Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2014, Article ID 432654, 14 pages
http://dx.doi.org/10.1155/2014/432654

Research Article

A Modified Artificial Bee Colony Algorithm Based on Search Space Division and Disruptive Selection Strategy

Zhen-an He,[1,2,3] Caiwen Ma,[1] Xianhong Wang,[1] Lei Li,[4] Ying Wang,[4] Yuan Zhao,[1,2,5] and Huinan Guo[1]

[1] *Xi'an Institute of Optics and Precision Mechanics, Chinese Academy of Sciences, Xi'an, Shaanxi 710019, China*
[2] *University of Chinese Academy of Sciences, Beijing 100049, China*
[3] *Unit 96165 of the Chinese People's Liberation Army, Shangrao, Jiangxi 334000, China*
[4] *North Optoelectronic Stock Ltd., Xi'an, Shaanxi 710032, China*
[5] *Engineering University of the Chinese Armed Police Force, Xi'an, Shaanxi 710086, China*

Correspondence should be addressed to Zhen-an He; std_2009@163.com

Received 27 May 2014; Revised 28 August 2014; Accepted 11 September 2014; Published 28 September 2014

Academic Editor: Fang Zong

Artificial bee colony (ABC) algorithm has attracted much attention and has been applied to many scientific and engineering applications in recent years. However, there are still some insufficiencies in ABC algorithm such as poor quality of initial solution, slow convergence, premature, and low precision, which hamper the further development and application of ABC. In order to further improve the performance of ABC, we first proposed a novel initialization method called search space division (SSD), which provided high quality of initial solutions. And then, a disruptive selection strategy was used to improve population diversity. Moreover, in order to accelerate convergence rate, we changed the definition of the scout bee phase. In addition, we designed two types of experiments to testify our proposed algorithm. On the one hand, we conducted experiments to make sure how much each modification makes contribution to improving the performance of ABC. On the other hand, comprehensive experiments were performed to prove the superiority of our proposed algorithm. The experimental results indicate that SDABC significantly outperforms other ABCs, contributing to higher solution accuracy, faster convergence speed, and stronger algorithm stability.

1. Introduction

Optimization problems exist extensively in information science, control engineering, industrial design, operational research, and so on. Therefore, optimization algorithms, which are of great importance to engineering and scientific research, have attracted more and more of the researchers' attention and many achievements have been got. However, conventional optimization algorithms fail to process problems characterized as nonconvex, nondifferentiable, discontinuous, high dimensional, and so forth; swarm intelligence algorithms that have some advantages, such as scalability, fault tolerance, adaptation, speed, modularity, autonomy, and parallelism [1], provide a promising way to solve the complex optimization problems mentioned above. Some excellent algorithms, such as genetic algorithm (GA) [2], particle swarm optimization (PSO) [3], differential evolutionary (DE) algorithm [4], ant colony optimization (ACO) [5], simulate anneal arithmetic (SAA) [6], artificial bee colony (ABC) algorithm [7], have been proposed and successful applications have been made in many fields. Among them, by simulating foraging behavior of honey bee swarm, ABC developed by Karaboga is competitive to other swarm intelligence algorithms [8]. As a result, it has been widely used in multiobjective optimization, circuit design, SAR image segmentation, flow shop scheduling, and related optimization problems [9–11]. Nevertheless, as research on ABC is at the primary phase, many problems such as poor quality of initial solutions, slow convergence, premature, and low precision are still hampering the development and application of ABC.

To overcome the above mentioned shortages, researchers have presented many kinds of ABC variants. Zhu and Kwong [12] proposed a gbest-guided ABC (GABC) algorithm by incorporating the information of global best (gbest) solution

into the solution search equation to improve the exploitation. Kang et al. [13] proposed a memetic algorithm which combined Hooke-Jeeves pattern search with artificial bee colony algorithm. Alatas [14] introduced the chaotic maps into the initialization and chaotic search into the scout bee phase to improve the convergence characteristics and to prevent the ABC to get stuck on local solutions. Gao and Liu [15] proposed a modified ABC by using a modified solution search equation and new initial method. Akay and Karaboga [1] proposed a modified ABC algorithm by controlling the frequency of perturbation and introducing the ratio of the variance operator into search equation. Mohammed [16] proposed a generalized opposition-based ABC by introducing the concept of generalized opposition-based learning through the initialization step and generation jumping. Luo et al. [17] used segmental-search strategy to make sure of fast convergence and to avoid trapping into local optimum. More extensive review of ABC can be seen in [18, 19].

According to the literatures mentioned above, the ABC variants have achieved a better performance, but some shortcomings of slow convergence and premature and low precision have not been solved yet. We note that to improve the performance of ABC, researchers mainly focus on search strategy research, but other important factors affecting performance, such as initialization method, selective strategy, and fitness value, are insufficient.

In order to further improve the performance of ABC, we proposed a modified artificial bee colony algorithm based on search space division and disruptive selection strategy (SDABC). In SDABC, first of all, a new initialization method called search space division, which steadily provided high quality of initial solutions, was presented. Note that convergence speed, final solution accuracy, and stability were improved by using search space division. It is well known that greedy strategy used in ABC is useful because it is quick to think up and often gives good approximations to the optimum. However, greedy strategy mostly (but not always) fails to find the globally optimal solution, because it usually does not operate exhaustively on all the data. It can make commitments to certain choices too early which prevent it from finding the best overall solution later. And thus, in this paper, we introduced the disruptive selection strategy to increase the diversity of population. Moreover, in order to accelerate convergence speed, a new search equation was presented in the scout bee phase. In addition, our proposed algorithm was evaluated on fourteen standard benchmark functions which had ever been applied to verify optimization methods in continuous optimization problems. The simulation results show that SDABC significantly outperforms other ABCs, contributing to higher solution accuracy, faster convergence speed, and stronger algorithm stability.

The rest of this paper is structured as follows. In Section 2, we make a brief review of standard ABC. Section 3 provides detailed descriptions of our proposed algorithm. Subsequently, some experiments are conducted to verify the performance of SDABC in Section 4. This is followed by conclusion and future research directions in Section 5.

2. Standard Artificial Bee Colony Algorithm

The ABC algorithm which was originally introduced by Karaboga is a recently proposed optimization algorithm inspired by simulating the foraging behavior of a honey bee swarm [1]. In the minimal model, a colony of artificial bees consists of three kinds of bees [7, 8]: employed bees, onlooker bees, and scout bees. Both of the number of employed bees and onlooker bees are equal to half of the bee colony. There is a one-to-one correspondence between each employed bee and every food source. Employed bees are responsible for searching food sources, gathering information, and sharing food information with onlooker bees around the hive. The employed bee whose food source has been exhausted by the bees becomes a scout. Scout bee, just as its name implies, could have the fast discovery of the group of feasible solutions as a task. In the ABC algorithm, the position of a food source represents a possible solution to the optimization problem and the nectar amount of a food source corresponds to the quality (fitness) of the associated solution. The ABC algorithm can be concluded into four phases: initialization phase, employed bee phase, onlooker bee phase, and scout bee phase, as described below.

(1) Initialization Phase. To begin with, a population of SN individuals is generated randomly. Each initial food source X_i $(i = 1, 2, \ldots, FN)$ as a D-dimensional vector is produced by (1), where FN denotes food source, namely, solution

$$x_{ij} = \mathrm{lb}_j + \mathrm{rand}\,(0, 1)\left(\mathrm{ub}_j - \mathrm{lb}_j\right), \tag{1}$$

where $i = 1, 2, \ldots, FN$, $j = 1, 2, \ldots, D$, and D is the number of optimization parameters; ub_j and lb_j are the upper and lower bounds for the dimension j, respectively. And then, each solution is evaluated by

$$\mathrm{fitness}_i = \begin{cases} \dfrac{1}{(1 + f_i)}, & f_i \geq 0 \\ 1 + \left|f_i\right|, & f_i < 0, \end{cases} \tag{2}$$

where $\mathrm{fitness}_i$ denotes fitness value of solution i and f_i represents cost function of solution i.

After the initialization, solutions repeat processes of employed bee phase, onlooker bee phase, and scout bee phase until stop conditions are satisfied.

