Jestr

Journal of Engineering Science and Technology Review 7 (3) (2014) 96 – 107

JOURNAL OF
Engineering Science and Technology Review

www.jestr.org

Research Article

An Effective Hybrid Artificial Bee Colony Algorithm for Nonnegative Linear Least Squares Problems

Xiangyu Kong[1, 2, *], Sanyang Liu[1] and Zhen Wang[2]

[1] *Dept. of Mathematics and Statistics, Xidian University, Xi'an 710071, China*
[2] *Dept. of Mathematics and Information Science, Beifang University of Nationalities, Yinchuan 750021, China*

Received 2 March 2014; Accepted 19 July 2014

Abstract

An effective hybrid artificial bee colony algorithm is proposed in this paper for nonnegative linear least squares problems. To further improve the performance of algorithm, orthogonal initialization method is employed to generate the initial swarm. Furthermore, to balance the exploration and exploitation abilities, a new search mechanism is designed. The performance of this algorithm is verified by using 27 benchmark functions and 5 nonnegative linear least squares test problems. And the comparison analyses are given between the proposed algorithm and other swarm intelligence algorithms. Numerical results demonstrate that the proposed algorithm displays a high performance compared with other algorithms for global optimization problems and nonnegative linear least squares problems.

Keywords: Artificial Bee Colony, Orthogonal Initialization Method, Nonnegative Linear Least Squares, Optimization Problem

1. Introduction

The method of least squares is an important approach to simulate the approximate solution of over determined systems, i.e. sets of equations in which more equations than unknowns. Least squares problems are divided into two categories: linear least squares and non-linear least squares, depending on whether or not the residuals are linear in all unknowns. In mathematics and statistics, linear least squares is an approach to fitting a statistics or a mathematical model to data, in cases where the idealized value provided by the model for any data point is expressed linearly in terms of the unknown parameters of the model. The adapted model results can be used to predict unobserved values from the same system, to summarize the data, and to understand the mechanisms that may underlie the system. Without loss of generality, the Nonnegative Linear Least Squares (NLLS) problem can be formulated as follows:

$$\min_{x\geq 0} f(x) = \frac{1}{2}\|Ax-b\|^2 = \frac{1}{2}(Ax-b)^T(Ax-b) \quad (1)$$

where $A\in R^{m\times n}$, $m \geq n$, $rank(A) = n$, $b\in R^m$.

Over the past decade, solving the linear least squares via using classical mathematical programming methods has attracted much attention. These methods require matrix updates or factorizations, and can become overmuch expensive for very large-scale problems. Classical optimization methods are highly sensitive to the initial point, having very slow convergence and frequently converging to local optimum solution.

Motivated by foraging behavior of honey bees, Karaboga in 2005 [1] proposed the artificial bee colony (ABC) algorithm to optimize unconstrained problems. In a natural bee swarm, there are three groups of bees including employed bees, onlooker bees and scout bees. Half of the colony consists of employed bees, and the other half includes onlooker bees. A bee that is currently exploiting a food source is named as an employed bee. Employed bees perform waggle dance upon returning to the hive to propagate the information of its food source to the rest of the colony. A bee around the dance floor to choose any of the employed bees to follow is called an onlooker bee. A bee carrying out a random search for a new food source is named as a scout bee. In ABC algorithm, each food source is a possible solution for the problem under consideration and the nectar amount of a food source represents the quality of the solution represented by the fitness value. The ABC algorithm starts with a population of randomly generated food sources. Then the following three steps are repeated until a termination criterion is satisfied [2]:

1). Send the employed bees onto the food sources and measure their nectar amounts.
2). Select the food sources by the onlooker bees after share the information of employed bees and determine one nectar amount of the food sources.
3). Determine the scout bees and randomly generate a new food source.

The initial population containing *SN* solutions is generated randomly, *SN* is equal to the number of employed bees. Each solution $x_i\ (i = 1,2,...,SN)$ is an n-dimensional real-valued vector.

Let $x_i = \{x_{i1}, x_{i2}, ..., x_{in}\}$represent the ith food source in the population, and then each food source is formulated as follows:

* E-mail address: kxywz08@163.com

$$x_i^j = x_{\min} + rand(0,1)\left(x_{\max}^j - x_{\min}^j\right), \quad i = 1,2,...,SN, j = 1,2,...,n \tag{2}$$

These food sources are randomly assigned to SN number of employed bees and their finesses are evaluated.

The search equation for employed bees and onlooker bees can be described as follows:

$$x_{new(j)} = x_{i,j} + \phi_{i,j}\left(x_{i,j} - x_{k,j}\right) \tag{3}$$

where $k \in \{1,2,...,SN\}$, $k \neq j$ and $j \in \{1,2,...,n\}$ are randomly chosen indexes. $\phi_{i,j}$ is a uniform random number in the range $[-1,1]$. Once x_{new} is obtained, a greedy selection then performs between the old and candidate solutions.

In the onlooker bee stage, an onlooker bee selects a food source x_i depending on the probability value p_i calculated as follows:

$$p_i = \frac{f_i}{\sum_{i=1}^{SN} f_i} \tag{4}$$

where f_i is the nectar amount (i.e., the fitness value) of the ith food source x_i. Obviously, the higher the f_i is, the more probability that the ith food source is selected.

In the basic ABC algorithm, if a food source x_i cannot be improved further through a predetermined number of "limit", the food source should be abandoned, and the corresponding employed bee becomes a scout. The scout produces a food source randomly, which can be defined as:

$$x_i^j = x_{\min}^j + \left(x_{\max}^j - x_{\min}^j\right) \times rand(0,1), \quad j = 1,2,...,n. \tag{5}$$

Compared with DE, PSO, and other intelligent algorithms, the ABC algorithm shows better performance than other algorithms [2]. Then, the ABC algorithm was extended for constrained optimization problems in [3] and was applied to medical pattern classification and clustering problems [4, 5], to train neural networks [6, 7], to solve clustering problem [8], TSP problems [9],a large-scale capacitated facility location problem[10]. In this paper, an effective hybrid artificial bee colony (EH-ABC) algorithm is proposed for solving Nonnegative Linear Least Squares problems. The orthogonal initialization method is employed to generate initial population. Inspired by PSO and DE, a new search mechanism is proposed. The experimental results tested on 27 benchmark functions and a set of Nonnegative Linear Least Squares problems show that the EH-ABC algorithm can outperform ABC algorithm in most of the experiments.

Paper is organized as follows. In Section 2, the ABC algorithm for solving Nonnegative Linear Least Squares problems are introduced. In Section 3, simulation results of ABC and EH-ABC are presented and compared. Finally, a conclusion is provided.

2. The proposed ABC algorithm for Nonnegative Linear Least Squares problems

2.1 Orthogonal initialization

Population initialization is an important step in swarm intelligence-based algorithms, which can affect the quality of solution. It is desirable that the initial population be scattered uniformly over the feasible solution space, so that the algorithm can search the whole solution space evenly. Before an optimization problem is solved, there is no information about the location of the solution. Notice that an orthogonal array specifies a small number of combinations that are scattered uniformly over the space of all possible combinations. The orthogonal design can make the initial population be scattered evenly over the solution space. Therefore, in this paper we generate initial population by using the orthogonal initialization method [11][12].

The algorithm for generating an initial population is given as follows.

Algorithm 1: Generation of Initial Population.

Step 1: Divide the feasible solution space $[l,u]$ into S subspaces $[l_1,u_1],[l_2,u_2],\text{L},[l_S,u_S]$ based on the following equations:

$$\begin{cases} l_i = l + (i-1)\left(\dfrac{u(s)-l(s)}{S}\right)l_S, \\ u_i = u - (S-i)\left(\dfrac{u(s)-l(s)}{S}\right)l_S, \end{cases} \quad i = 1,2,\text{L},S. \tag{6}$$

Here, $u(s)-l(s) = \max_{1\le i\le D}\{u_i - l_i\}$.

