Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2016, Article ID 9791060, 10 pages
http://dx.doi.org/10.1155/2016/9791060

Research Article

Particle Swarm and Bacterial Foraging Inspired Hybrid Artificial Bee Colony Algorithm for Numerical Function Optimization

Li Mao,[1] Yu Mao,[1] Changxi Zhou,[1] Chaofeng Li,[1,2] Xiao Wei,[3] and Hong Yang[3]

[1]*Key Laboratory of Advanced Process Control for Light Industry (Ministry of Education), School of Internet of Things, Jiangnan University, Wuxi, Jiangsu 214122, China*
[2]*Laboratory of Computational Geodynamics, University of Chinese Academy of Sciences, Beijing 100049, China*
[3]*Freshwater Fisheries Research Center, Chinese Academy of Fishery Science, Wuxi, Jiangsu 214081, China*

Correspondence should be addressed to Chaofeng Li; wxlichaofeng@126.com

Received 4 November 2015; Revised 4 January 2016; Accepted 4 January 2016

Academic Editor: Matjaz Perc

Artificial bee colony (ABC) algorithm has good performance in discovering the optimal solutions to difficult optimization problems, but it has weak local search ability and easily plunges into local optimum. In this paper, we introduce the chemotactic behavior of Bacterial Foraging Optimization into employed bees and adopt the principle of moving the particles toward the best solutions in the particle swarm optimization to improve the global search ability of onlooker bees and gain a hybrid artificial bee colony (HABC) algorithm. To obtain a global optimal solution efficiently, we make HABC algorithm converge rapidly in the early stages of the search process, and the search range contracts dynamically during the late stages. Our experimental results on 16 benchmark functions of CEC 2014 show that HABC achieves significant improvement at accuracy and convergence rate, compared with the standard ABC, best-so-far ABC, directed ABC, Gaussian ABC, improved ABC, and memetic ABC algorithms.

1. Introduction

In recent years, facing optimization problems, people have put forward a series of traditional solving methods, such as linear programming and dynamic planning. Limited by the huge time complexity, these methods are not applied to solve these large-scale problems. With the development of biotechnology, people found that when individuals work together for a complex job the ability of individuals is not a simple addition of every individual but a very complex behavioral feature. So, many swarm intelligence based optimization methods have been proposed. Inspired by the law of survival of the fittest, Tang et al. [1] proposed the genetic algorithm (GA). Inspired by the foraging behavior of ant colonies, Dorigo et al. [2] proposed the ant colony optimization (ACO). Inspired by the social behavior of bird flocking, Kennedy and Eberhart [3] proposed the particle swarm optimization (PSO). Li et al. [4] proposed artificial fish swarm algorithm (AFSA). Yang [5] proposed firefly algorithm (FA) and Fister et al. [6] summarized some improvement of chaos-based firefly algorithms. Motivated by the behaviors of honeybee swarms, artificial bee colony (ABC) algorithm was first proposed by Karaboga in 2005 [7]. ABC algorithm has been widely used in many fields, for example, determining the optimal size and selecting optimum locations of shunt capacitors by El-Fergany and Abdelaziz [8], solving the economic lot scheduling problem by Bulut and Tasgetiren [9], segmenting SAR image by Ma et al. [10], enhancing image contrast by Draa and Bouaziz [11], and solving the Leaf-Constrained Minimum Spanning Tree (LCMST) problem by Singh [12].

Despite the success of the ABC algorithm in many applications, it still has numerous drawbacks. As we all know, strong capabilities in both exploration and exploitation are important for population-based optimization algorithms. Although the standard ABC algorithm works fine with exploration, it performs poorly in exploitation. To improve the performance, researchers have been working on modifying ABC algorithm and integrating ABC algorithm with other evolutionary computation based optimization methods. For example, Bansal et al. proposed a self-adaptive ABC algorithm, which updates step size and parameters for searching

solutions adaptively according to the current fitness values; therefore it gives the solutions more chances to update themselves [13]. Wang et al. presented multistrategy ensemble ABC algorithm by utilizing different characteristics of solution search to construct a strategy pool. During the search process, the strategy for each food source is dynamically changed in order to achieve better performance [14]. Kang et al. presented Rosenbrock ABC algorithm [15] by combing standard ABC and RM-based local search techniques. To improve global searching capability by escaping the local solutions, Alatas [16] adopted a method to adjust parameters for ABC algorithm using random numbers generated from different chaotic systems. Xiang et al. proposed a particle swarm inspired multielitist ABC algorithm [17] which updates the parameters of the solutions using global best solution and an elitist randomly selected from an elitist archive. To efficiently solve the constraint optimization problems, Li and Yin [18] presented a self-adaptive constrained ABC algorithm by introducing feasible rule and multiobjective optimization methods.

In this paper, we propose a hybrid ABC algorithm to improve the performance of standard ABC algorithm. To enhance the ability of local searching and exploitation, we applied chemotactic behavior in Bacterial Foraging Optimization algorithm [19] into employed bees and adopted the global best solution search equation in PSO [20, 21] algorithm into onlooker bees. Moreover, we also use inertia weight [22, 23] like contraction-expansion coefficient in PSO algorithm to balance exploration and exploitation dynamically. Finally, our algorithms are evaluated by the average values and standard deviation (SD) of 16 CEC 2014 benchmark functions.

This paper is organized as follows. Section 2 describes the standard ABC algorithm. Section 3 introduces the HABC algorithm. The experimental results are shown and discussed in Section 4. Section 5 presents the conclusions.

2. Standard ABC Algorithm

When solving optimization problems, ABC algorithm abstracts the food sources as feasible solutions. The process of artificial bee colony seeking for quality food sources is used to simulate the process of finding the global optimal solution. The honeybee swarms are categorized into three groups: employed, onlooker, and scout bees, where the number of employed bees equals the number of high quality food sources. The job of employed bees is to find quality food sources and then share food location information with onlooker bees. Once onlooker bees obtain the food source information, they search closer toward the selected food sources according to the probability distribution functions. Higher quality food sources have a higher possibility to be selected. When food sources found by employed bees are identified as low quality ones, the corresponding employed bees turn into scout bees which search for new food sources randomly. The algorithm can be divided into four steps: initialization, behavior of employed bees, behavior of onlooker bees, and behavior of scout bees.

