Hindawi Publishing Corporation
Computational Intelligence and Neuroscience
Volume 2016, Article ID 8085953, 13 pages
http://dx.doi.org/10.1155/2016/8085953

Research Article

An Enhanced Artificial Bee Colony Algorithm with Solution Acceptance Rule and Probabilistic Multisearch

Alkın Yurtkuran and Erdal Emel

Department of Industrial Engineering, Uludag University, Görükle Campus, 16059 Bursa, Turkey

Correspondence should be addressed to Alkın Yurtkuran; alkin@uludag.edu.tr

Received 3 July 2015; Accepted 9 September 2015

Academic Editor: Ezequiel López-Rubio

The artificial bee colony (ABC) algorithm is a popular swarm based technique, which is inspired from the intelligent foraging behavior of honeybee swarms. This paper proposes a new variant of ABC algorithm, namely, enhanced ABC with solution acceptance rule and probabilistic multisearch (ABC-SA) to address global optimization problems. A new solution acceptance rule is proposed where, instead of greedy selection between old solution and new candidate solution, worse candidate solutions have a probability to be accepted. Additionally, the acceptance probability of worse candidates is nonlinearly decreased throughout the search process adaptively. Moreover, in order to improve the performance of the ABC and balance the intensification and diversification, a probabilistic multisearch strategy is presented. Three different search equations with distinctive characters are employed using predetermined search probabilities. By implementing a new solution acceptance rule and a probabilistic multisearch approach, the intensification and diversification performance of the ABC algorithm is improved. The proposed algorithm has been tested on well-known benchmark functions of varying dimensions by comparing against novel ABC variants, as well as several recent state-of-the-art algorithms. Computational results show that the proposed ABC-SA outperforms other ABC variants and is superior to state-of-the-art algorithms proposed in the literature.

1. Introduction

Optimization techniques play an important role in the field of science and engineering. Over the last five decades, numerous algorithms have been developed to solve complex optimization algorithms. Since more and more present-day problems turn out to be nonlinear, multimodal, discontinuous, or dynamic in nature, derivative-free, nonexact solution methods attract ever-increasing attention. Evolutionary biology or swarm behaviors inspired most of these methods. There have been several classes of algorithms proposed in this evolutionary or swarm intelligence framework including genetic algorithms [1, 2], memetic algorithms [3], differential evolution (DE) [4], ant colony optimization (ACO) [5], particle swarm optimization (PSO) [6], artificial bee colony algorithm (ABC) [7], cuckoo search [8], and firefly algorithm [9].

The ABC is a biologically inspired population-based metaheuristic algorithm that mimics the foraging behavior of honeybee swarms [7]. Due to its simplicity and ease of application, the ABC has been widely used to solve both continuous and discrete optimization problems since its introduction [10]. It has been shown that ABC tends to suffer poor intensification performance on complex problems [11–13]. To improve the intensification performance of ABC, many researchers have focused on the search rules as they control the tradeoff between diversification and intensification. Diversification means the ability of an algorithm to search for unvisited points in the search region, whereas intensification is the process of refining those points within the neighborhood of previously visited locations to improve solution quality. Various new search strategies, mostly inspired from PSO and DE, have been proposed in the literature. Zhu and Kwong [14] proposed a global best guided ABC, which utilizes the global best individual's information within the search equation similar to PSO. Gao et al. [15] introduced another variant of global best ABC. Inspired by DE, Gao and Liu [13] introduced a modified version of the ABC in which ABC/Best/1 and ABC/Rand/1 were employed as local search equations. Kang et al. [16] described the Rosenbrock ABC, which combines Rosenbrock's rotational method with the original ABC. To improve diversification, Alatas [11] employed chaotic maps for initialization and chaotic searches within a search strategy. Akay and Karaboga [17] introduced a modified version of the ABC in which frequency of perturbation is controlled adaptively and the ratio of variance operator was introduced. Liao et al. [18] proposed a detailed experimental analysis and comparison of an ABC variant with different search equations. Gao et al. [19] introduced two new search equations for onlooker and employed bee phases and a new robust comparison technique for candidate solutions. ABC. Qiu et al. [20] were inspired from DE/current-to-best/1 strategy in DE algorithm and proposed a modified ABC. Banitalebi et al. [21] proposed an enhanced compact ABC, which did not store the actual population of candidate solution; instead their approach employed probabilistic representation. Wang et al. [22] presented multistrategy ABC, in which a pool of different search strategies was constructed and various search strategies were used during the search process. Gao et al. [23] introduced a bare bones ABC with parameter adaptation and fitness-based neighborhood to improve the intensification performance of standard ABC. Ma et al. [24] reduced the redundant search moves and maintained the diversity of the swarm by introducing hybrid ABC with life cycle and social learning. Furthermore, ABC has been successfully applied to solve various types of optimization problems, such as production scheduling [25, 26], vehicle routing [27], location-allocation problem [28], image segmentation [29], wireless sensor network routing [30], leaf-constrained minimum spanning tree problem [31], clustering problem [32], fuel management optimization [33], and many others [34–36]. Readers can refer to Karaboga et al. [10] for an extensive literature review of the ABC and its applications.

This study presents an enhanced ABC with solution acceptance rule and probabilistic multisearch (ABC-SA) in order to solve global optimization problems efficiently. In ABC-SA, three search mechanisms with different diversification and intensification characteristics are employed. Moreover, search mechanism selection probabilities p_{s1}, p_{s2}, and p_{s3} are introduced to control the balance between diversification and intensification. In our proposed approach, a search mechanism is established using the selection probabilities to generate a new neighbor solution from the current one. Additionally, a solution acceptance rule is implemented, in which not only better solutions but also worse solutions may be accepted by using a probability function. A nonlinearly decreasing acceptance probability function is employed, thus allowing worse solutions to be more likely accepted in the early phases of the search. Therefore, ABC-SA algorithm explores the search space more widespread, especially in the early phases of the search process. By using solution acceptance rule and implementing different search mechanisms of contrasting nature, ABC-SA balances the trade-off between diversification and intensification efficiently. The proposed approach is tested on six different benchmark functions with varying dimensions and compared to novel ABC, PSO, and DE variants. Computational results reveal that ABC-SA outperforms competitor algorithms in terms of solution quality.

The main contributions of the proposed study are as follows:

(i) Three different search mechanisms are employed with varying diversification and intensification abilities. Probabilistic multisearch with predetermined probability values are employed to determine the search mechanism to be used to generate candidate solutions. Therefore, ABC-SA explores and exploits the search space efficiently.

(ii) Instead of a greedy selection, a new candidate solution acceptance rule is integrated, where a worse solution may have a chance to be accepted as new solution. By the help of this new acceptance rule, ABC-SA achieves better diversification performance, specifically in the early phases of the search.

The remainder of this paper is structured as follows: Section 2 presents the traditional ABC; Section 3 introduces the proposed framework; the instances, parameter settings, and computational results are presented in Section 4 and finally Section 5 concludes the paper.

2. Artificial Bee Colony Algorithm

The ABC has inspired from the organizational nature and foraging behavior of honeybee swarms. In the ABC algorithm, the bee colony comprises three kinds of bees: employed bees, onlooker bees, and scout bees. Each bee has a specialized task in the colony to maximize the nectar amount that is stored in the hive. In ABC, each food source is placed in the D-dimensional search space and represents a potential solution to the optimization problem. The amount of nectar in the food source is assumed to be the fitness value of a food source. Generally, the number of employed and onlooker bees is the same and equal to the number of food sources.

Each employed bee belongs to a food source and is responsible for mining the corresponding food source. Then, employed bees pass the nectar information to onlooker bees in the "dance area." Onlooker bees wait in the hive and select a food source to mine based on the information coming from the employed bees. Here, more beneficial food sources will have higher selection probabilities to be selected by onlooker bees. In ABC, in order to decide if a food source is abandoned or not, trial counters and a predetermined limit parameter are used. If a solution represented by a food source does not improve during a number of trials (limit), the food source is abandoned. When the food source is abandoned, the corresponding employed bee will become a scout bee and randomly generate a new food source and replace it with the abandoned one.

