2014 IEEE Congress on Evolutionary Computation (CEC)
July 6-11, 2014, Beijing, China

A Novel Artificial Bee Colony Algorithm with Integration of Extremal Optimization for Numerical Optimization Problems

Min-Rong Chen[1,2]
[1]School of Computer Science
South China Normal University
Guangzhou, China
[2] College of Information Engineering
Shenzhen University
Shenzhen, China
Email: optmrchen@gmail.com

Guo-Qiang Zeng
Department of Electrical and Electronic Engineering,
Wenzhou University,
Wenzhou, China

Xia Li
College of Information Engineering
Shenzhen University
Shenzhen, China

Wei Zeng
College of Information Engineering
Shenzhen University
Shenzhen, China

Jian-Ping Luo
College of Information Engineering
Shenzhen University
Shenzhen, China

***Abstract*—Artificial Bee Colony (ABC) algorithm is an optimization algorithm based on a particular intelligent behaviour of honeybee swarms. The standard ABC is weak at the local-search capability and precision. Extremal Optimization (EO) is a general-purpose heuristic method which has strong local-search capability and has been successfully applied to a wide variety of hard optimization problems. In order to strengthen the local-search capability of ABC, this work proposes a novel hybrid optimization method, called ABC-EO algorithm, through introducing EO to ABC. The simulation results show that the performance of the proposed method is as good as or superior to those of the state-of-the-art algorithms in complex numerical optimization problems.**

Keywords—Artificial Bee Colony; Extremal Optimization; numerical optimization problems

I. INTRODUCTION

Artificial Bee Colony (ABC) algorithm is a novel swarm intelligent algorithm inspired by the foraging behaviors of honeybee. It was first introduced by Karaboga in 2005 [1]. After that, ABC was applied to solving the binding numerical optimization problems by Karaboga and Basturk[2], and satisfactory results were achieved. Since the ABC algorithm has many advantages, such as simple in concept, easy to implement, and fewer control parameters, it has attracted the attention of many researchers and been used in solving many real-world optimization problems [3-7].

This work was supported by National Natural Science Foundation of China No. 61005049, 61373158, 51207112, 61171124, 61301298.

In 1999, a general-purpose local-search optimization approach, so-called Extremal Optimization (EO), was proposed by Boettcher and Percus [8,9]. EO is an optimization heuristic inspired by the Bak-Sneppen model [10], and thus EO is based on the fundamentals of statistical physics and Self-Organized Criticality (SOC) [11]. The evolution in this method is driven by a process in which the weakest species in the population, together with its nearest neighbors, is always forced to mutate. EO successively eliminates those worst components in the sub-optimal solutions and has been successfully applied to many continuous and discrete optimization problems [12-17].

However, the standard ABC algorithm also has its limitations, such as premature convergence, slow convergence speed at the later stage of evolution and low convergence accuracy. In order to overcome the limitations of ABC, inspired by the hybrid PSOEO algorithm proposed by Chen et al.[18], an idea of combining ABC with EO is addressed in this paper. In this work, we develop a hybrid optimization method, called ABC-EO algorithm, which makes full use of the global-search ability of ABC and the local-search ability of EO. The performance of the proposed approach was testified on six unimodal/multimodal benchmark functions and furthermore the ABC-EO algorithm was compared with other five state-of-the-art optimization algorithms, i.e., standard ABC, PSOEO[18], standard Particle Swarm Optimization (PSO), Population-based EO (PEO) [15] and standard Genetic

Algorithm (GA). The experimental results indicate that the proposed approach may be a good alternative for complex numerical optimization problems.

This paper is organized as follows. In Section II, ABC and EO algorithms are briefly introduced. In Section III, we propose the hybrid ABC-EO method and describe it in detail. In Section IV, the proposed approach is used to solve six unconstrained benchmark functions from the usual literature. Finally, the simulation results obtained are presented and discussed in Section V.

II. ARTIFICIAL BEE COLONY AND EXTREMAL OPTIMIZATION

A. Artificial Bee Colony(ABC)

ABC algorithm is a recently proposed optimization algorithm that simulates the foraging behavior of a bee colony. In the ABC algorithm, the search space corresponds to a food source that the artificial bees can exploit. The position of a food source represents a possible solution to the optimization problem. The nectar amount of a food source represents the fitness of the associated solution. There are three kinds of bees in a bee colony: employed bees, onlooker bees and scout bees. Half of the colony comprises employed bees and the other half includes the onlooker bees.

