An Effective Hybrid Butterfly Optimization Algorithm with Artificial Bee Colony for Numerical Optimization

Sankalap Arora[1], Satvir Singh[2]

[1]Research Scholar, I.K. Gujral Punjab Technical University, Jalandhar, Punjab, India
[2]Shaheed Bhagat Singh State Technical Campus, Ferozpur, Punjab,India

Abstract — **In this paper, a new hybrid optimization algorithm which combines the standard Butterfly Optimization Algorithm (BOA) with Artificial Bee Colony (ABC) algorithm is proposed. The proposed algorithm used the advantages of both the algorithms in order to balance the trade-off between exploration and exploitation. Experiments have been conducted on the proposed algorithm using ten benchmark problems having a broad range of dimensions and diverse complexities. The simulation results demonstrate that the convergence speed and accuracy of the proposed algorithm in finding optimal solutions is significantly better than BOA and ABC.**

Keywords — **Butterfly Optimization Algorithm, Artificial Bee Colony, Hybrid algorithm, Optimization.**

I. Introduction

Nature-inspired metaheuristic algorithms have received much attention by researchers in the past as they have the ability to solve real world complex problems. These problems require optimal solution in less computational time [1]. Their potential has recognized themselves as numerical optimization techniques in various real world complex problems [2]. These algorithms find their source of inspiration in nature. Various algorithms have been proposed in the past like Particle Swarm Optimization (PSO) [3], Firefly Algorithm (FA) [4], Cuckoo Search (CS) [5], Ant Colony Optimization (ACO) [6] and many more [7,8].

Another example is Butterfly Optimization Algorithm (BOA) which is inspired by the food foraging behavior of the butterflies [9]. The underlying mechanism of BOA is to mimic the food searching abilities of biological butterflies. It has demonstrated better results over other population based algorithm [9]. Recently, chaos is introduced in BOA so as to increase the global search mobility for global optimization problems [10]. BOA is a powerful algorithm in exploitation (i.e., local search) but at times it may trap into some local optima so that it cannot perform global search well.

On the other hand, Artificial Bee Colony (ABC) which was based on the intelligent behavior of the honey bee swarm was proposed by karaboga in 2007 [11]. In the past, it has been applied to various real world problems which demonstrate its superiority over many other algorithms [12].

The main strength of ABC lies in its strategy which allows the solutions to move towards those solutions which have better fitness probability. The aim of the paper is to propose a hybrid algorithm, namely BOA/ABC which have strengths of both the algorithms viz. BOA and ABC. The proposed BOA/ABC algorithm will have advantages of both the algorithms which will enable the algorithm to demonstrate fast convergence and avoid local optima trap problem. The remainder of this paper is organized as follows. Previous work is described in Section II. The conventional BOA and ABC are reviewed in Sections III and IV, respectively. Section V describes the proposed hybrid algorithm BOA/ABC. Experimental results demonstrating the performance of BOA/ABC in comparison with the conventional BOA and ABC over a subset of ten numerical optimization problems are presented in Section VI and conclusions are drawn in Section VII.

II. Related Work

In the past, various researchers have successfully hybridized optimization algorithms in order to increase performance of algorithms in terms of solution quality. One of the finest example is Differential Evolution (DE) hybridized with Biogeography Based Optimization (BBO) algorithm [13]. DE/BBO utilizes the exploration capability of DE along with the exploitation capability of BBO effectively and efficiently in order to generate the promising solutions. In [14] a hybrid algorithm Harmony Search (HS) and FA is proposed which uses the exploration of HS and the exploitation of FA, efficiently, so that HS/FA shows faster convergence speed than individual algorithms i.e. HS and FA. In [15] PSO is hybridized with Genetic Algorithm (GA) in order to utilize the unique advantages of both the algorithms. The hybridized algorithm efficiently uses the operations of PSO and GA such as single or multiple crossover, mutation, and the PSO formula. Further the selection of these operators is based on fuzzy probability. In [16] a hybrid optimization method named hybrid evolutionary FA is proposed. The algorithm combines the classical FA with the evolutionary operations of DE algorithm aiming to improve the searching accurateness and information sharing among the solutions in the search space.

In [17] a hybrid algorithm combining two swarm intelligence algorithms i.e., ABC and PSO is presented. The proposed method is component based technique in which PSO is augmented with ABC in order to improve the overall efficiency of the algorithm. Another hybridized algorithm is proposed integrating ABC and DE [18]. In this hybridized algorithm the basic drawback of DE i.e. it requires a relatively large population size to avoid premature convergence is overcome by the use of ABC which have proven to demonstrate excellent abilities of global searching. Furthermore, there are various algorithms which have been proposed in the past in order to accelerate the performance of algorithms [19-23].