The ABC algorithm consists of four main steps: initialization, employed bee phase, onlooker bee phase, and scout bee phase. After the initialization step, the other three main steps of the algorithm are carried out repeatedly in a loop until the termination condition is met. The main steps of the ABC algorithm are as follows.

Step 1 (initialization). In the initialization step, the ABC generates a randomly distributed population of SN solutions (food sources), where SN also denotes the number of employed or onlooker bees. Let $x_i = \{x_{i,1}, x_{i,2}, \ldots, x_{i,D}\}$ represent the ith food source, where D is the problem size. Each food source is generated within the limited range of jth index by

$$x_{i,j} = x_j^{\min} + \varphi_{i,j}\left(x_j^{\max} - x_j^{\min}\right), \quad (1)$$

where $i = 1, 2, \ldots, SN$, $j = 1, 2, \ldots, D$, $\varphi_{i,j}$ is a uniformly distributed random real number in $[0, 1]$, and $x_j^{\min}$ and $x_j^{\max}$ are the lower and upper bounds for the dimension j, respectively. Moreover, a trial counter for each food source is initialized.

Step 2 (employed bee phase). In the employed bee phase, each employed bee visits a food source and generates a neighboring food source in the vicinity of the selected food source. Employed bees search a new solution, v_i, by performing a local search around each food source $i = 1, 2, \ldots, SN$ as follows:

$$v_{i,j} = x_{i,j} + \phi\left(x_{i,j} - x_{r1,j}\right), \quad (2)$$

where j is a randomly selected index $j \in \{1, 2, \ldots, D\}$ and $r1 \in \{1, 2, \ldots, SN\}$ is a randomly chosen food source that is not equal to i; that is, $(r1 \neq i)$. ϕ is a random number within the range $[-1, 1]$ generated specifically for each i and j combination. A greedy selection is applied between x_i and v_i by selecting the better one.

Step 3 (onlooker bee phase). Unlike the employed bees, onlooker bees select a food source depending on the probability value p, which is determined by nectar amount associated with that food source. The value of p_i is calculated for ith food source as follows:

$$p_i = \frac{\text{fit}_i}{\sum_{j=1}^{SN} \text{fit}_j}, \quad (3)$$

$$\text{fit}_i = \begin{cases} \dfrac{1}{1 + f_i}, & f_i \geq 0 \\ 1 + \text{abs}\left(f_i\right), & f_i < 0, \end{cases} \quad (4)$$

where fit_i is the fitness value of solution i and calculated as in (4) for minimization problems. Different fitness functions are employed for maximization problems. By using this type of roulette wheel based probabilistic selection, better food sources will more likely be visited by onlooker bees. Therefore, onlooker bees try to find new candidate food sources around good solutions. Once the onlooker bee chooses the food source, it generates a new solution using (2). Similar to the employed bee phase, a greedy selection is carried out between x_i and v_i.

Step 4 (scout bee phase). A trial counter is associated with each food source, which depicts the number of tries that the food source cannot be improved. If a food source cannot be improved for a predetermined number of tries (limit) during the onlooker and employed bee phases, then the employed bee associated with that food source becomes a scout bee. Then, the scout bee finds a new food source using (1). By implementing the scout bee phase, the ABC algorithm easily escapes from minimums and improves its diversification performance.

It should be noted that, in the employed bee phase, a local search is applied to each food source, whereas in the onlooker bee phase better food sources will more likely be updated. Therefore, in ABC algorithm, the employed bee phase is responsible for diversification whereas the onlooker bee phase is responsible of intensification. The flow chart of the ABC is given in Figure 1.

3. Proposed Framework

In this section, the proposed algorithm is described in detail. First, a solution acceptance rule is presented. Second, a novel probabilistic multisearch mechanism is proposed. Finally, the complete ABC-SA mechanism is given.

3.1. Solution Acceptance Rule

3.1. Solution Acceptance Rule. In order to strengthen the diversification ability of ABC-SA mechanism, a solution acceptance rule is proposed. Instead of greedy selection in both employed and onlooker bee phases, an acceptance probability is given to worse solutions. The main idea behind this acceptance probability is not to restrict the search moves to only better solutions. By accepting a worse solution, the procedure may escape from a local optimum and explore the search space effectively. In ABC-SA algorithm, if a worse solution is generated, it is accepted if the following condition holds:

$$r < p_a \quad (5)$$

$$p_a = p_o \frac{1 + \cos\left(\left(\text{iter}/\text{Max.iter}\right)\pi\right)}{2}, \quad (6)$$

where r is a random real number within $[0, 1]$, p_a is the acceptance probability, p_o denotes the initial probability, and iter and Max.iter represent the current iteration number and the maximum iteration number, respectively. According to (6), the acceptance probability p_a is nonlinearly decreased from p_0 to zero during the search process. As can be seen from (6), $p_a = 0$ when iter $=$ Max.iter and the range of p_a is $[0, p_0]$. A typical p_a graph is given in Figure 2 and Algorithm 1 presents the implementation of the solution acceptance rule. At this point, it is important to note that the trial counter is incremented, whether a worse candidate solution is accepted or not.

3.2. Probabilistic Multisearch Strategy. In standard ABC, a candidate solution is generated using the information of the parent food source with the guidance of the term

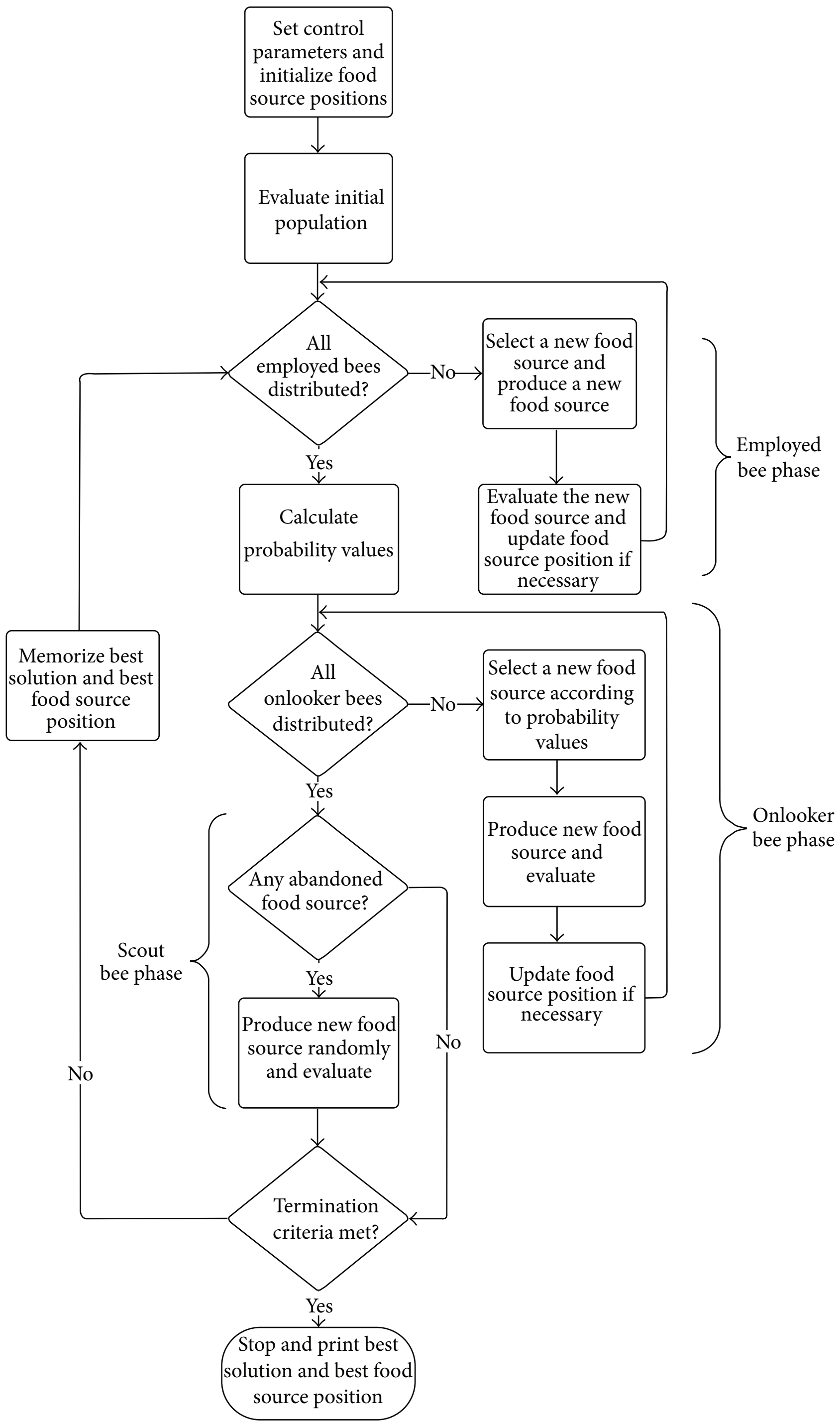
Set control parameters and initialize food source positions
Evaluate initial population
All employed bees distributed?
No
Select a new food source and produce a new food source
Evaluate the new food source and update food source position if necessary
Employed bee phase
Yes
Calculate probability values
Memorize best solution and best food source position
All onlooker bees distributed?
No
Select a new food source according to probability values
Produce new food source and evaluate
Update food source position if necessary
Onlooker bee phase
Yes
Any abandoned food source?
Scout bee phase
Yes
Produce new food source randomly and evaluate
No
No
Termination criteria met?
Yes
Stop and print best solution and best food source position