2.1. Initialization. In the initialization step, food sources are initialized with random positions given by the following equation:

$$x_{i,j} = x_j^{\min} + \text{rand}\,[0,1]\left(x_j^{\max} - x_j^{\min}\right). \tag{1}$$

Every solution x_i $(i = 1, 2, \ldots, S_N)$ is a D-dimensional vector, S_N is the number of food sources which is equal to the number of employed bees or onlooker bees, and D denotes the number of optimization parameters. $x_j^{\min}, x_j^{\max}$ are the lower and upper bounds of food source positions at dimension j, respectively.

2.2. Behavior of Employed Bees. Starting from the initial locations, employed bees search around the surrounding areas for better food sources. The algorithm assumes that employed bees can record all the food source locations that the colony has reached; thus employed bees can move a random distance toward another food source. The updated location is calculated as follows:

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

where $v_{i,j}$ is the new candidate food source location, $i \in \{1, 2, \ldots, S_N\}$ (S_N is the population of swarm), $j \in \{1, 2, \ldots, D\}$ (D represents the dimension); $x_{i,j}$ is the previous food source location; φ is a random number between $[-1, 1]$; $x_{k,j}$ is another food source location, $k \in \{1, 2, \ldots, S_N\}$; and $k \neq i$.

The fitness of a solution can be calculated from the objective function value by using the following equation:

$$\text{fitness}\left(x_i\right) = \begin{cases} \dfrac{1}{\left(1 + f\left(x_i\right)\right)} & \text{if } f\left(x_i\right) \geq 0 \\ 1 + \left|f\left(x_i\right)\right| & \text{if } f\left(x_i\right) < 0. \end{cases} \tag{3}$$

2.3. Behavior of Onlooker Bees. When bee colony foraging for food, onlooker bees stay around the hive and search locally for high quality food sources by observing the information of food sources carried by employed bees. The information interaction is an important reflection of the intelligent behavior of the bee colonies in searching for food. The employed bees provide the information of food sources to onlooker bees after returning to the hive. The onlooker bees decide whether to update the food source using the greedy algorithm when searching for food and the probability of updating the food source is

$$P_i = \frac{\text{fitness}\left(x_i\right)}{\sum_{i=1}^{S_N} \text{fitness}\left(x_i\right)}, \tag{4}$$

where fitness(x_i) is the fitness function of ith onlooker bees. If the new value of fitness function is better, the onlooker bee will update its position using (2); thus, the bee colony can move closer to better-quality food sources gradually.

2.4. Behavior of Scout Bees. If food sources cannot be improved after a predefined number of iterations, they will be abandoned. The corresponding employed bees will change to scout bees which search for a new feasible food source randomly across a wide range using same (1) for initialization.

3. Hybrid Artificial Bee Colony Algorithm

3.1. For Employed Bees. For employed bees, we adopt the chemotactic behavior in Bacterial Foraging Optimization (BFO) to help employed bees to escape local optimum trap and enter a relatively large searching space gradually. Chemotactic behavior is that bacteria gather together at a more favorable environment instead of a noxious one, which includes two operators: tumble and swim. A unit walk with random direction represents tumble operator. After finishing a tumble operator, if the fitness value is not improved, then bacteria move to another random direction with a unit walk; if the fitness value improved, then bacteria move to the same direction with a few unit walks until the fitness value reaches the maximum swim steps, where the process represents swim operator. In iterative procedure, standard ABC algorithm implement global search by employed bees. As shown in (2), $x_{i,j}$ is a random dimension of an individual, and standard ABC algorithm updates only one randomly selected dimension of the solutions in employed bees' process, which causes some redundancy in searching for solutions. Compared with the standard ABC algorithm, in our proposed algorithm the frequency and range of the neighborhood search are increased when processing the employed bees. Once the employed bees forage, the fitness value of the solutions will be calculated. If the fitness value improves, the old position is substituted for the new position. Otherwise, the employed bees remain in the old position. When the number of swim steps (called N_s) reaches the limit (called $N_{s\max}$) or the fitness value is compounded, employed bees stop foraging at current dimension and tumble to another dimension. These operations can ameliorate the convergence speed of employed bees and increase potential solutions. After all the employed bees have tumbled in all dimensions, the process finishes. Benefitting from these, the bee colony gets into a larger search space and avoids plunging into local optimum. In HABC algorithm, the positions of employed bees can be updated as follows:

$$v_{i,j} = x_{i,j} + N_s \times \varphi\left(x_{i,j} - x_{k,j}\right), \tag{5}$$

where $v_{i,j}$, $x_{i,j}$, $x_{k,j}$, and φ are the same as (2) and N_s is the number of advances, like swim steps in Bacterial Foraging Optimization algorithm. The improvement is shown in Figures 1 and 2.

The pseudocode of the behavior of employed bees in HABC algorithm is as given in Pseudocode 1.

3.2. For Onlooker Bees. In the standard ABC algorithm, employed bees and onlooker bees update their position by searching for a candidate solution in a randomly selected dimension of current position toward a random position. The purpose of employed bees is for global searching across a relatively large space. While the purpose of onlooker bees is for local searching in a neighboring area, which means that employed bees have the fastest convergence rate and onlooker bees have the best exploration ability. Eventually, we can locate the global optimization solution. In standard ABC algorithm, step size and dimension are randomly chosen for onlooker bees' process, so the probabilities of choosing good and bad quality food sources are the same, which lead to its low robustness.

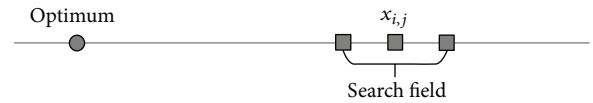

Figure 1: An illustrative example of employed bees' search field in standard ABC.

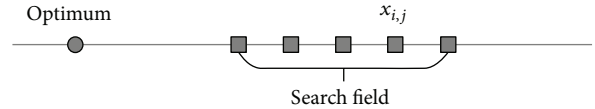

Figure 2: An illustrative example of employed bees' search field in HABC.

