Artificial Bee Colony Algorithm with Hierarchical Groups for Global Numerical Optimization

Laizhong Cui, Yanli Luo, Genghui Li(✉), and Nan Lu

College of Computer Science and Software Engineering,
Shenzhen University, Shenzhen, People's Republic of China
{cuilz,lunan}@szu.edu.cn, meidaile210@163.com,
li_genghui@126.com

Abstract. Artificial Bee Colony (ABC) algorithm is a relatively new swarm-based optimization algorithm, which has been shown to be better than or at least competitive to other evolutionary algorithms (EAs). Since ABC generally performs well in exploration but poorly in exploitation, ABC often shows a slow convergence. In order to address this issue and improve its performance, in this paper, we present a novel artificial bee colony algorithm with hierarchical groups, named HGABC. In employed bee phase of HGABC, the population is divided into three groups based on the fitness values of the food source positions, and three solution search strategies with different characteristics are correspondingly employed by different groups. Moreover, in onlooker bee phase, onlooker bees conduct exploitation in the most promising area of search space, instead of around some good solutions. In order to demonstrate the performance of HGABC, we compare HGABC with four other state-of-the-art ABC variants on 22 benchmark functions with 30*D*. The experimental results show that HGABC is better than other competitors in terms of solution accuracy and convergence rate.

Keywords: Artificial bee colony algorithm · Hierarchical group · Exploitation in the most promising area · Global numerical optimization

1 Introduction

Global optimization problems (GOPs) always arise in almost all of science research and engineering fields. population-based random optimization algorithms, such as genetic algorithm (GA) [1, 2], ant colony optimization (ACO) [3], particle swarm optimization (PSO) [4] and artificial bee colony algorithm (ABC), have been becoming a popular and promising way to handle these GOPs. ABC was developed by Karaboga [5] firstly, inspired by the collective foraging behavior of honey bee colony. The performance of ABC was demonstrated by comparing ABC with other evolutionary algorithms (EAs). Due to its simple structure, easy implementation and good performances, ABC has successfully attracted numerous researcher's attention and been applied to solve many practical engineering optimization problems [6–9].

DOI: 10.1007/978-3-319-52015-5_8

However, like other EAs, ABC often shows a slow convergence speed [10] since its solution search equation does well in exploration but poorly in exploitation. The search equation is the core operator of ABC, which significantly affects the performance of ABC. Therefore, in order to keep a better balance between exploration and exploitation, many new search equations were proposed. Inspired by PSO, Zhu and Kwong [11] introduced the information of the global best solution into the solution search equation to improve the exploitation ability of ABC (GABC). The experimental results showed that GABC is better than ABC on most benchmark functions. Karaboga and Akay [13] introduced two new parameters *i.e.*, modification rate (*MR*) and scaling factor (*SF*), into the solution equation to control frequency and magnitude of perturbation, respectively. In order to combine the advantage of different solution search equations, Kiran *et al.* [14] proposed a new method, which integrates five search equations to generate candidate solutions by the way of cooperation and competition. Moreover, Wang *et al.* [12] proposed the MEABC algorithm to improve the local and global search capability of the ABC, in which a pool of three distinct solution search strategies coexists throughout the search process and produces new solutions competitively. Recently, Karabaga *et al.* [15] proposed a new search equation for onlooker bees (qABC), which uses the valuable information of the best solution among the neighbors to improve the search efficiency of ABC. At the same time there are some improvements that blend with other operations [16, 17], and so on.

According to above considerations, the performance of ABC mainly depends on its solution search equation. Therefore, it is a promising way to improve the performance of ABC by introducing new search equation or integrating multiple search equations. In this paper, we follow this basic idea and propose an improved ABC algorithm, named HGABC. In employed bee phase of HGABC, all employed bees are divided into three groups according to the quality of their food source positions (fitness values), and different groups use different solution search equations. Moreover, to enhance the local exploitation ability in a promising area, in onlooker bee phase of HGABC, the most promising area is firstly recognized based on the quality of all food source positions, and onlooker bees conduct exploitation only around the positions located in the most promising area. The experimental results on 22 benchmark functions show that HGABC performs more competitively and effectively when it is compared with the other ABC variants.

The rest of this paper is organized as follows. Section 2 introduces ABC algorithm briefly. The proposed algorithm is presented detailedly in Sect. 3. Section 4 discusses and analyzes the experimental results. Finally, Sect. 5 concludes this paper.

2 Artificial Bee Colony Algorithm

Inspired by the waggle dance and foraging behaviors of honey bee colony, ABC algorithm has been developed. In ABC algorithm, the position of a food source represents a possible solution to the optimization problem, and the nectar amount of a food source position corresponds to the quality (fitness value) of the associated solution. The number of the employed bees or the onlooker bees is equal to the number of food sources. The basic ABC algorithm consists of four basic phases, namely initialization phase, employed bee phase, onlooker bee phase and scout bee phase.