Figure 1: Flowchart of ABC.

$\phi_{i,j}(x_{i,j} - x_{r1,j})$ in (3). However, there is no guarantee that a better individual influences the candidate solution; therefore, it is possible to have a poor convergence speed and intensification performance. In fact, studying search equations is a trending topic to improve the ABC's performance. Recently, numerous search equations have been proposed, such as [13–16, 19, 20, 37, 38]. It is well known that the balance between diversification and intensification is the most critical part of any metaheuristic algorithm.

In ABC-SA approach, instead of employing a single search mechanism throughout the search process, a probabilistic multisearch mechanism with three different search rules is used. A probabilistic selection is employed using predefined probability parameters to select the search rule

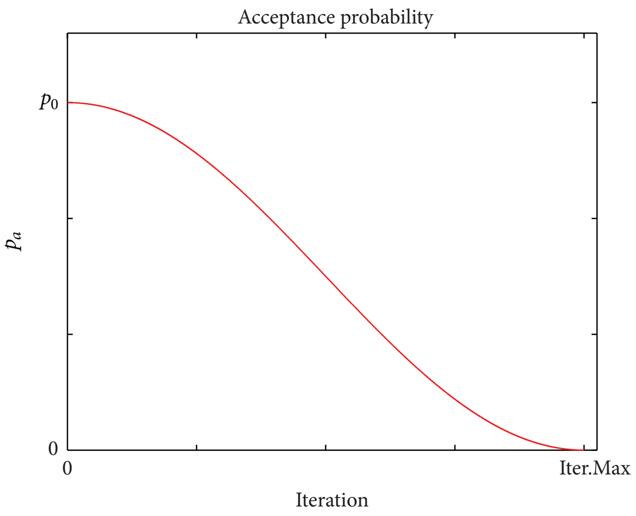

FIGURE 2: Acceptance probability curve.

```
Input: $x_i$, $v_i$, trial$_i$, $p_0$
Output: $x_i$
(1)  Evaluate $x_i$ and $v_i$. //Set $f(v_i)$ and $f(x_i)$
(2)  if $f(v_i) \leq f(x_i)$ then
(3)    set $x_i = v_i$
(4)    trial$_i = 0$
(5)  else
(6)    Calculate $p_a$. //Use (6).
(7)    Produce a random number $r$ within the range $[0, 1]$
(8)    if $r < p_a$ then
(9)      set $x_i = v_i$
(10)     trial$_i$ = trial$_i + 1$
(11)   else
(12)     trial$_i$ = trial$_i + 1$
(13)   end if
(14) end if
```

ALGORITHM 1: Solution acceptance rule.

within both employed and onlooker bee phases. The three search rules which were proposed by [7, 13, 14], respectively, are presented as follows:

$$v_{i,j} = x_{i,j} + \phi\left(x_{i,j} - x_{r1,j}\right), \tag{7}$$

$$v_{i,j} = x_{i,j} + \phi\left(x_{i,j} - x_{r1,j}\right) + \psi\left(x_{gbest,j} - x_{i,j}\right), \tag{8}$$

$$v_{i,j} = x_{lbest,j} + \phi\left(x_{i,j} - x_{r1,j}\right), \tag{9}$$

where i is a food source, j is a randomly selected index for all $i = 1, 2, \ldots, SN$, and $j \in \{1, 2, \ldots, D\}$, respectively. $r1$ is a randomly chosen food source where $r1 \neq i$. *gbest* stands for the global best solution, whereas *lbest* is the best solution in the current population. ϕ represents a real random number within the range of $[-1, 1]$ [7]. Finally, ψ is a real random number within the range of $[0, C]$ where C is a predetermined number [13].

Equation (7) is the original search rule, which was discussed in previous sections. Equation (8) is presented to improve the intensification capability of ABC. Equation (8) uses the information provided by the global best solution which is similar to PSO. In (9), *lbest* guides the search with the random effect of the term $\phi(x_{i,j} - x_{r1,j})$. Equation (7) has an explorative character, whereas (9) favors intensification. On the other hand, (8) explores the search space using the second term and exploits effectively by the third term. Therefore, (8) balances diversification and intensification performance. In summary, the proposed ABC-SA uses three different search rules to achieve a trade-off between diversification and intensification. In ABC-SA, search probabilities p_{s1}, p_{s2}, and p_{s3} are introduced such that $\sum_k p_{sk} = 1$ and $k = 1, 2, 3$ to select a search rule to be used in the employed and the onlooker bee phases. A roulette wheel method is employed with three cumulative ranges $0 \leq p_{s1} \leq P_{s1}$, $0 \leq p_{s2} < P_{s2} - P_{s1}$, and $0 \leq p_{s3} \leq P_{s3} - P_{s2}$ assigned to (7), (8), and (9), respectively, where $P_{s3} = 1$. Algorithm 2 shows the mechanism of probabilistic multisearch.

3.3. Proposed Approach. Algorithm 3 summarizes the ABC-SA framework. The novel parts of the ABC-SA mechanism are the probabilistic multisearch (Lines 9 and 19) and the solution acceptance rule (Lines 10 and 20) sections.

4. Computational Results

4.1. Test Instances. In literature, many test functions with different characters were used to test algorithms [11, 13, 15, 17, 19–21, 34, 37–40]. Unimodal functions have one local minimum as the global optimum. These functions are generally used to test the intensification ability of algorithms. Multimodal functions have one or more local optimums which may be the global optimum. Therefore, diversification behavior of algorithms is analyzed on multimodal instances. Separable functions can be written as sum of n functions with one variable, whereas nonseparable functions can not be reformulated as subfunctions. In this study, to analyze the performance of the proposed ABC-SA algorithm, 13 scalable benchmark functions with dimensions $D = 50$, $D = 100$, and $D = 200$ are used and listed in Table 1. They are Rosenbrock, Ackley, Rastrigin, Weierstrass, Schwefel 2.26, Shifted Sphere, Shifted Schwefel 1.2, Shifted Rosenbrock, Shifted Rastrigin, Step, Penalized 2, and Alpine. In Table 1, function label, name, formulation, type (UN: unimodal and nonseparable, MS: multimodal and separable, and MN: multimodal and nonseparable), range, and optimal values ($f(x^*)$) are given.

