Taylor & Francis
Taylor & Francis Group

Applied Artificial Intelligence

An International Journal

ISSN: (Print) (Online) Journal homepage: https://www.tandfonline.com/loi/uaai20

Improved Artificial Bee Colony Algorithm with Adaptive Parameter for Numerical Optimization

Ming Zhao, Xiaoyu Song & Shuangyun Xing

To cite this article: Ming Zhao, Xiaoyu Song & Shuangyun Xing (2022) Improved Artificial Bee Colony Algorithm with Adaptive Parameter for Numerical Optimization, Applied Artificial Intelligence, 36:1, 2008147, DOI: 10.1080/08839514.2021.2008147

To link to this article: https://doi.org/10.1080/08839514.2021.2008147

APPLIED ARTIFICIAL INTELLIGENCE
2022, VOL. 36, NO. 1, e2008147 (541 pages)
https://doi.org/10.1080/08839514.2021.2008147

Taylor & Francis
Taylor & Francis Group

OPEN ACCESS Check for updates

Improved Artificial Bee Colony Algorithm with Adaptive Parameter for Numerical Optimization

Ming Zhao[a], Xiaoyu Song[a], and Shuangyun Xing[b]

[a]Information & Control Engineering Faculty, Shenyang Jianzhu University, Shenyang, China; [b]School of Science, Shenyang Jianzhu University, Shenyang, China

ABSTRACT

The problem that ABC (Artificial Bee Colony) algorithm is good at exploration but poor at exploitation for the numerical optimization is investigated in this paper. PA-ABC (Parameter Adaptive ABC) algorithm is proposed, which adopts different search equations with different search abilities for the employed bee and the onlooker bee. Firstly, the best-so-far solution is introduced into each search equation to enhance exploitation; secondly, the employed bee uses two random solutions to search, so as to keep high ability of exploration; thirdly, the onlooker bee searches around a random solution to keep population diversity; most importantly, adaptive parameter computed by fitness function is introduced in the search equation of the onlooker bee, which makes the search step adjust according to the search process. So the search equation of the employed bee has balanced abilities of exploration and exploitation, while the search equation of the onlooker bee can make the search focus transfer from exploration to exploitation adaptively. The experiment results on benchmark functions show that the search performance of PA-ABC is higher than or at least comparable to basic ABC and typical improved ABCs. In addition, compared to the performance of the state-of-the-art ABC variants under their original parameter configuration, PA-ABC is verified to have similar performance to them.

ARTICLE HISTORY
Received 11 September 2021
Revised 10 November 2021
Accepted 15 November 2021

Introduction

In the fields of the engineering design, economics, statistical physics, information theory, and computing science, there are many kinds of the optimization problems. However, these problems are usually NP hard problems of which the optimal solutions cannot be obtained in a reasonable time. In recent years, enlightened by biology, some researchers proposed a large number of artificial life computation optimization to solve these optimization problems, such as genetic algorithm (GA) (Tang, Man, Kwong, He 1996), differential evolution algorithm (DE) (Das and Suganthan 2010; Storn and Price 1997), particle swarm optimization algorithm (PSO) (Kennedy and Eberhart 1995), and ant colony optimization algorithm (ACO) (Dorigo and Stutzle 2004), etc. Since

CONTACT Ming Zhao 1094301006@qq.com Information & Control Engineering Faculty, Shenyang Jianzhu University, Shenyang, China

these algorithms can obtain approximate optimal solutions quickly (Simon 2008), they have been widely studied and applied in optimization problems of engineering and science areas.

In 2005, inspired by the foraging behavior of the bee swarm, Karaboga proposed the artificial bee colony algorithm, which had become one of the latest and hottest swarm intelligence algorithms (Karaboga 2005). Compared with other optimization algorithms, such as GA, PSO, DE and ACO, ABC can search better solution more effectively with less control parameters (Karaboga and Akay 2009; Karaboga and Basturk 2008). Thus, ABC is widely used to solve complex practical optimization problems (Karaboga, Gorkemli, Ozturk, et al. 2014), such as job-shop scheduling (Li, Pan, Tasgetiren 2014), filter design (Vural, Yildirim, Kadioglu, et al. 2012), image segmentation (Bhandari, Kumar, and Singh 2015), biological medicine (Li, Li, Gong 2014), transit network design (Rajasekhar, Lynn, Das, Suganthan 2017), vehicle routing (Shi, Pun, Hu, Gao 2016), cooper strip production (Yang, Chen, Yu, Gu, Li, Zhang, Zhang 2017), etc.

Nevertheless, there also exists some problems in ABC that need to improve, such as lower convergence speed (Kong, Chang, Dai, Wang, Sun 2018) and insufficient exploitation (2018). This is mainly because that although ABC performs best on exploration, it cannot take full advantage of searching history, leading to poor exploitation (Karaboga and Basturk 2008). The weakness has limited the application of ABC. So to speed up convergence, many researches took advantages of this kind of priori knowledge to modify basic search equation. These ABCs can be classified into two classes: ABCs with added guiding information and ABCs with adaptive adjusting mechanism.

Many improved ABCs use the best-so-far solution to guide the searching direction. In (Gao, Liu, and Huang 2012), Gao et al. proposed ABC/best in which bees only searched around the best-so-far solution. Furthermore, Gao et al. used probability to choose between searching based on the best-so-far solution and searching randomly in (Cui, Li, Wang, Lin, Chen, Lu, Lu, 2017; Gao and Liu 2012). Luo et al. (Luo, Wang, and Xiao 2013) proposed COABC in which onlooker bees made roulette selection based on cumulative nectar amount. Zhu et al. (Zhu and Kwong 2010) proposed Gbest-guided ABC, which used the information of the best-so-far solution in the search equation. In (Banharnsakun, Achalakul, and Sirinaovakul 2011), Banharnsakun et al. proposed the best-so-far ABC, which shared the best-so-far solution in the whole population. Babayigit et al. (Babayigit and Ozdemir 2012) proposed ABCclobest in which the onlooker bee generated the candidate solutions based on its current position and the best-so-far solution. Xiang et al. (Xiang and An 2012) proposed ERABC, which added the best-so-far solution into the search process. In (Li, Niu, and Xiao 2012), Li et al. proposed I-ABC, which used the best-so-far solution, inertia weight, and acceleration coefficient to correct the searching process. Lin et al. introduced the information of the best-so-far solution in the neighborhood and the elite solutions, respectively, in the search equation of ABCLGII (Lin et al. 2018).

