$	$\mathbf{3.60 \times 10^{-46}}$	$\mathbf{1.35 \times 10^{-31}}$	$\mathbf{3.60 \times 10^{-46}}$	$\mathbf{1.35 \times 10^{-31}}$	$\mathbf{3.60 \times 10^{-46}}$	$\mathbf{1.35 \times 10^{-31}}$	$\mathbf{3.60 \times 10^{-46}}$
f_{20}	4.54×10^{-1}	1.24×10^{-1}	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
f_{21}	$\mathbf{-7.83 \times 10^{+01}}$	$\mathbf{2.23 \times 10^{-14}}$	$\mathbf{-7.83 \times 10^{+01}}$	**1.35E−13**	$\mathbf{-7.83 \times 10^{+01}}$	$\mathbf{9.43 \times 10^{-14}}$	$\mathbf{-7.83 \times 10^{+01}}$	$\mathbf{1.01 \times 10^{-13}}$	$\mathbf{-7.83 \times 10^{+01}}$	$\mathbf{1.65 \times 10^{-13}}$
f_{22}	-8.32×10^{1}	4.53×10^{0}	-8.76×10^{1}	3.25×10^{0}	$\mathbf{-8.95 \times 10^{1}}$	$\mathbf{2.73 \times 10^{0}}$	-8.78×10^{1}	3.41×10^{0}	-8.79×10^{1}	2.86×10^{0}

In the onlooker bee search, when the quality of offspring V_i is lower than X_i, the Gaussian perturbation is conducted on X_i to produce another V_i. Then, evolutionary rates δ_i and δ_a are calculated by the onlooker bee phase at the last iteration. The function values of all solutions in the population are preserved before the onlooker bee stage. After completing the onlooker bee search, Eqs. (8) and (9) are used to calculate the evolution rates δ_i and δ_a, respectively. The main steps of the Gaussian perturbation are described in Algorithm 1, where $trial_i$ monitors the quality changes of X_i.

```
Algorithm 2: Framework of ABCNG
Begin
  An initial population with $SN$ solutions is randomly generated by Eq. (1);
  Initialize $trial_i$=0;
  while $FEs \le MaxFEs$ do
    for $i$=1 to $SN$ do
      Generate $V_i$ according to Eq. (7);
      Calculate the function value of $V_i$ and set $FEs$=$FEs$+1;
      if $V_i$ is better than $X_i$ then
        $X_i$=$V_i$ and set $trial_i$=0;
      else
        $trial_i$= $trial_i$+1;
      end if
      Update the neighborhood radius $k$ according to Eqs. (5) and (6).
    end for
    for $i$=1 to $SN$ do
      Generate $V_i$ according to Eq. (7);
      Calculate the function value of $V_i$ and set $FEs$=$FEs$+1;
      if $V_i$ is better than $X_i$ then
        $X_i$=$V_i$ and set $trial_i$=0;
      else
        $trial_i$= $trial_i$+1;
        Execute the Gaussian perturbation according to Algorithm 1;
      end if
      Update the neighborhood radius $k$ according to Eq. (5);
      Update the evolutionary rates $\delta_i$ and $\delta_a$ according to Eqs. (8) and (9);
    end for
    if max{$trial_i$}>$limit$ then
      Replace $X_i$ with a new solution generated by Eq. (1);
    end if
  end while
  Output the best result;
End
```

4.3. Framework of the proposed approach

As mentioned before, ABCNG employs two strategies including adaptive neighborhood search and Gaussian perturbation. Algorithm 2 gives the overall framework of ABCNG. In the framework, *FEs* is the evaluation times of computing the objective function, and *MaxFEs* is the maximum value of *FEs*. Compared with the original ABC, the search pattern of onlooker bees is improved in ABCNG. In ABC, the roulette method is adopted to select a better solution and further search the neighborhood of the better solution. In ABCNG, the onlooker bees no longer use roulette to select good individuals for searching. When the current solution is not replaced by the new one, the Gaussian perturbation is used to produce another new solution.

To solve a specific problem, we assume that $O(f)$ is the computational complexity of the objective function and T_{max} is the maximum number of iterations. For ABC, its computational complexity is $O(T_{max} \cdot (SN \cdot f + SN \cdot f + f)) = O(T_{max} \cdot f \cdot (2SN + 1)) = O(T_{max} \cdot SN \cdot f)$. For the proposed ABCNG, its complexity is $O(T_{max} \cdot (SN \cdot f + 2 \cdot SN \cdot f + f)) = O(T_{max} \cdot f \cdot (3SN + 1)) = O(T_{max} \cdot SN \cdot f)$. Though both ABCNG and ABC have the same computational complexity, ABCNG cost more computing time than ABC.

Table 4
Mean ranking values ABC, ABCNG-*ii*, ABCNG-*ia*, ABCNG-*aa*, and ABCNG-*ai* achieved by Friedman test.

Algorithms	$D = 30$	$D = 50$	$D = 100$
ABC	4.23	3.95	4.23
ABCNG-*ii*	3.20	3.34	3.00
ABCNG-*aa*	2.70	2.82	2.77
ABCNG-*ai*	2.59	2.45	2.70
ABCNG-*ia*	**2.27**	**2.43**	**2.30**

5. Experimental study

5.1. Benchmark functions and parameter settings

To verify the performance of ABCNG, two groups of benchmark functions are used. The first group contains 22 classical test functions, and the second group is the CEC 2013 benchmark set. The dimensions of the first group functions are set to 30, 50 and 100 respectively [46,51]. The specific representations of the first group functions are shown in [36].

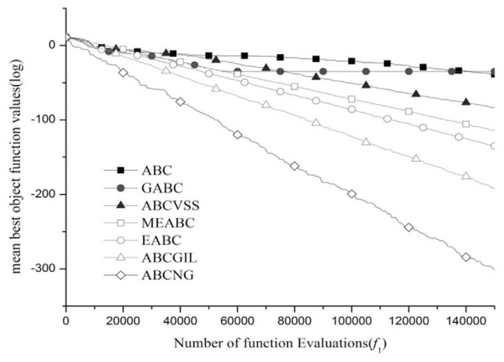
Two metrics (*Mean* and *std*) are used to evaluate the performance of ABCNG. The *mean* represents the average value of the optimal results obtained by the algorithm in 25 independent runs, and *std* represents the corresponding variance. They show the accuracy and stability of the global optimal value obtained by the algorithm. There are three series of experiments. Experiment 1 studies the parameters δ_1 and δ_2 in ABCNG. Experiment 2 compares the performance of ABCNG with ABC and five other improved ABC variants (GABC, ABCVSS, MEABC, EABC, EABC and ABCLGII). Experiment 3 further evaluates the performance of ABCNG on complex functions (CEC2013 benchmark). For the former two experiments, the population size *SN*, *limit*, *MaxFEs* and *runtime* are uniformly set to 50, $SN \cdot D$, $5000 \cdot D$, and 25, respectively. For the third experiment, Section 5.4 describes the specific parameter settings. All algorithms involved in this section are written in C language. The computing platform was with CPU Intel (R) Core (TM) i5-5200U 2.2 GHz, RAM 4 GB, and Microsoft Visual Studio 2010.

5.2. Effects of δ_1 and δ_2 in Gaussian perturbation

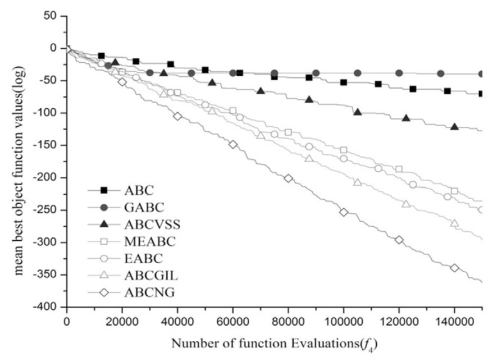
There are two parameters δ_1 and δ_2 in the new Gaussian perturbation, and they may affect the efficiency of the proposed ABCNG. δ_1 and δ_2 are selected from the evolutionary rates δ_i and δ_a. In order to clearly illustrate the effects of δ_1 and δ_2, ABCNG with four different combinations are tested on the first group functions. The detailed test combinations are listed as follows.