Step2: Quantize subspace $[l_i,u_i]$ into Q_1 levels based on

$$\alpha_{ij} = \begin{cases} l_i, & j = 1, \\ l_i + (j-1)\left(\dfrac{u_i - l_i}{Q_1 - 1}\right), & 2 \le j \le Q_1 - 1, \\ u_i, & j = Q_1, \end{cases} \tag{7}$$

where Q_1 is odd. Then, construct orthogonal array $L_{M_1}(Q_1^N) = \left[a_{ij}\right]_{M_1 \times N}$ to select M_1 individuals based on

$$\begin{cases} (\alpha_{1,a_{11}},\alpha_{2,a_{12}},\text{L},\alpha_{N,a_{1N}}) \\ (\alpha_{1,a_{21}},\alpha_{2,a_{22}},\text{L},\alpha_{N,a_{2N}}) \\ \text{L} \\ (\alpha_{1,a_{M11}},\alpha_{2,a_{M12}},\text{L},\alpha_{N,a_{M1N}}). \end{cases} \tag{8}$$

Here, $L_{M_1}(Q_1^N)$ can be generated as follows. Select the smallest J_1 fulfilling $(Q_1^{J_1}-1)/(Q_1-1) \ge N$. If $(Q_1^{J_1}-1)/(Q_1-1) = N$, then $N' = N$ else $N' = (Q_1^{J_1}-1)/(Q_1-1)$. Then, construct the basic columns based on $j = \dfrac{Q_1^{k-1}-1}{Q_1-1}+1$, $a_{ij} = \left\lfloor \dfrac{i-1}{Q_1^{J_1-k}} \right\rfloor \bmod Q_1$, for $i = 1,\text{L},M_1$, $k = 1,\text{L}\ J_1$. Construct the non-basic columns as

$j=\frac{Q_1^{k-1}-1}{Q_1-1}+1$, $a_{j+(s-1)(Q_1-1)+t}=(a_s\times t+a_j)\bmod Q_1$, for $s=1,\mathrm{L}\ ,j-1$, $t=1,\mathrm{L}\ ,Q_1$. Thus, the orthogonal array $L_{M_1}(Q_1^{N'})$ is constructed. Delete the last $N'-N$ columns of $L_{M_1}(Q_1^{N'})$ to get $L_{M_1}(Q_1^{N})$ where $M_1=Q_1^{J_1}$.

Step 3: Among the M_1S individuals, select SN individuals having the smallest cost as the initial population.

2.2 New Search Equations

As we all known that how to balance exploration and exploitation abilities to achieve good optimization performance is an important problem for the population based algorithms, such as GA, PSO, DE, and so on. The exploration refers to the ability to search the unknown regions in the solution space to find the global optimum, while the exploitation refers to the ability to discover better solutions based on the information of the previous good solutions. Actually, the exploration and exploitation abilities contradict with each other, so that the two abilities should be well balanced. In ABC algorithm, the search equation proposed is good at exploration but poor at exploitation. In order to improve the exploitation, a new search mechanism is proposed in this section.

Differential evolution (DE) is a population based algorithm, whose main strategy is to generate a new position for an individual by calculating vector differences between other randomly selected individuals in the population. It has been shown the efficiency for many optimization problems in real-world applications. It follows the general stages of an evolutionary algorithm. In DE algorithm, three evolutionary operations including mutation, crossover and selection will be executed. There are several kinds of mutation operation, which formulates different DE algorithms. Among them, "DE/best/1" can improve the exploitation abilities of algorithm, which can be described as follows:

$$\text{DE/best/1: } v_i=x_{best}+F(x_{r1}-x_{r2}), \tag{9}$$

where $i\in\{1,2,\mathrm{L}\ ,SN\}$; $r1$ and $r2$ are different random integer indices selected from $\{1,2,\mathrm{L}\ ,SN\}$; x_{best} is the global best solution; $F\in[0.5,1]$ is a positive real number.

Motivated by DE and based on the property of ABC algorithm, a new search equation is proposed as follows:

$$v_{i,j}=x_{best,j}+\varphi_{i,j}(x_{i,j}-x_{r1,j}), \tag{10}$$

where $i\in\{1,2,\mathrm{L}\ ,SN\}$; $r1$ is integers randomly selected from $\{1,2,\mathrm{L}\ ,SN\}$, and is different from i; $x_{best,j}$ is the global best solution; $j\in\{1,2,\mathrm{L}\ ,D\}$ is a randomly selected index; and $\varphi_{ij}\in[0,1]$ is uniformly distributed random number. Similar to DE, search equation (10) can improve the convergence performance which can enhance the exploitation ability of the algorithm.

Inspired of PSO, in order to improve the exploitation ability of ABC algorithm, takeing the advantages of the search equation in PSO, Zhu et al. [13] proposed the g-best ABC algorithm. In this paper, the modified search equation is described as follows:

$$v_{ij}=x_{ij}+\phi_{ij}(x_{ij}-x_{kj})+\varphi_{ij}(y_j-x_{ij}), \tag{11}$$

where $k\in\{1,2,...,SN\}$ is a random selected index which is different from i , $j\in\{1,2,...,D\}$ is a random selected index, y_j is the jth element of the global best solution, $\phi_{ij}\in[-1,1]$ and $\varphi_{ij}\in[0,1.5]$ are both uniformly distributed random numbers. This search equation can also improve the exploitation ability of the algorithm.

2.3 The proposed algorithm

From the above analysis, in order to take advantage of search equation (10) and (11), a new algorithm is proposed by hybridizing the two search equations. In the new artificial bee colony algorithm, a selective probability *p* is introduced to control the frequency of using search equation (10) and (11). Thus, the main steps of the new artificial bee colony algorithm with orthogonal initialization are given as follows.

Algorithm 2: Effective hybrid artificial bee colony algorithm

1). Initialization: Preset selective probability p and population size SN .
2). Perform Algorithm 1 to create an initial population, calculate the function values of the population.
3). While (FE< Max. FE) do
4). for $i=1$ *to SN do*
5). Produce new solutions v_i *by using equation (10)*
6). If $f(v_i)<f(x_i)$ *then* $x_i=v_i$
7). else then
8). if $rand(0,1)<P$ *then*
9). Produce new solutions v_i *by using equation (11)*
10). if $f(v_i)<f(x_i)$ *then* $x_i=v_i$
11). end if
12). end if
13). end if
14). end for
15). end while (FE= Max. FE)

3. Computational results and comparisons

In order to illustrate the efficiency of EH-ABC algorithm, we test 27 benchmark functions and 5 nonnegative linear least squares test problems in this section.

3.1 Experiments on 27 benchmark functions

3.1.1 Test functions

In this subsection, the EH-ABC algorithm proposed in this paper is applied to minimize 27 benchmark functions, as shown in Table 1 and 2. In Table 1, the dimensions of the benchmark functions are given in the third column. In function Hartman3,

$$a=\begin{bmatrix}3.0 & 10 & 30\\ 0.1 & 10 & 35\\ 3.0 & 10 & 30\\ 0.1 & 10 & 35\end{bmatrix}^T,\ p=\begin{bmatrix}0.3689 & 0.1170 & 0.2673\\ 0.4699 & 0.4387 & 0.7470\\ 0.1091 & 0.8732 & 0.5547\\ 0.03815 & 0.5743 & 0.8828\end{bmatrix}^T.$$

In function

$$a=\begin{bmatrix} 4.0 & 1.0 & 8.0 & 6.0 & 3.0 & 2.0 & 5.0 & 8.0 & 6.0 & 7.0 \\ 4.0 & 1.0 & 8.0 & 6.0 & 7.0 & 9.0 & 5.0 & 1.0 & 2.0 & 3.6 \\ 4.0 & 1.0 & 8.0 & 6.0 & 3.0 & 2.0 & 3.0 & 8.0 & 6.0 & 7.0 \\ 4.0 & 1.0 & 8.0 & 6.0 & 7.0 & 9.0 & 3.0 & 1.0 & 2.0 & 3.6 \end{bmatrix}$$,

$$c=\frac{1}{10}[1,2,2,4,4,6,3,7,5,5]^T.$$

The benchmark functions presented in Table 2 are tested of dimension D=30 and D=60.

Tab.1. Benchmark functions f_1 - f_{16} used in experiments. D: Dimension, C: Characteristic, U: Unimodal, M: Multimodal, S: Separable, N: Non-Separable.