Artificial colony search activities can be summarized as follows [20]: Initially, the ABC generates a randomly distributed initial population of *SN/2* solutions (i.e., food source positions), where *SN* denotes the size of population. Each solution X_i *(i=1,2,…,SN/2)* is a *D*-dimensional vector. Here, *D* is the number of optimization parameters. After initialization, the population of solutions is subject to repeated cycles of the search processes of the employed bees, the onlooker bees and the scout bees. Employed bees exploit the specific food sources they have explored before and give the quality information about the food sources to the onlooker bees waiting outside the hive. Onlooker bees receive information about the food sources and choose a food source to exploit depending on the quality information. The more nectar the food source contains, the larger probability the onlooker bees choose it. In the ABC algorithm, one of the employed bees is selected and classified as the scout bee. The classification is controlled by a control parameter called "limit". If a solution representing a food source is not improved by a predetermined number of trials, then that food source is abandoned by its employed bee and the employed bee associated with that food source becomes a scout. Here we use "trial" to record the non-improvement number of the solution X_i , used for the abandonment. Finally, scout bees search the whole environment randomly.

Note that each food source is exploited by only one employed bee. That is, the number of the employed bees or the onlooker bees is equal to the number of food sources.

The pseudo-code of the standard ABC algorithm is described in Fig.1.[21].

1. Initialize the food source positions;
2. Evaluate the nectar amount (i.e., fitness) of each food source;
3. cycle=1
4. repeat (if the termination conditions are not met)
5. Employed Bees Phase
6. Calculate probabilities for onlooker bees;
7. Onlooker Bees Phase
8. Scout Bees Phase
9. Memorize the best solution found so far;
10. cycle=cycle+1
11. until cycle=Maximum Cycle Number

Fig.1. Pseudo-code of ABC algorithm

In order to produce a candidate food position $X_i^{'}$ from the old one X_i in memory, the ABC uses the following expression [21]:

$$X_{i,j}^{'} = X_{i,j} + \varphi_{i,j}(X_{i,j} - X_{k,j}) \qquad (1)$$

where $k \in \{1,2,...,SN/2\}$ and $j \in \{1,2,...,D\}$ are randomly chosen indexes; *k* has to be different from *i*; *D* is the number of variables(problem dimension); φ_{ij} is a random number between [-1,1].

The pseudo-code of Employed Bees Phase of ABC algorithm is shown as Fig.2.[21].

1. for *i*=1 to *SN*/2 do
2. for *j*=1 to *D* do
3. Produce a new food source $X_i^{'}$ for the employed bee of the food source X_i using (1);
4. End for
5. Evaluate the fitness of $X_i^{'}$;
6. Apply the selection process between $X_i^{'}$ and X_i based on greedy selection;
7. If the solution $X_i^{'}$ does not improve , let trial=trial+1,otherwise trial=0
8. End for

Fig.2. Pseudo-code of Employed Bees Phase

An artificial onlooker bee chooses a food source depending on the probability value (denoted as P), which is associated with that food source. P is calculated by the following expression [22]:

$$\mathrm{P} = \frac{fit_i}{\sum_{n=1}^{SN/2} fit_n} \qquad (2)$$

where fit_i is the fitness value of the solution X_i which is proportional to the nectar amount of the food source in the position X_i , and *SN/2* is the number of food sources which is equal to the number of employed bees or onlooker bees.

The pseudo-code of Onlooker Bees Phase of ABC algorithm is shown as Fig.3.[21].

1. t=0,*i*=1
2. repeat (if the termination conditions are not met)
3. if random<P then (Note that P is calculated by (2))
4. t=t+1
5. for *j*=1 to *D* do
6. Produce a new food source $X_i^{'}$ for the onlooker bee of the food source X_i by using (1);
7. End for
8. Apply the selection process between $X_i^{'}$ and X_i based on greedy selection;
9. If the solution X_i does not improve, let trial=trial+1,otherwise trial=0
10. End if
11. *i*=*i*+1;
12. *i*=*i* mod(*SN*/2+1);
13. until t= *SN*/2.

