Computers & Operations Research 39 (2012) 687–697

Contents lists available at ScienceDirect

Computers & Operations Research

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

A modified artificial bee colony algorithm

Wei-feng Gao *, San-yang Liu

Department of Mathematics, Xidian University, Xi'an, Shannxi 710071, PR China

ARTICLE INFO

Available online 25 June 2011

Keywords:
Artificial bee colony algorithm
Initial population
Solution search equation
Differential evolution

ABSTRACT

Artificial bee colony algorithm (ABC) 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 ABC regarding its solution search equation, which is good at exploration but poor at exploitation. Inspired by differential evolution (DE), we propose an improved solution search equation, which is based on that the bee searches only around the best solution of the previous iteration to improve the exploitation. Then, in order to make full use of and balance the exploration of the solution search equation of ABC and the exploitation of the proposed solution search equation, we introduce a selective probability P and get the new search mechanism. In addition, to enhance the global convergence, when producing the initial population, both chaotic systems and opposition-based learning methods are employed. The new search mechanism together with the proposed initialization makes up the modified ABC (MABC for short), which excludes the probabilistic selection scheme and scout bee phase. Experiments are conducted on a set of 28 benchmark functions. The results demonstrate good performance of MABC in solving complex numerical optimization problems when compared with two ABC-based algorithms.

1. Introduction

Learning from life system, people have developed many optimization computation methods to solve complicated problems in recent decades, such as genetic algorithm (GA) inspired by the Darwinian law of survival of the fittest [1], particle swarm optimization (PSO) inspired by the social behavior of bird flocking or fish schooling [2], ant colony optimization (ACO) inspired by the foraging behavior of ant colonies [3], and biogeography-based optimization (BBO) inspired by the migration behavior of island species [4]. We call this kind of algorithms for scientific computation as "artificial-life computation" [5]. Artificial bee colony algorithm (ABC) is such a new computation technique developed by Karaboga [6] based on simulating the foraging behavior of honey bee swarm. Numerical comparisons demonstrated that the performance of ABC 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 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 also faces up to some challenging problems. For example, the convergence speed of ABC 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 can easily get trapped in the local optima when solving complex multimodal problems [9]. The reasons are as follows. It is well known that both exploration and exploitation are necessary for a population-based optimization algorithm. In the optimization algorithms, the exploration refers to the ability to investigate the various unknown regions in the solution space to discover the global optimum. While, the exploitation refers to the ability to apply the knowledge of the previous good solutions to find better solutions. 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, the solution search equation of ABC which is used to generate new candidate solutions based on the information of previous solutions, is good at exploration but poor at exploitation [14], 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 (CABC) in [16] focuses on avoiding the local optima, but brings in a more extra function evaluations in chaotic search as a result.

To achieve the both goals, inspired by DE, we propose an improved solution search equation, which is based on that the bee searches only around the best solution of the previous iteration to improve the exploitation. Then, in order to make full use of and

* Corresponding author.
E-mail address: gaoweifeng2004@126.com (W.-f. Gao).

0305-0548/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2011.06.007

balance the exploration of the solution search equation of ABC and the exploitation of the proposed solution search equation, we introduce a selective probability P and get the new search mechanism. In addition, to enhance the global convergence, when producing the initial population, both chaotic systems and opposition-based learning method are employed. The new search mechanism together with the proposed initialization makes up the modified ABC (MABC for short), which excludes the probabilistic selection scheme and scout bee phase. The rest of this paper is organized as follows. Section 2 summarizes ABC. The improved ABC algorithm is presented in Section 3. Section 4 presents and discusses the experimental results. Finally, the conclusion is drawn in Section 5.

2. Artificial bee colony algorithm

Artificial bee colony algorithm (ABC), proposed by Karaboga in 2005 for real-parameter optimization, is a recently introduced optimization algorithm which simulates the foraging behavior of a bee colony [6]. ABC classifies the foraging artificial bees into three groups, namely, employed bees, onlooker bees and scout bees. Half of the colony consists of employed bees, and the other half includes onlooker bees. Employed bees search the food around the food source in their memory, meanwhile they pass their food information to onlooker bees. Onlooker bees tend to select good food sources from those founded by the employed bees, then further search the foods around the selected food source. Scout bees are translated from a few employed bees, which abandon their food sources and search new ones.

Similar to the other population-based algorithms, ABC is an iterative process. The units of the basic ABC can be explained as follows:

2.1. Initialization of the population

The initial population of solutions is filled with SN number of randomly generated n-dimensional real-valued vectors (i.e., food sources). Let $X_i=\{x_{i,1},x_{i,2},\ldots,x_{i,n}\}$ represent the ith food source in the population, and then each food source is generated as follows:

$$x_{i,j}=x_{min,j}+rand(0,1)(x_{max,j}-x_{min,j}), \tag{2.1}$$

where $i=1,2,\cdots,SN, j=1,2,\cdots,n$. $x_{min,j}$ and $x_{max,j}$ are the lower and upper bounds for the dimension j, respectively. These food sources are randomly assigned to SN number of employed bees and their fitnesses are evaluated.

2.2. Initialization of the bee phase

At this stage, each employed bee X_i generates a new food source V_i in the neighborhood of its present position by using solution search equation as follows:

$$v_{i,j}=x_{i,j}+\phi_{i,j}(x_{i,j}-x_{k,j}), \tag{2.2}$$

where $k\in\{1,2,\cdots,SN\}$ and $j\in\{1,2,\cdots,n\}$ are randomly chosen indexes; k has to be different from i; $\phi_{i,j}$ is a random number in the range $[-1, 1]$.

Once V_i is obtained, it will be evaluated and compared to X_i. If the fitness of V_i is equal to or better than that of X_i, V_i will replace X_i and become a new member of the population; otherwise X_i is retained. In other words, a greedy selection mechanism is employed between the old and candidate solutions.

2.3. Calculating probability values involved in probabilistic selection

After all employed bees complete their searches, they share their information related to the nectar amounts and the positions of their sources with the onlooker bees on the dance area. An onlooker bee evaluates the nectar information taken from all employed bees and chooses a food source site with a probability related to its nectar amount. This probabilistic selection depends on the fitness values of the solutions in the population. A fitness-based selection scheme might be a roulette wheel, ranking based, stochastic universal sampling, tournament selection or another selection scheme. In basic ABC, roulette wheel selection scheme in which each slice is proportional in size to the fitness value is employed as follows:

$$p_i=f_i\Big/\sum_{j=1}^{SN}f_j, \tag{2.3}$$

where f_i is the fitness value of solution i. Obviously, the higher the f_i is, the more probability that the ith food source is selected.

2.4. Onlooker bee phase

An onlooker bee evaluates the nectar information taken from all the employed bees and selects a food source X_i depending on its probability value p_i. Once the onlooker has selected her food source X_i, she produces a modification on X_i by using Eq. (2.2). As in the case of the employed bees, if the modified food source has a better or equal nectar amount than X_i, the modified food source will replace X_i and become a new member in the population.

