Journal of Computational and Applied Mathematics 236 (2012) 2741–2753

Contents lists available at SciVerse ScienceDirect

Journal of Computational and Applied Mathematics

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

A global best artificial bee colony algorithm for global optimization

Weifeng Gao *, Sanyang Liu, Lingling Huang

Department of Applied Mathematics, Xidian University, Xi'an 710071, China

ARTICLE INFO

Article history:
Received 9 September 2010
Received in revised form 31 October 2011

Keywords:
Artificial bee colony algorithm
Initial population
Variant artificial bee colony algorithm
Search strategy

ABSTRACT

The artificial bee colony (ABC) algorithm is a relatively new optimization technique which has been shown to be competitive to other population-based algorithms. However, there is still an insufficiency in the ABC algorithm regarding its solution search equation, which is good at exploration but poor at exploitation. Inspired by differential evolution (DE), we propose a modified ABC algorithm (denoted as ABC/best), which is based on that each bee searches only around the best solution of the previous iteration in order to improve the exploitation. In addition, to enhance the global convergence, when producing the initial population and scout bees, both chaotic systems and opposition-based learning method are employed. Experiments are conducted on a set of 26 benchmark functions. The results demonstrate good performance of ABC/best in solving complex numerical optimization problems when compared with two ABC based algorithms.

Contents

1. Introduction

Optimization problems arise in a variety of fields, including engineering design, operational research, information science and related areas. Effective and efficient optimization algorithms are always needed to tackle increasingly complex real

* Corresponding author. Tel.: +86 02988201214; fax: +86 02988204396.
E-mail address: gaoweifeng2004@126.com (W. Gao).

0377-0427/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.cam.2012.01.013

world optimization problems. Stochastic optimization algorithms, such as genetic algorithm (GA) [1], particle swarm optimization (PSO) [2], ant colony optimization (ACO) [3], biogeography-based optimization (BBO) [4], harmony search (HS) [5], and artificial bee colony (ABC) algorithm [6], have been shown to be successful in dealing with many optimization problems.

In this paper, we concentrate on artificial bee colony algorithm, developed by Karaboga [6] based on simulating the foraging behavior of honey bee swarm. Numerical comparisons demonstrated that the performance of ABC algorithm is competitive to other population-based algorithms with an advantage of employing fewer control parameters [7–9]. Due to its simplicity and ease of implementation, ABC algorithm has captured much attention and has been applied to solve many practical optimization problems [10–12] since its invention in 2005.

However, similar to other evolutionary algorithms, ABC algorithm also faces up to some challenging problems. For example, the convergence speed of ABC algorithm is typically slower than those of representative population-based algorithms (e.g., differential evolution (DE) [13] and PSO) when handling those unimodal problems [9]. What is more, ABC algorithm can easily get trapped in the local optima when solving complex multimodal problems [9]. The reasons are as follows [14]. It is well known that both exploration and exploitation are necessary for a population-based optimization algorithm. In practice, the exploration and exploitation contradicts to each other. In order to achieve good performances on problem optimizations, the two abilities should be well balanced. While, we observed that the solution search equation of ABC algorithm which is used to generate new candidate solutions based on the information of previous solutions, is good at exploration but poor at exploitation, which results in the above two insufficiencies.

Therefore, accelerating convergence speed and avoiding the local optima have become two most important and appealing goals in ABC research. A number of variant ABC algorithms have, hence, been proposed to achieve these two goals [14–16]. However, so far, it is seen to be difficult to simultaneously achieve both goals. For example, the chaotic ABC algorithm (CABC3) in [16] focuses on avoiding the local optima, but brings in a more extra function evaluations in chaotic search as a result.

To achieve both goals, inspired by DE, we propose an improved ABC algorithm called ABC/best, which is based on that each bee searches only around the best solution of the previous iteration to improve the exploitation. In addition, to enhance the global convergence, when producing the initial population and scout bees, both chaotic systems and opposition-based learning method are employed. The rest of this paper is organized as follows. Section 2 summarizes ABC algorithm. The improved ABC algorithm called ABC/best algorithm is presented and analyzed in Section 3. Section 4 presents and discusses the experimental results. Finally, the conclusion is drawn in Section 5.

2. Artificial bee colony algorithm

The artificial bee colony consists of three groups of bees: employed bees, onlookers and scout bees. All bees which are currently exploiting a food source are known as employed. The employed bees exploit the food source, carry the information about food source back to the hive and share this information with onlooker bees. Onlookers bees are waiting in the hive for the information to be shared by the employed bees about their discovered food sources and scouts bees will always be searching for new food sources near the hive. Employed bees share information about food sources by dancing in the designated dance area inside the hive. The nature of dance is proportional to the nectar content of food source just exploited by the dancing bee. Onlooker bees watch the dance and choose a food source according to the probability proportional to the quality of that food source. Therefore, good food sources attract more onlooker bees compared to bad ones. Whenever a food source is exploited fully, all the employed bees associated with it abandon the food source, and become scout. Scout bees can be visualized as performing the job of exploration, whereas employed and onlooker bees can be visualized as performing the job of exploitation.

In ABC algorithm, each food source is a possible solution for the problem under consideration and the nectar amount of a food source represents the quality of the solution represented by the fitness value. The number of food sources is the same as the number of employed bees and there is exactly one employed bee for every food source.

An onlooker bee chooses a food source depending on the probability value P_i associated with that food source,

$$P_i = \frac{fit_i}{\sum_{i=1}^{SN} fit_i}, \tag{1}$$

where fit_i is the fitness value of solution i; SN is the number of food sources which is equal to the number of employed bees or onlooker bees.

In order to produce a candidate food position $\boldsymbol{V}_i = [v_{i,1}, v_{i,2}, \ldots, v_{i,D}]$ from the old one $\boldsymbol{X}_i = [x_{i,1}, x_{i,2}, \ldots, x_{i,D}]$ in memory, the ABC uses the following expression:

$$v_{i,j} = x_{i,j} + \Phi_{i,j}(x_{i,j} - x_{k,j}), \tag{2}$$

where $k \in \{1, 2, \ldots, SN\}$ and $j \in \{1, 2, \ldots, D\}$ are randomly chosen indexes; k has to be different from i; D is the number of variables (problem dimension); $\Phi_{i,j}$ is a random number in the range $[-1, 1]$.

After each candidate source position is produced and evaluated by the artificial bee, its performance is compared with that of the old one. If the new food source has an equal or better quality than the old source, the old one is replaced by the new one. Otherwise, the old one is retained.

If a position cannot be improved further through a predetermined number of cycles, then the food source is assumed to be abandoned. The value of predetermined number of cycles is an important control parameter of ABC algorithm, which is called *limit* for abandonment. Assume that the abandoned source is $\boldsymbol{X}_i$ and $j \in \{1, 2, \ldots, D\}$, then the scout discovers a new food source to be replaced with X_i. This operation can be defined as

$$x_{i,j} = x_{\min,j} + \text{rand}(0, 1)(x_{\max,j} - x_{\min,j}). \tag{3}$$

3. Variant artificial bee colony algorithm

3.1. Initial population

Population initialization is a crucial task in evolutionary algorithms because it can affect the convergence speed and the quality of the final solution. If no information about the solution is available, then random initialization is the most commonly used method to generate candidate solutions (initial population). This paper proposes a novel initialization approach which employs chaotic systems [16] and opposition-based learning method [17] which possess ergodicity, randomness and irregularity to generate initial population. Here, sinusoidal iterator is selected and its equation is defined as follows

$$ch_{k+1} = \sin(\pi ch_k), \quad ch_k \in (0, 1), k = 0, 1, 2, \ldots, K, \tag{4}$$

where k is the iteration counter, K is the preset maximum number of chaotic iterations.