2.1 Initialization Phase

In the initialization phase, the necessary parameters, *i.e.*, the number of food source position *SN*, the termination condition and the parameter *limit*, should be initialized firstly. Then, the initial food source positions are randomly produced in the whole search space by Eq. (1) as follows,

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

where $i = 1, 2, \cdots, SN$, $j = 1, 2, \cdots, D$, SN is the population size, and $x_{i,j}$ is the jth dimension of the ith solution. $x_{\min,j}$ and $x_{\max,j}$ are the lower and upper bounds of the jth dimension of the problem, respectively. $rand(0,1)$ is a random number in the range of [0,1]. The fitness value of the food source positions are calculated as follows,

$$fit(x_i) = \begin{cases} 1/(1 + f(x_i)) & \text{if}(f(x_i) \geq 0) \\ 1 + abs(f(x_i)) & \text{else} \end{cases} \tag{2}$$

where $f(x_i)$ is the objective function value of the ith food source position, and $fit(x_i)$ is the fitness value of the ith food source position.

2.2 Employed Bee Phase

In this phase, each employed bee flies to a distinct food source position to search for better food source position, and the candidate food source position is generated as follows,

$$v_{i,j} = x_{i,j} + \phi_{i,j}\left(x_{i,j} - x_{k,j}\right) \tag{3}$$

where $i = 1, 2, \cdots, SN$ and $j = 1, 2, \cdots, D$; $k \in \{1, 2, \ldots, SN\}$ and it is different from i; D is the dimension of the problem; $\phi_{i,j}$ is a random number in the range of [−1,1]. After the generation of the candidate solution v_i, if the candidate solution is better than the old one, the old solution will be replaced by the candidate solution. Otherwise, the old solution will be kept.

2.3 Onlooker Bee Phase

After all employed bees complete their search process, they will share the information (quality and position of food source) of their food source position to onlooker bees by assigning each food source position a selection probability, which is calculated as follows,

$$p_i = fit(x_i) \Big/ \sum_{i=1}^{SN} fit(x_i) \tag{4}$$

where p_i is the selection probability of the ith food source position, and each onlooker bee selects a food source position to perform search according to the selection probability of each food source position. The same search strategy and greedy selection method are employed by onlooker bees to perform further exploitation.

2.4 Scout Bees Phase

In the scout bee phase, if a certain food source position (solution) fails to be updated during a predetermined cycle (defined as "*limit*"), the corresponding employed bee becomes a scout bee and the food source position should be replaced by a new one, which is generated randomly according to Eq. (1).

After the initialization, ABC enters a loop of employed bee phase, onlooker phase and scout bee phase until the terminal condition is satisfied.

3 Artificial Bee Colony Algorithm with Hierarchical Groups (HGABC)

In the original ABC or other ABC variants [11], only one search strategy is employed by employed bee and onlooker bee, which may result in that the search ability of these methods are limited. Inspired by the observation in the team work of human being, since each member in the team has different characteristics, such as knowledge, attitude and skill, the whole team usually is divided into multiple groups according to their abilities, and each group takes different responsibilities or tasks. By this way, the work efficiency can be significantly improved. In original ABC, although the colony contains three types of bees, *i.e.*, employed bee, onlooker bee and scout bee, different types of bees are responsible for different search abilities. However, ABC treats all employed bees (or onlooker bee) equally because all employed bees (or onlooker bees) employ the same search strategy. While in real bee colony, each employed bee (or onlooker bee) is a unique individual, and the search ability of them may be different from each other. Therefore, different employed bees (and onlooker bees) may adopt different search strategies in fact.

According to above consideration, in this paper, we propose a novel artificial bee colony algorithm with hierarchical groups, named HGABC. To be specific, in HGABC, the employed bees are divided into three groups based on the quality of their food source positions, and different groups employ different search strategies so as to be responsible for different search abilities. Moreover, in order to pay more attention to the exploitation in the most promising area, all onlooker bees only search around the food source positions which locate in the most promising area. Similarly, three search strategies could be used by onlooker bees in a random manner. The proposed strategies are described in detail as follows.

3.1 Division of Employed Bees and Search Strategies

In ABC, each employed bee occupies a food source position. Since each employed bee has distinct ability and should adopt different search strategy, in order to differentiate employed bees, we firstly divide the employed bees into three group based on the quality of their own food source positions. To be specific, the employed bees firstly sort from best to worst based on the quality of their food source positions. The first $a \cdot SN$ employed bees, the medium $b \cdot SN$ employed bees and the last $c \cdot SN$ employed bees, respectively constitute the high group, medium group and low group, where $a, b, c \in [0,\ 1]$ and $a + b + c = 1$.

