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

A Levy Flight-Based Hybrid Artificial Bee Colony Algorithm for Solving Numerical Optimization Problems

Hai Shan, Toshiyuki Yasuda, and Kazuhiro Ohkura

Abstract— **An artificial bee colony (ABC) algorithm is one of numerous swarm intelligence algorithms that employs the foraging behavior of honeybee colonies. To improve the convergence performance and search speed of finding the best solution using this approach, we propose a levy flight-based hybrid ABC algorithm in this paper. To evaluate the performance of the standard and proposed ABC algorithms, we implemented numerical optimization problems based on the IEEE Congress on Evolutionary Computation 2013 test suite. The proposed ABC algorithm demonstrated competitive performance on these optimization problems as compared to standard ABC, differential evolution, and particle swarm optimization algorithms with dimension sizes of 10, 30, and 50, respectively.**

I. INTRODUCTION

Optimization is an applied science that determines the best values of given parameters for a given problem. The aim of optimization is to obtain the relevant parameter values that enable an objective function to generate the minimum or maximum value [1]. An objective function must be designed to mathematically define the related problem. To solve more difficult optimization problems, the IEEE Congress on Evolutionary Computation(CEC) 2013 test suite [2] is an invaluable resource; the CEC 2013 test suite includes 28 benchmark functions and does not make use of exact equations. Effective and efficient optimization algorithms are required to solve more difficult problems, including complex real-world optimization problems.

In the past several years, swarm intelligence (SI), which is a discipline of artificial intelligence, has attracted the interest of many research scientists in related fields. Bonabeau [3] defined SI as "any attempt to design algorithms or distributed problem-solving devices inspired by the collective behavior of social insect colonies and other animal societies". Some SI algorithms, inspired by the social behaviors of birds, fish, bees, and insects, have been proposed to solve optimization problems; such algorithms include particle swarm optimization (PSO) [4], differential evolution (DE) [5], ant colony optimization [6], artificial bee colony (ABC) [7], the firefly algorithm (FA) [8], and cuckoo search (CS) [9]. A recent study showed that the ABC algorithm performs significantly better or at least comparably to other SI algorithms.

The ABC algorithm was introduced by Karaboga in 2005 as a technical report. Its performance was initially measured using benchmark optimization functions [10], [11]. The ABC algorithm has been applied to several fields in various ways, such as training neural networks [12], protein structure prediction [13], sensor deployment [14], Wireless Sensor Network [15], the redundancy allocation problem [16], and engineering design optimization [17].

The ABC algorithm is superior to other algorithms in terms of its simplicity, flexibility, and robustness. Karaboga [18] implemented comparison experiments and the result showed that the performance of ABC algorithm was better or similar to genetic algorithm [GA] [19], DE, and PSO algorithms. In addition, the ABC algorithm requires fewer training parameters, so combining it with other algorithms is easier. Given its flexibility, the ABC algorithm has been revised in many recent studies. For example, Alatas [20] proposed a chaotic ABC algorithm, in which many chaotic maps for parameters adapted from the original ABC algorithm were introduced to improve its convergence performance. Zhu and Kwong [21] proposed a gbest-guided ABC algorithm by incorporating the information of the global best solution into the solution search equation to improve the exploitation. In addition, Gao and Liu [22] proposed a modified ABC (MABC) algorithm that used a modified solution search equation with chaotic initialization; further MABC excluded the onlooker bees and scout bees phases.

However, along with the advantages of the improved versions of ABC, few disadvantages still exist. For example, ABC algorithms have low convergence speeds, low exploitation abilities, poor performance on initialization, and are also easily trapped in local optima. To overcome these disadvantages, we propose an improved ABC algorithm inspired by levy flight [23], [24], a self-adaptive mechanism, DE, PSO, and chaotic opposition-based learning (OBL) [25].

We implemented comparative experiments and set up parameters for our proposed ABC algorithm to demonstrate the efficacy of the algorithm; more specifically, we used the CEC 2013 test suite benchmark problems. Finally, we implemented comparative experiments using our proposed ABC and the standard ABC, DE, and PSO algorithms.

In addition to this introductory section, the remainder of this paper is organized as follows. The ABC algorithm is introduced in Section II. In Section III, we describe our proposed ABC algorithm. The experimental setup and results are discussed in Section IV, and we conclude our paper in Section V.

Hai Shan is with the Faculty of Engineering, Hiroshima University, Hiroshima, Japan (haishan@ohk.hiroshima-u.ac.jp)

Toshiyuki Yasuda is with the Faculty of Engineering, Hiroshima University, Hiroshima, Japan (yasu@hiroshima-u.ac.jp)

Kazuhiro Ohkura is with the Faculty of Engineering, Hiroshima University, Hiroshima, Japan (kohkura@hiroshima-u.ac.jp)

978-1-4799-1488-3/14/$31.00 ©2014 IEEE

II. THE ARTIFICIAL BEE COLONY (ABC) ALGORITHM

The ABC algorithm is a swarm-based meta-heuristic algorithm introduced by karaboga [7] that has successfully applied to numerical optimization problems [10], [11], [12]. In the ABC algorithm, the artificial bee colony comprises three kinds of bees: employed bees, onlooker bees, and scout bees. Employed bees search for food source sites by modifying the site in their memory, evaluating the nectar amount of each new source, and memorizing the more productive site through a selection process; these bees share information related to the quality of the food sources they exploit in the "dance area". Onlooker bees search for food sources based on the information coming from employed bees. As such, more beneficial sources have higher probability to be selected by onlookers. Further, onlooker bees choose food sources depending on the given information through probabilistic selection and modify these sources. When the food source is abandoned, a new food source is randomly selected by a scout bee to replace the abandoned source. The main steps of the algorithm are given below:

1. Initialize the population of solutions x_{ij} with

$$x_{ij} = x_{min\ j} + rand[0,1](x_{max\ j} - x_{min\ j}) \tag{1}$$

Here, $i \in 1, 2, ..., SN$ and $j \in 1, 2, ..., D$ are randomly selected indexes, SN is the number of food source, and D is the dimension size.