4.2. Parameters Settings. Parameter settings may have a great influence on the computational results. The ABC-SA mechanism has seven control parameters such as maximum iteration number (Max.iter), limit, population size (SN), p_{s1}, p_{s2}, ψ, and p_0. Maximum iteration number is the termination condition, and p_0 is the initial acceptance probability. First, Max.iter is set to 4,000, limit $= 0.2 \times D \times SN$, where D is the dimension of the problem [21], SN is taken to be 40 for $50D$ and $100D$ problems and 50 for $200D$ problems [40], and ψ is set to be a random real number within (0, 1.5) [14]. Then,

```
**Input**: $x_i$, $P_{s1}$, $P_{s2}$, $lbest$, $gbest$
**Output**: $v_i$
(1) Produce a random number $r$ within the range $[0, 1]$
(2) if $r =< P_{s1}$ then // $P_{s1}$ and $P_{s2}$ are cumulative probabilities
(3)     Produce a new neighbor solution $v_i$ by (7)
(4) elseif $r =< P_{s2}$ then
(5)     Produce a new neighbor solution $v_i$ by (8)
(6) Else // $r$ is in the range of $(P_{s2}, 1)$
(7)     Produce a new neighbor solution $v_i$ by (9)
(8) end if
```

Algorithm 2: Probabilistic multisearch.

```
(1)  set control parameters: $SN$, Max.iter, limit, $P_{s1}$, $P_{s2}$, $\psi$, $p_0$.
(2)  Generate initial population //Use (1).
(3)  Evaluate initial population //Calculate $f(x)$, record local and global best.
(4)  set iter = 1
(5)  for each food source $i$ do, Set trial$_i$ = 0 end for
(6)  do while iter ≤ Max.iter
(7)  //****EMPLOYED BEE PHASE****
(8)    for each food source $i$ do
(9)        Generate a neighbor solution $v_i$ from $x_i$ by Algorithm 2.
(10)       Make a selection between $x_i$ and $v_i$ by Algorithm 1.
(11)   end for
(12) //****ONLOOKER BEE PHASE****
(13)   Calculate cumulative probability values $P_i$ //Use (3).
(14)   set $t = 0$, $i = 1$
(15)   while $t <$ PopSize do
(16)       Produce random number $r$ within the range $[0, 1]$
(17)       if $r \leq P_i$ then
(18)           set $t = t + 1$
(19)           Generate a neighbor solution $v_i$ from $x_i$ by Algorithm 2.
(20)           Make a selection between $x_i$ and $v_i$ by Algorithm 1.
(21)       end if
(22)       set $i = i + 1$
(23)       if $i >$ PopSize then set $i = 1$ end if
(24)   end while
(25) //****SCOUT BEE PHASE****
(26)   set $i$ = index of maximum(trial) //Find the index that has the maximum trial value.
(27)   if limit =< trial$_i$ then
(28)       Replace $x_i$ with a new randomly generated solution //Use (1)
(29)       set trial$_i$ = 0
(30)   end if
(31)   Save necessary information //Record local and global best.
(32)   set iter = iter + 1
(33) end while
```

Algorithm 3: ABC-SA framework.

preliminary experiments were conducted with appropriate combinations of the following parameter values to determine the best settings:

p_0 = 0.25, 0.20, 0.15, 0.10, and 0.05,

p_{s1} = 0.2, 0.4, and 0.6,

p_{s2} = 0.2, 0.4, and 0.6,

p_{s3} = 0.2, 0.4, and 0.6.

From the results of the pilot studies, $p_0 = 0.10$, $p_{s1} = 0.20$, $p_{s2} = 0.60$, and $p_{s3} = 0.20$ settings achieved the best results. Therefore, these parameter settings are used for further experiments.

4.3. Comparison with ABC Variants. In this section, aforementioned ABC-SA is implemented and evaluated by benchmarking with other well-known ABC variants including

Table 1: Test functions used in experiments.

Label	Name	Formulation	Type	Range	$f(x^*)$
F1	Rosenbrock	$f_1(\vec{X}) = \sum_{i=1}^{D-1}\left[100\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2\right]$	UN	$[-2.048, 2.048]^D$	0
F2	Ackley	$f_2(\vec{X}) = -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)\right) + 20 + e$	MS	$[-32.768, 32.768]^D$	0
F3	Rastrigin	$f_3(\vec{X}) = \sum_{i=1}^{D}\left[x_i^2 - 10\cos(2\pi x_i) + 10\right]$	MS	$[-5.12, 5.12]^D$	0
F4	Griewank	$f_4(\vec{X}) = \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	MN	$[-600, 600]^D$	0
F5	Weierstrass	$f_5(\vec{X}) = \sum_{i=1}^{D}\left(\sum_{k=0}^{k_{\max}}\left[a^k\cos\left(2\pi b^k(x_i + 0.5)\right)\right]\right) - D\sum_{k=0}^{k_{\max}}\left[a^k\cos\left(2\pi b^k 0.5\right)\right], \quad a = 0.5,\ b = 3,\ k_{\max} = 20$	MS	$[-0.5, 0.5]^D$	0
F6	Schwefel 2.26	$f_6(\vec{X}) = 418.9829 \times D - \sum_{i=1}^{D} -x_i\sin\left(\sqrt{\lvert x_i\rvert}\right)$	MS	$[-500, 500]^D$	$-418.98 \times D$
F7	Shifted Sphere	$f_7(\vec{X}) = \sum_{i=1}^{D} z_i^2 - f_{\text{bias}}, \quad z = x - o,\ f_{\text{bias}} = -450$	US	$[-100, 100]^D$	f_{bias}
F8	Shifted Schwefel 1.2	$f_8(\vec{X}) = \sum_{i=1}^{D}\left(\sum_{j=1}^{i} z_j\right) + f_{\text{bias}}, \quad z = x - o,\ f_{\text{bias}} = -450$	UN	$[-100, 100]^D$	f_{bias}
F9	Shifted Rosenbrock	$f_9(\vec{X}) = \sum_{i=1}^{D-1}\left(100\left(z_i^2 - z_{i+1}\right)^2 + (z_i - 1)^2\right) + f_{\text{bias}}, \quad z = x - o + 1,\ f_{\text{bias}} = 390$	MN	$[-100, 100]^D$	f_{bias}
F10	Shifted Rastrigin	$f_{10}(\vec{X}) = \sum_{i=1}^{D}\left[z_i^2 - 10\cos(2\pi z_i) + 10\right] + f_{\text{bias}}, \quad z = x - o,\ f_{\text{bias}} = -330$	MS	$[-5, 5]^D$	f_{bias}
F11	Step	$f_{11}(\vec{X}) = \sum_{i=1}^{D}\left(\lfloor x_i + 0.5\rfloor^2\right)$	US	$[-100, 100]^D$	0
F12	Penalized 2	$f_{12}(\vec{X}) = \frac{1}{10}\left\{\sin^2(\pi x_1) + \sum_{i=1}^{D-1}(x_i - 1)^2\left[1 + \sin^2(3\pi x_{i+1})\right] + (x_n - 1)^2\right\}$ $\left[1 + \sin^2(2\pi x_{i+1})\right] + \sum_{i=1}^{D} u(x_i, 5, 100, 4)$	MN	$[-50, 50]^D$	0
F13	Alpine	$f_{13}(\vec{X}) = \sum_{i=1}^{D}\lvert x_i \cdot \sin(x_i) + 0.1 \cdot x_i\rvert$	MS	$[-10, 10]^D$	0

Table 2: Comparisons of ABC-SA and ABC variants on 50D problems.