Fig.3. Pseudo-code of Onlooker Bees Phase

The positions of the new food sources found by the scout bees will be produced by the following expression [22]:

$$X_{i,j}^{'} = X_{\min,j} + rand(0,1)(X_{\max,j} - X_{\min,j}) \qquad (3)$$

where *i* is the index of the employed bees whose "trial" value reaches the "limit" value first, $j = 1,2,...,D$, $X_{\min}$ and $X_{\max}$ are the lower bound and the upper bound of each solution respectively, and rand(0,1) is a random number between [0,1].

The pseudo-code of Scout Bees Phase of ABC algorithm is worked as Fig.4.[21].

1. If max(trial)>limit then
2. Replace X_i with a new randomly produced solution $X_i^{'}$ by (3);
3. End if

Fig.4. Pseudo-code of Scout Bees Phase

The fitness of ABC algorithm is proportional to the nectar amount of that food source. The fitness is determined by (4) and (5)[21]:

$$fitness_i = 1/(1+f_i) \quad \text{if } f_i \geq 0 \qquad (4)$$

$$fitness_i = 1 + abs(f_i) \quad \text{if } f_i < 0 \qquad (5)$$

where f_i is the cost value of the solution X_i and $abs(f_i)$ is the absolute value of f_i.

B. Extremal Optimization(EO)

Extremal Optimization (EO) is inspired by recent progress in understanding far-from-equilibrium phenomena in terms of self-organized criticality, a concept introduced to describe emergent complexity in physical systems. EO successively updates extremely undesirable variables of a single sub-optimal solution, assigning them new random values. Moreover, any change in the fitness value of a variable engenders a change in the fitness values of its neighboring variable. Large fluctuations emerge dynamically, efficiently exploring many local optima [23]. Thus, EO has strong local-search ability.

Note that in the EO algorithm, each variable in the current solution *X* is considered "species". In this study, we adopt the term "component" to represent "species" which is usually used in biology. For example, if $X = (x_1, x_2, x_3)$, then x_1 , x_2 and x_3 are called "components" of *X*. From the EO algorithm, it can be seen that unlike genetic algorithms which work with a population of candidate solutions, EO evolves a single sub-optimal solution *X* and makes local modification to the worst component of *X*. A fitness value λ_i is required for each component x_i in the problem. In each iteration, components are ranked according to the value of their fitness. This differs from holistic approaches such as evolutionary algorithms that assign equal-fitness to all components of a solution based on their collective evaluation against an objective function. The pseudo-code of EO algorithm for a minimization problem is shown in Fig. 5 [18].

1. Randomly generate a solution $X = (x_1, x_2, ..., x_D)$. Set optimal solution X_{best}=X and the minimum cost function $C(X_{best})$=$C(X)$.
2. For the current solution *X*,
   (a) evaluate the fitness λ_i for each component x_i , $i \in \{1, 2, \cdots, D\}$,
   (b) rank all the fitness and find the component x_j with the lowest fitness, i.e., $\lambda_j \leq \lambda_i$ for all *i*,
   (c) choose one solution *X'* in the neighborhood of *X*, such that the *j*-th component must change its state,
   (d) accept *X*=*X'* unconditionally,
   (e) if $C(X) < C(X_{best})$ then set X_{best}=X and $C(X_{best})$=$C(X)$.
3. Repeat Step 2 as long as desired.
4. Return X_{best} and $C(X_{best})$.

Fig.5. Pseudo-code of EO procedure

III. THE PROPOSED APPROACH

Note that ABC has great global-search ability, while EO has strong local-search capability. In this work, we propose a novel hybrid ABC–EO algorithm which combines the merits of ABC and EO. This hybrid approach makes full use of the exploration ability of ABC and the exploitation ability of EO. When the global optimum found by ABC algorithm is unchanged for several iterations, which indicates that ABC has got trapped into local optima, we will use EO to help ABC to escape form local optima. Consequently, through introducing EO to ABC, the proposed approach may overcome the limitations of ABC and have capability of escaping from local optima. However, if EO is introduced to ABC each iteration, the computational cost will increase sharply. And at the same time, the fast convergence ability of ABC may be weakened.