As we all know, the distance of global optimal solution and suboptimal solution is short, and then the current best solution can drive the new candidate solution step towards the best direction, so we adopt the idea of tracing the current best particle from PSO algorithm to enhance the global search capability of the standard ABC algorithm. At the same time, to strike a balance between exploration and exploitation, we introduce the inertia weight ω into our algorithm. In PSO algorithm, inertia weight represents the ability that particles inherit from their previous velocity; it was first introduced into PSO algorithm by Shi and Eberhart in 1998 [22]. Analysis states that a relatively larger inertia weight is good for global searching and a relatively smaller inertia weight is good for local searching. In HABC algorithm, we adopt linear decreasing inertia weight (called LDIW), which contains smaller step size and difficult trap into local optimum. At the initial stages, in order to avoid trapping into a local optimal solution, a relatively large inertia weight is used to make onlooker bees spread into a larger search space. At the end stages, a relatively small inertia weight is used to protect onlooker bees from disturbing the current best solution. This approach enhances the overall convergence speed and makes the algorithm more efficient for obtaining the global optimization solution. The equation of LDIW is given as follows:

$$\omega\,(\text{iter}) = \frac{\omega_{\text{start}}\left(\omega_{\text{start}} - \omega_{\text{end}}\right)\left(\text{iter}_{\max} - \text{iter}\right)}{\text{iter}_{\max}}. \tag{6}$$

Among all parameters, ω_{start} is the initial inertia weight, ω_{end} is the inertia weight when iteration reach the maximum; iter respect the current iteration, and $\text{iter}_{\max}$ means the maximum number of iterations. In our algorithm, according to the experience, we set ω_{start} as 0.9 and ω_{end} as 0.4.

In HABC algorithm, the positions of onlooker bees can be updated as follows:

$$v_{i,j} = \omega x_{i,j} + \varphi\left(x_{i,j} - x_{k,j}\right) + \varphi\left(x_{\text{best},j} - x_{k,j}\right), \tag{7}$$

where $v_{i,j}$, $x_{i,j}$, $x_{k,j}$, and φ are the same as (2), $x_{\text{best},j}$ is dimension j of current best solution, and ω is the LDIW.

(1) Set the source position X_i, produce new solution V_i.
(2) for i (as a counter) from 1 to colony size
(3) j = rand(1, D) (as a random dimension of source position)
(4) k = rand(1, colonysize) & $k! = i$ (as another random dimension of source position)
(5) set temp = rand($-1 \sim 1$) $\times$ $(x_{i,j} - x_{k,j})$
(6) for dim (as a counter from 1 to max dimension)
(7) while $N_s < N_{s\max}$
(8) new solution $v_{i,\dim} = x_{i,\dim}$ + temp
(9) if new fitness value $f(v_{i,\dim})$ is better than fitness value $f(x_{i,\dim})$
(10) then $x_{i,\dim} = v_{i,\dim}$
(11) else
(12) set $N_s = N_{s\max}$
(13) end if
(14) end while
(15) end for
(16) end for

Pseudocode 1

(1) Set the source position X_i, produce new solution V_i, maximum convergence iterations $\text{iter}_{\max}$, current iteration iter.
(2) for i (as a counter) from 1 to colony size
(3) calculate selective probability p
(4) j = rand(1, D) (as a random dimension of source position)
(5) k = rand(1, colonysize) & $k! = i$ (as another random dimension of source position)
(6) set $\omega = 0.9(0.9 - 0.5)(\text{iter}_{\max} - \text{iter})/\text{iter}_{\max}$
(7) if selective probability $p >$ rand($0 \sim 1$)
(8) for dim (as a counter) from 1 to max dimension
(9) new solution $v_{i,\dim} = \omega x_{i,\dim} + \varphi(x_{i,j} - x_{k,j}) + \varphi(x_{\text{best},j} - x_{k,j})$
(10) end for
(11) if new fitness value $f(v_{i,\dim})$ is better than fitness value $f(x_{i,\dim})$
(12) then $x_{i,\dim} = v_{i,\dim}$
(13) end if
(14) end if
(15) end for

Pseudocode 2

The pseudocode of onlooker bees is as given in Pseudocode 2.

4. Experimental Comparison and Analysis

4.1. Benchmark Functions Used. In this section, algorithms are used to find the global optimum values of 16 benchmark functions from the CEC 2014 competition [24]. The details of the functions are as follows.

Unimodal functions include the following.

(1) Rotated High Conditioned Elliptic Function.
(2) Rotated Bent Cigar Function.
(3) Rotated Discus Function.

Multimodal functions include the following.

(4) Shifted and Rotated Rosenbrock's Function.
(5) Shifted and Rotated Ackley's Function.
(6) Shifted and Rotated Weierstrass Function.
(7) Shifted and Rotated Griewank's Function.
(8) Shifted Rastrigin's Function.
(9) Shifted and Rotated Rastrigin's Function.
(10) Shifted Schwefel's Function.
(11) Shifted and Rotated Schwefel's Function.
(12) Shifted and Rotated Katsuura Function.
(13) Shifted and Rotated HappyCat Function.
(14) Shifted and Rotated HGBat Function.
(15) Shifted and Rotated Expanded Griewank's plus Rosenbrock's Functions.
(16) Shifted and Rotated Expanded Scaffer's *F*6 Function.

4.2. Parameter Settings. To test the performance of HABC, the experimental results are compared with those of the standard ABC, best-so-far ABC (BSABC)[25], directed ABC (dABC) [26], Gaussian ABC (GABC) [27], improved ABC

Table 1: Results of HABC and other algorithms for 10D.

