


Article

An Improved Artificial Bee Colony Algorithm Based on Elite Strategy and Dimension Learning

Songyi Xiao ^{1,2}, Wenjun Wang ³, Hui Wang ^{1,2,*} , Dekun Tan ^{1,2}, Yun Wang ^{1,2}, Xiang Yu ^{1,2} and Runxiu Wu ^{1,2}

¹ School of Information Engineering, Nanchang Institute of Technology, Nanchang 330099, China; speaknow@126.com (S.X.); dktan@nit.edu.cn (D.T.); wangyun@nit.edu.cn (Y.W.); xyuac@ust.hk (X.Y.); wurunxiu@tom.com (R.W.)

² Jiangxi Province Key Laboratory of Water Information Cooperative Sensing and Intelligent Processing, Nanchang Institute of Technology, Nanchang 330099, China

³ School of Business Administration, Nanchang Institute of Technology, Nanchang 330099, China; wangwenjun881@126.com

* Correspondence: huiwang@whu.edu.cn; Tel.: +86-0791-82086956

Received: 19 February 2019; Accepted: 13 March 2019; Published: 21 March 2019



Abstract: Artificial bee colony is a powerful optimization method, which has strong search abilities to solve many optimization problems. However, some studies proved that ABC has poor exploitation abilities in complex optimization problems. To overcome this issue, an improved ABC variant based on elite strategy and dimension learning (called ABC-ESDL) is proposed in this paper. The elite strategy selects better solutions to accelerate the search of ABC. The dimension learning uses the differences between two random dimensions to generate a large jump. In the experiments, a classical benchmark set and the 2013 IEEE Congress on Evolutionary (CEC 2013) benchmark set are tested. Computational results show the proposed ABC-ESDL achieves more accurate solutions than ABC and five other improved ABC variants.

Keywords: Artificial bee colony; swarm intelligence; elite strategy; dimension learning; global optimization

1. Introduction

In many real-world applications various optimization problems exist, which aim to select optimal parameters (variables) to maximize (minimize) performance indicators. In general, a minimization optimization problem can be defined by:

$$\min f(X), \quad (1)$$

where X is the vector of the decision variables.

To effectively solve optimization problems, intelligent optimization methods have been presented. Some representative algorithms are particle swarm optimization [1–5], artificial bee colony (ABC) [6,7], differential evolution [8,9], firefly algorithm [10–13], earthworm optimization algorithm [14], cuckoo search [15,16], moth search [17], pigeon inspired optimization [18], bat algorithm [19–23], krill herd algorithm [24–27], and social network optimization [28]. Among these algorithms, ABC has few control parameters and strong exploration abilities [29,30].

ABC simulates the foraging behaviors of bees in nature [6]. The processes of bees finding food sources are analogous to the processes of searching candidate solutions for a given problem. Although ABC is effective in many problems, it suffers from poor exploitation and slow convergence rates [31,32]. The possible reasons can be summarized in two ways: (1) offspring are in the neighborhood of their corresponding parent solutions and they are near to each other, and (2) offspring and their corresponding parent solutions are similar because of one-dimension perturbation.

In this work, a new ABC variant based on elite strategy and dimension learning (ESDL), called ABC-ESDL, is presented to enhance the performance of ABC. For the elite strategy, better solutions are chosen to guide the search. Moreover, the differences between different dimensions are used to generate candidate solutions with large dissimilarities. In the experiments, a classical benchmark set (with dimensions 30 and 100) and the 2013 IEEE Congress on Evolutionary (CEC 2013) benchmark set are tested. Results of ABC-ESDL are compared with ABC and five other modified ABCs.

The remainder of this work is organized as follows. In Section 2, the concept and definitions of ABC are introduced. Some recent work on ABC is given in Section 3. The proposed ABC-ESDL is described in Section 4. Test problems, results, and discussions are presented in Section 5. Finally, this work is summarized in Section 6.

2. Artificial Bee Colony

Like other bio-inspired algorithms, ABC is also a population-based stochastic method. Bees in the population try to find new food sources (candidate solutions). According to the species of bees, ABC consists of three types of bees: employed bees, onlooker bees, and scouts. The employed bees search the neighborhood of solutions in the current population, and they share their search experiences with the onlooker bees. Then, the onlooker bees choose better solutions and re-search their neighborhoods to find new candidate solutions. When solutions cannot be improved during the search, the scouts randomly initialize them [33].

Let $X_i = (x_{i1}, x_{i2}, \dots, x_{iD})$ be the i -th solution in the population at the t -th iteration. An employed bee randomly selects a different solution X_k from the current population and chooses a random dimension index j . Then, a new solution V_i is obtained by [33]:

$$v_{ij} = x_{ij} + \phi_{ij}(x_{ij} - x_{kj}), \tag{2}$$

where $i = 1, 2, \dots, N$, and ϕ_{ij} is randomly chosen from $[-1.0, 1.0]$. As seen, the new solution V_i is similar to its parent solution X_i , and their differences are only on the j -th dimension. If V_i is better than X_i , X_i is updated by V_i . This means that bees find better solutions during the current search. However, this search process is slow, because the similarities between X_i and V_i are very large.

When employed bees complete the search around the neighborhood for solutions, all solutions will be updated by comparing each pair of $\{X_i, V_i\}$. Then, the selection probability p_i for each X_i is defined as follows [33]:

$$p_i = \frac{fit_i}{\sum_{j=1}^N fit_j}, \tag{3}$$

where fit_i is the fitness value of X_i and fit_i is calculated by:

$$fit_i = \begin{cases} 1/(1 + f_i), & \text{if } f_i \geq 0 \\ 1 + abs(f_i), & \text{otherwise} \end{cases} \tag{4}$$

where f_i is the function value of X_i . It is obvious that a better solution will have a larger selection probability. So, the onlooker bees focus on searching the neighborhoods for better solutions. This may accelerate the convergence.

For a specific solution X , if employed bees and onlooker bees cannot find any new solutions in its neighborhood to replace it, the solutions maybe trapped into local minima. Then, a scout re-initializes it as follows [33]:

$$x_j = L_j + rand_j(U_j - L_j), \tag{5}$$

where $j = 1, 2, \dots, D$, $[L_j, U_j]$ is the search range of the j -th dimension, and $rand_j$ is randomly chosen from $[0, 1.0]$ for the j -th dimension.

3. Related Work

Since the introduction of ABC, many different ABC variants and applications have been proposed. Some recent work on ABC is presented as follows.

Zhu and Kwong [31] modified the search model by introducing the global best solution (*Gbest*). Experiments confirmed that the modifications could improve the search efficiency. Karaboga and Gorkemli [34] presented a quick ABC (qABC) by employing a new solution search equation for the onlooker bees. Moreover, the neighborhood of *Gbest* was used to help the search. Gao and Liu [35] used the mutation operator in differential evolution (DE) to modify the solution search equation of ABC. Wang et al. [32] integrated multiple solution search strategies into ABC. It was expected that the multi-strategy mechanism could balance exploration and exploitation abilities. Cui et al. [36] proposed a new ABC with depth-first search framework and elite-guided search equation (DFSABC-elite), which assigned more computational resources to the better solutions. In addition, elite solutions were incorporated to modify the solution search equation. Li et al. [37] embedded a crossover operator into ABC to obtain a good performance. Yao et al. [38] used a multi-population technique in ABC. The entire population consisted of three subgroups, and each one used different evolutionary operators to play different roles in the search. Kumar and Mishra [39] introduced covariance matrices into ABC. Experiments on comparing continuous optimiser (COCO) benchmarks showed the approach was robust and effective. Yang et al. [40] designed an adaptive encoding learning based on covariance matrix learning. Furthermore, the selection was also adaptive according to the successful rate of candidate solutions. Chen et al. [41] firstly employed multiple different solution search models in ABC. Then, an adaptive method was designed to determine the chosen rate of each model.

In [42], a binary ABC was used to solve the spanning tree construction problem. Compared to the traditional Kruskal algorithm, the binary ABC could find sub-optimal spanning trees. In [43], a hybrid ABC was employed to tackle the effects of over-fitting in high dimensional datasets. In [44], chaos and quantum theory were used to improve the performance of ABC. Dokeroglu et al. [45] used a parallel ABC variant to optimize the quadratic assignment problem. Kishor et al. [46] presented a multi-objective ABC based on non-dominated sorting. A new method was used for employed bees to achieve convergence and diversity. The onlooker bees use similar operations with the standard ABC. Research on wireless sensor networks (WSNs) has attracted much attention [47–49]. Hashim et al. [50] proposed a new energy efficient optimal deployment strategy based on ABC in WSNs, in which ABC was used to optimize the network parameters.

4. Proposed Approach

In this section, a new ABC variant based on elite strategy and dimension learning (ABC-ESDL) is proposed. The proposed strategies and algorithm framework are described in the following subsections.

4.1. Elite Strategy

Many scholars have noticed that the original ABC was not good at exploitation during the search. To tackle this issue, several elite strategies were proposed. It is expected that elite solutions could help the search and save computational resources.