2.5. Scout bee phase

If a food source X_i cannot be further improved through a predetermined number of trials *limit*, the food source is assumed to be abandoned, and the corresponding employed bee becomes a scout. The scout produces a food source randomly as follows:

$$x_{i,j}=x_{min,j}+rand(0,1)(x_{max,j}-x_{min,j}), \tag{2.4}$$

where $j=1,2,\cdots,n$.

2.6. Main steps of the artificial bee colony algorithm

Based on the above explanation of initializing the algorithm population, employed bee phase, probabilistic selection scheme, onlooker bee phase and scout bee phase, the pseudo-code of the ABC algorithm is given below:

Algorithm 1 (*Artificial bee colony algorithm*).

01: Initialize the population of solutions $x_{i,j}$, $i=1,2\cdots SN$, $j=1,2\cdots n$, $trial_i=0$, $trial_i=0$ is the non-improvement number of the solution X_i, used for abandonment
02: Evaluate the population
03: cycle=1
04: **repeat**
{– – Produce a new food source population for employed bees – –}
06: **for** $i=1$ to SN **do**
07: Produce a new food source V_i for the employed bee of the food source X_i using (2.2) and evaluate its quality
08: Apply a greedy selection process between V_i and X_i and select the better one
09: If solution X_i does not improve $trial_i=trial_i+1$, otherwise $trial_i=0$
10: **end for**
11: Calculate the probability values p_i by (2.3) for the solutions using fitness values
{– – Produce a new food source population for onlooker bees – –}

12: $t=0, i=1$
13: **repeat**
14: **if** $random < p_i$ **then**
15: Produce a new V_i food source by (2.2) for onlooker bee
16: Apply a greedy selection process between V_i and X_i and select the better one
17: If solution X_i does not improve $trial_i = trial_i + 1$, otherwise $trial_i = 0$
18: $t = t+1$
19: **endif**
20: **until** ($t=SN$)
{– – Determine Scout – –}
21: **if** $max(trial_i) > limit$ **then**
22: Replace X_i with a new randomly produced solution by (2.4)
23: **end if**
24: Memorize the best solution achieved so far
25: cycle=cycle+1
26: **until** (cycle=Maximum Cycle Number)

3. Modified 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). Owing to the randomness and sensitivity dependence on the initial conditions of chaotic maps, chaotic maps have been used to initialize the population so that the search space information can be extracted to increase the population diversity in [16]. At the same time, according to [17], replacing the random initialization with the opposition-based population initialization can get better initial solutions and then accelerate convergence speed. So this paper proposes a novel initialization approach which employs opposition-based learning method and chaotic systems to generate initial population. Here, sinusoidal iterator is selected and its equation is defined as follows:

$$ch_{k+1} = \sin(\pi ch_k), ch_k \in (0,1), k = 0,1,2,\ldots,K, \quad (3.1)$$

where k is the iteration counter and K is the preset maximum number of chaotic iterations. The mapped variables in Eq. (3.1) can distribute in search space with ergodicity, randomness and irregularity. Based on these operations, we propose the following algorithm to generate initial population which can be used instead of a pure random initialization.

Algorithm 2 (*A novel initialization approach*).

01: Set the maximum number of chaotic iteration $K \geq 300$, the population size SN, and the individual counter $i=1, j=1$
{– – chaotic systems – –}
03: **for** $i=1$ to SN **do**
04: **for** $j=1$ to n **do**
05: Randomly initialize variables $ch_{0,j} \in (0,1)$, set iteration counter $k=0$
06: **for** $k=1$ to K **do**
07: $ch_{k+1,j} = \sin(\pi ch_{k,j})$
08: **end for**
09: $x_{i,j} = x_{min,j} + ch_{k,j}(x_{max,j} - x_{min,j})$
10: **end for**
11: **end for**
{– – Opposition-based learning method – –}
13: Set the individual counter $i=1, j=1$
14: **for** $i=1$ to SN **do**
15: **for** $j=1$ to n **do**
16: $ox_{i,j} = x_{min,j} + x_{max,j} - x_{i,j}$
17: **end for**
18: **end for**
19: Selecting SN fittest individuals from set the $\{X(SN) \cup OX(SN)\}$ as initial population.

3.2. A modified search equation

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 is a mutation strategy frequently used in the literature:

$$DE/best/1 : V_i = X_{best} + F(X_{r1} - X_{r2}), \quad (3.2)$$

where $i = \{1,2,\cdots,SN\}$ and $r1$ and $r2$ are mutually different random integer indices selected from $\{1,2,\cdots,SN\}$. F, commonly known as scaling factor or amplification factor, is a positive real

Table 1
Effect of the selective probability P on the performance of MABC.

Algorithm		Sphere	Rosebrock	Griewank	Rastrigin	NC-Rastrigin	Ackley
MABC (P=0.0)	Mean	1.86e−36	3.07e−00	3.28e−04	3.32e−02	0	2.88e−14
	SD	1.46e−36	3.45e−00	1.83e−03	1.78e−01	0	3.63e−15
MABC (P=0.1)	Mean	2.57e−35	3.04e−00	2.46e−04	6.22e−02	0	3.00e−14
	SD	2.44e−35	3.95e−00	1.36e−03	2.41e−01	0	1.42e−15
MABC (P=0.3)	Mean	5.48e−34	1.01e−00	4.00e−05	5.02e−03	0	3.12e−14
	SD	3.87e−34	1.41e−00	2.15e−04	7.05e−02	0	1.77e−15
MABC (P=0.5)	Mean	7.57e−33	1.67e−00	0	0	0	3.59e−14
	SD	3.28e−33	1.43e−00	0	0	0	3.39e−15
MABC (P=0.7)	Mean	9.43e−32	6.11e−01	0	0	0	4.13e−14
	SD	6.67e−32	4.55e−01	0	0	0	2.17e−15
MABC (P=0.9)	Mean	2.47e−30	9.55e−01	3.54e−18	0	0	5.27e−14
	SD	2.53e−30	1.03e−00	7.28e−17	0	0	6.19e−15
MABC (P=1.0)	Mean	4.33e−30	1.47e−00	6.66e−15	0	0	6.62e−14
	SD	4.21e−30	1.22e−00	3.44e−14	0	0	1.10e−14

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 example is the DE/best/1, where the best solutions explored in the history are used to direct the movement of the current population. Based on the variant DE algorithm and the property of ABC, the solution search equation is devised as follows:

$$ABC/best/1: v_{i,j} = x_{best,j} + \phi_{i,j}(x_{r1,j} - x_{r2,j}), \tag{3.3}$$

where the indices $r1$ and $r2$ are mutually exclusive integers randomly chosen from $\{1,2,\cdots,SN\}$, and different from the base index i; X_{best} is the best individual vector with the best fitness in the current population and $j \in \{1,2,\ldots,n\}$ is randomly chosen indexes; $\phi_{i,j}$ is a random number in the range $[-1,1]$. In Eq. (2.2), 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. (2.2) is random enough for exploration. In other words, the solution search equation described by Eq. (2.2) is good at exploration but poor at exploitation. However, according to Eq. (3.3), ABC/best/1 can drive the new candidate solution only around the best solution of the previous iteration. Therefore, the proposed solution search equation described by Eq. (3.3) can increase the exploitation of ABC.