In order to perfectly integrate ABC with EO, EO is introduced to ABC when the global optimal solution (i.e., X_{best}) is unchanged continuously for *INV*-iterations. Therefore, the hybrid ABC–EO approach is able to keep fast convergence in most of the time under the help of ABC, and capable of escaping from a local optimum with the aid of EO. The value of parameter *INV* is predefined by the user according to the complexity of problems.

A. Hybrid ABC-EO algorithm

To improve the efficiency and accuracy of the standard ABC, in this study, we present two improved versions of ABC-EO. One is the combination of standard ABC and EO, and the other is the combination of IABC[24] and EO, in which its search way of employed bees is changed as follows[24]:

$$X'_{i,j} = X_{best,j} + \varphi_{i,j}(X_{best,j} - X_{k,j}) \qquad (6)$$

We called them ABC-EO and IABCEO, respectively. The pseudo-code of ABC-EO and IABC-EO for a minimization problem with *D* dimensions is described in Fig.6.

1. Initialize the food source positions and set iteration=0.
2. Evaluate the nectar amount (i.e. fitness) of food sources, and the search way of employed bees is changed according to (1) (for ABC-EO algorithm) or (6) (for IABC-EO algorithm).
3. If the global optimal solution X_{best} is unchanged for *INV* iterations, then the EO procedure is introduced to change the positions of food sources. Otherwise, continue the next step.
4. If the terminal condition is satisfied, go to the next step; otherwise, set iteration=iteration+1, and go to Step 2.
5. Output the optimal solution and the optimal objective function value.

Fig.6. Pseudo-code of ABC-EO and IABC-EO algorithm

In the main procedure of ABC-EO algorithm, the fitness of each individual is evaluated by (4) and (5). However, in the EO procedure, in order to find out the worst component, each component of a solution should be assigned a fitness value. We define the fitness of each component of a solution for an unconstrained minimization problem as follows. For the *i*-th position of food sources, the fitness $\lambda_{i,k}$ of the *k*-th component is defined as the mutation cost, i.e. $OBJ(X'_{i,k}) - OBJ(X_{best})$, where $X'_{i,k}$ is the new position of the *i*-th position obtained by performing mutation only on the *k*-th component and leaving all other components fixed, $OBJ(X'_{i,k})$ is the objective value of $X'_{i,k}$, and $OBJ(X_{best})$ is the objective value of the best position in the bee colony found so far. The EO procedure is described in Fig.7.

B. Mutation operator

Since there is merely mutation operator in EO, the mutation plays a key role in EO search. In this work, we adopt the hybrid Gaussian-Cauchy mutation (G-C mutation for short) [18], which combines the coarse search and grained search perfectly.

1. For each position $X_i = (X_{i,1}, X_{i,2}, ..., X_{i,D})$ of the food source, $i = 1, \cdots, SN/2$
   a) Perform mutation on each component of X_i one by one, while keeping other components fixed. Then *D* new positions $X'_{i,k}(k = 1, ..., D)$ can be obtained;
   b) Evaluate the fitness $\lambda_{ik} = OBJ(X'_{i,k}) - OBJ(X_{best})$ of each component $X_{i,k}, k \in \{1, 2, ..., D\}$;
   c) Compare all the components according to their fitness values and find out the worst adapted component $X_{i,w}$, and then $X'_{i,w}$ is the new position corresponding to $X_{i,w}, w \in \{1, 2, ..., D\}$;
   d) If $OBJ(X'_{i,w}) < OBJ(X_i)$, then set $X_i = X'_{i,w}$ and $OBJ(X_i) = OBJ(X'_{i,w})$, and update X_{best} using X_i; Otherwise X_i keeps unchanged;

Fig.7. Pseudo-code of EO procedure

C. Differences from EABC

Note that Azadehgan et al. [19] have proposed a hybrid algorithm called EABC, which also combines ABC with EO. In the EABC algorithm, EO was used to determine how to choose the neighbor of employed bees or onlooker bees, i.e. X_k in (2). While in our proposed algorithm, EO is introduced to update the positions of food sources when the global optimal position is unchanged for several iterations. The EABC in the literature [19] was applied to solving three numerical optimization problems. However, they did not explain the mechanism of the proposed algorithm in detail and the experimental results were poor [19]. Readers may refer to the literature [19] for more detail. Thus, in this paper, our proposed algorithms are not compared with EABC.