In some papers, the search for the candidate solutions is guided by search experience and related individual information. For example, Imanian et al. (Imanian, Shiri, and Moradi 2014) proposed VABC in which onlooker bees used PSO search strategy to search the candidate solutions. In (Tsai, Pan, and Liao 2009), Tsai et al. introduced universal gravitation and proposed IABC, in which onlooker bees were attracted to the locations of the employed bees and evaluated their fitness values. Liu et al. (Liu et al. 2012) proposed Mutual ABC, which made bees to learn between each other. Li et al. (Li, Li, Gong 2014proposed IF-ABC, which made full use of the internal state information of each iteration. In (Gao, Liu, and Huang 2013), Gao et al. proposed CABC with the altered search equation and OCABC by introducing the orthogonal experiment design to get more valuable information from search experience. In our previous work (Song, Zhao, Yan, Xing 2019), the individual selection range of the search equation is limited to the elite solutions. In (Brajević 2021), a modified search operator is proposed, which exploits the useful information of the best-so-far solution in the onlooker bee phase to improve exploitation tendency.

Different search equations have different search characteristics, so some researchers have applied the mixture of multiple strategies to the search process of bees. In (Zhou, Yao, Chan, et al. 2019), Zhou et al. divided the whole population into three subgroups and designed evolutionary operators with different search biases for each subgroup to play different roles. Song et al. selected and designed a variety of strategies with different search capabilities, which were combined at different proportions in the search stages of the employed bees and onlooker bees (Song, Zhao, and Xing 2019). In (Brajević, Stanimirović, Li, Cao 2020), Brajević et al mixed fireflies and artificial colonies and used a new multi-strategy ABC to conduct local search.

In addition, a slide of papers introduced other dynamic adjustment or local mechanism. Li et al. proposed DABC in which employed bees used tabu local search (Li, Pan, Tasgetiren 2014). Alam et al. (Gao and Liu 2011) proposed ABC-SAM, which was adaptively mutated and used step size adjusted dynamically. Rajasekhar et al. (Anguluri, Ajith, and Millie 2011) proposed L-ABC with mutation ability that generated candidate solution by Levy distribution. In (Kang, Li, and Ma 2011), Kang et al. proposed RABC that used ABC to explore globally and Rosenbrock rotation of the direction to exploit locally. Zhang et al. (Zhang, Zheng, and Zhou 2015) proposed GEM ABC, which added GEM into the search process. Alatas et al. (Alatas 2010) proposed Chaotic ABC based on the adaptive parameter with the chaotic map. In (Akay and Karaboga 2012), Akay et al. introduced the control parameter setting into ABC, to dynamically adjust the dimension to modify and the search step. In our previous work (Song, Yan, and Zhao 2017), we introduced the objective function value in the search equation in the form of trigonometric mutation.

Exploration and exploitation are important components of the any metaheuristic algorithm, and in order to be successful, a search algorithm needs to establish a good ratio between these two processes (Brajević and Stanimirović 2018; Das, Biswas, and Kundu 2013; Das, Biswas, Panigrahi, Kundu, Basu 2014). Exploration is the ability to visit various regions in the problem landscape, aiming to locate a good optimum, while exploitation is the ability to concentrate the search around a promising candidate solution, trying to find the optimum more precisely (Lin et al. 2018). So based on previous study, this paper proposes two novel search equations for the employed bee and the onlooker bee to form PA-ABC. The design of search equation for the employed bee focuses on enhancing exploitation while improving exploration, and the balance between them. The design of the search equation for the onlooker bee focuses on improving exploitation while keeping high population diversity, and the adaptive adjusting on the search step size.

In the following sections, PA-ABC is presented in Section 2; Section 3 provides parameter setting, some experiment results and analysis of PA-ABC compared with the related algorithms on sets of benchmark functions; Section 4 concludes all the work.

Artificial Bee Colony Algorithm with Adaptive Parameter (PA-ABC)

In this section, firstly, we propose the two novel search equations called TRC-ABC (ABC that searches around the center of two random solutions) and RA-ABC (ABC that searches around one random solution with adaptive parameter) individually and analyze their search scopes according to their items in detail. Furthermore, we propose PA-ABC, which uses these two equations to search for optimal solution.

TRC-ABC

The search equation of TRC-ABC is shown in Equation 1:

$$v_{i,j} = (x_{r1,j} + x_{r2,j})/2 + \phi_{i,j}(x_{r1,j} - x_{r2,j}) + \varphi_{i,j}(y_j - x_{r1,j}) \quad (1)$$

where *r1* and *r2* are distinct numbers different with *i* on {1, . . ., *FN}* (*FN* is the number of food sources, which equal to *NP*/2, and *NP* is the population size) selected randomly. $\phi_{i,j}$ is a random number on [−0.25, 0.25]. $\varphi_{i,j}$ is a

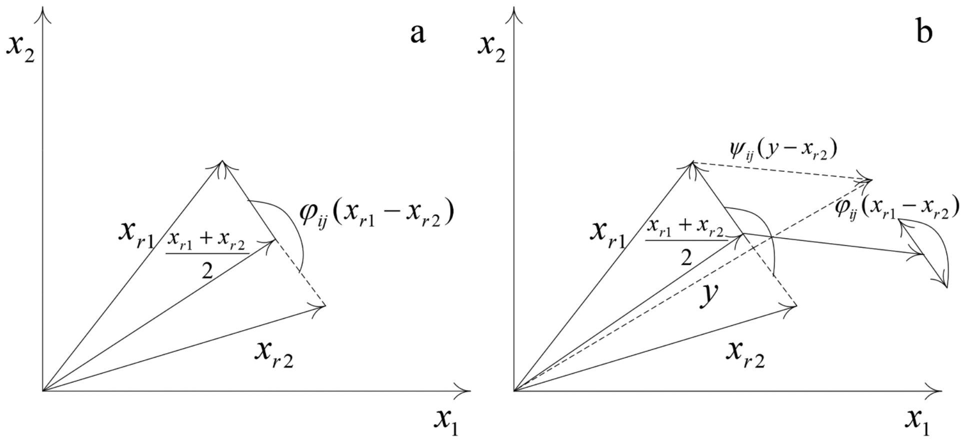

Figure 1. Search scope of TRC-ABC. (a) Search scope affected by the first two items of TRC-ABC. (b) Search scope of TRC-ABC when $y \neq x_{r1}$.

random number on [0, 1]. y is the best-so-far solution. In Equation 1, since the search equation adopts trinomials, to avoid excessive search step length, the value range of the random number in the second item is set as [−0.25, 0.25], in combination with the value range design of the third random number.

The first item at the right of Equation 1 illustrates that the candidate solution is generated around the center of two solutions selected randomly. Equation 1 also applies slight turbulence with the second item based on the difference vector of the two solutions. Both the first two items ensure the strong ability of exploration. Based on the range of $\varphi_{i,j}$, the search scope affected by these two items is shown in Figure 1a.

Furthermore, the third item is added with the best-so-far solution as guiding information to increase the exploitation probability around it. If x_{r1} is just the best-so-far solution ($y = x_{r1}$), then $\varphi_{i,j}(y_j - x_{r1,j}) = 0$, meaning that the search process only affected by the first two items and the search scope is equal to Figure 1a. Now the search process is near to the global optimal solution, so the solution better than the best-so-far solution is inclined to find. Otherwise, exploitation around the best-so-far solution is enhanced, because of the guiding information of it in the third item. The search scope of this case is shown in Figure 1b. The arc lines in Figure 1 represent the theoretical search scope of the search equation.