Zhu and Kwong used *Gbest* to modify the solution search model as below [31]:

$$v_{ij} = x_{ij} + \phi_{ij}(x_{ij} - x_{kj}) + \varphi_{ij}(Gbest_j - x_{ij}), \quad (6)$$

where ϕ_{ij} and φ_{ij} are two random values between -1.0 and 1.0 .

Motivated by the mutation strategy of DE, new search equations were designed as follows [32,35]:

$$v_{ij} = Gbest_j + \phi_{ij}(x_{rj} - x_{kj}), \quad (7)$$

$$v_{ij} = Gbest_j + \phi_{ij}(Gbest_j - x_{kj}), \tag{8}$$

where X_r and X_k are two different solutions.

In our previous work [51], an external archive was constructed to store $Gbests$ during the iterations. Then, these $Gbests$ are used to guide the search:

$$v_{ij} = \tilde{A}_j + \phi_{ij}(x_{rj} - x_{kj}), \tag{9}$$

where \tilde{A} is randomly chosen from the external archive.

Similar to [51], Cui et al. [36] designed an elite set E , which stores the best $\rho \cdot N$ solutions in the current population, where $\rho \in (0,1)$. Based on the elite set, two modified search equations are defined as below:

$$v_{ij} = E_{lj} + \phi_{ij}(E_{lj} - x_{kj}), \tag{10}$$

$$v_{ij} = \frac{1}{2}(E_{lj} + Gbest_j) + \phi_{ij}(Gbest_j - x_{kj}), \tag{11}$$

where E_l is randomly chosen from the set E .

Inspired by the above work, a new search model for the employed bees is designed:

$$v_{ij} = \frac{1}{2}(E_{lj} + Gbest_j) + \phi_{ij}(x_{ij} - E_{lj}) + \varphi_{ij}(x_{ij} - Gbest_j), \tag{12}$$

where E_l is randomly chosen from the elite set E , ϕ_{ij} is a random value between -0.5 and 0.5 , and φ_{ij} is a random value between 0 and 1.0 .

As mentioned before, the onlooker bees re-search the neighborhoods of good solutions to find potentially better solutions. Therefore, further searching by the onlooker bees can be regarded as the exploitation phase. How to improve the effectiveness of the onlooker bees is important to the quality of exploitation. Thus, a different method is designed for the onlooker bees:

$$v_{ij} = \frac{1}{2}(E_{mj} + Gbest_j) + \phi_{ij}(x_{ij} - E_{lj}) + \varphi_{ij}(x_{ij} - Gbest_j), \tag{13}$$

where $m = 1, 2, \dots, M$; M is the elite set size; and E_l is randomly chosen from the set E . If a solution X_i is selected based on the probability p_i , an onlooker bee generates M candidate solutions according to Equation (13). Each candidate solution is compared with X_i , and the better one is used as the new X_i . The size of the elite set should be small, because a large M will result in high computational time complexity.

To maintain the size of the elite set E , a simple replacement method is used. Initially, the best M solutions in the population are selected into E . During the search, if the offspring V_i is better than the worst solution E_w in the elite set E , we replace E_w with V_i . Then, the size of E will be M in the whole search.

4.2. Dimensional Learning

In ABC, a random dimension j is selected for conducting the solution search equation. Under this dimension, if their component values are similar, the difference $(x_{ij} - x_{kj})$ will be very small. This means that the step size $(x_{ij} - x_{kj})$ cannot help X_i jump to a far position. If the solution is trapped into local minima, it hardly escapes from the minima. In [52], a concept of dimension learning was proposed. The difference $(x_{ij} - x_{kh})$ between two different dimensions is used as the step size, where j and h are two randomly selected dimension indices and $j \neq h$. In general, the difference between two different dimensions is large. A large step size may help trapped solutions jump to better positions.

Based on the above analysis, dimension learning is embedded into Equations (12) and (13). Then, the new search models are rewritten as below:

$$v_{ij} = \frac{1}{2}(E_{lh} + Gbest_j) + \phi_{ij}(x_{ih} - E_{lj}) + \varphi_{ij}(x_{ih} - Gbest_j), \quad (14)$$

$$v_{ij} = \frac{1}{2}(E_{mj} + Gbest_h) + \phi_{ij}(x_{ij} - E_{lh}) + \varphi_{ij}(x_{ij} - Gbest_h), \quad (15)$$

where h is a random dimension and $j \neq h$.

4.3. Framework of Artificial Bee Colony-Elite Strategy and Dimension Learning

Our approach, ABC-ESDL, consists of four main operations: an elite set updating, an employed bee phase, an onlooker bee phase, and a scout bee phase. The first operation exists in the employed and onlooker bee phases. So, we only present the latter three operations.

In the employed bee phase, for each X_i , a new candidate solution V_i is created by Equation (12). The better one between V_i and X_i is chosen as X_i . If V_i is better than E_w in the elite set E , E_w is replaced by V_i . The procedure of the employed bee phase is presented in Algorithm 1, where FES is the number of function evaluations.

Algorithm 1: Framework of the Employed bee phase

```

Begin
  for  $i = 1$  to  $N$  do
    Generate  $V_i$  by Equation (14);
    Compute  $f(V_i)$  and  $FES = FES + 1$ ;
    if  $f(V_i) < f(X_i)$  then
      Update  $X_i$  by  $V_i$ , and set  $trial_i = 0$ ;
      Update  $E_w$ , if possible;
    else
       $trial_i = trial_i + 1$ ;
    end if
  end for
End

```

The onlooker bee phase is described in Algorithm 2, where $rand(0,1)$ is a random value in the range $[0, 1]$. Compared to the employed bees, a different search model is employed for the onlooker bees. In Algorithm 1, an elite solution E_l is chosen from E randomly, and it is used for generating a new V_i . In Algorithm 2, all elite solutions in E are used to generate M new solutions V_i because there are M elite solutions. All M new solutions are compared with the original X_i , and the best one is used as the new X_i .

Algorithm 2: Framework of the Onlooker bee phase

```

Begin
  Calculate the probability  $p_i$  by Equation (3);
   $I = 1, t = 1$ ;
  while  $t \leq N$  do
    if  $rand(0,1) < p_i$  then
      for  $h = 1$  to  $M$  do
        Generate  $V_i$  by Equation (15);
        Compute  $f(V_i)$  and  $FES = FES + 1$ ;
        if  $f(V_i) < f(X_i)$  then

```

```

    Update  $X_i$  by  $V_i$ , and set  $trial_i = 0$ ;
    Update  $E_w$ , if possible;
  else
     $trial_i = trial_i + 1$ ;
  end if
end for
 $t++$ ;
end if
 $i = (I + 1) \% N + 1$ ;
end while
End

```

When $trial_i$ is set to 0, it means that the solution X_i has been improved. If the value of $trial_i$ exceeds a predefined value $limit$, it means that the solution X_i may fall into local minima. Thus, the current X_i should be reinitialized. The main steps of the scout bee phase are given in Algorithm 3.

Algorithm 3: Framework of the Scout bee phase

```

Begin
  if  $trial_i \geq limit$  then
    Initialize  $X_i$  by Equation (5);
    Compute  $f(X_i)$  and  $FES = FES + 1$ ;
  end if
  Update the global best solution;
End

```

The framework of our approach, ABC-ESDL, is presented in Algorithm 4, where N represents the population size, M is the elite set size, and $MaxFES$ is the maximum value of FES . To clearly illustrate the proposed ABC-ESDL, Figure 1 gives its flowchart.

Algorithm 4: Framework of ABC-ESDL

```

Begin
  Initialize  $N$  solution in the population;
  Initialize the elite set  $E$ ;
  Set  $trial_i = 0, I = 1, 2, \dots, N$ ;
  while  $FES \leq MaxFES$  do
    Execute Algorithm 1;
    Execute Algorithm 2;
    Execute Algorithm 3;
    Update the global best solution;
  end while
End

```

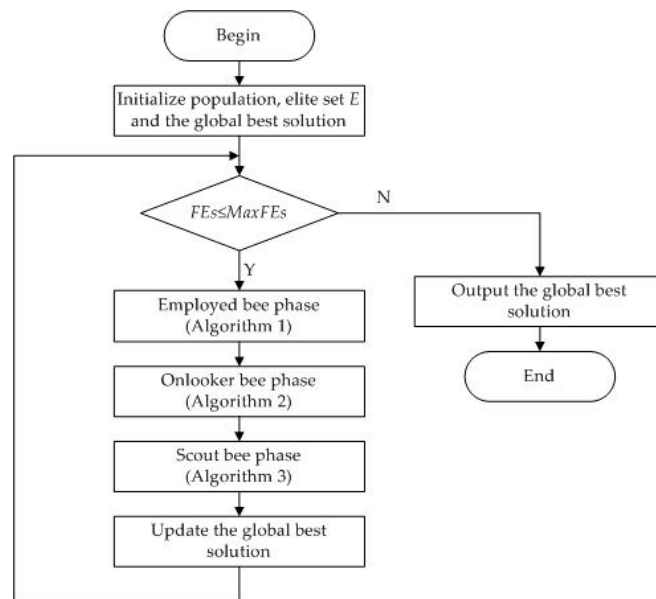


Figure 1. The flowchart of the proposed artificial bee colony-elite strategy and dimension learning (ABC-ESDL) algorithm.

