Cluster Comput (2019) 22:S2581–S2588
https://doi.org/10.1007/s10586-017-1343-0

A hybrid artificial bee colony algorithm with modified search model for numerical optimization

Xiuqin Pan[1] · Yong Lu[1] · Na Sun[1] · Sumin Li[1]

Received: 15 September 2017 / Revised: 19 October 2017 / Accepted: 4 November 2017 / Published online: 13 November 2017

Abstract Artificial bee colony (ABC) is an effective optimization algorithm, which has been used in various practical applications. However, the standard ABC suffers from low accuracy of solutions and slow convergence rate. To address these issues, a hybrid ABC (called HABC) is proposed in this paper. In HABC, two improved strategies are utilized. First, a new search model is designed based on the best-of-random mutation scheme. Second, new solutions are generated by updating multiple dimensions. To verify the performance of HABC, twelve numerical optimization problems are tested in the experiments. Results of HABC are compared the standard ABC and two other improved ABC versions. The comparison show that our approach can effectively improve the optimization performance.

Keywords Artificial bee colony · Swarm intelligence · Search model · Numerical optimization · Global optimization

1 Introduction

Many engineering problems can be boiled down to optimization problems. Therefore, optimization has become an indispensable computational tool in various engineering areas. Traditional optimization techniques include gradient descent, Newton method, conjugate gradient, Lagrange multiplier, least square method, and so on. The above methods have complete mathematical theory and fast convergence speed, but they always lack of generality and depend on the characteristics of problems. With the development of science and technology, optimization problems become more and more complex, such as multimodal, discrete, strong constraints, large scale, and nonlinear. For such complex problems, it is difficult to solve them by traditional mathematical methods.

In the past decades, some new optimization algorithms based on swarm intelligence have been proposed/designed, such as particle swarm optimization (PSO) [1], ant colony optimization (ACO) [2], firefly algorithm (FA) [3], artificial bee colony (ABC) [4], cuckoo search (CS) [5], and bat algorithm (BA) [6]. These algorithms are modeled on some social behaviors of nature. For example, PSO simulates the fish schooling and birds flocking. FA is inspired by the mating attraction of fireflies. Among these swarm intelligence algorithms, ABC shows some advantages: (1) it has only one control parameter except for population size; and (2) it has good global search ability. Some previous studies show that ABC performs better than PSO and some population-based algorithms [7]. Due to ABC's advantages, it has been widely used in many optimization problems.

Although ABC has shown good search performance, it has some shortcomings, slow convergence and low accuracy of solutions. To tackle these issues, this paper proposes a hybrid ABC (called HABC), which employs two improved strategies. In the first strategy, a new search model is constructed based on the best-of-random mutation scheme [8]. In the standard ABC, new solutions are generated by randomly updating a dimension of their corresponding parent solutions. It means that the differences between a new solution and its corresponding parent solution are very small. This easily results in slow convergence. Then, the second strategy focuses on updating multiple dimensions to generate offspring. Experimental study is carried out on twelve numerical optimization problems. HABC is compared with

✉ Xiuqin Pan
amycun@163.com; lucun1009@126.com

[1] School of Information Engineering, Minzu University of China, Beijing 100081, China

ABC and two other ABC variants. Results show that HABC can achieve better solutions on most test problems.

The rest of the paper is organized as follows. ABC and its literature review are presented in Sect. 2. The proposed approach is described in Sect. 3. Results and discussions are given in Sect. 4. Finally, we summarize the conclusions in Sect. 5.

2 Related works

2.1 Artificial bee colony

ABC is an effective optimization algorithm developed by Karaboga [4]. In ABC, there are three different kinds of bees, which are called employed bees, onlooker bees, and scouts, respectively. The task of bees is to find good food sources with more nectar. Therefore, each food source is regarded as a solution. The nectar of a food source determines the fitness of the corresponding solution.

Let $X_i = \{x_{i,1}, x_{i,2}, \ldots, x_{i,D}\}$ be the ith food source (solution) in the population, where $i = 1, 2, \ldots, N$. Each employed bee chooses a solution (X_i) and searches its neighborhood to find a new solution (V_i). This process can be defined as follows [4].

$$v_{i,j} = x_{i,j} + \phi_{i,j} \cdot \left(x_{i,j} - x_{k,j}\right) \tag{1}$$

where $j \in [1, D]$ is a randomly selected dimension index, and D is the problem size. X_k is a randomly chosen solution, and $k \in [1, N]$. The weight $\phi_{i,j}$ is randomly generated between -1.0 and 1.0.