RA-ABC

Referring to the improved ABCs with high performance, such as CABC, ABC/best and GABC, RA-ABC adopts the search equation shown in Equation 2.

$$v_{i,j} = x_{r1,j} + \phi_{i,j}(x_{r1,j} - x_{r2,j})ap + \varphi_{i,j}(y_j - x_{r1,j}) \quad (2)$$

$$ap = (f(x_{r1,j}) - f(x_{r2,j}))/(f(x_{r1,j}) + f(x_{r2,j})) \quad (3)$$

where *r1* and *r2* are distinct integers different with *i* on {1,2, …,*FN*}. $\phi_{i,j}$ is a random value on [−0.75,0.75]. *ap* is the adaptive parameter which is computed by the fitness function value of x_{r1} and x_{r2} shown in Equation 3. *y* is the best-so-far solution. $\varphi_{i,j}$ is a random number on [0,0.5]. *f(x)* is the fitness function (minimum optimization) value of *x*. In Equation 2, since the search equation adopts trinomials, to avoid excessive search step length, the value range of the random number in the second item is set as [−0.75, 0.75], in combination with the value range design of the third random number.

The first item at the right of Equation 2 illustrates that the candidate solution is generated around x_{r1}, which selected randomly from the population to keep high diversity. The second item without *ap* can bring more information making equation getting the higher search performance (Gao, Liu, and Huang 2013). Because the first two items provide the equation with enough exploration, we shrink the range of $\phi_{i,j}$ from [−1, 1] to [−0.75, 0.75], so as to enhance exploitation and furthermore balance exploration and exploitation. The search scope affected by these two items is shown in Figure 2a. Then consider the effect of the adaptive parameter *ap* in the second item to adjust the search step size according to the search process. At early stage of the search process, the difference between x_{r1} and x_{r2} is large, so the computing result of *ap* based on their fitness function values is large too, which means big step size, indicating that the search equation focuses on exploration; at late stage of the search process, with converging to the optimal solution gradually, the difference between x_{r1} and x_{r2} becomes small, so the computing result of *ap* based on their fitness function values is small which means that the step size is small as well, indicating

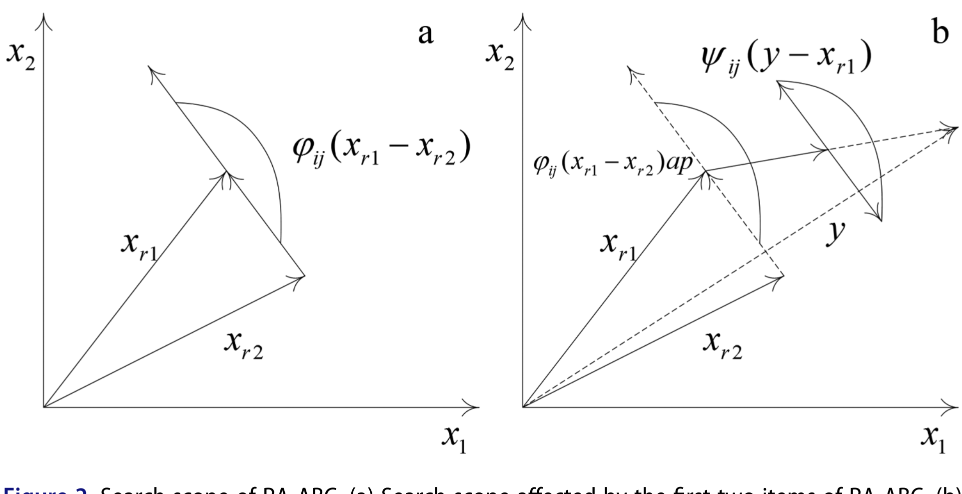

Figure 2. Search scope of RA-ABC. (a) Search scope affected by the first two items of RA-ABC. (b) Search scope affected by all the items of RA-ABC.

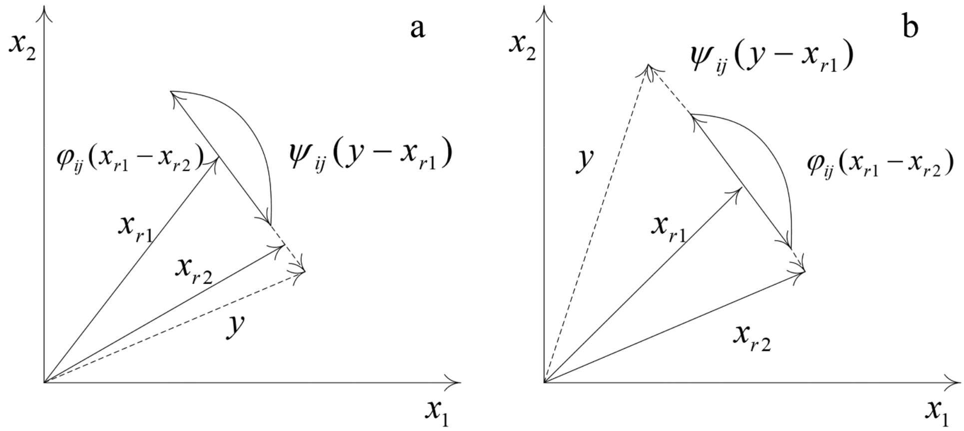

Figure 3. Search scope of the last item of RA-ABC. (a) In same direction. (b) In opposite direction.

that the search equation works toward exploitation. In general, parameters $\phi_{i,j}$ and $\varphi_{i,j}$ are set properly to balance exploration and exploitation, and the adding of parameter *ap* can make the search process adaptively adjusted from exploration to exploitation according to the search process.

The third item at the right of Equation 2 uses the best-so-far solution to guide the search direction. Because $\varphi_{i,j}$ is a positive number on [0, 0.5], making the search process toward the direction of the best-so-far solution to improve exploitation, as shown in Figure 2b.

Consider the value ranges of the last two items at the right of the equation. The step of the second step is 1.5, and the step of the third is 0.5, that is the step size of the third is 1/3 of the second. Thus, if they are in opposite directions, the search process will work toward y with x_{r1} as the center, as shown in Figure 3a.When the two items are in the same direction, the search process will work toward y with larger step. Therefore, the search process according to the equation can realize adaptive adjusting on search step, as shown in Figure 3b.

PA-ABC

In PA-ABC, TRC-ABC is used as search equation of the employed bee to realize the balance between exploration and exploitation while keeping strong ability of exploration, and RA-ABC is embedded in the search of the onlooker bee phase to improve exploitation while keeping population diversity. Moreover, due to the enough exploitation of PA-ABC, roulette wheel selection is canceled.

Table 1. Basic benchmark functions.