5. Experimental Study

5.1. Test Problems

To verify the performance of ABC-ESDL, 12 benchmark functions with dimensions 30 and 100 were utilized in the following experiments. These functions were employed to test the optimization [53–58]. Table 1 presents the descriptions of the benchmark set where D is the dimension size, and the global optimum is listed in the last column.

Table 1. Benchmark problems.

Name	Function	Global Optimum
Sphere	$f_1(X) = \sum_{i=1}^D x_i^2$	0
Schwefel 2.22	$f_2(X) = \sum_{i=1}^D x_i + \prod_{i=1}^D x_i $	0
Schwefel 1.2	$f_3(X) = \sum_{i=1}^D (\sum_{j=1}^i x_j)^2$	0
Schwefel 2.21	$f_4(X) = \max\{ x_i , 1 \leq i \leq D\}$	0
Rosenbrock	$f_5(X) = \sum_{i=1}^D [100(x_{i+1} - x_i^2)^2 + (1 - x_i^2)^2]$	0
Step	$f_6(X) = \sum_{i=1}^D [x_i + 0.5]$	0
Quartic	$f_7(X) = \sum_{i=1}^D i \cdot x_i^4 + \text{rand}[0, 1]$	0
Schwefel 2.26	$f_8(X) = \sum_{i=1}^D -x_i \sin(\sqrt{ x_i })$	-418.98*D
Rastrigin	$f_9(X) = \sum_{i=1}^D [x_i^2 - 10 \cos 2\pi x_i + 10]$	0
Ackley	$f_{10}(X) = -20 \exp(-0.2 \sqrt{\frac{1}{D} \sum_{i=1}^D x_i^2}) - \exp(\frac{1}{D} \sum_{i=1}^D \cos(2\pi x_i)) + 20 + e$	0
Griewank	$f_{11}(X) = \frac{1}{4000} \sum_{i=1}^D (x_i)^2 - \prod_{i=1}^D \cos(\frac{x_i}{\sqrt{i}}) + 1$	0
Penalized	$f_{12}(X) = \frac{\pi}{D} \left\{ \sum_{i=1}^D (y_i - 1)^2 [1 + \sin(\pi y_i + 1)] + (y_D - 1)^2 + (10 \sin^2(\pi y_1)) \right\}$ $+ \sum_{i=1}^D u(x_i, 10, 100, 4),$ $y_i = 1 + \frac{x_i + 1}{4}$ $u(x_i, a, k, m) = \begin{cases} u(x_i, a, k, m), & x_i > a \\ 0, & -a < x_i < a \\ k(-x_i - a)^m, & x_i < -a \end{cases}$	0

5.2. Parameter Settings

In the experiments, ABC-ESDL was tested on the benchmark set with $D = 30$ and 100 , respectively. Results of ABC-ESDL were compared with several other ABCs. The involved ABCs are listed as follows:

- ABC;
- *Gbest* guided ABC (GABC) [31];
- Improved ABC (IABC) [51];
- Modified ABC (MABC) [35];
- ABC with variable search strategy (ABCVSS) [59];
- ABC with depth-first search framework and elite-guided search equation (DFSABC-elite) [36];
- Our approach, ABC-ESDL.

To attain a fair comparison, the same parameter settings were used. For both $D = 30$ and 100 , N and *limit* were equal to 100 . For $D = 30$, *MaxFEs* was set to 1.5×10^5 . For $D = 100$, *MaxFEs* was set to 5.0×10^5 . The constant value $C = 1.5$ was used in GABC [31]. In MABC, the parameter $p = 0.7$ was used [35]. The archive size m was set to 5 in IABC [51]. The number of solution search equations used in ABCVSS was 5 [59]. In DFSABC-elite, p and r were set to 0.1 and 10 , respectively [36]. In ABC-ESDL, the size (M) of the elite set was set to 5 . All algorithms ran 100 times for each problem. The computing platform was with CPU Intel (R) Core (TM) i5-5200U 2.2 GHz, RAM 4 GB, and Microsoft Visual Studio 2010.

5.3. Comparison between ABC-ESDL and Other ABC Variants

Table 2 shows the results of ABC-ESDL and six other ABCs for $D = 30$, where “Mean” indicates the mean function value and “Std Dev” represents the standard deviation. The term “ $w/t/l$ ” represents a summary for the comparison between ABC-ESDL and the six competitors. The symbol w represents that ABC-ESDL outperformed the compared algorithms on w functions. The symbol l means that ABC-ESDL was worse than its competitor on l functions. For the symbol t , ABC-ESDL and its compared algorithm obtained the same result on t functions. As shown, ABC-ESDL was better than ABC on all functions except for f_6 . For this problem, all ABCs converged to the global minima. Compared to GABC, our approach ABC-ESDL performed better on nine functions. Both of them attained similar results on three functions. For ABC-ESDL, IABC, and ABCVSS, the same performances were achieved on four functions. ABC-ESDL found more accurate solutions than IABC and ABCVSS for the rest of the eight functions. DFSABC-elite outperformed ABC-ESDL on only one function, f_4 , while ABC-ESDL was better than DFSABC-elite on seven functions.

Table 3 lists the results of ABC-ESDL and six other ABCs for $D = 100$. From the results, ABC-ESDL surpassed ABC on all problems. ABC-ESDL, ABC, and IABC retained the same results on f_6 and f_8 . ABC-ESDL obtained better solutions for the rest of the ten functions. Compared to MABC and ABCVSS, ABC-ESDL was better on seven functions. Three algorithms had the same performance on five functions. DFSABC-elite outperformed ABC-ESDL on two functions, but ABC-ESDL was better than DFSABC-elite on five functions. Both of them obtained similar performances on five functions.

Table 2. Results of ABC-ESDL and six other ABC algorithms for $D = 30$.

Functions	ABC		GABC		IABC		MABC		ABCVSS		DFSABC-Elite		ABC-ESDL	
	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev
f_1	1.14×10^{-15}	3.58×10^{-16}	4.52×10^{-16}	2.79×10^{-16}	1.67×10^{-35}	6.29×10^{-36}	9.63×10^{-42}	6.67×10^{-41}	1.10×10^{-36}	3.92×10^{-36}	4.72×10^{-75}	3.17×10^{-74}	2.30×10^{-82}	1.13×10^{-80}
f_2	1.49×10^{-10}	2.34×10^{-10}	1.43×10^{-15}	3.56×10^{-15}	3.09×10^{-19}	3.84×10^{-19}	1.5×10^{-21}	6.64×10^{-22}	8.39×10^{-20}	1.6×10^{-19}	6.01×10^{-38}	2.25×10^{-38}	3.13×10^{-41}	6.81×10^{-40}
f_3	1.05×10^4	3.37×10^3	4.26×10^3	2.17×10^3	5.54×10^3	2.71×10^3	1.48×10^4	1.44×10^4	9.92×10^3	9.36×10^3	4.90×10^3	9.80×10^3	3.61×10^3	1.28×10^3
f_4	4.07×10^1	1.72×10^1	1.16×10^1	6.32×10^0	1.06×10^1	4.26×10^0	5.54×10^{-1}	4.50×10^{-1}	4.36×10^{-1}	3.72×10^{-1}	2.60×10^{-2}	2.99×10^{-2}	2.11×10^{-1}	7.20×10^{-1}
f_5	1.28×10^0	1.05×10^0	2.30×10^{-1}	3.72×10^{-1}	2.36×10^{-1}	3.94×10^{-1}	1.10×10^0	3.45×10^0	1.20×10^0	1.03×10^1	1.58×10^1	1.00×10^2	1.16×10^{-3}	2.08×10^{-2}
f_6	0	0	0	0	0	0	0	0	0	0	0	0	0	0
f_7	1.54×10^{-1}	2.93×10^{-1}	5.63×10^{-2}	3.66×10^{-2}	4.23×10^{-2}	3.02×10^{-2}	2.77×10^{-2}	6.36×10^{-3}	3.25×10^{-2}	4.72×10^{-2}	1.64×10^{-2}	2.42×10^{-2}	1.46×10^{-2}	2.64×10^{-2}
f_8	-12,490.5	$5.87 \times 10^{+1}$	-12,569.5	3.25×10^{-10}	-12,569.5	1.31×10^{-10}	-12,569.5	1.97×10^{-13}	-12,569.5	1.94×10^{-11}	-12,569.5	1.97×10^{-11}	-12,569.5	4.65×10^{-11}
f_9	7.11×10^{-15}	2.28×10^{-15}	0	0	0	0	0	0	0	0	0	0	0	0
f_{10}	1.60×10^{-9}	4.32×10^{-9}	3.97×10^{-14}	2.83×10^{-14}	3.61×10^{-14}	1.76×10^{-14}	7.07×10^{-14}	2.36×10^{-14}	3.02×10^{-14}	2.04×10^{-14}	2.87×10^{-14}	1.46×10^{-14}	2.82×10^{-14}	2.00×10^{-14}
f_{11}	1.04×10^{-13}	3.56×10^{-13}	1.12×10^{-16}	2.53×10^{-16}	0	0	0	0	1.85×10^{-17}	3.87×10^{-16}	2.05×10^{-11}	6.04×10^{-10}	0	0
f_{12}	5.46×10^{-16}	3.46×10^{-16}	4.03×10^{-16}	2.39×10^{-16}	3.02×10^{-17}	0	1.57×10^{-32}	4.50×10^{-47}	1.57×10^{-32}	4.50×10^{-47}	1.57×10^{-32}	4.50×10^{-47}	1.57×10^{-32}	5.81×10^{-47}
$w/t/l$	11/1/0		9/3/0		8/4/0		7/5/0		8/4/0		7/4/1		-	

* The best result for each function is shown in boldface.