- ABCNG-*ii*: $\delta_1 = \delta_i$ and $\delta_2 = \delta_i$.
- ABCNG-*ia*: $\delta_1 = \delta_i$ and $\delta_2 = \delta_a$.
- ABCNG-*aa*: $\delta_1 = \delta_a$ and $\delta_2 = \delta_a$.
- ABCNG-*ai*: $\delta_1 = \delta_a$ and $\delta_2 = \delta_i$.

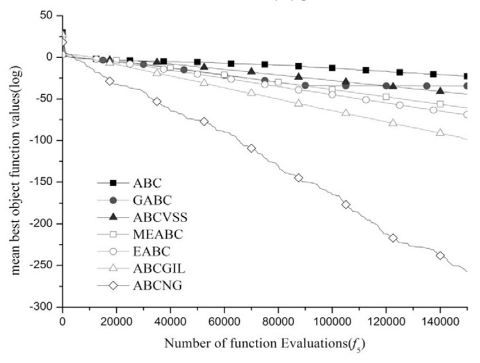
The performance of ABCNG-*ii*, ABCNG-*ia*, ABCNG-*aa*, and ABCNG-*ai* in different dimensions (D = 30,50 and 100) is given in Tables 1–3, respectively. From the results, the values of δ_1 and δ_2 can affect the performance of ABCNG. When the mean value of the Gaussian perturbation is δ_i ($\delta_1 = \delta_i$), ABCNG-*ia* performs better than ABCNG-*ii* on most test functions. It means that δ_a is more suitable than δ_i at the variance position of the Gaussian perturbation. When the variance value of the Gaussian perturbation is δ_i ($\delta_2 = \delta_i$), ABCNG-*ai* is much better than ABCNG-*ii* on unimodal functions. For multimodal functions, ABCNG-*ai* and ABCNG-*ii* have similar performance on functions f_{11}–f_{13}, f_{16}, and f_{19}–f_{21}, while the stability of ABCNG-*ai* is not as good as that



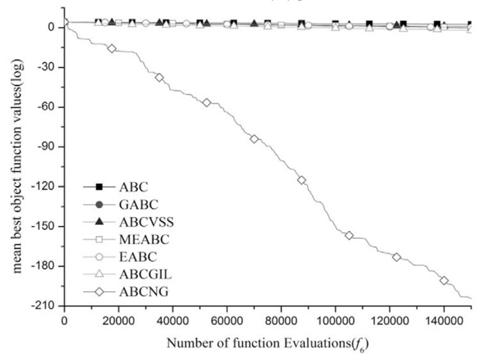
$(a)\, f_1$



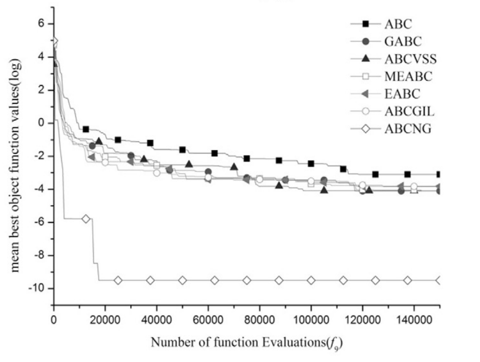
$(b)\, f_4$



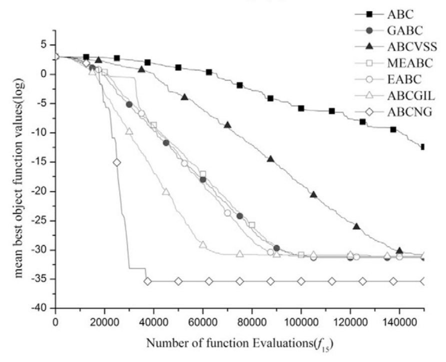
$(c)\, f_5$



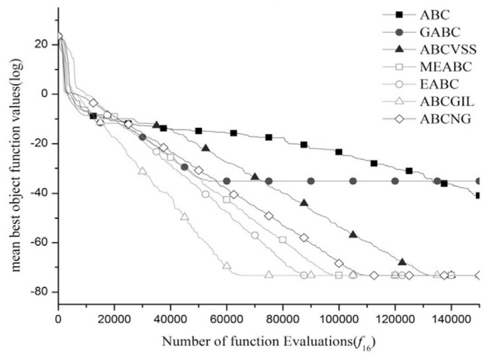
$(d)\, f_6$



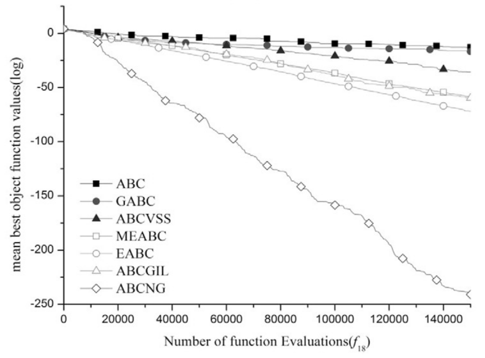
$(e)\, f_9$



$(f)\, f_{15}$



$(g)\, f_{16}$



$(h)\, f_{18}$

Fig. 2. Convergence curves of ABC, GABC, ABCVSS, MEABC, EABC, ABCGIL and ABCNG.

of ABCNG-*ii* on the rest of functions. When $\delta_2 = \delta_a$, ABCNG-*aa* surpasses ABCNG-*ia* on unimodal functions. For multimodal functions, ABCNG-*ia* is more stable than ABCNG-*aa*.

In order to select appropriate δ_1 and δ_2 in the Gaussian perturbation, Friedman test is used to calculate their mean ranking values on the first group functions [39]. Table 4 gives the mean ranking values of ABC, ABCNG-*ii*, ABCNG-*ia*, ABCNG-*aa*, and ABCNG-*ai*. As seen, ABCNG-*ia* on three types of dimensions achieves the best mean ranking values. Though ABCNG-*aa* and ABCNG-*ai* have better performances on unimodal functions, their stability on the whole benchmark set is not as good as ABCNG-*ia*. Therefore, ABCNG-*ia* ($\delta_1 = \delta_i$ and $\delta_2 = \delta_a$) is used in the following experiments.

Table 5
Results of ABCNG and six other ABC algorithms for *D*=30.