No of Function	Name	Search range	Optimum
f1	*Elliptic*	$[-100,100]^n$	0
f2	*Exponential*	$[0,1.28]^n$	0
f3	*Schwefel 2.21*	$[-100,100]^n$	0
f4	*Schwefel 2.22*	$[-10,10]^n$	0
f5	*Sphere*	$[-100,100]^n$	0
f6	*SumPower*	$[-1,1]^n$	0
f7	*SumSquare*	$[-10,10]^n$	0
f8	*Step*	$[-100,100]^n$	0
f9	*Quartic*	$[-1.28,1.28]^n$	0
f10	*Rosenbrock*	$[-5,10]^n$	0
f11	*Rastrigin*	$[-5.12,5.12]^n$	0
f12	*Ackley*	$[-32,32]^n$	0
f13	*Alpine*	$[-10,10]^n$	0
f14	*Griewank*	$[-600,600]^n$	0
f15	*Levy*	$[-10,10]^n$	0
f16	*NCRastrigin*	$[-5.12,5.12]^n$	0
f17	*Penalized 1*	$[-50,53]^n$	0
f18	*Penalized 2*	$[-50,53]^n$	0
f19	*Schwefel 2.26*	$[-500,500]^n$	0
f20	*Weierstrass*	$[-0.5,0.5]^n$	0
f21	*Himmelblau*	$[-5,5]^n$	−78.33236
f22	*Michalewicz*	$[0,\pi]^n$	−99.2784

Table 2. Shifted and rotated benchmark function.

No of Function	Name	Search range	Optimum
f23	*Shifte Schwefel's Problem 1.2*	$[-100,100]^n$	0
f24	*Shifted Rastrigin's function*	$[-5,5]^n$	0
f25	*Shifted Rotated Elliptic's Function*	$[-100,100]^n$	0
f26	*Shifted Rotated Rastrigin's function*	$[-5,5]^n$	0
f27	*Shifted Schwefel's Problem 1.2 with noise*	$[-100,100]^n$	0
f28	*Shifted Sphere's Function*	$[-100,100]^n$	0

Experiment and Analysis

Benchmark Functions and Parameters Setting

To test the algorithm on typical optimization functions of variable kinds and to find the best suitable problem the algorithm can solve effectively, we choose 28 benchmark functions to form test set shown in Tables 1 and 2, and provide the optimum and the search range of each function.

There are 22 basic scalable functions in Table 1: *f1*, *f3*-*f8* are continuous unimodal; *f2* is non-continuous unimodal; *f9* is noisy quartic; *f10* is Rosenbrock function, which is unimodal with 2 or 3 dimensions but multi-modal with higher dimensions; *f11*-*f22* are multimodal, and the number of optimum will increase with dimensions exponentially; *f19* is bounded. All functions are evaluated by the global optimal value searched.

There are six functions in Table 2: *f23*, *f24*, *f27*, *f28* are shifted functions; *f25* and *f26* are rotated functions (Zhang, Zheng, and Zhou 2015). These functions are mainly intended to avoid the situation that some algorithms copy one parameter to another to generate a neighbor solution, so they are more difficult to optimize.

Table 3. Parameter setting for compared ABCs.

Parameter Setting	*NP* for all ABCs	*limit* for all ABCs	*Max_FES* for all ABCs	*C* of GABC
D = 30 for *f1-f20* & *D* = 100 for *f21-f22*	100	200	100000	1.5
D = 60 for *f1-f20* & *D* = 200 for *f21-f22*			500000	

Because the dimension number of the problem is a vital factor affecting the performance of the algorithm, so we test *f1-f20* with *D* =30 and *D* =60, *f21-f22* with *D* =100 and *D* =200. Under each dimension setting, we run each algorithm on every function independently for 30 times, and compute mean value and standard deviation of 30 runs to analyze.

Moreover, to compare the algorithms fairly, we set the same value to each common parameter shown in Table 3.

In addition, we also tested the proposed algorithm PA-ABC based on CEC2021 (Ali, Anas, Ali, Prachi, Abhishek, Suganthan, 2021), and it found the optimums for solving the 10 basic functions in 10 dimensions with the general setting of acceptable solution 1E-8. For detailed test results, please refer to the supplementary experimental material.

Experiment Result and Analysis on Basic Benchmark Function

Table 4 lists the comparison results among PA-ABC, basic ABC and several typical improved ABCs including I-ABC, GABC, ABC/best and CABC running on 22 basic benchmark functions when *D* =30 for *f1-f20* and *D* =100 for *f21-f22*.

As seen from Table 4, compared with other ABCs, PA-ABC performs best on 17 of 22 basic benchmark functions. For *f3*, although ABC/best performs better slightly than PA-ABC, the results of them are in same magnitude, which means they have similar performance with each other; for *f7* only the performance of ABC/best is better than PA-ABC; for *f22*, although CABC performs better slightly than PA-ABC, the difference between their results is very small. Moreover, for *f15, f17* and *f18* of which the global optimal solutions are not equal to 0 due to influence of the precision of π, PA-ABC finds their global optimal solutions. Particularly for *f15*, the standard deviation of 30 runs of PA-ABC is 0, indicating that the global optimal solution is found by every running.

To further verify the effectiveness of PA-ABC, based on the experiment results in Table 4, SPSS is used to conduct non-parametric test including Wilcoxon test and Friedman test, and the test results are shown in the last two rows of Table 4. It can be seen from the results of Wilcoxon test about significant difference compared with PA-ABC that the *p*-values of ABC, I-ABC, GABC are all less than 0.05, indicating that there are significant difference between them and PA-ABC, and PA-ABC is far superior to these three

Table 4. Result comparisons of ABCs on 30-dimension functions *f1-f20* and 100-dimension functions *f21* and *f22*.