3.3. The proposed approach

From the above explanation, it is clear that ABC/best/1 has a good capacity of the exploitation. Unfortunately, ABC/best/1 can reduce the exploration of ABC. If all bees produce new food sources using (3.3), the algorithm can easily get trapped in the local optima when solving complex multimodal problems. In other words, ABC which is good at exploration but poor at

Table 2
Benchmark functions used in experiments.

Function	*Search range*	*Min*
$f_1(X)=\sum_{i=1}^{n} x_i^2$	$[-100,100]^n$	0
$f_2(X)=\sum_{i=1}^{n}(10^6)^{(i-1)/(n-1)}x_i^2$	$[-100,100]^n$	0
$f_3(X)=\sum_{i=1}^{n} ix_i^2$	$[-10,10]^n$	0
$f_4(X)=\sum_{i=1}^{n} \lvert x_i\rvert^{(i+1)}$	$[-10,10]^n$	0
$f_5(X)=\sum_{i=1}^{n} \lvert x_i\rvert+\prod_{i=1}^{n} \lvert x_i\rvert$	$[-10,10]^n$	0
$f_6(X)=max_i\{\lvert x_i\rvert, 1\le i\le n\}$	$[-100,100]^n$	0
$f_7(X)=\sum_{i=1}^{n}(\lfloor x_i+0.5\rfloor)^2$	$[-100,100]^n$	0
$f_8(X)=\sum_{i=1}^{n} ix_i^4$	$[-1.28,1.28]^n$	0
$f_9(X)=\sum_{i=1}^{n} ix_i^4+random[0,1)$	$[-1.28,1.28]^n$	0
$f_{10}(X)=\sum_{i=1}^{n-1}[100(x_{i+1}-x_i^2)^2+(x_i-1)^2]$	$[-10,10]^n$	0
$f_{11}(X)=[x_i^2-10\cos(2\pi x_i)+10]$	$[-5.12,5.12]^n$	0
$f_{12}(X)=[y_i^2-10\cos(2\pi y_i)+10]$	$[-5.12,5.12]^n$	0
$y_i=\begin{cases} x_i & \lvert x_i\rvert<\frac{1}{2} \\ \frac{round(2x_i)}{2} & \lvert x_i\rvert\ge\frac{1}{2}\end{cases}$		
$f_{13}(X)=\frac{1}{4000}\sum_{i=1}^{n} x_i^2-\prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$	$[-600,600]^n$	0
$f_{14}(X)=418.98288727243369*n-\sum_{i=1}^{n} x_i\sin(\sqrt{\lvert x_i\rvert})$	$[-500,500]^n$	0
$f_{15}(X)=-20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right)-\exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right)+20+e$	$[-32,32]^n$	0
$f_{16}(X)=\frac{\pi}{n}\{10\sin^2(\pi y_1)+\sum_{i=1}^{n-1}(y_i-1)^2[1+10\sin^2(\pi y_{i+1})]+(y_n-1)^2\}+\sum_{i=1}^{n} u(x_i,10,100,4)$ $y_i=1+\frac{1}{4}(x_i+1)$ $u_{x_i,a,k,m}=\begin{cases} k(x_i-a)^m & x_i>a \\ 0 & -a\le x_i\le a \\ k(-x_i-a)^m & x_i<-a\end{cases}$	$[-50,50]^n$	0
$f_{17}(X)=\frac{1}{10}\{\sin^2(\pi x_1)+\sum_{i=1}^{n-1}(x_i-1)^2[1+\sin^2(3\pi x_{i+1})]+(x_n-1)^2[1+\sin^2(2\pi x_{i+1})]\}+\sum_{i=1}^{n} u(x_i,5,100,4)$	$[-50,50]^n$	0
$f_{18}(X)=\sum_{i=1}^{n} \lvert x_i\cdot\sin(x_i)+0.1\cdot x_i\rvert$	$[-10,10]^n$	0
$f_{19}(X)=\sum_{i=1}^{n-1}(x_i-1)^2[1+\sin^2(3\pi x_{i+1})]+\sin^2(3\pi x_1)+\lvert x_n-1\rvert[1+\sin^2(3\pi x_n)]$	$[-10,10]^n$	0
$f_{20}(X)=\sum_{i=1}^{D}(\sum_{k=0}^{k_{max}}[a^k\cos(2\pi b^k(x_i+0.5))])-D\sum_{k=0}^{k_{max}}[a^k\cos(2\pi b^k 0.5)], a=0.5, b=3, k_{max}=20$	$[-0.5,0.5]^n$	0
$f_{21}(X)=0.5+\sin^2(\frac{\sqrt{\sum_{i=1}^{n} x_i^2}-0.5}{(1+0.001(\sum_{i=1}^{n} x_i^2))^2}$	$[-100,100]^n$	0
$f_{22}(X)=\frac{1}{n}\sum_{i=1}^{n}(x_i^4-16x_i^2+5x_i)$	$[-5,5]^n$	-78.33236
$f_{23}(X)=-\sum_{i=1}^{n}\sin(x_i)\sin^{20}\left(\frac{i\times x_i^2}{\pi}\right)$	$[0,\pi]^n$	-99.2784 for $n=100$
$f_{24}(X)=\sum_{i=1}^{n} z_i^2\ \ Z=X-O$	$[-100,100]^n$	0
$f_{25}(X)=[z_i^2-10\cos(2\pi z_i)+10]\ \ Z=X-O$	$[-5.12,5.12]^n$	0
$f_{26}(X)=\frac{1}{4000}\sum_{i=1}^{n} z_i^2-\prod_{i=1}^{n}\cos(\frac{z_i}{\sqrt{i}})+1\ \ Z=X-O$	$[-600,600]^n$	0
$f_{27}(X)=-20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} z_i^2}\right)-\exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi z_i)\right)$ $Z=X-O$	$[-32,32]^n$	0
$f_{28}(X)=\sum_{i=1}^{n} \lvert z_i\cdot\sin(z_i)+0.1\cdot z_i\rvert\ \ Z=X-O$	$[-10,10]^n$	0

exploitation results in a slow convergence. While ABC/best/1, which is good at exploitation but poor at exploration, cannot avoid premature convergence. To address this contradiction, we propose the new search mechanism which introduces the selective probability P to balance the exploration of the solution search equation (2.2) and the exploitation of the modified solution search equation (3.3). The new search mechanism with Algorithm 2 makes up MABC. Based on the above explanation, the pseudo-code of MABC is given below:

Algorithm 3 (*Modified artificial bee colony algorithm*).