Function	Range	D	C	Formulation
Beale	[-4.5,4.5]	2	UN	$f_1=(1.5-x_1+x_1x_2)^2+(2.25-x_1+x_1x_2^2)^2+(2.625-x_1+x_1x_2^3)^2$
Bohachevsky	[-100,100]	2	MS	$f_2=x_1^2+2x_2^2-0.3\cos(3\pi x_1)-0.4\cos(4\pi x_2)+0.7$
Booth	[-10,10]	2	MS	$f_3=(x_1+2x_2-7)^2+(2x_1+x_2-5)^2$
Branin	[-5,10]×[0,15]	2	MS	$f_4=\left(x_2-\frac{5.1}{4\pi^2}x_1^2+\frac{5}{\pi}x_1-6\right)^2+10\left(1-\frac{1}{8\pi}\right)\cos x_1+10$
Colville	[-10,10]	4	UN	$f_5=100(x_1^2-x_2)^2+(x_1-1)^2+(x_3-1)^2+90(x_3^2-x_4)^2+10.1((x_2-1)^2+(x_4-1)^2)$
Easom	[-100,100]	2	UN	$f_6=-\cos x_1\cos x_2\exp(-(x_1-\pi)^2-(x_2-\pi)^2)$
GoldStein-Price	[-2,2]	2	MN	$f_7=\begin{bmatrix}1+(x_1+x_2+1)^2\\(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2)\end{bmatrix}\cdot\begin{bmatrix}30+(2x_1-3x_2)^2\\(18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2)\end{bmatrix}$
Hartman3	[0,1]	3	MN	$f_8=-\sum_{i=1}^4 c_i\exp\left[-\sum_{j=1}^3 a_{ij}(x_j-p_{ij})^2\right];\ c=[1.0,1.2,3.0,3.2]$
Six Hump Camel Back	[-5,5]	2	MN	$f_9=4x_1^2-2.1x_1^4+\frac{1}{3}x_1^6+x_1x_2-4x_2^2+4x_2^4$
Matyas	[-10,10]	2	UN	$f_{10}=0.26(x_1^2+x_2^2)-0.48x_1x_2$
Perm	[-*D*,*D*]	2	MN	$f_{11}=\sum_{k=1}^n\left[\sum_{i=1}^2(i^k+0.5)((x_i/i)^k-1)\right]^2$
Powell	[-4,5]	4	UN	$f_{12}=\sum_{i=1}^{n/k}(x_{4i-3}+10x_{4i-2})^2+5(x_{4i-1}+10x_{4i})^2+(x_{4i-2}+10x_{4i-1})^4+10(x_{4i-3}+10x_{4i})^4$
PowerSum	[0, *D*]	24	MN	$f_{13}=\sum_{k=1}^n\left[\sum_{i=1}^4(x_i^k)-b_k\right]^2;\ b=[8,18,44,114]$
Shekel	[0,10]	4	MN	$f_{14}=-\sum_{j=1}^m\left[\sum_{i=1}^4(x_i-a_{ij})^2+c_i\right]^{-1}$
Shubert	[-10,10]	2	MN	$f_{15}=\left(\sum_{i=1}^5 i\cos((i+1)x_1+i)\right)\cdot\left(\sum_{i=1}^5 i\cos((i+1)x_2+i)\right)$
Trid6	[-36,36]	6	UN	$f_{16}=\sum_{i=1}^n(x_i-1)^2-\sum_{i=2}^n x_ix_{i-1}$

Tab.2. Benchmark functions f_{17}-f_{27} used in experiments. C: Characteristic, U: Unimodal, M: Multimodal, S: Separable, N: Non-Separable, n=D.

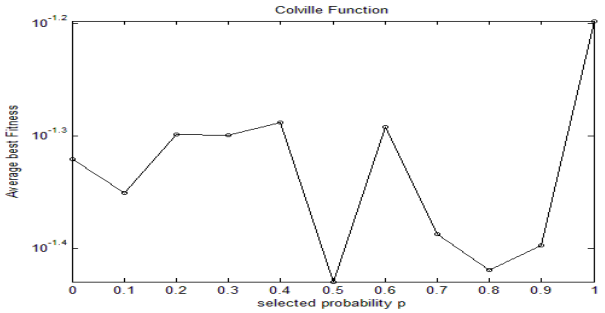
Function	Range	C	Formulation
Ackley	[-32,32]	MN	$f_{17}=-20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^n x_i^2}\right)-\exp\left(\frac{1}{n}\sum_{i=1}^n\cos(2\pi x_i)\right)+20+e$
Dixon-Price	[-10,10]	UN	$f_{18}=(x_1-1)^2+\sum_{i=2}^n i(2x_i^2-x_{i-1})^2$
Griewank	[-600,600]	MN	$f_{19}=\frac{1}{4000}\sum_{i=1}^n x_i^2-\prod_{i=1}^n\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$
Levy	[-10,10]	MN	$f_{20}=\sin^2(\pi y_1)+\sum_{i=1}^{n-1}\left[(y_i-1)^2(1+10\sin^2(\pi y_i+1))\right]+(y_n-1)^2(1+10\sin^2(2\pi y_n));\ y_i=1+\frac{x_i-1}{4},\quad i=1,\mathrm{L}\ ,n.$

Michalewicz	[0, π]	MS	$f_{21} = -\sum_{i=1}^{n} \sin(x_i)\left(\sin\left(ix_i^2/\pi\right)\right)^{2m}$, $m = 10$
Rastrigin	[-5.12,5.12]	MS	$f_{22} = \sum_{i=1}^{n}\left[x_i^2 - 10\cos(2\pi x_i) + 10\right]$
Rosenbrock	[-30,30]	UN	$f_{23} = \sum_{i=1}^{n-1}[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$
Schwefel	[-500,500]	MS	$f_{24} = \sum_{i=1}^{n} -x_i \sin\left(\sqrt{\lvert x_i \rvert}\right)$
Sphere	[-100,100]	US	$f_{25} = \sum_{i=1}^{n} x_i^2$
SumSquares	[-10,10]	US	$f_{26} = \sum_{i=1}^{n} ix_i^2$
Zakharov	[-5,10]	UN	$f_{27} = \sum_{i=1}^{n} x_i^2 + \left(\sum_{i=1}^{n} 0.5ix_i\right)^2 + \left(\sum_{i=1}^{n} 0.5ix_i\right)^4$

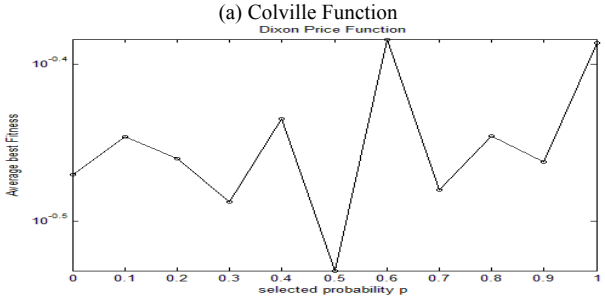
3.1.2 Effects of selective probability *p*

In this subsection, we investigate the impact of selective probability p on the new algorithm. Note that the test function Colville, Dixon-Price, Six Hump Camel Back and Schewefel are representative, so selective probability p is tested according to these four functions. The ABCO algorithm runs 30 times on each function, and the mean values of the final results are plotted in Figure 1. As all the test functions are minimization problems, the smaller the mean values, the better it is.

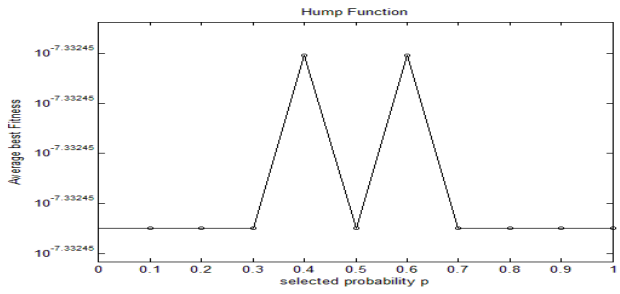
From Figure 1, we can see that the selective probability p can affect the results. For these four test functions, better results are obtained when p is around 0.5. Hence, the selective probability p will be equal to 0.5 for all test functions in the experiments.