The high group includes some current good solutions, which may be located in the local optimal areas or the global optimal area. Therefore, its employed bees should learn the beneficial information from the current best solution and conduct exploitation toward the current best solution. The employed bees belonging to the high group adopt the search strategy as follows,

$$v_{i,j} = x_{k,j} + \varphi_{i,j}(xbest_j - x_{k,j}) \tag{5}$$

where *xbest* is the current best solution; x_k is randomly selected from the population, which is different from x_i and *xbest*; $\varphi_{i,j}$ is a random number in the range of [0,1]; j is a randomly selected dimension.

With respect to the medium group, it consists of some neither better nor worse solutions that are not far from or close to the global optimal area. Their employed bees should take the responsibility of obtaining balance between the exploitation and exploration. Therefore, the employed bees in the medium group use the search strategy as follows,

$$v_{i,j} = x_{k,j} + \phi_{i.j}(x_{k,j} - x_{q,j}) + \varphi_{i,j}(xbest_j - x_{k,j}) \tag{6}$$

where *xbest* is the current best solution, and x_k and x_q are randomly selected from the population, which are distinct from each other and different from x_i and *xbest*. $\varphi_{i,j}$ is a random number in the range of [0,1], and $\phi_{i,j}$ is a random number in the range of $[-1,\ 1]$. j is a randomly selected dimension.

Regarding to the low group, it contains the current bad solutions that may be far from the local optimal areas or the global optimal area with a high probability, and its employed bees should be responsible for exploration by exploiting new areas randomly. Therefore, the third kind of employed bees employ the search strategy as follows,

$$v_{i,j} = x_{k,j} + \phi_{i.j}(x_{k,j} - x_{q,j}) \tag{7}$$

where x_k and x_q are randomly selected from the population, which are distinct from each other and different from x_i. $\phi_{i,j}$ is a random number in the range of $[-1,\ 1]$. j is a randomly selected dimension.

Overall, in our proposed algorithm, all employed bees are divided into three groups, namely the high group, the medium group and the low group. The employed bees in different groups adopt different search strategies and undertake different search tasks. More specifically, the high group's employed bees pay more attention to exploitation, the low group's employed bees focus on exploration, and the medium group's employed bees are responsible to balance between exploration and exploitation.

3.2 Search Strategy of Onlooker Bee

In original ABC, after all employed bees complete their search tasks, the onlooker bees start to work depending on the information provided by the employed bees. To be specific, each onlooker bee will select a food source position to conduct exploitation by the roulette wheel method, which is a time-consuming procedure. Moreover, the better the quality of the food source position is, the bigger the selection probability is. In order to pay more attention to the promising area and accelerate the convergence, in this paper, we present a most promising area search strategy for onlooker bee. The details are described as follows.

```
Algorithm 1. Procedure of onlooker bee phase
01: Locate in the most promising area
02: for i=1:SN
03:     Randomly select a position $x_s$ in the most promising area
04:     u=rand(0,1);
05:     if $u \le s_1$
06:         Generate the candidate solution $v_s$ use Eq. (5)
07:     else if $u \le s_2$
08:         Generate the candidate solution $v_s$ use Eq. (6)
09:     else
10:         Generate the candidate solution $v_s$ use Eq. (7)
11:     end if
12:     if $f(v_s) \le f(x_s)$
13:         Replace $x_s$ by $v_s$, counter(s)=0
14:     else
15:         counter(s)=counter(s)+1;
16:     end if
17 : end for
```

Fig. 1. The pseudo-code of onlooker bee phase

In order to recognize the most promising area, each food source position denotes an area. To be specific, for the *i*th food source position, if the Euclidean distance between food source position x_i and x_j ($j = 1, 2, \cdots, SN$ and $j \neq i$) is less than the radius *r*, the position x_j belongs to the area located by the position x_i. Moreover, the radius *r* is calculated as follows,

$$r = \frac{\sum_{i=1}^{SN-1} \sum_{j=i+1}^{SN} d(x_i, x_j)}{SN(SN-1)/2} \tag{8}$$

where $d(x_i, x_j)$ is the Euclidean distance between x_i and x_j, and *SN* is the number of the food source positions.

Obviously, there are *SN* areas in the search space and the best quality area based on the average fitness value of its members is treated as the most promising area. After the most promising area is identified, the onlooker bees only fly to a randomly selected food source position located in the most promising area to search.