Function	ABC			BSABC			dABC			GABC			IABC			MABC			HABC	
	Mean	Std.	Sign.	Mean	Std.	Sign.	Mean	Std.	Sign.	Mean	Std.	Sign.	Mean	Std.	Sign.	Mean	Std.	Sign.	Mean	Std.
$F1$	$2.26E+05$	$7.65E+04$	+	$3.74E+04$	$2.02E+04$	+	$1.76E+05$	$6.41E+04$	+	$4.94E+04$	$2.21E+04$	+	$8.50E+04$	$6.33E+04$	+	$1.09E+05$	$8.81E+04$	+	**$2.50E+04$**	**$9.77E+03$**
$F2$	$2.18E+01$	$7.93E+00$	+	$1.03E-02$	$1.32E-03$	+	$3.14E-02$	$1.86E-02$	+	$1.55E-03$	$1.03E-03$	−	$1.38E-02$	$1.03E-02$	+	**$4.80E-04$**	**$5.70E-04$**	−	$2.38E-03$	$2.33E-03$
$F3$	$1.71E-01$	$1.17E-01$	+	$3.54E-02$	$3.49E-02$	+	$6.42E-02$	$4.06E-02$	+	$1.74E-02$	$1.33E-02$	+	$7.33E-02$	$7.45E-02$	+	$3.63E-02$	$6.44E-02$	+	**$8.46E-03$**	**$3.88E-03$**
$F4$	$7.20E-01$	$3.78E-01$	+	$1.91E-02$	$1.37E-02$	+	$4.70E-02$	$2.33E-02$	+	$2.01E-02$	$1.03E-02$	+	$9.22E-02$	$8.13E-02$	+	$6.03E-03$	$1.15E-02$	+	**$4.38E-03$**	**$3.13E-03$**
$F5$	$1.75E+01$	$4.55E+00$	+	$8.96E+00$	$7.64E+00$	+	$1.39E+01$	$7.85E+00$	+	$5.66E+00$	$3.42E+00$	+	$9.35E+00$	$7.39E+00$	+	$1.51E+01$	$8.34E+00$	+	**$1.65E+00$**	**$1.01E+00$**
$F6$	$2.83E-02$	$4.93E-03$	+	$1.16E-02$	$3.07E-03$	+	$1.59E-02$	$4.08E-03$	+	$1.14E-02$	$2.31E-03$	+	$1.37E-02$	$3.32E-03$	+	**$3.39E-03$**	**$9.36E-04$**	−	$7.46E-03$	$1.53E-03$
$F7$	$3.24E-02$	$7.30E-03$	+	$1.09E-04$	$1.49E-04$	+	$3.56E-03$	$3.35E-03$	+	$7.63E-05$	$1.54E-05$	+	$4.48E-04$	$6.33E-04$	+	**$4.02E-08$**	**$5.97E-08$**	−	$8.46E-06$	$1.42E-05$
$F8$	**0**	**0**	NA	**0**	**0**	NA	**0**	**0**	NA	**0**	**0**	NA	**0**	**0**	NA	**0**	**0**	NA	**0**	**0**
$F9$	$7.86E+00$	$1.55E+00$	+	$4.53E+00$	$1.01E+00$	+	$4.93E+00$	$6.34E-01$	+	$5.61E+00$	$1.02E+00$	+	$3.27E+00$	$6.98E-01$	=	$3.42E+00$	$7.80E-01$	+	**$3.09E+00$**	**$6.80E-01$**
$F10$	**0**	**0**	NA	$8.93E-03$	$2.36E-02$	+	$3.93E-02$	$3.01E-02$	+	$6.25E-02$	$3.61E-02$	+	**0**	**0**	NA	$1.25E-01$	$5.10E-02$	+	**0**	**0**
$F11$	$2.89E+02$	$4.01E+01$	+	$7.33E+01$	$4.65E+01$	+	$1.31E+02$	$7.26E+01$	+	$1.08E+02$	$7.16E+01$	+	$8.26E+01$	$3.32E+01$	+	$4.91E+01$	$5.83E+01$	+	**$2.79E+01$**	**$1.01E+01$**
$F12$	$3.71E-03$	$5.64E-04$	+	$1.75E-03$	$2.71E-04$	+	$2.53E-03$	$5.01E-04$	+	$1.48E-03$	$3.03E-04$	+	$2.51E-03$	$2.83E-04$	+	**$3.98E-04$**	**$2.13E-04$**	−	$7.62E-04$	$1.50E-04$
$F13$	$1.56E-03$	$2.89E-04$	+	$9.41E-04$	$1.70E-04$	+	$1.05E-03$	$1.35E-04$	+	$9.70E-04$	$1.27E-04$	+	$9.11E-04$	$2.17E-04$	+	$7.11E-04$	$2.40E-04$	+	**$5.88E-04$**	**$1.22E-04$**
$F14$	$1.09E-03$	$2.37E-04$	+	$1.07E-03$	$1.21E-04$	+	$9.55E-04$	$1.01E-04$	+	$1.14E-03$	$2.12E-04$	+	$1.04E-03$	$1.81E-04$	+	$8.18E-04$	$2.42E-04$	+	**$7.16E-04$**	**$6.45E-05$**
$F15$	$1.53E+00$	$2.32E-01$	+	$6.17E-01$	$1.63E-01$	+	$8.71E-01$	$1.05E-01$	+	$5.04E-01$	$1.50E-01$	+	$6.48E-01$	$1.95E-01$	+	$4.54E-01$	$1.71E-01$	+	**$2.24E-01$**	**$1.16E-01$**
$F16$	$2.40E+00$	$1.39E-01$	+	$1.80E+00$	$3.13E-01$	+	$2.12E+00$	$2.23E-01$	+	$1.94E+00$	$1.62E-01$	+	$1.84E+00$	$1.92E-01$	+	$1.39E+00$	$4.06E-01$	=	**$1.33E+00$**	**$2.17E-01$**

Table 2: Results of HABC and other algorithms for 30D.