Func.	ABC-SA		ABC			GABC			IABC		
	Mean	Std. Dev.	Mean	Std. Dev.	Sign	Mean	Std. Dev.	Sign	Mean	Std. Dev.	Sign
F1	3.10E + 01	1.18E + 01	3.84E + 01	1.07E + 01	=	3.32E + 01	6.71E + 00	=	3.25E + 01	1.75E + 01	=
F2	5.30E − 14	4.10E − 15	1.17E − 13	1.62E − 14	+	7.96E − 14	1.18E − 15	+	7.44E − 14	9.63E − 15	+
F3	0.00E + 00	0.00E + 00	2.02E − 11	7.11E − 12	+	8.53E − 14	4.16E − 14	+	8.53E − 14	3.58E − 14	+
F4	1.11E − 16	2.17E − 16	4.78E − 12	2.61E − 13	+	6.11E − 16	3.62E − 16	+	1.26E − 13	6.80E − 14	+
F5	0.00E + 00	0.00E + 00	3.84E − 14	1.90E − 14	+	8.53E − 15	9.94E − 15	+	1.71E − 14	8.99E − 15	+
F6	−2.09E + 04	2.51E − 15	−2.09E + 04	6.09E + 00	+	−2.09E + 04	4.05E − 11	=	−2.09E + 04	5.57E − 14	=
F7	−4.50E + 02	0.00E + 00	−4.50E + 02	6.36E − 14	=	−4.50E + 02	2.84E − 14	=	−4.50E + 02	4.92E − 14	=
F8	1.92E + 04	5.19E + 03	3.22E + 04	1.79E + 03	+	3.61E + 04	5.89E + 03	+	2.92E + 04	5.21E + 03	+
F9	3.98E + 02	3.11E + 00	5.03E + 02	3.96E + 00	+	4.13E + 02	2.45E + 01	+	4.23E + 02	2.48E + 01	+
F10	−3.30E + 02	0.00E + 00	−3.30E + 02	3.14E − 14	=	−3.30E + 02	0.00E + 00	=	−3.30E + 02	1.02E − 14	=
F11	0.00E + 00	0.00E + 00	0.00E + 00	0.00E + 00	=	0.00E + 00	0.00E + 00	=	0.00E + 00	0.00E + 00	=
F12	4.69E − 15	1.90E − 16	5.90E − 14	9.82E − 15	+	1.93E − 14	4.77E − 15	+	8.77E − 15	7.43E − 15	+
F13	3.69E − 24	9.34E − 26	2.95E − 23	3.01E − 23	+	7.71E − 24	3.90E − 25	+	6.53E − 24	1.66E − 25	+

Table 3: Comparisons of ABC-SA and ABC variants on 100D problems.

Func.	ABC-SA		ABC			GABC			IABC		
	Mean	Std. Dev.	Mean	Std. Dev.	Sign	Mean	Std. Dev.	Sign	Mean	Std. Dev.	Sign
F1	7.65E + 01	4.76E + 01	1.26E + 02	3.55E + 01	+	1.35E + 02	2.95E + 01	+	1.34E + 02	2.27E + 01	+
F2	6.16E − 13	9.63E − 14	6.97E − 07	3.25E − 07	+	4.56E − 12	1.18E − 12	+	7.06E − 11	1.31E − 14	+
F3	2.27E − 13	5.41E − 13	3.97E − 01	5.81E − 01	+	2.37E − 11	5.97E − 11	+	1.21E − 12	6.69E − 13	+
F4	1.58E − 14	2.64E − 14	1.30E − 04	7.13E − 04	=	7.73E − 10	3.90E − 09	=	3.54E − 12	9.27E − 12	+
F5	1.48E − 13	4.40E − 14	1.64E − 04	4.98E − 05	+	7.62E − 13	3.24E − 13	+	1.20E − 10	1.25E − 11	+
F6	−4.19E + 04	6.84E + 01	−4.06E + 04	2.61E + 02	+	−4.16E + 04	1.53E + 02	+	−4.19E + 04	6.24E + 01	=
F7	−4.50E + 02	1.90E − 14	−4.50E + 02	1.81E − 13	=	−4.50E + 02	8.53E − 14	=	−4.50E + 02	4.90E − 14	=
F8	8.85E + 04	1.00E + 03	1.61E + 05	2.43E + 04	+	1.60E + 05	1.15E + 04	+	1.95E + 05	7.92E + 03	+
F9	3.89E + 02	9.55E + 00	3.95E + 02	1.04E + 01	+	4.14E + 02	4.12E + 01	+	4.10E + 02	1.76E + 01	+
F10	−3.30E + 02	0.00E + 00	−3.22E + 02	9.41E − 04	+	−3.30E + 02	7.20E − 07	=	−3.30E + 02	5.68E − 14	=
F11	1.17E + 01	5.24E + 00	4.32E + 01	6.43E + 00	+	9.10E + 01	6.92E + 00	+	2.08E + 01	9.52E + 00	+
F12	9.12E − 13	1.04E − 15	5.54E − 11	7.27E − 13	+	8.71E − 13	6.90E − 14	−	1.15E − 12	3.91E − 13	+
F13	1.52E − 20	5.48E − 19	8.14E − 19	9.17E − 20	+	7.03E − 20	4.45E − 19	=	8.44E − 20	1.06E − 19	=

the original ABC [7], GABC [14], and IABC [13] on problems F1–F13.

The parameters of test algorithms are set to their original values given in their corresponding papers, except for the maximum number of function evaluations, population size, and limit, which are set to the same values for all ABC variants. ABC-SA, ABC, and GABC implement random initialization mechanisms whereas IABC employs a chaotic initialization as described in [13]. All algorithms have been simulated in MATLAB environment and executed on the same computer with Intel Xeon CPU (2.67 GHz) and 16 GB of memory.

The computational results are presented in Table 2 for 50D problems, Table 3 for 100D problems, and Table 4 for 200D problems. In Tables 2–4, results are given in terms of mean and standard deviation of the objective values due to the repetitive runs for the global best solutions. All algorithms were run 30 times with random seeds and the stopping criteria set to 4,000 iteration, which means that 320,000 functions evaluations for 50D and 100D problems and 400,000 function evaluations for 200D problems approximately. For a precise and pairwise comparison, statistical significances of the differences between the means of two algorithms are analyzed using t-tests where significance level is set to 0.05. In Tables 2–4, "+" in the columns next to competing algorithms shows that ABC-SA outperforms the competitor algorithm, "=" indicates that the difference between the ABC-SA and the compared algorithm is not statistically significant, and "−" depicts that the competitor algorithm is better than ABC-SA at a level of 0.05 significance.

Tables 2–4 show that, according to pairwise t tests, ABC-SA obtains statistically better results on 25, 28, and 28 cases out of 39 comparisons for each of the 50D, 100D, and 200D problem types, respectively. Specifically, ABC-SA is inferior to IABC on F6 and F9 with 200D and GABC on F12 with 100D. There is no significant difference on the results that

Table 4: Comparisons of ABC-SA and ABC variants on 200D problems.