Algorithm 2.The procedure of HGABC

```
01: Initialization: Generate $SN$ solutions according to Eq. (1)
02: while $FES < maxFES$
03:     Sort the population and divide the population into three groups
04:     for $i = 1$ to $SN$ // employed bee phase
05:         if $i \le a \cdot SN$ // the high group
06:             Generate a new solution $v_i^G$ use Eq. (5).
07:         else if $i \le (a + b) \cdot SN$ // the medium group
08:             Generate a new solution $v_i^G$ use Eq. (6).
09:         else // the low group
10:             Generate a new solution $v_i^G$ use Eq. (7).
11:         end if
12:             Evaluate the new solution $v_i^G$
13:         if $f(v_i^G) \le f(x_i^G)$
14:             Replace $x_i^G$ by $v_i^G$, $counter(i)=0$
15:         else
16:             $counter(i) = counter(i)+1$
17:         end if
18:     end for
19:     Algorithm 1 // onlooker bee phase
20:     $FES=FES+2SN$
21:     Select $X_{\max}^G$ with max $counter$ value // scout bee phase
22:     if $counter(\max) > limit$
23:         Replace $X_{\max}^G$ by a new solution generated according to Eq.(1)
24:         $FES=FES+1$, $counter(\max)=0$
25:     end if
26: end while
Output: The food source (solution) with the smallest objective value
```

Fig. 2. The pseudo-code of HGABC

Moreover, to make the onlooker bees show different search abilities and keep a better balance between exploration and exploitation, the above three search equations (Eqs. (5), (6) and (7)) are employed by onlooker bees in a random manner based on two control parameters s_1 and s_2.

Table 1. Benchmark functions in experiments

Name	Function	Range	Min	Accept
Sphere	$f_1(x) = \sum_{i=1}^{D} x_i^2$	$[-100,\ 100]^D$	0	1×10^{-8}
Elliptic	$f_2(x) = \sum_{i=1}^{D} (10^6)^{\frac{i-1}{D-1}} x_i^2$	$[-100,\ 100]^D$	0	1×10^{-8}
SumSquare	$f_3(x) = \sum_{i=1}^{D} i x_i^2$	$[-10,\ 10]^D$	0	1×10^{-8}
sumPower	$f_4(x) = \sum_{i=1}^{D} \|x_i\|^{(i+1)}$	$[-1,\ 1]^D$	0	1×10^{-8}
Schwefel 2.22	$f_5(x) = \sum_{i=1}^{D} \|x_i\| + \prod_{i=1}^{D} \|x_i\|$	$[-10,\ 10]^D$	0	1×10^{-8}
Schwefel 2.21	$f_6(x) = max\{\|x_i\|,\ 1 \le i \le n\}$	$[-100,\ 100]^D$	0	1×10^{0}
Step	$f_7(x) = \sum_{i=1}^{D} (\lfloor x_i + 0.5 \rfloor)^2$	$[-100, 100]^D$	0	1×10^{-8}
Exponential	$f_8(x) = \exp\left(0.5 * \sum_{i=1}^{D} x_i\right)$	$[-10, 10]^D$	0	1×10^{-8}
Quartic	$f_9(x) = \sum_{i=1}^{D} i x_i^4 + random[0, 1]$	$[-1.28, 1.28]^D$	0	1×10^{-1}
Rosenbrock	$f_{10}(x) = \sum_{i=1}^{D-1} \left[100\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2\right]$	$[-5,\ 10]^D$	0	1×10^{-1}
Rastrigin	$f_{11}(x) = \sum_{i=1}^{D} \left[x_i^2 - 10\cos(2\pi x_i) + 10\right]$	$[-5.12, 5.12]^D$	0	1×10^{-8}
NCRastrigin	$f_{12}(x) = \sum_{i=1}^{D} \left[y_i^2 - 10\cos(2\pi y_i) + 10\right]$ $y_i = \begin{cases} x_i & \|x_i\| < \frac{1}{2} \\ \frac{round(2x_i)}{2} & \|x_i\| \ge \frac{1}{2} \end{cases}$	$[-5.12, 5.12]^D$	0	1×10^{-8}
Griewank	$f_{13}(x) = 1/4000 \sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$[-600,\ 600]^D$	0	1×10^{-8}
Schwefel2.26	$f_{14}(x) = 418.98288727243380 * D - \sum_{i=1}^{D} x_i \sin\left(\sqrt{\|x_i\|}\right)$	$[-500, 500]^D$	0	1×10^{-8}
Ackley	$f_{15}(x) = 20 + e - 20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D} \cos(2\pi x_i)\right)$	$[-50, 50]^D$	0	1×10^{-8}

(*continued*)

Table 1. (*continued*)