Based on these operations, Gao and Liu [18] proposed the following algorithm to generate initial population which can be used instead of a pure random initialization.

Algorithm 3.1.

```
Set the maximum number of chaotic iteration K = 300, the population scale SN, and the
individual counter i = 1, j = 1.
For i = 1 to SN
    For j = 1 to D
      Randomly initialize variables ch_{0,j} ∈ (0, 1); set iteration counter k = 0.
      For k = 1 to K
          ch_{k+1,j} = sin(π ch_k)
      End
      P_{i,j} = x_{min,j} + ch_{k,j}(x_{max,j} − x_{min,j})
    End
End
Set the individual counter i = 1, j = 1.
For i = 1 to SN
    For j = 1 to D
        OP_{i,j} = x_{min,j} + x_{max,j} − P_{i,j}
    End
End
Selecting SN fittest individuals from set the {P(SN) ∪ OP(SN)} as initial population.
```

Remark 3.1. Algorithm 3.1 is also used by scout bees to discover a new food source.

Note that in order to find the existing region of the optimal solutions faster, the population should cover the whole search space better in the initialization period of the evolutionary algorithms. Specially, the population diversity ("Diversity" for short) is a measurement of the cover degree, which is defined as follows:

$$\text{Diversity} = \frac{1}{SN}\sum_{i=1}^{SN}\sqrt{\frac{1}{D}\sum_{j=1}^{D}\left(x_{i,j} - \bar{x}_j\right)^2}, \tag{5}$$

where SN denotes the number of food sources which is equal to the number of employed bees or onlooker bees, D is the number of variables (problem dimension), and $\bar{x}$ is the center position of the colony. This definition will be used in Section 4.2.1.

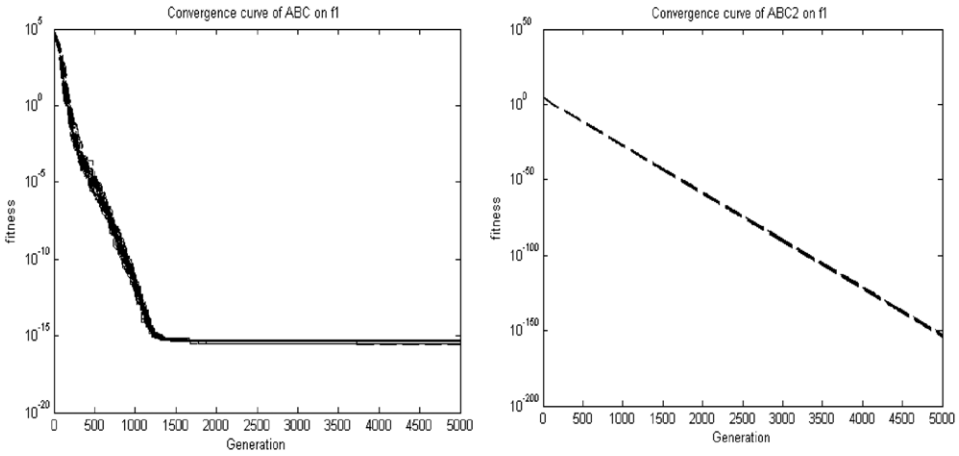

Fig. 1. The convergence characteristics of function f_1.

3.2. Two modified search strategies

Differential evolution (DE) [13] has been shown to be a simple yet efficient evolutionary algorithm for many optimization problems in real-world applications. It follows the general procedure of an evolutionary algorithm. After initialization, DE enters a loop of evolutionary operations: mutation, crossover, and selection. There are several variant DE algorithms which are different in that their mutation strategies are adopted differently. The following are different mutation strategies used in the literature:

$$\text{“DE/best/1”} : \boldsymbol{V}_i = \boldsymbol{X}_{\text{best}} + F(\boldsymbol{X}_{r1} - \boldsymbol{X}_{r2}), \tag{6}$$

$$\text{“DE/best/2”} : \boldsymbol{V}_i = \boldsymbol{X}_{\text{best}} + F(\boldsymbol{X}_{r1} - \boldsymbol{X}_{r2}) + F(\boldsymbol{X}_{r3} - \boldsymbol{X}_{r4}), \tag{7}$$

where $i = \{1, 2, \ldots, SN\}$ and $r1$, $r2$, $r3$ and $r4$ are mutually different random integer indices selected from $\{1, 2, \ldots, SN\}$. F, commonly known as scaling factor or amplification factor, is a positive real number, typically less than 1.0 that controls the rate at which the population evolves.

The best solutions in the current population are very useful sources that can be used to improve the convergence performance. The examples are the “*DE/best/1*” and “*DE/best/2*”, where the best solutions explored in the history are used to direct the movement of the current population. Based on these two variant DE algorithms and the property of ABC algorithm, the corresponding strategies in ABC algorithm are devised as follows

$$\text{“ABC/best/1”} : v_{i,j} = x_{\text{best},j} + \Phi_{i,j}(x_{r1,j} - x_{r2,j}), \tag{8}$$

$$\text{“ABC/ best/2”} : v_{i,j} = x_{\text{best},j} + \Phi_{i,j}(x_{r1,j} - x_{r2,j}) + \Phi_{i,j}(x_{r3,j} - x_{r4,j}), \tag{9}$$

where the indices $r1$, $r2$, $r3$ and $r4$ are mutually exclusive integers randomly chosen from $\{1, 2, \ldots, SN\}$, and different from the base index i. $\boldsymbol{X}_{\text{best}}$ is the best individual vector with the best fitness in the current population and $j \in \{1, 2, \ldots, D\}$ is randomly chosen indexes. $\Phi_{i,j}$ is a random number in the range $[-1, 1]$. In Eq. (2), the coefficient $\Phi_{i,j}$ is a uniform random number in $[-1, 1]$ and $x_{k,j}$ is a random individual in the population, therefore, the solution search dominated by Eq. (2) is random enough for exploration. In other words, the solution search equation described by Eq. (2) is good at exploration but poor at exploitation. However, according to Eq. (8) or Eq. (9), “*ABC/best*” can drive the new candidate solution only around the best solution of the previous iteration. Therefore, the modified solution search equation described by Eq. (8) or Eq. (9) can increase the exploitation of ABC algorithm.