Function	ABC			BSABC			dABC			GABC			IABC			MABC			HABC	
	Mean	Std.	Sign.	Mean	Std.	Sign.	Mean	Std.	Sign.	Mean	Std.	Sign.	Mean	Std.	Sign.	Mean	Std.	Sign.	Mean	Std.
$F1$	$3.45E+07$	$8.47E+06$	+	$8.07E+06$	$2.07E+06$	+	$2.41E+07$	$7.01E+06$	+	$6.00E+06$	$1.10E+06$	+	$1.42E+07$	$3.26E+06$	+	$6.33E+06$	$1.67E+06$	+	**$3.53E+06$**	**$7.02E+05$**
$F2$	$2.25E+03$	$8.56E+02$	+	$3.98E-02$	$2.06E-02$	+	$4.99E-02$	$4.99E-02$	+	$1.72E-02$	$2.38E-02$	+	$3.79E-02$	$2.30E-02$	+	$8.18E-03$	$7.98E-03$	+	**$1.21E-03$**	**$8.12E-04$**
$F3$	$1.81E+00$	$1.60E+00$	+	$3.03E-01$	$2.13E-01$	+	$1.08E+00$	$6.66E-01$	+	$1.94E-01$	$1.23E-01$	+	$2.40E-01$	$2.30E-01$	+	$3.27E-01$	$2.99E-01$	+	**$4.35E-02$**	**$2.50E-02$**
$F4$	$9.22E+01$	$9.86E+00$	+	$3.43E+00$	$3.75E+00$	+	$1.35E+01$	$1.11E+01$	+	$2.79E+00$	$1.29E+00$	+	$1.28E+01$	$9.58E+00$	+	$1.64E+01$	$2.75E+00$	+	**$5.49E-01$**	**$3.17E-01$**
$F5$	$2.04E+01$	$1.27E-02$	+	$2.03E+01$	$2.66E-02$	+	$2.04E+01$	$4.01E-02$	+	$2.03E+01$	$2.53E-02$	+	$2.04E+01$	$4.05E-02$	+	**$2.00E+01$**	**$5.06E-04$**	−	$2.01E+01$	$1.41E-02$
$F6$	$1.66E-01$	$1.45E-02$	+	$1.37E-01$	$1.17E-02$	+	$1.48E-01$	$5.65E-03$	+	$1.45E-01$	$1.05E-02$	+	$1.36E-01$	$6.54E-03$	+	$1.13E-01$	$1.50E-02$	+	**$9.94E-02$**	**$8.34E-03$**
$F7$	$2.25E-03$	$2.27E-03$	+	$1.76E-07$	$2.79E-07$	+	$1.13E-04$	$9.70E-05$	+	$6.03E-08$	$8.19E-08$	+	$1.41E-05$	$1.19E-05$	+	**0**	**0**	NA	**0**	**0**
$F8$	**0**	**0**	NA	**0**	**0**	NA	**0**	**0**	NA	**0**	**0**	NA	**0**	**0**	NA	**0**	**0**	NA	**0**	**0**
$F9$	$9.24E+01$	$1.17E+01$	+	$7.08E+01$	$1.34E+01$	+	$7.92E+01$	$9.03E+00$	+	$8.14E+01$	$7.53E+00$	+	$5.35E+01$	$4.92E+00$	+	$6.28E+01$	$8.47E+00$	+	**$4.83E+01$**	**$3.87E+00$**
$F10$	$9.81E-03$	$1.16E-02$	−	$4.24E-01$	$4.11E-01$	+	$1.47E+00$	$1.22E-01$	+	$8.32E-01$	$8.50E-01$	+	**0**	**0**	−	$2.77E-01$	$2.61E-02$	+	$1.87E-01$	$2.66E-02$
$F11$	$3.19E+03$	$1.93E+02$	+	$2.00E+03$	$1.62E+02$	+	$2.43E+03$	$1.70E+02$	+	$1.95E+03$	$2.04E+02$	+	$2.13E+03$	$7.28E+01$	+	$1.85E+03$	$2.99E+02$	+	**$1.27E+03$**	**$1.27E+02$**
$F12$	$6.25E-03$	$9.56E-04$	+	$3.44E-03$	$5.09E-04$	+	$5.12E-03$	$7.70E-04$	+	$3.04E-03$	$4.60E-04$	+	$4.61E-03$	$3.40E-04$	+	$1.63E-03$	$2.74E-04$	+	**$1.28E-03$**	**$1.45E-04$**
$F13$	$3.46E-03$	$2.44E-04$	+	$2.29E-03$	$2.74E-04$	+	$2.78E-03$	$1.65E-04$	+	$2.39E-03$	$1.71E-04$	+	$2.62E-03$	$2.52E-04$	+	$1.78E-03$	$2.95E-04$	+	**$1.60E-03$**	**$1.73E-04$**
$F14$	$2.39E-03$	$2.12E-04$	+	$1.83E-03$	$1.27E-04$	+	$2.04E-03$	$1.93E-04$	+	$1.79E-03$	$9.21E-05$	+	$2.22E-03$	$1.41E-04$	+	$1.50E-03$	$1.17E-04$	−	**$1.47E-03$**	**$8.57E-05$**
$F15$	$1.65E+01$	$1.05E+00$	+	$8.24E+00$	$1.93E+00$	+	$1.28E+01$	$1.58E+00$	+	$9.61E+00$	$1.11E+00$	+	$9.14E+00$	$1.06E+00$	+	$7.78E+00$	$1.70E+00$	+	**$4.54E+00$**	**$6.43E-01$**
$F16$	$1.08E+01$	$1.56E-01$	+	$1.01E+01$	$2.22E-01$	+	$1.07E+01$	$2.94E-01$	+	$1.04E+01$	$3.34E-01$	+	$1.03E+01$	$2.05E-01$	+	$9.78E+00$	$4.65E-01$	+	**$8.30E+00$**	**$4.50E-01$**

Table 3: Results of HABC and other algorithms for 50D.