01: Set the population size SN, give the maximum number of function evaluations, $Max.FE$
02: Perform **Algorithm 2** to create an initial population $\{X_i|i=1,2,\cdots,SN\}$, calculate the function values of the population $\{f_i|i=1,2,\cdots,SN\}$
03: **While** (stopping criterion is not met, namely $FE<Max.FE$) **do**
04: **for** $i=1$ to SN **do**
{– – Produce a new food source using the new search mechanism– –}
06: Choose X_{r1},X_{r2} randomly from the current population, The indices r_1,r_2 are mutually exclusive integers randomly chosen from the range [1,SN], which are also different from the index i
07: Randomly choose j from $\{1,2,\cdots,n\}$ and produce $\phi_{i,j}\in[-1,1]$
08: Generate a new food source V_i according to $v_{i,j}=x_{best,j}+\phi_{i,j}(x_{r1,j}-x_{r2,j})$
09: **if** $f(V_i)<f(X_i)$ **then**
10: $X_i=V_i$
11: **else then**
12: **if** $rand(0,1)<P$ **then**
14: Randomly choose j from $\{1,2,\cdots,n\}$, $k\in\{1,2,\cdots,SN\}$ which has to be different from i and produce $\phi_{i,j}\in[-1,1]$
15: Generate a new food source V_i according to $v_{i,j}=x_{i,j}+\phi_{i,j}(x_{i,j}-x_{k,j})$
16: **if** $f(V_i)<f(X_i)$ **then**
17: $X_i=V_i$
18: **endif**
19: **end if**
20: **end if**
21: **end for**
22: **end while** ($FE=Max.FE$)

Table 3
Best, worst, median, mean and standard deviation values obtained by ABC and MABC through 30 independent runs on function from f_1 to f_7.

Fun	Dim		Best	Worst	Median	Mean	SD	Significant
f_1	30	ABC	2.02e−10	1.07e−09	2.02e−10	5.21e−10	2.46e−10	
		MABC	1.69e−32	2.48e−31	1.80e−32	9.43e−32	6.67e−32	+
	60	ABC	3.49e−10	3.27e−09	3.27e−09	1.09e−09	9.37e−10	
		MABC	9.29e−30	1.55e−28	4.68e−29	6.03e−29	4.31e−29	+
	100	ABC	3.87e−10	3.11e−09	3.87e−10	1.64e−09	9.85e−10	
		MABC	4.65e−28	2.63e−27	1.86e−27	1.43e−27	8.12e−28	+
f_2	30	ABC	2.65e−07	1.32e−05	7.71e−07	4.10e−06	3.85e−06	
		MABC	2.55e−29	1.99e−27	1.99e−27	3.66e−28	5.96e−28	+
	60	ABC	2.14e−07	6.25e−06	4.53e−06	2.31e−06	2.18e−06	
		MABC	3.65e−26	8.69e−25	3.65e−26	3.51e−25	2.72e−25	+
	100	ABC	4.00e−07	6.46e−06	1.37e−06	1.79e−06	1.63e−06	
		MABC	3.17e−24	3.73e−24	3.17e−24	3.52e−24	2.47e−25	+
f_3	30	ABC	9.03e−12	4.62e−11	1.56e−11	2.22e−11	1.14e−11	
		MABC	3.57e−33	5.29e−32	4.62e−33	2.10e−32	1.56e−32	+
	60	ABC	7.95e−11	3.17e−10	3.17e−10	1.89e−10	9.14e−11	
		MABC	5.61e−30	3.29e−29	6.24e−30	1.39e−29	8.84e−30	+
	100	ABC	2.82e−10	2.85e−09	4.21e−10	1.25e−09	9.75e−10	
		MABC	1.50e−28	8.74e−28	1.72e−28	4.46e−28	2.08e−28	+
f_4	30	ABC	1.34e−18	4.29e−16	1.82e−16	1.45e−16	1.55e−16	
		MABC	1.26e−74	1.34e−68	7.02e−71	2.70e−69	5.38e−69	+
	60	ABC	4.88e−11	7.62e−10	4.88e−11	2.14e−10	2.75e−10	
		MABC	5.16e−65	9.54e−62	3.59e−64	3.00e−62	3.87e−62	+
	100	ABC	2.52e−08	2.05e−06	6.75e−08	4.83e−07	7.88e−07	
		MABC	7.76e−52	8.74e−48	8.01e−52	1.92e−48	3.42e−48	+
f_5	30	ABC	1.28e−06	2.63e−06	1.28e−06	1.83e−06	4.80e−07	
		MABC	8.41e−18	4.09e−17	8.41e−18	2.40e−17	9.02e−18	+
	60	ABC	5.77e−06	9.29e−06	6.81e−06	7.23e−06	1.28e−06	
		MABC	5.30e−16	8.72e−16	8.33e−16	6.96e−16	1.20e−16	+
	100	ABC	1.28e−06	1.59e−05	1.29e−05	1.30e−05	1.93e−06	
		MABC	2.08e−15	6.63e−15	6.46e−15	4.41e−15	1.50e−15	+
f_6	30	ABC	1.30e+01	2.07e+01	1.52e+01	1.80e+01	2.25e−00	
		MABC	9.16e−00	1.42e+01	1.26e+01	1.02e+01	1.49e−00	+
	60	ABC	3.79e+01	4.65e+01	4.18e+01	4.22e+01	2.73e−00	
		MABC	2.92e+01	4.03e+01	3.80e+01	3.77e+01	3.14e−00	+
	100	ABC	5.30e+01	6.14e+01	5.30e+01	5.76e+01	2.74e−00	
		MABC	5.70e+01	6.23e+01	5.76e+01	5.98e+01	1.60e−00	.
f_7	30	ABC	0	0	0	0	0	
		MABC	0	0	0	0	0	*NA*
	60	ABC	0	0	0	0	0	
		MABC	0	0	0	0	0	*NA*
	100	ABC	0	0	0	0	0	
		MABC	0	0	0	0	0	*NA*

3.4. Adjusting the selective probability P

Note that the parameter P plays an important role in balancing the exploration and exploitation of the candidate solution search. When P takes 0, only Eq. (3.3) is at work. When P increases from 0 to 1, the exploration of Eq. (2.2) will also increase correspondingly. However, P should not be too large because the large value of P might weaken the exploitation of the algorithm. Therefore the selective probability parameter P needs to be tuned. In this section, six different kinds of thirty-dimensional (30-D) test functions are used to investigate the impact of this parameter. They are the Sphere, Rosenbrock, Griewank, Rastrigin, NC-Rastrigin and Ackley functions [15] as defined in Section 4. MABC runs 30 times on each of these functions, and the mean and standard deviation values of the final results are presented in Table 1. As all test functions are minimization problems, the smaller the final result, the better it is. From Table 1, we can observe that P can influence the results. When P is 0, we obtain a faster convergence velocity and better results on the Sphere and Ackley functions. For the other four test functions, better results are obtained when P is around 0.7. At the same time, P has a smaller effect on the Sphere and Ackley functions than for the other four test functions. Hence, in our experiments, the selective probability P is set at 0.7 for all test functions. Above all, the proposed approach is able to reach the balance between exploration and exploitation.

4. Experimental studies on function optimization problems

4.1. Benchmark functions and parameter settings

In this section, MABC is applied to minimize a set of 26 scalable benchmark functions of dimensions $D=30$, 60 or 100 [9,14,17] and a set of two functions of higher dimension $D=100$, 200 or 300 [17], as shown in Table 2.