Fig. 1 graphically presents the convergence characteristics of function f_1 obtained in the 30 independent runs by ABC and ABC2 (ABC only combined with the search strategies). It can be seen from Fig. 1 that in the later stage of evolution, ABC enters a long period of stagnation. While, ABC2 always keeps convergent fast. The reason is that the search equation of ABC is good at exploration but poor at exploitation. Exploitation ability is important for an optimization or search algorithm to converge fast and to refine the solution for high accuracy, especially in the later stage of evolution. The best solution X_{best} in the current population is very useful source which can be used to improve the convergence performance. That is the motivation why we propose the new search equations. With the help of X_{best}, the rest of the bees will follow it and converge to the near region of X_{best}. Therefore, we modify the solution search equation by applying the global best solution to guide the search of new candidate solutions in order to improve the exploitation. Fig. 2 shows the population distribution observed at various stages of ABC and ABC2, respectively. We can directly see from Fig. 2 that compared to ABC, ABC2 with the help of X_{best} can converge to global optimum rapidly. Specially, following the initialization, the bees of ABC start to explore throughout the search space. Then, the bees converge to the best solution very slowly. However, with the guidance of X_{best},

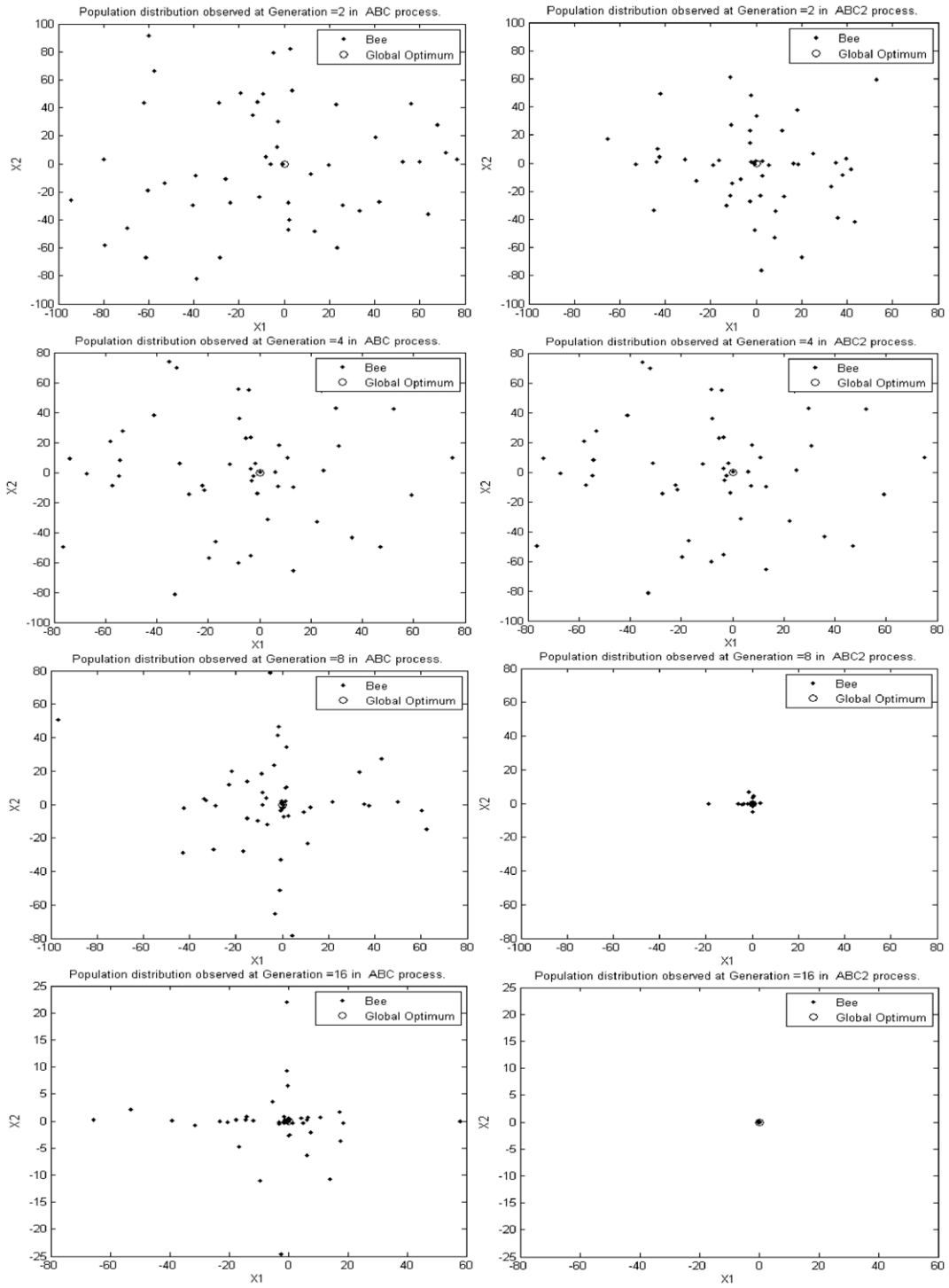

Fig. 2. Population distribution observed at various stages in ABC and ABC2.

ABC2 pull many bees to swarm together toward the optimal region. Then, the population converges to the global optimum quickly.

3.3. Two variant artificial bee colony algorithms (ABC/best)

Having discussed the proposed initialization and the search equations, we combine them and propose Algorithm 3.2. Due to the search equations play the crucial role in the algorithms, so we name the relevant algorithms after the

search equations. The complete computational procedure is outlined as follows:

Algorithm 3.2 (*ABC/Best Algorithm*)**.**

01 Initialization: Preset population size SN and $limit$.
02 Perform Algorithm 3.1 to create an initial population $\boldsymbol{P} = \{\boldsymbol{X}_i | i = 1, 2, \ldots, SN\}$, calculate the function values of the population $F = \{F_i | i = 1, 2, \ldots, SN\}$.
03 For $g = 1$ to G
04 For $i = 1$ to SN
05 Randomly choose $\boldsymbol{X}_{r1} \neq \boldsymbol{X}_i, \boldsymbol{X}_{r2} \neq \boldsymbol{X}_i, \boldsymbol{X}_{r2} \neq \boldsymbol{X}_{r1}$ from current population $\boldsymbol{P}$.
06 Randomly choose j from $\{1, 2, \ldots, D\}$.
07 $v_{i,j} = x_{\text{best},j} + \Phi_{i,j}(x_{r1,j} - x_{r2,j})$
08 If $F(\boldsymbol{V}_i) \leq F(\boldsymbol{X}_i)$
09 $\boldsymbol{X}_i = \boldsymbol{V}_i$, $trial_i = 1$
10 Else
11 $trial_i = trial_i + 1$
12 End
13 If $F > 0$
14 $fit_i = 1/(1 + F_i)$
15 Else
16 $fit_i = 1 + \text{abs}(F_i)$
17 End
18 End
19 Calculate the probability values P_i for the solutions $\boldsymbol{X}_i$ by (1) $(i = 1, 2, \ldots, SN)$.
20 Apply roulette wheel selection method to select SN onlookers $\boldsymbol{P1} = \{\boldsymbol{X}_{s(i)} | i = 1, 2, \ldots, SN\}$ from the employed bees $\boldsymbol{P}$.
21 For $i = 1$ to SN
22 Randomly choose $\boldsymbol{X}_{r1} \neq \boldsymbol{X}_{s(i)}, \boldsymbol{X}_{r2} \neq \boldsymbol{X}_{s(i)}, \boldsymbol{X}_{r2} \neq \boldsymbol{X}_{r1}$ from current population $\boldsymbol{P1}$.
23 Randomly choose j from $\{1, 2, \ldots, D\}$.
24 $v_{i,j} = x_{\text{best},j} + \Phi_{i,j}(x_{r1,j} - x_{r2,j})$
25 If $F(\boldsymbol{V}_i) \leq F(\boldsymbol{X}_{s(i)})$
26 $\boldsymbol{X}_{s(i)} = \boldsymbol{V}_i$, $trial_{s(i)} = 1$
27 Else
28 $trial_{s(i)} = trial_{s(i)} + 1$
29 End
30 End
31 If $\max(trial_i) > limit$
32 Replace $\boldsymbol{X}_i$ with a new produced solution by Algorithm 3.1.
33 End
34 Memorize the best solution achieved so far.
35 End

Remark 3.2. The solution search equation of ABC/best/1 algorithm is described by Eq. (7). While, that of ABC/best/2 algorithm is described by Eq. (8).