F	ABC		GABC		ABCVSS		MEABC		EABC		ABCLGII		ABCNG	
	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev
f_1	1.04×10^{-17}	1.20×10^{-17}	5.53×10^{-16}	6.02×10^{-16}	2.01×10^{-37}	1.07×10^{-36}	2.21×10^{-50}	8.02×10^{-50}	1.21×10^{-59}	7.95×10^{-59}	5.17×10^{-84}	6.36×10^{-83}	$\mathbf{2.88 \times 10^{-131}}$	$\mathbf{7.18 \times 10^{-130}}$
f_2	4.38×10^{-10}	4.72×10^{-10}	4.79×10^{-16}	4.22×10^{-16}	1.72×10^{-29}	1.22×10^{-28}	5.81×10^{-47}	3.88×10^{-46}	2.40×10^{-15}	7.09×10^{-14}	4.22×10^{-79}	4.51×10^{-78}	$\mathbf{6.11 \times 10^{-129}}$	$\mathbf{1.43 \times 10^{-127}}$
f_3	1.14×10^{-19}	9.89×10^{-20}	5.48×10^{-16}	5.36×10^{-16}	7.89×10^{-38}	2.58×10^{-37}	2.87×10^{-51}	1.26×10^{-50}	1.15×10^{-59}	8.40×10^{-59}	1.70×10^{-81}	4.84×10^{-80}	$\mathbf{1.29 \times 10^{-133}}$	$\mathbf{2.59 \times 10^{-132}}$
f_4	2.02×10^{-31}	5.30×10^{-31}	1.70×10^{-18}	4.24×10^{-18}	4.59×10^{-56}	9.99×10^{-55}	1.02×10^{-103}	2.08×10^{-102}	2.16×10^{-109}	6.37×10^{-108}	5.18×10^{-129}	1.27×10^{-128}	$\mathbf{1.72 \times 10^{-151}}$	$\mathbf{4.98 \times 10^{-150}}$
f_5	7.69×10^{-10}	3.04×10^{-10}	1.31×10^{-15}	6.91×10^{-16}	8.80×10^{-20}	1.57×10^{-19}	7.26×10^{-27}	1.37×10^{-26}	6.77×10^{-30}	2.84×10^{-30}	3.31×10^{-43}	1.77×10^{-42}	$\mathbf{1.79 \times 10^{-106}}$	$\mathbf{4.23 \times 10^{-105}}$
f_6	1.39×10^{1}	1.07×10^{1}	9.06×10^{-1}	8.42×10^{-1}	3.02×10^{0}	3.96×10^{0}	1.27×10^{0}	1.03×10^{0}	1.20×10^{0}	9.88×10^{-1}	1.38×10^{-1}	2.19×10^{-1}	$\mathbf{1.65 \times 10^{-83}}$	$\mathbf{4.77 \times 10^{-82}}$
f_7	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
f_8	$\mathbf{7.18 \times 10^{-66}}$	$\mathbf{5.92 \times 10^{-71}}$	$\mathbf{7.18 \times 10^{-66}}$	$\mathbf{1.73 \times 10^{-80}}$	$\mathbf{7.18 \times 10^{-66}}$	$\mathbf{1.73 \times 10^{-80}}$	$\mathbf{7.18 \times 10^{-66}}$	$\mathbf{1.73 \times 10^{-80}}$	$\mathbf{7.18 \times 10^{-66}}$	$\mathbf{1.73 \times 10^{-80}}$	$\mathbf{7.18 \times 10^{-66}}$	$\mathbf{1.73 \times 10^{-80}}$	$\mathbf{7.18 \times 10^{-66}}$	$\mathbf{1.73 \times 10^{-80}}$
f_9	4.52×10^{-2}	1.09×10^{-2}	1.65×10^{-2}	2.24×10^{-2}	1.69×10^{-2}	2.59×10^{-2}	1.67×10^{-2}	2.44×10^{-2}	2.22×10^{-2}	2.35×10^{-2}	1.97×10^{-2}	2.35×10^{-2}	$\mathbf{3.29 \times 10^{-5}}$	$\mathbf{1.58 \times 10^{-4}}$
f_{10}	5.45×10^{-2}	5.86×10^{-2}	1.42×10^{0}	2.07×10^{1}	$\mathbf{4.28 \times 10^{-2}}$	$\mathbf{3.43 \times 10^{-1}}$	1.06×10^{0}	1.51×10^{1}	1.23×10^{0}	2.82×10^{1}	4.83×10^{-1}	7.50×10^{0}	2.71×10^{1}	1.54×10^{0}
f_{11}	3.50×10^{-14}	1.35×10^{-13}	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
f_{12}	1.70×10^{-12}	4.36×10^{-12}	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
f_{13}	2.36×10^{-14}	5.62×10^{-14}	7.77×10^{-17}	2.79×10^{-16}	1.66×10^{-15}	4.78×10^{-14}	**0**	**0**	2.95×10^{-15}	8.71×10^{-14}	**0**	**0**	**0**	**0**
f_{14}	4.58×10^{-12}	1.59×10^{-12}	7.90×10^{0}	1.62×10^{2}	1.58×10^{-12}	3.39×10^{-12}	3.95×10^{0}	1.16×10^{2}	1.46×10^{-12}	3.99×10^{-12}	$\mathbf{1.09 \times 10^{-12}}$	$\mathbf{1.99 \times 10^{-12}}$	$1.76 \times$ 10–12	1.79×10^{-12}
f_{15}	4.31×10^{-6}	1.85×10^{-6}	3.01×10^{-14}	1.90×10^{-14}	5.27×10^{-14}	1.18×10^{-13}	3.01×10^{-14}	9.17×10^{-15}	2.91×10^{-14}	1.58×10^{-14}	3.40×10^{-14}	2.11×10^{-14}	$\mathbf{4.44 \times 10^{-16}}$	**0**
f_{16}	1.03×10^{-18}	6.90×10^{-19}	5.17×10^{-16}	4.31×10^{-16}	$\mathbf{1.57 \times 10^{-32}}$	$\mathbf{4.50 \times 10^{-47}}$	$\mathbf{1.57 \times 10^{-32}}$	$\mathbf{4.50 \times 10^{-47}}$	$\mathbf{1.57 \times 10^{-32}}$	$\mathbf{4.50 \times 10^{-47}}$	$\mathbf{1.57 \times 10^{-32}}$	$\mathbf{4.50 \times 10^{-47}}$	$\mathbf{1.57 \times 10^{-32}}$	$\mathbf{4.50 \times 10^{-47}}$
f_{17}	4.88×10^{-18}	5.03×10^{-18}	5.55×10^{-16}	5.04×10^{-16}	$\mathbf{1.35 \times 10^{-32}}$	$\mathbf{3.00 \times 10^{-47}}$	$\mathbf{1.35 \times 10^{-32}}$	$\mathbf{3.00 \times 10^{-47}}$	$\mathbf{1.35 \times 10^{-32}}$	$\mathbf{3.00 \times 10^{-47}}$	$\mathbf{1.35 \times 10^{-32}}$	$\mathbf{3.00 \times 10^{-47}}$	$\mathbf{1.35 \times 10^{-32}}$	$\mathbf{3.00 \times 10^{-47}}$
f_{18}	2.35×10^{-6}	1.66×10^{-6}	3.75×10^{-8}	5.64×10^{-8}	8.76×10^{-16}	1.64×10^{-15}	$\mathbf{1.46 \times 10^{-26}}$	$\mathbf{3.74 \times 10^{-25}}$	2.96×10^{-32}	4.13×10^{-31}	2.78×10^{-26}	6.32×10^{-25}	$\mathbf{2.99 \times 10^{-105}}$	$\mathbf{8.82 \times 10^{-104}}$
f_{19}	4.46×10^{-14}	5.39×10^{-14}	4.58×10^{-16}	4.69×10^{-16}	$\mathbf{1.35 \times 10^{-31}}$	$\mathbf{3.60 \times 10^{-46}}$	$\mathbf{1.35 \times 10^{-31}}$	$\mathbf{3.60 \times 10^{-46}}$	$\mathbf{1.35 \times 10^{-31}}$	$\mathbf{3.60 \times 10^{-46}}$	$\mathbf{1.35 \times 10^{-31}}$	$\mathbf{3.60 \times 10^{-46}}$	$\mathbf{1.35 \times 10^{-31}}$	$\mathbf{3.60 \times 10^{-46}}$
f_{20}	2.06×10^{-2}	2.35×10^{-2}	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
f_{21}	$\mathbf{-7.83} \times 10^{\mathbf{1}}$	**0**	$\mathbf{-7.83} \times 10^{\mathbf{1}}$	$\mathbf{6.51 \times 10^{-14}}$	$\mathbf{-7.83} \times 10^{\mathbf{1}}$	$\mathbf{6.03 \times 10^{-14}}$	$\mathbf{-7.83} \times 10^{\mathbf{1}}$	$\mathbf{6.36 \times 10^{-14}}$	$\mathbf{-7.83} \times 10^{\mathbf{1}}$	$\mathbf{5.68 \times 10^{-14}}$	$\mathbf{-7.83} \times 10^{\mathbf{1}}$	$\mathbf{2.01 \times 10^{-14}}$	$\mathbf{-7.83} \times 10^{\mathbf{1}}$	$\mathbf{6.36 \times 10^{-14}}$
f_{22}	-2.80×10^{1}	2.73×10^{-1}	-2.80×10^{1}	1.27×10^{0}	-2.83×10^{1}	1.40×10^{0}	-2.88×10^{1}	1.13×10^{0}	$\mathbf{-2.96 \times 10^{1}}$	$\mathbf{2.20 \times 10^{-1}}$	-2.87×10^{1}	1.49×10^{0}	-2.83×10^{1}	1.32×10^{0}
+/−/=	**18/1/3**		**15/1/6**		**10/2/10**		**10/2/10**		**10/3/9**		**9/3/10**		–	

Table 6
Results of ABCNG and six other ABC algorithms for *D*=50.