III. Butterfly Optimization Algorithm

Butterfly Optimization Algorithm (BOA) is a recently developed nature inspired optimization algorithm by Arora [9, 24]. It finds its source of inspiration in the food foraging behavior of butterflies. Biologically, the butterflies are very efficient in finding their food and the same food searching mechanism is used in the algorithm. In BOA, the butterflies are used as search agents in order to perform optimization. Nature has equipped the butterflies with sense receptors

which allow them to sense the fragrance of food and consequently they move towards the particular direction. These sense receptors, also called chemoreceptors, are scattered over the butterfly's body parts like legs, palps, antennae etc. [25]. In BOA, it is assumed that each butterfly is able to generate fragrance with some intensity. This fragrance is further correlated with fitness of the butterfly. It means that whenever a butterfly moves from one position to other particular position in the search space, its fitness will vary accordingly. Further, the fragrance which is being generated by the butterflies is propagated over distance to all the other butterflies in that region. The propagated fragrance is sensed by the other butterflies and a collective social knowledge network is formed. Whenever a butterfly is able to sense fragrance from the best butterfly in the region, it moves towards the best butterfly and this phase is termed as global search phase of BOA. In the second scenario, when a butterfly is not able to sense fragrance of any other butterfly in the search space, it will move randomly in the region and this phase is termed as local search phase in BOA.

The underlying strength of BOA lies in the mechanism of modulating the fragrance in the whole searching process. In order to understand the modulation of fragrance, first, it should be discussed that how any sense like sound, smell, heat, light etc. is processed by a stimulus of a living organism. The basic concept of sensing is dependent on three vital parameters i.e., sensory modality (c), stimulus intensity (I) and power exponent (a). Sensory modality defines the method by which the form of energy is measured and processed by the stimulus. Different modalities/senses can be smell, sound, light, temperature or pressure etc. and in BOA, it is fragrance. I represents the magnitude of the physical/actual stimulus and in BOA, it is correlated with the fitness of the butterfly/solution i.e. a butterfly with higher fragrance or greater fitness value attracts other butterflies in the search space. The parameter a allows response compression i.e. as the stimulus gets stronger; insects become increasingly less sensitive to the stimulus changes [26, 27].

1: Objective function $f(\mathbf{x})$, $x_i(i = 1, 2, \ldots, n)$

2: Generate initial population of butterflies

3: Find the best solution in the initial population

4: Define switch probability p

5: **while** stopping criteria not met **do**

6:     **for each** butterfly in population **do**

7:         Draw *rand* from a uniform distribution in

8:         Calculate *fragrance* of the butterfly using Eq. 1

9:         **if** *rand* < *p* **then**

10:             Global search using Eq. 2

11:         **else**

12:             Do Local search using Eq. 3

13:         **end if**

14:         Evaluate new solutions

15:         Update Better Solutions.

16:     **end for**

17:     Find the current best solution

18: **end while**

19: Output the best solution found.

Algorithm1. Pseudocode of Butterfly Optimization Algorithm.

Considering the facts of biological butterflies, the searching phenomenon is based on two important issues: (1) variation of *I*, (2) formulation of *f*. For simplicity, *I* of a butterfly is associated with

encoded objective function in BOA. However, *f* is relative i.e. it should be sensed by other butterflies in the search space. Therefore, bearing in mind these concepts of biological butterflies, the fragrance is formulated as a function of the physical intensity of stimulus in BOA [26] as follows:

$$f_i = cI^a \tag{1}$$

where is the perceived magnitude of fragrance, i.e., how stronger the fragrance is perceived by *i*-th butterfly, *c* is the sensory modality, *I* is the stimulus intensity and *a* is the power exponent dependent on modality, which accounts degree of absorption. There are two important phases in the BOA, they are; global search phase and local search phase. In global search phase, the butterfly takes a step towards the fittest butterfly/solution which can be represented as:

$$x_i^{t+1} = x_i^t + (\,lèvy(\lambda) \times g^* - x_i^t) \times f_i \tag{2}$$

where is the solution vector for *i*-th butterfly in iteration *t*. Here represents the best solution found among all the solutions in current generation. The fragrance of *i*-th butterfly is represented by while step size is represented as . Local search phase can be represented as:

$$x_i^{t+1} = x_i^t + (\,lèvy(\lambda) \times x_k^t - x_j^t) \times f_i \tag{3}$$

where and are *j*-th and *k*-th butterflies chosen randomly from the solution space. If and belongs to the same swarm and is the step size, then Eq. 3 becomes a local random walk. The food searching process can occur at local as well as global level, so considering this; a switch probability *p* is used in BOA to control the common global search and intensive local search. The above mentioned steps frame the complete algorithm of Butterfly Optimization Algorithm and its pseudocode is presented in Algorithm 1.

IV. Artificial Bee Colony Algorithm

The artificial bee colony algorithm draws its inspiration from the intelligent behavior of real honey bees [11]. In ABC, The honey bees are divided into three categories: employed bees, onlooker bees, and scout bees. The colony is divided into two halves; first half consists of employed bees whereas the second half consists of onlookers bees. Each solution in the search space consists of a set of optimization parameters which represent a food source population [28]. The number of food sources is equal to the number of employed bees, i.e., one employed bee is there for every food source. Employed bees share the information regarding food sources with onlooker bees, which wait in the hive. Based on the shared information, a food source is selected to be exploited. A few employed bees whose food source has been exhausted are translated to scout bees. After the initialization of population, the whole iterative process of ABC is divided into three phases; (1) employed bee search phase, (2) onlooker bee selection phase and (3) scout bee phase. In the first phase, a candidate food position is produced using the old one. It is achieved by the following equation:

$$v_i^j = x_i^j + \varphi_i^j(x_i^j - x_k^j) \tag{4}$$

where and are the indexes, chosen randomly whereas *k* is determined randomly, different from *i*. represents a random number in the range [-1, 1]. A food source V_i within the neighborhood of every food source site represented by x_i, is calculated by modifying one parameter of x_i. After the V_i is calculated, it will be evaluated and compared to x_i. A greedy selection method is used to select the better one between x_i and V_i depending on fitness values representing the