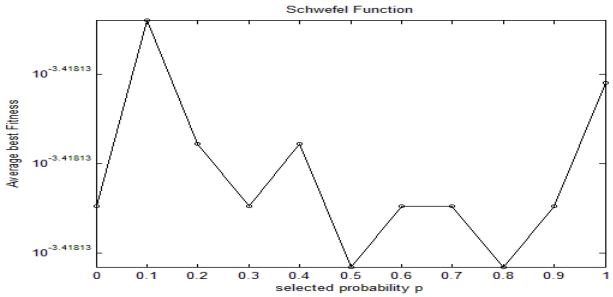
(a) Colville Function



(b) Dixion Price Function



(c)Hump Function



(d)Schwefel Function

Fig.1. Results on four test functions with different selective probability *p*.

3.1.3 Comparison of EH-ABC with ABC

In order to verify the performance of EH-ABC algorithm proposed in this paper, this subsection presents a comparison of EH-ABC algorithm with original ABC algorithm. In the experiments, both EH-ABC and ABC use the same parameter settings. The population size SN, limit, and maximum number of cycle (MSN) are set to 60, $(SN/2)*D$, 2000 for $D = 30$ and 4000 for $D = 60$, respectively. All experiments are repeated 30 times.

Table 3 and 4 shows the optimization results in terms of best, worst, mean and std, which represents the best, the worst, the mean, standard deviation of function value, respectively. The best results are marked in bold. As shown in Table 3 and 4, the mean function values of the EH-ABC algorithm are equal or closer to the optimal ones than which of the ABC algorithm, and the standard deviations are relatively small. Particularly, EH-ABC algorithm outperforms ABC algorithm on all test functions except function f_1, f_{18} with dimension $D = 30$ and $D = 60$, and f_{23} with dimension $D = 60$. At the mean time, the two algorithms have the same mean function values on function f_2 and f_4, which equal to the optimal ones. All these results show that EH-ABC algorithm has the better performance than ABC algorithm on unimodal and multimodal problems.

To further test the performance of EH-ABC algorithm, we compare EH-ABC algorithm with other population based algorithms, including some proposed ABC algorithms. The experiments focus on the comparison of EH-ABC algorithm with DE [2], PSO [2], CLPSO [12], CES [14], FES [15], ESLAT [2], CMA-ES [16], GABC [13], I-ABC [16], PS-ABC [16] and NABC [17]. For DE and PSO, the parameters are chosen as in [2]. For the rest algorithms, the parameter settings are followed the original papers of these algorithm.

Table 5 presents the comparison results of DE, PSO, CES, FES, ESLAT, CMA-ES and EH-ABC. Table 6 represents the comparison results of EH-ABC with four other ABC algorithms. In Table 5, results of DE, PSO, CES, FES, ESLAT and CMA-ES can be found in [2]. Results of GABC, I-ABC, PS-ABC and NABC in Table 6, can be found in [17]. In Table 5 and 6, the best results among these algorithms are shown in bold. The last row of Table 5 and the last column of Table 6 show the statistical significance level of the difference of the mean value of EH-ABC and the best algorithm among the other algorithms in the table. Here, "+" indicates the t value of 59 degrees of freedom which is significant at a 0.06 level of significance by two-tailed test; "·" represents the difference of mean values which is not statistically significant; and "NA" means two algorithms achieve the same accuracy results [18].

From Table 5, it can be seen that EH-ABC algorithm outperforms the other seven algorithms on all six test functions. The comparison results show that EH-ABC performs better than the other seven algorithms on whether unimodal functions or multimodal functions.

From the results in Table 6, EH-ABC outperforms GABC on all six test functions. Even more, EH-ABC outperforms all four ABC algorithms on function Ackley, Rosenbrock and Schwefel. For function Griewank and Rastrigin, EH-ABC has the same results with I-ABC and PS-ABC which achieve the optimal results, while I-ABC and PS-ABC perform better than EH-ABC on Sphere. Compared with NABC, EH-ABC outperforms on all six functions except function Rastrigin which obtained the same result.

In order to show the performance of EH-ABC algorithm more clearly, Figure 2-4 shows the mean best function value of ten test functions. It is clear that the EH-ABC algorithm has higher convergence rate than the ABC algorithm has.

Tab.3. Best, worst, mean and standard deviation obtained by ABC and EH-ABC for functions f_1-f_{16}.

	ABC				EH-ABC			
	Best	Mean	Worst	Std	Best	Mean	Worst	Std
f_1	7.62e-11	**2.49e-09**	2.75e-08	**5.03e-09**	1.58e-10	1.91e-04	4.61e-03	8.38e-04
f_2	0.00e+00	**0.00e+00**	0.00e+00	**0.00e+00**	0.00e+00	**0.00e+00**	0.00e+00	**0.00e+00**
f_3	1.43e-19	7.74e-18	2.34e-17	6.69e-18	0.00e+00	**0.00e+00**	0.00e+00	**0.00e+00**
f_4	3.98e-01	**3.98e-01**	3.98e-01	**0.00e+00**	3.98e-01	**3.98e-01**	3.98e-01	**0.00e+00**
f_5	4.29e-02	2.26e-01	4.97e-01	1.389e-01	8.72e-03	**4.53e-02**	1.26e-01	**3.02e-02**
f_6	-1	-0.99997	-0.99981	5.48e-05	-1	**-1**	-1	**0.00e+00**
f_7	3	3.000014	3.000257	4.97e-05	3	**3**	3	**1.75e-15**
f_8	-3.86278	**-3.86278**	-3.86278	**2.32e-15**	-3.86278	**-3.86278**	-3.86278	2.71e-15
f_9	4.65e-08	**4.65e-08**	4.65e-08	1.03e-16	4.65e-08	**4.65e-08**	4.65e-08	**0.00e+00**
f_{10}	6.99e-17	8.26e-14	2.09e-12	3.82e-13	1.74e-16	**2.99e-14**	2.16e-13	**4.63e-14**
f_{11}	8.30e+77	1.13e+83	1.13e+84	2.57e+83	2.27e+77	**1.31e+81**	8.12e+81	**2.16e+81**
f_{12}	6.17e-07	2.68e-05	5.31e-05	1.59e-05	2.43e-08	**1.08e-05**	4.65e-05	**1.15e-05**
f_{13}	7.21e-04	1.67e-02	4.74e-02	1.24e-02	1.93e-05	**1.17e-02**	3.94e-02	**1.15e-02**
f_{14}	-10.5364	**-10.5364**	-10.5364	3.43e-15	-10.5364	**-10.5364**	-10.5364	**2.14e-15**
f_{15}	-186.731	**-186.731**	-186.731	**5.28e-15**	-186.731	**-186.731**	-186.731	2.79e-14
f_{16}	-50	**-50**	-50	**9.27e-11**	-50	**-50**	-50	1.50e-06

Tab.4. Best, worst, mean and standard deviation obtained by ABC and EH-ABC for functions f_{17}-f_{27}.