Table 3. Results of ABC-ESDL and six other ABC algorithms for $D = 100$.

Functions	ABC		GABC		IABC		MABC		ABCVSS		DFSABC-Elite		ABC-ESDL	
	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev
f_1	7.42×10^{-15}	5.89×10^{-15}	3.37×10^{-15}	7.52×10^{-16}	3.23×10^{-33}	1.45×10^{-34}	7.98×10^{-38}	2.17×10^{-37}	6.18×10^{-35}	1.84×10^{-34}	1.04×10^{-73}	1.09×10^{-72}	6.82×10^{-85}	3.06×10^{-83}
f_2	1.09×10^{-9}	4.56×10^{-9}	6.54×10^{-15}	2.86×10^{-15}	4.82×10^{-18}	3.53×10^{-18}	2.68×10^{-20}	3.49×10^{-20}	1.18×10^{-18}	1.47×10^{-18}	2.80×10^{-37}	5.35×10^{-37}	1.02×10^{-52}	1.76×10^{-51}
f_3	1.13×10^5	2.62×10^4	9.28×10^4	2.71×10^4	9.76×10^4	2.81×10^4	1.58×10^5	9.39×10^4	1.10×10^5	5.12×10^4	6.42×10^4	5.17×10^4	7.82×10^4	8.55×10^4
f_4	8.91×10^1	4.37×10^1	8.37×10^1	3.68×10^1	8.29×10^1	1.28×10^1	3.88×10^1	3.70×10^0	3.82×10^0	1.09×10^0	7.32×10^{-1}	1.01×10^0	2.66×10^1	1.32×10^1
f_5	3.46×10^0	4.29×10^0	2.08×10^1	3.46×10^0	2.97×10^0	2.72×10^0	2.31×10^0	2.62×10^0	1.29×10^1	1.23×10^2	2.07×10^1	8.46×10^1	1.92×10^{-3}	3.22×10^{-2}
f_6	1.58×10^0	1.68×10^0	0	0	0	0	0	0	0	0	0	0	0	0
f_7	1.96×10^0	2.57×10^0	9.70×10^{-1}	7.32×10^{-1}	7.45×10^{-1}	2.27×10^{-1}	1.75×10^{-1}	1.67×10^{-2}	1.44×10^{-1}	1.72×10^{-1}	1.44×10^{-1}	8.16×10^{-2}	8.34×10^{-2}	9.15×10^{-2}
f_8	-40,947.5	7.34×10^2	-41,898.3	5.68×10^{-10}	-41,898.3	3.21×10^{-10}	-41,898.3	2.91×10^{-12}	-41,898.3	1.60×10^{-10}	-41,898.3	7.02×10^{-11}	-41,898.3	1.63×10^{-10}
f_9	1.83×10^{-11}	2.27×10^{-11}	1.95×10^{-14}	3.53×10^{-14}	1.42×10^{-14}	2.63×10^{-14}	0	0	0	0	0	0	0	0
f_{10}	3.54×10^{-9}	7.28×10^{-10}	1.78×10^{-13}	5.39×10^{-13}	1.50×10^{-13}	4.87×10^{-13}	3.58×10^{-11}	2.91×10^{-12}	1.32×10^{-13}	3.64×10^{-14}	1.25×10^{-13}	5.36×10^{-14}	1.25×10^{-13}	5.61×10^{-14}
f_{11}	1.12×10^{-14}	9.52×10^{-15}	1.44×10^{-15}	3.42×10^{-15}	7.78×10^{-16}	5.24×10^{-16}	0	0	0	0	1.81×10^{-16}	3.42×10^{-15}	0	0
f_{12}	4.96×10^{-15}	3.29×10^{-15}	2.99×10^{-15}	4.37×10^{-15}	9.05×10^{-18}	0	4.71×10^{-33}	0	4.71×10^{-33}	0	4.71×10^{-33}	0	4.71×10^{-33}	0
$w/t/l$	12/0/0		10/2/0		10/2/0		7/5/0		7/5/0		5/5/2		-	

* The best result for each function is shown in boldface.

Figure 2 presents the convergence processes of ABC-ESDL, DFSABC-elite, MABC, and ABC on selected problems with $D = 30$. As seen, ABC-ESDL was faster than DFSABC-elite, MABC, and ABC. For f_1, f_2, f_{10} , and f_{12} , DFSABC-elite converged faster than MABC and ABC. For f_5 , DFSABC-elite was the slowest algorithm. ABC was faster than DFSABC-elite on f_7 . For f_{10} , ABC-ESDL was slower than DFSABC-elite at the beginning search stage, and it was faster at the last search stage.

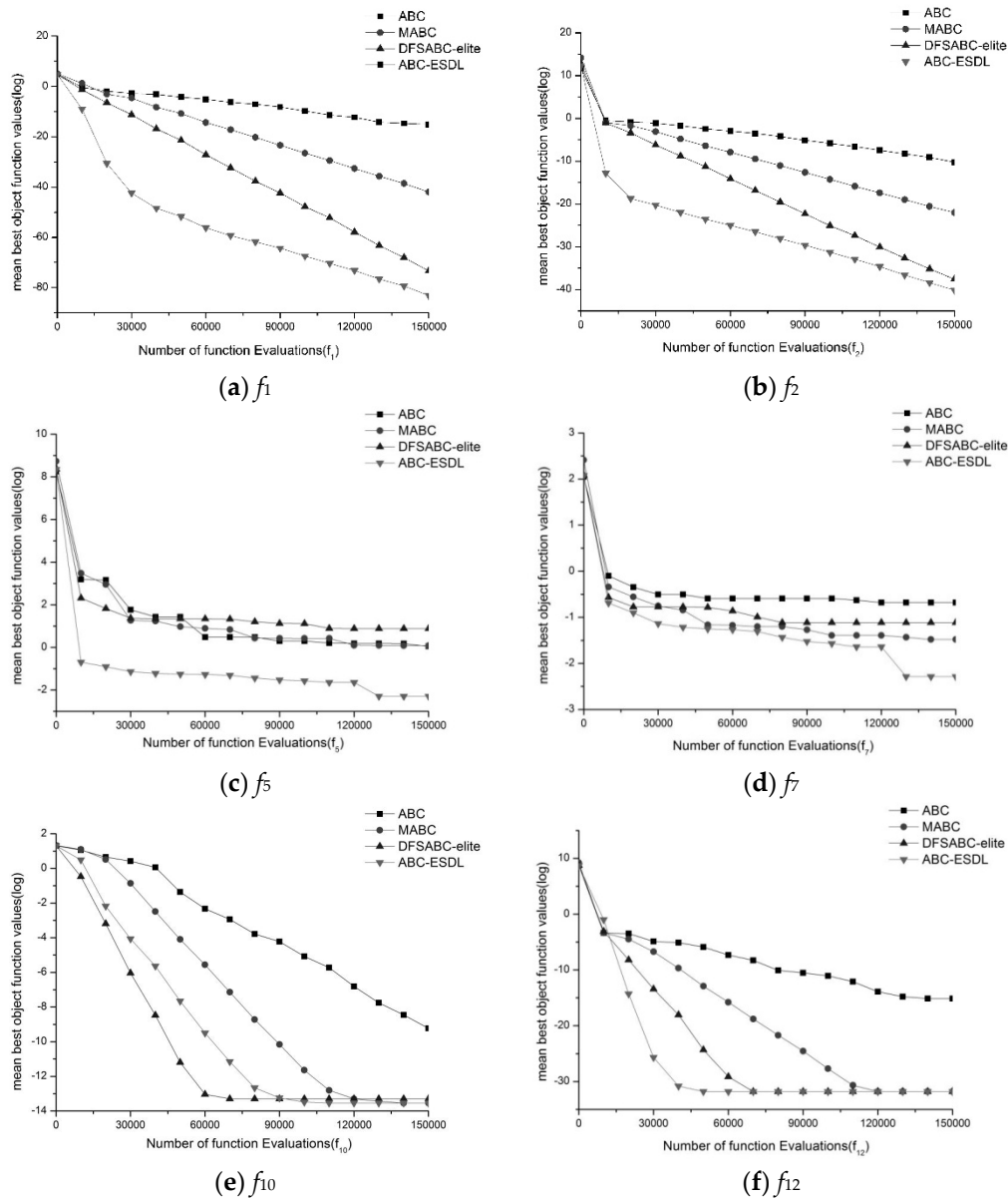


Figure 2. The convergence curves of ABC-ESDL, DFSABC-elite, MABC, and ABC on selected functions. (a) Sphere; (b) Schwefel 2.22; (c) Rosenbrock; (d) Quartic; (e) Ackley; and (f) Penalized.