F	ABC		GABC		ABCVSS		MEABC		EABC		ABCLGII		ABCNG	
	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev
f_1	1.03×10^{-16}	2.46×10^{-16}	1.13×10^{-15}	2.46×10^{-16}	1.20×10^{-35}	3.56×10^{-36}	7.95×10^{-49}	2.19×10^{-48}	1.16×10^{-56}	5.21×10^{-56}	6.15×10^{-80}	7.49×10^{-79}	**8.68×10^{-180}**	**0**
f_2	8.93×10^{-16}	2.30×10^{-16}	8.93×10^{-16}	2.30×10^{-16}	1.14×10^{-28}	1.76×10^{-28}	1.62×10^{-45}	6.26×10^{-45}	5.03×10^{-51}	2.78×10^{-50}	1.57×10^{-76}	2.40×10^{-75}	**2.05×10^{-178}**	**0**
f_3	1.06×10^{-15}	1.89×10^{-16}	1.06×10^{-15}	1.89×10^{-16}	1.38×10^{-36}	2.14×10^{-36}	2.01×10^{-49}	1.25×10^{-48}	1.06×10^{-57}	3.15×10^{-57}	8.11×10^{-76}	2.39×10^{-74}	**7.94×10^{-180}**	**0**
f_4	1.90×10^{-17}	1.25×10^{-17}	1.90×10^{-17}	1.25×10^{-17}	1.39×10^{-53}	5.82×10^{-53}	5.51×10^{-109}	1.62×10^{-107}	3.54×10^{-100}	5.17×10^{-99}	2.07×10^{-118}	6.10×10^{-117}	**4.25×10^{-201}**	**0**
f_5	2.40×10^{-15}	2.20×10^{-16}	2.40×10^{-15}	2.20×10^{-16}	3.39×10^{-19}	1.22×10^{-19}	5.49×10^{-26}	1.21×10^{-25}	1.88×10^{-7}	5.54×10^{-6}	1.99×10^{-41}	2.42×10^{-40}	**1.67×10^{-147}**	**4.61×10^{-146}**
f_6	6.51×10^{0}	1.08×10^{0}	6.51×10^{0}	1.08×10^{0}	1.34×10^{1}	3.84×10^{0}	8.79×10^{0}	4.41×10^{0}	6.53×10^{0}	4.52×10^{0}	1.78×10^{0}	1.27×10^{0}	**1.35×10^{-48}**	**2.77×10^{-47}**
f_7	0	0	0	0	0	0	0	0	0	0	0	0	**0**	**0**
f_8	**2.67×10^{-109}**	1.43×10^{-116}	**2.67×10^{-109}**	1.43×10^{-118}	**2.67×10^{-109}**	**2.59×10^{-124}**	**2.67×10^{-109}**	**2.59×10^{-124}**	**2.67×10^{-109}**	**2.59×10^{-124}**	**2.67×10^{-109}**	**2.59×10^{-124}**	**2.67×10^{-109}**	**2.59×10^{-124}**
f_9	3.93×10^{-2}	7.61×10^{-3}	3.93×10^{-2}	7.61×10^{-3}	6.53×10^{-2}	2.01×10^{-2}	4.34×10^{-2}	3.26×10^{-2}	3.84×10^{-2}	3.85×10^{-2}	2.77×10^{-2}	2.96×10^{-2}	**2.64×10^{-4}**	**8.97×10^{-4}**
f_{10}	4.67×10^{0}	1.56×10^{1}	4.67×10^{0}	1.56×10^{1}	**9.05×10^{-2}**	**2.84×10^{-1}**	9.47×10^{-1}	1.29×10^{1}	6.73×10^{0}	1.01×10^{2}	7.73×10^{-1}	1.02×10^{1}	4.70×10^{1}	5.92×10^{-1}
f_{11}	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
f_{12}	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
f_{13}	3.55×10^{-16}	4.55×10^{-16}	3.55×10^{-16}	4.55×10^{-16}	**0**	**0**	**0**	**0**	**0**	**0**	2.47×10^{-4}	7.27×10^{-3}	**0**	**0**
f_{14}	2.71×10^{-10}	1.13×10^{-9}	2.71×10^{-10}	1.13×10^{-9}	**2.26×10^{-11}**	**7.97×10^{-12}**	3.55×10^{1}	2.97×10^{2}	2.29×10^{-4}	4.77×10^{-3}	4.74×10^{1}	3.59×10^{2}	2.09×10^{0}	1.62×10^{0}
f_{15}	5.73×10^{-14}	1.00×10^{-14}	5.73×10^{-14}	1.00×10^{-14}	1.11×10^{-13}	6.88×10^{-14}	3.66×10^{-2}	1.08×10^{0}	5.40×10^{-14}	3.48×10^{-14}	6.58×10^{-14}	2.17×10^{-14}	**4.44×10^{-16}**	**0**
f_{16}	1.00×10^{-15}	3.26×10^{-16}	1.00×10^{-15}	3.26×10^{-16}	**9.42×10^{-33}**	**0**	**9.42×10^{-33}**	**1.50×10^{-47}**	**9.42×10^{-33}**	**1.50×10^{-47}**	**9.42×10^{-33}**	**1.50×10^{-47}**	**9.42×10^{-33}**	**1.50×10^{-47}**
f_{17}	1.02×10^{-15}	1.37×10^{-16}	1.02×10^{-15}	1.37×10^{-16}	**1.35×10^{-32}**	**3.00×10^{-47}**	**1.35×10^{-32}**	**3.00×10^{-47}**	**1.35×10^{-32}**	**3.00×10^{-47}**	**1.35×10^{-32}**	**3.00×10^{-47}**	**1.35×10^{-32}**	**3.00×10^{-47}**
f_{18}	1.39×10^{-7}	6.03×10^{-7}	1.39×10^{-7}	6.03×10^{-7}	7.72×10^{-16}	2.90×10^{-15}	1.55×10^{-16}	3.61×10^{-15}	1.13×10^{-16}	3.34×10^{-15}	2.04×10^{-17}	6.00×10^{-16}	**2.21×10^{-136}**	**6.51×10^{-135}**
f_{19}	8.71×10^{-16}	2.09×10^{-16}	8.71×10^{-16}	2.09×10^{-16}	**1.35×10^{-31}**	**3.60×10^{-46}**	**1.35×10^{-31}**	**3.60×10^{-46}**	**1.35×10^{-31}**	**3.60×10^{-46}**	**1.35×10^{-31}**	**3.60×10^{-46}**	**1.35×10^{-31}**	**3.60×10^{-46}**
f_{20}	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
f_{21}	**-7.83×10^{1}**	**1.42×10^{-14}**	**-7.83×10^{1}**	**1.42×10^{-14}**	**-7.83×10^{1}**	**6.03×10^{-14}**	**-7.83×10^{1}**	**6.96×10^{-14}**	**-7.83×10^{1}**	**6.51×10^{-14}**	**-7.83×10^{1}**	**7.11×10^{-14}**	**-7.83×10^{1}**	**6.82×10^{-14}**
f_{22}	-4.55×10^{1}	6.61×10^{-1}	-4.55×10^{1}	6.61×10^{-1}	-4.61×10^{1}	9.85×10^{-1}	-4.67×10^{1}	1.50×10^{0}	**-4.96×10^{1}**	**1.59×10^{-1}**	-4.68×10^{1}	1.67×10^{0}	-4.59×10^{1}	2.09×10^{0}
+/−/=	**15/2/5**		**10/10/2**		**9/3/10**		**10/2/10**		**9/3/10**		**11/2/9**		–	

Table 7
Results of ABCNG and six other ABC algorithms for *D*=100.