	D	ABC				EH-ABC			
		Best	Mean	Worst	Std	Best	Mean	Worst	Std
f_{17}	30	0.00e+00	1.66e-15	1.78e-14	3.46e-15	0.00e+00	**0.00e+00**	0.00e+00	**0.00e+00**
	60	0.00e+00	2.61e-15	1.42e-14	3.22e-15	0.00e+00	**0.00e+00**	0.00e+00	**0.00e+00**
f_{18}	30	1.86e-05	**9.93e-05**	7.09e-04	**1.24e-04**	1.22e-01	3.67e-01	6.67e-01	1.87e-01
	60	3.89e-05	**1.83e-04**	8.55e-04	1.49e-04	6.67e-01	6.67e-01	6.67e-01	**1.78e-10**
f_{19}	30	0.00e+00	7.50e-04	2.25e-02	4.11e-03	0.00e+00	**0.00e+00**	0.00e+00	**0.00e+00**
	60	0.00e+00	6.85e-16	1.61e-14	2.92e-15	0.00e+00	**0.00e+00**	0.00e+00	**0.00e+00**
f_{20}	30	3.08e-16	5.06e-16	7.09e-16	7.35e-17	2.14e-31	**1.66e-30**	5.49e-30	**1.19e-30**
	60	7.77e-16	1.21e-15	1.42e-15	1.53e-16	2.75e-30	**9.36e-30**	1.75e-29	**4.45e-30**
f_{21}	30	-29.5462	-29.4879	-29.4313	2.97e-02	-29.6309	**-29.6285**	-29.6212	**2.83e-03**
	60	-59.2538	-59.0873	-58.9513	7.21e-02	-59.5723	**-59.5117**	-59.4571	**3.43e-02**
f_{22}	30	0.00e+00	3.22e-14	5.68e-14	2.86e-14	0.00e+00	**0.00e+00**	0.00e+00	**0.00e+00**
	60	0.00e+00	1.74e-13	4.55e-13	1.11e-13	0.00e+00	**5.31e-14**	1.14e-13	**5.77e-14**
f_{23}	30	3.76e-03	5.12e-02	2.10e-01	5.95e-02	1.49e-02	**4.46e-02**	8.35e-02	**1.90e-02**
	60	2.15e-03	**7.10e-02**	3.34e-01	7.70e-02	8.04e-02	1.79e-01	2.60e-01	**4.29e-02**
f_{24}	30	3.82e-04	4.13e-04	1.16e-03	1.44e-04	3.82e-04	**3.82e-04**	3.82e-04	**8.48e-13**
	60	7.64e-04	7.90e+00	1.18e+02	3.00e+01	7.64e-04	**7.64e-04**	7.64e-04	**2.28e-12**
f_{25}	30	4.02e-16	5.47e-16	7.07e-16	7.8e-17	4.97e-33	**1.37e-32**	2.48e-32	**5.06e-33**
	60	1.1e-15	1.28e-15	1.44e-15	1.15e-16	5.24e-32	**2.84e-31**	5.73e-31	**1.34e-31**
f_{26}	30	3.19e-16	5.25e-16	7.49e-16	9.11e-17	1.23e-31	**4.11e-31**	9.13e-31	**2.07e-31**
	60	9.54e-16	1.21e-15	1.61e-15	1.68e-16	3.25e-30	**1.05e-29**	2.06e-29	**4.01e-30**
f_{27}	30	1.58e+02	2.28e+02	2.83e+02	2.94e+01	2.25e+00	**4.86e+00**	9.79e+00	**1.97e+00**
	60	5.08e+02	6.93e+02	7.89e+02	5.52e+01	2.25e+00	**6.08e+00**	15.01e+00	**3.42e+00**

Tab.5. Comparison of EH-ABC with other algorithms.

	Ackley	Griewank	Rastrigin	Rosenbrock	Schwefel	Sphere
	Mean	Mean	Mean	Mean	Mean	Mean
DE	3.99e-08	6.15e-04	1.47e+02	4.71e+03	7.27e+03	3.43e-14
PSO	3.23e-01	1.34e-02	3.85e+01	5.74e+03	4.16e+03	2.13e-16
CLPSO	2.01e-12	6.45e-13	2.57e-11	1.10e+01	1.19e+01	1.89e-19
CES	6.00e-13	6.00e-14	1.34e+01	2.77e+01	4.57e+03	1.70e-26
FES	1.20e-02	3.70e-02	1.60e-01	3.33e+01	1.31e+01	2.50e-04
ESLAT	1.80e-08	1.40e-03	4.65e+00	1.93e+00	1.03e+04	2.00e-17
CMA-ES	6.90e-12	7.40e-04	5.18e+01	4.00e-01	4.93e+03	9.70e-23
EH-ABC	**0.00e+00**	**0.00e+00**	**0.00e+00**	**4.46e-02**	**3.82e-04**	**1.37e-32**
Sig.	+	+	+	+	+	+

Tab.6. Comparison of EH-ABC with other ABC algorithms.

Fun	GABC	I-ABC	PS-ABC	NABC	EH-ABC	Sig.
	Mean	Mean	Mean	Mean	Mean	
Ackley	7.78e-10	8.88e-16	8.88e-16	1.07e-13	**0.00e+00**	+
Griewank	6.96e-04	**0.00e+00**	**0.00e+00**	1.11e-16	**0.00e+00**	NA
Rastrigin	3.31e-02	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	NA
Rosenbrock	7.48e+00	2.64e+01	1.59e+00	1.45e-01	**4.46e-02**	+
Schwefel	1.62e+02	3.18e+02	5.30e+00	5.73e-01	**3.82e-04**	+
Sphere	6.26e-16	**0.00e+00**	**0.00e+00**	5.43e-16	1.37e-32	•

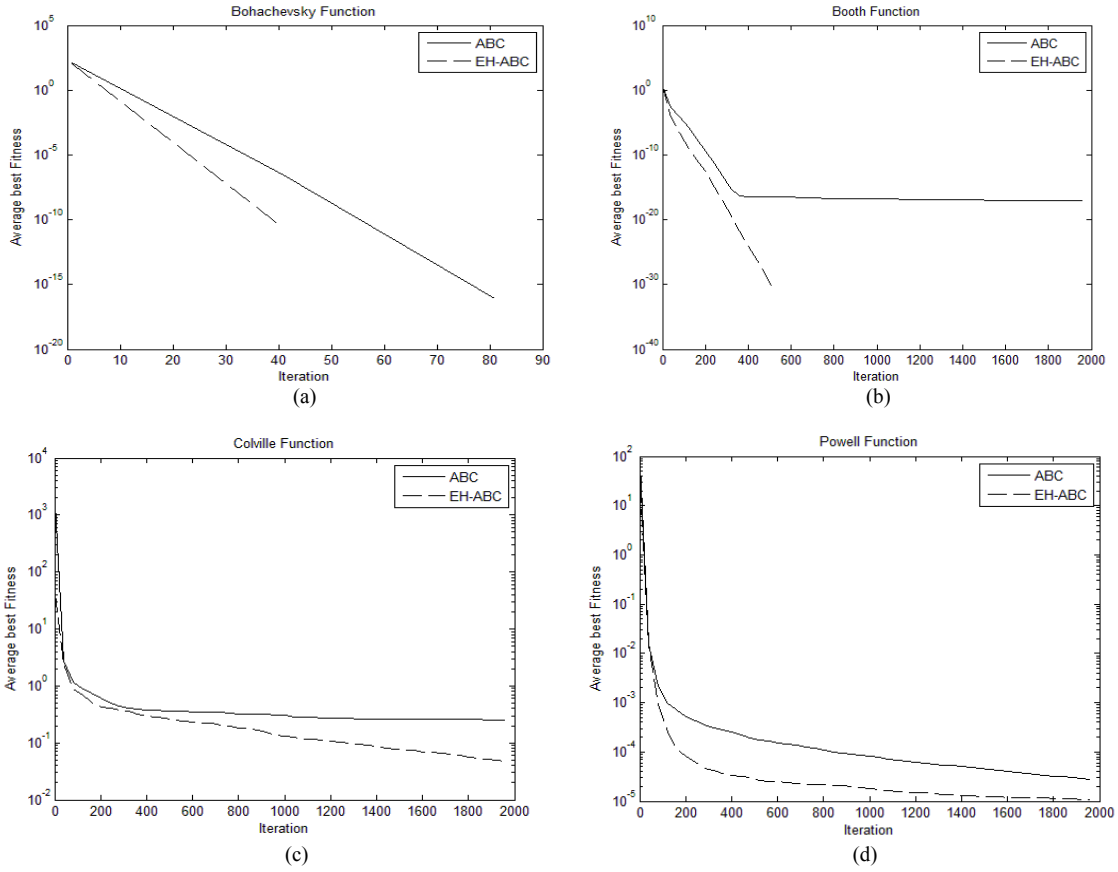


Fig.2. The convergence processes of ABC and EH-ABC on some test functions in Table 1.