Func.	ABC-SA		ABC			GABC			IABC		
	Mean	Std. Dev.	Mean	Std. Dev.	Sign	Mean	Std. Dev.	Sign	Mean	Std. Dev.	Sign
F1	$4.06E+02$	$3.34E+01$	$4.32E+02$	$4.21E+01$	=	$4.47E+02$	$5.89E+01$	+	$4.58E+02$	$6.77E+01$	+
F2	$1.85E-05$	$3.30E-05$	$4.44E-02$	$2.13E-02$	+	$6.68E-05$	$1.65E-05$	+	$1.47E-04$	$3.69E-05$	+
F3	$8.27E-06$	$2.53E-05$	$3.17E+01$	$5.89E+00$	+	$6.51E+00$	$1.74E+00$	+	$4.02E+00$	$1.03E+00$	+
F4	$6.05E-10$	$2.26E-09$	$5.75E-04$	$2.58E-03$	=	$1.79E-08$	$5.29E-08$	=	$6.05E-08$	$4.68E-08$	+
F5	$6.42E-03$	$4.93E-04$	$1.24E-01$	$1.35E-02$	+	$9.76E-03$	$1.21E-03$	+	$1.83E-02$	$1.40E-03$	+
F6	$-8.19E+04$	$1.00E+02$	$-7.63E+04$	$7.61E+02$	+	$-7.89E+04$	$4.20E+02$	+	$-8.37E+04$	$3.68E+02$	−
F7	$8.56E+05$	$1.81E+04$	$8.60E+05$	$1.46E+04$	=	$8.73E+05$	$2.49E+04$	+	$8.62E+05$	$1.75E+04$	=
F8	$2.89E+06$	$9.10E+04$	$2.91E+06$	$2.07E+05$	=	$2.93E+06$	$3.13E+05$	=	$2.98E+06$	$3.90E+05$	=
F9	$7.63E+11$	$4.37E+08$	$8.02E+11$	$5.05E+09$	+	$7.87E+11$	$4.72E+08$	+	$7.61E+11$	$4.93E+08$	−
F10	$3.74E+03$	$4.19E+01$	$3.78E+03$	$5.94E+01$	+	$3.76E+03$	$5.55E+01$	+	$3.76E+03$	$4.60E+01$	=
F11	$7.34E+02$	$4.40E+02$	$8.82E+03$	$9.22E+02$	+	$5.39E+03$	$9.03E+02$	+	$1.04E+03$	$6.01E+02$	+
F12	$3.79E-11$	$4.90E-13$	$1.91E-09$	$9.41E-14$	+	$4.88E-11$	$2.25E-12$	+	$6.22E-11$	$3.89E-12$	+
F13	$9.55E-18$	$1.04E-18$	$5.02E-15$	$1.55E-15$	+	$4.44E-16$	$5.03E-17$	+	$1.92E-17$	$7.33E-18$	+

Table 5: The comparisons of ABC-SA and DE variants on 30D problems.

Func.	Max.FE	ABC-SA		jDE		JADE		SaDE	
		Mean	Std. Dev.	Mean	Std. Dev.	Mean	Std. Dev.	Mean	Std. Dev.
F1	300,000	**1.19E − 01**	**7.91E − 02**	$1.30E+01$	$1.40E+01$	$3.20E-01$	$1.10E+00$	$2.10E+01$	$7.70E+01$
F2	50,000	**3.01E − 09**	**7.44E − 10**	$2.37E-04$	$7.10E-05$	$3.35E-09$	$2.84E-09$	$3.81E-06$	$8.26E-07$
F3	100,000	**2.77E − 09**	**6.09E − 10**	$2.37E-04$	$7.10E-05$	$3.35E-09$	$2.84E-09$	$3.81E-06$	$8.26E-07$
F4	50,000	$7.23E-09$	$4.11E-09$	$7.29E-06$	$1.05E-05$	$1.57E-08$	$1.09E-07$	**2.52E − 09**	**1.24E − 09**
F6	100,000	**8.63E − 11**	**4.94E − 11**	$1.70E-10$	$2.62E-10$	$2.62E-04$	$3.59E-04$	$1.13E-08$	$1.08E-08$
F11	10,000	**4.41E + 00**	**3.90E + 00**	$6.13E+02$	$1.72E+02$	$5.62E+00$	$1.87E+00$	$5.07E+01$	$1.34E+01$
F12	50,000	**1.41E − 10**	**7.23E − 11**	$1.80E-05$	$1.42E-05$	$1.87E-10$	$1.09E-09$	$1.93E-09$	$1.53E-09$
F13	300,000	**4.99E − 11**	**2.49E − 10**	$6.08E-10$	$8.36E-10$	$2.78E-05$	$8.43E-06$	$2.94E-06$	$3.47E-06$

are obtained by ABC and ABC-SA on F1 (50D and 100D), F4 (100D and 200D), F7 (all dimensions), F8 200D, F10 50D, and F11 50D. Moreover, on F1 50D, F6 50D, F4 (100D and 200D), F7 (50D and 100D), F8 200D, F10 (50D and 100D), F11 50D, and F13 100D, GABC and ABC-SA perform statistically similar. Further, ABC-SA and IABC perform equally well, namely, on F1 50D, F6 (50D and 100D), F7 (all dimensions), F8 200D, F10 (all dimensions), F11 50D, and F13 100D. Standard deviation of the results also indicates that ABC-SA has a stable performance. According to the results of Tables 2–4, one can safely conclude that ABC-SA significantly surpasses ABC, GABC, and IABC on 50D, 100D, and 200D problems.

To vividly describe the effectiveness of ABC-SA framework, the convergence curves of some benchmark problems are given in Figure 3. According to the figure, ABC-SA shows better convergence behavior on the majority of test cases when compared to ABC, GABC, and IABC.

Furthermore, mean acceptance rate curves for solution acceptance rule in ABC-SA framework are given in Figure 4 and the acceptance rate is determined as follows:

$$\text{acceptance rate} = \frac{\text{number of accepted worse solutions}}{\text{total number of worse solutions}}. \quad (10)$$

The curves in Figure 4 clearly coincide with the acceptance probability curve given in Figure 2. Figure 4 also shows the nonlinear decreasing of acceptance rate throughout the search process.

4.4. Comparison with PSO and DE Variants. The performance of ABC-SA is also tested against novel and powerful variants of DE and PSO. The competitor algorithms are self-adapting DE (jDE) [41], adaptive DE with optional external archive (JADE) [42], self-adaptive DE (SaDE) [43], comprehensive learning PSO (CLPSO) [44], self-organizing hierarchical PSO with time-varying acceleration coefficients (HPSO-TVAC) [45], and fully informed particle swarm (FIPS) [46]. The results of these algorithms are taken directly from corresponding studies. The experimental results for 30D problems are shown in Tables 5 and 6 for DE and PSO variants, respectively. Some of the benchmarks problems are not included in the comparisons, since results on these problems were not reported in competitor studies. The previous parameter setting for ABC-SA is used, but this time maximum function evaluation number (Max.FE) is employed as the stopping criteria. Since the results of competitor algorithms are taken directly from corresponding studies, statistical significance tests could not be applied. Therefore, in this part of the analysis, mean and standard