2. Evaluate the population.

3. Initialize cycle to 1.

4. Produce new solutions v_i for the employed bees by using (1), then evaluate them as follows

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

where ϕ_{ij} is uniformly distributed random number in the range [-1,1]; $i, k \in 1, 2, ..., SN$ are randomly selected indexes with k different from i and $j \in 1, 2, ..., D$ is a randomly selected index.

5. Apply the greedy selection process for the employed bees.

6. If the solution does not improve, add 1 to the trail, otherwise, set the trail to 0.

7. Calculate probability values p_i for the solutions using (3) as

$$p_i = \frac{fit_i}{\sum_{n=1}^{SN} fit_n} \tag{3}$$

where fit_i is the fitness value of solution i.

8. Produce new solutions for the onlooker bees from solutions x_i which is selected depending on p_i then evaluate them.

9. Apply the greedy selection process for the onlooker bees.

10. If the solution does not improve, add 1 to the trail, otherwise, set the trail to 0.

11. Determine the abandoned solution for the scout, if it exists, and replace it with a new random solution using (1).

12. Memorize the best solution achieved so far.

13. Add 1 to cycle.

14. Repeat until cycle reaches a predefined maximum cycle number (MCN).

III. PROPOSED ARTIFICIAL BEE COLONY (ABC) ALGORITHM

One can recognize two common aspects in population-based heuristic algorithms: exploration and exploitation [26]. Exploration is the ability to expand the search space, and exploitation is the ability to find optima around a good solution. Exploration and exploitation play key roles in SI algorithms; They coexist in the evolutionary process of algorithms such as PSO, DE, and ABC, but they contradict each other.

Population initialization is a crucial step in SI algorithms because it can affect convergence speeds and the quality of the final solution. If no information about the solution is available, then random initialization is the most commonly used method for generating an initial population. To initialize the population such that the search space information can be extracted to increase population diversity, we introduced a levy flight distribution. In the past, the flight behaviors of animals and insects that exhibit important properties of levy flight have been analyzed in various studies. This levy flight behavior has been applied to optimization and search algorithms, and reported results show its importance in the field of solution search algorithms [9], [27]. Recently, Yang proposed new meta-heuristic algorithms, such as FA and CS. Levy flight is a random walk in which the step lengths have a heavy-tailed probability distribution. Random step lengths drawn from a levy flight distribution [28], [29] are shown as

$$L(s) \sim |s|^{\ \ 1} \tag{4}$$

where $(0 < \leq 2)$ is an index and s is the step length.

Initialization for our proposed ABC using levy flight is calculated as

$$x_{ij}^{t+1} = x_{i\ j}^{t} + \alpha \oplus levy(\ ) \tag{5}$$

where $i \in 1, 2, ..., SN$ and $j \in 1, 2, ..., D$ are randomly selected indexes, t is the iteration number, α is uniformly distributed number selected from $U(0, 1)$. The product $\oplus$ means entry-wise multiplications.

The levy flight is calculated as

$$Levy(\ ) \sim 0.01 \left(\frac{u}{|v|}\right)^{\frac{1}{\ }} (x_j^t - x_i^t) \tag{6}$$

where u and v are derived from normal distributions as

$$u \sim N(0, \sigma_u^2) \quad v \sim N(0, \sigma_v^2) \tag{7}$$

$$\sigma_u = \left(\frac{\Gamma(1+\ )sin(\pi\ /2)}{\Gamma[(1+\ )/2]2^{(\ \ 1)2}}\right)^{1}, \quad \sigma_v = 1 \tag{8}$$

To achieve a good optimization performance with higher convergence speeds and without trapped in local optima, we introduced a self-adaptive mechanism to change the search range related with a cycle number, and then combined it with DE to improve the performance of the employed bees. In the

standard ABC algorithm, a random perturbation is added to the current solution to produce a new solution. This random perturbation is weighted by ϕ_{ij} selected from [-1,1] and is a uniformly distributed real random number in the standard ABC. A low value of ϕ_{ij} results in small steps to find the optimal value, therefore achieving convergence slowly. A high value of ϕ_{ij} accelerates the search, but reduces the exploration ability of the perturbation process.

Therefore, we used a self-adaptive mechanism to balance the exploration ability and the convergence speed of the algorithm for employed bees. The self- adaptive ABC approach has a very simple structure and is easy to implement. ϕ_{ij} is changed with the cycle number according to a random value called rand in the range [0,1] for food searching process of employed bee; ϕ_{ij} is determined as

$$\phi_{ij} = \begin{cases} -\mathrm{e}^{\ 3*cycle\ (25*MCN)}, 0 \leq rand \leq 0.5 \\ \mathrm{e}^{\ 3*cycle\ (25*MCN)}, 0.5 < rand \leq 1 \end{cases} \quad (9)$$

The DE algorithm has proved to be a simple yet powerful and efficient population-based algorithm for many global optimization problems. To further improve the performance of the DE algorithm, researchers have suggested different schemes of DE [30]. Like other evolutionary algorithms, DE also relies on an initial random population generation and then improves its population via mutation, crossover and selection processes. The DE equation is shown below as (10). The searching food source process in ABC algorithm is similar to the mutation process of DE. And in DE, the best solutions in the current population are very useful for improving convergence performance. One scheme of the mutations of DE, "DE/best/1" can effectively maintain population diversity. We therefore combined the "DE/best/1" mutation strategy with the food searching process of the ABC algorithm to produce a new search equation (shown below as (11)) and improved the convergence ability.