fun	ABC	I-ABC	GABC	ABC/best	CABC	PA-ABC
*f*1	7.15E-19 (8.73E-19)	5.39E-25 (6.79E-25)	8.51E-27 (9.12E-27)	2.03E-35 (2.24E-35)	2.99E-39 (3.87E-39)	**8.87E-44 (6.36E-44)**
*f*2	8.74E-12 (2.65E-13)	2.48E-08 (8.97E-09)	3.87E-16 (2.30E-16)	**2.77E-21 (2.23E-21)**	6.04E-20 (2.57E-20)	3.32E-11 (3.76E-11)
*f*3	2.06E+01 (2.46E+01)	2.25E+01 (3.48E+01)	9.17E+00 (3.78E+00)	**4.32E+00 (9.57E-01)**	8.12E+00 (3.69E+00)	7.29E+00 (1.56E+00)
*f*4	2.03E-06 (3.18–06)	3.89E-07 (1.58E-08)	2.42E-11 (3.41E-11)	6.93E-17 (5.78E-17)	4.87E-17 (1.23E-17)	**3.98E-20 (1.33E-20)**
*f*5	1.21E-11 (2.45E-11)	2.42E-13 (3.18E-13)	3.04E-20 (2.86E-20)	3.40E-30 (6.12E-30)	3.01E-31 (4.82E-31)	**1.57E-36 (1.13E-36)**
*f*6	2.36E-09 (2.03E-09)	8.75E-11 (8.14E-11)	3.78E-18 (2.81E-18)	3.83E-28 (7.38E-28)	5.87E-29 (6.04E-29)	**6.51E-35 (5.57E-35)**
*f*7	1.21E-24 (2.14E-24)	5.34E-27 (1.17E-26)	3.18E-37 (6.10E-37)	**5.98E-59 (4.42E-58)**	3.76E-40 (4.29E-40)	1.95E-51 (4.91E-51)
*f*8	**0(0)**	**0(0)**	**0(0)**	**0(0)**	**0(0)**	**0(0)**
*f*9	2.99E-01 (1.68E-01)	3.71E-01 (2.27E-01)	2.30E-01 (6.90E-02)	7.84E-02 (4.10E-02)	1.56E-01 (6.47E-02)	**1.01E-02 (4.80E-03)**
*f*10	3.89E-01 (2.83E-01)	2.01E+00 (3.41E+00)	4.00E+00 (5.81E+00)	1.96E+01 (3.20E+01)	4.01E-01 (1.84E-01)	**2.63E-01 (1.76E-01)**
*f*11	5.77E-02 (3.42E-01)	6.63E-05 (3.50E-04)	5.41E-13 (3.80E-12)	**0(0)**	**0(0)**	**0(0)**
*f*12	6.01E-06 (4.24E-06)	3.84E-05 (2.25E-04)	2.37E-10 (3.82E-11)	4.10E-14 (4.33E-15)	3.35E-14 (2.14E-15)	**2.88E-14 (2.89E-15)**
*f*13	3.10E-04 (3.14E-04)	5.02E-04 (3.78E-04)	2.10E-05 (3.11E-05)	2.47E-15 (2.17E-15)	2.71E-17 (2.69E-17)	**4.33E-20 (1.11E-19)**
*f*14	1.01E-09 (2.83E-09)	5.39E-10 (4.39E-10)	9.99E-16 (1.27E-15)	**0(0)**	**0(0)**	**0(0)**
*f*15	4.15E-09 (5.97E-09)	6.57E-11 (8.45E-11)	4.31E-17 (8.82E-17)	4.68E-30 (1.56E-29)	1.35E-31 (2.32E-33)	**1.35E-31 (0.00E+00)**
*f*16	6.01E-02 (1.94E-01)	9.22E-06 (2.75E-05)	4.87E-14 (2.60E-13)	**0(0)**	**0(0)**	**0(0)**
*f*17	7.76E-12 (5.12E-12)	8.32E-14 (5.63E-14)	1.71E-22 (1.25E-22)	2.34E-32 (6.14E-33)	1.70E-32 (1.22E-33)	**1.57E-32 (2.74E-48)**
*f*18	4.74E-10 (4.60E-10)	4.51E-12 (3.82E-12)	7.28E-21 (4.92E-21)	4.16E-31 (1.12E-30)	1.44E-31 (1.51E-31)	**1.35E-32 (5.47E-48)**
*f*19	8.84E-10 (2.30E-09)	2.62E-09 (3.14E-09)	3.17E-12 (8.83E-13)	**2.01E-12 (7.71E-13)**	3.91E-06 (3.10E-05)	1.39E-05 (7.49E-05)
*f*20	6.24E-04 (8.87E-05)	4.70E-04 (8.82E-05)	2.20E-09 (3.37E-09)	**0(0)**	**0(0)**	**0(0)**
*f*21	−7.82E+01 (5.19E-02)	−7.82E+01 (6.17E-02)	−7.83E+01 (3.51E-03)	−7.83E+01 (3.90E-07)	−7.83E+01 (5.11E-09)	**7.83E+01 (5.05E-10)**

(*Continued*)

Table 4. (Continued).

fun	ABC	I-ABC	GABC	ABC/best	CABC	PA-ABC
*f*22	−8.36E+01 (6.61E-01)	−8.30E+01 (6.83E-01)	−8.59E+01 (7.49E-01)	−8.96E+01 (6.49E-01)	**−9.32E+01 (4.87E-01)**	−9.30E+01 (6.53E-01)
Wilcoxon R^- R^+ *p*-value	16 215 0.000543	12 219 0.000321	27 183 0.003592	38 98 0.120839	36 84 0.172848	PA-ABC vs.
Friedman *p*-value = 5.62E-14	5.36	5.00	3.77	2.55	2.39	1.93

algorithms. It can be seen from the ranking results of Friedman test, PA-ABC <CABC<ABC/best<GABC<I-ABC<ABC, indicating that PA-ABC performs best, and *p*-value is far less than 0.05.

Table 5 lists the comparison results among PA-ABC, basic ABC and several typical improved ABCs including I-ABC, GABC, ABC/best and CABC running on 22 basic benchmark functions when D =60 for *f1*-*f20* and D =200 for *f21*-*f22*.

As seen from Table 5, compared with other ABCs, PA-ABC performs best on 18 of 22 basic benchmark functions. Moreover, for *f15* and *f18* which the global optimal solutions are not equal to 0 due to influence of the precision of π, PA-ABC finds their global optimal solutions. Particularly for *f15*, the standard deviation of 30 runs of PA-ABC is 0, indicating that the global optimal solution is found by every running.

It can be seen from the results of Wilcoxon test about significant difference compared with PA-ABC that the *p*-values of ABC, I-ABC, GABC, ABC/best, and CABC are all less than 0.05, indicating that there are significant differences between them and PA-ABC, and PA-ABC is far superior to these five algorithms. It can be seen from the ranking results of Friedman test, PA-ABC <CABC<ABC/best<GABC<I-ABC<ABC, indicating that PA-ABC performs best, and *p*-value is far less than 0.05.

Experiment Result on Shifted and Rotated Functions

To compare the algorithms objectively and avoid using the known characteristics of basic benchmark functions to increase performance of the algorithms artificially, based on the experiments on 22 basic benchmark functions, we test PA-ABC, basic ABC and its several typical variants on the 6 shifted or rotated functions in Table 2, with results shown in Table 6. From Table 6, we can see that PA-ABC performs best among all the algorithms on all of the functions. Particularly for *f23, f25, f27*, and *f28*, only PA-ABC finds the global optimal solution. Therefore, we can conclude that PA-ABC is an effective optimization algorithm with better performance.