(2) Employed Bee Phase. At this phase, each employed bee produces a new food source (solution) in the neighborhood of the previous selected solution. The position of the new food source is calculated by

$$v_{ij} = x_{ij} + \phi_{ij}\left(x_{ij} - x_{kj}\right), \tag{3}$$

where $j \in \{1, 2, \ldots, D\}$ and $k \in \{1, 2, \ldots, FN\}$ are randomly chosen indexes and k has to be different from i and v_{ij} is a new solution and ϕ_{ij} is a random number between $[-1, 1]$.

After that, fitness value of the new food source is evaluated, and compares with that of the old food source X_i. The old food source will be replaced by the new one if the new

Begin
Step 1. population initialization based on (1).
Step 2. Calculate fitness value by (2).
Step 3. *cycle* = 1
Step 4. Repeat
Step 5. Update the position of the employed bee by (3).
Step 6. A greedy selection strategy is applied to select the better food source.
Step 7. Calculate probability value by (4).
Step 8. Update the position of the onlooker bee by (3).
Step 9. A greedy selection strategy is applied to select the better food source.
Step 10. Determine the abandoned position, and generate a new position by (1).
Step 11. Memorize the best food source achieved till now. *cycle* = *cycle* + 1
Step 12. Until (*cycle* = maximum cycle number)
End.

Algorithm 1: Standard artificial bee colony algorithm.

one has a better fitness value. Otherwise, the old one will be stored.

(3) Onlooker Bee Phase. When all employed bees have finished their searching process, they share positions and fitness information of the food sources with onlooker bees, each of which selects a food source depending on probability p_i calculated by (4). After an onlooker bee chooses a food source with a probability, a new food source is generated and fitness value is calculated. Subsequently, a greedy selection is applied between the selected food source and the old one:

$$p_i = \frac{\text{fitness}_i}{\left(\sum_{j=1}^{FN} \text{fitness}_j\right)}. \tag{4}$$

(4) Scout Bee Phase. If the fitness value of a food source does not improve for a certain number of cycles, the food source will be abandoned and the corresponding employed bee becomes a scout bee. Then, the scout bee produces a new position randomly as in (1) to replace the position of abandoned food source.

On the basis of the above analysis, the pseudocode of standard artificial bee colony algorithm is simply described in Algorithm 1.

3. Our Proposed Algorithm: SDABC

Attracted by the prospect and potential of the ABC algorithm, researchers have focused on its variants. However, there are still some problems which need to be solved. In this section, to further improve its performance, we introduce the modified artificial bee colony from three aspects, namely, population initialization based on search space division, disruptive selection strategy, and the improved scout bee phase. The detailed descriptions are as follows.

3.1. Population Initialization Based on Search Space Division. An initialization method capable of providing a better exploration of the search-space and presenting only high-quality solutions should improve the performance of a metaheuristic algorithm [20]. Specifically, population initialization influences convergence speed, final solution accuracy, and stability. Generally speaking, in absence of a prior knowledge about the solution, the random initialization is the most commonly used way to produce initial population. However, because of its randomness, it is impossible for random initialization to get high quality of initial solution steadily. That is to say, once in a while, an initial solution produced by random initialization close to optimal solution could result in a fast convergence. Otherwise, it will take considerably more time. Logically, we should be looking in all directions simultaneously [21], in other words, initial solutions should be distributed evenly across the whole search space.

Accordingly, we proposed a novel initialization algorithm called search space division (SSD). In order to make it easy to understand, we take one-dimensional space as an example, as shown in Figure 1. The basic ideal of SSD is given as follows: to begin with, SSD divides the search space into FN segments (e.g., $FN = 4$). There is one and only one initial solution in every segment (searching subspace), which makes sure initial solutions should be distributed relatively evenly across the whole search space. After that, SSD uses each midpoint in searching subspace as a base point, and initial solutions are random bias from base points which can move from left or right for up to half the length of segment. Here, random numbers generated in a limited scope are used to produce different initial solutions at each run, which make a chance for algorithm to succeed in the next run while it failed in the previous run. It is necessary to emphasize that SSD can easily work on the high dimensions. Based on the research described above, the main steps of SSD can be summarized as in Figure 1.

Let $x_{ij} \in [x_{\min ij}, x_{\max ij}]$ be a real number, which is the jth dimension of ith solution. Where $i \in [1, 2, \ldots, FN]$ (FN denotes the number of initial solutions) and $j \in [1, 2, \ldots, D]$ (D represents the number of parameters), $x_{\max ij}$ and $x_{\min ij}$ are the upper and lower bounds for the jth dimension of ith

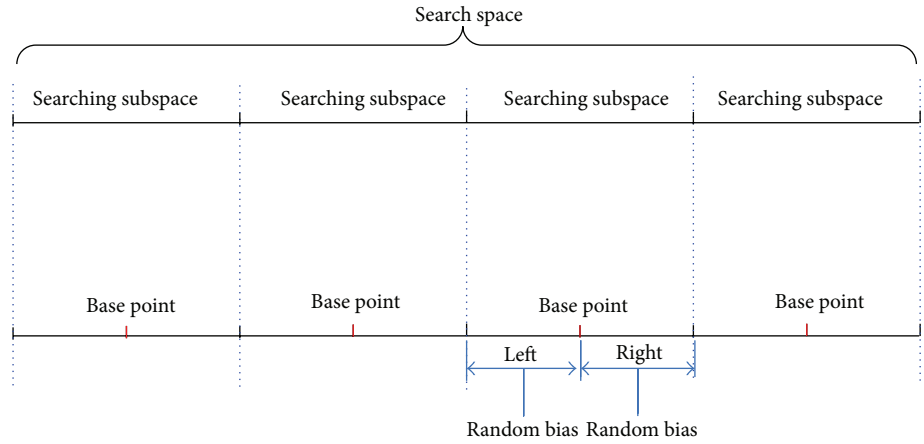


Figure 1: Initial solutions produced by SSD for $FN = 4$.

Begin

Step 1. Set the number of employed bees FN and the number of parameters D

Step 2. **for** $i = 1$ to FN **then do**

Step 3. **for** $j = 1$ to D **then do**

$$x_{\text{center}ij} = x_{\min ij} + (2i - 1)\frac{(x_{\max ij} - x_{\min ij})}{2FN}$$

$$x_{ij} = x_{\text{center}ij} + \text{rand}(-1, 1) \times \frac{(x_{\max ij} - x_{\min ij})}{2FN}$$

Step 4. **end for**

Step 5. **end for**

End.

Algorithm 2: Population initialization based on search space division.

solution, respectively. Then, base points are generated by (5), and initial solutions are produced by using (6)

$$x_{\text{center}ij} = x_{\min ij} + (2i - 1)\frac{\left(x_{\max ij} - x_{\min ij}\right)}{2FN}, \quad (5)$$

where $x_{\text{center}ij}$ is the base point corresponding to the jth dimension of ith solution

$$x_{ij} = x_{\text{center}ij} + \text{rand}\,(-1, 1) \times \frac{\left(x_{\max ij} - x_{\min ij}\right)}{2FN}. \quad (6)$$

Through the above analysis, we use SSD instead of pure random initialization to produce initial solutions, and its pseudocode is given in Algorithm 2.

3.2. Disruptive Selection Strategy. Fitness value is of crucial importance to the ABC algorithm because it is the sole criterion of nectar amount. In the ABC algorithm, fitness value is generated by (2). As described in [22], there is a vital problem which needs to be solved. For example, when finding the minimum value, (2) is used to evaluate function value. However, when the function value is very close to zero, for example, $1E - 20$, fitness value calculated by (2) is rounded up to be 1 ($1E - 20$ is ignored). Subsequently, the fitness of all solutions will be equal to 1 in later iterations. That is to say, a new solution that gives a better fitness value than the old solution will be ignored and the solution will stagnate at the old solution [22]. In order to solve this problem, literature [22] directly used the objective value of function for comparison and selection of the better solution. To some extent, the issue has been solved by the above amendment, but it has raised a new problem: onlooker bees lose search direction, which results in falling into local optimum. As the number of the iterations increases, difference of the objective function value becomes smaller and smaller, which requires fitness value to have a sensitive response to the slight change. Otherwise, onlooker bees cannot select appropriate food sources. Through the above analysis, we proposed a new equation to calculate the fitness value, which is given as follows:

$$\text{fitness}_i = \begin{cases} \dfrac{1}{f_i}, & f_i > 0 \\ \left|f_i\right|, & f_i \le 0, \end{cases} \quad (7)$$

where f_i is the objective value of function and fitness_i is the fitness value.