nectar amount of the food sources at x_i and V_i respectively. V_i will replace x_i if the fitness of V_i is equal to or better than that of x_i, and become a new member of the populations; otherwise x_i is retained. In the second phase, one food source is selected by each onlooker bee on the basis of fitness value obtained from the employed bees. Now, any fitness based probability election strategy can be used like rank based, roulette wheel, tournament selection etc. In the basic version of ABC, roulette wheel selection strategy is used, which can be mathematically defined as:

$$P_i = \frac{fit(x_i)}{\sum_{m=1}^{n} fit(x_m)}$$

(5)

In Eq. 5 the fitness of the solution i is represented as fit(x_i). Obviously, higher fitness value means more probability of getting selected. After the selection of food source, onlooker bees will move towards the selected food source and a new candidate position is produced in the neighborhood of the selected food source by using Eq. 4.

1: Generate the initial population $x_i (i = 1, 2, \ldots, n)$

2: Evaluate the fitness of the population

3: **while** stopping criteria not met **do**

4:   **for each** employed bee **do**

5:     Produce new solution V_i by using Eq. 4

6:     Calculate its fitness value fit(V_i)

7:     Apply greedy selection process

8:     Calculate P_i for the solution (x_i) by Eq. 5

9:   **end for**

10: **for each** onlooker bee **do**

11:     Select a solution x_i depending on P_i

12:     Produce new solution V_j

13:     Calculate its fitness value fit(V_j)

14:     Apply greedy selection process

15:   end for

16:   If there is an abandoned solution for the scout,

     then replace it with a new solution using Eq. 6

17:   Memorize better solutions

18: **end while**

19: Output the best solution found.

Algorithm 2. Pseudocode of Artificial Bee Colony Algorithm.

In the last phase, after the completion of searches by employed and onlooker bees, the algorithm checks whether the source which is exhausted, needs to be abandoned. This means that if a better position cannot be attained in predetermined number of chances i.e. cycles, then that particular food source is assumed as abandoned and a new food source is calculated using:

$$x_i = x_{min} + r(x_{max} - x_{min})$$

(6)

In Eq. 6, r is a random number in the range [0, 1] whereas x_{min} and x_{max} represents the respective lower and upper bounds of variable x_i. The last phase helps the algorithm to avoid suboptimal solutions. Detailed pseudocode of the ABC algorithm is given in Algorithm 2.

V. The Proposed Hybrid BOA/ABC Algorithm

In the past, various hybrid optimization algorithms have been developed which demonstrate efficient results [13-23]. So, based on the description of BOA and ABC in the previous sections, the two approaches are combined and a hybrid BOA/ABC algorithm is proposed. Using the strengths of both the approaches, BOA/ABC is able to update the poor solutions which accelerate its convergence speed. BOA and ABC are very efficient in exploring the search space and exploiting the solutions. In the proposed hybrid algorithm, the optimization process of ABC and BOA is used effectively. Therefore, the lack of exploitation is overcome in BOA. In the current study, a modified version of ABC is hybridized with BOA to solve numerical optimization problems [12, 29]. Another modification is that in BOA, lèvy flights are used but in BOA/ABC pseudorandom numbers are used [9]. By incorporating ABC into BOA, the BOA/ABC algorithm is developed as shown in Algorithm 3.

1: Objective function $f(\mathbf{x})$, $x_i (i = 1, 2, \ldots, n)$

2: Generate initial population of individuals

3: Find the best solution in the initial population

4: Define switch probability p

5: **while** stopping criteria not met **do**

6:   **for each** butterfly in population **do**

7:     Draw $r1$ and $r2$ from a uniform distribution in

8: **if** $r1<0.5$ **then**

9:     Calculate *fragrance* of the butterfly using Eq. 1

10:    **if** $r2< p$ **then**

11:     Global search using Eq. 2

12: **else**

13:       Do Local search using Eq. 3

14:    **end if**

15:    Evaluate new solutions

16:    Update Better Solutions.

17: **end if**

18: **else**

19: **for each** employed bee **do**

20:     Produce new solution V_i by using Eq. 4

21:     Calculate its fitness value fit(V_i)

22:    Apply greedy selection process

23:     Calculate P_i for the solution (x_i) by Eq. 5

24: **end for**

25: **for each** onlooker bee **do**

26:     Select a solution x_i depending on P_i

27:     Produce new solution V_j

28:     Calculate its fitness value fit(V_j)

29:     Apply greedy selection process

30: **end for**

31:   If there is an abandoned solution for the scout,

     then replace it with a new solution using Eq. 6

32:   Memorize better solutions

33:   **end else**

34: **end while**

35: Output the best solution found

Algorithm 3. Pseudocode of the proposed BOA/ABC Algorithm.

In comparison to the original BOA, the proposed approach needs a very small amount of computational cost additionally. The incorporation of ABC in BOA enables the proposed algorithm to avoid

the local optima trap problem and increase its convergence speed. The proposed BOA/ABC algorithm is aimed at hybridizing components from both BOA and ABC in order to have an algorithm that easily solve separable problems as BOA while having a rotationally invariant behavior as ABC, at the same time. Detailed pseudocode of the BOA/ABC algorithm is given in Algorithm 3.