Table 5. Result comparisons of ABCs on 60-dimension functions *f1-f20* and 200-dimension functions *f21* and *f22*.

fun	ABC	I-ABC	GABC	ABC/best	CABC	PA-ABC
*f*1	2.68E-17 (2.53E-17)	3.66E-22 (4.63E-22)	6.85E-25 (3.63E-25)	7.95E-33 (4.14E-33)	9.86E-39 (7.93E-39)	**5.02E-42 (2.81E-42)**
*f*2	2.43E-21 (2.77E-22)	5.10E-16 (6.65E-16)	2.39E-23 (3.69E-24)	**3.41E-28 (7.84E-29)**	7.96E-28 (2.75E-28)	1.87E-19 (2.16E-19)
*f*3	5.35E+01 (5.44E+00)	5.20E+01 (3.94E+00)	4.59E+01 (5.86E+00)	3.23E+01 (3.66E+00)	3.17E+01 (3.85E+00)	**2.87E+01 (4.98E+00)**
*f*4	7.97E-06 (2.78E-06)	9.97E-07 (4.69E-07)	8.06E-11 (2.72E-11)	3.29E-15 (5.26E-15)	8.84E-17 (7.10E-17)	**3.62E-19 (1.15E-19)**
*f*5	8.54E-10 (9.72E-10)	2.01E-11 (2.64E-11)	4.95E-19 (8.52E-19)	5.96E-17 (3.72E-16)	2.89E-30 (2.59E-30)	**3.92E-35 (2.66E-35)**
*f*6	3.60E-08 (3.32E-08)	9.87E-10 (5.54E-10)	5.47E-17 (3.78E-17)	2.70E-23 (8.97E-23)	1.01E-27 (2.21E-27)	**5.89E-33 (3.36E-33)**
*f*7	2.57E-20 (5.06E-20)	1.10E-24 (3.56E-24)	1.53E-37 (4.21E-37)	**6.69E-58 (4.11E-57)**	1.87E-40 (3.60E-40)	1.60E-40 (8.15E-40)
*f*8	**0(0)**	**0(0)**	**0(0)**	**0(0)**	**0(0)**	**0(0)**
*f*9	8.56E-01 (2.68E-01)	9.88E-01 (2.54E-01)	3.81E-01 (1.20E-01)	2.93E-01 (7.28E-02)	2.84E-01 (8.26E-02)	**1.87E-02 (6.05E-03)**
*f*10	1.24E+00 (1.18E+00)	1.13E+01 (3.60E+01)	1.67E+01 (3.00E+01)	6.81E+01 (6.11E+01)	4.75E-01 (4.40E-01)	**1.85E-01 (1.91E-01)**
*f*11	7.86E-01 (7.72E-01)	5.91E-01 (7.37E-01)	1.36E-12 (4.48E-12)	**0(0)**	**0(0)**	**0(0)**
*f*12	1.38E-05 (5.62E-06)	3.85E-06 (1.86E-06)	5.87E-10 (2.16E-10)	1.61E-13 (4.19E-14)	8.27E-14 (6.91E-15)	**7.00E-14 (4.92E-15)**
*f*13	4.59E-03 (4.68E-03)	3.76E-03 (2.85E-03)	9.29E-05 (8.19E-05)	6.43E-14 (5.49E-14)	9.44E-17 (2.74E-16)	**7.74E-19 (2.61E-18)**
*f*14	9.30E-09 (7.62E-09)	1.43E-07 (8.87E-06)	**0(0)**	**0(0)**	**0(0)**	**0(0)**
*f*15	4.64E-09 (4.82E-09)	1.20E-10 (2.16E-10)	7.21E-17 (2.07E-16)	3.52E-27 (2.26E-26)	1.42E-31 (2.99E-32)	**1.35E-31 (0.00E+00)**
*f*16	7.97E-01 (7.02E-01)	5.57E-01 (5.93E-01)	1.25E-11 (6.11E-11)	**0(0)**	**0(0)**	**0(0)**
*f*17	3.69E-11 (3.70E-11)	5.49E-13 (5.85E-13)	1.52E-21 (1.46E-21)	7.78E-31 (9.15E-31)	1.37E-32 (6.14E-33)	**7.85E-33 (1.37E-48)**
*f*18	1.89E-09 (1.97E-09)	4.80E-11 (4.39E-11)	1.68E-19 (1.52E-19)	3.12E-29 (2.05E-29)	9.97E-31 (8.35E-31)	**1.35E-32 (5.47E-48)**
*f*19	4.49E-10 (4.73E-10)	5.27E-09 (6.86E-09)	5.09E-11 (9.83E-12)	**4.17E-11 (3.61E-12)**	5.22E-11 (4.25E-12)	1.41E-08 (6.05E-08)
*f*20	2.43E-03 (2.72E-04)	2.95E-03 (3.35E-03)	6.97E-08 (6.12E-08)	**0(0)**	**0(0)**	**0(0)**
*f*21	−7.81E+01 (8.40E-02)	−7.81E+01 (6.26E-02)	−7.83E+01 (2.17E-02)	−7.83E+01 (6.15E-07)	−7.83E+01 (9.22E-09)	**−7.83E+01 (2.47E-09)**
*f*22	−1.50E+02 (2.79E+00)	−1.51E+02 (9.50E-01)	−1.55E+02 (2.38E+00)	−1.62E+02 (8.88E-01)	−1.80E+02 (9.43E-01)	**−1.81E+02 (8.16E-01)**
Wilcoxon R^- R^+ *p*-value	11 220 0.000281	9 222 0.000214	18 172 0.001944	20 116 0.013064	21 115 0.015086	PA-ABC vs.
Friedman *p*-value = 5.62E-14	5.41	5.09	3.75	2.70	2.16	1.89

It can be seen from the results of Wilcoxon test about significant difference compared with PA-ABC that the *p*-values of ABC, I-ABC, GABC, ABC/best, and CABC are all less than 0.05, indicating that there are significant difference

Table 6. Comparisons between PA-ABC and ABCs on shifted and rotated functions.

fun	ABC	I-ABC	GABC	ABC/best	CABC	PA-ABC
f23	3.26E-06 (2.17E-06)	2.18E-06 (8.46E-07)	3.19E-08 (2.47E-08)	3.07E-11 (2.33E-10)	5.10E-12 (4.12E-12)	**0(0)**
f24	2.12E+04 (3.06E+03)	2.05E+04 (6.11E+03)	2.07E+04 (4.87E+03)	2.30E+04 (7.01E+03)	2.53E+04 (5.87E+03)	**5.18E+03 (1.96E+03)**
f25	6.23E-02 (8.95E-02)	1.26E-01 (4.02E-01)	2.20E-05 (2.12E-05)	5.82E-12 (2.33E-11)	1.98E-03 (8.85E-03)	**0(0)**
f26	3.49E+04 (5.83E+03)	2.47E+04 (4.12E+03)	2.11E+04 (3.94E+03)	2.40E+04 (4.13E+03)	3.01E+04 (7.02E+03)	**6.48E+03 (1.46E+03)**
f27	4.28E+0 (2.07E+0)	4.02E+0 (1.13E+0)	3.00E-01 (3.86E-01)	7.14E-11 (2.17E-10)	7.12E-11 (2.07E-10)	**0(0)**
f28	3.22E+0 (1.73E+0)	3.04E+0 (1.38E+0)	9.11E-02 (3.08E-01)	2.88E-11 (4.72E-11)	2.16E-11 (3.91E-11)	**0(0)**
Wilcoxon R^- R^+ *p*-value	0 21 0.027708	0 21 0.027708	0 21 0.027708	0 21 0.027708	0 21 0.027708	PA-ABC vs.
Friedman *p*-value = 0.00169	5.42	4.50	3.42	3.00	3.67	1.00

between them and PA-ABC, and PA-ABC is far superior to these five algorithms. It can be seen from the ranking results of Friedman test, PA-ABC <ABC/best<GABC<CABC<I-ABC<ABC, indicating that PA-ABC performs best, and *p*-value is far less than 0.05.