The employed bees can pass their search information to the onlooker bees, which select some good food sources and conduct the second search. This can help to find better solutions. The selection method is decided by the fitness values of solutions. In [4], a roulette selection is used and the selection probability is calculated as follows.

$$p_i = \frac{fit_i}{\sum_{i=1}^{N} fit_i} \tag{2}$$

where fit_i is the fitness value of X_i.

If the fitness value of a solution cannot be improved in *limit* generations, where *limit* is a predefined value, a scout generate a new solution to update the original one [4].

$$x_{i,j} = low_j + rand_j \cdot (up_j - low_j) \tag{3}$$

where $\left[low_j, up_j\right]$ is the constraint, and $rand_j$ is a random value between 0 and 1.0.

2.2 Literature review

In recent years, the research of ABC has attracted much attention. Different versions of ABC have been proposed to solve benchmark or practical problems. In the following, a brief literature review of these works is presented.

Zhu and Kwong [9] designed a *gbest* (the global best solution) guided ABC (called GABC), which shares the search experiences of the *gbest* with other solutions. Simulation results show that GABC can improve the convergence rate on several benchmark functions. Akay and Karaboga [10] presented a modified ABC, which introduced two strategies. A new parameter *MR* is used to control the number of updating dimensions. The literature [10] pointed out that changing only one dimension of parent solutions in the standard ABC can slow down the convergence rate. By adjusting *MR*, the ABC algorithm can be accelerated. In [11], Gao and Liu introduced another modified ABC (called MABC), which used a new search equation based on DE mutation scheme. It is known that different search models may result in different search abilities. Some researchers used multiple search models in ABC. Wang et al. [12] presented a multi-strategy ensemble of ABC (called MEABC), which constructed a strategy pool consisting of three search models. Kiran et al. [13] designed a variable search strategy for ABC (ABCVSS). Gao et al. [14] developed a novel ABC with three search strategy (MuABC), which also used a strategy pool.

In [15], Zhou et al. combined Gaussian distribution model with ABC to improve the search performance. In [16], Zhou et al. proposed a neighborhood search operator to make a balance between explorative and exploitative abilities. Cui et al. [17] developed a ranking based adaptive ABC (called ARABC), in which a food source with a high rank can be chosen in a high probability. Results show that ARABC outperforms many state-of-art algorithms. Liang et al. [18] used adaptive differential evolution (DE) operators to enhance the performance of ABC. In [19], Wang et al. used an external archive to accelerate the convergence of ABC. In the search process, some good solutions are stored in an external archive. The search model is modified by introduced these good solutions to guide the search. Like [19], Li and Yang [20] proposed a new ABC with memory (called ABCM), which memorized pervious successful search information to guide the behavior of bees. Simulation results show that ABCM is superior to ABC and quick ABC (qABC) [21]. Sharma and Pant [22] introduced a shuffled ABC to solve constrained optimization problems, in which ABC is combined with the shuffled frog-leaping algorithm.

3 Proposed approach

As mentioned before, the standard ABC suffers from slow convergence rate and low accuracy of solutions. To address these issues, a hybrid ABC (HABC) is proposed as follows.

The search equation used in ABC is similar to the mutation scheme of DE. Then, some researchers modified DE mutation schemes to construct new search equations for ABC. In [11], two new models called ABC/best/1 and ABC/best/2 were constructed based on DE/best/1 and DE/best/2, respectively. In [17], Cui et al. designed another search equation inspired by the DE/rand/1. In this paper, a new search model based on the best-of-random mutation scheme is proposed. In [8], Lin et al. presented a novel mutation base strategy, called best-of-random (BoR), for DE. The best one among randomly selected individuals is chosen as the mutation base. For example, X_{r1}, X_{r2}, and X_{r3} are three randomly selected individuals. Then, the best one among X_{r1}, X_{r2}, and X_{r3} is chosen as the mutation base [8].

$$v_{i,j} = x_{rb,j} + F \cdot \left(x_{r1,j} - x_{r2,j}\right) \tag{4}$$

where X_{r1}, X_{r2}, and X_{rb} are three randomly selected solutions from the population, $i \neq r1 \neq r2 \neq rb$, and X_{rb} is best one among three individuals. The parameter $F \in [0, 1]$ is called scale factor. Based on Eqs. (1) and (4), a new search model is proposed as follows.