4. Experiments

4.1. Benchmark functions and parameter settings

In order to test the performance of ABC/best algorithm, it is applied to minimize a set of 15 scalable benchmark functions of dimensions $D = 30$ or 60 [9,15], 2 functions of dimensions $D = 100$ or 200 [17] and a set of functions of lower dimensions $D = 3$ or 4 [9], as shown in Table 1.

Some comparative experiments on numerical function optimization have been conducted for ABC algorithm in [7–9]. The experimental results show that ABC algorithm is competitive with some conventional optimization algorithms, such as GA,

Table 1
Benchmark functions used in experiments.

Test functions	D	Search range	Optimum value
$f_1(X) = \sum_{i=1}^{D} x_i^2$	30 and 60	$(-100, 100)^D$	0
$f_2(X) = \sum_{i=1}^{D} \lvert x_i\rvert + \prod_{i=1}^{D} \lvert x_i\rvert$	30 and 60	$(-10, 10)^D$	0
$f_3(X) = \max_i\{\lvert x_i\rvert, 1 \le i \le D\}$	30 and 60	$(-100, 100)^D$	0
$f_4(X) = \sum_{i=1}^{D} (\lfloor x_i + 0.5\rfloor)^2$	30 and 60	$(-100, 100)^D$	0
$f_5(X) = \sum_{i=1}^{D} ix_i + \text{rand}[0, 1)$	30 and 60	$(-1.28, 1.28)^D$	0
$f_6(X) = \sum_{i=1}^{D} [x_i^2 - 10\cos(2\pi x_i) + 10]$	30 and 60	$(-5.12, 5.12)^D$	0
$f_7(X) = \sum_{i=1}^{D} [y_i^2 - 10\cos(2\pi y_i) + 10]$ where $y_i = \begin{cases} x_i & \lvert x_i\rvert < 0.5 \\ \dfrac{\text{round}(2x_i)}{2} & \lvert x_i\rvert \ge 0.5 \end{cases}$	30 and 60	$(-5.12, 5.12)^D$	0
$f_8(X) = \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	30 and 60	$(-600, 600)^D$	0
$f_9(X) = D * 418.982887 - \sum_{i=1}^{D} \left(x_i \sin\left(\sqrt{\lvert x_i\rvert}\right)\right)$	30 and 60	$(-500, 500)^D$	0
$f_{10}(X) = -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos 2\pi x_i\right) + 20 + e$	30 and 60	$(-32, 32)^D$	0
$f_{11}(X) = \frac{\pi}{D}\left\{10\sin^2(\pi y_1) + \sum_{i=1}^{D-1}(y_i - 1)^2[1 + \sin^2(\pi y_{i+1})] + (y_D - 1)^2\right\} + \sum_{i=1}^{D} u(x_i, 10, 100, 4)$	30 and 60	$(-50, 50)^D$	0
$f_{12}(X) = 0.1\left\{\sin^2(3\pi x_1) + \sum_{i=1}^{D-1}(x_i - 1)^2[1 + \sin^2(3\pi x_{i+1})] + (x_D - 1)^2[1 + \sin^2(3\pi x_D)]\right\} + \sum_{i=1}^{D} u(x_i, 5, 100, 4)$	30 and 60	$(-50, 50)^D$	0
$f_{13}(X) = \sum_{i=1}^{D} \lvert x_i \cdot \sin(x_i) + 0.1 \cdot x_i\rvert$	30 and 60	$(-10, 10)^D$	0
$f_{14}(X) = \sum_{i=1}^{D-1}(x_i - 1)^2[1 + \sin^2(3\pi x_{i+1})] + \sin^2(3\pi x_1) + \lvert x_D - 1\rvert[1 + \sin^2(3\pi x_D)]$	30 and 60	$(-10, 10)^D$	0
$f_{15}(X) = \sum_{i=1}^{D}\left(\sum_{k=0}^{k\max}\left[a^k\cos\left(2\pi b^k(x_i + 0.5)\right)\right]\right) - D\sum_{k=0}^{k\max}\left[a^k\cos\left(2\pi b^k \cdot 0.5\right)\right]$ where $a = 0.5$, $b = 3$, $kamx = 20$.	30 and 60	$(-0.5, 0.5)^D$	0
$f_{16}(X) = \frac{1}{n}\sum_{i=1}^{D}\left(x_i^4 - 16x_i^2 + 5x_i\right)$	100 and 200	$(-5, 5)^D$	-78.33236 for $D = 100$
$f_{17}(X) = -\sum_{i=1}^{D}\sin(x_i)\sin^{20}\left(\frac{i\times x_i^2}{\pi}\right)$	100 and 200	$(0, \pi)^D$	-99.2084 for $D = 100$
$f_{18}(X) = \sum_{i=1}^{10}\left(a_i - \frac{x_1\left(b_i^2 + b_i x_2\right)}{b_i^2 + b_i x_3 + x_4}\right)^2$	4	$(-5, 5)^D$	0.0003075
$f_{19}(X) = 100\left(x_1^2 - x_2\right)^2 + (x_1 - 1)^2 + (x_3 - 1)^2 + 90\left(x_3^2 - x_4\right)^2 + 10.1\left((x_2 - 1)^2 + (x_4 - 1)^2\right) + 19.8(x_2 - 1)(x_4 - 1)$	4	$(-10, 10)^D$	0
$f_{20}(X) = \sum_{k=1}^{D}\left[\left(\sum_{i=1}^{D} x_i^k\right) - b_k\right]^2$	4	$(-0, D)^D$	0
$f_{21}(X) = \left(\sum_{i=1}^{D} x_j^2\right) + \left(\sum_{i=1}^{D} 0.5ix_i\right)^2 + \left(\sum_{i=1}^{D} 0.5ix_i\right)^4$	3 and 4	$(-5, 5)^D$	0
$f_{22}(X) = 0.5 + \frac{\sin^2\left(\sqrt{\sum_{i=1}^{D} x_i^2}\right) - 0.5}{\left(1 + 0.001\left(\sum_{i=1}^{D} x_i^2\right)\right)^2}$	3 and 4	$(-100, 100)^D$	0
$f_{23}(X) = \cos\left(2\pi\sqrt{\sum_{i=1}^{N} x_i^2}\right) + 0.1\sqrt{\sum_{i=1}^{N} x_i^2} + 1$	3 and 4	$(-100, 100)^D$	0
$f_{24}(X) = \sum_{i=1}^{D-1}\left[100\left(x_i^2 - x_{i+1}\right)^2 + (x_i - 1)^2\right]$	3 and 4	$(-30, 30)^D$	0
$f_{25}(X) = -\sum_{i=1}^{D}\left[(X - a_i)(X - a_i)^T + c_i\right]^{-1}$	7 and 10	$(0, 10)^D$	-10.4 and 10.5
$f_{26}(X) = -\sum_{i=1}^{4} c_i \exp\left[\sum_{j=1}^{D} a_{ij}\left(x_j - p_{ij}\right)^2\right]$	3 and 6	$(0, 1)^D$	-3.86 and -3.32

DE and PSO. In this section, a set of experiments tested on 26 numerical benchmark function are performed to compare the performance of ABC/best algorithm with that of ABC algorithm. The population size is 100 which is the same as that in [8], *limit* is $0.6 * SN * D$ and the maximum number of generations is listed in Table 3. Each of the experiments is repeated 30 times independently. And the reported results are the means and standard deviations of the statistical experimental data. For clarity, the results of the best algorithm are marked in boldface, respectively; if not all algorithms produce identical results. All the algorithms are coded in Matlab 7.0 and the simulations are run on a Pentium IV 2.4 GHz with 512 MB memory capacity. And you can get the source code of ABC/best from me by Email.