Summarized in Table 2 are the 28 scalable benchmark functions. f_1–f_6 and f_8 are continuous unimodal functions. f_7 is a discontinuous step function, and f_9 is a noisy quartic function. f_{10} is the Rosenbrock function which is unimodal for $D=2$ and 3 but may have multiple minima in high dimension cases [18]. f_{11}–f_{23} are multimodal and the number of their local minima increases exponentially with the problem dimension. f_{24}–f_{28} are shifted functions and O is a randomly generated shift vector located in search range. In addition, f_{14} is the only bound-constrained function investigated in this paper.

Table 4
Best, worst, median, mean and standard deviation values obtained by ABC and MABC through 30 independent runs on function from f_8 to f_{14}.

Fun	Dim		Best	Worst	Median	Mean	SD	Significant
f_8	30	ABC	1.26e−29	1.86e−28	1.64e−29	5.51e−29	6.70e−29	
		MABC	8.80e−69	5.97e−67	1.01e−68	1.45e−67	2.28e−67	+
	60	ABC	4.90e−28	2.00e−26	4.90e−28	6.53e−27	7.23e−27	
		MABC	4.49e−64	2.37e−61	2.37e−61	5.00e−62	9.38e−62	+
	100	ABC	1.31e−26	1.41e−25	1.34e−26	5.65e−26	4.90e−26	
		MABC	3.99e−61	1.56e−59	3.99e−61	5.72e−60	5.32e−60	+
f_9	30	ABC	6.03e−02	1.27e−01	7.67e−02	8.74e−02	1.77e−02	
		MABC	1.84e−02	4.58e−02	4.48e−02	3.71e−02	8.53e−03	+
	60	ABC	1.87e−01	2.65e−01	2.43e−01	2.39e−01	2.86e−02	
		MABC	9.20e−02	1.33e−01	1.21e−01	1.14e−01	1.16e−02	+
	100	ABC	3.96e−01	4.92e−01	4.70e−01	4.55e−01	3.20e−02	
		MABC	1.64e−01	2.56e−01	2.56e−01	2.31e−01	2.79e−02	+
f_{10}	30	ABC	2.12e−02	2.20e−00	5.56e−01	4.23e−01	4.34e−01	
		MABC	4.09e−02	1.95e−00	2.34e−01	6.11e−01	4.55e−01	.
	60	ABC	1.88e−01	5.75e−00	1.80e−00	1.86e−00	1.36e−00	
		MABC	2.17e−01	5.26e−00	1.30e−00	1.51e−00	1.34e−00	+
	100	ABC	5.48e−01	3.94e−00	6.33e−01	1.59e−00	1.23e−00	
		MABC	5.13e−01	7.77e−00	6.71e−01	1.98e−00	1.30e−00	.
f_{11}	30	ABC	3.58e−10	1.46e−01	5.50e−10	4.81e−03	2.57e−02	
		MABC	0	0	0	0	0	+
	60	ABC	3.21e−10	1.99e−00	1.28e−05	3.71e−01	5.97e−01	
		MABC	0	0	0	0	0	+
	100	ABC	5.40e−09	1.99e−00	1.99e−00	1.10e−00	8.21e−01	
		MABC	0	0	0	0	0	+
f_{12}	30	ABC	8.96e−10	1.00e−00	1.77e−08	1.12e−01	2.97e−01	
		MABC	0	0	0	0	0	+
	60	ABC	2.17e−07	3.025e−00	1.09e−00	1.47e−00	9.47e−01	
		MABC	0	0	0	0	0	+
	100	ABC	2.00e−00	7.21e−00	5.02e−00	4.74e−00	2.01e−00	
		MABC	0	0	0	0	0	+
f_{13}	30	ABC	4.74e−12	1.35e−07	1.77e−08	1.61e−08	3.99e−08	
		MABC	0	0	0	0	0	+
	60	ABC	7.99e−12	1.05e−09	1.45e−11	1.39e−10	3.10e−10	
		MABC	0	0	0	0	0	+
	100	ABC	6.29e−13	3.05e−08	1.01e−10	2.01e−09	1.32e−09	
		MABC	0	0	0	0	0	+
f_{14}	30	ABC	1.54e−06	2.37e+02	3.76e−01	8.86e+01	8.62e+01	
		MABC	−1.81e−12	0	0	−1.21e−13	4.53e−13	+
	60	ABC	3.55e+02	7.69e+02	7.69e+02	5.40e+02	1.41e+02	
		MABC	2.91e−11	3.63e−11	2.91e−11	3.56e−11	2.18e−12	+
	100	ABC	7.81e+02	1.55e+03	1.51e+03	1.29e+03	2.23e+02	
		MABC	1.09e−10	1.23e−10	1.16e−10	1.19e−10	4.06e−12	+

The set of experiments tested on 28 numerical benchmark function are performed to compare the performance of MABC with that of ABC. In all simulations, as the number of optimization parameters increases, we set the maximum number of function evaluations to be 150,000, 300,000 and 500,000 for each function (the population size is 150, namely, $SN=75$), respectively. All results reported in this section are obtained based on 30 independent runs.

4.2. Experimental results

The performance on the solution accuracy of ABC is compared with that of MABC. The results are shown in Tables 3–6 in terms of the best, worst, median, mean and standard deviation of the solutions obtained in the 30 independent runs by each algorithm. Fig. 1 graphically presents the comparison in terms of convergence characteristics of the evolutionary processes in solving the eight different problems.

An interesting result is that the two ABC-based algorithms have most reliably found the minimum of f_7. It is a region rather than a point in f_7 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. 1 and Tables 3–6. These results suggest that the convergence rate of MABC is better than ABC on the most test functions. In particular, MABC can find optimal solutions on functions $f_{11}-f_{13}$, f_{25}, f_{26} and f_{20}, f_{24} with $D=30$. MABC offers the higher accuracy on almost all the functions except functions f_6 with $D=100$ and f_{10} with $D=30$, 100. In the case of functions f_6 with $D=100$ and f_{10} with $D=30$, 100, simulation results show that the convergence rate of MABC is worse than ABC. While, as the results obtained by MABC are of the same order of magnitude as the results by ABC on these two functions, the superiority of ABC to MABC is not very obvious in terms of the best, worst, median, mean and standard deviation of the solutions. In a word, the superiority in terms of search ability and efficiency of MABC should be attributed to an appropriate balance between exploration and exploitation.

In the 9th columns of Tables 3–6, we report the statistical significance level of the difference of the means of the two algorithms. Note that here '+' indicates the t value is significant at a 0.05 level of significance by two-tailed test, '.' stands for the difference of means is not statistically significant and '*NA*' means not applicable, covering cases for which the two

Table 5
Best, worst, median, mean and standard deviation values obtained by ABC and MABC through 30 independent runs on function from f_{15} to f_{21}.