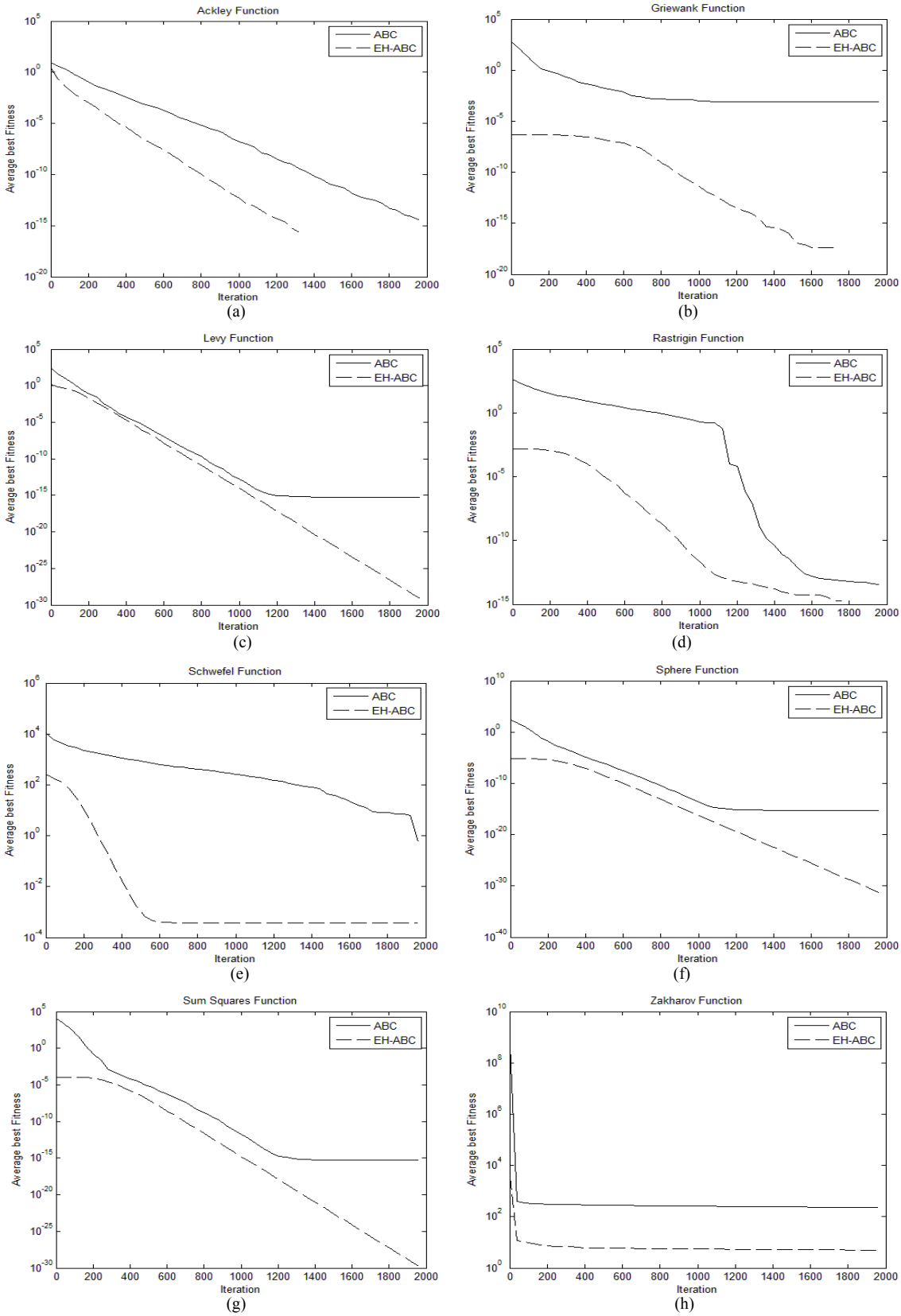


Fig.3. The convergence processes of ABC and EH-ABC on some test functions in Table 2 with *D*=30.

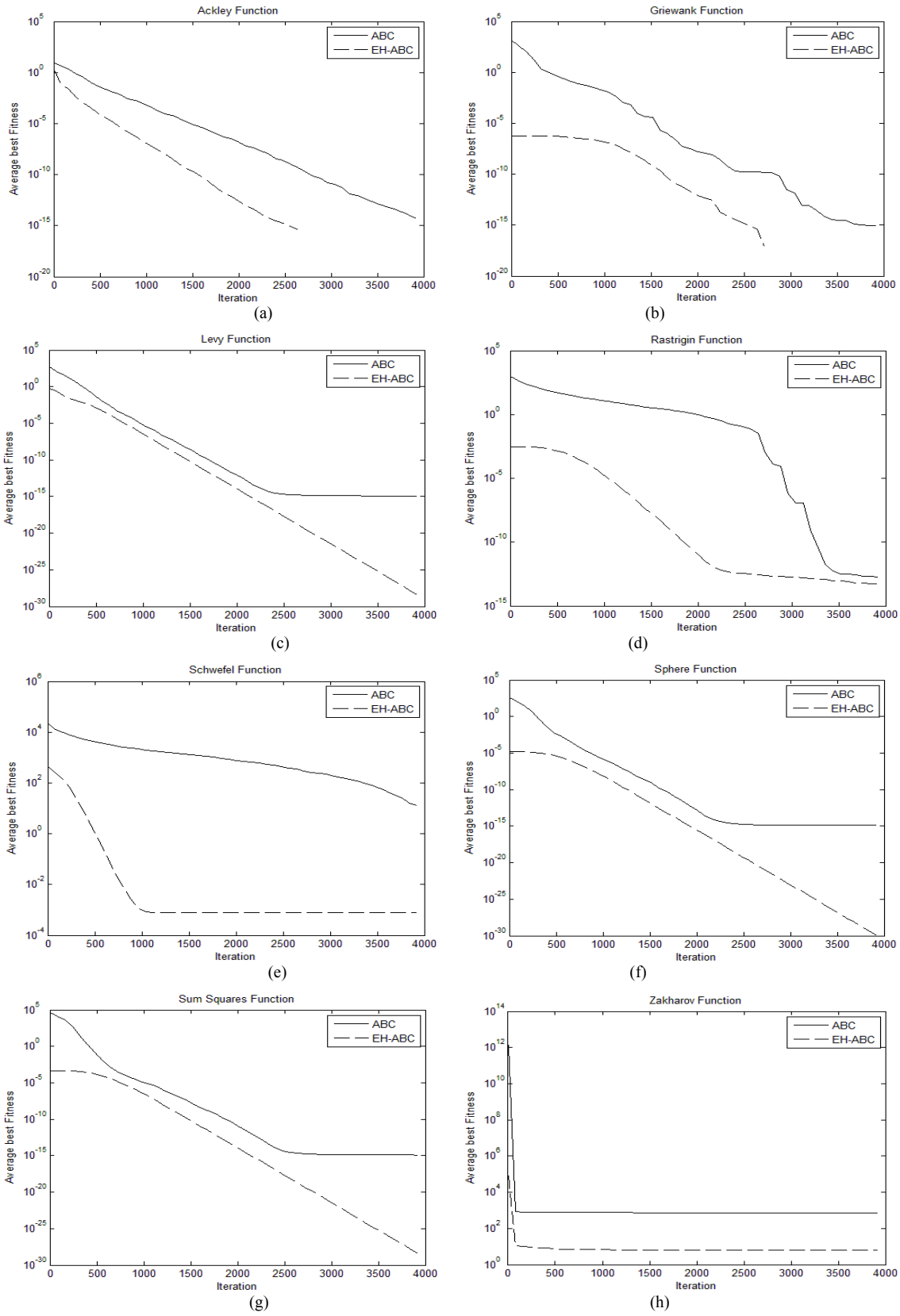


Fig.4. The convergence processes of ABC and EH-ABC on some test functions in Table 2 with D=60.

3.2 Experiments on NLLS problems

3.2.1 Test problems

In this subsection, we perform five NLLS problems in order to illustrate the implementation and efficiency of the EH-ABC algorithm proposed in this paper. All five test problems are illustrated as follows.

NLLS 1: Consider the following NLLS problem, where

$$A = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 4 & -1 \\ 2 & -2 & 1 \end{bmatrix}, \; b = \begin{pmatrix} 6 \\ 5 \\ 1 \end{pmatrix},$$

The optimal solution is $x^* = (1,1,3)^T$.

NLLS 2: Let be a matrix whose diagonal elements are 500 and the non-diagonal elements are chosen randomly from the interval such that A is symmetric. Let $b = Ae$ where e is $n \times 1$ vector whose elements are all equal to unity such that $x^* = (1,1,...,1)^T \in R^n$ is the unique solution.

NLLS 3: Let the matrix A is given by $a_{i,i} = 4n$, $a_{i,i+1} = a_{i+1,i} = n$, $a_{i,j} = 0, i = 1,2,...,n$. Let $b = Ae$. Thus the unique solution is $x^* = (1,1,...,1)^T \in R^n$.