$$v_{i,j} = x_{rb,j} + \phi_{i,j} \cdot \left(x_{r1,j} - x_{r2,j}\right) \tag{5}$$

where $j \in [1, D]$ is a random dimension index, and $\phi_{i,j}$ is a random value within [−1,1]. Firstly, we randomly select three solutions X_{r1}, X_{r2}, and X_{rb} from the population. Secondly, we compare X_{rb} with X_{r1} and X_{r2}, respectively. If X_{r1} is better than X_{rb}, then swap X_{rb} with X_{r1}. Now, X_{rb} is better than X_{r1}. Similarly, we swap X_{rb} with X_{r2}, when X_{r2} is better than X_{rb}. After two comparisons, X_{rb} is the best one among X_{r1}, X_{r2}, and X_{rb}. The base vector X_{rb} acts as the local search center. The weighted vector differences determines the step size around the center. Therefore, new solutions are in the neighborhood of X_{rb}.

The literature [10] illustrated that changing only one dimension of parent solutions in ABC may result in slow convergence speed. To tackle this problem, Akay and Karaboga [10] introduced a parameter *MR* to control the frequency of updating dimensions. In this paper, this method is used as the second strategy to enhance the performance of ABC. For each food source (solution), we check all dimensions $j, j = 1, 2, \ldots, D$. If $rand_j < MR$, then update $v_{i,j}$ according to Eq. (5). Then, the new search equation is rewritten as follows.

$$v_{i,j} = \begin{cases} x_{rb,j} + \phi_{i,j} \cdot \left(x_{r1,j} - x_{r2,j}\right), & \text{if } rand_j < MR \\ v_{i,j}, & \text{otherwise} \end{cases} \tag{6}$$

where *MR* is a predefined probability parameter and $rand_j$ is a random value of the jth dimension.

The steps of the proposed HABC are given as follows.

Step 1. Randomly initialize N solutions, and compute their fitness values. Set $t = 0$.

Step 2. For each solution X_i, a new solution V_i is generated by Eq. (6). The better one between X_i with V_i is used as the new X_i. If $f(V_i) < f(X_i)$, then $trial_i = 0$; otherwise $trial_i = trial_i + 1$ (this paper only considers minimization problems).

Step 3. Calculate the probability p_i according to Eq. (2). If X_i is selected, then a new solution V_i is created by Eq. (6). The better one between X_i with V_i is used as the new X_i. If $f(V_i) < f(X_i)$, then $trial_i = 0$; otherwise $trial_i = trial_i + 1$.

Step 4. If $\max\{trail_i\} > limit$, the solution X_i is updated by Eq. (3).

Step 5: Set $t = t + 1$. If $t < MaxG$, then go to Step 2; otherwise stop the algorithm.

4 Experimental study

4.1 Test problems

In the experiments, there are twelve numerical optimization problems [23,24]. The descriptions of these test problems are displayed in Table 1, where D is the dimensional size and *Opt* is the global optimum. All these problems are to be minimized.

4.2 Parameter analysis of MR

In the standard ABC, only one dimension is updated during the search. This may result in slow convergence. In HABC, the parameter *MR* can control the number of updating dimensions. Therefore, the value of*MR* may seriously affect the search performance of HABC. To investigate the effect of *MR*, different values of *MR* are evaluated. In the following experiment, *MR* is set to 0.1, 0.2, 0.3, 0.5, and 1.0, respectively.

In HABC, the population size (N) and maximum number of generation (*MaxG*) are set to 50 and 1500, respectively. The*limit* is equal to 100. For each test problem, HABC with each *MR* runs 30 times. Results of HABC with different *MR* are given in Table 2, where “Mean” represents the mean function value. The performance of HABC is not affected on problems f_6, f_{11} and f_{12}. For f_1, f_2, and f_4, $MR = 0.3$