$$v_{i\ G} = x_{best\ G} + F * (x_{r_1^i\ G} - x_{r_2^i\ G}) \quad (10)$$

where $i \in 1, 2, ..., NP$ is a randomly selected index, NP is the population number, G is the generation number, $v_{i\ G}$ is the donor vector, and r_1^i and r_2^i are random numbers chosen from the range [1, NP].

$$v_{ij} = x_{best\ j} + \phi_{ij}(x_{ij} - x_{kj}) \quad (11)$$

where $i, k \in 1, 2, ..., SN$ are randomly selected indexes with k different from i; $j \in 1, 2, ..., D$ is a randomly selected index and ϕ_{ij} is the parameter given in equation (9).

To the best of our knowledge, the search ability of the ABC algorithm is good at exploration, but poor in terms of exploitation. Specifically, we can view the relationship of employed bees and onlooker bees as focused on exploration and exploitation, respectively. Employed bees explore new food sources and send information to onlooker bees, and onlooker bees exploit the food sources explored by employed bees.

In the standard ABC algorithm, much time is required to find the food source due to poor exploitation abilities and lower convergence speeds. To improve the exploitation ability of the algorithm, we incorporated PSO into the ABC algorithm. PSO is based on the simulation of simplified social animal behaviors. The equation governing PSO is shown as (12) below. We modified the onlooker bee search solution by taking advantage of the search mechanism of PSO; our modified search equation for onlooker bees is shown as (13) below.

$$v_{i\ d} = \omega v_{i\ d} + c_1 r_1 (p_{i\ d} - x_{i\ d}) + c_2 r_2 (p_{g\ d} - x_{i\ d}) \quad (12)$$

Here, $d \in 1, 2, ..., D$, $i \in 1, 2, ..., M$, M is the total number of particles in the swarm, ω is the inertia weight, r_1, r_2 are random numbers in the range [0,1], c_1, c_2 are acceleration coefficients, $p_{i\ d}$ is the personal best and $p_{g\ d}$ is the global best.

$$v_{ij} = x_{ij} + \varphi_{ij}(x_{ij} - x_{kj}) + \psi_{ij}(x_{best\ j} - x_{ij}) \quad (13)$$

where $i, k \in 1, 2, ..., SN$ are randomly selected indexes with k different from i; $j \in 1, 2, ..., SN$ is a randomly selected index; $x_{best\ j}$ is the jth element of the best dominant solution, and $\varphi_{ij} \in [-1, 1]$ and $\psi_{ij} \in [0, 1.5]$ are uniformly distributed random numbers.

The concept of OBL was introduced by Tizhoosh [25] and has been applied to accelerate reinforcement learning and back-propagation learning in neural networks [31]. The main idea behind OBL is to simultaneously consider an estimate and its corresponding opposite estimate to achieve better approximations for candidate solutions. According to [32], OBL was introduced to DE and improved the convergence performance. Therefore, to accelerate convergence speed, we introduced chaotic OBL initialization to generate the population of scout bees. Here, a sinusoidal iterator is selected, and its equation is defined as

$$ch_{kj} = sin(\pi ch_{k\ \ 1\ j}) \quad (14)$$

where $ch_k \in (0, 1)$, $k = 1, 2, ..., Max$, $j = 1, 2, ..., D$

The initialization population for scout bees is shown as

$$x_{ij} = x_{min\ j} + ch_{kj}(x_{max\ j} - x_{min\ j}) \quad (15)$$

and the chaotic OBL equation is shown as

$$ox_{ij} = x_{min\ j} + x_{max\ j} - x_{ij} \quad (16)$$

where ox indicates the opposition-based population.

We selected SN individuals from the set $\{X(SN) \cup OX(SN)\}$ as the initial scout bees population.

For our proposed ABC algorithm, we specifically modified steps 1, 4, 8, and 11 of standard ABC algorithm. The modification (5) substituted for step 1, modifications (9) and (11) substituted for step 4, modification (13) substituted for step 8, and modifications (14), (15), and (16) substituted for step 11, respectively.

IV. Experiments

A. Experimental setup

The CEC 2013 test suite extends its predecessor CEC 2005 test suite. In the CEC 2013 test suite, the previously

proposed composition functions are improved, and additional test functions are included. There are 28 numerical test functions that are minimization problems categorized into the following three groups: unimodal functions (F1-F5), multimodal functions (F6-F20), and composition functions (F21-F28). The functions and their names are summarized in Table I. The detailed description of the CEC 2013 test suite is available in [2].

All test functions are minimization problems defined as follows:

To minimize $f(x), x = [x_1, x_2, ...x_D]^T$

where D is the number of dimensions.

Given $o = [o_1, o_2, ...o_D]^T$, the shifted global optimum distributed randomly in the range $[-80, 80]^D$, all test functions are shifted to o and are scalable. For convenience, the same search ranges $[-100, 100]^D$ is defined for all test functions.