Name	Function	Range	Min	Accept
Penalized1	$f_{16}(x) = \frac{\pi}{D}\left\{10\sin^2(\pi y_1) + \sum_{i=1}^{D-1}(y_i - 1)^2\left[1 + 10\sin^2(\pi y_{i+1})\right] + (y_D - 1)^2\right\} + \sum_{i=1}^{D} u(x_i, 10, 100, 4)$ $y_i = 1 + 1/4(x_i + 1),\ u_{x_i,a,k,m} = \begin{cases} k(x_i - a)^m & x_i > a \\ 0 & -a \le x_i \le a \\ k(-x_i - a)^m & x_i < a \end{cases}$	$[-100, 100]^D$	0	1×10^{-8}
Penalized2	$f_{17}(x) = \frac{1}{10}\left\{\sin^2(\pi x_1) + \sum_{i=1}^{D-1}(x_i - 1)^2\left[1 + \sin^2(3\pi x_{i+1})\right] + (x_D - 1)^2\left[1 + \sin^2(2\pi x_{i+1})\right]\right\} + \sum_{i=1}^{D} u(x_i, 5, 100, 4)$	$[-100,\ 100]^D$	0	1×10^{-8}
Alpine	$f_{18}(x) = \sum_{i=1}^{D-1} \lvert x_i \cdot \sin(x_i) + 0.1 \cdot x_i \rvert$	$[-10,\ 10]^D$	0	1×10^{-8}
Levy	$f_{19}(x) = \sum_{i=1}^{D-1}(x_i - 1)^2\left[1 + \sin^2(3\pi x_{i+1})\right] + \sin^2(3\pi_1) + \lvert x_D - 1\rvert\left[1 + \sin^2(3\pi x_D)\right]$	$[-10,\ 10]^D$	0	1×10^{-8}
Weierstrass	$f_{20}(x) = \sum_{i=1}^{D}\left(\sum_{k=0}^{k_{\max}}\left[a^k\cos\left(2\pi b^k(x_i + 0.5)\right)\right] - D\sum_{k=0}^{k_{\max}}\left[a^k \cos\left(2\pi b^k 0.5\right)\right], a = 0.5, b = 3, k_{\max} = 20$	$[-1,\ 1]^D$	0	1×10^{-8}
Himmelblau	$f_{21}(x) = 1/D\sum_{i=1}^{D}\left(x_i^4 - 16x_i^2 + 5x_i\right)$	$[-5,\ 5]^D$	−78.3	−78
Michalewicz	$f_{22}(x) = -\sum_{i=1}^{n}\sin(x_i)\sin^{20}\left(\frac{i \times x_i^2}{\pi}\right)$	$[0,\ \pi]^D$	$-D$	$-D + 1$

According to the proposed modifications, the pseudo-code of onlooker bee phase is shown in Fig. 1 and the completed pseudo-code of the proposed algorithm HGABC is shown in Fig. 2.

4 Experiments

In order to demonstrate the performance of our proposed algorithm HGABC, we compare HGABC with four ABC methods, *i.e.*, the basic ABC [5], GABC [11], qABC [15] and MEABC [12] on 22 benchmark functions with 30*D*, which are listed in Table 1. To make a fair comparison, for all compared algorithms, *SN* and *limit* are set to 50 and $SN \cdot D$, respectively. Other parameters are set the same as the original papers. For HGABC, a, b and c are respectively set to 0.2, 0.3 and 0.5; s_1 and s_2 are set to 0.25 and 0.75, respectively. The maximal number of function evaluation (*maxFES*) is used as the termination condition, which is set to $5000 \cdot D$. All algorithms conduct 25 times independent runs on each function. The experimental results are given in Table 2. For the sake of clarity, the best results are marked in **boldface**. Moreover, the Wilcoxon's rank sum test at 5% significance level on results gained by two competing algorithms is also conducted to show the significant differences between HGABC and other ABC methods. The results of the test are represented as "+", "−", "=", which mean that the compared algorithm is significantly better than, worse than, equal to HGABC, respectively.

As shown in Table 2, the metric of *mean* and *std* respectively denote the average value and standard deviation of the best objective function value of 25 independent runs. According to these metrics, HGABC successfully gets the best results on all functions except that f_4, f_{10} and f_{14}. To be specific, HGABC is better than ABC, GABC, qABC and MEABC on 18, 12, 18 and 9 functions, respectively. On the contrary, HGABC is only beaten by GABC and MEABC on 1 and 1 function, respectively. Moreover, ABC and qABC is unable to perform better than HGABC on any cases.