After that, greedy selection strategy is applied to select food source. However, as is known to all, greedy selection strategy often falls into locally optimal solution, which results from lack of population diversity. In order to solve this issue,

Table 1: Benchmark functions.

Name	Function	Search space	Minimum
Sphere	$f_1(X) = \sum_{i=1}^{D} x_i^2$	$[-100, 100]^D$	0
SumSquare	$f_2(X) = \sum_{i=1}^{D} i x_i^2$	$[-10, 10]^D$	0
Rosenbrock	$f_3(X) = \sum_{i=1}^{D-1} \left[100\left(x_{i+1} - x_i^2\right)^2 + \left(x_i - 1\right)^2\right]$	$[-2.048, 2.048]^D$	0
Schwefel2.21	$f_4(X) = \max_{i=1}^{D} \{\lvert x_i\rvert, 1 \le i \le D\}$	$[-100, 100]^D$	0
Schwefel2.22	$f_5(x) = \sum_{i=1}^{D} \lvert x_i\rvert + \prod_{i=1}^{D} \lvert x_i\rvert$	$[-10, 10]^D$	0
Step	$f_6(X) = \sum_{i=1}^{D} \lvert x_i + 0.5\rvert^2$	$[-100, 100]^D$	0
SumPower	$f_7(X) = \sum_{i=1}^{D} \lvert x_i\rvert^{(i+1)}$	$[-1, 1]^D$	0
Exponential	$f_8(X) = \left\lvert \exp\left(0.5 * \sum_{i=1}^{D} x_i\right) - 1\right\rvert$	$[-1.28, 1.28]^D$	0
Rotated	$f_9(X) = \sum_{i=1}^{D} \sum_{j=1}^{i} x_j^2$	$[-65.536, 65.536]^D$	0
Griewank	$f_{10}(X) = \frac{1}{4000} \sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$[-600, 600]^D$	0
Schwefel	$f_{11}(X) = 418.982887 * D - \sum_{i=1}^{D} \left(x_i \sin\left(\sqrt{\lvert x_i\rvert}\right)\right)$	$[-500, 500]^D$	0
Ackley	$f_{12}(X) = -20 \exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D} \cos\left(2\pi x_i\right)\right) + 20 + e$	$[-32.768, 32.768]^D$	0
Rastrigin	$f_{13}(X) = \sum_{i=1}^{D} \left(x_i^2 - 10\cos\left(2\pi x_i\right) + 10\right)$	$[-5.12, 5.12]^D$	0
Weierstrass	$f_{14}(X) = \sum_{i=1}^{D} \left(\sum_{k=0}^{20} \left[a^k \cos\left(2\pi b^k \left(x_i + 0.5\right)\right)\right]\right) - D\sum_{k=0}^{20} \left[a^k \cos\left(2\pi b^k 0.5\right)\right]$ $a = 0.5, b = 3$	$[-0.5, 0.5]^D$	0

we introduced the disruptive selection strategy to maintain population diversity. The disruptive selection strategy gives more chances for higher and lower individuals to be selected by changing the definition of the fitness function as in (8) and (9) [23]

$$\text{fit}_i = \left|\text{fitness}_i - \overline{\text{fitness}_a}\right|, \tag{8}$$

$$p_i = \frac{\text{fit}_i}{\sum_{n=1}^{FN} \text{fit}_n}, \tag{9}$$

where $\overline{\text{fitness}_a}$ is the average value of the fitness value fitness_i of the individuals in the population. Based on the above amendments, the main steps of disruptive selection strategy are shown in Algorithm 3.

Begin
  % Change the definition of the fitness function
  If $f_i > 0$ **then do**
    $fitness_i = \frac{1}{f_i}$;
  Else do
    $fitness_i = \lvert f_i\rvert$;
  End if
  $fit_i = \left|fitness_i - \overline{fitness_a}\right|$;
  % Calculate probability
  $p_i = \frac{fit_i}{\sum_{n=1}^{FN} fit_n}$;
End.

Algorithm 3: Disruptive selection strategy.

3.3. Improved Scout Bee Phase. When an employed bee becomes a scout bee, (1) is used to randomly generate a new position. Because of its randomness, the new position produced by (1) is more likely to be far away from global optimum or locates in the search area which has been explored. The guidance of the best solution will rapidly accelerate convergence speed, which has been proved in literature [12]. It means that the scout bee has yet to take advantage of

Begin

{--Population initialization--}

Step 1. Set population size *CS*, the number of food sources *FN*, the non-improvement number of food source *limit*, the number of maximum iterations max *Cycle*, and the number of parameters *D*.

Step 2. Generate initial solutions using population initialization based on SSD as in (5) and (6), and calculate their fitness values by (7) and (8).

Step 3. $cycle = 1$

Step 4. **While** $cycle \leq \max Cycle$ **do**

{--Employed bee phase--}

For $i = 1 : FN$ **do**

Update the position of the employed bee based on (3).

Check whether it is out of boundaries or not.

Calculate fitness value by (7) and (8).

A greedy selection strategy is applied to select the better food source.

A food source doesn't improve $trial(i) = trial(i) + 1$, otherwise $trial(i) = 0$ where $trial(i)$ represents the number of the ith trial.

End for

{--Onlooker bee phase--}

Step 5. Calculate p_i based on (9), and initialize $t = 0$, $trial = 0$.

While $t < FN$ **then do**

If random $< p_i$ **then do**

Update the position of the onlooker bee by (3).

Check whether it is out of boundaries or not.

Calculate fitness value by (7) and (8).

A greedy selection strategy is applied to select the better food source.

A food source doesn't improve $trial(i) = trial(i) + 1$, otherwise $trial(i) = 0$.

End if

End while ($t = FN$)

{--Scout bee phase--}

Step 6. **if** $\max(trial) > limit$ **then do**

Generate a new food source for the scout bee by (10).

End if

Step 7. Memorize optimal solution achieved till now. $cycle = cycle + 1$;

Step 8. **End while** ($cycle = \max Cycle$)

End.

Algorithm 4: SDABC.

the information from the optimal solution achieved so far. Moreover, it is a hard work to determine *limit* value because we are facing a big dilemma: setting the value of *limit* that is too large results in poor population diversity, and the value of *limit* that is too small renders slow convergence. As a matter of fact, the sole purpose of a scout bee is to maintain the population diversity, which can be replaced by other strategies. As a result, in this paper, we adopted the disruptive selection strategy to keep population diversity. In addition, for the purpose of accelerating convergence speed, we assigned a new task to the scout bee, and that is to further exploit the promising position. Therefore, a new equation generating the new position for scout bee is given as follows:

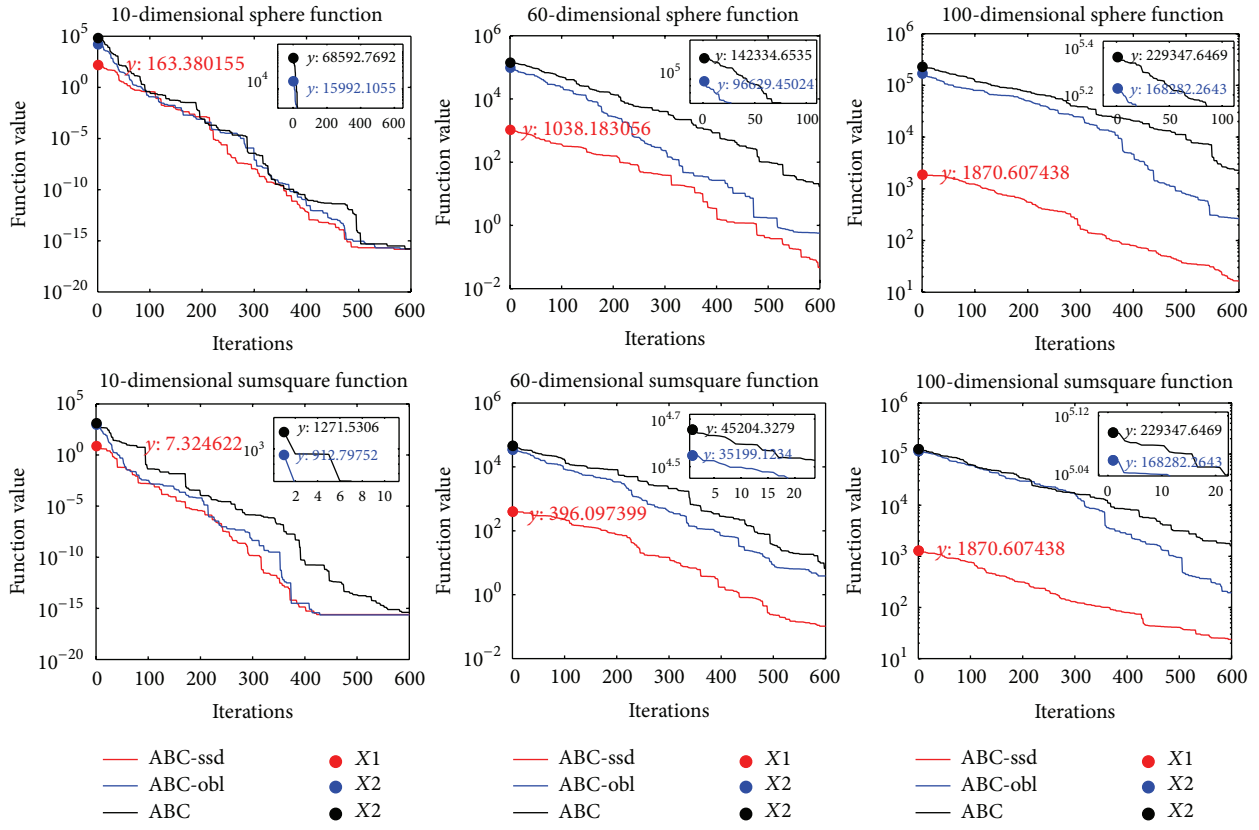
$$x_{ij} = x_{\text{best},j} + \text{rand}(-1, 1)\left(x_{\text{best},j} - x_{\text{abandon},j}\right), \tag{10}$$

where $j \in [1, 2, \ldots, D]$, x_{best} is the best solution achieved till now, and x_{abandon} is the solution abandoned by employed bee. Through the above operations, the main steps of SDABC can be simply presented in Algorithm 4.

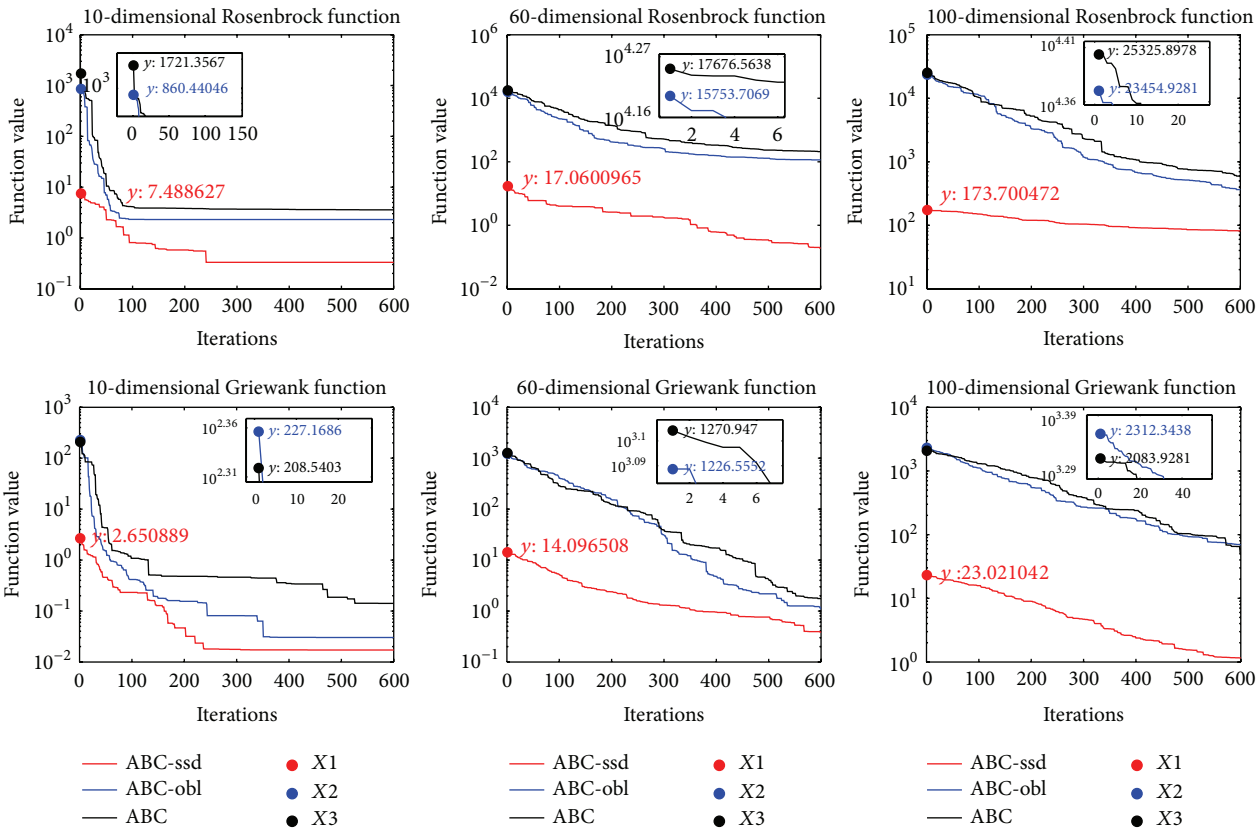
4. Simulation Experiments

In this section, a series of experiments were conducted to certify the effectiveness of our proposed algorithm SDABC in standard benchmark functions. Well-defined benchmark functions which are based on mathematical functions benefit the testing of our proposed algorithm in this paper. As a result, we selected fourteen benchmark functions to fulfill such experiments. The formulations and properties of these benchmark functions were summarized in Table 1. Specifically, $f_1 \sim f_9$ are unimodal functions and $f_{10} \sim f_{14}$ are multimodal functions. Hereinto, f_6 is discontinuous step function, f_{11} is bound-constrained function, and f_3 is a nonconvex function with multiple minima in high dimension case.

It is necessary to emphasize that all the experiments were implemented in the same hardware and software environment. Specifically, computer's hardware configuration is given as follows: an Intel Core 2 Duo processor running at 2.20 GHz, 512 M of RAM, and an 80 G hard driver. The operating system is Microsoft Windows sp3. Our execution was compiled by the default setting of Matlab R2010a.