Table 1 Twelve test problems

Problems	Search range	D	*Opt*
$f_1(x) = \sum_{i=1}^{D} x_i^2$	$[-100, 100]$	30	0
$f_2(x) = \sum_{i=1}^{D} \lvert x_i \rvert + \prod_{i=1}^{D} x_i$	$[-10, 10]$	30	0
$f_3(x) = \sum_{i=1}^{D} \left(\sum_{j=1}^{i} x_j\right)^2$	$[-100, 100]$	30	0
$f_4(x) = \max_i \left(\lvert x_i \rvert, 1 \le i \le D\right)$	$[-100, 100]$	30	0
$f_5(x) = \sum_{i=1}^{D-1} \left[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right]$	$[-30, 30]$	30	0
$f_6(x) = \sum_{i=1}^{D} \left(\lfloor x_i + 0.5 \rfloor\right)^2$	$[-100, 100]$	30	0
$f_7(x) = \sum_{i=1}^{D} i x_i^4 + rand[0, 1)$	$[-1.28, 1.28]$	30	0
$f_8(x) = \sum_{i=1}^{D} -x_i \sin\left(\sqrt{\lvert x_i \rvert}\right)$	$[-500, 500]$	30	-12569.5
$f_9(x) = \sum_{i=1}^{D} [x_i^2 - 10\cos(2\pi x_i) + 10]$	$[-5.12, 5.12]$	30	0
$f_{10}(x) = -20 \cdot \exp\left(-0.2 \cdot \sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right)$ $- \exp\left(\frac{1}{D}\sum_{i=1}^{D} \cos(2\pi x_i)\right) + 20 + e$	$[-32, 32]$	30	0
$f_{11}(x) = \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$[-600, 600]$	30	0
$f_{12}(x) = 0.1\{\sin^2(3\pi x_1)$ $+ \sum_{i=1}^{D-1} (x_i - 1)^2[1 + \sin^2(3\pi x_{i+1})] + (x_D - 1)^2$ $[1 + \sin^2(2\pi x_D)]\} + \sum_{i=1}^{D} u(x_i, 10, 100, 4)$	$[-50, 50]$	30	0

Table 2 Results of HABC with different *MR*

Problems	$MR = 0.1$	$MR = 0.2$	$MR = 0.3$	$MR = 0.5$	$MR = 1.0$
	Mean	Mean	Mean	Mean	Mean
f_1	4.76E−42	3.25E−49	**5.64E−51**	2.53E−47	5.38E−40
f_2	4.34E−25	4.42E−30	**9.47E−32**	1.34E−29	2.95E−20
f_3	1.39E+04	1.58E+04	2.28E+04	3.09E+04	**6.74E−04**
f_4	9.74E−01	5.24E−04	**1.29E−04**	2.57E−04	3.54E−02
f_5	7.70E+01	2.52E+01	2.23E+01	2.03E+01	**1.81E+01**
f_6	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
f_7	4.84E−02	**2.67E−02**	3.01E−02	2.72E−02	3.75E−02
f_8	**−12569.5**	**−12569.5**	**−12569.5**	−12332.6	−9298.7
f_9	**0.00E+00**	**0.00E+00**	4.54E+01	1.16E+02	5.27E+01
f_{10}	1.84E−14	1.12E−14	**7.69E−15**	**7.69E−15**	**7.69E−15**
f_{11}	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
f_{12}	**3.02E−17**	**3.02E−17**	**3.02E−17**	**3.02E−17**	**3.02E−17**

Best results among different *MR* values are given in bold

is the best choice. A larger value of MR is more suitable for problems f_3 and f_5. $MR = 0.2$ performs the best on f_7. $MR = 0.1$ and $MR = 0.2$ can find the global optimum on f_9. For f_8, the values of *MR* between 0.1 and 0.3 are good settings. Results show that the performance of HABC is affected by the parameter *MR* on most test problems. A fixed *MR* may not be a good choice. According to the results, $MR = 0.3$ is a balanced setting.

To clearly identify which *MR* is the best, Figs. 1, 2, 3 present the convergence characteristics of HABC with different values of *MR*. From Figs. 1 and 2, all *MR* values can help HABC converge fast, and $MR = 0.3$ is faster than other values. $MR = 0.2$ and $MR = 0.5$ obtain the second and third places, respectively. For Fig. 3, $MR = 0.3$ shows the fast convergence, while $MR = 0.1$ is the worst.

4.3 Comparison of HABC with other ABC algorithms

To evaluate the performance of HABC, it is compared with ABC, GABC [9], and MABC [11] on twelve test problems. For ABC, GABC, MABC, and HABC, the parameters N, *limit*, and *MaxG* are set to 50, 100, and 1500, respectively. The parameter p in MABC is equal to 0.7 [11]. The parameter C is set to 1.5 [9]. The *MR* used in HABC is set to 0.3.