4.2. Experimental results

4.2.1. Performance comparison between the proposed initialization with the random initialization

In order to illustrate the randomness and sensitivity dependence on the initial conditions of Eq. (4), we present Figs. 3 and 4, which show the distribution of the chaotic sequence. As is shown in Fig. 3, the chaotic sequence is relatively scattered,

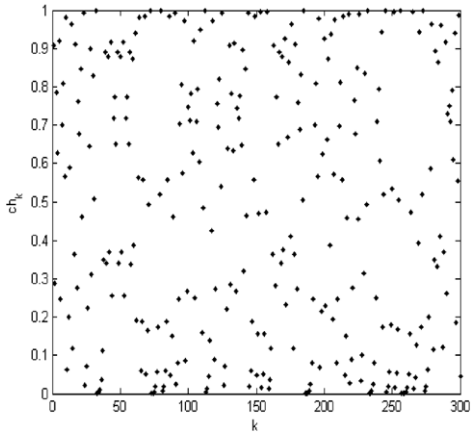

Fig. 3. The relationship between k and ch_k.

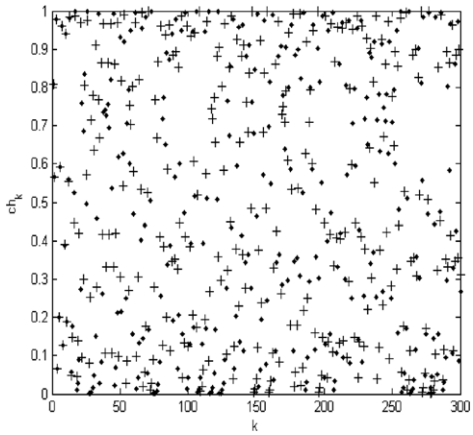

Fig. 4. The sensitivity to initial condition.

Table 2
Performance comparison of random initialization and proposed initialization.

Fun	Random initialization		Proposed initialization	
	Fitness	Diversity	Fitness	Diversity
f_1	1.0254e+05	56.7520	8.6928e+04	67.8944
f_2	9.3031e+19	5.7977	5.4117e+15	6.8821
f_3	97.5630	58.3586	94.0856	69.9976
f_4	1.0042e+05	56.6417	9.1010e+04	67.9087
f_5	256.4803	0.7280	190.4312	0.8618
f_6	549.4686	2.9257	518.3174	3.5084
f_7	563.2750	2.9521	483.3585	3.5343
f_8	898.7446	342.3789	812.9512	405.7734
f_9	1.2830e+04	281.3224	1.1720e+04	352.6507
f_{10}	21.1550	18.1921	21.0200	22.2830
f_{11}	6.2992e+08	28.7773	3.7820e+08	34.0883
f_{12}	1.1253e+09	28.2784	8.0963e+08	34.2690
f_{13}	85.3119	5.4509	77.4637	6.8258
f_{14}	1.4741e+03	5.6438	1.2560e+03	6.9533
f_{15}	30.1261	0.2819	28.2750	0.3349

and almost traverses the whole interval [0, 1] when k gets big enough. In Fig. 4, '+' indicates the initial value of the sequence is set to 0.7. While '.' stands for the initial value of the sequence is set to 0.700001. It can be seen from Fig. 4 that even a small difference between the initial values (only 0.000001) can exert non-ignorable influence on the sequence.

Next, we compare the performance of the proposed initialization with the random initialization in terms of Fitness and Diversity. As all test functions are minimization problems, the smaller the Fitness, the better it is. While Diversity is the opposite. The results are shown in Table 2. So, it can be seen from Table 2 that the Mean and Diversity of the proposed

Table 3
Performance comparison of ABC, ABC/best/2 and ABC/best/1.

Fun	D	G	ABC		ABC/best/2		ABC/best/1	
			Mean	SD	Mean	SD	Mean	SD
f_1	30	1000	6.99e−10	5.91e−10	4.37e−22	2.14e−22	**1.57e−27**	**1.14e−27**
	60	2000	1.94e−09	8.33e−10	1.57e−20	4.90e−21	**2.42e−25**	**1.09e−25**
f_2	30	1000	2.36e−06	8.32e−07	2.73e−12	4.61e−13	**3.45e−15**	**8.79e−16**
	60	2000	8.30e−06	8.93e−07	2.14e−11	4.39e−12	**5.94e−14**	**1.28e−14**
f_3	30	1000	1.70e+01	1.95e−00	7.89e−00	1.77e−00	**7.02e−00**	**9.94e−01**
	60	2000	3.79e+01	3.66e−00	**3.34e+01**	**2.45e−00**	3.41e+01	2.89e−00
f_4	30	1000	0	0	0	0	0	0
	60	2000	0	0	0	0	0	0
f_5	30	1000	1.01e−01	2.44e−02	3.72e−02	1.73e−02	**3.20e−02**	**6.03e−03**
	60	2000	2.58e−01	2.92e−02	1.02e−01	1.35e−02	**9.91e−02**	**1.07e−02**
f_6	30	1000	6.63e−03	1.71e−02	**0**	**0**	**0**	**0**
	60	2000	3.03e−01	4.53e−01	**0**	**0**	**0**	**0**
f_7	30	1000	4.12e−01	4.81e−01	**0**	**0**	**0**	**0**
	60	2000	2.55e−00	1.45e−00	**0**	**0**	**0**	**0**
f_8	30	1000	8.73e−09	1.47e−08	4.47e−08	1.05e−07	**4.23e−11**	**2.16e−11**
	60	2000	4.46e−09	6.68e−09	2.18e−10	4.33e−10	**0**	**0**
f_9	30	1000	2.05e+02	1.63e+02	**0**	**0**	**0**	**0**
	60	2000	6.93e+02	1.39e+02	3.67e−11	3.17e−12	**3.74e−11**	**2.59e−12**
f_{10}	30	1000	1.02e−05	4.15e−06	1.89e−11	4.75e−12	**1.26e−13**	**3.48e−14**
	60	2000	2.05e−05	5.54e−06	1.08e−10	4.97e−12	**3.40e−13**	**6.35e−14**
f_{11}	30	1000	1.60e−11	1.95e−11	9.84e−24	9.84e−24	**2.85e−30**	**2.19e−30**
	60	2000	1.18e−10	8.99e−11	9.92e−23	2.82e−23	**2.89e−28**	**1.54e−28**
f_{12}	30	1000	3.72e−09	1.79e−09	2.17e−22	1.55e−22	**3.88e−29**	**1.57e−29**
	60	2000	1.06e−08	6.25e−09	5.54e−21	1.35e−21	**2.06e−26**	**5.74e−27**
f_{13}	30	1000	1.38e−04	6.38e−05	7.34e−10	1.01e−09	**1.32e−14**	**5.04e−15**
	60	2000	1.32e−03	1.21e−03	2.68e−08	1.84e−08	**4.46e−13**	**4.13e−13**
f_{14}	30	1000	3.21e−09	4.41e−09	3.58e−21	5.69e−21	**6.05e−28**	**3.30e−28**
	60	2000	2.35e−09	2.35e−09	5.28e−21	2.64e−21	**3.29e−26**	**1.29e−26**
f_{15}	30	1000	1.52e−01	2.65e−02	6.32e−05	4.44e−05	**2.84e−15**	**2.13e−15**
	60	2000	4.32e−01	3.33e−02	7.76e−04	4.28e−04	**3.17e−12**	**4.59e−12**
f_{16}	100	1000	−77.3547	1.59e−01	−78.3323	1.13e−05	**−78.3323**	**9.77e−07**
	200	2000	−77.1908	2.64e−01	−78.3322	2.23e−05	**−78.3323**	**2.50e−06**
f_{17}	100	1000	−83.3160	2.90e−01	−84.9950	4.40e−01	**−90.3619**	**3.54e−01**
	200	2000	−163.4126	9.91e−01	−165.5409	7.12e−01	**−174.0796**	**8.74e−01**
f_{18}	4	2000	4.31e−04	4.51e−05	4.13e−04	7.48e−05	**3.16e−04**	**5.33e−06**
f_{19}	4	2000	1.17e−01	6.94e−02	2.41e−02	2.18e−02	**2.14e−03**	**2.21e−03**
f_{20}	4	2000	6.67e−03	5.98e−03	9.24e−03	5.21e−03	**3.22e−03**	**3.01e−03**
f_{21}	3	1000	9.65e−74	1.93e−73	7.1e−120	1.4e−119	**4.9e−216**	**0**
	4	2000	5.03e−61	9.96e−61	5.87e−06	1.17e−06	**2.04e−63**	**6.11e−63**
f_{22}	3	1000	2.07e−03	3.91e−03	3.55e−05	5.63e−05	**2.57e−12**	**1.12e−11**
	4	2000	4.06e−02	4.84e−02	1.35e−03	3.27e−03	**2.51e−06**	**7.54e−06**
f_{23}	3	1000	1.28e−06	3.79e−06	5.75e−05	3.92e−05	**0**	**0**
	4	2000	3.39e−05	2.80e−04	3.92e−04	7.70e−05	**0**	**0**
f_{24}	3	1000	3.93e−02	3.11e−02	1.95e−03	1.52e−03	**9.06e−06**	**1.41e−05**
	4	2000	3.21e−02	3.26e−02	1.86e−03	1.03e−03	**1.29e−07**	**3.83e−07**
f_{25}	4	1000	−10.4029	1.58e−15	−10.4029	1.77e−15	−10.4029	**1.12e−15**
	4	1000	−10.5364	3.39e−14	−10.5364	1.65e−15	−10.5364	**1.58e−15**
f_{26}	3	1000	−3.8628	2.11e−15	−3.8628	**8.88e−16**	−3.8628	**8.88e−16**
	6	1000	−3.3220	4.44e−16	−3.3220	4.44e−16	−3.3220	4.44e−16