In addition, in order to clearly show the convergence speed and robustness of different algorithms, more experimental results about the average FES (AVEN) and success rate (SR) are also given in Table 1. AVEN represents the average FES needed to reach the threshold defined in Table 1. In Table 1, "NAN" denotes that the algorithm cannot get any solutions, whose objective function is smaller than the acceptable value in 25 independent runs. SR represents the ratio of the number of success runs in the 25 independent runs. The success run means that algorithm can find the solution, whose objective function value is less than the acceptable value. Obviously, the search accuracy of HGABC is better than or equal to other algorithms on all functions, excluding f_4, f_{10} and f_{14}. Similarly, the SR of HGABC is 100% on all functions except f_{10}, on which all algorithms are unable to get a 100% success rate. Overall, HGABC is better than the competitors in terms of solution accuracy, convergence speed and robustness.

Table 2. Comparison results on 22 test functions with 30D

Alg	ABC
	mean(std)
	AVEN(SR)
		GABC
		mean(std)
		AVEN(SR)
			qABC
			mean(std)
			AVEN(SR)
				MEABC
				mean(std)
				AVEN(SR)
					HGABC
					mean(std)
					AVEN(SR)

f_1	8.05e−18 (6.08e−18) − **100**/82934	6.97e−33 (4.93e−33) − **100**/50130	1.60e−15 (1.32e−15) − **100**/72618	3.45e−40 (4.66e−40) − **100**/44910	**2.18e−57** **(2.33e−57)** **100/32314**
f_2	4.77e−10 (3.76e−10) − **100**/135230	1.92e−26 (2.12e−26) − **100**/76150	1.53e−10 (3.87e−10) − **100**/123870	6.17e−37 (6.37e−37) − **100**/55908	**7.59e−55** **(1.12e−54)** **100/40498**
f_3	1.55e−19 (1.31e−19) − **100**/75366	2.98e−34 (2.38e−34) − **100**/45478	3.14e−16 (2.92e−16) − **100**/63946	2.74e−41 (2.06e−41) − **100**/41710	**2.91e−58** **(2.97e−58)** **100/30186**
f_4	2.41e−31 (9.09e−31) − **100**/23266	1.83e−52 (6.33e−52) − **100**/14106	3.01e−21 (1.31e−20) − **100**/13342	**4.93e−86** **(1.20e−85) +** **100**/12014	5.32e−76 (1.81e−75) **100/10022**
f_5	6.55e−11 (2.12e−11) − **100**/125030	5.95e−18 (1.76e−18) − **100**/77478	1.09e−08 (3.89e−09) − 48/148340	1.47e−21 (6.87e−22) − **100**/66784	**5.85e−30** **(3.45e−30)** **100/48882**
f_6	4.35e+00 (8.60e−01) − 0/NAN	2.55e−01 (1.30e−01) − **100**/109060	9.36e−02 (1.79e−02) − **100/35898**	3.00e+00 (1.37e+00) − 4/131500	**4.57−03** **(3.39−03)** **100**/58034
f_7	**0.00e+00** **(0.00e+00) =** **100**/11314	**0.00e+00** **(0.00e+00) =** **100**/10314	**0.00e+00** **(0.00e+00) =** **100/6482**	**0.00e+00** **(0.00e+00) =** **100**/18974	**0.00e+00** **(0.00e+00)** **100**/11962
f_8	**7.18e−66** (4.37e−73**) =** **100**/150	**7.18e−66** (9.22e−77**) =** **100**/150	**7.18e−66** (2.98e−72**) =** **100**/150	**7.18e−66** (3.63e−79**) =** **100/100**	**7.18e−66** **(7.66e−80)** **100**/150
f_9	6.42e−02 (1.37e−02) − **100**/93186	2.80e−02 (6.51e−03) − **100**/41966	2.78e−02 (8.01e−03) − **100/11018**	2.98e−02 (8.07e−03) − **100**/45748	**1.26e−02** **(2.78e−03)** **100**/20674
f_{10}	**6.79e−02** **(5.93e−02) =** 72/120030	8.21e−01 (3.73e+00) **=** 68/77515	5.56e−01 (6.12e−01) − 36/**75828**	9.34e−02 (1.17e−01) **=** **80**/115880	1.99e−01 (2.77e−01) 56/98729
f_{11}	2.68e−14 (1.03e−13) − **100**/99214	**0.00e+00** **(0.00e+00) =** **100**/68134	1.23e−10 (1.68e−10) − **100**/112510	**0.00e+00** **(0.00e+00) =** **100**/51876	**0.00e+00** **(0.00e+00)** **100/36926**
f_{12}	4.25e−13 (1.57e−12) − **100**/110050	**0.00e+00** **(0.00e+00) =** **100**/76642	4.95e−10 (5.78e−10) − **100**/119730	**0.00e+00** **(0.00e+00) =** **100**/55650	**0.00e+00** **(0.00e+00)** **100/39794**
f_{13}	3.08e−04 (1.54e−03) − 96/96783	4.51e−08 (2.25e−07) **=** 96/61688	2.48e−12 (6.35e−12) − **100**/95790	**0.00e+00** **(0.00e+00) =** **100**/53762	**0.00e+00** **(0.00e+00)** **100/38790**
f_{14}	4.51e−12 (1.59e−12) − **100**/84338	**2.18e−13** (6.03e−13**) +** **100**/65670	3.88e−10 (1.46e−09) − **100**/112170	2.76e−12 (1.50e−12) − **100**/53292	3.64e−12 **(0.00e+00)** **100/40506**