Computational results of HABC, ABC, GABC, and MABC are presented in Table 3, where "Mean" indicates

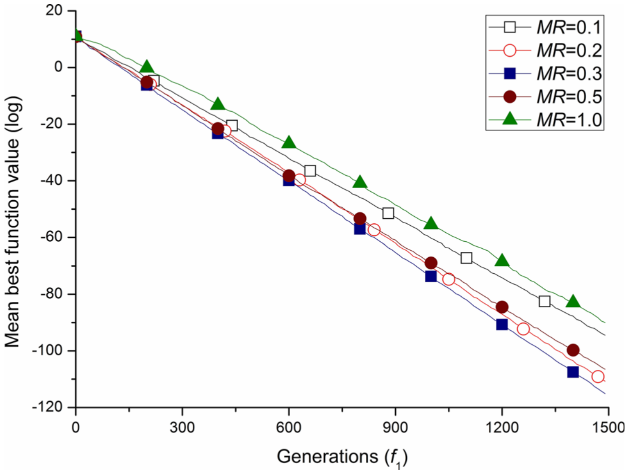

Fig. 1 The convergence characteristics of HABC with different values of *MR* on problem f_1

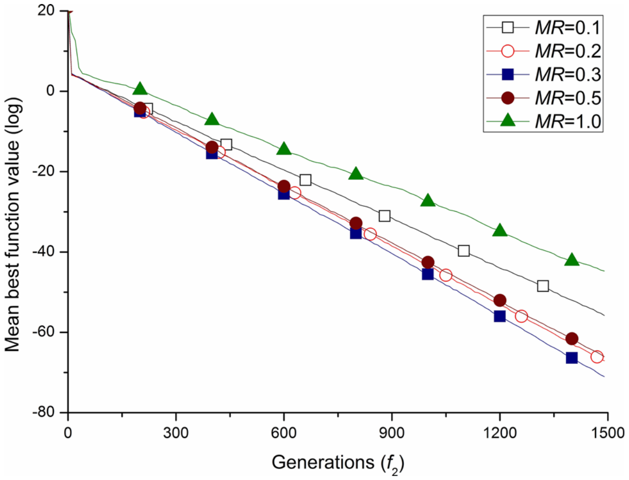

Fig. 2 The convergence characteristics of HABC with different values of *MR* on problem f_2

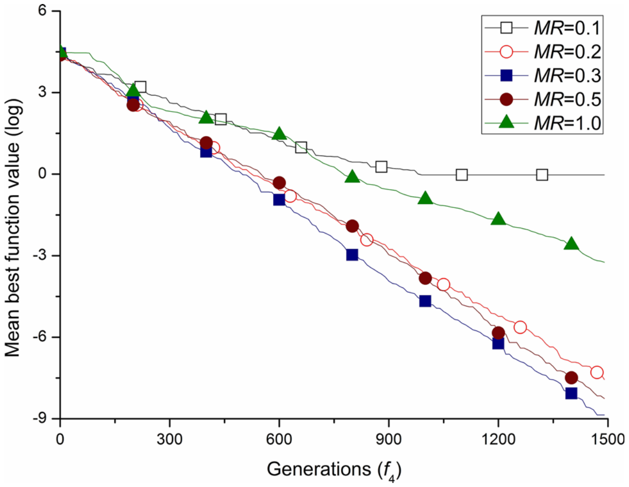

Fig. 3 The convergence characteristics of HABC with different values of *MR* on problem f_4

Table 3 Results of ABC, GABC, MABC, and HABC

Problems	ABC Mean	GABC Mean	MABC Mean	HABC Mean
f_1	9.31E−16	5.11E−16	6.34E−27	**5.64E−51**
f_2	1.25E−10	1.62E−15	6.20E−15	**9.47E−32**
f_3	8.53E+03	8.23E+03	**8.21E+03**	2.28E+04
f_4	4.45E+01	1.73E+01	6.06E+00	**1.29E−04**
f_5	6.79E−01	3.32E−01	**2.54E−01**	2.23E+01
f_6	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
f_7	1.87E−01	8.41E−02	3.81E−02	**3.01E−02**
f_8	−12545.9	**−12569.5**	**−12569.5**	**−12569.5**
f_9	5.15E−14	**0.00E+00**	**0.00E+00**	4.54E+01
f_{10}	9.23E−10	3.61E−14	1.82E−13	**7.69E−15**
f_{11}	7.56E−14	2.21E−16	**0.00E+00**	**0.00E+00**
f_{12}	7.38E−16	6.81E−16	**3.02E−17**	**3.02E−17**