initialization is better than the random initialization for all the test functions. In a word, by combining the advantages of chaotic systems and opposition-based learning method, the proposed initialization can increase the population diversity and obtain good initial solutions.

4.2.2. Performance comparison between the proposed algorithms with two ABC based algorithms

The performance on the solution accuracy of ABC is compared with ABC/best. The results are shown in Table 3 in terms of the mean and SD of the solutions obtained in the 30 independent runs by each algorithm. Fig. 5 graphically

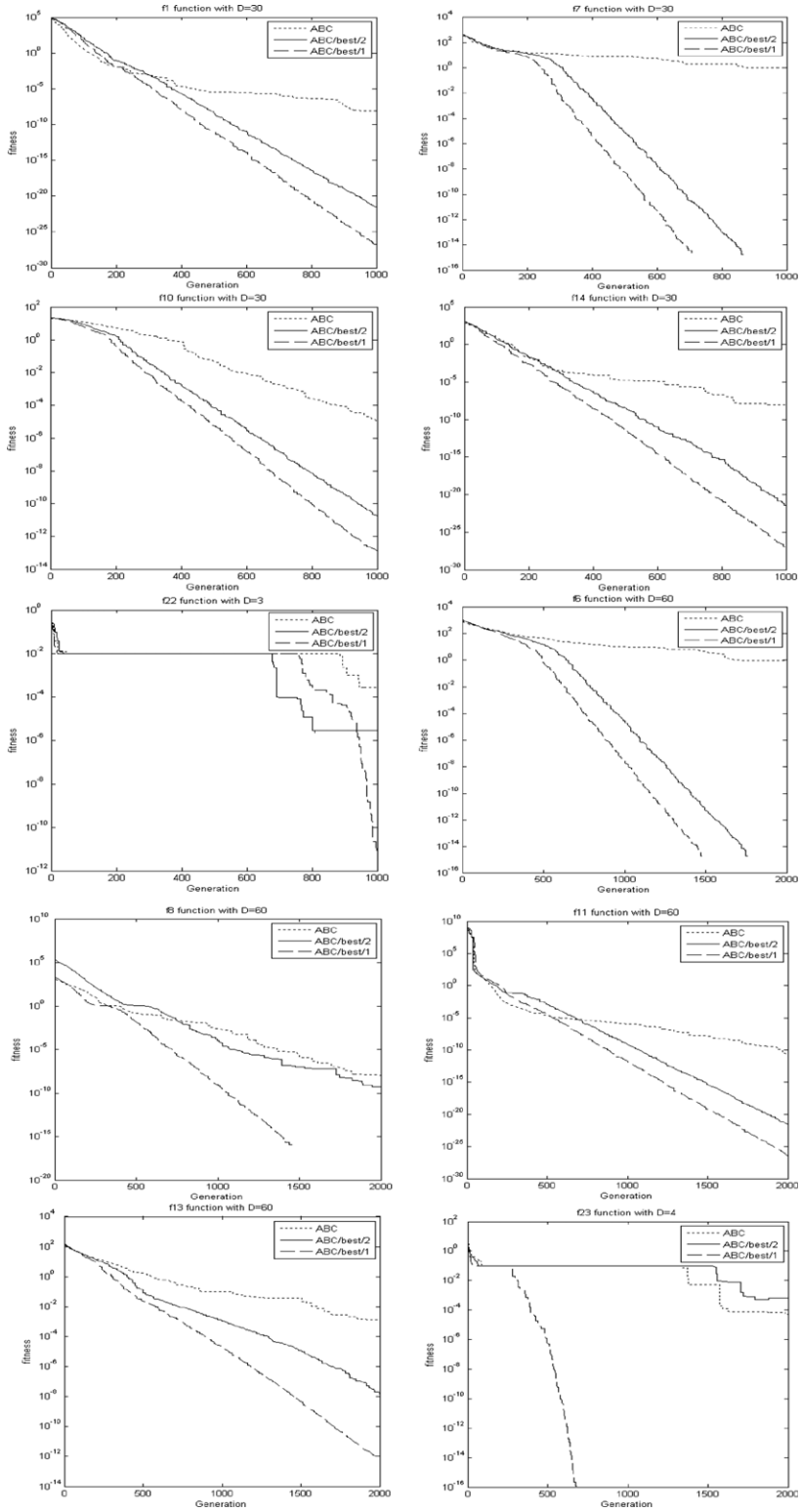

Fig. 5. Convergence performance of the different ABCs on the 10 test functions.

presents the comparison in terms of convergence characteristics of the evolutionary processes in solving the 10 different problems.

Table 4
Performance comparison of GABC, E-ABC, ABC/best/2 and ABC/best/1.