(*continued*)

Table 2. (*continued*)

Alg	ABC
	mean(std)
	AVEN(SR)
		GABC
		mean(std)
		AVEN(SR)
			qABC
			mean(std)
			AVEN(SR)
				MEABC
				mean(std)
				AVEN(SR)
					HGABC
					mean(std)
					AVEN(SR)

f_{15}	3.83e−09 (2.27e−09) − 96/144000	1.49e−14 (2.92e−15) − **100**/89178	1.61e−06 (8.36e−07) − 0/NAN	6.79e−15 (1.97e−15) − **100**/76954	**3.84e−15** **(9.84e−16)** **100/55678**
f_{16}	1.29e−18 (1.76e−18) − **100**/79398	**1.57e−32** **(5.59e−48) =** **100**/45786	4.12e−15 (7.77e−15) − **100**/63282	**1.57e−32** **(5.59e−48) =** **100**/40080	**1.57e−32** **(5.59e−48)** **100/28266**
f_{17}	8.19e−18 (1.71e−17) − **100**/84730	4.06e−33 (2.30e−33) − **100**/49750	1.83e−15 (1.51e−15) − **100**/75322	**1.50e−33** **(0.00e+00) =** **100**/44876	**1.50e−33** **(0.00e+00)** **100/30854**
f_{18}	3.15e−06 (1.85e−06) − 0/NAN	3.88e−07 (6.54e−07) − 16/129980	1.43e−05 (3.92e−05) − 0/NAN	1.78e−17 (6.15e−17) − **100**/68064	**7.80e−31** **(1.20e−30)** **100/49198**
f_{19}	8.23e−14 (1.25e−13) − **100**/91734	1.39e−31 (1.41e−32) **=** **100**/50934	9.29e−10 (9.59e−10) − **100**/123530	**1.35e−31** **(2.23e−47) =** **100**/42450	**1.35e−31** **(2.23e−47)** **100/32006**
f_{20}	3.06e−02 (3.75e−02) − 0/NAN	3.60e−02 (4.19e−02) − 0/NAN	8.71e−03 (8.44e−03) − 0/NAN	**0.00e+00** **(0.00e+00) =** **100**/89724	**0.00e+00** **(0.00e+00)** **100/68534**
f_{21}	**−7.83e+01** (4.10e−15) **=** **100**/26934	**−7.83e+01** (5.02e−15) **=** **100**/15986	**−7.83e+01** (7.11e−15) **=** **100/6838**	**−7.83e+01** (5.80e−15) **=** **100**/12760	**−7.83e+01** **(2.90e−15)** **100**/7666
f_{22}	−2.999e+01 (8.26e−04) − **100**/25362	−2.999e+01 (1.01e−03) − **100**/21778	**−3.00e+01** (1.12e−05) **=** **100/2310**	**−3.00e+01** **(2.19e−07) =** **100**/17126	**−3.00e+01** (3.29e−06) **100**/9530
+/ =/−	0/4/18	1/9/12	0/4/18	1/12/9	

To clearly show the advantages of HGABC, the convergence curves of the *mean* on some representative functions are plotted in Fig. 3. It can be seen from Fig. 3 that HGABC converges faster than ABC, GABC, qABC and MEABC on both unimodal functions and multimodal functions. In conclusion, the experimental results demonstrate that our modifications of employed bee phase and onlooker bee phase can ontain a better balance between exploration and exploitation, and effectively improve the performance of ABC.