TABLE I
CEC 2013 TEST FUNCTIONS

Function Number	Function Name
F1	Sphere
F2	Rotated High Conditioned Elliptic
F3	Rotated Bent Cigar
F4	Rotated Discus
F5	Different Powers
F6	Rotated Rosenbrock
F7	Rotated Schaffers F7
F8	Rotated Ackley
F9	Rotated Weierstrass
F10	Rotated Griewank
F11	Rastrigin
F12	Rotated Rastrigin
F13	Non-Continuous Rotated Rastrigin
F14	Schwefel
F15	Rotated Schwefel
F16	Rotated Katsuura
F17	Lunacek Bi_Rastrigin
F18	Rotated Lunacek Bi_Rastrigin
F19	Expanded Griewank's Plus Rosenbrock
F20	Expanded Scaffer F6
F21	Composition Function 1 (n=5,Rotated)
F22	Composition Function 2 (n=3,Unrotated)
F23	Composition Function 3 (n=3,Rotated)
F24	Composition Function 4 (n=3,Rotated)
F25	Composition Function 5 (n=3,Rotated)
F26	Composition Function 6 (n=5,Rotated)
F27	Composition Function 7 (n=5,Rotated)
F28	Composition Function 8 (n=5,Rotated)

In this paper, we evaluated both the standard ABC algorithm and our proposed ABC algorithm for all 28 test functions defined in the CEC 2013 test suite with parameters selected by comparing experiments at three dimension sizes, i.e., 10, 30, and 50. The 28 test functions were executed 51 times with respect to each test function at each problem dimension size. The algorithms were terminated when the MCN was reached for function evaluations or when the error value was smaller than 10^{-8}. In our experiments, we set maximum evaluation sizes to 100,000, 300,000, and 500,000 for problem dimension sizes of 10, 30, and 50, respectively. We also performed the Wilcoxon signed rank test with significance level of 0.05, and conducted comparative experiments for the standard and proposed ABC algorithms at first and then broadened our experiments to include the DE [33], and PSO [34] algorithms.

We used the C language for our experiments on a Linux system with an Intel Core i3 CPU 540 @3.07GHz * 4 with 64-bit processing.

B. Experimental results

TABLE II
PARAMETER ADJUSTMENT EXPERIMENTS

Limit/NP	20	50	100	150	200	300
50	-/-/-	-/-/-	-/-/-	-/-/-	-/-/-	-/-/-
100	-/-/-	-/-/-	10/30/-	10/-/50	-/-/-	-/-/-
150	-/-/-	10/30/-	10/30/50	10/-/50	10/-/50	-/-/-
250	-/-/-	10/30/-	10/30/-	-/30/50	-/30/50	-/-/-
400	-/-/-	10/-/-	-/30/50	-/-/50	-/-/-	-/-/-

Table II shows our parameter adjustment experiment results with NP representing population size. In this table, "-/-/-" indicates competitive performance of our algorithm on the three dimension sizes 10, 30, and 50. "-" indicates that its performance was significantly worse than others on that dimension size.

From Table II, we observe that the ABC is not very sensitive to the choice of parameters given much lower or much higher population sizes and limits. We therefore selected a limit of 150 and population size of 100.

After implementing comparative experiments on our proposed ABC algorithm, the standard ABC algorithm, the DE algorithm, and the PSO algorithm, we listed, in Table III, the number of better, similar, and worse performances of mean values of these algorithms on 28 test functions.

Tables IV, V, and VI illustrate the function error of mean values of our proposed ABC algorithm, the standard ABC algorithm, the DE algorithm, and the PSO algorithm for 100,000, 300,000, and 500,000 evaluations of dimension size of 10, 30, and 50, respectively. The symbols of "+", "-", "=", and "≈" indicate better, worse, equal, or similar performance of these standard ABC, DE and PSO algorithms compared to our proposed ABC algorithm.

TABLE III
COMPARISON PERFORMANCE OF MEAN VALUES OF OUR PROPOSED ABC ALGORITHM TO STANDARD ABC, DE, AND PSO ALGORITHMS

Proposed ABC VS (10D)	ABC	DE	PSO
Better	9	19	23
Similar	16	3	5
Worse	3	6	0
Proposed ABC VS (30D)	ABC	DE	PSO
Better	10	13	19
Similar	13	8	7
Worse	5	7	2
Proposed ABC VS (50D)	ABC	DE	PSO
Better	12	14	17
Similar	11	5	7
Worse	5	9	4

When the dimension sizes increased, the number of functions in our proposed ABC algorithm with better and

similar performance decreased as compared to the other algorithms for the function error of mean values. From these comparative tables, we conclude that the better or similar performance of our proposed ABC algorithm is very competitive as compared to the other algorithms, especially in comparison with PSO algorithm.

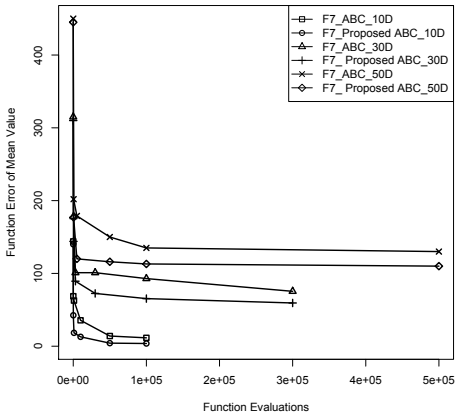

Fig. 1. Comparative convergence for function F7 on 10D, 30D, and 50D

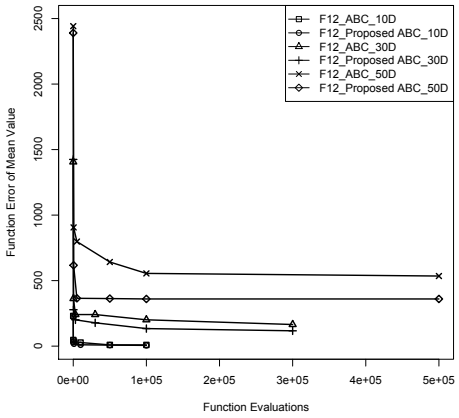

Fig. 2. Comparative convergence for function F12 on 10D, 30D, and 50D

Figures 1–8 illustrate the convergence performance for function error of mean values on both the standard ABC algorithm and our proposed ABC algorithm with increasing function evaluations on dimension sizes of 10, 30, and 50. According to Figures 1–4, the performance of our proposed ABC algorithm was much better than that of the standard ABC algorithm; note that the same results were achieved by both algorithms for functions F2, F3, F4, and F6.