Best results among four ABC variants are given in bold

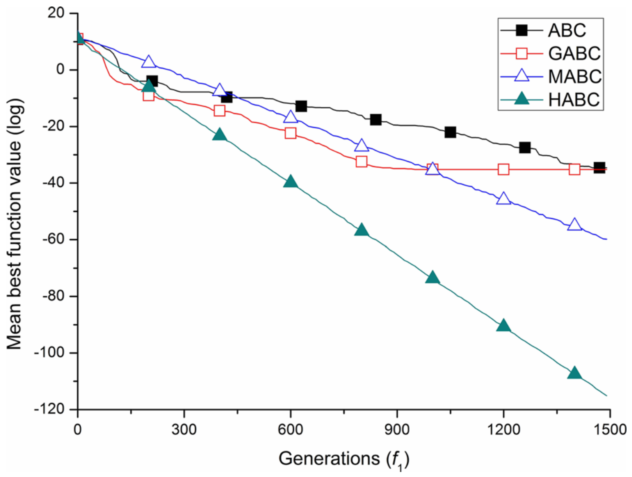

Fig. 4 The convergence curves of ABC, GABC, MABC, and HABC on problem f_1

the mean function value. From the results, all four ABCs can converge to the global optimum on f_6. HABC is superior to ABC on eight problems, while ABC outperforms HABC on three problems. GABC, MABC, and HABC achieve the same result on f_8. HABC obtains better solutions than GABC on seven problems, but HABC is worse than GABC on three problems. MABC is better than HABC on three problems, while HABC achieves better solutions on five problems. For problems f_{11} and f_{12}, MABC and HABC obtain the same results.

Figures 4, 5, 6, 7, 8 and 9 present the convergence curves of ABC, GABC, MABC, and HABC on six selected problems. HABC converges much faster than ABC, GABC, and MABC. For f_{11} and f_{12}, GABC and ABC are faster than HABC at the initial search stage. HABC shows faster convergence rate at the middle and last search stages. The