Function	ABC			BSABC			dABC			GABC			IABC			MABC			HABC	
	Mean	Std.	Sign.	Mean	Std.	Sign.	Mean	Std.	Sign.	Mean	Std.	Sign.	Mean	Std.	Sign.	Mean	Std.	Sign.	Mean	Std.
$F1$	$4.64E+07$	$3.68E+06$	+	$1.60E+07$	$3.17E+06$	+	$2.89E+07$	$3.65E+06$	+	$1.50E+07$	$3.49E+06$	+	$2.00E+07$	$2.92E+06$	+	$1.60E+07$	$3.49E+06$	+	**$9.39E+06$**	**$1.05E+06$**
$F2$	$1.15E+04$	$7.19E+03$	+	$1.67E+00$	$8.86E-01$	+	$1.90E+00$	$7.53E-01$	+	$7.48E-01$	$3.63E-01$	+	$3.56E+00$	$3.36E+00$	+	**$2.16E-02$**	**$2.08E-02$**	−	$3.36E-01$	$1.82E-01$
$F3$	$6.54E+00$	$2.87E+00$	+	$6.43E+00$	$1.88E+00$	+	$6.69E+00$	$2.05E+00$	+	$5.89E+00$	$1.12E+00$	+	$4.96E+00$	$2.94E+00$	=	$7.10E+00$	$2.68E+00$	+	**$2.80E+00$**	**$7.40E-01$**
$F4$	$1.30E+02$	$6.80E+00$	+	$4.21E+01$	$2.34E+01$	+	$6.65E+01$	$2.57E+01$	+	$4.00E+01$	$2.65E+01$	+	$4.74E+01$	$2.10E+01$	+	$6.25E+01$	$2.67E+01$	+	**$8.60E+00$**	**$2.48E+00$**
$F5$	$2.06E+01$	$2.77E-02$	+	$2.05E+01$	$2.37E-02$	+	$2.06E+01$	$2.91E-02$	+	$2.05E+01$	$3.22E-02$	+	$2.05E+01$	$3.33E-02$	+	**$2.00E+01$**	**$1.50E-04$**	−	$2.01E+01$	$1.61E-02$
$F6$	$3.80E-01$	$1.40E-02$	+	$3.23E-01$	$2.10E-02$	+	$3.40E-01$	$1.04E-02$	+	$3.29E-01$	$1.74E-02$	+	$3.10E-01$	$2.35E-02$	+	$2.70E-01$	$1.48E-02$	+	**$2.46E-01$**	**$2.01E-02$**
$F7$	$2.70E-02$	$9.14E-03$	+	$3.21E-03$	$2.46E-03$	+	$8.32E-03$	$6.07E-03$	+	$1.45E-03$	$1.17E-03$	+	$6.92E-03$	$3.28E-03$	+	$6.33E-04$	$4.97E-04$	+	**$1.28E-05$**	**$1.14E-05$**
$F8$	**0**	**0**	NA	**0**	**0**	NA	**0**	**0**	NA	$4.05E-03$	$7.58E-03$	+	**0**	**0**	NA	**0**	**0**	NA	**0**	**0**
$F9$	$2.25E+02$	$1.35E+01$	+	$1.87E+02$	$1.52E+01$	+	$1.96E+02$	$2.63E+01$	+	$2.05E+02$	$1.87E+01$	+	$1.49E+02$	$1.27E+01$	+	$1.67E+02$	$5.97E+00$	+	**$1.20E+02$**	**$3.13E+00$**
$F10$	$1.51E-01$	$1.48E-01$	−	$3.60E+00$	$7.64E-01$	+	$6.29E+00$	$2.87E+00$	+	$4.90E+00$	$1.26E+00$	+	**$1.67E-06$**	**$4.20E-06$**	−	$2.52E+00$	$3.37E-01$	+	$8.08E-01$	$5.23E-01$
$F11$	$7.28E+03$	$2.56E+02$	+	$4.75E+03$	$1.37E+02$	+	$5.60E+03$	$3.45E+02$	+	$4.83E+03$	$3.74E+02$	+	$5.26E+03$	$1.64E+02$	+	$3.78E+03$	$4.24E+02$	−	**$3.30E+03$**	**$2.26E+02$**
$F12$	$8.85E-03$	$2.76E-04$	+	$4.54E-03$	$4.97E-04$	+	$6.31E-03$	$5.06E-04$	+	$4.04E-03$	$6.80E-04$	+	$5.72E-03$	$7.76E-04$	+	$1.96E-03$	$1.04E-04$	+	**$1.23E-03$**	**$4.42E-05$**
$F13$	$4.20E-03$	$3.27E-04$	+	$3.20E-03$	$3.16E-04$	+	$3.47E-03$	$2.24E-04$	+	$3.25E-03$	$2.67E-04$	+	$3.40E-03$	$2.53E-04$	+	$2.84E-03$	$2.10E-04$	+	**$2.14E-03$**	**$1.57E-04$**
$F14$	$2.91E-03$	$1.78E-04$	+	$2.33E-03$	$1.67E-04$	+	$2.41E-03$	$1.51E-04$	+	$2.30E-03$	$1.91E-04$	+	$2.68E-03$	$2.41E-04$	+	$2.18E-03$	$1.54E-04$	+	**$1.64E-03$**	**$5.98E-05$**
$F15$	$3.82E+01$	$2.29E+00$	+	$2.53E+01$	$1.44E+00$	+	$3.21E+01$	$2.45E+00$	+	$2.61E+01$	$3.30E+00$	+	$2.43E+01$	$1.38E+00$	+	$1.82E+01$	$3.01E+00$	+	**$1.25E+01$**	**$7.77E-01$**
$F16$	$1.98E+01$	$2.31E-01$	+	$1.91E+01$	$4.44E-01$	+	$1.95E+01$	$3.61E-01$	+	$1.93E+01$	$1.94E-01$	+	$1.90E+01$	$3.48E-01$	+	$1.82E+01$	$6.13E-01$	+	**$1.71E+01$**	**$2.63E-01$**

Table 4: Time complexity of the HABC and other algorithms for 10D.

Function	Acceptance	ABC	BSABC	dABC	GABC	IABC	MABC	HABC
1	2E + 5	33.834	2.408	8.757	3.138	6.615	18.846	**1.885**
2	3E − 2	40.722	4.321	9.534	4.371	13.097	**1.788**	4.271
3	7E − 2	37.691	10.426	18.707	12.005	9.208	8.260	**5.809**
4	9E − 2	31.858	2.693	10.626	3.102	9.720	2.500	**1.666**
5	1E + 1	32.927	10.110	24.425	21.747	8.149	24.291	**7.486**
6	1E − 2	55.436	35.492	46.767	21.747	51.064	21.471	**21.491**
7	1E − 3	32.420	**6.812**	23.474	13.248	8.243	7.310	9.418
8	1E − 10	1.013	1.139	1.478	1.370	0.861	2.167	**0.497**
9	5E + 0	31.994	13.734	23.098	22.442	6.418	11.457	**3.159**
10	1E − 1	2.488	5.108	10.637	11.348	**0.715**	11.893	11.053
11	1E + 2	33.579	12.927	28.197	26.390	16.641	22.121	**10.235**
12	2E − 3	35.915	16.459	26.651	16.741	23.218	**1.594**	1.944
13	1E − 3	28.187	13.513	23.821	24.465	29.591	12.792	**10.736**
14	1E − 3	29.062	26.741	19.538	23.257	30.848	25.700	**11.597**
15	8E − 1	30.249	14.986	18.243	14.007	12.919	10.799	**8.261**
16	2E + 0	30.795	14.643	23.071	16.134	21.005	12.926	**10.204**

(IABC) [28], and memetic ABC [29] (MABC). In HABC algorithm, we set 3 as the value of $N_{s\max}$. For all algorithms, the colony size is set to be 50, and the stopping criterion was to run for up to 10000D FEs. Moreover, search space for all functions are $[-100, 100]^D$. Each experiment is repeated 51 times independently with random seeds, and then we record the mean and standard deviations of benchmark functions.