VI. BENCHMARK PROBLEMS AND EXPERIMENT SETTINGS

Every novel optimization algorithm must be subject to a testbed of benchmark functions in order to validate the algorithm. Ideally, the selected test functions should have characteristics alike to those of a real-world problem in order to better assess an algorithm [30]. However, no standard benchmark function testbed available. Still there are many benchmark functions which are well known and recommended by various researchers in the past [31]. So in this study, ten benchmark functions are used to validate the proposed algorithm. All the functions used in this study are minimization problems. The benchmark functions which are selected in this study are chosen in such a way that the proposed algorithm is tested on almost all types of

problems. Considering this viewpoint a diverse subset of benchmark functions is chosen in this study. This subset can be classified into four major categories. In the first category, function can either have single optima or multiple optima. In second category, the number of dimensions can be low or high. High dimensional problems are very difficult to solve that's why most of the benchmark functions used in this work are high dimensional.

Another category is that some functions are separable and some are non-separable. In the last category, functions with noisy data are used. These functions are alike real world problems which contains noisy data which makes them difficult to solve. New algorithms must be tested on all these kinds of test functions in order to properly validate and demonstrate the efficiency of the algorithm. These benchmark functions are described in Table 1 along with their dimensions and range.

Rigorous nonparametric statistical framework is used to compare the performance of the proposed algorithm with BOA and ABC algorithm. All initial solutions of the population are randomly generated for each run of the algorithm. The population size is fixed to 30 for all the algorithms. In order to avoid discrepancy due to

TABLE I.
BENCHMARK FUNCTIONS USED IN THE PRESENT COMPUTATIONAL ANALYSIS.

S. No.	Benchmark functions	Formula	Dimensions	Range	Optima
f_1	Beale	$$P_i = \frac{fit(x_i)}{\sum_{m=1}^{n} fit(x_m)}$$	2	(-4.5,4.5)	0
f_2	Cigar	$$f(x) = x_1^2 + 10^6 \sum_{i=2}^{n} x_i^2$$	30	(-10,10)	0
f_3	Easom	$$f(x) = -\cos(x_1)\cos(x_2)\exp(-(x_1-\pi)^2-(x_2-\pi)^2)$$	2	(-100,100)	-1
f_4	Griewank	$$f(x) = 1 + \frac{1}{4000}\sum_{i=1}^{n}x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right)$$	30	(-600, +600)	0
f_5	Quartic function with noise	$$f(x) = \sum_{i=1}^{n} ix_i^4 + rand(0,1)$$	30	(-1.28 1.28)	0
f_6	Rastrigin	$$f(x) = \sum_{i=1}^{n}(x_i^2 - 10\cos(2\pi x_i) + 10)$$	30	(-5.12, 5.12)	0
f_7	Shubert	$$f(x) = \left(\sum_{i=1}^{5} i\cos((i+1)x_1 + i)\right)\left(\sum_{i=1}^{5} i\cos((i+1)x_2 + i)\right)$$	2	(-10,10)	-186.73
f_8	Step	$$f(x) = \sum_{i=1}^{n}(\lfloor x_i \rfloor + 0.5)^2$$	30	(-100,100)	0
f_9	Levy	$$f(x) = sin^2(\pi w_1) + \sum_{i=1}^{d-1}(w_i-1)^2[1+10\sin^2(\pi w_i+1)] + (w_d-1)^2[1+\sin^2(2\pi w_d)]$$	30	(-10,10)	-21.5023
f_{10}	Power Sum	$$f(x) = \sum_{i=1}^{d}\left(\left[\sum_{j=1}^{d} x_j^i\right] - b_i\right)^2 \; ; \; b = [8\ 18\ 44\ 144]$$	4	(0,d)	0

the stochastic properties of the algorithms, 30 independent runs for each optimization algorithm having 30 different initial population. The proposed algorithm is implemented in C++ and compiled using Qt Creator 2.4.1 (MinGW) under Microsoft Windows 8 operating system. All simulations are carried out on a computer with an Intel(R) Core(TM) i5-3210@2.50Ghz CPU.

VII.   RESULTS AND DISCUSSIONS

Our proposed BOA/ABC approach is compared with classical BOA and ABC algorithm in order to demonstrate its superiority. The values of mean and standard deviation of all the algorithms are presented in Table 2. The best values are highlighted in bold. According to the simulation results, it can be analyzed that BOA/ABC is significantly better than BOA on all the ten benchmark functions used in the study. It can be observed from the Table 2 that the BOA/ABC has ability to converge faster and escape from local optima. The underlying reason behind the better performance of BOA/ABC can be explained by the fact that exploration of BOA is good but it lacks the exploitation. Compared to the ABC algorithm, the proposed BOA/ABC can converge faster, if proper terminating conditions are set.

For example, after a number of iterations the best fitness value is not improved whereas the computation time can be reduced significantly. BOA has a disadvantage that exploitation to found good solutions is very bad whereas the exploration of solution of BOA algorithm is good. On the other hand the BOA/ABC can overcome this shortcoming by incorporating the strengths of ABC which allows the BOA/ABC to avoid the local optima trap problem and on the same side it maintains the overall good solution quality during the optimization process. The proposed BOA/ABC requires less population size and needs less computational time to reach global optima.