Fun	Dim		Best	Worst	Median	Mean	SD	Significant
f_{15}	30	ABC	2.26e−06	8.32e−06	7.17e−06	4.83e−06	2.12e−06	
		MABC	3.64e−14	4.35e−14	3.99e−14	4.13e−14	2.17e−15	+
	60	ABC	2.44e−06	1.57e−05	2.44e−06	7.79e−06	3.63e−06	
		MABC	1.14e−13	1.57e−13	1.32e−13	1.37e−13	1.24e−14	+
	100	ABC	5.12e−06	1.38e−05	5.94e−06	1.02e−05	2.92e−06	
		MABC	3.27e−13	3.98e−13	3.27e−13	3.56e−13	2.29e−14	+
f_{16}	30	ABC	7.83e−12	1.93e−11	1.31e−11	1.39e−11	3.82e−12	
		MABC	1.57e−32	2.73e−32	1.57e−32	1.90e−32	3.70e−33	+
	60	ABC	2.64e−11	9.54e−11	2.64e−11	4.98e−11	2.69e−11	
		MABC	1.49e−31	1.17e−30	9.50e−31	6.19e−31	3.62e−31	+
	100	ABC	3.71e−11	1.88e−10	1.24e−10	9.50e−11	5.34e−11	
		MABC	1.05e−30	3.12e−30	1.05e−30	1.89e−30	8.42e−31	+
f_{17}	30	ABC	3.85e−10	1.53e−09	7.63e−10	1.06e−09	4.24e−10	
		MABC	5.91e−32	4.47e−31	1.12e−31	2.23e−31	1.46e−31	+
	60	ABC	9.13e−10	7.83e−09	6.41e−09	4.42e−09	2.44e−09	
		MABC	1.84e−29	7.44e−29	4.50e−29	3.80e−29	1.87e−29	+
	100	ABC	2.43e−11	2.35e−10	6.85e−11	1.11e−10	7.39e−11	
		MABC	1.07e−28	2.95e−28	1.40e−28	1.81e−28	6.44e−29	+
f_{18}	30	ABC	2.99e−05	1.05e−04	1.05e−04	7.66e−05	2.76e−05	
		MABC	3.74e−18	6.54e−16	1.48e−17	1.58e−16	2.48e−16	+
	60	ABC	2.18e−04	1.24e−03	3.25e−04	5.78e−04	3.51e−04	
		MABC	2.55e−16	1.90e−15	4.45e−16	8.20e−16	4.69e−16	+
	100	ABC	7.13e−04	1.27e−02	7.13e−04	7.62e−03	5.10e−03	
		MABC	2.38e−15	9.68e−15	7.76e−15	5.83e−15	1.97e−15	+
f_{19}	30	ABC	1.11e−10	9.73e−10	1.11e−10	7.34e−10	3.26e−10	
		MABC	1.34e−31	2.08e−31	1.34e−31	1.48e−31	2.30e−32	+
	60	ABC	9.00e−11	4.52e−09	7.00e−10	1.63e−09	1.60e−09	
		MABC	9.48e−31	8.80e−30	3.35e−30	4.08e−30	2.58e−30	+
	100	ABC	1.68e−10	3.50e−09	3.50e−09	2.42e−09	1.24e−09	
		MABC	4.02e−29	1.56e−28	1.31e−28	8.49e−29	3.57e−29	+
f_{20}	30	ABC	1.38e−01	1.62e−01	1.39e−01	1.46e−01	1.09e−02	
		MABC	0	0	0	0	0	+
	60	ABC	1.87e−01	3.51e−01	2.93e−01	2.77e−01	6.77e−02	
		MABC	0	1.42e−14	7.10e−15	9.94e−15	5.68e−15	+
	100	ABC	6.66e−00	7.48e−00	7.26e−00	7.07e−00	4.08e−00	
		4.26e−14	5.68e−14	4.26e−14	5.21e−14	6.69e−15	+	
f_{21}	30	ABC	4.147e−01	4.598e−01	4.524e−01	4.413e−01	1.81e−02	
		MABC	2.277e−01	3.455e−01	3.121e−01	2.952e−01	3.17e−02	+
	60	ABC	4.960e−01	4.976e−01	4.974e−01	4.971e−01	5.90e−04	
		MABC	4.796e−01	4.903e−01	4.850e−01	4.840e−01	3.62e−03	+
	100	ABC	4.996e−01	4.998e−01	4.998e−01	4.997e−01	4.51e−05	
		MABC	4.988e−01	4.992e−01	4.991e−01	4.990e−01	1.75e−04	+

Table 6
Best, worst, median, mean and standard deviation values obtained by ABC and MABC through 30 independent runs on function f_{22} to f_{28}.

Fun	Dim		Best	Worst	Median	Mean	SD	Significant
f_{22}	100	ABC	−77.8713	−77.3299	−77.8523	−77.5964	2.23e−01	
		MABC	−78.3323	−78.3323	−78.3323	−78.3323	2.06e−07	+
	200	ABC	−77.3471	−77.0723	−77.3471	−77.2602	9.93e−02	
		MABC	−78.3323	−78.3323	−78.3323	−78.3323	2.40e−07	+
	300	ABC	−77.6555	−77.2090	−77.4704	−77.4076	1.37e−01	
		MABC	−78.3323	−78.3323	−78.3323	−78.3323	1.84e−08	+
f_{23}	100	ABC	−84.5683	−82.8617	−82.8617	−83.6626	6.42e−01	
		MABC	−91.8149	−89.7489	−89.7489	−90.7238	5.03e−01	+
	200	ABC	−164.7125	−162.6145	−164.0683	−164.0177	7.32e−01	
		MABC	−176.2015	−172.9625	−176.2015	−174.3186	9.91e−01	+
	300	ABC	−251.3510	−246.8869	−249.5962	−249.4527	1.23e−00	
		MABC	−264.3302	−262.2419	−263.2270	−263.3121	7.14e−01	+
f_{24}	30	ABC	8.66e−10	2.53e−09	1.49e−09	1.55e−09	5.54e−10	
		MABC	0	0	0	0	0	+
	60	ABC	2.18e−09	1.13e−08	2.61e−09	4.25e−09	3.54e−09	
		MABC	7.88e−31	1.27e−28	7.29e−30	5.61e−29	4.18e−29	+
	100	ABC	2.63e−09	1.11e−08	4.05e−09	6.63e−09	3.39e−09	
		MABC	5.15e−28	3.20e−27	1.21e−27	1.44e−27	9.16e−28	+
f_{25}	30	ABC	1.92e−09	9.95e−01	1.35e−07	1.49e−01	3.55e−01	
		MABC	0	0	0	0	0	+
	60	ABC	1.17e−08	3.01e−00	9.95e−01	1.08e−00	8.64e−01	
		MABC	0	0	0	0	0	+
	100	ABC	1.53e−00	6.14e−00	6.06e−00	4.55e−00	1.92e−00	
		MABC	0	0	0	0	0	+
f_{26}	30	ABC	4.45e−10	9.92e−03	5.88e−09	4.93e−04	2.25e−03	
		MABC	0	0	0	0	0	+
	60	ABC	4.13e−10	1.36e−07	1.32e−09	1.28e−08	2.91e−08	
		MABC	0	0	0	0	0	+
	100	ABC	5.13e−10	5.57e−08	3.84e−09	8.49e−09	1.33e−08	
		MABC	0	0	0	0	0	+
f_{27}	30	ABC	1.81e−05	1.82e−04	1.82e−04	9.73e−05	5.69e−05	
		MABC	4.35e−14	5.77e−14	4.35e−14	4.92e−14	5.31e−15	+
	60	ABC	6.21e−05	1.83e−04	9.75e−05	1.11e−04	3.97e−05	
		MABC	1.60e−13	2.53e−13	2.10e−13	2.00e−13	3.07e−14	+
	100	ABC	1.31e−04	5.87e−04	5.11e−04	3.24e−04	1.90e−04	
		MABC	4.20e−13	6.51e−13	5.26e−13	5.33e−13	7.58e−14	+
f_{28}	30	ABC	7.63e−05	1.72e−03	7.63e−05	5.89e−04	6.22e−04	
		MABC	6.24e−17	2.91e−16	2.91e−16	1.38e−16	8.11e−17	+
	60	ABC	2.10e−03	1.12e−02	9.56e−03	6.81e−03	3.53e−03	
		MABC	3.46e−16	1.78e−15	4.87e−16	9.71e−16	5.70e−16	+
	100	ABC	8.60e−04	5.04e−02	5.04e−02	2.64e−02	2.00e−02	
		MABC	1.28e−15	5.39e−15	1.28e−15	3.01e−15	1.39e−15	+