4.3. Comparisons between HABC and ABC Variants. All algorithms are coded in MATLAB 7.9.0, and all experiments were running on a Windows XP operation system with an Intel Pentium Dual-Core CPU E5300 2.6 GHZ and 2 GB RAM. Experimental results are shown in Tables 1–3, which also include the results by Wilcoxon signed-rank test.

It can be seen that all algorithms cannot get the global optimum value of $F1$ function and have equal performance for $F8$ function. From Table 1, our proposed HABC is better than the other algorithms for $F3$, $F4$, $F5$, $F6$, $F7$, $F9$, $F10$, and $F11$ functions, and the MABC algorithm is slightly better than the other algorithms for $F2$, $F6$, $F7$, and $F12$ functions. According to Table 2, our proposed HABC algorithm shows the best performance on most of the functions, except for functions $F10$ and $F5$. From Table 3, our proposed HABC also has better performance than the other algorithms on most of the benchmark functions, except for $F2$, $F5$, and $F10$ functions. The MABC algorithm is slightly better than the other algorithms in $F2$ and $F5$ functions, and the IABC algorithm is slightly better than the other algorithms in $F10$ function.

In the last column of Tables 1–3, we give the Wilcoxon signed-rank test results. A value of "+" in the table indicates that HABC algorithm is statically superior to the compared algorithm, whereas a "−" indicates that the HABC algorithm is statistically worse than the compared algorithm. A value of "=" indicates that the HABC algorithm is statistically equivalent with the compared algorithm and a value of "NA" indicates that two algorithms are statistically indistinguishable. We can conclude that the HABC algorithm is statistically better than the other algorithms on most of the benchmark functions.

4.4. Time Complexity Analysis. The improvement of computation precision in HABC algorithm usually needs the sacrifice of time complexity. Because the limit of neighbor searching in employed bees' stage is not constant, in order to analyze time complexity properly, we calculated the computational complex of each algorithm by recording the average time to reach the given precision. All algorithms ran 51 times independently with random seeds. The mean time is measured in seconds. Results are listed in Tables 4–6, and the best is shown in bold font.

It can be seen from Tables 4–6 that HABC can find the global optimal solution with less time on most of the benchmark functions, because of its faster converge ability.

5. Conclusion

In this paper, hybrid ABC (HABC) is proposed by introducing the chemotactic behavior of Bacterial Foraging Optimization into employed bees and adopting the principle of moving the particles toward the best solutions in PSO to improve the global search ability of onlooker bees. Experiment result shows that HABC algorithm has better solution accuracy and higher evolution speed and reaches the global optimal solution with fewer time, compared with the current reported standard ABC, best-so-far ABC, directed ABC, Gaussian ABC, improved ABC, and memetic ABC.

Table 5: Time complexity of the HABC and other algorithms for 30D.

Function	Acceptance	ABC	BSABC	dABC	GABC	IABC	MABC	HABC
1	$2E+7$	33.507	2.650	29.906	5.003	9.133	10.397	**1.036**
2	$5E-2$	41.292	15.957	24.061	20.076	12.157	13.907	**11.375**
3	$1E+0$	30.586	**2.366**	10.839	10.026	17.048	6.866	3.285
4	$3E+1$	33.474	18.456	15.021	19.598	16.970	18.874	**11.977**
5	$2E+1$	34.488	28.730	29.090	34.081	31.562	**7.014**	9.018
6	$2E-1$	7.803	5.079	3.875	3.778	4.009	6.588	**3.740**
7	$1E-4$	33.431	6.452	22.666	8.930	10.027	11.655	**3.463**
8	$1E-10$	12.926	4.233	8.706	6.034	2.743	8.019	**1.677**
9	$8E+1$	33.747	12.545	23.286	28.586	15.180	12.862	**3.247**
10	$2E+0$	6.943	8.417	20.827	23.533	14.992	15.113	**2.073**
11	$2E+3$	34.934	20.299	33.104	16.409	27.451	17.469	**14.429**
12	$5E-3$	42.940	4.817	24.498	4.838	19.227	1.641	**1.002**
13	$3E-3$	28.561	1.900	17.227	2.216	10.758	**1.150**	1.583
14	$2E-3$	32.531	11.278	11.826	9.853	27.508	5.909	**3.640**
15	$1E+1$	36.000	20.175	27.876	25.809	13.963	8.852	**7.548**
16	$1E+1$	34.890	28.978	29.000	33.354	28.754	**9.084**	12.534

Table 6: Time complexity of the HABC and other algorithms for 50D.

Function	Acceptance	ABC	BSABC	dABC	GABC	IABC	MABC	HABC
1	$3E+7$	38.114	14.070	28.462	6.069	4.553	7.693	**2.140**
2	$4E+0$	42.527	28.695	22.211	**9.943**	20.540	13.990	10.660
3	$7E+0$	33.428	8.440	23.177	14.614	11.959	21.812	**7.322**
4	$7E+1$	35.891	20.545	30.861	18.126	24.237	29.541	**14.176**
5	$2E+1$	36.314	30.805	32.358	36.157	31.126	27.856	**20.181**
6	$4E-1$	19.899	9.505	11.194	9.218	7.051	13.054	**5.060**
7	$8E-3$	31.095	16.599	25.868	15.362	19.242	20.779	**12.879**
8	$1E-10$	28.889	9.694	25.136	31.707	9.869	22.936	**4.583**
9	$2E+2$	31.607	13.149	24.496	26.831	15.387	10.370	**3.228**
10	$6E+0$	**3.103**	11.702	30.879	33.913	3.529	19.146	20.840
11	$5E+3$	35.728	15.498	35.817	36.229	34.458	9.318	**7.696**
12	$7E-3$	47.806	2.850	13.292	2.934	12.257	2.287	**2.822**
13	$4E-3$	27.932	1.798	11.401	2.694	4.320	1.872	**1.411**
14	$3E-3$	14.669	2.801	4.350	2.633	8.051	3.142	**1.924**
15	$3E+1$	35.294	11.286	35.595	9.801	5.398	8.149	**5.073**
16	$2E+1$	14.752	5.442	8.805	6.890	3.984	4.503	**2.757**

In our future work, we will find a better way for searching for the best $N_{s\max}$ to improve the performance of HABC.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This research is supported in part by the Program for New Century Excellent Talents in University (NCET-12-0881), National Science and Technology Support Program of China (no. 2015 BAD17B02), China Agriculture Research System (CARS-49), and the Fundamental Research Funds for the Central Universities (JUSRP51410B).