F	ABC		GABC		ABCVSS		MEABC		EABC		ABCLGII		ABCNG	
	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev
f_1	4.78×10^{-16}	4.79×10^{-16}	2.34×10^{-15}	1.18×10^{-15}	6.90×10^{-35}	2.17×10^{-34}	1.88×10^{-46}	6.53×10^{-46}	1.54×10^{-54}	8.30×10^{-54}	3.96×10^{-77}	3.92×10^{-76}	**4.52×10^{-314}**	**0**
f_2	5.30×10^{-9}	4.89×10^{-9}	1.97×10^{-15}	8.34×10^{-16}	6.92×10^{-28}	3.33×10^{-27}	3.78×10^{-43}	1.43×10^{-42}	5.80×10^{-49}	2.48×10^{-48}	6.08×10^{-63}	1.79×10^{-61}	**3.15×10^{-299}**	**0**
f_3	1.03×10^{-16}	8.69×10^{-17}	2.25×10^{-15}	1.52×10^{-15}	2.30×10^{-35}	7.36×10^{-35}	6.81×10^{-47}	1.85×10^{-46}	3.53×10^{-15}	1.04×10^{-13}	6.50×10^{-77}	1.16×10^{-75}	**2.02×10^{-312}**	**0**
f_4	1.17×10^{-31}	4.89×10^{-31}	2.90×10^{-17}	4.91×10^{-17}	2.15×10^{-55}	2.44×10^{-54}	1.40×10^{-109}	4.11×10^{-108}	3.97×10^{-98}	1.17×10^{-96}	2.42×10^{-111}	2.42×10^{-111}	**5.92×10^{-319}**	**0**
f_5	1.27×10^{-9}	4.11×10^{-10}	5.17×10^{-15}	1.32×10^{-15}	1.07×10^{-18}	1.07×10^{-18}	1.03×10^{-24}	1.46×10^{-24}	1.20×10^{-28}	1.54×10^{-28}	1.72×10^{-34}	5.07×10^{-33}	**1.46×10^{-266}**	**0**
f_6	2.85×10^{1}	1.31×10^{0}	2.67×10^{-1}	8.49×10^{0}	3.83×10^{1}	9.91×10^{0}	3.05×10^{1}	9.59×10^{0}	2.59×10^{1}	6.78×10^{0}	1.12×10^{0}	7.35×10^{0}	**1.71×10^{-38}**	**5.04×10^{-37}**
f_7	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
f_8	**7.12×10^{-218}**	**0**	**7.12×10^{-218}**	**0**	**7.12×10^{-218}**	**0**	**7.12×10^{-218}**	**0**	**7.12×10^{-218}**	**0**	**7.12×10^{-218}**	**0**	**7.12×10^{-218}**	**0**
f_9	2.89×10^{-1}	5.49×10^{-2}	1.09×10^{-1}	7.39×10^{-2}	1.49×10^{-1}	9.03×10^{-2}	1.11×10^{-1}	2.44×10^{-2}	9.50×10^{-2}	5.60×10^{-2}	6.28×10^{-2}	4.56×10^{-2}	**1.55×10^{-4}**	**5.74×10^{-4}**
f_{10}	1.82×10^{-1}	2.04×10^{-1}	9.69×10^{-1}	2.13×10^{1}	1.29×10^{-1}	1.04×10^{0}	9.54×10^{-1}	7.48×10^{0}	5.93×10^{0}	9.78×10^{1}	1.27×10^{1}	1.45×10^{2}	9.65×10^{2}	8.53×10^{-1}
f_{11}	2.28×10^{-8}	1.14×10^{-7}	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
f_{12}	1.69×10^{-1}	3.97×10^{-1}	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
f_{13}	9.77×10^{-17}	1.90×10^{-16}	8.73×10^{-16}	2.12×10^{-15}	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
f_{14}	**5.79×10^{-11}**	**7.42×10^{-12}**	4.21×10^{1}	3.48×10^{2}	1.31×10^{-10}	2.81×10^{-11}	6.32×10^{1}	3.24×10^{2}	3.95×10^{0}	1.16×10^{2}	1.07×10^{2}	6.97×10^{2}	4.19×10^{1}	3.21×10^{2}
f_{15}	1.42×10^{-8}	4.33×10^{-9}	1.22×10^{-13}	4.44×10^{-14}	2.79×10^{-13}	4.90×10^{-13}	1.33×10^{-2}	3.93×10^{-1}	1.20×10^{-13}	4.23×10^{-14}	1.38×10^{-13}	5.64×10^{-14}	**4.44×10^{-16}**	**0**
f_{16}	1.48×10^{-17}	1.08×10^{-17}	2.41×10^{-15}	7.56×10^{-16}	**4.71×10^{-33}**	**7.50×10^{-48}**	**4.71×10^{-33}**	**7.50×10^{-48}**	**4.71×10^{-33}**	**7.50×10^{-48}**	**4.71×10^{-33}**	**7.50×10^{-48}**	**4.71×10^{-33}**	**7.50×10^{-48}**
f_{17}	3.27×10^{-16}	4.05×10^{-16}	2.36×10^{-15}	1.18×10^{-15}	**1.35×10^{-32}**	**3.00×10^{-47}**	**1.35×10^{-32}**	**3.00×10^{-47}**	**1.35×10^{-32}**	**3.00×10^{-47}**	**1.35×10^{-32}**	**3.00×10^{-47}**	**1.35×10^{-32}**	**3.00×10^{-47}**
f_{18}	3.96×10^{-4}	4.66×10^{-4}	5.73×10^{-6}	5.50×10^{-5}	1.40×10^{-15}	1.49×10^{-14}	9.07×10^{-17}	1.93×10^{-15}	2.74×10^{-16}	3.18×10^{-15}	7.18×10^{-16}	1.44×10^{-14}	**3.16×10^{-224}**	**0**
f_{19}	4.47×10^{-14}	5.35×10^{-14}	2.07×10^{-15}	1.05×10^{-15}	**1.35×10^{-31}**	**3.60×10^{-46}**	**1.35×10^{-31}**	**3.60×10^{-46}**	**1.35×10^{-31}**	**3.60×10^{-46}**	**1.35×10^{-31}**	**3.60×10^{-46}**	**1.35×10^{-31}**	**3.60×10^{-46}**
f_{20}	4.54×10^{-1}	1.24×10^{-1}	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
f_{21}	**-7.83×10^{1}**	**2.23×10^{-14}**	**-7.83×10^{1}**	**1.12×10^{-13}**	**-7.83×10^{1}**	**1.32×10^{-13}**	**-7.83×10^{1}**	**1.96×10^{-13}**	**-7.83×10^{1}**	**3.11×10^{-13}**	**-7.83×10^{1}**	**1.82×10^{-13}**	**-7.83×10^{1}**	**9.43×10^{-14}**
f_{22}	-8.32×10^{1}	4.53×10^{0}	-8.95×10^{1}	2.49×10^{0}	-8.98×10^{1}	3.33×10^{0}	-9.12×10^{1}	1.36×10^{0}	-9.23×10^{1}	2.93×10^{0}	**-9.24×10^{1}**	**3.29×10^{0}**	-8.95×10^{1}	2.73×10^{0}
+/−/=	**18/1/3**		**14/1/7**		**9/3/10**		**10/2/10**		**9/3/10**		**10/2/10**		–	

Table 8
Mean rank values of seven ABC algorithms achieved by Friedman test.

Algorithms	$D = 30$	$D = 50$	$D = 100$
ABC	6.02	5.07	5.64
GABC	5.11	5.11	5.05
ABCVSS	4.09	4.11	4.02
MEABC	3.59	4.02	3.84
EABC	3.52	3.61	3.58
ABCLGII	3.02	3.27	3.25
ABCNG	**2.64**	**2.80**	**2.73**

5.3. Comparison of ABCNG with other ABC algorithms

Experiment 2 is described in this section. For the first group functions with different dimensions (30, 50 and 100), ABC and other five ABCs are compared with ABCNG. The detailed ABCs are listed as follows.

- ABC [7].
- GABC [47].
- ABCVSS [52].
- MEABC [53].
- EABC [54].
- ABCLGII [55].
- Our approach ABCNG.