Algorithm		Schaffer		Rosenbrock		Sphere	
		$D = 2$	$D = 3$	$D = 2$	$D = 3$	$D = 30$	$D = 60$
GABC ($C = 1.5$)	Mean	0	1.85e−18	1.68e−04	2.65e−03	4.17e−16	1.43e−15
	SD	0	1.01e−17	4.42e−04	2.22e−03	7.36e−17	1.37e−16
E-ABC	Mean	0	2.79e−07	4.63e−04	1.20e−02	1.67e−16	1.41e−15
	SD	0	2.24e−07	4.57e−04	7.06e−03	2.70e−16	1.82e−15
ABC/best/2	Mean	0	3.56e−06	4.42e−04	9.90e−04	1.7e−126	3.72e−58
	SD	0	1.27e−06	2.39e−04	6.92e−04	2.7e−126	2.67e−58
ABC/best/1	Mean	0	0	**4.99e−06**	**5.52e−06**	**1.1e−150**	**4.40e−69**
	SD	0	0	**8.22e−06**	**3.03e−06**	**1.4e−150**	**2.56e−69**
Algorithm		Griewank		Rastrigin		Ackley	
		$D = 30$	$D = 60$	$D = 30$	$D = 60$	$D = 30$	$D = 60$
GABC ($C = 1.5$)	Mean	2.96e−17	7.54e−16	1.32e−14	3.52e−13	3.21e−14	1.66e−13
	SD	4.99e−17	4.12e−16	2.44e−14	1.24e−13	3.25e−15	2.21e−14
E-ABC	Mean	4.90e−14	4.19e−14	9.97e−15	7.51e−13	1.22e−10	1.55e−07
	SD	7.31e−03	9.05e−03	3.87e−15	6.15e−13	4.86e−11	2.84e−08
ABC/best/2	Mean	**0**	**0**	**0**	**0**	2.50e−14	7.12e−14
	SD	**0**	**0**	**0**	**0**	3.48e−15	4.14e−15
ABC/best/1	Mean	**0**	**0**	**0**	**0**	**1.72e−14**	**6.62e−14**
	SD	**0**	**0**	**0**	**0**	**2.84e−15**	**1.74e−15**

Table 5
Effect of *limit* on the performance of ABC/best.

Function		$limit = 0.2 * SN * D$		$limit = 0.6 * SN * D$		$limit = SN * D$	
		ABC/best/2	ABC/best/1	ABC/best/2	ABC/best/1	ABC/best/2	ABC/best/1
Sphere	Mean	3.63e−22	2.28e−27	4.37e−22	1.57e−27	5.08e−22	1.30e−27
	SD	2.72e−22	1.21e−27	2.14e−22	1.14e−27	3.00e−22	7.44e−28
Griewank	Mean	2.46e−04	7.28e−11	4.47e−08	4.23e−11	8.18e−08	2.31e−06
	SD	1.33e−03	2.95e−11	1.05e−07	2.16e−11	3.71e−07	5.66e−06
Rastrigin	Mean	0	0	0	0	0	5.85e−02
	SD	0	0	0	0	0	2.34e−01
Ackley	Mean	1.92e−11	1.43e−13	1.89e−11	1.26e−13	1.96e−11	1.28e−13
	SD	5.75e−12	1.86e−14	4.75e−12	3.48e−14	6.04e−12	2.66e−14

An interesting result is that all the ABC algorithms have most reliably found the minimum of f_4. It is a region rather than a point in f_4 that is the optimum. Hence, this problem may relatively be easy to solve with a 100% success rate. Important observations about the convergence rate and reliability of different algorithms can be made from the results presented in Fig. 5 and Table 3. These results suggest that the convergence rate of ABC/best/1 and ABC/best/2 is the best and second best for the most test functions. In particular, ABC/best/1 offers the highest accuracy on almost all the functions except f_3 with $D = 60$. The ABC/best/2 ranks first on functions f_3 with $D = 60, f_6, f_7$ and f_9 with $D = 30$, and ranks second on functions $f_1 f_2, f_3$ with $D = 30, f_5, f_8$ with $D = 60, f_9$ with $D = 60, f_{10}$–f_{19}, f_{21} with $D = 3, f_{22}$ and f_{24}. In the case of functions f_{25} and f_{26}, simulation results show that there is no obviously superior algorithm in the means. While, the ABC/best/1 greatly outperforms ABC and ABC/best/2 with better SD. In a word, the superiority in terms of search ability and efficiency of ABC/best should be attributed to an appropriate balance between exploration and exploitation.

In Table 4, ABC/best is further compared with (GABC) in [14] and E-ABC in [19]. ABC/best and E-ABC follow the parameter settings in the original paper of GABC [14]. It is clear that ABC/best works best and second best in most cases and achieves better performance than GABC and E-ABC.

Summarizing the earlier statements, the ability of ABC/best is that it can prevent bees from falling into the local minimum, reduce evolution process significantly and more efficiently (converges faster), compute with more efficiency, and improve bees' searching abilities for ABC algorithms.

4.3. Effects of limit on the performance of ABC/best

In this section, four different kinds of 30-dimensional test functions are used to investigate the impact of *limit*. They are Sphere, Griewank, Rastrigin and Ackley. ABC/best is run 30 times on each of these functions, and the mean values of the final results are shown in Table 5. From Table 5, we can observe that *limit* can influence the results. When *limit* is $0.6 * SN * D$, we obtain a faster convergence speed and better results on the Griewank function. For the other three test functions, the effect of *limit* on the performance of ABC/best is very little except Rastrigin at $limit = SN * D$. Therefore, in our experiments, the selective probability *limit* is set at $0.6 * SN * D$ for all test functions.

Table 6
Performance comparison of ABC, ABC1, ABC2 and ABC/best/1.

Fun	Accept	ABC		ABC1		ABC2		ABC/best/1	
		FEs	SR	FEs	SR	FEs	SR	FEs	SR
f_1	1e−10		0		0	46 985	30	45 890	30
f_2	1e−10		0		0	73 080	30	71 815	30
f_3	5e−00		0		0	99 860	3	90 693	20
f_4	1e−10	15 520	30	15 300	30	13 957	30	13 163	30
f_5	1e−01	87 497	16	85820	18	35 060	30	32 207	30
f_6	1e−10		0		0	52 183	30	51 277	30
f_7	1e−10		0		0	54 067	30	53 163	30
f_8	1e−10		0		0	57 750	30	60 383	30
f_9	1e−10		0		0	50 830	30	48 803	30
f_{10}	1e−10		0		0	78 030	30	76 972	30
f_{11}	1e−10	96 571	28	94 810	30	40 353	30	39 477	30
f_{12}	1e−10		0		0	44 207	30	43 137	30
f_{13}	1e−10		0		0	76 367	30	75 117	30
f_{14}	1e−10	99 840	1	98 960	2	44 930	30	43 897	30
f_{15}	1e−10		0		0	84 683	30	83 857	30