By the suggestions of [53,56], a nonparametric statistical test was used to compare the overall performances of seven ABCs. In the following, the mean rank of each algorithm on the whole benchmark set was calculated by the Friedman test. Table 4 gives the mean rank values of seven ABCs for $D = 30$ and 100. The smallest rank value meant that the corresponding algorithm obtained the best performance. For $D = 30$ and 100, ABC-ESDL achieved the best performances, and DFSABC-elite was in second place. For $D = 30$, both MABC and ABCVSS had the same rank. When the dimension increased to 100, ABCVSS obtained a better rank than MABC.

Table 4. Mean ranks achieved by the Friedman test for $D = 30$ and 100 .

Algorithms	Mean Rank	
	$D = 30$	$D = 100$
ABC	6.50	6.67
GABC	4.58	5.33
IABC	4.08	4.42
MABC	3.79	3.58
ABCVSS	3.79	3.29
DFSABC-elite	3.29	2.67
ABC-ESDL	1.96	2.04

* The best rank for each dimension is shown in boldface.

5.4. Effects of Different Strategies

There are two modifications in ABC-ESDL: elite strategy (ES) and dimension learning (DL). To investigate the effects of different strategies (ES and DL), we tested different combinations between ABC, ES, and DL on the benchmark set. The involved combinations are listed as below:

- ABC without ES or DL;
- ABC-ES: ABC with elite strategy;
- ABC-DL: ABC with dimension learning;
- ABC-ESDL: ABC with elite strategy and dimension learning.

For the above four ABC algorithms, the parameter settings were kept the same as in Section 5.3. The parameters $MaxFEs$, N , $limit$, and M were set to $5000 \times D$, 100, 100, and 5, respectively. All algorithms ran 100 times for each problem for $D = 30$ and 100 .

Table 5 presents the comparison of ABC-ESDL, ABC-ES, ABC-DL, and ABC for $D = 30$. The best result for each function is shown in boldface. From the results, all four algorithms obtained the same results on f_6 . ABC was worse than ABC-ES on eight problems, but ABC-ES obtained worse results on three problems. ABC-DL outperformed ABC on ten problems, while ABC-DL was worse than ABC on only one problem. ABC-ESDL outperformed ABC-DL and ABC on 11 problems. Compared to ABC-ES, ABC-ESDL was better on ten problems, and both of them had the same performances on the rest of the two problems.

Table 5. Comparison of ABC with different strategies ($D = 30$).

Problems	ABC		ABC-ES		ABC-DL		ABC-ESDL	
	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev
f_1	1.14×10^{-15}	3.58×10^{-16}	1.37×10^{-33}	2.51×10^{-34}	4.67×10^{-17}	4.78×10^{-17}	2.30×10^{-82}	1.13×10^{-80}
f_2	1.49×10^{-10}	2.34×10^{-10}	2.82×10^{-21}	3.23×10^{-21}	1.02×10^{-10}	3.46×10^{-11}	3.13×10^{-41}	6.81×10^{-40}
f_3	1.05×10^4	3.37×10^3	6.71×10^3	2.94×10^3	7.62×10^3	3.27×10^3	3.61×10^3	1.28×10^3
f_4	4.07×10^1	1.72×10^1	2.21×10^0	2.06×10^0	3.82×10^1	1.24×10^1	2.11×10^{-1}	7.20×10^{-1}
f_5	1.28×10^0	1.05×10^0	3.88×10^1	1.65×10^1	9.63×10^{-2}	1.09×10^{-2}	1.16×10^{-3}	2.08×10^{-2}
f_6	0	0	0	0	0	0	0	0
f_7	1.54×10^{-1}	2.93×10^{-1}	9.40×10^{-2}	1.77×10^{-2}	2.82×10^{-1}	2.51×10^{-2}	1.46×10^{-2}	2.64×10^{-2}
f_8	-12,490.5	$5.87 \times 10^{+1}$	-12557.8	1.62×10^1	-12,533.1	1.93×10^2	-12,569.5	4.65×10^{-11}
f_9	7.11×10^{-15}	2.28×10^{-15}	7.94×10^{-14}	2.58×10^{-15}	2.43×10^{-15}	0	0	0
f_{10}	1.60×10^{-9}	4.32×10^{-9}	3.49×10^{-14}	1.87×10^{-14}	6.45×10^{-10}	1.99×10^{-14}	2.82×10^{-14}	2.00×10^{-14}
f_{11}	1.04×10^{-13}	3.56×10^{-13}	7.55×10^{-3}	6.38×10^{-3}	2.49×10^{-15}	1.52×10^{-15}	0	0
f_{12}	5.46×10^{-16}	3.46×10^{-16}	1.57×10^{-32}	0	1.56×10^{-19}	0	1.57×10^{-32}	5.81×10^{-47}
$w/t/l$	11/1/0		10/2/0		11/1/0		-	

* The best result for each function is shown in boldface.

Table 6 gives the results of ABC-ESDL, ABC-ES, ABC-DL, and ABC for $D = 100$. The best result for each function is shown in boldface. Similar to $D = 30$, we can get the same conclusion. ABC-ESDL performed better than ABC, ABC-ES, and ABC-DL. ABC-ES was better than ABC-DL on most test problems, and both of them outperformed the original ABC.

Table 6. Comparison of ABC with different strategies ($D = 100$).

Problems	ABC		ABC-ES		ABC-DL		ABC-ESDL	
	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev
f_1	7.42×10^{-15}	5.89×10^{-15}	1.53×10^{-27}	2.87×10^{-26}	3.96×10^{-15}	1.91×10^{-14}	6.82×10^{-85}	3.06×10^{-83}
f_2	1.09×10^{-9}	4.56×10^{-9}	1.11×10^{-16}	1.18×10^{-15}	9.41×10^{-10}	1.86×10^{-9}	1.02×10^{-52}	1.76×10^{-51}
f_3	1.13×10^5	2.62×10^4	9.39×10^4	2.63×10^4	1.08×10^5	3.91×10^4	7.82×10^4	8.55×10^4
f_4	8.91×10^1	4.37×10^1	3.44×10^1	3.44×10^1	8.73×10^1	7.92×10^0	2.66×10^1	1.32×10^1
f_5	3.46×10^0	4.29×10^0	1.19×10^2	1.19×10^2	2.21×10^{-1}	1.83×10^0	1.92×10^{-3}	3.22×10^{-2}
f_6	1.58×10^0	1.68×10^0	0	0	3.13×10^0	6.59×10^0	0	0
f_7	1.96×10^0	2.57×10^0	1.87×10^{-1}	1.35×10^{-1}	1.43×10^0	1.04×10^0	8.34×10^{-2}	9.15×10^{-2}
f_8	$-40,947.5$	7.34×10^2	$-41,762.1$	5.95×10^2	$-41,240.7$	8.02×10^2	$-41,898.3$	1.63×10^{-10}
f_9	1.83×10^{-11}	2.27×10^{-11}	1.29×10^{-9}	3.78×10^{-8}	2.07×10^{-6}	5.32×10^{-5}	0	0
f_{10}	3.54×10^{-9}	7.28×10^{-10}	1.57×10^{-13}	4.35×10^{-14}	2.17×10^{-9}	5.06×10^{-9}	1.25×10^{-13}	5.61×10^{-14}
f_{11}	1.12×10^{-14}	9.52×10^{-15}	9.13×10^{-4}	1.49×10^{-2}	1.89×10^{-15}	7.52×10^{-15}	0	0
f_{12}	4.96×10^{-15}	3.29×10^{-15}	4.29×10^{-28}	8.78×10^{-27}	3.21×10^{-18}	2.19×10^{-17}	4.71×10^{-33}	7.50×10^{-48}
$w/t/l$	12/0/0		11/1/0		12/0/0		-	

* The best result for each function is shown in boldface.

For the above analysis, ABC with a single strategy (ES or DL) achieved better results than the original ABC. By introducing ES and DL into ABC, the performance of ABC-ESDL was further enhanced, and it outperformed ABC and ABC with a single strategy. This demonstrated that both ES and DL were helpful in strengthening the performance of ABC.

5.5. Results of the CEC 2013 Benchmark Set

In Section 5.3, ABC-ESDL was tested on several classical benchmark functions. To verify the performance of ABC-ESDL on difficult functions, the 2013 IEEE Congress on Evolutionary (CEC 2013) benchmark set was utilized in this section [60].