According to Figures 5–6, the performance of our proposed ABC algorithm was better than that of the standard ABC algorithm, but was not obvious; further, note that the same results were achieved by both algorithms for functions F14, F19, and F22.

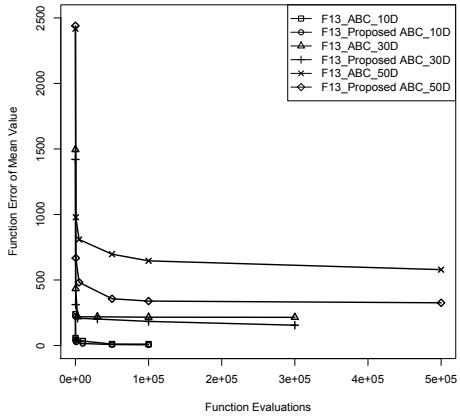

Fig. 3. Comparative convergence for function F13 on 10D, 30D, and 50D

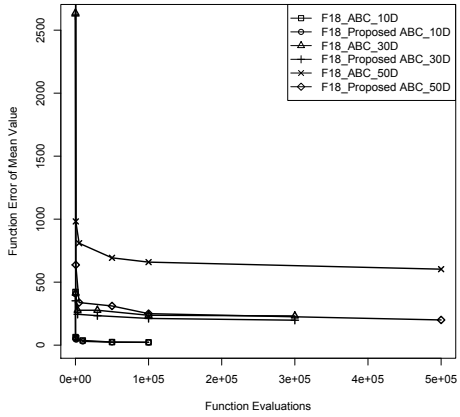

Fig. 4. Comparative convergence for function F18 on 10D, 30D, and 50D

Very similar performance was achieved for both algorithms as shown in Figure 7, and the same results were achieved for functions F20, F26, and F27. Convergence performance was the same for both algorithms, as shown in Figures 8. Further, the same were achieved by both algorithms for functions F1, F5, and F11, however, these three functions have the best performance because all function error values reached zero. For functions F10, F15, F16, F17, F21, F23, and F28, the performance of our proposed ABC algorithm was not better than that of the standard ABC algorithm on the same dimensions. More specifically, the performance of our proposed ABC algorithm was not better than that of standard ABC algorithm for functions F15 and F21 with 10, 30, and 50 dimensions; the performance of our proposed ABC algorithm was not better than that of the standard ABC algorithm for functions F10, F16, and F17 with 30 and 50 dimensions; and the performance of our proposed ABC algorithm was not better than that of the standard ABC algorithm for function F28 with 10 and 30 dimensions.

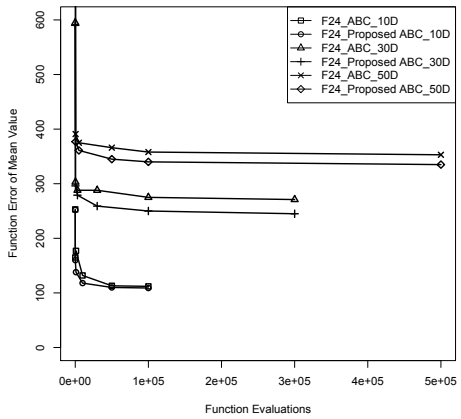

Fig. 5. Comparative convergence for function F24 on 10D, 30D, and 50D

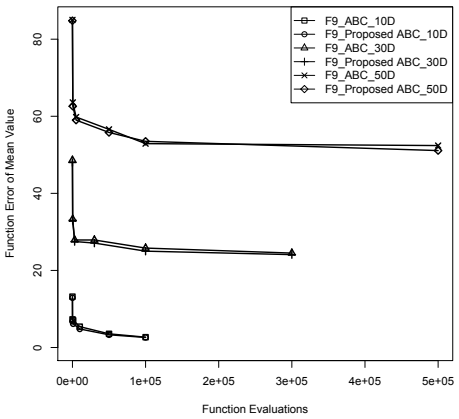

Fig. 7. Comparative convergence for function F9 on 10D, 30D, and 50D

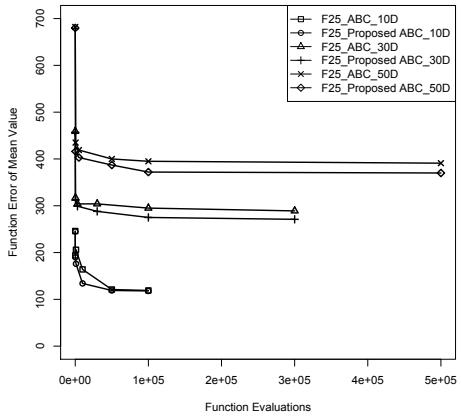

Fig. 6. Comparative convergence for function F25 on 10D, 30D, and 50D

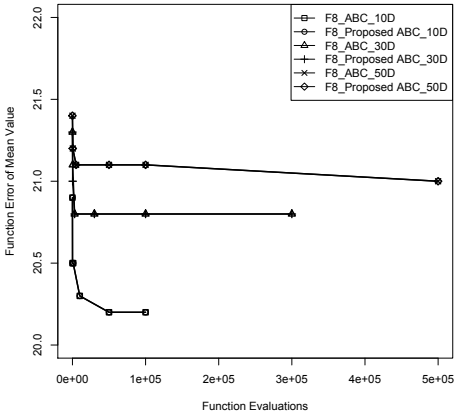

Fig. 8. Comparative convergence for function F8 on 10D, 30D, and 50D

According to the above tables and figures, our proposed ABC algorithm and the standard ABC algorithm were not very effective at solving unimodal functions, especially functions F2, F3, and F4. For all evaluations, functions F1, F5, and F11 achieved the best performance because the error values of the function-target pair reached zero. Moreover, function F8 had the same performance for all evaluation stages.