NLLS 4: Following we consider one randomly generated NLLS problem where the data (A,b) are generated by the Matlab scripts: $rand('state',0)$, m=200, n=100, $A = rand(m,n)$, $b = A * ones(n,1)$, where $A \in R^{m \times n}$,and the unique solution is $x^* = (1,1,...,1)^T \in R^n$.

NLLS 5: Following we consider another randomly generated NLLS problem where the data (A,b) are generated by the Matlab scripts: $rand('state',0)$, $n = 100$, $A1 = rand(n,n)$, $A = A1' * A1 + n * eye(n,n)$, $b = A * ones(n,1)$, here A is a positive definite symmetric matrix, and the unique solution is $x^* = (1,1,...,1)^T$.

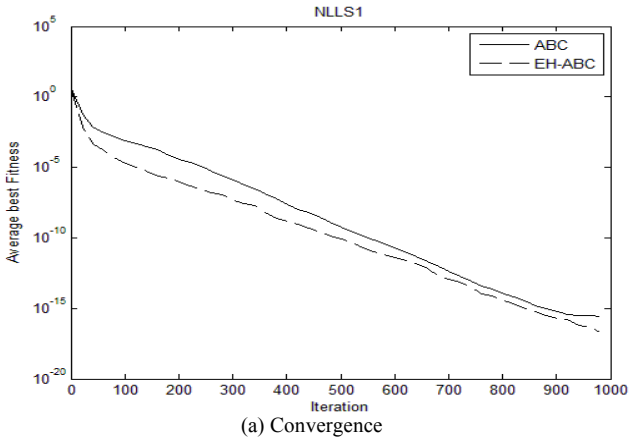
3.2.2 Experiment results

Simulations were carried out to compare the optimization capabilities of the EH-ABC algorithm with respect to Classical ABC algorithm. To make the comparison fair, the populations for all the competitor algorithms were initialized using the same random food sources. In the experiments, both EH-ABC and ABC use the same parameter settings. The population size SN, limit, and maximum number of cycle (MSN) are set to 60, $(SN/2) * D$, 1000, respectively. To judge the accuracy of different algorithms, 30 independent runs of each of the two algorithms were carried out and the best, the mean, the worst fitness values, and the standard deviation (Std) were recorded. Table 7 compares the algorithms on the quality of the optimum solution for given NLLS problems, and the best results are marked in bold.

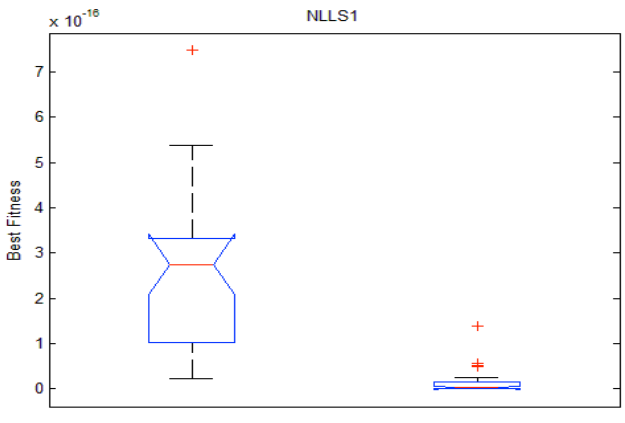
Figure 5-9 show the convergence and its boxplot figure of the best fitness in the population for the different algorithms. The values plotted for every generation are averaged over 30 independent runs. As can be seen, the EH-ABC algorithm is the best not only for simple NLLS problems, but also for complex NLLS problems.

Tab.7. The statistical results for 30 runs tested on given NLLS problems.

Functions	Algorithms	Best	Mean	Worst	Std
NLLS 1	ABC	2.36e-17	2.65e-16	7.49e-16	1.87e-16
	EH-ABC	7.05e-21	**1.42e-17**	1.38e-16	**2.83e-17**
NLLS 2	ABC	7.98e+01	2.53e+02	7.47e+02	1.69e+02
	EH-ABC	2.13e+01	**3.85e+01**	6.21e+01	**9.13e+00**
NLLS 3	ABC	4.93e+03	1.69e+04	3.77e+04	9.27e+04
	EH-ABC	2.36e+02	**2.85e+02**	3.61e+02	**3.28e+01**
NLLS 4	ABC	9.25e+01	1.66e+02	3.06e+02	4.94e+01
	EH-ABC	5.17e-01	**9.71e-01**	2.08e+00	**4.23e-01**
NLLS 5	ABC	2.52e+05	4.28e+05	6.52e+05	1.22e+05
	EH-ABC	6.40e+02	**1.35e+03**	3.43e+03	**6.69e+02**

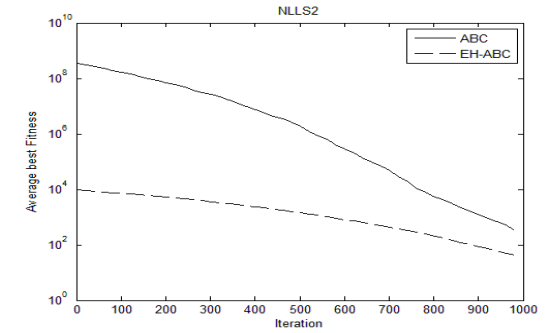


(a) Convergence

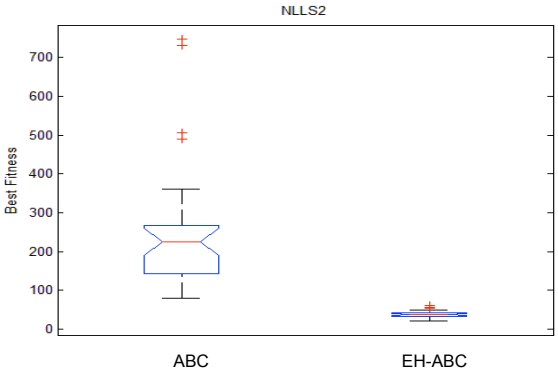


(b) Boxplot

Fig.5. The convergence and its boxplot of the best fitness for NLLS1.

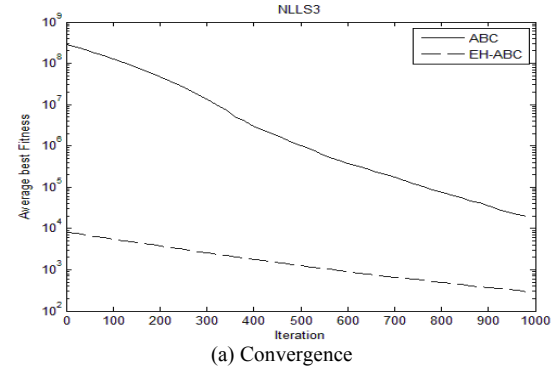


(a) Convergence

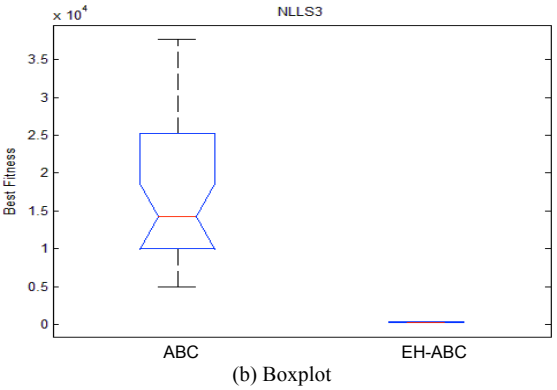


(b) Boxplot

Fig.6. The convergence and its boxplot of the best fitness for NLLS2 with n=100.