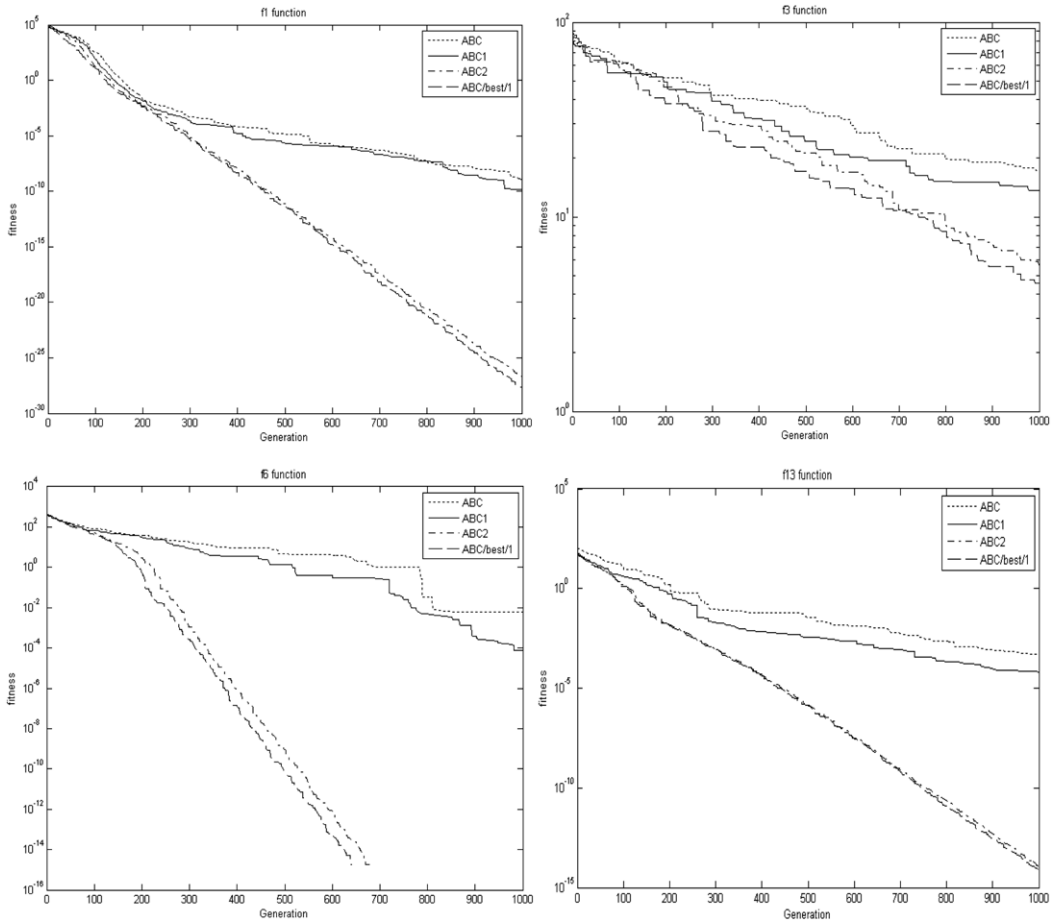

Fig. 6. Convergence performance of the different ABCs on the 4 test functions.

4.4. Effects of each technique on the performance of ABC/best/1

We compare the ABC/best/1 with ABC, ABC1 (ABC only combined with the initialization) and ABC2 (ABC only combined with the search strategies). The computational results are summarized in Table 6. The results given there are the average FEs needed to reach the threshold expressed as acceptable solutions (column 2) specified in Table 6. In addition, successful runs (SR) of the 30 independent runs for each function are also compared. Fig. 6 presents the convergence performance of the different ABCs on the 4 test functions.

It can be observed from Table 6 that, both in the aspects of FEs and SR, ABC1 and ABC2 are superior to ABC, which implies that both the initialization and the search strategies have positive effect on the performance of the algorithm. Specifically, ABC2 greatly outperforms ABC on all the 15 functions. On the other hand, the performance comparisons of ABC2 and ABC/best/1 are not so apparent as those of ABC1 and ABC/best/1, which means that the search strategies play a pivotal role in the proposed algorithm. However, though the contribution of the initialization is far less than the search strategies, the comparison of ABC/best/1 and ABC2 show the initialization is at work. At the same time, it can be seen from Fig. 6 that both the initialization and search equations can increase the performance of the algorithm. Furthermore, we can see that how much they make contribution to improving the performance of the algorithm respectively.

5. Conclusion

In this paper, we have developed a novel algorithm to solve global optimization problems, called ABC/best in which the initial population and scout bees are generated by combining chaotic systems with opposition-based learning method, and the solution search equation is based on that each bee searches only around the best solution of the previous iteration to improve the exploitation. The experimental results tested on 26 benchmark functions show that ABC/best/1 algorithm outperforms ABC and GABC algorithms.

Since the performance of ABC/best/2 algorithm is not as good as that of ABC/best/1 algorithm, ameliorating ABC/best/2 algorithm is our future work. Practical applications of this hybrid approach in areas of clustering, data mining, design and optimization of communication networks, would also be worth studying.

Acknowledgments

The authors are grateful to the editor and two anonymous reviewers for their valuable comments and suggestions on this paper. This work is supported by National Nature Science Foundation of China (Grant Nos. 60974082, 11126287), Fundamental Research Funds for the Central Universities (Grant Nos. JY10000970006) and Foundation of State Key Laboratory of Integrated Service Networks of China.

References

[1] K.S. Tang, K.F. Man, S. Kwong, Q. He, Genetic algorithms and their applications, IEEE Signal Processing Magazine 13 (1996) 22–37.
[2] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proc. IEEE Int. Conf. Neural Networks, vol. 4, 1995, pp. 1942–1948.
[3] M. Dorigo, T. Stutzle, Ant Colony Optimization, MIT Press, Cambridge, MA, 2004.
[4] D. Simon, Biogeography-based optimization, IEEE Transactions on Evolutionary Computation 12 (2008) 702–713.
[5] Z. Geem, J. Kim, G. Loganathan, A new heuristic optimization algorithm: harmony search, Simulation 76 (2001) 60–68.
[6] D. Karaboga, An idea based on honey bee swarm for numerical optimization, Erciyes University, Technical Report-TR06, Kayseri, Turkey, 2005.
[7] D. Karaboga, B. Basturk, A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm, Journal of Global Optimization 39 (2007) 459–471.
[8] D. Karaboga, B. Basturk, On the performance of artificial bee colony (ABC) algorithm, Applied Soft Computing 8 (2008) 687–697.
[9] D. Karaboga, B. Basturk, A comparative study of artificial bee colony algorithm, Applied Mathematics and Computation 214 (2009) 108–132.
[10] A. Singh, An artificial bee colony algorithm for the leaf-constrained minimum spanning tree problem, Applied Soft Computing 9 (2009) 625–631.
[11] F. Kang, et al., Structural inverse analysis by hybrid simplex artificial bee colony algorithms, Computers & Sturctures 87 (2009) 861–870.
[12] L. Samrat, et al., Artificial bee colony algorithm for small signal model parameter extraction of MESFET, Engineering Applications of Artificial Intelligence 11 (2010) 1573–2916.
[13] R. Storn, K. Price, Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces, Journal of Global Optimization 23 (2010) 689–694.
[14] G.P. Zhu, S. Kwong, Gbest-guided artificial bee colony algorithm for numerical function optimization, Applied Mathematics and Computation 217 (2010) 3166–3173.
[15] B. Akay, D. Karaboga, A modified artificial bee colony algorithm for real-parameter optimization, Information Sciences (2010) doi:10.1016/j.ins.2010.07.015.
[16] B. Alatas, Chaotic bee colony algorithms for global numerical optimization, Expert Systems with Applications 37 (2010) 5682–5687.
[17] S. Rahnamayan, et al., Opposition-based differential evolution, IEEE Transactions on Evolutionary Computation 12 (2008) 64–79.
[18] W.F. Gao, S.Y. Liu, A modified artificial bee colony algorithm, Computers & Operations Research 39 (2012) 687–697.
[19] E.M. Montes, et al., Elitist artificial bee colony for constrained real-parameter optimization, IEEE Congress on Evolutionary Computation 11 (2010) 1–8.