V. CONCLUSIONS

In this paper, we implemented comparative experiments of our proposed ABC and the standard ABC algorithm using benchmark problems from the CEC 2013 test suite. We introduced levy flight initialization, chaotic OBL scout initialization, and incorporated DE and PSO into the standard ABC algorithm to form our proposed ABC algorithm. We selected the best parameter settings through a number of initial comparative experiments and then evaluated the performance of both algorithms with dimension sizes of 10, 30, and 50. In addition, we implemented comparative experiments of our proposed ABC algorithm with that of the DE and PSO algorithms. From our results, we conclude that the performance of our proposed ABC algorithm equals or exceeds that of the standard ABC, DE, and PSO algorithms.

REFERENCES

[1] P. Civicioglu and E. Besdok. "A conceptual comparison of the cuckoo-search, particle swarm optimization, differential evolution and artificial bee colony algorithms." *Artificial Intelligence Review,* vol.39(4), pp. 315-346, 2013.

[2] J. J. Liang, B. Y. Qu, P. N. Suganthan, and A. G. Hernández-Díaz, "Problem definitions and evaluation criteria for the CEC 2013 special session on real-parameter optimization," *Computational Intell. Lab., Zhengzhou University, Zhengzhou, China, Tech. Rep. 201212, Nanyang Technological Univ., Singapore, Tech. Rep.* 2013.

[3] E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm Intelligence: From Natural to Artificial Systems.* New York: Oxford University Press, 1999, pp. 1-21.

[4] J. Kennedy and R. Eberhart. "Particle swarm optimization," *IEEE International Conference on Neural Networks,* pp. 1942-1948, 1995.

[5] R. Storn and K. Price. "Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces." *Journal of Global Optimization,* vol. 11(4), pp. 341-359, 1997.

[6] M. Dorigo, M. Birattari, and T. Stützle. "Ant colony optimization: artificial ants as a computational intelligence technique." *IEEE Computational Intelligence Magazine,* vol. 1(4), 2006.

TABLE IV

FUNCTION ERROR OF MEAN VALUES FOR OUR PROPOSED ABC ALGORITHM, THE STANDARD ABC ALGORITHM, THE DE AND PSO ALGORITHMS IN 10D

F./Eva.	Proposed ABC	ABC	DE	PSO
F1	0.00e+00	0.00e+00(=)	0.00e+00(=)	0.00e+00(=)
F2	2.20e+04	4.91e+05(-)	2.42e+03(+)	2.83e+05(-)
F3	7.27e+03	1.42e+05(-)	1.41e+00(+)	6.19e+05(-)
F4	3.50e+03	3.57e+03(≈)	2.71e+01(+)	6.76e+03(-)
F5	0.00e+00	0.00e+00(=)	0.00e+00(=)	0.00e+00(=)
F6	1.62e-03	3.77e-02(≈)	3.29e+00(-)	4.03e+00(-)
F7	3.72e+00	1.15e+01(-)	1.44e-03(+)	1.01e+01(-)
F8	2.02e+01	2.02e+01(=)	2.04e+01(≈)	2.03e+01(≈)
F9	2.61e+00	2.68e+00(≈)	1.14e+00(+)	3.47e+00(-)
F10	4.96e-01	4.68e-01(≈)	4.92e-02(+)	4.58e-01(≈)
F11	0.00e+00	0.00e+00(=)	1.14e+00(-)	4.83e+00(-)
F12	5.67e+00	9.23e+00(-)	8.24e+00(-)	1.09e+01(-)
F13	5.08e+00	1.03e+01(-)	1.21e+01(-)	1.83e+01(-)
F14	0.00e+00	0.00e+00(=)	2.20e+02(-)	2.91e+02(-)
F15	3.07e+02	2.71e+02(+)	1.13e+03(-)	5.93e+02(-)
F16	5.10e-01	3.29e-01(≈)	1.01e+00(-)	6.18e-01(≈)
F17	2.02e+00	2.13e+00(≈)	1.78e+01(-)	1.55e+01(-)
F18	2.23e+01	2.25e+01(≈)	3.16e+01(-)	2.64e+01(-)
F19	1.00e-06	1.00e-06(=)	1.07e+00(-)	6.56e-01(-)
F20	1.92e+00	2.13e+00(-)	2.36e+00(-)	2.73e+00(-)
F21	8.00e+01	1.43e+01(+)	3.73e+02(-)	2.59e+02(-)
F22	0.00e+00	1.73e-03(≈)	2.23e+02(-)	2.65e+02(-)
F23	3.50e+02	4.53e+02(-)	9.77e+02(-)	6.99e+02(-)
F24	1.09e+02	1.12e+02(≈)	2.02e+02(-)	2.00e+02(-)
F25	1.18e+02	1.19e+02(≈)	2.02e+02(-)	2.03e+02(-)
F26	1.07e+02	9.39e+01(+)	1.67e+02(-)	1.18e+02(-)
F27	3.20e+02	3.36e+02(-)	3.37e+02(-)	4.09e+02(-)
F28	4.80e+01	5.49e+01(-)	2.92e+02(-)	2.61e+02(-)

TABLE V

FUNCTION ERROR OF MEAN VALUES FOR OUR PROPOSED ABC ALGORITHM, THE STANDARD ABC ALGORITHM, THE DE AND PSO ALGORITHMS IN 30D