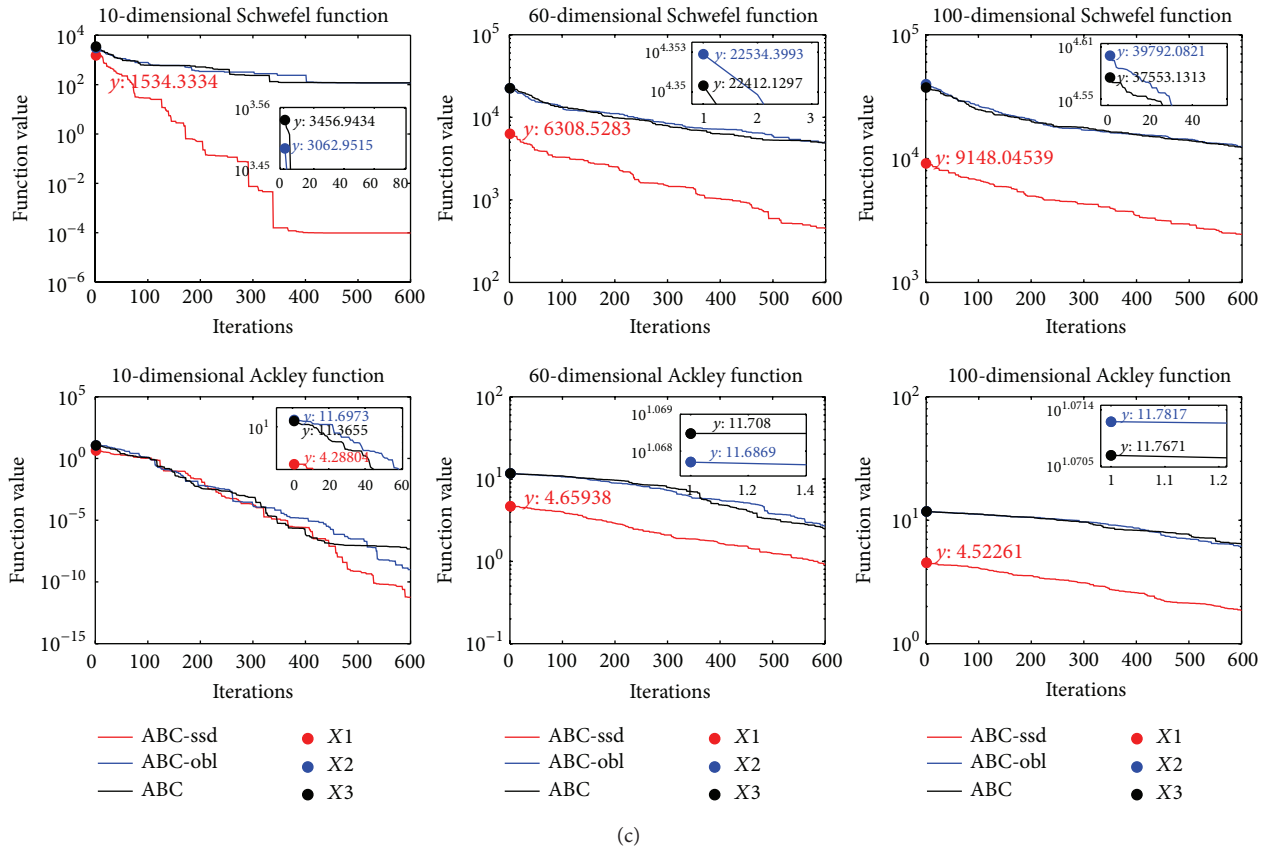


(a)



(b)

Figure 2: Continued.



(c)

Figure 2: Convergence process of convergence process of the best solution values over 100 runs.

4.1. Effects of Each Modification on the Performance of SDABC. In this section, we conducted two types of experiments to analyze each modification, respectively. On the one hand, in order to further certify the superiority of our proposed initialization algorithm, we compared SSD with other initialization algorithms including opposition-based learning (OBL for short) proposed in [21] and random initialization (RI for short). For more clarity, we called the original ABC with our proposed population initialization algorithm (i.e., replacing RI in the original ABC with SSD) as ABC-ssd and the original ABC with OBL as ABC-obl. Note that ABC utilizes RI to generate initial population. In the experiments, the population size was set to be 20, the maximum number of cycles was set to be 600, and the value of *limit* was set to be 150. And then, we compared the convergence performance and final solution accuracy of different initialization methods on a set of six benchmark functions including three unimodal functions and three multimodal functions with $D = 10$, 60, and 100, respectively. For all functions, each algorithm was performed 100 independent runs. The results were shown in Table 2 in terms of mean and standard deviation (Std for short) of solutions obtained by each algorithm. Moreover, the best and the second best solutions were marked in bold and a two-tailed t-test at a 0.05 level of significance was used. Note that the value "+" denotes our proposed algorithm significantly better than other algorithms. It can be observed that ABC-ssd has higher solution accuracy on all functions.

In addition, convergence process of the best solution values over 100 runs on six functions is presented in Figure 2. Hereinto, $X1$, $X2$, and $X3$ denote the best initial solutions generated by SSD, OBL, and RI, respectively. Moreover, some partial enlarged drawings are also provided in Figure 2 to see how much the different population initialization methods make contribution to improving quality of initial solutions. It can be seen clearly that SSD provides the highest quality of initial solution $X1$. For the six benchmark functions with $D = 30, 60$ and 100, SSD is significantly superior to OBL and RI during the whole progress of finding global minimum. There is a tendency that the more the number of optimization parameters is, the more obvious the superiority of SSD shows. Eventually, OBL performs the second best at finding global minimum. Specifically, for unimodal functions, OBL is somewhat superior to RI in terms of initial solution and convergence performance. However, for multimodal functions, the advantages may disappear.

On the other hand, because of correlative dependence, we combined disruptive selection strategy and the improved scout bee to investigate their effects on SDABC in terms of population diversity and convergence performance. To begin with, population diversity is defined as follows [24]:

$$D(X) = \frac{1}{D} \frac{\sum_{i=1}^{D} \left\| X_i - \overline{X} \right\|}{\max_{1 \le i, j \le D} \left\| X_i - X_j \right\|}, \tag{11}$$

Table 2: Mean and Std deviation of solutions obtained by the original ABC with different initialization algorithms.

Function	D	ABC		ABC-obl		ABC-ssd		Significant
		Mean	Std	Mean	Std	Mean	Std	
Sphere	10	$5.24e-16$	$3.12e-16$	**4.41e − 16**	**3.06e − 16**	**2.20e − 16**	**1.01e − 16**	+
	60	30.25	6.77	**0.93**	**0.44**	**8.09e − 2**	**7.31e − 2**	+
	100	2.98 + 3	$4.86e+2$	**3.70e + 2**	**2.45e + 1**	**2.63e + 1**	**3.44**	+
SumSquare	10	$6.33e-16$	$2.61e-16$	**3.00e − 16**	**2.07e − 16**	**2.24e − 16**	**1.85e − 16**	+
	60	9.67	3.48	**6.01**	**2.38**	**3.04e − 1**	**2.76e − 1**	+
	100	$2.68e+3$	$4.09e+2$	**2.51e + 2**	**9.16e + 1**	**3.09**	**1.15**	+
Rosenbrock	10	6.06	3.85	**3.92**	**2.78**	**5.62e − 1**	**8.11e − 2**	+
	60	$3.18e+2$	$1.09e+2$	**1.72e + 2**	**8.45e + 1**	**2.85e − 1**	**7.71e − 2**	+
	100	$8.92e+2$	$3.84e+2$	**4.73e + 2**	**2.66e + 2**	**1.06e + 2**	**4.97e + 1**	+
Griewank	10	$2.81e-1$	$1.39e-1$	**4.47e − 2**	**3.74e − 2**	**2.23e − 2**	**2.45e − 2**	+
	60	3.24	2.37	**1.83**	**1.44**	**5.02e − 1**	**3.80e − 1**	+
	100	**6.62e + 1**	**2.58e + 1**	$7.99e+1$	$3.01e+1$	**2.84**	**1.22**	+
Schwefel	10	**1.18e + 2**	**3.12e − 2**	$1.18e+2$	$4.63e-2$	**4.46e − 3**	**6.69e − 3**	+
	60	**5.96e + 3**	**5.44e + 2**	$6.46e+3$	$1.05e+3$	**4.92e + 2**	**4.09e + 1**	+
	100	**1.23e + 4**	**2.08e + 2**	$1.84e+4$	$3.47e+2$	**4.04e + 3**	**8.99e + 1**	+
Ackley	10	$8.44e-8$	$5.38e-8$	**2.81e − 9**	**9.40e − 9**	**1.03e − 11**	**4.92e − 11**	+
	60	**4.50**	**4.55**	5.82	3.46	**1.65**	**8.08e − 1**	+
	100	$2.86e+1$	8.98	**2.43e + 1**	**8.66**	**3.42**	**1.01**	+

where D is the number of optimization parameters and $\overline{X}$ is the average point which is defined as follows:

$$\overline{X} = \frac{1}{D} \sum_{1 \le i \le D} X_i. \tag{12}$$

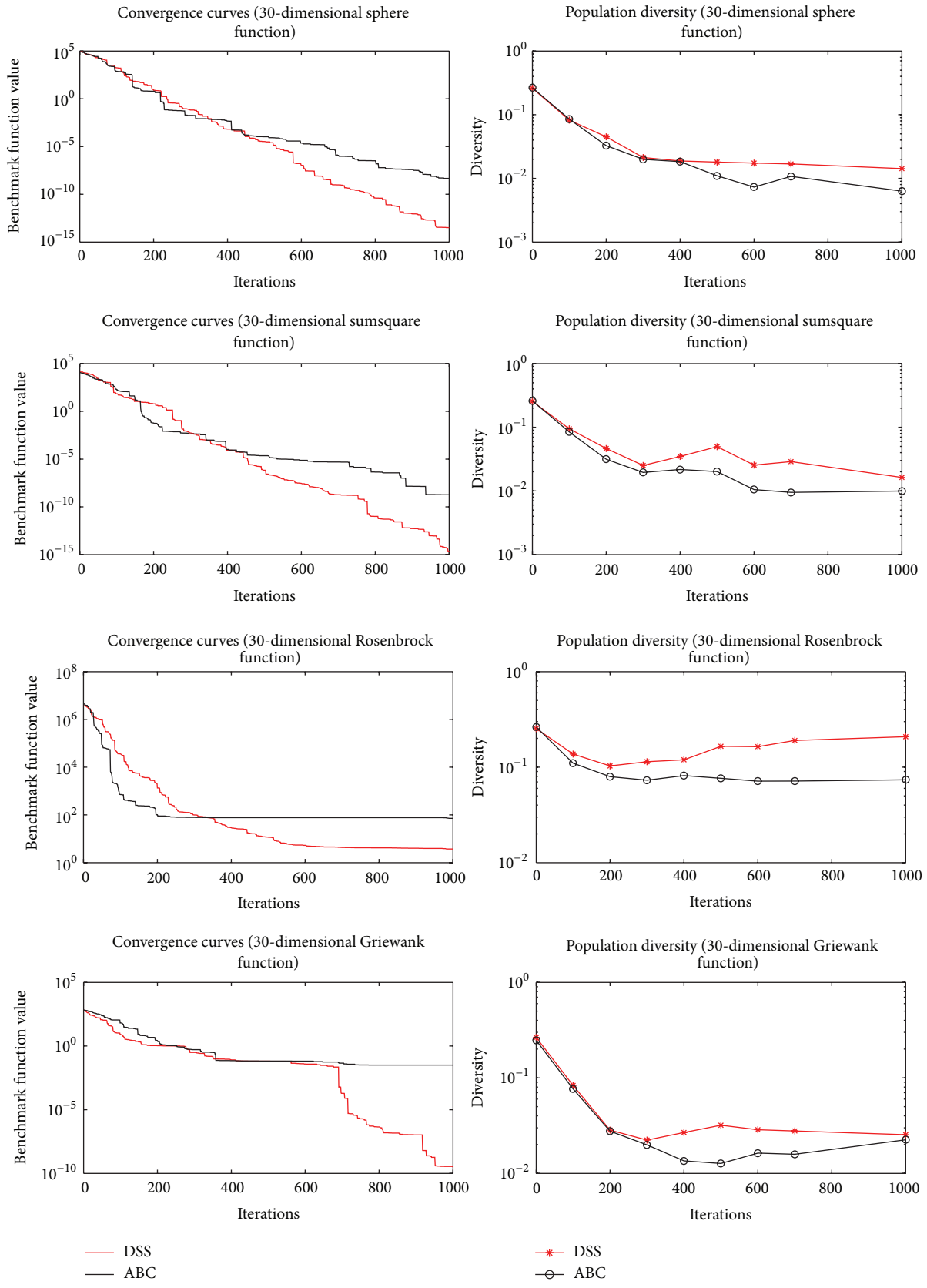
A comparison was conducted between ABC and the standard ABC algorithm with both the disruptive selection strategy and the improved scout bee (DSS for short) on the above six benchmark functions with $D = 30$. The number of population size was set to be 20 and the maximum number of cycles was set to be 1000. Note that the value of *limit* was set to be 150 $(0.5 \times FN \times D)$ for ABC [25] and 20 for DSS. Because of changing the definition of the scout bee, we obtained the value of *limit* through testing again and again several times. For all functions, each algorithm was performed 100 independent runs. The convergence performance of different ABCs and their corresponding population diversity are presented in Figure 3. It can be observed that both the population diversity of ABC and DSS are almost the same as each other in the early stage, this is because the same initialization method is applied to obtain initial solutions. It means that they start with almost the same diversity towards global optimum. However, with the increasing iteration times, DSS shows higher population diversity and better convergence performance than that of ABC. For Rosenbrock function, we note that ABC has a fast convergence initially towards the known local optimum. As the procedure proceeds, ABC falls into the local minimum, while DSS gets closer and closer to the minimum. In a word, the better results are obtained by the combined effects of disruptive selection strategy and the improved scout bee.

4.2. Comparison between ABC and SDABC. In this section, a set of experiments were conducted to testify the efficiency of our proposed algorithm on 14 benchmark functions with $D = 30$ and 100; as presented in Table 1, the experiment results were compared with those of ABC. To allow a fair comparison, all the experiments were performed using the following parameter settings: population size was set to be 20; namely, $FN = 10$, and the maximum number of cycles was set to be 3000. It is necessary to emphasize that the value of *limit* was set to be 200 for ABC and 20 for SDABC, and the reason has been shown in the previous section. For all functions, the algorithms carry out 50 independent runs. For more clarity, the best solutions were marked in bold. The results are shown in Table 3 in terms of the best, worst, mean, and standard deviation of solutions obtained by each algorithm over 50 independent runs. In Table 3, "Std" and "AT" denote the standard deviation of solutions and the average elapsed time, respectively. It can be observed that SDABC has higher accuracy on all the functions than ABC. In particular, SDABC can find optimum solutions on functions f_6, f_8, f_{10}, f_{13}, and f_{14} with $D = 30$. Moreover, the average elapsed time of SDABC on almost all functions is less than that of ABC except functions f_3 with $D = 100$, f_8 and f_{10} with $D = 30$ and 100. However, the superiority of ABC to SDABC is not significantly obvious in terms of the computational time. In other words, SDABC has higher convergence rate and accuracy.

In addition, the statistical comparison of SDABC with ABC uses a two-tailed t-test at a 0.05 level of significance. Note that value "+" in 9th column in Table 3 represents that our proposed algorithm performs significantly better than

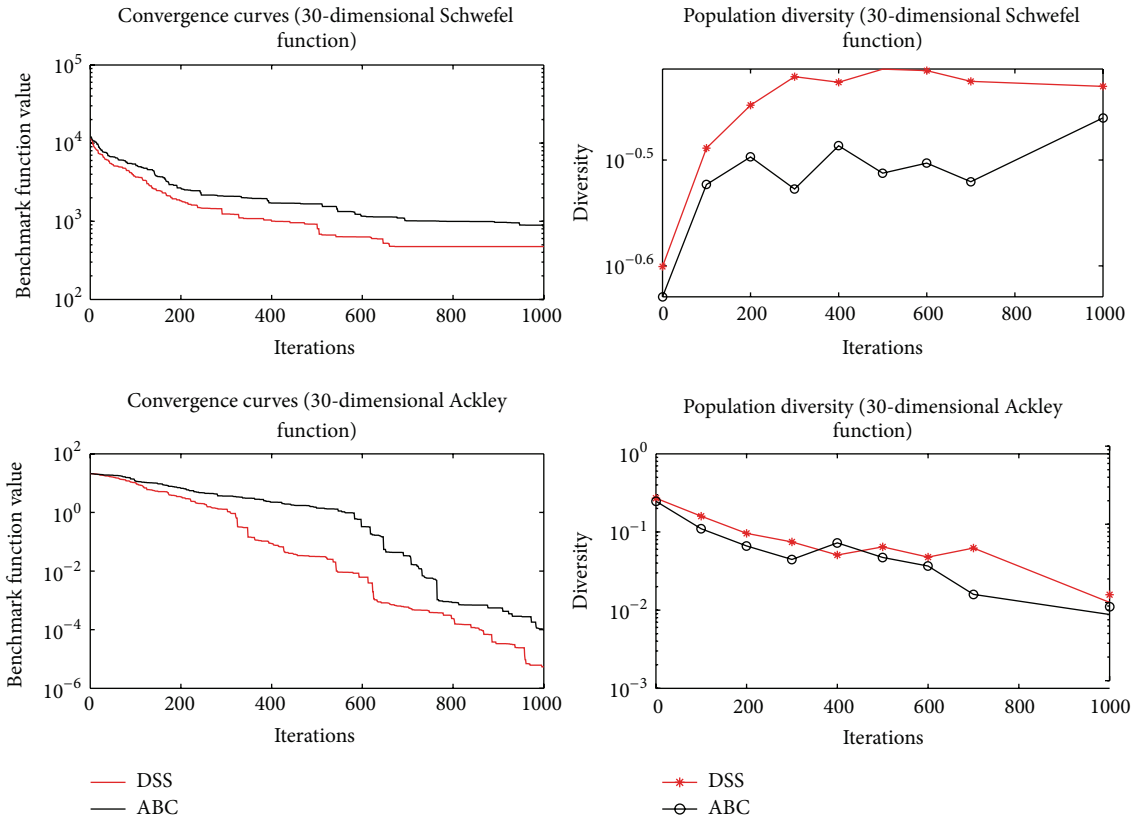
Table 3: Best, worst, mean, Std, and AT deviation of solutions obtained by ABC and SDABC through 50 independent runs on 14 functions.