The proposed BOA/ABC demonstrates its better search ability than the individual algorithms i.e. BOA and ABC algorithm. Integrating the BOA and ABC's local and global search abilities, BOA/ABC hybrid algorithm demonstrate better global and local search ability than the original BOA. If we consider the same population size of BOA/ABC and ABC, then BOA/ABC obtains optimized results at least as good as the results obtain by ABC algorithm. When the population size of BOA/ABC is less than ABC, then the BOA/ABC performs better whereas when ABC has large population size than BOA/ABC, ABC shows better performance.
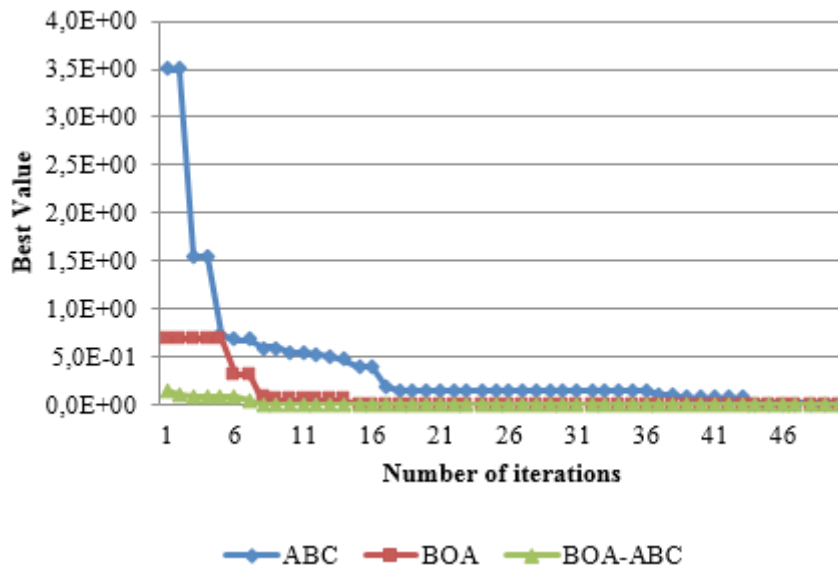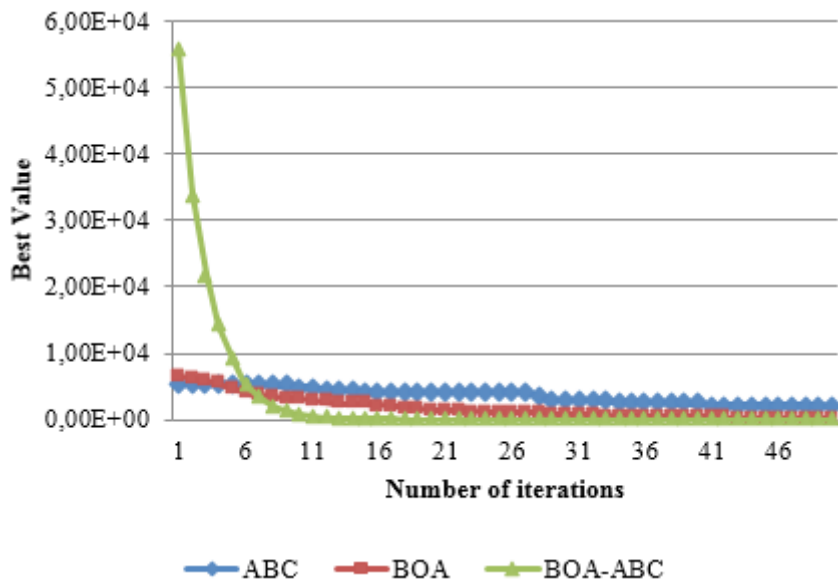

Fig.1. Convergence curves of f_1 function.

Fig. 2. Convergence curves of f_8 function.

TABLE II.
Statistical results of different methods.

		ABC	BOA	BOA/ABC
f_1	Mean	3.636E-15	9.322E-08	**1.537E-15**
	Std. Dev	1.001E-14	1.626E-07	**2.687E-15**
f_2	Mean	3.679E-16	1.830E-07	**1.162E-16**
	Std. Dev	7.568E-17	3.010E-08	**1.525E-17**
f_3	Mean	-1.000E+00	-1.850E-04	**-1.00E+00**
	Std. Dev	1.826E-07	1.013E-03	**0.000E+00**
f_4	Mean	3.896E-16	1.539E-05	**1.153E-16**
	Std. Dev	6.874E-17	2.131E-06	**6.220E-18**
f_5	Mean	1.983E-01	1.333E-01	**2.622E-03**
	Std. Dev	5.231E-02	1.142E-02	**8.868E-04**
f_6	Mean	2.632E-08	2.689E+02	**2.309E-15**
	Std. Dev	1.440E-07	2.111E+01	**1.054E-15**
f_7	Mean	-8.481E+01	-4.741E+01	**-8.481E+01**
	Std. Dev	1.445E-14	2.145E+01	**2.340E-03**
f_8	Mean	0.000E+00	5.573E+01	**0.000E+00**
	Std. Dev	**0.000E+00**	9.036E+00	**0.000E+00**
f_9	Mean	4.860E-16	8.838E-01	**4.116E-16**
	Std. Dev	1.151E-16	1.964E-01	**1.134E-16**
f_{10}	Mean	2.513E-02	2.542E-01	**6.028E-03**
	Std. Dev	2.081E-02	2.405E-10	**5.369E-03**

The strategy of BOA/ABC is designed in such a way that it balances the local and global search processes of BOA and further it should enhance the performance of BOA in order to use limited computation resources. The faster convergence of BOA/ABC is shown in the Fig.1 and Fig.2. In addition, the BOA/ABC allows the BOA to enhance the exploration and exploitation capabilities. It is worth mentioning that using ABC in BOA enables the algorithm to improve the poor solutions and meanwhile, it is able to avert the good solution from being destroyed during the optimization process. Hence, the better performance of BOA/ABC.