References

[1] K. S. Tang, K. F. Man, S. Kwong, and Q. He, "Genetic algorithms and their applications," *IEEE Signal Processing Magazine*, vol. 13, no. 6, pp. 22–37, 1996.

[2] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: optimization by a colony of cooperating agents," *IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetics*, vol. 26, no. 1, pp. 29–41, 1996.

[3] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural*

Networks, vol. 4, pp. 1942–1948, IEEE, Perth, Australia, December 1995.

[4] X.-L. Li, Z.-J. Shao, and J.-X. Qian, "Optimizing method based on autonomous animats: fish-swarm Algorithm," *System Engineering Theory and Practice*, vol. 22, no. 11, pp. 32–38, 2002.

[5] X.-S. Yang, "Firefly algorithms for multimodal optimization," in *Stochastic Algorithms: Foundations and Applications*, vol. 5792, pp. 169–178, Springer, Berlin, Germany, 2009.

[6] I. Fister, M. Perc, S. M. Kamal, and I. Fister, "A review of chaos-based firefly algorithms: perspectives and research challenges," *Applied Mathematics and Computation*, vol. 252, pp. 155–165, 2015.

[7] D. Karaboga, "An idea based on honey bee swarm for numerical optimization (vol. 200)," Tech. Rep. tr06, Computer Engineering Department, Engineering Faculty, Erciyes University, Kayseri, Turkey, 2005.

[8] A. A. El-Fergany and A. Y. Abdelaziz, "Capacitor placement for net saving maximization and system stability enhancement in distribution networks using artificial bee colony-based approach," *International Journal of Electrical Power & Energy Systems*, vol. 54, pp. 235–243, 2014.

[9] O. Bulut and M. F. Tasgetiren, "An artificial bee colony algorithm for the economic lot scheduling problem," *International Journal of Production Research*, vol. 52, no. 4, pp. 1150–1170, 2014.

[10] M. Ma, J. Liang, M. Guo, Y. Fan, and Y. Yin, "SAR image segmentation based on artificial bee colony algorithm," *Applied Soft Computing Journal*, vol. 11, no. 8, pp. 5205–5214, 2011.

[11] A. Draa and A. Bouaziz, "An artificial bee colony algorithm for image contrast enhancement," *Swarm and Evolutionary Computation*, vol. 16, pp. 69–84, 2014.

[12] A. Singh, "An artificial bee colony algorithm for the leaf-constrained minimum spanning tree problem," *Applied Soft Computing Journal*, vol. 9, no. 2, pp. 625–631, 2009.

[13] J. C. Bansal, H. Sharma, K. V. Arya, K. Deep, and M. Pant, "Self-adaptive artificial bee colony," *Optimization*, vol. 63, no. 10, pp. 1513–1532, 2014.

[14] H. Wang, Z. Wu, S. Rahnamayan, H. Sun, Y. Liu, and J.-S. Pan, "Multi-strategy ensemble artificial bee colony algorithm," *Information Sciences*, vol. 279, pp. 587–603, 2014.

[15] F. Kang, J. Li, and Z. Ma, "Rosenbrock artificial bee colony algorithm for accurate global optimization of numerical functions," *Information Sciences*, vol. 181, no. 16, pp. 3508–3531, 2011.

[16] B. Alatas, "Chaotic bee colony algorithms for global numerical optimization," *Expert Systems with Applications*, vol. 37, no. 8, pp. 5682–5687, 2010.

[17] Y. Xiang, Y. Peng, Y. Zhong, Z. Chen, X. Lu, and X. Zhong, "A particle swarm inspired multi-elitist artificial bee colony algorithm for real-parameter optimization," *Computational Optimization and Applications*, vol. 57, no. 2, pp. 493–516, 2014.

[18] X. Li and M. Yin, "Self-adaptive constrained artificial bee colony for constrained numerical optimization," *Neural Computing and Applications*, vol. 24, no. 3-4, pp. 723–734, 2014.

[19] K. M. Passino, "Biomimicry of bacterial foraging for distributed optimization and control," *IEEE Control Systems Magazine*, vol. 22, no. 3, pp. 52–67, 2002.

[20] J. Kennedy, "Particle swarm optimization," in *Encyclopedia of Machine Learning*, pp. 760–766, Springer, New York, NY, USA, 2010.

[21] R. C. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proceedings of the 6th International Symposium on Micro Machine and Human Science*, vol. 1, pp. 39–43, IEEE, Nagoya, Japan, October 1995.

[22] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," in *Proceedings of the IEEE International Conference on Evolutionary Computation Proceedings and the IEEE World Congress on Computational Intelligence*, pp. 69–73, IEEE, Anchorage, Alaska, USA, May 1998.

[23] Y. Shi and R. C. Eberhart, "Empirical study of particle swarm optimization," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '99)*, vol. 3, Washington, DC, USA, July 1999.

[24] J. J. Liang, B. Y. Qu, and P. N. Suganthan, "Problem definitions and evaluation criteria for the CEC 2014 special session and competition on single objective real-parameter numerical optimization," Tech. Rep., Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou, China; Nanyang Technological University, Singapore, 2013.

[25] A. Banharnsakun, T. Achalakul, and B. Sirinaovakul, "The best-so-far selection in artificial bee colony algorithm," *Applied Soft Computing Journal*, vol. 11, no. 2, pp. 2888–2901, 2011.

[26] M. S. Kiran and O. Findik, "A directed artificial bee colony algorithm," *Applied Soft Computing Journal*, vol. 26, pp. 454–462, 2015.

[27] L. Dos Santos Coelho and P. Alotto, "Gaussian artificial bee colony algorithm approach applied to Loney's solenoid benchmark problem," *IEEE Transactions on Magnetics*, vol. 47, no. 5, pp. 1326–1329, 2011.

[28] W. Gao and S. Liu, "Improved artificial bee colony algorithm for global optimization," *Information Processing Letters*, vol. 111, no. 17, pp. 871–882, 2011.

[29] I. Fister, I. Fister Jr., J. Brest, and V. Žumer, "Memetic artificial bee colony algorithm for large-scale global optimization," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '12)*, pp. 1–8, Brisbane, Australia, June 2012.