Function	D	Algorithm	Best	Worst	Mean	Std	AT(s)	Significant
f_1	30	ABC	7.77$e-16$	1.06$e-15$	9.23$e-16$	1.06$e-16$	7.47	+
		SDABC	**4.71e − 42**	**9.69e − 39**	**3.45e − 39**	**3.93e − 39**	**7.29**	
	100	ABC	2.12$e-7$	1.59$e-5$	5.98$e-6$	9.32$e-6$	7.66	+
		SDABC	**2.77e − 10**	**6.84e − 9**	**2.54e − 9**	**2.65e − 9**	**7.62**	
f_2	30	ABC	6.84$e-16$	9.67$e-16$	7.70$e-16$	1.12$e-16$	9.05	+
		SDABC	**3.10e − 44**	**3.42e − 42**	**1.43e − 42**	**1.82e − 42**	**7.29**	
	100	ABC	3.95$e-8$	5.42$e-7$	2.65$e-7$	2.10$e-7$	12.65	+
		SDABC	**1.17e − 11**	**7.74e − 11**	**2.80e − 11**	**2.79e − 11**	**7.52**	
f_3	30	ABC	4.01	20.32	9.85	7.04	8.49	+
		SDABC	**4.96e − 7**	**6.35e − 5**	**2.10e − 5**	**2.48e − 5**	**8.35**	
	100	ABC	1.53$e+2$	2.51$e+2$	2.03$e+2$	3.47$e+1$	**8.55**	+
		SDABC	**4.74e − 2**	**1.05e − 1**	**6.81e − 2**	**2.37e − 2**	8.57	
f_4	30	ABC	6.63	1.55$e+1$	1.13$e+1$	3.74	10.02	+
		SDABC	**0.38**	**1.32**	**0.70**	**0.37**	**7.18**	
	100	ABC	8.27$e+1$	9.01$e+1$	8.73$e+1$	3.00	8.24	+
		SDABC	**1.05e + 1**	**1.40e + 1**	**1.18e + 1**	**1.31**	**8.19**	
f_5	30	ABC	1.32$e-15$	1.87$e-15$	1.52$e-15$	2.20$e-16$	11.66	+
		SDABC	**3.21e − 22**	**2.74e − 21**	**1.37e − 21**	**1.20e − 21**	**11.46**	
	100	ABC	3.35$e-4$	8.56$e-4$	4.16$e-4$	2.21$e-4$	16.24	+
		SDABC	**7.30e − 7**	**1.84e − 6**	**1.10e − 6**	**4.33e − 7**	**16.11**	
f_6	30	ABC	5.47$e-16$	7.23$e-16$	6.74$e-16$	7.34$e-17$	8.32	+
		SDABC	**0**	**0**	**0**	**0**	**8.27**	
	100	ABC	2.35$e-7$	1.20$e-6$	5.74$e-7$	4.51$e-7$	8.74	+
		SDABC	**2.60e − 10**	**1.05e − 9**	**6.49e − 10**	**3.01e − 10**	**8.59**	
f_7	30	ABC	4.34$e-17$	1.06$e-16$	7.06$e-17$	2.52$e-17$	12.13	+
		SDABC	**6.11e − 86**	**1.69e − 78**	**3.39e − 79**	**7.58e − 79**	**12.09**	
	100	ABC	2.59$e-12$	8.27$e-10$	1.95$e-10$	3.56$e-10$	22.23	+
		SDABC	**8.67e − 26**	**9.11e − 20**	**1.82e − 20**	**4.07e − 20**	**22.14**	
f_8	30	ABC	1.79$e-8$	1.39$e-6$	6.38$e-7$	5.51$e-7$	**8.16**	+
		SDABC	**0**	**0**	**0**	**0**	8.21	
	100	ABC	1.65$e-7$	2.10$e-6$	1.16$e-6$	7.39$e-7$	**8.37**	+
		SDABC	**0**	**1.11e − 17**	**8.88e − 18**	**4.96e − 18**	8.43	
f_9	30	ABC	6.99$e-16$	7.49$e-16$	7.31$e-16$	2.05$e-17$	34.88	+
		SDABC	**7.34e − 42**	**3.67e − 39**	**8.95e − 40**	**1.56e − 39**	**34.82**	
	100	ABC	2.72$e-7$	1.52$e-6$	9.38$e-7$	6.28$e-7$	56.84	+
		SDABC	**3.15e − 14**	**4.26e − 12**	**1.43e − 12**	**2.18e − 12**	**54.21**	
f_{10}	30	ABC	9.99$e-16$	1.73$e-2$	4.19$e-3$	7.51$e-3$	**13.97**	+
		SDABC	**0**	**0**	**0**	**0**	14.01	
	100	ABC	1.26$e-7$	1.78$e-2$	2.62$e-3$	9.93$e-3$	**14.91**	+
		SDABC	**1.37e − 9**	**4.95e − 5**	**9.94e − 6**	**2.21e − 5**	14.96	
f_{11}	30	ABC	3.82$e-4$	1.90$e+2$	6.17$e+1$	8.82$e+1$	8.90	+
		SDABC	**3.81e − 4**	**3.81e − 4**	**3.81e − 4**	**8.15e − 13**	**8.86**	
	100	ABC	2.23$e+3$	3.81$e+3$	2.88$e+3$	5.96$e+2$	10.12	+
		SDABC	**1.34e − 3**	**118.45**	**24.25**	**17.64**	**10.06**	
f_{12}	30	ABC	8.35$e-14$	7.41$e-13$	5.99$e-13$	3.89$e-14$	9.51	+
		SDABC	**5.06e − 15**	**8.61e − 14**	**3.62e − 14**	**1.54e − 15**	**9.42**	
	100	ABC	4.98$e-4$	2.03$e-3$	1.22$e-3$	7.05$e-4$	10.25	+
		SDABC	**2.10e − 5**	**7.04e − 5**	**3.66e − 5**	**2.01e − 5**	**10.21**	
f_{13}	30	ABC	5.68$e-14$	2.05$e-4$	6.85$e-5$	1.18$e-4$	7.78	+
		SDABC	**0**	**0**	**0**	**0**	**7.65**	
	100	ABC	7.19	13.16	9.52	2.42	9.23	+
		SDABC	**3.60**	**7.27**	**5.65**	**1.37**	**9.08**	
f_{14}	30	ABC	1.32$e-8$	1.76$e-8$	1.55$e-8$	1.63$e-9$	41.86	+
		SDABC	**0**	**0**	**0**	**0**	**41.60**	
	100	ABC	2.12$e-7$	3.08$e-7$	2.54$e-7$	3.92$e-8$	85.12	+
		SDABC	**4.65e − 8**	**6.47e − 8**	**5.57e − 8**	**7.25e − 9**	**84.35**	



(a)

Figure 3: Continued.



(b)

Figure 3: Population diversity and convergence performance of different ABCs on six benchmark functions.

ABC algorithm. It also indicates that SDABC outperforms ABC.