Due to limitations of space, few representative convergence graphs on benchmark functions are shown in Fig. 1-2. By carefully analyzing the Fig. 1-2, it can be observed that the solution quality of BOA/ABC improves continuously during the optimization process. Moreover, in the whole iteration process, BOA/ABC keeps an edge over the BOA and ABC algorithm. The reason might be that BOA/ABC better manages the global search and local search process in the optimization process. This indicates that BOA/ABC has the ability to escape from poor local optima and locate a good near-global optimum. Overall, the performance of BOA/ABC is highly competitive with ABC and BOA, particularly for the high-dimensional and noisy problems. Besides, BOA/ABC is much better than BOA for all the problems.

In order to better demonstrate the performance of the proposed algorithm, pair-wise comparison of the simulation results is done using the Wilcoxon signed rank test. In this method, the algorithm having a lower rank is considered better on the particular benchmark function. If Algo-1 and Algo-2 perform better than Algo-3, then it can be concluded that Algo-1 and

Algo-2 outperform Algo-3 on the specified benchmark function. However, if Algo-1outperforms Algo-2 but both Algo-1 and Algo-2 perform similarly with Algo-3, all of Algo-1, Algo-2, and Algo-3 are positioned in the same rank. The order of algorithms within same rank is alphabetical and has no implication on performance. In Table 3,

Algo-1 < Algo-2 stands for "Algo-1 shows better performance than Algo-2", and Algo-1 ≈ Algo-2 stands for "Algo-1 and Algo-2 performs similarly" on the particular benchmark function. On the basis of the pair-wise findings, partial orderings of the algorithms is constructed as shown in Table 3 [33].

TABLE III.
Pair-wise Wilcoxon signed rank test results

Function	Wilcoxon Signed Rank Test Order
f_1	BOA/ABC< ABC < BOA
f_2	BOA/ABC< ABC < BOA
f_3	ABC≈BOA/ABC< BOA
f_4	BOA/ABC< ABC < BOA
f_5	BOA/ABC< BOA < ABC
f_6	BOA/ABC< ABC < BOA
f_7	ABC≈BOA/ABC< BOA
f_8	ABC≈BOA/ABC< BOA
f_9	BOA/ABC< ABC < BOA
f_{10}	BOA/ABC< ABC < BOA

According to the results shown in the Table 3, BOA/ABC indicates its outstanding capability in fast converging to the global optimum while avoiding premature convergence. It can be analyzed that BOA/ABC demonstrates better statistical results as well as higher robustness. This statement is supported by the results in Table 2 and Table 3 which shows the better performance of BOA/ABC over other algorithms.

With the intention to better demonstrate the performance of proposed BOA/ABC, the results in the Wilcoxon signed rank test are used to further rank the algorithms according to their performance [32]. For every benchmark function, the best/first algorithm is assigned value 1; the second best is assigned value 2 and so on. In case of ties,

an average rank is assigned to the algorithms involved in the tie. For example, in case of function f_3, ABC and BOA/ABC shares same rank, and they are first and second rank algorithm for the particular function. As a result, an average rank value of 1.5 is assigned to these two algorithms. Further, the rank values of each algorithm is summed together in order to have overall assessment of the algorithm in solving general optimization problems. Similar evaluation methods have been adopted in the past by various researchers for testing of different metaheuristic algorithms [34].The test results are presented in Table 4.The results obtained by the statistical assessment analysis supports our previous observations. In all the benchmark functions used in this study, BOA/ABC possesses a superior position on all the functions. It can be observed from Table 3 and Table 4 that BOA/ABC has the most stable position on the benchmark functions. The performance of BOA and ABC is reasonable but neither of these two catches up with BOA/ABC. So it can be concluded that in general BOA/ABC has better optimization performance in terms of efficient results and stability.

TABLE IV.
RANK SUMMARY OF STATISTICAL ASSESSMENT RESULTS

Function	ABC	BOA	BOA/ABC
f_1	2	3	1
f_2	2	3	1
f_3	1.5	3	1.5
f_4	2	3	1
f_5	3	2	1
f_6	2	3	1
f_7	1.5	3	1.5
f_8	1.5	3	1.5
f_9	2	3	1
f_{10}	2	3	1
Sum	**19.5**	**29**	**11.5**

Furthermore, in order to demonstrate the efficiency of the proposed BOA/ABC algorithm in terms of finding the optimum value, an experiment in conducted on the particular algorithms. In this experiment, all the algorithms were initialized in regions that include the global optimum for a fair evaluation. The algorithms were run for 50 times to catch their stochastic properties. The goal of this experiment is not to find the global optimum values but to find out the potential of the algorithms. The success rate of algorithm is defined in Eq. (7) which has been used for comparison in this study.

$$S = 100 \frac{NT_{successful}}{NT_{all}} \Big|_{Qlevel}$$

(7)

Where is the number of trials, which found the solution on their the allowable maximum iteration which is set to 500 in this case study. is the number of alltrials. The results of this experiment are shown in the Table 5. It can be analyzed from these simulation results that BOA/ABC has the ability to reach the global optima in almost every time it is executed. The simulation results of ABC and BOA are also satisfactory but not as good as of BOA/ABC. The underlying reason behind the better performance of BOA/ABC is that incorporating the ABC in BOA has improved the reliability of the global optimality and it has also enhanced the quality of the results.