IV. EXPERIMENTS AND RESULTS

In order to demonstrate the performance of the proposed hybrid ABC-EO, we use six well-known benchmark functions shown in Table I. All the functions are to be minimized. The experimental results of the proposed approach are compared with five state-of-the-art algorithms, i.e., standard ABC, PSOEO, standard PSO, PEO and GA. For these functions, there are many local optima and/or saddles in their solution spaces. The amount of local optima and saddles increases with increasing complexity of the functions, i.e. with increasing dimension.

Note that all the algorithms were run on the same hardware and software platform. Each algorithm was run independently for 50 trials. *INV* in our proposed algorithms is set to 100 for each test function. Table II shows the settings of problem dimension, maximum generation, population size and initialization range of each algorithm.

After 50 trials of running each algorithm for each test function, the simulation results were obtained and shown in Table III-Table VIII. Denote F as the result found by the algorithms and F^* as the optimum value of the functions (F_1^*= -9.66, F_2^*= -12569.5, $F_3^*=F_4^*=F_5^*=F_6^*$=0). The simulation is considered successful, or in other words, the near-optimal solution is found, if F satisfies that$|(F^*-F)/F^*|$<1E-3(for the case $F^* \neq 0$) or $|F^*-F|$<1E-3(for the case F^*=0). In these tables, "Success" represents the successful rate, and "Runtime" is the average runtime of fifty runs when the near-optimal solution is found or otherwise when the maximum generation is reached. The Worst, Mean, Best and Standard deviation of solutions found by all the algorithms are also listed in these tables.

The Michalewicz function is a highly multimodal test function. As can be seen from Table III, IABCEO, ABCEO, ABC and PSOEO could find the global optimum with 100% successful rate, GA could find the global optimum with 56% successful rate, but PSO and PEO algorithm could not find the global optimum. It can be observed that ABCEO has the fastest convergence speed, and IABCEO converged to the global optimum almost as quickly as ABC, and faster than PSOEO, PSO, PEO and GA. IABCEO also had a good performance in terms of stability.

With regard to the Schwefel function, its surface is composed of a great number of peaks and valleys. The function has a second best minimum far from the global minimum where many search algorithms are trapped. Also it is very hard to solve for many state-of-the-art optimization algorithms. From Table IV, it is clear that IABCEO was the winner which was capable of converging to the global optimum with the 100% successful rate, the fastest convergence speed and the lowest standard deviation. ABCEO and ABC were better than PSOEO, PSO, PEO and GA in terms of successful rate, the convergence speed and solution accuracy. It is interesting to notice that IABCEO and ABCEO converged to the global optimum more than 100 times faster than PSOEO and PEO.

Table V and Table VII show the simulation results of each algorithm on functions Griewank and Ackley, respectively. Both of the two functions are highly multimodal. As can be seen from Table V, all algorithms could find the optimal solution with 100% successful rate, except for GA. But IABCEO, ABCEO and ABC were a little worse than PSOEO and PSO with respect to solution accuracy. From Table VII, we can see that IABCEO and ABCEO could find the optimum with 100% successful rate in a short time. At the same time, Table VII indicates that the proposed algorithms were not better than PSO and PSOEO, but better than PEO and GA with respect to convergence speed and solution accuracy.

From Table VI which shows the simulation results of each algorithm on the function Rastrigin, we can see that PSO and PSOEO were the best performers in all aspects, and IABCEO and ABCEO could find the optimal solution in a short time with higher successful rate and solution accuracy, almost as good as ABC.

The last test function Rosenbrock is a unimodal function, the global optimal point of the function is located in a long, narrow flat valley. Search to the canyon is very easy, but for most optimization algorithms, it is very difficult to converge to the global optimal point. As can be seen from Table VIII, IABCEO was capable of finding the optimal solution quickly with 96% successful rate. Moreover, IABCEO converged to the global optimum more than 60 times faster than PSOEO, although PSOEO could find the optimum with 100% successful rate. IABCEO significantly outperformed other algorithms, except for PSOEO, in terms of solution quality, convergence speed and successful rate.