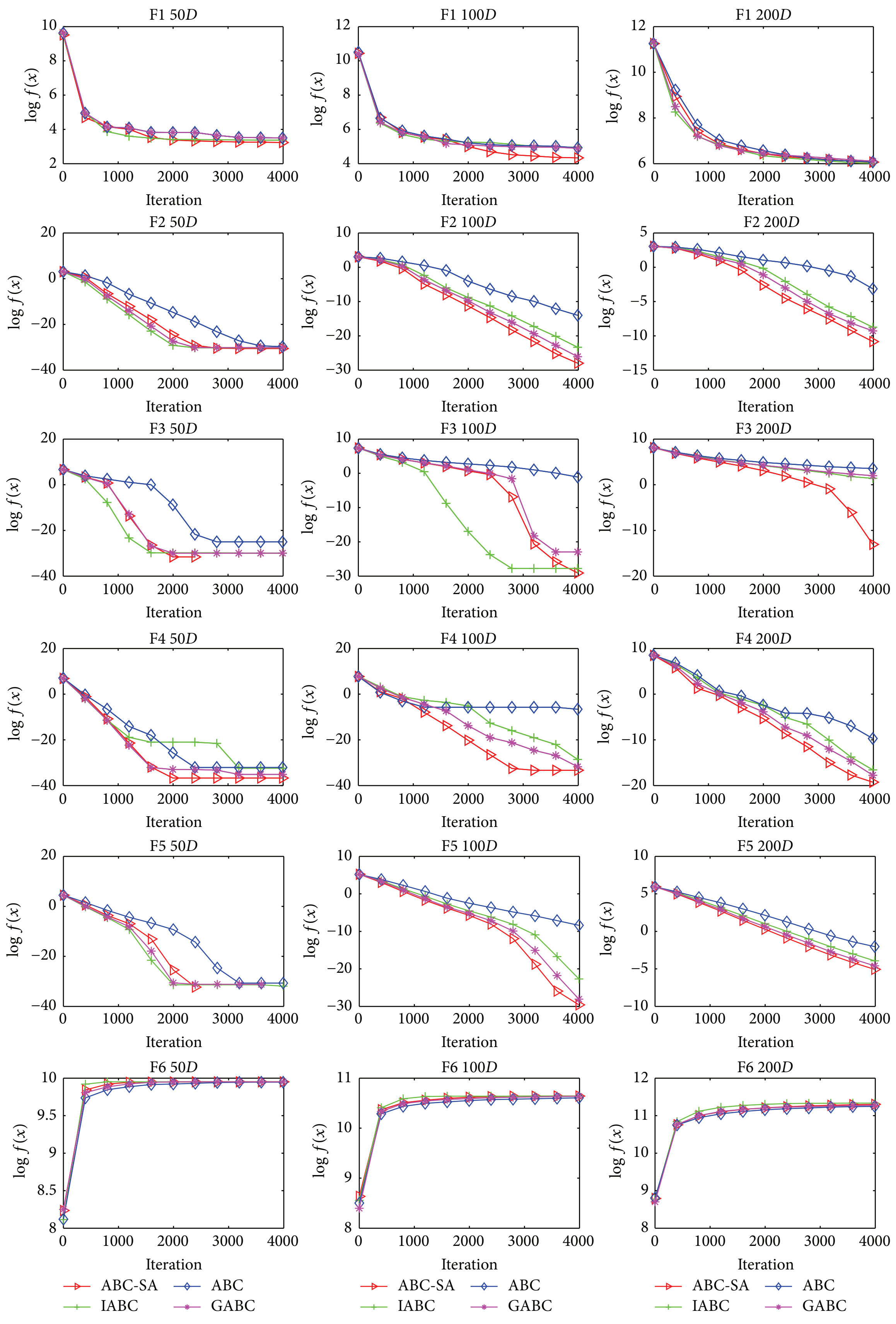

Figure 3: Convergence curves for ABC-SA and ABC variants.

Table 6: The comparisons of ABC-SA and PSO variants on 30D problems.

Func.	Max.FE	ABC-SA		FIPS		HPSO-TVAC		CLPSO	
		Mean	Std. Dev.	Mean	Std. Dev.	Mean	Std. Dev.	Mean	Std. Dev.
F1	200,000	**7.61E + 00**	**2.18E − 01**	2.51E + 01	5.10E − 01	2.39E + 01	2.65E + 01	1.13E + 01	9.85E + 00
F2	200,000	4.01E − 13	1.05E − 13	2.33E − 07	7.19E − 08	**7.29E − 14**	**3.00E − 14**	3.66E − 07	7.57E − 08
F3	200,000	**6.91E − 10**	**9.44E − 11**	6.51E + 01	1.33E + 01	9.43E + 00	3.48E + 00	9.05E − 05	1.25E − 04
F4	200,000	**8.89E − 12**	**6.56E − 13**	9.01E − 12	1.84E − 11	9.75E − 03	8.33E − 03	9.02E − 09	8.57E − 09
F6	200,000	**1.79E − 12**	**8.89E − 11**	9.93E + 02	5.09E + 02	1.59E + 03	3.26E + 02	3.82E − 04	1.28E − 05
F11	200,000	0.00E + 00	0.00E + 00	0.00E + 00	0.00E + 00	0.00E + 00	0.00E + 00	0.00E + 00	0.00E + 00
F12	200,000	5.03E − 23	1.44E − 24	2.70E − 14	1.57E − 14	**2.79E − 28**	**2.18E − 28**	1.25E − 12	9.45E − 12

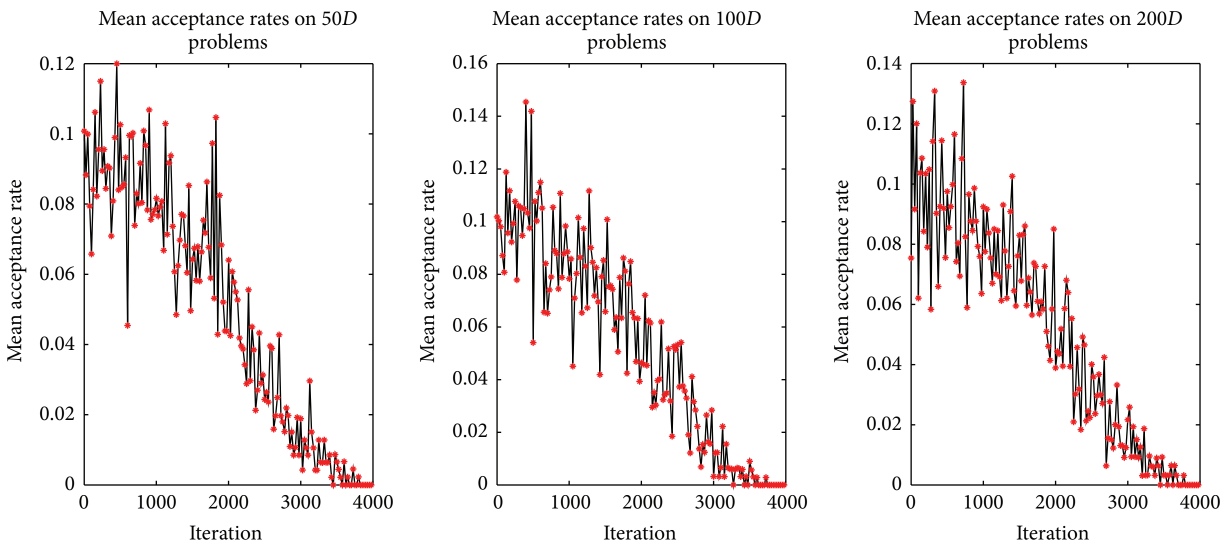

Figure 4: Mean acceptance rates.

deviations of results are compared directly. In Tables 5 and 6, the winner algorithms are indicated in bold character according to the mean results of 30 independent runs. As can be seen from Tables 5 and 6, ABC-SA outperforms other algorithms on all cases, except in the case of F2, F4, and F12. SaDE performs better than the ABC-SA on F4 and HPSO-TVAC outperforms ABC-SA on only F2 and F12. ABC achieves better results on the majority of the instances in terms of robustness according to the standard deviations of the results. These results also indicate the effectiveness of ABC-SA when compared to other novel swarm based and evolutionary algorithms.

5. Conclusion and Future Work

This paper presents a modified ABC algorithm, namely, the ABC-SA, enhanced with a solution acceptance rule and a probabilistic multisearch strategy. In ABC-SA, instead of a greedy selection, a new acceptance rule is presented, where a worse candidate solution has a probability to be accepted. Furthermore, to balance the diversification and intensification tendency of the algorithm, a probabilistic multisearch mechanism is employed. In the probabilistic multisearch, a search rule is selected among three alternatives according to their predetermined probabilities. The proposed algorithm is very effective as compared to other novel ABC variants and state-of-the-art algorithms. Several experimental studies are conducted and results show that ABC-SA outperforms all other competitor algorithms on the majority of the test cases. Future research will be along the line of implementing the ABC-SA to solve complex engineering problems.

Conflict of Interests

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

References

[1] D. E. Golberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, vol. 1989, Addison-Wesley, 1989.

[2] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*, MIT Press, 1992.

[3] P. Moscato, "On evolution, search, optimization, genetic algorithms and martial arts: towards memetic algorithms," C3P Report 826, Caltech Concurrent Computation Program, 1989.