Table 5 shows the results of seven ABCs on $D = 30$, where the last row "+/−/=" summarizes the total comparison results. For the compared algorithm, ABCNG achieves better and worse solutions on "+" and "−" functions, respectively. Both of them have similar results on "=" functions. It can be seen that the accuracy of the solution obtained by ABCNG on most test functions is superior to other compared algorithms. Specifically, the performance of ABCNG on unimodal functions f_1–f_6 is much better than all competitors. For functions f_7–f_{10}, ABCNG is better or at least comparable to other ABCs. However, ANCNG is beaten by all competitors on the Rosenbrock function f_{10}. For multimodal functions f_{11}–f_{22}, ABCNG can find the global optima on f_{11}–f_{13} and f_{20}–f_{21}. In addition, ABCNG is obviously superior to all compared algorithms on f_{18}. For the Schwefel 2.26 function f_{14}, the performance of ABCNG is similar to ABCVSS, EABC, and ABCGIL. For the rest of functions, ABCNG is not worse than other ABCs. The overall comparison results exhibit that among the 22 functions ABCNG is better than ABC, GABC and ABCGIL on 18, 15 and 9 functions, respectively. ABCNG outperforms ABCVSS, MEABC and EABC on 10 functions.

Results of ABCNG and six other ABCs on $D = 50$ and 100 are presented in Tables 6 and 7, respectively. It aims to investigate the sensitivity of ABCNG on scalable dimensions. From those results, ABCNG still surpasses other ABCs on most test cases. For $D = 50$, the overall comparison results exhibit that among the 22 functions ABCNG superior to ABC and ABCGIL on 15 and 11 respectively. ABCNG is outperforms GABC, MEABC in 10 functions and ABCVSS, EABC in 9 functions. For $D = 100$, the corresponding results on the symbol " + " are 18, 14, 9, 10, 9 and 10.

Fig. 2 illustrates the convergence curves of five unimodal functions and three multimodal functions selected from the first set of test functions for seven ABCs ($D = 30$). As shown in the figure, none of the other six ABC algorithms converges as fast as ABCNG in both unimodal and multimodal functions.

Table 8 presets mean ranking values of seven ABCs under the Friedman test. ABCNG apparently ranks the best in all dimensions. It means ABCNG is superior to the other six algorithms in the comparison of the first group functions.

5.4. Results on CEC2013 benchmark set

Experiment 3 is presented in this section. A more complex test functions (CEC2013 benchmark) is used [55]. The benchmark set contains 28 test functions. In the experiments, ABC, GABC, ABCVSS, MEABC, EABC and ABCGIL are still used to compare with ABCNG. According to the requirements of the literature [56], $D = 30$ and $MaxFEs = 10000 \cdot D$ are used. All test functions in this test set are run independently 51 times. Other common parameters are set as in Section 5.1. The independent parameters of each algorithm are the same as in Section 5.3. The mean error results for each function are reported.

Table 9 exhibits the comparison of ABCNG and other ABC variants on the CEC2013 benchmark, where "ME" represents the mean error values. Among functions F_1–F_5, ABCNG has the best performance on F_2, F_4 and F_5, and it is inferior to EABC on F_1. For multimodal functions F_6–F_{20}, ABCNG finds the best solutions on F_6, F_7, F_{11}, F_{15}, F_{16}, F_{17}, and F_{19}, F_{20}. All the comparison algorithms nearly obtain the same results on F_8. For combinatorial functions F_{21}–F_{28}, ABCNG is superior to other compared algorithms on F_{23}, F_{24}, F_{25}, F_{26}. For the overall comparison results, ABCNG surpasses ABC, GABC, ABCVSS, MEABC, EABC and ABCGIL on 19, 24, 16, 21, 13 and 23 out of 28 functions, respectively. By analyzing the experimental results, ABCNG is not worse than other six ABC variants on the CEC2013 benchmark set.

6. Conclusion

In this paper, a new ABC variant called ABCNG is proposed. The new approach employs three modifications: (1) an adaptive method is used to dynamically update the neighborhood size; (2) a modified global best solution guided search strategy is constructed based on the neighborhood structure; and (3) a new Gaussian perturbation with evolutionary rate is designed. Performance of ABCNG is tested on two benchmark sets including classical functions and the complex CEC 2013 functions. Results of ABCNG are compared with ABC, GABC, ABCVSS, MEABC, EABC and ABCGIL.

To select the best parameters δ_1 and δ_2, four different parameter combinations are tested. Results show ABCNG-*ia* ($\delta_1 = \delta_i$ and $\delta_2 = \delta_a$) is more suitable than other parameters on the test set. For the classical test functions, ABCNG surpasses ABC, GABC, ABCVSS, MEABC, EABC and ABCGIL on $D = 30$, 50, and 100 according to the statistical results. For the complex CEC 2013 benchmark set, ABCBG is also superior to other compared algorithms.

Neighborhood search is significant to the performance of IOAs. To enhance the search efficiency of neighborhood search, a simple method is proposed to adjust the neighborhood size. Other strategies will be tried in the future. Moreover, the idea of this work may be applied to other IOAs.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This work was supported by the National Natural Science Foundation of China (No. 61663028).

Table 9
Comparison results on the CEC 2013 benchmark set.