TABLE V.
SUCCESS RATES OF DIFFERENT METHODS

Function	ABC	BOA	BOA/ABC
f_1	82	78	100
f_2	80	75	100
f_3	95	88	95
f_4	80	75	93
f_5	80	87	98
f_6	78	70	90
f_7	92	90	94
f_8	90	86	92
f_9	89	86	94
f_{10}	90	87	95

These simulation results do not indicate that BOA/ABC is "better" than BOA and ABC. As this common statement would be an unjustified remark, particularly in view of the no free lunch theorem [35]. However, these results illustrate that BOA/ABC demonstrates superior performance than BOA and ABC on the particular benchmark functions used in this study. The simulation results point towards the fact that BOA/ABC is competitive with the other algorithms and it has the ability to provide efficient results on real world complex problems.

VIII. CONCLUSION

In the present work, a hybrid BOA/ABC is proposed for numerical optimization problems. Although BOA is an effective algorithm, however, it faces the problem of entrapment into local optima. In order to solve this problem, ABC is introduced in BOA. ABC enables the proposed BOA/ABC to improve the poor solutions and along with it is able to save the good solutions in order to maintain the diversity of the population. Ten benchmark functions are used to investigate the performance of BOA/ABC. The results demonstrated that BOA/ABC makes better use of exploration and exploitation of the butterflies' information, than original BOA. The convergence of BOA/ABC is faster than original BOA and it shows superior results on higher dimensional problems.

REFERENCES

[1] Yang, Xin-She. *Nature-inspired metaheuristic algorithms*. Luniver press, 2010.

[2] Vasant, Pandian. *Handbook of Research on Artificial Intelligence Techniques and Algorithms, 2 Volumes*. Information Science Reference-Imprint of: IGI Publishing, 2015.

[3] Kennedy, James. "Particle swarm optimization." In *Encyclopedia of machine learning*, pp. 760-766. Springer US, 2011.

[4] Yang, Xin-She. "Firefly algorithm, stochastic test functions and design optimisation." *International Journal of Bio-Inspired Computation* 2, no. 2 (2010): 78-84.

[5] Yang, Xin-She, and Suash Deb. "Cuckoo search via Lévy flights." In *Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on*, pp. 210-214. IEEE, 2009.

[6] Dorigo, Marco, Mauro Birattari, and Thomas Stutzle. "Ant colony optimization." *IEEE computational intelligence magazine* 1, no. 4 (2006): 28-39.

[7] Onwubolu, Godfrey C., and B. V. Babu. *New optimization techniques in*

engineering. Vol. 141. Springer, 2013.

[8] Gosavi, Abhijit. "Simulation-based optimization." Parametric Optimization Techniques and Reinforcement Learning. Kluwer Academic Publishers (2003).

[9] Arora, Sankalap, and Satvir Singh. "Butterfly algorithm with Lèvy Flights for global optimization." In *Signal Processing, Computing and Control (ISPCC), 2015 International Conference on*, pp. 220-224. IEEE, 2015.

[10] Arora, Sankalap, and Satvir Singh. "An improved butterfly optimization algorithm with chaos." *Journal of Intelligent &Fuzzy Systems* (2016)

[11] Karaboga, Dervis, and Bahriye Basturk. "A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm." *Journal of global optimization* 39, no. 3 (2007): 459-471.

[12] Karaboga, Dervis, Beyza Gorkemli, Celal Ozturk, and Nurhan Karaboga. "A comprehensive survey: artificial bee colony (ABC) algorithm and applications." *Artificial Intelligence Review* 42, no. 1 (2014): 21-57.

[13] Gong, Wenyin, Zhihua Cai, and Charles X. Ling. "DE/BBO: a hybrid differential evolution with biogeography-based optimization for global numerical optimization." *Soft Computing* 15.4 (2010): 645-665.

[14] Guo, Lihong, et al. "An effective hybrid firefly algorithm with harmony search for global numerical optimization." *The Scientific World Journal* 2013 (2013).

[15] Sahnehsaraei, M. Andalib, et al. "A hybrid global optimization algorithm: particle swarm optimization in association with a genetic algorithm." *Complex System Modelling and Control Through Intelligent Soft Computations*. Springer International Publishing, 2015. 45-86.

[16] Abdullah, Afnizanfaizal, et al. "A new hybrid firefly algorithm for complex and nonlinear problem." *Distributed Computing and Artificial Intelligence*. Springer Berlin Heidelberg, 2012. 673-680.