In the experiments, ABC-ESDL was compared with ABC, GABC, MABC, ABCVSS, and DFSABC-elite on the CEC benchmark set with $D = 30$. By the suggestions of [60], $MaxFEs$ was set to $10,000 \times D$. For other parameters, the same settings were used as described in Section 5.3. For each test function, each algorithm was run 51 times. Throughout the experiments, the mean function error value ($f(X) - f(X^*)$) was reported, where X was the best solution found by the algorithm in a run, and X^* was the global optimum of the test function [60].

Table 7 presents the computational results of ABC-ESDL, DFSABC-elite, ABCVSS, MABC, GABC, and ABC on the 2013 IEEE Congress on Evolutionary (CEC 2013) benchmark set, where “Mean” indicates the mean function error values and “Std Dev” represents the standard deviation. The best result for each function is shown in boldface. From the results, ABC-ESDL outperformed ABC and GABC on 25 functions, but it was worse on the rest of the three functions. Compared to MABC, ABC-ESDL achieved better results on 20 functions, but MABC was better than ABC-ESDL on the rest of the eight functions. ABC-ESDL performed better than ABCVSS and DFSABC-elite on 21 and 22 functions, respectively. From the above analysis, even for difficult functions, ABC-ESDL still obtained better performances than the compared algorithms.

Table 7. Results on the CEC 2013 benchmark set.

Problems	ABC		GABC		MABC		ABCVSS		DFSABC-elite		ABC-ESDL	
	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev
f_1	6.82×10^{-14}	2.18×10^{-13}	5.71×10^{-13}	9.44×10^{-14}	4.55×10^{-14}	1.33×10^{-12}	6.82×10^{-14}	1.88×10^{-12}	4.55×10^{-14}	1.19×10^{-12}	7.29×10^{-5}	1.66×10^{-3}
f_2	1.05×10^5	5.86×10^6	3.43×10^7	5.63×10^6	1.88×10^7	4.34×10^7	2.54×10^5	7.24×10^5	1.98×10^7	4.55×10^7	2.55×10^6	1.16×10^6
f_3	2.49×10^9	6.24×10^9	1.05×10^{10}	1.19×10^9	5.58×10^7	2.94×10^7	1.89×10^8	5.60×10^8	7.83×10^7	2.26×10^7	1.95×10^6	3.08×10^6
f_4	6.81×10^3	2.09×10^3	3.51×10^5	1.48×10^4	9.83×10^3	2.56×10^3	8.18×10^3	2.05×10^3	6.58×10^3	1.79×10^3	5.94×10^3	1.81×10^5
f_5	4.97×10^{-10}	8.53×10^{-9}	4.70×10^{-13}	5.65×10^{-14}	5.68×10^{-14}	1.52×10^{-12}	1.71×10^{-13}	6.98×10^{-12}	1.02×10^{-13}	2.50×10^{-12}	1.07×10^{-3}	3.00×10^{-3}
f_6	1.73×10^0	5.07×10^0	1.77×10^2	1.82×10^0	1.76×10^0	6.81×10^0	2.25×10^0	7.12×10^0	1.67×10^0	1.14×10^0	8.58×10^{-1}	3.57×10^{-1}
f_7	1.29×10^2	3.58×10^1	4.09×10^2	1.29×10^1	1.06×10^1	3.52×10^1	1.26×10^1	3.96×10^1	9.27×10^0	2.53×10^0	7.16×10^0	2.15×10^0
f_8	2.10×10^0	5.96×10^0	2.13×10^1	3.51×10^{-2}	2.09×10^0	5.97×10^0	2.11×10^0	5.97×10^0	2.10×10^0	5.96×10^0	2.08×10^0	5.95×10^0
f_9	3.02×10^1	8.69×10^0	1.40×10^2	2.43×10^1	2.79×10^1	8.69×10^0	3.05×10^1	8.81×10^0	3.04×10^1	8.47×10^0	2.97×10^1	8.08×10^0
f_{10}	3.40×10^{-1}	8.22×10^{-1}	1.43×10^0	8.48×10^{-1}	1.62×10^{-1}	4.60×10^{-1}	3.05×10^{-1}	1.32×10^{-1}	2.46×10^{-1}	5.59×10^{-1}	2.51×10^{-2}	6.91×10^{-2}
f_{11}	3.30×10^{-13}	4.73×10^{-13}	1.54×10^{-13}	2.86×10^{-14}	1.14×10^{-14}	3.24×10^{-14}	1.71×10^{-14}	4.30×10^{-14}	5.68×10^{-15}	2.50×10^{-15}	6.81×10^{-4}	1.91×10^{-4}
f_{12}	3.14×10^1	8.42×10^1	1.60×10^3	5.64×10^1	1.57×10^1	5.52×10^0	2.46×10^1	6.01×10^1	2.20×10^1	5.61×10^0	1.51×10^1	4.91×10^0
f_{13}	3.14×10^1	9.36×10^0	1.81×10^3	5.60×10^1	2.63×10^1	7.29×10^0	2.27×10^1	7.64×10^0	2.13×10^1	6.65×10^0	2.70×10^1	7.51×10^0
f_{14}	1.09×10^0	4.05×10^0	2.85×10^0	1.28×10^0	2.48×10^{-1}	5.23×10^{-1}	6.25×10^{-3}	1.14×10^{-2}	2.10×10^{-2}	1.80×10^{-2}	7.47×10^{-1}	3.05×10^{-1}
f_{15}	3.49×10^3	1.22×10^2	1.57×10^4	6.11×10^2	3.21×10^3	1.06×10^2	2.65×10^3	1.28×10^2	5.09×10^3	1.40×10^2	2.62×10^3	1.10×10^2
f_{16}	1.65×10^0	5.11×10^0	2.07×10^0	2.57×10^{-1}	1.57×10^0	3.98×10^0	2.15×10^0	5.66×10^0	2.49×10^0	5.97×10^0	8.49×10^{-1}	3.75×10^{-1}
f_{17}	3.11×10^0	8.81×10^1	1.07×10^2	1.06×10^2	3.04×10^0	8.67×10^0	3.04×10^0	8.66×10^0	3.27×10^0	8.66×10^0	3.09×10^0	8.76×10^0
f_{18}	3.88×10^2	1.01×10^2	1.76×10^3	5.02×10^2	1.90×10^2	6.59×10^1	3.45×10^2	9.27×10^1	2.84×10^2	7.53×10^1	1.44×10^2	4.99×10^1
f_{19}	1.07×10^{-1}	3.67×10^{-1}	2.25×10^0	2.94×10^{-1}	6.81×10^{-2}	2.26×10^{-2}	1.54×10^{-1}	6.58×10^{-1}	4.56×10^{-2}	2.29×10^{-2}	4.49×10^{-2}	9.31×10^{-2}
f_{20}	1.54×10^1	4.17×10^0	5.00×10^1	6.93×10^0	1.46×10^1	4.11×10^0	1.48×10^1	4.11×10^0	1.46×10^1	4.02×10^0	1.41×10^1	3.82×10^0
f_{21}	2.01×10^2	5.59×10^1	3.67×10^2	9.04×10^1	2.06×10^2	5.77×10^1	2.18×10^2	6.27×10^1	2.00×10^2	9.20×10^1	1.02×10^2	5.31×10^1
f_{22}	1.16×10^2	3.71×10^1	7.47×10^1	2.64×10^1	1.05×10^2	3.06×10^1	1.15×10^2	4.13×10^1	1.19×10^2	3.20×10^1	1.47×10^1	2.04×10^0
f_{23}	5.53×10^3	1.52×10^2	2.16×10^4	8.78×10^3	4.11×10^3	1.35×10^2	5.57×10^3	1.63×10^2	5.87×10^3	1.74×10^2	3.18×10^3	1.26×10^2
f_{24}	3.02×10^2	8.33×10^1	6.00×10^2	7.56×10^1	2.88×10^2	8.13×10^1	2.86×10^2	8.27×10^1	2.86×10^2	8.09×10^1	2.80×10^2	8.07×10^1
f_{25}	3.15×10^2	8.88×10^1	7.16×10^2	9.05×10^0	2.96×10^2	8.53×10^1	3.02×10^2	8.67×10^1	3.00×10^2	8.53×10^1	3.00×10^2	8.56×10^1
f_{26}	2.01×10^2	5.73×10^1	2.07×10^2	3.95×10^{-1}	2.01×10^2	5.72×10^1	2.01×10^2	5.73×10^1	2.01×10^2	5.72×10^1	2.00×10^2	5.71×10^1
f_{27}	4.02×10^2	1.34×10^1	3.81×10^3	6.27×10^2	1.11×10^3	3.00×10^2	4.02×10^2	1.90×10^1	4.02×10^2	1.14×10^1	4.00×10^2	1.26×10^1
f_{28}	1.64×10^2	7.10×10^1	4.27×10^3	5.67×10^2	3.00×10^2	8.54×10^1	3.13×10^2	9.29×10^1	3.00×10^2	8.70×10^1	1.04×10^2	8.45×10^1
w/t/l	25/0/3		25/0/3		20/0/8		21/0/7		22/1/5			

* The best result for each function is shown in boldface.