Problems	ABC	GABC	MEABC	ABCVSS	EABC	ABCGIL	ABCNG
	ME	ME	ME	ME	ME	ME	ME
F_1	6.82×10^{-13}	5.71×10^{-13}	4.55×10^{-13}	8.19×10^{-13}	$\mathbf{2.61 \times 10^{-13}}$	3.64×10^{-13}	3.30×10^{-13}
F_2	2.44×10^{7}	3.43×10^{7}	1.72×10^{7}	2.47×10^{7}	2.27×10^{7}	2.14×10^{7}	$\mathbf{1.54 \times 10^{7}}$
F_3	2.49×10^{8}	$1.05 \times 10^{+10}$	$\mathbf{2.02 \times 10^{8}}$	4.92×10^{9}	3.33×10^{8}	3.95×10^{8}	9.63×10^{8}
F_4	7.34×10^{4}	3.51×10^{4}	9.45×10^{4}	6.45×10^{4}	8.73×10^{4}	8.22×10^{4}	$\mathbf{6.24 \times 10^{4}}$
F_5	4.97×10^{-10}	4.70×10^{-13}	4.55×10^{-13}	1.03×10^{-11}	4.32×10^{-13}	7.33×10^{-13}	$\mathbf{4.09 \times 10^{-13}}$
F_6	3.49×10^{1}	1.77×10^{2}	1.72×10^{1}	2.94×10^{1}	3.01×10^{1}	4.08×10^{1}	$\mathbf{2.63 \times 10^{1}}$
F_7	1.23×10^{2}	4.09×10^{2}	1.06×10^{2}	1.34×10^{2}	1.06×10^{2}	1.25×10^{2}	$\mathbf{9.09 \times 10^{1}}$
F_8	2.10×10^{1}	2.13×10^{1}	$\mathbf{2.09 \times 10^{1}}$	2.10×10^{1}	$\mathbf{2.09 \times 10^{1}}$	2.10×10^{1}	2.09×10^{1}
F_9	2.96×10^{1}	1.40×10^{2}	3.00×10^{1}	3.05×10^{1}	$\mathbf{2.83 \times 10^{1}}$	3.62×10^{1}	3.07×10^{1}
F_{10}	1.18×10^{0}	1.43×10^{0}	5.57×10^{0}	1.91×10^{1}	$\mathbf{7.53 \times 10^{-1}}$	1.09×10^{0}	1.69×10^{0}
F_{11}	3.30×10^{-13}	1.54×10^{-13}	1.14×10^{-13}	1.79×10^{-13}	5.97×10^{-14}	$\mathbf{5.68 \times 10^{-14}}$	$\mathbf{5.68 \times 10^{-14}}$
F_{12}	2.70×10^{2}	1.60×10^{3}	2.03×10^{2}	2.38×10^{2}	$\mathbf{1.31 \times 10^{2}}$	2.81×10^{2}	1.88×10^{2}
F_{13}	3.17×10^{2}	1.81×10^{2}	2.29×10^{2}	3.06×10^{2}	$\mathbf{1.97 \times 10^{2}}$	3.02×10^{2}	3.43×10^{2}
F_{14}	4.78×10^{0}	2.85×10^{0}	2.38×10^{1}	3.80×10^{1}	$\mathbf{9.16 \times 10^{-1}}$	2.58×10^{0}	3.11×10^{0}
F_{15}	5.10×10^{3}	1.57×10^{4}	5.12×10^{2}	5.28×10^{3}	4.88×10^{3}	5.78×10^{3}	$\mathbf{4.17 \times 10^{3}}$
F_{16}	$\mathbf{1.60 \times 10^{0}}$	2.07×10^{0}	1.65×10^{0}	2.11×10^{0}	1.83×10^{0}	1.88×10^{0}	$\mathbf{1.60 \times 10^{0}}$
F_{17}	3.11×10^{1}	1.07×10^{2}	$\mathbf{3.05 \times 10^{1}}$	3.18×10^{1}	$\mathbf{3.05 \times 10^{1}}$	4.80×10^{1}	$\mathbf{3.05 \times 10^{1}}$
F_{18}	3.88×10^{2}	1.76×10^{3}	$\mathbf{2.38 \times 10^{2}}$	3.61×10^{2}	3.67×10^{2}	4.24×10^{2}	2.45×10^{2}
F_{19}	3.07×10^{0}	2.25×10^{0}	8.73×10^{0}	3.43×10^{0}	3.69×10^{0}	3.58×10^{0}	$\mathbf{3.04 \times 10^{0}}$
F_{20}	1.44×10^{1}	5.00×10^{1}	1.67×10^{1}	1.44×10^{1}	$\mathbf{1.42 \times 10^{1}}$	1.50×10^{1}	$\mathbf{1.42 \times 10^{1}}$
F_{21}	2.01×10^{2}	3.67×10^{2}	$\mathbf{2.00 \times 10^{2}}$	2.77×10^{2}	2.31×10^{2}	6.55×10^{2}	3.72×10^{2}
F_{22}	9.16×10^{2}	7.47×10^{2}	2.15×10^{3}	3.43×10^{2}	$\mathbf{8.31 \times 10^{1}}$	1.90×10^{3}	1.16×10^{2}
F_{23}	5.53×10^{3}	2.16×10^{4}	5.16×10^{3}	6.17×10^{3}	5.19×10^{3}	7.36×10^{3}	$\mathbf{5.11 \times 10^{3}}$
F_{24}	3.02×10^{2}	6.00×10^{2}	2.82×10^{2}	2.89×10^{2}	2.84×10^{2}	3.08×10^{2}	$\mathbf{2.76 \times 10^{2}}$
F_{25}	3.15×10^{2}	7.16×10^{2}	3.06×10^{2}	3.02×10^{2}	2.98×10^{2}	3.36×10^{2}	$\mathbf{2.85 \times 10^{2}}$
F_{26}	2.01×10^{2}	2.07×10^{2}	$\mathbf{2.01 \times 10^{2}}$	2.02×10^{2}	$\mathbf{2.01 \times 10^{2}}$	2.35×10^{2}	$\mathbf{2.01 \times 10^{2}}$
F_{27}	4.14×10^{2}	3.81×10^{3}	$\mathbf{4.02 \times 10^{2}}$	4.22×10^{2}	5.63×10^{2}	1.31×10^{3}	9.00×10^{2}
F_{28}	3.64×10^{2}	4.27×10^{3}	$\mathbf{3.00 \times 10^{2}}$	4.67×10^{2}	$\mathbf{3.00 \times 10^{2}}$	3.17×10^{3}	4.17×10^{2}
+/-/=	**19/7/2**	**24/4/0**	**16/9/3**	**24/4/0**	**13/11/4**	**23/4/1**	–

References

[1] M.A. Islam, D.T. Anderson, F. Petry, P. Elmore, An efficient evolutionary algorithm to optimize the Choquet integral, Int. J. Intell. Syst. 34 (3) (2019) 366–385.

[2] F. Wang, Y. Li, F. Liao, H. Yan, An ensemble learning based prediction strategy for dynamic multi-objective optimization, Appl. Soft Comput. (2020) http://dx.doi.org/10.1016/j.asoc.2020.106592.

[3] R.P. Parouha, K.N. Das, Economic load dispatch using memory based differential evolution, Int. J. Bio-Inspir. Comput. 11 (3) (2018) 159–170.

[4] M.S. Rahman, A.K. Manna, A.A. Shaikh, A.K. Bhunia, An application of interval differential equation on a production inventory model with interval-valued demand via center-radius optimization technique and particle swarm optimization, Int. J. Intell. Syst. 35 (8) (2020) 1280–1326.

[5] Y. Xue, B. Xue, M. Zhang, Self-adaptive particle swarm optimization for large-scale feature selection in classification, ACM Trans. Knowl. Discov. Data 13 (5) (2019) 1–27.

[6] Y. Xue, T. Tang, W. Pang, A.X. Liu, Self-adaptive parameter and strategy based particle swarm optimization for large-scale feature selection problems with multiple classifiers, Appl. Soft Comput. 88 (2020) 106031.

[7] R. Jovanovic, M. Tuba, S. Voß, An efficient ant colony optimization algorithm for the blocks relocation problem, European J. Oper. Res. 274 (1) (2019) 78–90.

[8] D. Karaboga, B. Akay, A modified artificial bee colony (ABC) algorithm for constrained optimization problems, Appl. Soft Comput. 11 (3) (2011) 3021–3031.

[9] A. Banharnsakun, T. Achalakul, B. Sirinaovakul, The best-so-far selection in artificial bee colony algorithm, Appl. Soft Comput. 11 (2) (2011) 2888–2901.

[10] F. Wang, Y. Li, A. Zhou, K. Tang, An estimation of distribution algorithm for mixed-variable newsvendor problems, IEEE Trans. Evol. Comput. 24 (3) (2020) 479–493.

[11] H. Wang, W. Wang, Z. Cui, X. Zhou, J. Zhao, Y. Li, A new dynamic firefly algorithm for demand estimation of water resources, Inform. Sci. 438 (2018) 95–106.

[12] H. Wang, W. Wang, L. Cui, H. Sun, J. Zhao, Y. Wang, Y. Xue, A hybrid multi-objective firefly algorithm for big data optimization, Appl. Soft Comput. 69 (2018) 806–815.

[13] M. Elsisi, Optimal design of nonlinear model predictive controller based on new modified multitracker optimization algorithm, Int. J. Intell. Syst. 35 (11) (2020) 1857–1878.

[14] R. Devarapalli, B. Bhattacharyya, N.K. Sinha, An intelligent EGWO-SCA-CS algorithm for PSS parameter tuning under system uncertainties, Int. J. Intell. Syst. 35 (10) (2020) 1520–1569.

[15] H. Liu, B. Xu, D.J. Lu, G.J. Zhang, A path planning approach for crowd evacuation in buildings based on improved artificial bee colony algorithm, Appl. Soft Comput. 68 (2018) 360–376.

[16] J.Q. Li, Q.K. Pan, Solving the large-scale hybrid flow shop scheduling problem with limited buffers by a hybrid artificial bee colony algorithm, Inform. Sci. 316 (2015) 487–502.

[17] A. Benyoucef, A. Chouder, K. Kara, S. Silvestre, O.A. sahed, Artificial bee colony based algorithm for maximum power point tracking (MPPT) for PV systems operating under partial shaded conditions, Appl. Soft Comput. 32 (2015) 38–48.

[18] H. Peng, C. Deng, Z. Wu, Best neighbor guided artificial bee colony algorithm for continuous optimization problems, Soft Comput. 23 (18) (2019) 8723–8740.

[19] M. Tian, X. Gao, Differential evolution with neighborhood-based adaptive evolution mechanism for numerical optimization, Inform. Sci. 478 (2019) 422–448.