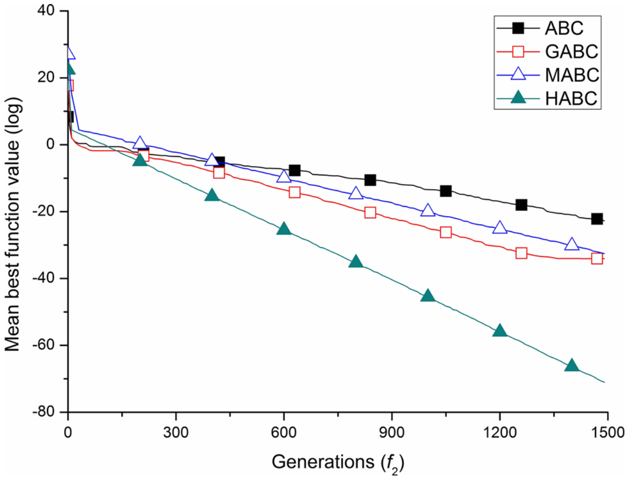

Fig. 5 The convergence curves of ABC, GABC, MABC, and HABC on problem f_2

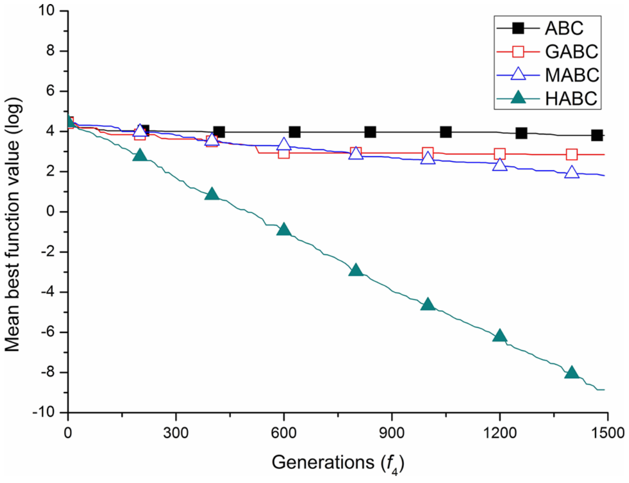

Fig. 6 The convergence curves of ABC, GABC, MABC, and HABC on problem f_4

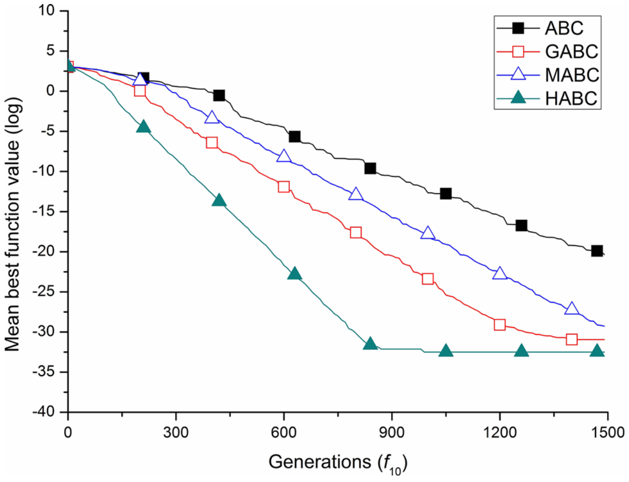

Fig. 7 The convergence curves of ABC, GABC, MABC, and HABC on problem f_{10}

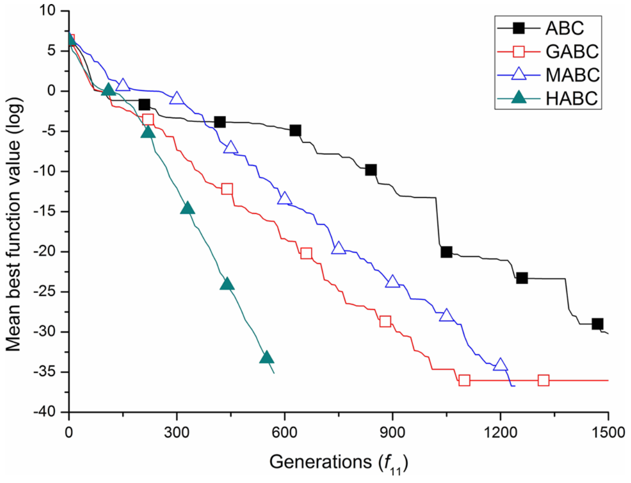

Fig. 8 The convergence curves of ABC, GABC, MABC, and HABC on problem f_{11}

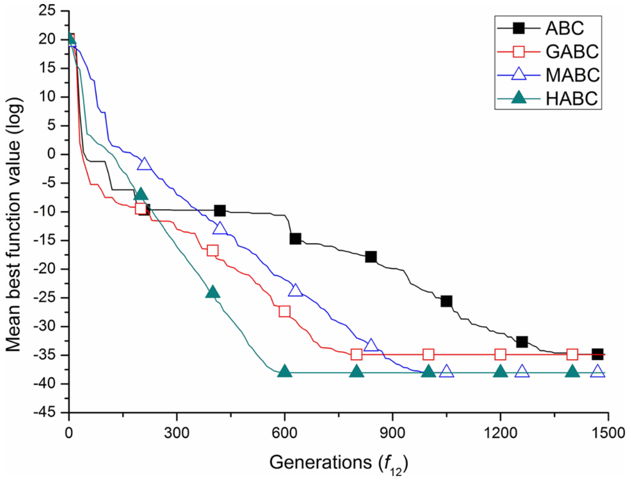

Fig. 9 The convergence curves of ABC, GABC, MABC, and HABC on problem f_{12}

convergence processes confirm that our designed strategies can significantly improve the convergence rate of ABC.

5 Conclusions

In order to address the issues of low accuracy of solutions and slow convergence, a hybrid ABC (HABC) is presented. In HABC, a modified search model is designed based on the best-of-random DE mutation scheme. Moreover, a new parameter *MR* is introduced to control the number of updating dimensions. To verify the performance of HABC, twelve test problems are used in the experiments. HABC is compared with ABC, GABC, and MABC. Results show that HABC is superior to other three ABC algorithms on the majority of test problems.

The parameter *MR* can seriously affect the performance of HABC on most test problems. It demonstrates that a fixed vale of MR may not be a good setting for all test problems. In the future work, we will try some dynamic strategies to adjust *MR*. It is hopeful that the dynamic *MR* can help HABC to solve different kinds of problems.

Acknowledgements This work is supported by the project of the First-Class University and the First-Class Discipline(10301-017004011501), and the National Natural Science Foundation of China.