[17] El-Abd, Mohammed. "A hybrid ABC-SPSO algorithm for continuous function optimization." *Swarm Intelligence (SIS), 2011 IEEE Symposium on*. IEEE, 2011.

[18] Li, Li, Fangmin Yao, Lijing Tan, Ben Niu, and Jun Xu. "A novel DE-ABC-based hybrid algorithm for global optimization." In International Conference on Intelligent Computing, pp. 558-565. Springer Berlin Heidelberg, 2011.

[19] Fleurent, Charles, and Jacques A. Ferland. "Genetic and hybrid algorithms for graph coloring." *Annals of Operations Research* 63, no. 3 (1996): 437-461.

[20] Prosser, Patrick. "Hybrid algorithms for the constraint satisfaction problem." *Computational intelligence* 9, no. 3 (1993): 268-299.

[21] Kalra, Shifali, and Sankalap Arora. "Firefly Algorithm Hybridized with Flower Pollination Algorithm for Multimodal Functions." In *Proceedings of the International Congress on Information and Communication Technology*, pp. 207-219. Springer Singapore, 2016.

[22] Gupta, Samiti, and Sankalap Arora. "A Hybrid Firefly Algorithm and Social Spider Algorithm for Multimodal Function." In *Intelligent Systems Technologies and Applications*, pp. 17-30. Springer International Publishing, 2016.

[23] Fukuda, Sho, Yuuma Yamanaka, and Takuya Yoshihiro. "A Probability-based Evolutionary Algorithm with Mutations to Learn Bayesian Networks." *International Journal of Interactive Multimedia and Artificial Intelligence* 3.1 (2014): 7-13.

[24] Arora, Sankalap, and Satvir Singh. "A conceptual model of Butterfly algorithm" In *Latest initiatives and Innovations in Communication and Electronics (IICE), 2015 National Conference on*, pp. 69-72. , 2015.

[25] Blair, Robert B., and Alan E. Launer. "Butterfly diversity and human land use: Species assemblages along an urban grandient." *Biological conservation* 80, no. 1 (1997): 113-125.

[26] Zwislocki, Jozef J. *Sensory neuroscience: Four laws of psychophysics*. Springer Science & Business Media, 2009.

[27] Stevens, Stanley Smith. *Psychophysics*. Transaction Publishers, 1975.

[28] Karaboga, Dervis, and Bahriye Akay. "A comparative study of artificial bee colony algorithm." *Applied mathematics and computation* 214, no. 1 (2009): 108-132.

[29] Akay, Bahriye, and Dervis Karaboga. "A modified artificial bee colony algorithm for real-parameter optimization." *Information Sciences* 192 (2012): 120-142.

[30] Meza, Joaquín, Helbert Espitia, Carlos Montenegro, and Rubén González Crespo. "Statistical analysis of a multi-objective optimization algorithm based on a model of particles with vorticity behavior." *Soft Computing* (2015): 1-16

[31] Liang, J. J., B. Y. Qu, P. N. Suganthan, and Q. Chen. "Problem definitions and evaluation criteria for the CEC 2015 competition on learning-based real-parameter single objective optimization." *Technical Report201411A, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore* (2014).

[32] Maesono, Yoshihiko. "Competitors of the Wilcoxon signed rank test." *Annals of the Institute of Statistical Mathematics* 39, no. 1 (1987): 363-375.

[33] Houck, Christopher R., Jeff Joines, and Michael G. Kay. "A genetic algorithm for function optimization: a Matlab implementation." *NCSU-IE TR 95*, no. 09 (1995).

[34] Lam, Albert YS, Victor OK Li, and J. Q. James. "Real-coded chemical reaction optimization." *IEEE Transactions on Evolutionary Computation* 16, no. 3 (2012): 339-353.

[35] Ho, Yu-Chi, and David L. Pepyne. "Simple explanation of the no-free-lunch theorem and its implications." *Journal of optimization theory and applications* 115, no. 3 (2002): 549-570.

Sankalap Arora was born on Nov 3, 1988. He received his Bachelor's degree (B.Tech.) and Master's degree (M.Tech.) from Lovely Professional University, Phagwara, Punjab (India) with specialization in Computer Science & Engineering. He is currently pursuing his Doctoral degree (Ph.D.) from Punjab technical University, Kapurthala, Punjab (India). He has more than 5 years' research and teaching experience. His fields of special interest include Nature Inspired Algorithms, Engineering Design problems and Wireless Sensor Networks. He has published nearly 20 research papers in reputed International Journals and Conferences.

Satvir Singh was born on Dec 7, 1975. He received his Bachelor's degree (B.Tech.) from Dr. B.R.Ambedkar National Institute of Technology, Jalandhar, Punjab (India) with specialization in Electronics & Communication Engineering in year 1998,Master's degree (M.E.) from Delhi Technological University (Formerly, Delhi College of Engineering), Delhi (India) with distinction in Electronics & Communication Engineering in year 2000 and Doctoral degree (Ph.D.) from Maharshi Dayanand University, Rohtak, Haryana (India) in year 2011. He has more than 13 years' research and teaching experience years of teaching experience. His fields of special interest include Evolutionary Algorithms, High Performance Computing, Type-1 & Type-2 Fuzzy Logic Systems and Wireless Sensor Networks for solving engineering problems. He is active member of an editorial board of International Journal of Electronics Engineering and published nearly 30 research papers in International Journals and Conferences