[20] X. Fu, F.T.S. Chan, B. Niu, N.S.H. Chung, T. Qu, A three-level particle swarm optimization with variable neighbourhood search algorithm for the production scheduling problem with mould maintenance, Swarm Evol. Comput. 50 (2019) 100572.

[21] J.A. Delgado-Osuna, M. Lozano, C. García-Martínez, An alternative artificial bee colony algorithm with destructive–constructive neighbourhood operator for the problem of composing medical crews, Inform. Sci. 326 (2016) 215–226.

[22] H. Wang, W. Wang, S. Xiao, Z. Cui, M. Xu, X. Zhou, Improving artificial bee colony algorithm using a new neighborhood selection mechanism, Inform. Sci. 527 (2020) 227–240.

[23] R. Lu, H. Hu, M. Xi, H. Gao, C.-M. Pun, An improved artificial bee colony algorithm with fast strategy, and its application, Comput. Electr. Eng. 78 (2019) 79–88.

[24] C. Dong, Z. Xiong, X. Liu, Y. Ye, Y. Yang, W. Guo, Dual-search artificial bee colony algorithm for engineering optimization, IEEE Access 7 (2019) 24571–24584.

[25] M.R. Chen, J.H. Chen, G.Q. Zeng, K.D. Lu, X.F. Jiang, An improved artificial bee colony algorithm combined with extremal optimization and Boltzmann selection probability, Swarm Evol. Comput. 49 (2019) 158–177.

[26] G.H. Li, L.Z. Cui, X.H. Fu, Z.K. Wen, N. Lu, J. Lu, Artificial bee colony algorithm with gene recombination for numerical function optimization, Appl. Soft Comput. 52 (2017) 146–159.

[27] X. Chen, H. Tianfield, K. Li, Self-adaptive differential artificial bee colony algorithm for global optimization problems, Swarm Evol. Comput. 45 (2019) 70–91.
[28] S.S. Jadon, J.C. Bansal, R. Tiwari, H. Sharma, Artificial bee colony algorithm with global and local neighborhoods, Int. J. Syst. Assur. Eng. Manag. 9 (3) (2014) 589–601.
[29] D.P. Kong, T.Q. Chang, W.J. Dai, Q.D. Wang, H.Z. Sun, An improved artificial bee colony algorithm based on elite group guidance and combined breadth-depth search strategy, Inform. Sci. 442–443 (2018) 54–71.
[30] X.Y. Song, M. Zhao, Q.F. Yan, S.Y. Xing, A high-efficiency adaptive artificial bee colony algorithm using two strategies for continuous optimization, Swarm Evol. Comput. 50 (2019) 100549.
[31] D. Bajer, B. Zorić, An effective refined artificial bee colony algorithm for numerical optimisation, Inform. Sci. 504 (2019) 221–275.
[32] M.D. Li, H. Zhao, X.W. Weng, H.Q. Huang, Artificial bee colony algorithm with comprehensive search mechanism for numerical optimization, J. Syst. Eng. Electron. 26 (3) (2015) 603–617.
[33] W.F. Gao, Z.F. Wei, Y.T. Luo, J. Cao, Artificial bee colony algorithm based on Parzen window method, Appl. Soft Comput. 74 (2019) 679–692.
[34] E. Saad, M.A. Elhosseini, A.Y. Haikal, Culture-based artificial bee colony with heritage mechanism for optimization of wireless sensors network, Appl. Soft Comput. 79 (2019) 59–73.
[35] W.F. Gao, H.L. Sheng, J. Wang, S.Y. Wang, Artificial bee colony algorithm based on novel mechanism for fuzzy portfolio selection, IEEE Trans. Fuzzy Syst. 27 (5) (2019) 966–978.
[36] L.Z. Cui, G.H. Li, Y.L. Luo, F. Chen, Z. Ming, N. Lu, J. Lu, An enhanced artificial bee colony algorithm with dual-population framework, Swarm Evol. Comput. 43 (2018) 184–206.
[37] G. Yavuz, D. Aydın, Improved self-adaptive search equation-based artificial bee colony algorithm with competitive local search strategy, Swarm Evol. Comput. 51 (2019) 100582.
[38] B. Alatas, Chaotic bee colony algorithms for global numerical optimization, Expert Syst. Appl. 37 (8) (2010) 5682–5687.
[39] X. Li, G. Yang, Artificial bee colony algorithm with memory, Appl. Soft Comput. 41 (2016) 362–372.
[40] C.Q. Zhang, J.G. Zheng, Y.Q. Zhou, Two modified artificial bee colony algorithms inspired by Grenade explosion method, Neurocomputing 151 (2015) 1198–1207.
[41] J.Y. Yang, Q.Y. Jiang, L. Wang, S. Liu, Y.D. Zhang, W. Li, B. Wang, An adaptive encoding learning for artificial bee colony algorithms, J. Comput. Sci. 30 (2019) 11–27.
[42] P. He, Z.L. Deng, C.Z. Gao, X.N. Wang, J. Li, Model approach to grammatical evolution: deep-structured analyzing of model and representation, Soft Comput. 21 (18) (2017) 5413–5423.
[43] P. He, Z.L. Deng, H.F. Wang, Z.S. Liu, Model approach to grammatical evolution: theory and case study, Soft Comput. 20 (9) (2016) 3537–3548.
[44] S. Das, A. Abraham, U.K. Chakraborty, A. Konar, Differential evolution using a neighborhood-based mutation operator, IEEE Trans. Evol. Comput. 13 (3) (2009) 526–553.
[45] H. Wang, H. Sun, C.H. Li, S. Rahnamayan, J.S. Pan, Diversity enhanced particle swarm optimization with neighborhood search, Inform. Sci. 223 (2013) 119–135.
[46] H. Wang, W.J. Wang, X.Y. Zhou, H. Sun, J. Zhao, X. Yu, Z.H. Cui, Firefly algorithm with neighborhood attraction, Inform. Sci. 382-383 (2017) 374–387.
[47] G. Zhu, S. Kwong, Gbest-guided artificial bee colony algorithm for numerical function optimization, Appl. Math. Comput. 217 (7) (2010) 3166–3173.
[48] W.F. Gao, F.T.S. Chan, L.L. Huang, S.Y. Liu, Bare bones artificial bee colony algorithm with parameter adaptation and fitness-based neighborhood, Inform. Sci. 316 (2015) 180–200.
[49] H. Wang, S. Rahnamayan, H. Sun, M.G. Omran, Gaussian bare-bones differential evolution, IEEE Trans. Cybern. 43 (2) (2013) 634–647.
[50] M. Zhang, N. Tian, V. Palade, Z.C. Ji, Y. Wang, Cellular artificial bee colony algorithm with Gaussian distribution, Inform. Sci. 462 (2018) 374–401.
[51] H. Wang, Z.J. Wu, S. Rahnamayan, Y. Liu, M. Ventresca, Enhancing particle swarm optimization using generalized opposition-based learning, Inform. Sci. 181 (20) (2011) 4699–4714.
[52] M.S. Kiran, H. Hakli, M. Gunduz, H. Uguz, Artificial bee colony algorithm with variable search strategy for continuous optimization, Inform. Sci. 300 (2015) 140–157.
[53] H. Wang, Z.J. Wu, S. Rahnamayan, H. Sun, Y. Liu, J.S. Pan, Multi-strategy ensemble artificial bee colony algorithm, Inform. Sci. 279 (2014) 587–603.
[54] W.F. Gao, S.Y. Liu, L.L. Huang, Enhancing artificial bee colony algorithm using more information-based search equations, Inform. Sci. 270 (2014) 112–133.
[55] Q.Z. Lin, M.M. Zhu, G.H. Li, W.J. Wang, L.Z. Cui, J.Y. Chen, J. Lu, A novel artificial bee colony algorithm with local and global information interaction, Appl. Soft Comput. 62 (2018) 702–735.
[56] J. Liang, B. Qu, P.J.C.I.L. Suganthan, Problem Definitions and Evaluation Criteria for the CEC 2014 Special Session and Competition on Single Objective Real-Parameter Numerical Optimization, N.T.U. Technical Report, Zhengzhou University, Zhengzhou China, Singapore, 2013.