Table 7
Comparison between MABC and ABC on optimizing six benchmark functions.

Function	Dim	GABC			MABC		
		Max.FE	Mean	SD	Max.FE	Mean	SD
Schaffer	30	4.0e+05	2.81e−01	9.12e−02	4.0e+05	2.56e−01	4.65e−02
	60	4.0e+05	4.77e−01	8.04e−03	4.0e+05	4.68e−01	7.40e−03
Rosenbrock	30	4.0e+05	7.93e−01	1.36e−00	4.0e+05	1.73e−01	1.61e−01
	60	4.0e+05	1.90e−00	1.97e−00	4.0e+05	3.32e−01	2.11e−01
Sphere	30	4.0e+05	4.17e−16	7.36e−17	1.5e+05	9.43e−32	6.67e−32
	60	4.0e+05	1.43e−15	1.37e−16	3.0e+05	6.03e−29	4.31e−29
Griewank	30	4.0e+05	2.96e−17	4.99e−17	1.5e+05	0	0
	60	4.0e+05	7.54e−16	4.12e−16	3.0e+05	0	0
Rastrigin	30	4.0e+05	1.32e−14	2.44e−14	1.5e+05	0	0
	60	4.0e+05	3.52e−13	1.24e−13	3.0e+05	0	0
Ackley	30	4.0e+05	3.21e−14	3.25e−15	4.0e+05	2.98e−14	2.26e−15
	60	4.0e+05	1.00e−13	6.08e−15	4.0e+05	6.73e−14	5.91e−15

algorithms achieve the same accuracy results. It also clearly indicates that the proposed MABC is superior to ABC on almost all the functions.

In Table 7, MABC is further compared with Gbest-guided artificial bee colony algorithm (GABC) based on its results reported in the literature [14]. MABC follows the parameter

settings in the original paper of GABC [14]. It is clear that MABC works better in all cases and achieves better performance than GABC.

Summarizing the earlier statements, the ability of MABC 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.

4.3. Effects of each modification on the performance of MABC

In order to analyze the modifications respectively, we call the basic ABC with the proposed initialization as ABC1, and the random initialization with the proposed search mechanism (i.e., MABC without the proposed initialization) as ABC2. We compare the convergence performance of the different ABCs on the four test functions to see that how much the initialization and

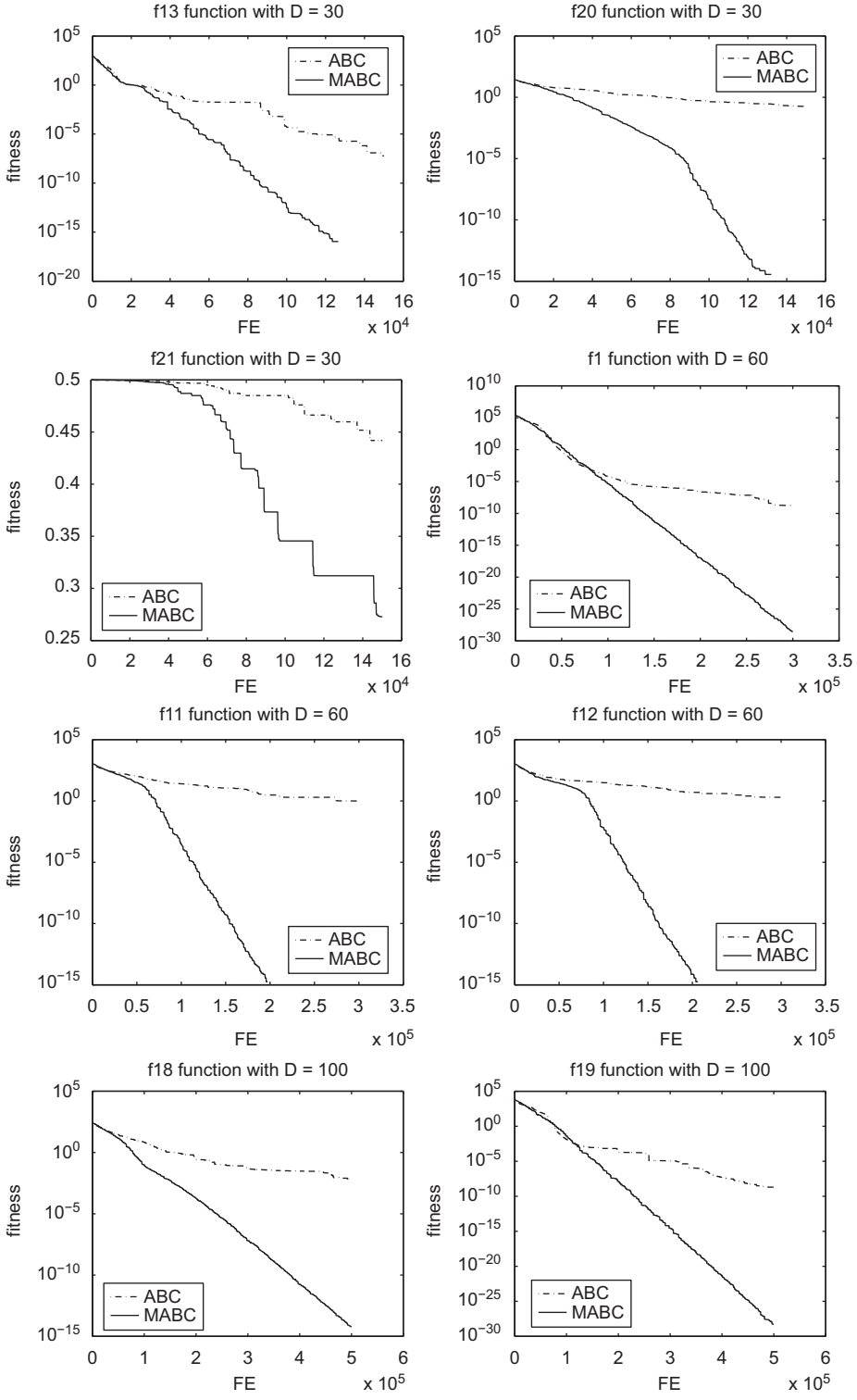

Fig. 1. Convergence performance of the different ABCs on the eight test functions.