From the simulation results, it can be concluded that the proposed approaches possess good or superior performance in solution accuracy, convergence speed and successful rate, as compared to standard ABC, PSOEO, standard PSO, PEO and standard GA. As a result, our approaches can be considered as perfectly good performers in optimization of those complex high-dimensional functions.

V. CONCLUSION AND FUTURE WORK

In this paper, we have developed a novel hybrid optimization method, called ABC-EO algorithm, through introducing EO to ABC. The hybrid approach combines the exploration ability of ABC with the exploitation ability of EO, and thus has strong capability of preventing premature convergence. Compared with standard ABC, PSOEO, standard PSO, PEO and standard GA on six well-known benchmark functions, the proposed algorithms have been testified to have good or superior performance in terms of solution accuracy, convergence speed and successful rate. The simulation results demonstrate that ABC-EO is well suited to those complex unimodal/multimodal functions with high dimension. As a result, ABC-EO may be a promising tool to deal with complex numerical optimization problems. It is desirable to further apply ABC-EO to handling those combinatorial optimization problems, such as production scheduling, vehicle routing and graph coloring.

TABLE I TEST FUNCTIONS

Function	Function expression	Search space	Global minimum
Michalewicz	$f_1(x) = -\sum_{i=1}^{n} \sin(x_i) \sin^{2m}\left\{\frac{(i+1)x_i^2}{\pi}\right\}, m = 10$	$(0,\pi)$	-9.66
Schwefel	$f_2(x) = -\sum_{i=1}^{n}\left(x_i \sin\left(\sqrt{\lvert x_i\rvert}\right)\right)$	(-500,500)	-12569.5
Griewank	$f_3(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	(-600,600)	0
Rastrigin	$f_4(x) = \sum_{i=1}^{n}\left[x_i^2 - 10\cos(2\pi x_i) + 10\right]$	(-5.12,5.12)	0
Ackley	$f_5(x) = 20 + e - 20e^{\left[-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right]} - e^{\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)}$	(-32.768,32.768)	0
Rosenbrock	$f_6(x) = \sum_{i=1}^{n-1}\left[100\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2\right]$	(-30,30)	0

TABLE II PARAMETER SETTINGS

Function	Dimension	Maximum generation	Population size	Initialization range
Michalewicz	10	20000	10	$(0,\pi)$
Schwefel	30	20000	30	(-500,500)
Griewank	30	20000	30	(-600,600)
Rastrigin	30	20000	10	(-5.12,5.12)
Ackley	30	10000	30	(-32.768,32.768)
Rosenbrock	30	100000	30	(-30,30)

TABLE III COMPARISON RESULTS FOR MICHALEWICZ FUNCTION F_1 (F_1*=-9.66)

Algorithm	Runtime(s)	Success(%)	Best	Mean	Worst	Standard deviation
IABCEO	0.27	**100**	**-9.66**	**-9.66**	**-9.66**	**8.12E-5**
ABCEO	**0.258**	**100**	**-9.66**	**-9.66**	**-9.66**	8.75E-5
ABC	0.272	**100**	**-9.66**	**-9.66**	**-9.66**	8.75E-5
PSOEO	0.563	**100**	**-9.66**	**-9.66**	**-9.66**	9.02E-5
PSO	0.71	0	-9.65	-9.34	-8.44	0.28
PEO	8.131	0	-9.61	-9.55	-9.49	0.029
GA	0.567	56	**-9.66**	-9.63	-9.46	0.037

TABLE IV COMPARISON RESULTS FOR SCHWEFEL FUNCTION F_2 （F_2*=-12569.5）

Algorithm	Runtime(s)	Success(%)	Best	Mean	Worst	Standard deviation
IABCEO	**0.084**	**100**	**-12569.5**	**-12569.5**	-12569.4	**0.03**
ABCEO	0.172	**100**	**-12569.5**	-12569.4	-12569.4	0.04
ABC	0.163	**100**	**-12569.5**	-12569.4	-12569.4	**0.03**
PSOEO	28.38	78	**-12569.5**	-12543.4	-12451	49.55
PSO	1.508	0	-11532.1	-9382.2	-7599.5	933.74
PEO	45.193	2	-12561.7	-12254.3	-12095.7	115.84
GA	1.685	0	-9845.2	-8690.9	-7693.5	489.03