[4] R. Storn and K. Price, "Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[5] A. Colorni, M. Dorigo, and V. Maniezzo, "Distributed optimization by ant colonies," in *Proceedings of the 1st European Conference on Artificial Life*, vol. 142, pp. 134–142, Paris, France, 1991.

[6] R. C. Eberhart and J. Kennedy, "New optimizer using particle swarm theory," in *Proceedings of the 6th International Symposium on Micro Machine and Human Science*, vol. 1, pp. 39–43, New York, NY, USA, October 1995.

[7] D. Karaboga and B. Basturk, "A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm," *Journal of Global Optimization*, vol. 39, no. 3, pp. 459–471, 2007.

[8] X.-S. Yang and S. Deb, "Engineering optimisation by cuckoo search," *International Journal of Mathematical Modelling and Numerical Optimisation*, vol. 1, no. 4, pp. 330–343, 2010.

[9] I. Fister Jr., X.-S. Yang, and J. Brest, "A comprehensive review of firefly algorithms," *Swarm and Evolutionary Computation*, vol. 13, pp. 34–46, 2013.

[10] D. Karaboga, B. Gorkemli, C. Ozturk, and N. Karaboga, "A comprehensive survey: artificial bee colony (ABC) algorithm and applications," *Artificial Intelligence Review*, vol. 42, no. 1, pp. 21–57, 2014.

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

[12] A. Banharnsakun, B. Sirinaovakul, and T. Achalakul, "Job shop scheduling with the best-so-far ABC," *Engineering Applications of Artificial Intelligence*, vol. 25, no. 3, pp. 583–593, 2012.

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

[14] G. Zhu and S. Kwong, "Gbest-guided artificial bee colony algorithm for numerical function optimization," *Applied Mathematics and Computation*, vol. 217, no. 7, pp. 3166–3173, 2010.

[15] W. Gao, S. Liu, and L. Huang, "A global best artificial bee colony algorithm for global optimization," *Journal of Computational and Applied Mathematics*, vol. 236, no. 11, pp. 2741–2753, 2012.

[16] 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.

[17] B. Akay and D. Karaboga, "A modified artificial bee colony algorithm for real-parameter optimization," *Information Sciences*, vol. 192, no. 1, pp. 120–142, 2012.

[18] T. Liao, D. Aydin, and T. Stützle, "Artificial bee colonies for continuous optimization: experimental analysis and improvements," *Swarm Intelligence*, vol. 7, no. 4, pp. 327–356, 2013.

[19] W.-F. Gao, S.-Y. Liu, and L.-L. Huang, "Enhancing artificial bee colony algorithm using more information-based search equations," *Information Sciences*, vol. 270, no. 1, pp. 112–133, 2014.

[20] J. Qiu, J. Wang, D. Yang, and J. Xie, "An artificial bee colony algorithm with modified search strategies for global numerical optimization," *Journal of Theoretical & Applied Information Technology*, vol. 48, no. 1, pp. 293–302, 2013.

[21] A. Banitalebi, M. I. A. Aziz, A. Bahar, and Z. A. Aziz, "Enhanced compact artificial bee colony," *Information Sciences*, vol. 298, pp. 491–511, 2015.

[22] 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.

[23] W. Gao, F. T. Chan, L. Huang, and S. Liu, "Bare bones artificial bee colony algorithm with parameter adaptation and fitness-based neighborhood," *Information Sciences*, vol. 316, pp. 180–200, 2015.

[24] L. Ma, K. Hu, Y. Zhu, and H. Chen, "A hybrid artificial bee colony optimizer by combining with life-cycle, Powell's search and crossover," *Applied Mathematics and Computation*, vol. 252, pp. 133–154, 2015.

[25] Q.-K. Pan, M. F. Tasgetiren, P. N. Suganthan, and T. J. Chua, "A discrete artificial bee colony algorithm for the lot-streaming flow shop scheduling problem," *Information Sciences*, vol. 181, no. 12, pp. 2455–2468, 2011.

[26] J.-Q. Li, Q.-K. Pan, and K.-Z. Gao, "Pareto-based discrete artificial bee colony algorithm for multi-objective flexible job shop scheduling problems," *The International Journal of Advanced Manufacturing Technology*, vol. 55, no. 9–12, pp. 1159–1169, 2011.

[27] W. Y. Szeto, Y. Wu, and S. C. Ho, "An artificial bee colony algorithm for the capacitated vehicle routing problem," *European Journal of Operational Research*, vol. 215, no. 1, pp. 126–135, 2011.

[28] A. Yurtkuran and E. Emel, "A modified artificial bee colony algorithm for *p*-center problems," *The Scientific World Journal*, vol. 2014, Article ID 824196, 9 pages, 2014.

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

[30] D. Karaboga, S. Okdem, and C. Ozturk, "Cluster based wireless sensor network routing using artificial bee colony algorithm," *Wireless Networks*, vol. 18, no. 7, pp. 847–860, 2012.

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

[32] D. Karaboga and C. Ozturk, "A novel clustering approach: Artificial Bee Colony (ABC) algorithm," *Applied Soft Computing Journal*, vol. 11, no. 1, pp. 652–657, 2011.

[33] I. M. S. De Oliveira and R. Schirru, "Swarm intelligence of artificial bees applied to in-core fuel management optimization," *Annals of Nuclear Energy*, vol. 38, no. 5, pp. 1039–1045, 2011.

[34] W.-F. Gao, S.-Y. Liu, and F. Jiang, "An improved artificial bee colony algorithm for directing orbits of chaotic systems," *Applied Mathematics and Computation*, vol. 218, no. 7, pp. 3868–3879, 2011.

[35] W.-C. Hong, "Electric load forecasting by seasonal recurrent SVR (support vector regression) with chaotic artificial bee colony algorithm," *Energy*, vol. 36, no. 9, pp. 5568–5578, 2011.

[36] S. K. Kumar, M. K. Tiwari, and R. F. Babiceanu, "Minimisation of supply chain cost with embedded risk using computational intelligence approaches," *International Journal of Production Research*, vol. 48, no. 13, pp. 3717–3739, 2010.

[37] W.-F. Gao and S.-Y. Liu, "A modified artificial bee colony algorithm," *Computers & Operations Research*, vol. 39, no. 3, pp. 687–697, 2012.

[38] W.-F. Gao, S.-Y. Liu, and L.-L. Huang, "A novel artificial bee colony algorithm with Powell's method," *Applied Soft Computing Journal*, vol. 13, no. 9, pp. 3763–3775, 2013.

[39] D. Karaboga and B. Akay, "A comparative study of artificial bee colony algorithm," *Applied Mathematics and Computation*, vol. 214, no. 1, pp. 108–132, 2009.

[40] D. Karaboga and B. Basturk, "On the performance of artificial bee colony (ABC) algorithm," *Applied Soft Computing*, vol. 8, no. 1, pp. 687–697, 2008.

[41] J. Brest, S. Greiner, B. Bošković, M. Mernik, and V. Zumer, "Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 6, pp. 646–657, 2006.

[42] J. Zhang and A. C. Sanderson, "JADE: adaptive differential evolution with optional external archive," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 5, pp. 945–958, 2009.

[43] A. K. Qin, V. L. Huang, and P. N. Suganthan, "Differential evolution algorithm with strategy adaptation for global numerical optimization," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 2, pp. 398–417, 2009.

[44] J. J. Liang, A. K. Qin, P. N. Suganthan, and S. Baskar, "Comprehensive learning particle swarm optimizer for global optimization of multimodal functions," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 3, pp. 281–295, 2006.

[45] A. Ratnaweera, S. K. Halgamuge, and H. C. Watson, "Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 240–255, 2004.

[46] R. Mendes, J. Kennedy, and J. Neves, "The fully informed particle swarm: simpler, maybe better," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 204–210, 2004.