Conclusions

To improve the performance, PA-ABC adopts two novel search equations for the employed bee and the onlooker bee. The search equation of the employed bee focuses on the balance of exploration and exploitation on high level, enhancing exploitation by slight turbulence and the introduce of the best-so-far solution as guiding information while ensuring high exploration by randomly selecting food sources to search around. The search equation of the onlooker bee realizes enhancing exploitation by the introduce of the best-so-far solution as guiding information while ensuring high population diversity by selecting random food source to search around, and adjusting of the search step size automatically by the introduce of adaptive parameter, which can make the search process transform from early exploration to late exploitation. Experiment results on benchmark functions have shown that PA-ABC has better effectiveness and robustness. Future work will focus on the application of PA-ABC algorithm to solve practical problems, such as circular antenna array design (Bose, Kundu, Biswas, Das 2012), dynamic optimization problems (Biswas, Bose, and Kundu 2012), parameter optimization (Biswas, Saha, De, Cobb, Das, Jalaian 2021), etc.

Disclosure Statement

No potential conflict of interest was reported by the author(s).

Funding

This work was supported by the Department of Education of Liaoning Province (lnzd202004).

References

Akay, B., and D. Karaboga. 2012. A modified artificial bee colony algorithm for real-parameter optimization. *Information Sciences* 192:120–42. doi:10.1016/j.ins.2010.07.015.

Alatas, B. 2010. Chaotic bee colony algorithms for global numerical optimization. *Expert Systems with Applications* 37 (8):5682–87. doi:10.1016/j.eswa.2010.02.042.

Ali, W.-M., A.-H. Anas, K.-M. Ali, Prachi, A., Abhishek, K., Suganthan, P.-N.Problem definitions and evaluation criteria for the CEC 2021 special session and competition on single objective bound constrained numerical optimization. Technical Report, Nanyang Technological University, Singapore.

Anguluri, R., A. Ajith, and P. Millie 2011. Levy mutated artificial bee colony algorithm for global optimization. IEEE International Conference on Systems, Man and Cybernetics, Anchorage: Alaska.

Babayigit, B., and R. Ozdemir 2012. A modified artificial bee colony algorithm for numerical function optimization. IEEE Symposium on Computers and Communications, Cappadocia: Turkey.

Banharnsakun, A., T. Achalakul, and B. Sirinaovakul. 2011. The best-so-far selection in artificial bee colony algorithm. *Applied Soft Computing Journal* 11 (2):2888–901. doi:10.1016/j.asoc.2010.11.025.

Bhandari, A. K., A. Kumar, and G. K. Singh. 2015. Modified artificial bee colony based computationally efficient multilevel thresholding for satellite image segmentation using Kapur’s, Otsu and Tsallis functions. *Expert Systems with Applications* 42 (3):1573–160. doi:10.1016/j.eswa.2014.09.049.

Biswas, S., D. Bose, and S. Kundu 2012. A clustering particle based artificial bee colony algorithm for dynamic environment. International Conference on Swarm. Springer, Berlin, Heidelberg.

Biswas, S., D. Saha, S. De, Cobb, A.-D., Das, S., Jalaian, B.-A.2021. Improving differential evolution through Bayesian hyperparameter optimization. 2021 IEEE Congress on Evolutionary Computation (CEC), PolandIEEE. doi: 10.1109/CEC45853.2021.9504792.

Bose, D., S. Kundu, S. Biswas, Das, S. 2012. Circular antenna array design using novel perturbation based artificial bee colony algorithm. *Lecture Notes in Computer Ence* 7677:459–66.

Brajević, I., and P. Stanimirović. 2018. An improved chaotic firefly algorithm for global numerical optimization. *International Journal of Computational Intelligence Systems* 12 (1):131–48. doi:10.2991/ijcis.2018.25905187.

Brajević, I., P.-S. Stanimirović, S. Li, X. Cao. 2020. A hybrid firefly and multi-strategy artificial bee colony algorithm. *International Journal of Computational Intelligence Systems*. 13 (1):810–21. doi:10.2991/ijcis.d.200612.001.

Brajević, I. 2021. A shuffle-based artificial bee colony algorithm for solving integer programming and minimax problems. *Mathematics* 9 (11):1211. doi:10.3390/math9111211.

Cui, L., G. Li, X. Wang, Q. Lin, J. Chen, N. Lu, J. Lu. 2017. A ranking-based adaptive artificial bee colony algorithm for global numerical optimization. *Information Sciences* 417:169–85. doi:10.1016/j.ins.2017.07.011.

Das, S., and P.-N. Suganthan. 2010. Differential evolution: A survey of the state-of-the-art. *IEEE Transactions on Evolutionary Computation* 15 (1):4–31. doi:10.1109/TEVC.2010.2059031.

Das, S., S. Biswas, B.-K. Panigrahi, S. Kundu, D. Basu. 2014. A spatially informative optic flow model of bee colony with saccadic flight strategy for global optimization. *IEEE Transactions on Cybernetics*. 44(10):1884–97. doi:10.1109/TCYB.2014.2298916.

Das, S., S. Biswas, and S. Kundu. 2013. Synergizing fitness learning with proximity-based food source selection in artificial bee colony algorithm for numerical optimization. *Applied Soft Computing* 13 (12):4676–94. doi:10.1016/j.asoc.2013.07.009.

Dorigo, M., and T. Stutzle. 2004. *Ant colony optimization*. Cambridge: Massachusetts Institute of Technology Press.

Gao, G.-W., S.-Y. Liu, and -L.-L. Huang. 2013. A novel artificial bee colony algorithm based on modified search equation and orthogonal learning. *IEEE Transactions on Cybernetics* 43 (3):1011–24. doi:10.1109/TSMCB.2012.2222373.

Gao, W.-F., S.-Y. Liu, and -L.-L. Huang. 2012. A global best artificial bee colony algorithm for global optimization. *Journal of Computational and Applied Mathematics* 236 (11):2741–53. doi:10.1016/j.cam.2012.01.013.

Gao, W.-F., and S.-Y. Liu. 2011. Improved artificial bee colony algorithm for global optimization. *Information Processing Letters* 111 (17):871–82. doi:10.1016/j.ipl.2011.06.002.

Gao, W.-F., and S.-Y. Liu. 2012. A modified artificial bee colony algorithm. *Computers & Operations Research* 39 (3):687–97. doi:10.1016/j.cor.2011.06.007.

Gao, W., H. Sheng, J. Wang, S. Wang. 2018. Artificial bee colony algorithm based on novel mechanism for fuzzy portfolio selection. *IEEE Transactions on Fuzzy Systems*. 27(5):966–78. doi:10.1109/TFUZZ.2018.2856120.