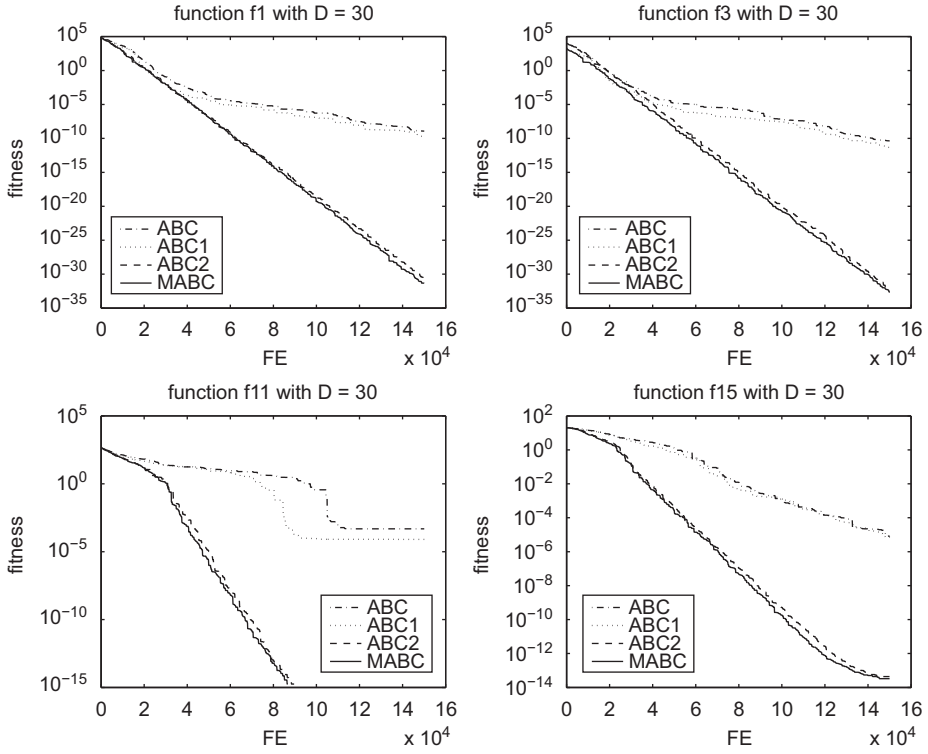

Fig. 2. Convergence performance of the different ABCs on the four test functions.

the search mechanism make contribution to improving the performance of the algorithm respectively. The results are presented in Fig. 2. It can be observed that, ABC1 and ABC2 are superior to ABC, which implies that both the initialization and the search mechanism have positive effect on the performance of the algorithm. Especially, ABC2 greatly outperforms ABC. On the other hand, the performance comparisons of ABC2 and MABC are not so apparent as those of ABC1 and MABC, which means that the search mechanism plays a pivotal role in the proposed algorithm. However, though the contribution of the initialization is far less than the search mechanism, the comparisons of MABC and ABC2, ABC and ABC1 show the initialization is at work.

5. Conclusion

In this paper, we have developed a novel optimization algorithm, called MABC, through introducing the modified solution search equation to ABC and proposing a new framework without probabilistic selection scheme and scout bee phase. In addition, the initial population is generated by combining chaotic systems with opposition-based learning method to enhance the global convergence. The experimental results tested on 28 benchmark functions show that MABC outperforms ABC and MABC. As a consequence, MABC may be a promising and viable tool to deal with complex numerical optimization problems. It is desirable to further apply MABC to solving those more complex real-world continuous optimization problems, such as clustering, data mining, design and optimization of communication networks. The future work includes the studies on how to extend MABC to handle those combinatorial optimization problems, such as flow shop scheduling problem, vehicle routing problem and traveling salesman problem.

Acknowledgments

This work is supported by National Nature Science Foundation of China (No. 60974082), Fundamental Research Funds for the Central Universities (No. JY10000970006, No. K50510700004) and Foundation of State Key Lab. of Integrated Services Networks of China.

References

[1] Tang KS, Man KF, Kwong S, He Q. Genetic algorithms and their applications. IEEE Signal Processing Magazine 1996;13:22–37.
[2] Kennedy J, Eberhart R. Particle swarm optimization. In: IEEE international conference on neural networks; 1995. p. 1942–8.
[3] Dorigo M, Stutzle T. Ant colony optimization. Cambridge: MA MIT Press; 2004.
[4] Simon D. Biogeography-based optimization. IEEE Transaction on Evolutionary Computation 2008;12:702–13.
[5] Wang DW. Colony location algorithm for assignment problems. Journal of Control Theory and Applications 2004;2:111–6.
[6] Karaboga D. An idea based on honey bee swarm for numerical optimization. Technical Report-TR06, Kayseri, Turkey: Erciyes University; 2005.
[7] Karaboga D, Basturk B. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. Journal of Global Optimization 2007;39:171–459.
[8] Karaboga D, Basturk B. On the performance of artificial bee colony (ABC) algorithm. Applied Soft Computing 2008;8:687–97.
[9] Karaboga D, Basturk B. A comparative study of artificial bee colony algorithm. Applied Mathematics and Computation 2009;214:108–32.
[10] Singh A. An artificial bee colony algorithm for the leaf-constrained minimum spanning tree problem. Applied Soft Computing 2009;9:625–31.
[11] Kang F, et al. Structural inverse analysis by hybrid simplex artificial bee colony algorithms. Computers & Structures 2009;87:861–70.
[12] Samrat L, et al. Artificial bee colony algorithm for small signal model parameter extraction of MESFET. Engineering Applications of Artificial Intelligence 2010;11:1573–2916.
[13] Storn R, Price K. Differential evolution–A simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization 2010;23:689–94.

[14] Zhu GP, Kwong S. Gbest-guided artificial bee colony algorithm for numerical function optimization. Applied Mathematics and Computation 2010, doi:10.1016/j.amc.2010.08.049.
[15] Akay B, Karaboga D. A modified artificial bee colony algorithm for real-parameter optimization. Information Sciences 2010, doi:10.1016/j.ins.2010.07.015.
[16] Alatas B. Chaotic bee colony algorithms for global numerical optimization. Expert Systems with Applications 2010;37:5682–7.
[17] Rahnamayan S, et al. Opposition-based differential evolution. IEEE Transaction on Evolutionary Computation 2008;12:64–79.
[18] Shang YW, Qiu YH. A note on the extended Rosenbrock function. Evolutionary Computation 2006;14:119–26.