F./Eva.	Proposed ABC	ABC	DE	PSO
F1	0.00e+00	0.00e+00(=)	0.00e+00(=)	0.00e+00(=)
F2	4.83e+06	4.89e+06(≈)	1.54e+05(+)	5.02e+06(-)
F3	1.73e+07	5.07e+07(-)	3.65e+06(+)	2.85e+08(-)
F4	4.36e+04	4.73e+04(≈)	4.62e+02(+)	1.92e+04(+)
F5	0.00e+00	0.00e+00(=)	3.09e-05(≈)	0.00e+00(=)
F6	1.11e-02	3.29e+00(-)	1.98e+01(-)	2.38e+01(-)
F7	5.94e+01	7.54e+01(-)	1.58e+00(+)	6.45e+01(-)
F8	2.08e+01	2.08e+01(=)	2.09e+01(≈)	2.09e+01(≈)
F9	2.41e+01	2.45e+01(≈)	9.17e+00(+)	2.70e+01(-)
F10	2.18e-01	1.10e-01(≈)	7.62e-02(≈)	1.55e+00(-)
F11	0.00e+00	0.00e+00(=)	1.42e+01(-)	5.65e+01(-)
F12	1.17e+02	1.65e+02(-)	1.14e+02(≈)	9.05e+01(+)
F13	1.55e+02	2.15e+02(-)	1.53e+02(≈)	1.48e+02(≈)
F14	0.00e+00	1.65e-01(-)	5.72e+02(-)	1.95e+03(-)
F15	3.48e+03	2.91e+03(+)	7.01e+03(-)	4.00e+03(-)
F16	1.33e+00	9.26e-01(+)	2.45e+00(-)	1.70e+00(-)
F17	3.04e+01	2.92e+01(≈)	5.62e+01(-)	1.01e+02(-)
F18	1.98e+02	2.32e+02(-)	1.99e+02(≈)	1.88e+02(≈)
F19	9.53e-02	1.36e-01(≈)	3.93e+00(-)	6.29e+00(-)
F20	1.23e+01	1.25e+01(≈)	1.19e+01(≈)	1.18e+01(≈)
F21	2.10e+02	1.35e+02(+)	3.07e+02(-)	2.10e+02(=)
F22	4.00e+01	1.28e+01(+)	4.44e+02(-)	2.19e+03(-)
F23	4.15e+03	3.61e+03(+)	7.11e+03(-)	4.53e+03(-)
F24	2.45e+02	2.71e+02(-)	2.17e+02(+)	2.74e+02(-)
F25	2.71e+02	2.89e+02(-)	2.48e+02(+)	2.87e+02(-)
F26	2.00e+02	2.00e+02(=)	2.37e+02(-)	2.29e+02(-)
F27	4.00e+02	4.00e+02(=)	4.95e+02(-)	1.02e+03(-)
F28	1.08e+02	1.23e+02(-)	3.00e+02(-)	2.96e+02(-)

[7] D. Karaboga. *An idea based on honey bee swarm for numerical optimization,* Erciyes Univ. Press, Erciyes, Tech. Rep. TR06, 2005.

[8] X. S. Yang. "Firefly algorithm, stochastic test functions and design optimization." *International Journal of Bio-Inspired Computing,* vol. 2 (2), pp. 78-84, 2010.

[9] X. S. Yang and S. Deb. "Cuckoo search via Lévy flights," in *Proc. of World Congress on Nature & Biologically Inspired Computing,* 2009, pp. 210-214.

[10] D. Karaboga and B. Basturk. "A powerful and efficient algorithm for numerical function optimization: Artificial bee colony(ABC) algorithm." *Journal of Global Optimization,* vol. 39(3), pp. 459-471, 2007.

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

[12] D. Karaboga and B. Akay. "An artificial bee colony (abc) algorithm on training artificial neural networks," in *15th IEEE Signal Processing and Communications Applications,* 2007, pp.1-4.

[13] C. V. Benítez and H. Lopes. "Artificial bee colony algorithm approaches for protein structure prediction using the 3dhp-sc model." *Intelligent Distributed Computing IV,* pp. 255-264, 2010.

[14] S. K. Udgata, S. L. Sabat, and S. Mini. "Sensor deployment in irregular terrain using artificial bee colony algorithm," in *IEEE Congress on Nature & Biologically Inspired Computing,* 2009, pp. 1309-1314.

[15] S. Okdem, D. Karaboga, and C. Ozturk. "An application of wireless sensor network routing based on artificial bee colony algorithm," in *IEEE Congress on Evolutionary Computation,* 2011, pp. 326-330.

[16] W. C. Yeh and T. J. Hsieh. "Solving reliability redundancy allocation problems using an artificial bee colony algorithm." *Computers & Operations Research,* vol. 38, pp. 1465-1473, 2010.

[17] B. Akay and D. Karaboga. "Artificial bee colony algorithm for large-scale problems and engineering design optimization." *Journal of Intelligent Manufacturing,* vol. 23(4), pp. 1001-1014, 2010.

[18] D. Karaboga and B. Akay. "A comparative study of artificial Bee colony algorithm." *Applied Mathematics and Computation,* vol. 214(1), pp. 108-132, 2009.

[19] D. Whitley. "A genetic Algorithm tutorial." *Statistics and Computing,* vol. 4, pp. 65-85, 1994.

[20] B. Alatas. "Chaotic bee colony algorithms for global numerical optimization." *Expert Systems with Applications,* vol. 37, pp. 5682-5687, 2010.

[21] G. Zhu and S. Kwong. "Gbest-guided artificial bee colony algorithm for numerical function optimization." *Applied Mathematics and Computation,* vol. 217, pp. 3166-3173, 2010.