6. Conclusions

To balance exploration and exploitation, an improved version of ABC, called ABC-ESDL, is proposed in this paper. In ABC-ESDL, there are two modifications: elite strategy (ES) and dimension learning (DL). The elite strategy is used to guide the search. Good solutions are selected into the elite set. These elite solutions are used to modify the search model. To maintain the size of the elite set, a simple replacement method is employed. In dimension learning, the difference between different dimensions can achieve a large jump to help trapped solutions escape from local minima. The performance of our approach ABC-ESDL is verified on twelve classical benchmark functions (with dimensions 30 and 100) and the 2013 IEEE Congress on Evolutionary (CEC 2013) benchmark set.

Computational results of ABC-ESDL are compared with ABC, GABC, IABC, MABC, ABCVSS, and DFSABC-elite. For $D = 30$ and 100 , ABC-ESDL is not worse than ABCVSS, MABC, IABC, GABC, and ABC. DFSABC-elite is better than ABC-ESDL on only one problem for $D = 30$ and two problems for $D = 100$. For the rest of problems, ABC-ESDL outperforms DFSABC-elite. For the 2013 IEEE Congress on Evolutionary (CEC 2013) benchmark set, ABC-ESDL still achieves better performances than the compared algorithms.

Another experiment investigates the effectiveness of ES and DL. Results show that ES or DL can achieve improvements. ABC with two strategies (both ES and DL) surpasses ABC and ABC with a single strategy (ES or DL). It confirms the effectiveness of our proposed strategies.

For the onlooker bees, offspring is generated for each elite solution in the elite set. So, an onlooker bee generates M new solutions when a parent solution X_i is selected. This complexity will increase the computational time. To reduce the effects of such computational effort, a small parameter M is used. In the future work, other strategies will be considered to replace the current method. In addition, more test functions [61] will be considered to further verify the performance of our approach.

Author Contributions: Writing—original draft preparation, S.X. and H.W.; writing—review and editing, W.W.; visualization, S.X.; supervision, H.W., D.T., Y.W., X.Y., and R.W.

Funding: This work was supported by the National Natural Science Foundation of China (Nos. 61663028, 61703199), the Distinguished Young Talents Plan of Jiangxi Province (No. 20171BCB23075), the Natural Science Foundation of Jiangxi Province (No. 20171BAB202035), the Science and Technology Plan Project of Jiangxi Provincial Education Department (Nos. GJJ170994, GJJ180940), and the Open Research Fund of Jiangxi Province Key Laboratory of Water Information Cooperative Sensing and Intelligent Processing (No. 2016WICSIP015).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Kennedy, J. Particle Swarm Optimization. In Proceedings of the 1995 International Conference on Neural Networks, Perth, WA, Australia, 27 November–1 December 1995; Volume 4, pp. 1942–1948.
2. Wang, F.; Zhang, H.; Li, K.S.; Lin, Z.Y.; Yang, J.; Shen, X.L. A hybrid particle swarm optimization algorithm using adaptive learning strategy. *Inf. Sci.* **2018**, *436–437*, 162–177. [[CrossRef](#)]
3. Souza, T.A.; Vieira, V.J.D.; Souza, M.A.; Correia, S.E.N.; Costa, S.L.N.C.; Costa, W.C.A. Feature selection based on binary particle swarm optimisation and neural networks for pathological voice detection. *Int. J. Bio-Inspir. Comput.* **2018**, *11*, 91–101. [[CrossRef](#)]
4. Sun, C.L.; Jin, Y.C.; Chen, R.; Ding, J.L.; Zeng, J.C. Surrogate-assisted cooperative swarm optimization of high-dimensional expensive problems. *IEEE Trans. Evol. Comput.* **2017**, *21*, 644–660. [[CrossRef](#)]
5. Wang, H.; Wu, Z.J.; Rahnamayan, S.; Liu, Y.; Ventresca, M. Enhancing particle swarm optimization using generalized opposition-based learning. *Inf. Sci.* **2011**, *181*, 4699–4714.
6. Karaboga, D. *An Idea Based on Honey Bee Swarm for Numerical Optimization*; Technical Report-tr06; Engineering Faculty, Computer Engineering Department, Erciyes University: Kayseri, Turkey, 2005.
7. Amiri, E.; Dehkordi, M.N. Dynamic data clustering by combining improved discrete artificial bee colony algorithm with fuzzy logic. *Int. J. Bio-Inspir. Comput.* **2018**, *12*, 164–172. [[CrossRef](#)]

8. Meang, Z.; Pan, J.S. HARD-DE: Hierarchical archive based mutation strategy with depth information of evolution for the enhancement of differential evolution on numerical optimization. *IEEE Access* **2019**, *7*, 12832–12854. [[CrossRef](#)]
9. Meang, Z.; Pan, J.S.; Kong, L.P. Parameters with adaptive learning mechanism (PALM) for the enhancement of differential evolution. *Knowl.-Based Syst.* **2018**, *141*, 92–112. [[CrossRef](#)]
10. Yang, X.S. *Engineering Optimization: An Introduction with Metaheuristic Applications*; John Wiley & Sons: Etobicoke, ON, Canada, 2010.
11. Wang, H.; Wang, W.; Sun, H.; Rahnamayan, S. Firefly algorithm with random attraction. *Int. J. Bio-Inspir. Comput.* **2016**, *8*, 33–41. [[CrossRef](#)]
12. Wang, H.; Wang, W.J.; Cui, Z.H.; Zhou, X.Y.; Zhao, J.; Li, Y. A new dynamic firefly algorithm for demand estimation of water resources. *Inf. Sci.* **2018**, *438*, 95–106. [[CrossRef](#)]
13. Wang, H.; Wang, W.J.; Cui, L.Z.; Sun, H.; Zhao, J.; Wang, Y.; Xue, Y. A hybrid multi-objective firefly algorithm for big data optimization. *Appl. Soft Comput.* **2018**, *69*, 806–815. [[CrossRef](#)]
14. Wang, G.G.; Deb, S.; Coelho, L.S. Earthworm optimisation algorithm: A bio-inspired metaheuristic algorithm for global optimisation problems. *Int. J. Bio-Inspir. Comput.* **2018**, *12*, 1–22. [[CrossRef](#)]
15. Yang, X.S.; Deb, S. Cuckoo Search via Levy Flights. *Mathematics* **2010**, *1*, 210–214.
16. Zhang, M.; Wang, H.; Cui, Z.; Chen, J. Hybrid multi-objective cuckoo search with dynamical local search. *Memet. Comput.* **2018**, *10*, 199–208. [[CrossRef](#)]
17. Wang, G.G. Moth search algorithm: A bio-inspired metaheuristic algorithm for global optimization problems. *Memet. Comput.* **2016**, *10*, 1–14. [[CrossRef](#)]
18. Cui, Z.; Wang, Y.; Cai, X. A pigeon-inspired optimization algorithm for many-objective optimization problems. *Sci. China Inf. Sci.* **2019**, *62*, 070212. [[CrossRef](#)]
19. Yang, X.S. A new metaheuristic bat-inspired algorithm. *Comput. Knowl. Technol.* **2010**, *284*, 65–74.
20. Wang, Y.; Wang, P.; Zhang, J.; Cui, Z.; Cai, X.; Zhang, W.; Chen, J. A novel bat algorithm with multiple strategies coupling for numerical optimization. *Mathematics* **2019**, *7*, 135. [[CrossRef](#)]
21. Cai, X.J.; Gao, X.Z.; Xue, Y. Improved bat algorithm with optimal forage strategy and random disturbance strategy. *Int. J. Bio-Inspir. Comput.* **2016**, *8*, 205–214. [[CrossRef](#)]
22. Cui, Z.H.; Xue, F.; Cai, X.J.; Gao, Y.; Wang, G.G.; Chen, J.J. Detection of malicious code variants based on deep learning. *IEEE Trans. Ind. Inf.* **2018**, *14*, 3187–3196. [[CrossRef](#)]
23. Cai, X.; Wang, H.; Cui, Z.; Cai, J.; Xue, Y.; Wang, L. Bat algorithm with triangle-flipping strategy for numerical optimization. *Int. J. Mach. Learn. Cybern.* **2018**, *9*, 199–215. [[CrossRef](#)]
24. Wang, G.G.; Guo, L.H.; Gandomi, A.H.; Hao, G.S.; Wang, H.Q. Chaotic krill herd algorithm. *Inf. Sci.* **2014**, *274*, 17–34. [[CrossRef](#)]
25. Wang, G.G.; Gandomi, A.H.; Alavi, A.H. An effective krill herd algorithm with migration operator in biogeography-based optimization. *Appl. Math. Model.* **2014**, *38*, 2454–2462. [[CrossRef](#)]
26. Wang, G.G.; Gandomi, A.H.; Alavi, A.H. Stud krill herd algorithm. *Neurocomputing* **2014**, *128*, 363–370. [[CrossRef](#)]
27. Wang, G.G.; Guo, L.H.; Wang, H.Q.; Duan, H.; Luo, L.; Li, J. Incorporating mutation scheme into krill herd algorithm for global numerical optimization. *Neural Comput. Appl.* **2014**, *24*, 853–871. [[CrossRef](#)]
28. Grimaccia, F.; Gruosso, G.; Mussetta, M.; Niccolai, A.; Zich, R.E. Design of tubular permanent magnet generators for vehicle energy harvesting by means of social network optimization. *IEEE Trans. Ind. Electron.* **2018**, *65*, 1884–1892. [[CrossRef](#)]
29. Karaboga, D.; Akay, B. A survey: Algorithms simulating bee swarm intelligence. *Artif. Intell. Rev.* **2009**, *31*, 61–85. [[CrossRef](#)]
30. Kumar, A.; Kumar, D.; Jarial, S.K. A review on artificial bee colony algorithms and their applications to data clustering. *Cybern. Inf. Technol.* **2017**, *17*, 3–28. [[CrossRef](#)]
31. Zhu, G.; Kwong, S. Gbest-guided artificial bee colony algorithm for numerical function optimization. *Appl. Math. Comput.* **2010**, *217*, 3166–3173. [[CrossRef](#)]
32. Wang, H.; Wu, Z.J.; Rahnamayan, S.; Sun, H.; Liu, Y.; Pan, J.S. Multi-strategy ensemble artificial bee colony algorithm. *Inf. Sci.* **2014**, *279*, 587–603. [[CrossRef](#)]
33. Karaboga, D.; Akay, B. A comparative study of artificial bee colony algorithm. *Appl. Math. Comput.* **2009**, *214*, 108–132. [[CrossRef](#)]