TABLE V COMPARISON RESULTS FOR GRIEWANK FUNCTION F_3 （F_3*=0）

Algorithm	Runtime(s)	Success(%)	Best	Mean	Worst	Standard deviation
IABCEO	0.029	**100**	1.95E-6	7.82E-6	9.96E-6	1.98E-6
ABCEO	0.045	**100**	1.22E-6	7.67E-6	1.00E-5	2.35E-6
ABC	0.043	**100**	1.91E-6	7.63E-6	9.96E-6	2.07E-6
PSOEO	0.018	**100**	**0**	**0**	**0**	**0**
PSO	**0.013**	**100**	**0**	**0**	**0**	**0**
PEO	3.288	**100**	8.30E-4	9.44E-4	9.99E-4	4.76E-5
GA	2.189	0	0.0055	0.061	0.1949	0.036

TABLE VI COMPARISON RESULTS FOR RASTRIGIN FUNCTION F_4 （F_4*=0）

Algorithm	Runtime(s)	Success(%)	Best	Mean	Worst	Standard deviation
IABCEO	0.314	**100**	2.94E-7	5.62E-6	9.98E-6	3.02E-6
ABCEO	0.163	98	3.90E-8	1.96E-3	0.098	0.014
ABC	0.179	**100**	1.62E-7	7.27E-6	1.27E-4	1.77E-5
PSOEO	0.018	**100**	**0**	**0**	**0**	**0**
PSO	**0.011**	**100**	**0**	**0**	**0**	**0**
PEO	16.04	0	1.544	2.242	2.727	0.273
GA	0.596	8	8.33E-4	0.02	0.533	0.075

TABLE VII COMPARISON RESULTS FOR ACKLEY FUNCTION F_5 （F_5*=0）

Algorithm	Runtime(s)	Success(%)	Best	Mean	Worst	Standard deviation
IABCEO	0.083	**100**	2.80E-7	7.63E-6	9.94E-6	2.22E-6
ABCEO	0.091	**100**	1.27E-6	8.24E-6	9.98E-6	1.71E-6
ABC	0.092	**100**	3.29E-6	7.79E-6	9.97E-6	1.95E-6
PSOEO	0.016	**100**	**-8.88E-16**	**-8.88E-16**	**-8.88E-16**	**9.96E-32**
PSO	**0.015**	**100**	**-8.88E-16**	**-8.88E-16**	**-8.88E-16**	**9.96E-32**
PEO	24.312	0	0.089	0.108	0.122	0.008
GA	0.969	0	0.022	0.052	0.117	0.019

TABLE VIII COMPARISON RESULTS FOR ROSENBROCK FUNCTION F_6（F_6*=0）

Algorithm	Runtime(s)	Success(%)	Best	Mean	Worst	Standard deviation
IABCEO	**3.88**	96	**2.09E-6**	7.43E-4	0.026	3.79E-3
ABCEO	7.75	2	8.03E-4	0.012	0.045	0.011
ABC	7.64	4	6.41E-4	0.014	0.057	0.013
PSOEO	247.7	**100**	9.31E-6	**8.17E-5**	**2.73E-4**	**5.93E-5**
PSO	8.46	2	2.25E-4	24.89	27.39	4.79
PEO	253.9	0	5.425	7.497	8.761	0.788
GA	10.214	0	20.970	29.542	51.995	6.385

REFERENCES

[1] D.Karaboga, "An Idea Based On Honey Bee Swarm for Numerical Optimization," Turkey: Erciyes University, 2005.

[2] B.Basturk and D. Karaboga, "Artificial Bee Colony (ABC) Optimization Algorithm for Solving Constrained Optimization Problems," Foundations of Fuzzy Logic and Soft Computing, vol. 4529, pp.789-798, 2007.

[3] X.H.Yan, Y.L.Zhu, and W.P.Zou, "A Hybrid Artificial Bee Colony Algorithm for Numerical Function Optimization," 2011 11th International Conference on Hybrid Intelligent Systems, pp. 127-132.

[4] L. P.Rangel, "Putative role of an ABC transporter in Fonsecaea pedrosoi multidrug resistance," International Journal of Antimicrobial Agents, vol. 40, pp.409-415, 2012.