References

1. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of IEEE International Conference on Neural Networks, pp. 1942–1948 (1995)
2. Dorigo, M., Maniezzo, V., Colorni, A.: Ant system: optimization by a colony of cooperating agents. IEEE Trans. Syst. Man Cybern. B (Cybern.) **26**(1), 29–41 (1996)
3. Yang, X.S.: Firefly algorithm, stochastic test functions and design optimization. Int. J. Bio-Inspired Comput. **2**(2), 78–84 (2010)
4. Karaboga, D.: An idea based on honey bee swarm for numerical optimization. Technical Report-TR06, Erciyes University, engineering Faculty, Computer Engineering Department (2005)
5. Yang, X.S., Deb, S.: Engineering optimisation by cuckoo search. Int. J. Math. Model. Numer. Optim. **1**(4), 330–343 (2010)
6. Yang, X.S.: A new metaheuristic bat-inspired algorithm. In: Nature Inspired Cooperative Strategies for Optimization (NICSO 2010), pp. 65–74 (2010)
7. Karaboga, D., Akay, B.: A comparative study of artificial bee colony algorithm. Appl. Math. Comput. **214**, 108–132 (2009)
8. Lin, C., Qing, A., Feng, Q.: A new differential mutation base generator for differential evolution. J. Glob. Optim. **49**(1), 69–90 (2011)
9. Zhu, G., Kwong, S.: Gbest-guided artificial bee colony algorithm for numerical function optimization. Appl. Math. Comput. **217**, 3166–3173 (2010)
10. Akay, B., Karaboga, D.: A modified artificial bee colony algorithm for real-parameter optimization. Inf. Sci. **192**, 120–142 (2012)
11. Gao, W.F., Liu, S.Y., Huang, L.L.: A global best artificial bee colony algorithm for global optimization. J. Comput. Appl. Math. **236**(11), 2741–2753 (2012)
12. Wang, H., Wu, Z.J., Rahnamayan, S., Sun, H., Liu, Y., Pan, J.S.: Multi-strategy ensemble artificial bee colony algorithm. Inf. Sci. **279**, 587–603 (2014)
13. Kiran, M.S., Hakli, H., Gunduz, M., Uguz, H.: Artificial bee colony algorithm with variable search strategy for continuous optimization. Inf. Sci. **300**, 140–157 (2015)
14. Gao, W.F., Huang, L.L., Liu, S.Y., Chan, F.T.S., Dai, C.: Artificial bee colony algorithm with multiple search strategies. Appl. Math. Comput. **271**, 269–287 (2015)
15. Zhou, X.Y., Wu, Z.J., Wang, H., Rahnamayan, S.: Gaussian bare-bones artificial bee colony algorithm. Soft. Comput. **20**(3), 907–924 (2016)
16. Zhou, X.Y., Wang, H., Wang, M.W., Wan, J.Y.: Enhancing the modified artificial bee colony algorithm with neighborhood search. Soft. Comput. **21**(10), 2733–2743 (2017)
17. Cui, L.Z., Li, G.H., Wang, X.Z., Lin, Q.Z., Chen, J.Y., Lu, N., Lu, J.: A ranking-based adaptive artificial bee colony algorithm for global numerical optimization. Inf. Sci. **417**, 169–185 (2017)
18. Liang, Z.P., Hu, K.F., Zhu, Q.X., Zhu, Z.X.: An enhanced artificial bee colony algorithm with adaptive differential operators. Appl. Soft Comput. **58**, 480–494 (2017)
19. Wang, H., Wu, Z., Zhou, X., Rahnamayan, S.: Accelerating artificial bee colony algorithm by using an external archive. In: IEEE Congress on Evolutionary Computation (CEC 2013), pp. 517–521 (2013)
20. Li, X.N., Yang, G.F.: Artificial bee colony algorithm with memory. Appl. Soft Comput. **41**, 362–372 (2016)
21. Karaboga, D., Gorkemli, B.: A quick artificial bee colony (qABC) algorithm and its performance on optimization problems. Appl. Soft Comput. **23**, 227–238 (2014)
22. Sharma, T.K., Pant, M.: Shuffled artificial bee colony algorithm. Soft. Comput. **21**(20), 6085–6104 (2017)
23. Wang, H., Sun, H., Li, C., Rahnamayan, S., Pan, J.S.: Diversity enhanced particle swarm optimization with neighborhood search. Inf. Sci. **223**, 119–135 (2013)
24. Wang, H., Rahnamayan, S., Sun, H., Omran, M.G.H.: Gaussian bare-bones differential evolution. IEEE Trans. Cybern. **43**(2), 634–647 (2013)

Xiuqin Pan received her PhD in control science from Beijing Institute of Technology in 2002. Now, she is a Professor at the School of Information Engineering, Minzu University of China. Her current research interest focuses on parallel algorithm and intelligent system.

Yong Lu received his master degree in 2005. Now, he is an engineer at the School of Information Engineering, Minzu University of China. His research interest includes parallel algorithm and intelligent system.

Na Sun received her PhD in signal and information processing from Beijing University of Posts and Telecommunications in 2010. Now, she is a lecturer at the School of Information Engineering, Minzu University of China. Her current research interest focuses on parallel algorithm and intelligent system.

Sumin Li received her Master degree in 2004. Now, she is an instructor at the School of Information Engineering, MinZu University of China. Her research interest includes parallel algorithm and intelligent system.