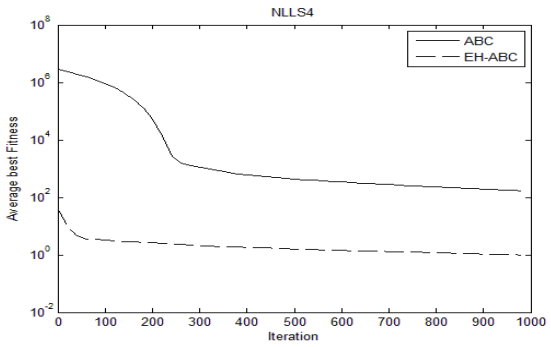


(a) Convergence

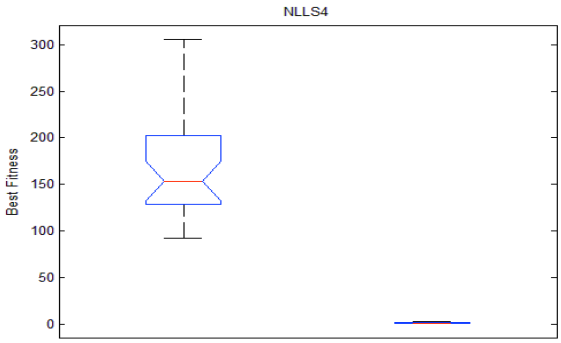


(b) Boxplot

Fig.7. The convergence and its boxplot of the best fitness for NLLS3 with n=100.

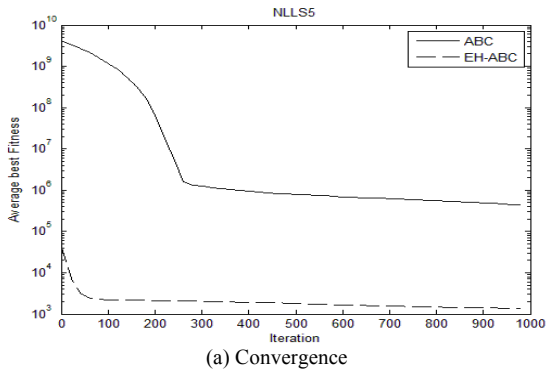


(a) Convergence

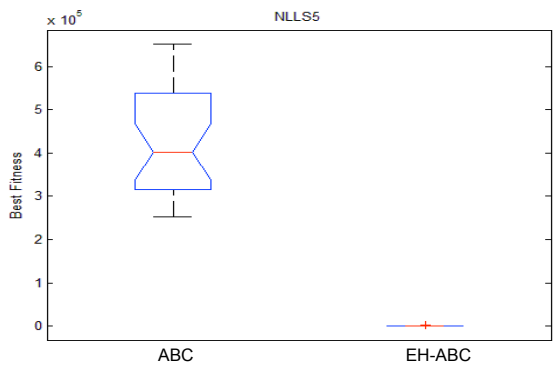


(b) Boxplot

Fig.8. The convergence and its boxplot of the best fitness for NLLS4 with n=100.



(a) Convergence



(b) Boxplot

Fig.9. The convergence and its boxplot of the best fitness for NLLS5 with n=100.

4. Conclusions

Artificial bee colony is a new swarm-based optimization technique which has shown to be competitive to other population-based stochastic algorithms. However, ABC and other stochastic algorithms suffer from the same problems,

such as lower convergence speed and easily trapped in local optima when handling complex multimodal problems. The main reason is that the search pattern is good at exploration but poor at exploitation. To overcome this issue, an effective hybrid artificial bee colony algorithm (EH-ABC) is proposed. In EH-ABC algorithm, orthogonal initial method is employed and a new search mechanism is designed.

To verify the performance of the proposed algorithm, a set of 27 test functions and 5 nonnegative linear least squares test problems are used in the experiments. Comparison of EH-ABC with other algorithms indicates that EH-ABC can effectively accelerate the convergence speed and improve the accuracy of solutions. Therefore, the EH-ABC algorithm proposed in our paper is more effective for NLLS problems.

Acknowledgement

This work is supported by 2013 Narure Science Foundation of Ningxia (No. NZ13096), 2013 Higher educational scientific research project of Ningxia (No. NGY2013086), 2013 scientific research project of Beifang University of Nationalities (2013XYZ021), institute of information and system computation science of Beifang University (13xyb01), National Nature Science Foundation of China (No. 61373174) and Foundation of State Key Lab. of Integrated Services Networks of China.

References

1. Dervis KARABOGA, "*An Idea Based On Honey Bee Swarm for Numerical Optimization*", Technical Report TR06, Erciyes University, Engineering Faculty, Computer Engineering Department, 2005.
2. Dervis Karaboga, Bahriye, Akay, "A comparative study of artificial bee colony algorithm", *Applied Mathematics and Computation*, 214(1), 2009, pp. 108-132.
3. Karaboga, D., Akay, B., "A modified Artificial Bee Colony (ABC) algorithm for constrained optimization", *Applied Soft Computing*, 11(3), 2011, pp. 3021-3031.
4. D. Karaboga, C. Ozturk, B. Akay, "*Training neural networks with ABC optimization algorithm on medical pattern classification*", International Conference on Multivariate Statistical Modeling and High Dimensional Data Mining (Kayseri, TURKEY), June 19-23, 2008.
5. C. Ozturk, D. Karaboga, "*Classification by neural networks and clustering with artificial bee colony (ABC) algorithm*", in Proceedings of the 6th International Symposium on Intelligent and Manufacturing Systems, Features, Strategies and Innovation (Sakarya, Turkiye), October 14-17, 2008.
6. Karaboga, D., Akay, B., "*Artificial bee colony (ABC) algorithm on training artificial neural networks*", Signal Processing and Communications Applications, 2007. SIU 2007. IEEE 15th, June 2007, pp. 1-4.
7. Karaboga, D., Akay, B., Ozturk, C., "*Modeling decisions for artificial intelligence, Artificial Bee Colony (ABC) Optimization Algorithm for Training Feed-Forward Neural Networks*", LNCS 4617, Springer-Verlag, 2007, pp. 318-329.
8. Wenping Zou, Yunlong Zhu, Hanning Chen, and Xin Sui, "A Clustering Approach Using Cooperative Artificial Bee Colony Algorithm" , *Discrete Dynamics in Nature and Society*, vol. 2010, Article ID 459796, 16 pages, 2010. doi:10.1155/2010/459796.
9. Sonam Mittal, Neha Nirwal,Harsh Sardana, "Enhanced artificial bees colony algorithm for traveling salesman problem", *Journal of Advanced Computing and Communication Technologies*, 2(2), 2014, pp. 1-3.
10. Cabrera G Guillermo., Cabrera Enrique, Soto Ricardo, Miguel Rubio L. Jose, Crawford Broderick, Paredes Fernando, "A Hybrid Approach Using an Artificial Bee Algorithm with Mixed Integer Programming Applied to a Large-Scale Capacitated Facility Location Problem", *Mathematical Problems in Engineering*, vol. 2012, Article ID 954249, 14 pages, 2012. doi:10.1155/2012/954249.
11. Kong, X., et al. , "Hybrid Artificial Bee Colony Algorithm for Global Numerical Optimization", *Journal of Computational Information Systems*, 8(6), 2012, pp. 2367-2374.
12. Liang, Jing J., Qin, A. K., Suganthan, Ponnuthurai Nagaratnam, Baskar, S., "Comprehensive learning particle swarm optimizer for global optimization of multimodal functions", *IEEE Transactions on Evolutionary Computation*, 10 (3), 2006, pp. 281-295.
13. Zhu, Guopu, Kwong Sam, "Gbest-guided artificial bee colony algorithm for numerical function optimization", *Applied Mathematics and Computation*, 217(7), 2010, pp. 3166-3173.
14. Yao, Xin, Liu, Yong, "Fast evolution strategies", *Lecture Notes in Computer Science*, 1213, 1997, pp. 467-496.
15. Hansen, Nikolaus, Ostermeier, Andreas, "*Adapting arbitrary normal mutation distributions in evolution strategies: the covariance matrix adaptation*", Proceedings of the IEEE Conference on Evolutionary Computation, May 1996, pp. 312-317.
16. Guoqiang Li, Niu Peifeng, Xiao Xingjun, "Development and investigation of efficient artificial bee colony algorithm for numerical function optimization", *Applied Soft Computing Journal*, 12 (1), 2012, pp.320-332.
17. Xu Yunfeng, Fan Ping, Yuan Ling, "A Simple and Efficient Artificial Bee Colony Algorithm," *Mathematical Problems in Engineering*, vol. 2013, Article ID 526315, 9 pages, 2013. doi:10.1155/2013/526315.
18. Das Swagatam, Abraham Ajith, Chakraborty Uday K., Konar, Amit, "Differential evolution using a neighborhood-based mutation operator", *IEEE Transactions on Evolutionary Computation*, 13(3), 2009, pp. 526-553.