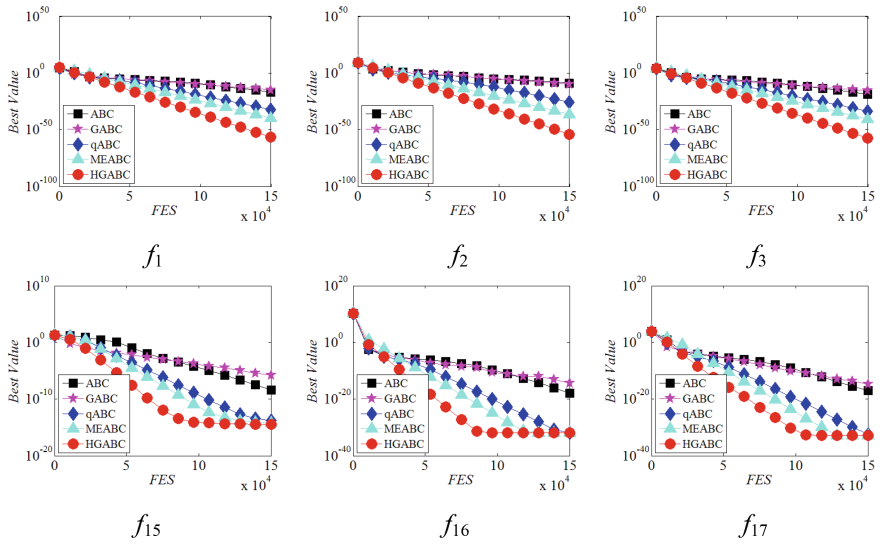

Fig. 3. Convergence curve of all ABCs on some representative functions

5 Conclusion

This paper presents a new ABC algorithm, called HGABC. In HGABC, in order to differentiate the employed bees, the employed bees are divided into three groups according to the quality of their food source positions. The employed bees belonging to different groups employ different search strategies and are responsible for different search abilities. Moreover, to speed up convergence and pay more attention to the most promising area, the onlooker bees using three search strategies in a random manner only exploit in the most promising area. The comparison results on 22 benchmark functions show that HGABC can significantly improve the performance of ABC and outperform other ABC methods in terms of solution accuracy, convergence speed and robustness. In future, we can apply HGABC to handle real world engineering problems.

Acknowledgments. This work is supported by the National Natural Science Foundation of China under Grant 61402294, Major Fundamental Research Project in the Science and Technology Plan of Shenzhen under Grants JCYJ20140509172609162, JCYJ20-140828163633977, JCYJ20140418181958501, and JCYJ20160310095523765.

References

1. Holland, J.H.: Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. U Michigan Press (1975)
2. Yang, C., Gui, W., Kong, L., et al.: A genetic algorithm based optimal scheduling system for full-filled tanks in the processing of starting materials for alumina production. Can. J. Chem. Eng. **86**(4), 804–812 (2008)
3. Dorigo, M., Birattari, M., Stutzle, T.: Ant colony optimization. IEEE Comput. Intell. Mag. **1**(4), 28–39 (2006)
4. Kennedy, J.: Particle swarm optimization. In: Encyclopedia of Machine Learning, pp. 760–766. Springer US, Heidelberg (2011)
5. Karaboga, D.: An idea based on honey bee swarm for numerical optimization. Technical report-tr06, Erciyes university, engineering faculty, computer engineering department (2005)
6. Karaboga, D., Basturk, B.: A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. J. Global Optim. **39**(3), 459–471 (2007)
7. Karaboga, D., Akay, B.: A modified artificial bee colony (ABC) algorithm for constrained optimization problems. Appl. Soft Comput. **11**(3), 3021–3031 (2011)
8. Singh, A.: An artificial bee colony algorithm for the leaf-constrained minimum spanning tree problem. Appl. Soft Comput. **9**(2), 625–631 (2009)
9. Li, G., Niu, P., Xiao, X.: Development and investigation of efficient artificial bee colony algorithm for numerical function optimization. Appl. Soft Comput. **12**(1), 320–332 (2012)
10. Karaboga, D., Akay, B.: A comparative study of artificial bee colony algorithm. Appl. Math. Comput. **214**(1), 108–132 (2009)
11. Zhu, G., Kwong, S.: Gbest-guided artificial bee colony algorithm for numerical function optimization. Appl. Math. Comput. **217**(7), 3166–3173 (2010)
12. Wang, H., Wu, Z., Rahnamayan, S., et al.: Multi-strategy ensemble artificial bee colony algorithm. Inf. Sci. **279**, 587–603 (2014)
13. Akay, B., Karaboga, D.: A modified artificial bee colony algorithm for real-parameter optimization. Inf. Sci. **192**, 120–142 (2012)
14. Kiran, M.S., Hakli, H., Gunduz, M., et al.: Artificial bee colony algorithm with variable search strategy for continuous optimization. Inf. Sci. **300**, 140–157 (2015)
15. Karaboga, D., Gorkemli, B.: A quick artificial bee colony (qABC) algorithm and its performance on optimization problems. Appl. Soft Comput. **23**, 227–238 (2014)
16. Qiu, M., Ming, Z., Li, J., et al.: Phase-change memory optimization for green cloud with genetic algorithm. IEEE Trans. Comput. **64**(12), 3528–3540 (2015)
17. Gai, K., Qiu, M., Zhao, H.: Cost-aware multimedia data allocation for heterogeneous memory using genetic algorithm in cloud computing. IEEE Trans. Comput. (2016) doi:10.1109/TCC.2016.2594172