Imanian, M., E. Shiri, and P. Moradi. 2014. Velocity based artificial bee colony algorithm for high dimensional continuous optimization problems. *Engineering Applications of Artificial Intelligence* 36 (12):148–63. doi:10.1016/j.engappai.2014.07.012.

Kang, F., -J.-J. Li, and Z.-Y. Ma. 2011. Rosenbrock artificial bee colony algorithm for accurate global optimization of numerical functions. *Information Sciences* 181 (16):3508–31. doi:10.1016/j.ins.2011.04.024.

Karaboga, D. 2005. An idea based on honey bee swarm for numerical optimization. Technical Report-TR06,Turkey Kayseri: Erciyes University.

Karaboga, D., and B. Akay. 2009. A comparative study of artificial bee colony algorithm. *Applied Mathematics and Computation* 214 (1):108–32. doi:10.1016/j.amc.2009.03.090.

Karaboga, D., and B. Basturk. 2008. On the performance of artificial bee colony (ABC) algorithm. *Applied Soft Computing Journal* 8 (1):687–97. doi:10.1016/j.asoc.2007.05.007.

Karaboga, D., B. Gorkemli, C. Ozturk, et al. 2014. A comprehensive survey: Artificial bee colony (ABC) algorithm and applications. *Artificial Intelligence Review* 42 (1):21–57.

Kennedy, J., and R.-C. Eberhart. 1995. Particle swarm optimization. *IEEE International Conference Neural Networks* 4 (8):1942–48.

Kong, D., T.-Q. Chang, W.-J. Dai, Q. Wang, H. Sun. 2018. An improve d artificial b ee colony algorithm based on elite group guidance and combined breadth-depth search strategy. *Information Sciences* 442-443:54–71. doi:10.1016/j.ins.2018.02.025.

Li, B., Y. Li, and L.G. Gong. 2014. Protein secondary structure optimization using an improved artificial bee colony algorithm based on AB off-lattice model. *Engineering Applications of Artificial Intelligence* 27 (1):70–79. doi:10.1016/j.engappai.2013.06.010.

Li, G.-Q., P.-F. Niu, and X.-J. Xiao. 2012. Development and investigation of efficient artificial bee colony algorithm for numerical function optimization. *Applied Soft Computing Journal* 12 (1):320–32. doi:10.1016/j.asoc.2011.08.040.

Li, J.-Q., Q.-K. Pan, and M. Tasgetiren. 2014a. A discrete artificial bee colony algorithm for the multi-objective flexible job-shop scheduling problem with maintenance activities. *Applied Mathematical Modelling* 38 (3):1111–32. doi:10.1016/j.apm.2013.07.038.

Lin, Q.-Z., -M.-M. Zhu, G. Li, W. Wang, L. Cui, J. Chen, and J. Lu. 2018. A novel artificial bee colony algorithm with local and global information interaction. *Applied Soft Computing* 62:702–35. doi:10.1016/j.asoc.2017.11.012.

Liu, Y., -X.-X. Ling, Y. Liang, and G. Liu. 2012. Improved artificial bee colony algorithm with mutual learning. *Journal of Systems Engineering and Electronics* 23 (2):265–75. doi:10.1109/JSEE.2012.00034.

Luo, J., Q. Wang, and X.-H. Xiao. 2013. A modified artificial bee colony algorithm based on converge-onlookers approach for global optimization. *Applied Mathematics and Computation* 219 (20):10253–62. doi:10.1016/j.amc.2013.04.001.

Rajasekhar, A., N. Lynn, S. Das, P. N. Suganthan. 2017. Computing with the collective intelligence of honey bees –a survey. *Swarm and Evolutionary Computation* 32:25–48. doi:10.1016/j.swevo.2016.06.001.

Shi, Y., C.-M. Pun, H. Hu, H. Gao. 2016. An improved artificial bee colony and its application. *Knowledge-Based Systems*. 107(9):14–31. doi:10.1016/j.knosys.2016.05.052.

Simon, D. 2008. Biogeography-basefigured optimization. *IEEE Transactions on Evolutionary Computation* 12 (6):702–13. doi:10.1109/TEVC.2008.919004.

Song, X.-Y., M. Zhao, Q.-F. Yan, and S.-Y. Xing. 2019. A high-efficiency adaptive artificial bee colony algorithm using two strategies for continuous optimization. *Swarm and Evolutionary Computation* 50:100549. doi:10.1016/j.swevo.2019.06.006.

Song, X.-Y., Q.-F. Yan, and M. Zhao. 2017. An fadaptive artificial bee colony algorithm based on objective function value information. *Applied Soft Computing* 55:384–401. doi:10.1016/j.asoc.2017.01.031.

Song, X., M. Zhao, and S. Xing. 2019. A multi-strategy fusion artificial bee colony algorithm with small population. *Expert System Application* 142:112921. doi:10.1016/j.eswa.2019.112921.

Storn, R., and K. Price. 1997. Differential evolution-A simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization* 11 (4):341–59. doi:10.1023/A:1008202821328.

Tang, K.-S., K.-F. Man, S. Kwong, Q. He. 1996. Genetic algorithms and their applications. *IEEE Signal Processing Magazine*. 13(6):22–37. doi:10.1109/79.543973.

Tsai, P.-W., J.-S. Pan, and B.-Y. Liao. 2009. Enhanced artificial bee colony optimization. international journal of innovative computing. *Information and Control* 5 (12):5081–92.

Vural, R.-A., Yildirim, T., Kadioglu, T. et al, 2012. Performance Evaluation of Evolutionary Algorithms for Optimal Filter Design. *IEEE Transactions on Evolutionary Computation*, 16 (1): 135–147. doi: 10.1109/TEVC.2011.2112664.

Xiang, W.-L., and M.-Q. An. 2012. An efficient and robust artificial bee colony algorithm for numerical optimization. *Computers & Operations Research* 40 (5):1256–65. doi:10.1016/j.cor.2012.12.006.

Yang, Q., W. Chen, Z. Yu, T. Gu, Y. Li, H. Zhang, J. Zhang. 2017. Adaptive multimodal continuous ant colony optimization. *IEEE Transactions on Evolutionary Computation*. 21 (2):191–205. doi:10.1109/TEVC.2016.2591064.

Zhang, C.-Q., J.-G. Zheng, and Y.-Q. Zhou. 2015. Two modified artificial bee colony algorithms inspired by Grenade Explosion Method. *Neurocomputing* 151 (P3):1198–207. doi:10.1016/j.neucom.2014.04.082.

Zhou, J., X. Yao, F.-T.-S. Chan, et al. 2019. An individual dependent multi-colony artificial bee colony algorithm. *Information Science* 485:114–40. doi:10.1016/j.ins.2019.02.014.

Zhu, G.-P., and S. Kwong. 2010. Gbest-guided artificial bee colony algorithm for numerical function optimization. *Applied Mathematics and Computation* 217 (7):3166–73. doi:10.1016/j.amc.2010.08.049.