[5] H. Potschka, "Targeting regulation of ABC efflux transporters in brain diseases: A novel therapeutic approach," Pharmacology&Therapeutics, vol. 125, pp.118-127, 2010.

[6] D.Yan, S. Z.Ahmad, and D. Yang, "Matthew effect, ABC analysis and project management of scale-free information systems," The Journal of Systems and Software, vol. 86, pp.247-254, 2013.

[7] J.X. Chen, "Peer-estimation for multiple criteria ABC inventory classification," Computer &Operations Research, vol.38, pp.1784-1791, 2011.

[8] S.Boettcher and A.G.Percus, "Extremal optimization: methods derived from co-evolution," in:Proceedings of the Genetic and Evolutionary Computation Conference, pp.825-832, 1999.

[9] S.Boettcher and A.G.Percus, "Nature's way of optimizing," Artificial Intelligence, vol.119, pp. 275-286,2000.

[10] P.Bak and K.Sneppen, "Punctuated equilibrium and criticality in a simple model of evolution," Physical Review Letters, vol. 71, pp.4083-4086, 1993.

[11] P.Bak, C.Tang, and K.Wiesenfeld, "Self-organized criticality," Physical Review Letters, vol. 59, vol.381-384, 1987.

[12] M.R.Chen, Y.Z.Lu, and G.K.Yang, "Multiobjective extremal optimization with applications to engineering design," Journal of Zhejiang University: SCIENCE A, vol. 8, pp.1905-1911,2007.

[13] M.R.Chen, Y.Z.Lu, and G.Yang, "Multiobjective optimization using population-based extremal optimization," Journal of Neural Computing and Applications, vol. 7, pp. 101-109, 2008.

[14] M.R.Chen and Y.Z.Lu, "A novel elitist multiobjective optimization algorithm: multiobjective extremal optimisation," European Journal of Operational Research, vol. 188, pp. 637-651, 2008.

[15] M.R.Chen, Y.Z.Lu, and G. Yang, "Population-based extremal optimization with adaptive Lévy mutation for constrained optimization," in: Proceedings of 2006 International Conference on Computational Intelligence and Security (CIS'06), pp. 258-261, 2006.

[16] Y.Z.Lu, M.R.Chen, and Y.W.Chen, "Studies on extremal optimization and its applications in solving real world optimization problems," in: Proceedings of the 2007 IEEE Symposium on Foundations of Computational Intelligence (FOCI 2007), Hawaii, USA ,pp. 162-168,2007.

[17] X.Li, J.P.Luo, M.R.Chen, and N.Wang, "An improved shuffled frog-leaping algorithm with extremal optimisation for continuous optimization," Information Sciences, vol. 192, pp. 143-151,2012.

[18] M.R.Chen, X.Li, X.Zhang, and Y.Z. Lu, "A novel particle swarm optimizer hybridized with extremal optimization," Applied Soft Computing, vol. 10, pp. 367-373, 2010.

[19] V.Azadehgan, N.Jafarian, and F.Jafarieh, "A Novel Hybrid Artificial Bee Colony with Extremal Optimization," in: Proceedings of 4th International Conference on Computer and Electrical Engineering (ICCEE 2011), pp. 45-49, 2011.

[20] D.Karaboga and B.Basturk, "On the performance of artificial bee colony (ABC) algorithm," Applied Soft Computing, vol. 8, pp.687-697, 2008.

[21] D.Karaboga and B.Akay, "A modified Artificial Bee Colony (ABC) algorithm for constrained optimization," Applied Soft Computing, vol. 11, pp.3021-3031, 2011.

[22] D.Karaboga and B.Akay, "A comparative study of Artificial Bee Colony algorithm," Applied Mathematics and Computation, vol. 214, pp.108-132, 2009.

[23] S.Boettcher and A.G.Percus, "Optimization with extremal dynamics," Physical Review Letters, vol. 86, pp.5211-5214, 2001.

[24] W.F.Gao, S.Y.Liu, and L.L.Huang, "Inspired Artificial Bee Colony Algorithm for Global Optimization Problems," ACTA ELECTRONICA SINICA, vol.12, pp.2396-240, 2012.