34. Karaboga, D.; Gorkemli, B. A quick artificial bee colony (qABC) algorithm and its performance on optimization problems. *Appl. Soft Comput.* **2014**, *23*, 227–238. [[CrossRef](#)]
35. Gao, W.; Liu, S. A modified artificial bee colony algorithm. *Comput. Oper. Res.* **2012**, *39*, 687–697. [[CrossRef](#)]
36. Cui, L.Z.; Li, G.H.; Lin, Q.Z.; Du, Z.H.; Gao, W.F.; Chen, J.Y.; Lu, N. A novel artificial bee colony algorithm with depth-first search framework and elite-guided search equation. *Inf. Sci.* **2016**, *367–368*, 1012–1044. [[CrossRef](#)]
37. Li, G.H.; Cui, L.Z.; Fu, X.H.; Wen, Z.K.; Lu, N.; Lu, J. Artificial bee colony algorithm with gene recombination for numerical function optimization. *Appl. Soft Comput.* **2017**, *52*, 146–159. [[CrossRef](#)]
38. Yao, X.; Chan, F.T.S.; Lin, Y.; Jin, H.; Gao, L.; Wang, X.; Zhou, J. An individual dependent multi-colony artificial bee colony algorithm. *Inf. Sci.* **2019**. [[CrossRef](#)]
39. Kumar, D.; Mishra, K.K. Co-variance guided artificial bee colony. *Appl. Soft Comput.* **2018**, *70*, 86–107. [[CrossRef](#)]
40. Yang, J.; Jiang, Q.; Wang, L.; Liu, S.; Zhang, Y.; Li, W.; Wang, B. An adaptive encoding learning for artificial bee colony algorithms. *J. Comput. Sci.* **2019**, *30*, 11–27. [[CrossRef](#)]
41. Chen, X.; Tianfield, H.; Li, K. Self-adaptive differential artificial bee colony algorithm for global optimization problems. *Swarm Evol. Comput.* **2019**, *45*, 70–91. [[CrossRef](#)]
42. Zhang, X.; Zhang, X. A binary artificial bee colony algorithm for constructing spanning trees in vehicular ad hoc networks. *Ad Hoc Netw.* **2017**, *58*, 198–204. [[CrossRef](#)]
43. Zorarpacı, E.; Özel, S.A. A hybrid approach of differential evolution and artificial bee colony for feature selection. *Expert Syst. Appl.* **2016**, *62*, 91–103. [[CrossRef](#)]
44. Yuan, X.; Wang, P.; Yuan, Y.; Huang, Y.; Zhang, X. A new quantum inspired chaotic artificial bee colony algorithm for optimal power flow problem. *Energy Convers. Manag.* **2015**, *100*, 1–9. [[CrossRef](#)]
45. Dokeroglu, T.; Sevinc, E.; Cosar, A. Artificial bee colony optimization for the quadratic assignment problem. *Appl. Soft Comput.* **2019**, *76*, 595–606. [[CrossRef](#)]
46. Kishor, A.; Singh, P.K.; Prakash, J. NSABC: Non-dominated sorting based multi-objective artificial bee colony algorithm and its application in data clustering. *Neurocomputing* **2016**, *216*, 514–533. [[CrossRef](#)]
47. Wang, P.; Xue, F.; Li, H.; Cui, Z.; Xie, L.; Chen, J. A multi-objective DV-Hop localization algorithm based on NSGA-II in internet of things. *Mathematics* **2019**, *7*, 184. [[CrossRef](#)]
48. Pan, J.S.; Kong, L.P.; Sung, T.W.; Tsai, P.W.; Snasel, V. α -Fraction first strategy for hierarchical wireless sensor networks. *J. Internet Technol.* **2018**, *19*, 1717–1726.
49. Xue, X.S.; Pan, J.S. A compact co-evolutionary algorithm for sensor ontology meta-matching. *Knowl. Inf. Syst.* **2018**, *56*, 335–353. [[CrossRef](#)]
50. Hashim, H.A.; Ayinde, B.O.; Abido, M.A. Optimal placement of relay nodes in wireless sensor network using artificial bee colony algorithm. *J. Netw. Comput. Appl.* **2016**, *64*, 239–248. [[CrossRef](#)]
51. Wang, H.; Wu, Z.J.; Zhou, X.Y.; Rahnamayan, S. Accelerating artificial bee colony algorithm by using an external archive. In Proceedings of the IEEE Congress on Evolutionary Computation, Cancun, Mexico, 20–23 June 2013; pp. 517–521.
52. Li, B.; Sun, H.; Zhao, J.; Wang, H.; Wu, R.X. Artificial bee colony algorithm with different dimensional learning. *Appl. Res. Comput.* **2016**, *33*, 1028–1033.
53. Wang, H.; Wang, W.; Zhou, X.; Sun, H.; Zhao, J.; Yu, X.; Cui, Z. Firefly algorithm with neighborhood attraction. *Inf. Sci.* **2017**, *382*, 374–387. [[CrossRef](#)]
54. Wang, H.; Sun, H.; Li, C.H.; Rahnamayan, S.; Pan, J.S. Diversity enhanced particle swarm optimization with neighborhood search. *Inf. Sci.* **2013**, *223*, 119–135. [[CrossRef](#)]
55. Wang, G.G.; Tan, Y. Improving metaheuristic algorithms with information feedback models. *IEEE Trans. Cybern.* **2019**, *49*, 542–555. [[CrossRef](#)]
56. Wang, H.; Rahnamayan, S.; Sun, H.; Omran, M.G.H. Gaussian bare-bones differential evolution. *IEEE Trans. Cybern.* **2013**, *43*, 634–647. [[CrossRef](#)]
57. Wang, H.; Cui, Z.H.; Sun, H.; Rahnamayan, S.; Yang, X.S. Randomly attracted firefly algorithm with neighborhood search and dynamic parameter adjustment mechanism. *Soft Comput.* **2017**, *21*, 5325–5339. [[CrossRef](#)]
58. Sun, C.L.; Zeng, J.C.; Pan, J.S.; Xue, S.D.; Jin, Y.C. A new fitness estimation strategy for particle swarm optimization. *Inf. Sci.* **2013**, *221*, 355–370. [[CrossRef](#)]

59. Kiran, M.S.; Hakli, H.; Guanduz, M.; Uguz, H. Artificial bee colony algorithm with variable search strategy for continuous optimization. *Inf. Sci.* **2015**, *300*, 140–157. [[CrossRef](#)]
60. Liang, J.J.; Qu, B.Y.; Suganthan, P.N.; Hernández-Díaz, A.G. *Problem Definitions and Evaluation Criteria for the CEC 2013 Special Session and Competition on Real-Parameter Optimization*; Tech. Rep. 201212; Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Nanyang Technological University: Singapore, 2013.
61. Serani, A.; Leotardi, C.; Iemma, U.; Campana, E.F.; Fasano, G.; Diez, M. Parameter selection in synchronous and asynchronous deterministic particle swarm optimization for ship hydrodynamics problems. *Appl. Soft Comput.* **2016**, *49*, 313–334. [[CrossRef](#)]



© 2019 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).