4.3. Comparison with Other ABCs. In this section, in order to further prove the superiority of our proposed algorithm, we compared SDABC with other state-of-the-art ABCs which are OABC [16], GABC [12], CABC [14], and OOABC [26] on fourteen functions with $D = 30$ presented in Table 1. Given space limitations, this paper cannot fully explore all functions with different dimensions. Population size was set to be 20 and the maximum number of cycles was set to be 3000. The value of limit was set to be 20 for SDABC and 200 for other ABCs. Other parameter settings can be seen in [12, 14, 16, 26]. In addition, we used a two-tailed t-test at a 0.05 level of significance to compare the results obtained by the best and the second best algorithms. For more clarity, the best and the second best solutions were marked in bold. Note that, the value "NA" denotes not applicable, when the best and the second best algorithms achieve the same solution accuracy. The results are shown in Table 4. It can be seen that the performance of SDABC is significantly superior to that of other ABCs on almost all functions except f_6. In case of f_6, both OOABC and SDABC can find optimum solution, which means that they can achieve the same solution accuracy.

5. Conclusions and Future Work

To further improve the performance of ABC, we first proposed a novel initialization method called search space division, which provided high quality of initial solution. Subsequently, a disruptive selection strategy was used to improve population diversity. Moreover, in order to accelerate convergence rate, we changed the definition of the scout bee phase. In addition, we designed two types of experiments to testify our proposed algorithm. On the one hand, we conducted single-factor experiments to make sure of how much each modification makes contribution to improving the performance of ABC. On the other hand, comprehensive experiments were performed to prove the superiority of our proposed algorithm. The experimental results indicate that SDABC significantly outperforms other ABCs, contributing to higher solution accuracy, faster convergence speed, and stronger algorithm stability.

Our future work will focus on two issues. On the one hand, we would apply SDABC to solve the real-world problems such as data mining, industrial design, and clustering. On the other hand, we would extend SDABC to handle the combinational optimization problems.

Table 4: Mean and Std deviation of solutions obtained by SDABC and other ABCs through 50 independent runs on fourteen benchmark functions.

Function	OABC		GABC		CABC		OOABC		SDABC		Significant
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	
f_1	8.53$e-16$	4.65$e-16$	6.73$e-16$	2.01$e-16$	**3.55e − 16**	**1.64e − 16**	5.27$e-16$	2.46$e-16$	**1.03e − 39**	**2.90e − 39**	+
f_2	7.74$e-16$	2.52$e-16$	6.41$e-16$	2.56$e-16$	4.77$e-16$	1.81$e-17$	**3.37e − 16**	**2.45e − 17**	**1.97e − 42**	**1.24e − 42**	+
f_3	7.88	4.92	26.32	5.97	24.65	8.92	**4.34**	**2.07**	**2.96e − 5**	**3.36e − 5**	+
f_4	5.09	1.57	**4.13**	**1.19**	6.99	2.01	10.68	4.37	**3.11e − 1**	**2.92e − 1**	+
f_5	**1.40e − 15**	**2.00e − 16**	1.92$e-15$	3.44$e-16$	1.08$e-13$	1.22$e-13$	2.38$e-14$	1.09$e-14$	**8.63e − 22**	**3.00e − 23**	+
f_6	6.88$e-16$	2.34$e-16$	5.56$e-16$	5.33$e-17$	4.37$e-16$	1.16$e-17$	**0**	**0**	**0**	**0**	NA
f_7	5.85$e-17$	3.85$e-17$	**2.00e − 17**	**7.88e − 18**	2.74$e-17$	4.50$e-18$	3.31$e-16$	5.98$e-16$	**8.56e − 80**	**4.7e − 80**	+
f_8	**1.14e − 7**	**1.37e − 7**	5.83$e-6$	5.26$e-6$	1.32$e-5$	1.80$e-5$	4.56$e-6$	6.67$e-6$	**0**	**0**	+
f_9	5.87$e-16$	1.79$e-16$	5.06$e-16$	4.24$e-17$	**4.66e − 16**	**1.67e − 17**	8.99$e-16$	3.08$e-16$	**8.95e − 40**	**1.56e − 39**	+
f_{10}	**4.42e − 7**	**7.65e − 7**	1.13$e-2$	1.92$e-2$	3.39$e-2$	3.26$e-2$	9.17$e-3$	4.23$e-3$	**0**	**0**	+
f_{11}	**4.22e + 1**	**7.63e + 1**	1.49$e+2$	2.56$e+1$	1.15$e+2$	2.89$e+1$	**5.34e + 1**	**7.87e + 1**	**2.84e − 4**	**6.17e − 13**	+
f_{12}	7.86$e-14$	5.07$e-15$	**4.67e − 14**	**4.42e − 15**	5.72$e-14$	4.19$e-15$	6.61$e-14$	3.32$e-15$	**3.02e − 14**	**1.63e − 15**	+
f_{13}	**5.68e − 14**	**1.01e − 20**	2.05$e-11$	4.07$e-11$	**4.87e − 8**	**3.63e − 8**	4.17$e-5$	6.06$e-5$	**0**	**0**	+
f_{14}	2.23$e-8$	5.96$e-9$	2.10$e-8$	7.90$e-9$	3.47$e-8$	1.20$e-9$	**8.43e − 9**	**2.21e − 9**	**0**	**0**	+

Conflict of Interests

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

References

[1] 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.

[2] 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.

[3] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, pp. 1942–1948, December 1995.

[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] M. Dorigo and T. Stutzle, *Ant Colony Optimization*, MA MIT Press, Cambridge, Mass, USA, 2004.

[6] V. Granville, M. Krivanek, and J.-P. Rasson, "Simulated annealing: a proof of convergence," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 6, pp. 652–656, 1994.

[7] D. Karaboga, "An idea based on honey bee swarm for numerical optimization," Tech. Rep. TR06, Erciyes University, Kayseri, Turkey, 2005.

[8] 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.

[9] S. N. Omkar, J. Senthilnath, R. Khandelwal, G. Narayana Naik, and S. Gopalakrishnan, "Artificial Bee Colony (ABC) for multi-objective design optimization of composite structures," *Applied Soft Computing Journal*, vol. 11, no. 1, pp. 489–499, 2011.

[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] M. F. Tasgetiren, Q.-K. Pan, P. N. Suganthan, and A. H. Chen, "A discrete artificial bee colony algorithm for the total flowtime minimization in permutation flow shops," *Information Sciences*, vol. 181, no. 16, pp. 3459–3475, 2011.

[12] 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.

[13] F. Kang, J. J. Li, and H. J. Li, "Artificial bee colony algorithm and pattern search hybridized for global optimization," *Applied Soft Computing Journal*, vol. 13, no. 4, pp. 1781–1791, 2013.

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

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

[16] E. A. Mohammed, "Generalized opposition-based artificial bee colony algorithm," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '12)*, pp. 1–4, Brisbane, Australia, 2012.

[17] J. Luo, X.-H. Xiao, L. Fu, and Q. Wang, "Modified artificial bee colony algorithm based on segmental-search strategy," *Control and Decision*, vol. 27, no. 9, pp. 1402–1410, 2012.

[18] 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.

[19] D. Karaboga and B. Akay, "A survey: algorithms simulating bee swarm intelligence," *Artificial Intelligence Review*, vol. 31, no. 1–4, pp. 61–85, 2009.

[20] V. Melo and A. C. Delbem, "Investigating Smart Sampling as a population initialization method for Differential Evolution in continuous problems," *Information Sciences*, vol. 193, pp. 36–53, 2012.

[21] R. S. Rahnamayan, H. R. Tizhoosh, and M. M. A. Salama, "Opposition-based differential evolution," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 1, pp. 64–79, 2008.

[22] 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.

[23] T. Kuo and S.-Y. Hwang, "Using disruptive selection to maintain diversity in genetic algorithms," *Applied Intelligence*, vol. 7, no. 3, pp. 257–267, 1997.

[24] C. Zhang and Z. Yi, "Scale-free fully informed particle swarm optimization algorithm," *Information Sciences*, vol. 181, no. 20, pp. 4550–4568, 2011.

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

[26] X. Zhang, X. Zhang, S. Y. Yuen, S. L. Ho, and W. N. Fu, "An improved artificial bee colony algorithm for optimal design of electromagnetic devices," *IEEE Transactions on Magnetics*, vol. 49, no. 8, pp. 4811–4816, 2013.