[22] W. Gao and S. Liu. "A modified artificial bee colony algorithm." *Computers & Operations Research,* vol. 39, pp. 687-697, 2012.

[23] C. T. Brown, L. S. Liebovitch, and R. Glendon. "Lévy flights in Dobe Ju /' hoansi foraging patterns." *Human Ecology,* vol. 35, pp. 129-138, 2007.

[24] I. Pavlyukevich. "Lévy flights, non-local search and simulated annealing." *Computational Physics,* vol. 226, pp. 1830-1844, 2007.

[25] H. R. Tizhoosh. "Opposition-based learning: a new scheme for machine intelligence," in *Proc. of International Conference on Comp. Intell. for Modeling, Control and Automation,* 2005, col. 1, pp.695-701.

[26] E. Rashedi, H. Nezamabadi-pour, and S. Saryazdi. "GSA: a gravitational search algorithm." *Information Science,* vol. 179(13), pp. 2232-2248, 2009.

[27] X. S. Yang and S. Deb. "Multiobjective cuckoo search for design optimization." *Computers & Operations Research,* vol. 40(6), pp. 1616-1624, 2013.

[28] X. S. Yang. *Engineering optimization: an introduction with metaheuristic applications.* New Jersey: John Wiley and Sons, 2010, pp. 153-161.

[29] G. M. Viswanathan, V. Afanasyev, S. V. Buldyrev et al. "Lévy flight search patterns of wandering albatrosses." *Nature,* vol. 381, 1996.

[30] D. Swagatam and P. N. Suganthan. "Differential evolution: a survey of the state-of-the-art." *IEEE Trans. on Evol. Computation,* vol. 15(1), 2011.

TABLE VI

FUNCTION ERROR OF MEAN VALUES FOR OUR PROPOSED ABC ALGORITHM, THE STANDARD ABC ALGORITHM, THE DE AND PSO ALGORITHMS IN 50D

F./Eva.	Proposed ABC	ABC	DE	PSO
F1	0.00e+00	0.00e+00(=)	0.00e+00(=)	0.00e+00(=)
F2	1.51e+07	9.86e+06(+)	4.91e+05(+)	1.13e+07(+)
F3	2.99e+07	3.53e+07(-)	1.97e+07(+)	3.25e+09(-)
F4	9.85e+04	1.04e+05(-)	1.83e+03(+)	2.96e+04(+)
F5	0.00e+00	0.00e+00(=)	1.34e-04(≈)	0.00e+00(=)
F6	1.17e-01	2.98e+01(-)	4.46e+01(-)	4.65e+01(-)
F7	1.10e+02	1.30e+02(-)	1.24e+01(+)	1.17e+02(≈)
F8	2.10e+01	2.10e+01(=)	2.11e+01(≈)	2.11e+01(≈)
F9	5.11e+01	5.24e+01(≈)	2.94e+01(+)	5.57e+01(-)
F10	2.26e-01	1.16e-01(≈)	1.37e-01(≈)	1.01e+01(-)
F11	0.00e+00	0.00e+00(=)	3.64e+02(-)	1.20e+02(-)
F12	3.60e+02	5.35e+02(-)	2.94e+02(+)	2.24e+02(+)
F13	3.26e+02	5.79e+02(-)	3.50e+01(+)	3.54e+02(-)
F14	0.00e+00	1.42e+00(-)	1.25e+03(-)	3.69e+03(-)
F15	8.29e+03	6.64e+03(+)	1.38e+04(-)	8.85e+03(-)
F16	1.90e+00	1.47e+00(+)	3.25e+00(-)	2.49e+00(-)
F17	5.08e+01	4.89e+01(≈)	9.52e+01(-)	2.32e+02(-)
F18	2.00e+02	6.03e+02(-)	3.95e+02(-)	4.03e+02(-)
F19	4.38e-01	5.03e-01(≈)	5.89e+00(-)	1.83e+01(-)
F20	2.27e+01	2.35e+01(≈)	2.18e+01(≈)	2.15e+01(≈)
F21	4.55e+02	2.06e+02(+)	7.05e+02(-)	3.13e+02(+)
F22	0.00e+00	1.17e+01(-)	1.23e+03(-)	4.54e+03(-)
F23	8.32e+03	7.88e+03(+)	1.36e+04(-)	9.70e+03(-)
F24	3.35e+02	3.53e+02(-)	2.50e+02(+)	3.48e+02(-)
F25	3.70e+02	3.91e+02(-)	2.92e+02(+)	3.74e+02(≈)
F26	2.00e+02	2.01e+02(≈)	3.21e+02(-)	3.18e+02(-)
F27	3.83e+02	4.01e+02(-)	8.34e+02(-)	1.75e+03 (-)
F28	4.00e+02	4.00e+02(=)	5.16e+02(-)	4.00e+02(=)

[31] M. Ventresca and H. R. Tizhoosh. "Improving the convergence of back-propagation by opposite transfer functions," in *Proc. of IEEE World Congress Computation Intell.,* 2006, pp. 9527-9534.

[32] S. Rahnamayan, R. T. Hamid, and M. A. S. Magdy. "Opposition-based differential evolution." in *IEEE Trans. of Evolution Computation,* vol. 12(1), pp. 64-79, 2008.

[33] A. K. Qin and X. Li. "Differential evolution on the CEC-2013 single-objective continuous optimization testbed," in *Proc. IEEE Congress on Evolutionary Computation, Cancun, Mexico,* 2013, pp. 1099-1106.

[34] C. Stephen, M. James, and B. R. Antonio. "Standard Particle Swarm Optimization on the CEC2013 Real-Parameter Optimization Benchmark Functions," School of Information Technology, York University, Tech. Rep., 2013.