Memetic Comp. (2013) 5:213–227
DOI 10.1007/s12293-012-0104-0

REGULAR RESEARCH PAPER

Opposition based lévy flight artificial bee colony

Harish Sharma · Jagdish Chand Bansal · K. V. Arya

Received: 20 July 2012 / Accepted: 30 November 2012 / Published online: 13 December 2012
© Springer-Verlag Berlin Heidelberg 2012

Abstract Artificial Bee Colony (ABC) is a well known optimization approach to solve nonlinear and complex problems. It is relatively a simple and recent population based probabilistic approach for global optimization. Similar to other population based algorithms, ABC is also computationally expensive due to its slow nature of search process. The solution search equation of ABC is significantly influenced by a random quantity which helps in exploration at the cost of exploitation of the search space. In the solution search equation of ABC due to the large step size the chance of skipping the true solution is high. Therefore, in this paper, to balance the diversity and convergence capability of the ABC, Lévy Flight random walk based local search strategy is proposed and incorporated with ABC along with opposition based learning strategy. The proposed algorithm is named as Opposition Based Lévy Flight ABC. The experiments over 14 un-biased test problems of different complexities and five well known engineering optimization problems show that the proposed algorithm outperforms the basic ABC and its recent variants namely Gbest guided ABC, Best-So-Far ABC, and Modified ABC in most of the experiments.

Keywords Swarm intelligence · Evolutionary computation · Memetic algorithm · Lévy flight local search

H. Sharma (✉) · K. V. Arya
ABV-Indian Institute of Information Technology and Management, Gwalior, India
e-mail: harish.sharma0107@gmail.com

K. V. Arya
e-mail: kvarya@gmail.com

J. C. Bansal
South Asian University, New Delhi, India
e-mail: jcbansal@sau.ac.in

1 Introduction

Swarm Intelligence has become an emerging and interesting area in the field of nature inspired techniques that is used to solve optimization problems during the past decade. It is based on the collective behavior of social creatures. Swarm based optimization algorithms find solution by collaborative trial and error. Social creatures utilize their ability of social learning to solve complex tasks. Peer to peer learning behavior of social colonies is the main driving force behind the development of many efficient swarm based optimization algorithms. Researchers have analyzed such behaviors and designed algorithms that can be used to solve nonlinear, non-convex or discrete optimization problems. Previous research [8,16,22,32] have shown that algorithms based on swarm intelligence have great potential to find solutions of real world optimization problems. The algorithms that have emerged in recent years include ant colony optimization (ACO) [8], particle swarm optimization (PSO) [16], bacterial foraging optimization (BFO) [20] etc.

Artificial bee colony (ABC) optimization algorithm introduced by Karaboga [13] is a recent addition in this category. This algorithm is inspired by the behavior of honey bees when seeking a quality food source. Like any other population based optimization algorithm, ABC consists of a population of potential solutions. The potential solutions are food sources of honey bees. The fitness is determined in terms of the quality (nectar amount) of the food source. ABC is relatively a simple, fast and population based stochastic search technique in the field of nature inspired algorithms.

There are two fundamental processes which drive the swarm to update in ABC: the variation process, which enables exploring different areas of the search space, and the selection process, which ensures the exploitation of the previous experience. However, it has been shown that the ABC

may occasionally stop proceeding toward the global optimum even though the population has not converged to a local optimum [14]. It can be observed that the solution search equation of ABC algorithm is good at exploration but poor at exploitation [38]. Therefore, to maintain the proper balance between exploration and exploitation behavior of ABC, it is highly required to develop a local search approach in the basic ABC to exploit the search region. In past, very few efforts have been done in this direction. Kang et al. [12] proposed a Hooke Jeeves Artificial Bee Colony algorithm (HJABC) for numerical optimization. In HJABC, authors incorporated a local search technique which is based on Hooke Jeeves method (HJ) [11] with the basic ABC. Further, Mezura-Montes and Velez-Koeppel [18] introduced a variant of the basic ABC named Elitist Artificial Bee Colony. In this work, the authors integrated two local search strategies. The first local search strategy is used when 30, 40, 50, 60, 70, 80, 90, 95 and 97 % of function evaluations have been completed. The purpose of this is to improve the best solution achieved so far by generating a set of 1000 new food sources in its neighborhood. The other local search works when 45, 50, 55, 80, 82, 84, 86, 88, 90, 91,92, 93, 94, 95, 96, 97, 98, and 99 % of function evaluations have been reached.

In this paper, Lévy Flight random walk based local search strategy is proposed. The proposed local search strategy is used for finding the global optima of a unimodal and/or multimodel functions by iteratively reducing the step size to update the candidate solution in the search space inside which the optima is known to be exist. Further, to balance the diversity and convergence capability of ABC, the concept of opposition based learning (OBL) [31] is taken into consideration. The same concept has been applied in Differential Evolution (DE) [22] optimization algorithm to improve its convergence speed [24]. The main concept behind OBL is the simultaneous consideration of a solution and its corresponding opposite counter part in order to find out a better approximation for the current candidate solution. The quality of the solution is measures on the basis of its distance form the global optima. On the basis of probability theory, it can be easily state that there is 50 % chance for an opposite counter part solution to be nearer to the global optima. Therefore, in this paper, the concept of OBL and the proposed local search strategy are integrated with the basic ABC. Further, the proposed algorithm is compared by experimenting on 14 un-biased test problems (i.e. the problems which solutions do not exist on origin, axes or diagonal) and five popular engineering optimization problems to the basic ABC and its recent variants named, Gbest guided ABC ($GABC$) [38], Best-So-Far ABC ($BSFABC$) [3] and Modified ABC ($MABC$) [1].

Rest of the paper is organized as follows: Sect. 2 describes brief overview of the basic ABC. Lévy Flight local search strategy is proposed in Sect. 3. In Sect. 4, concept of opposition based learning is described. Opposition Based Lévy Flight ABC (OBLFABC) is proposed and tested in Sect. 5. In Sect. 6, a comprehensive set of experimental results are provided. Finally, in Sect. 7, paper is concluded.

2 Artificial bee colony (ABC) algorithm

The ABC algorithm is relatively recent swarm intelligence based algorithm. The algorithm is inspired by the intelligent food foraging behavior of honey bees. In ABC, each solution of the problem is called food source of honey bees. The fitness is determined in terms of the quality of the food source. In ABC, honey bees are classified into three groups namely employed bees, onlooker bees and scout bees. The number of employed bees are equal to the onlooker bees. The employed bees are the bees which searches the food source and gather the information about the quality of the food source. Onlooker bees stay in the hive and search the food sources on the basis of the information gathered by the employed bees. The scout bee, searches new food sources randomly in places of the abandoned foods sources. Similar to the other population-based algorithms, ABC solution search process is an iterative process. After, initialization of the ABC parameters and swarm, it requires the repetitive iterations of the three phases namely employed bee phase, onlooker bee phase and scout bee phase. Each of the phase is described as follows:

2.1 Initialization of the swarm

The parameters for the ABC are, number of food sources, number trials after which a food source is considered to be abandoned and termination criteria. In the basic ABC, the number of food sources are equal to the employed bees or onlooker bees. Initially, a uniformly distributed initial swarm of SN food sources where each food source $x_i (i = 1, 2, \ldots, SN)$ is a D-dimensional vector, generated. Here D is a number of variables in the optimization problem and x_i represent the ith food source in the swarm. Each food source is generated as follows:

$$x_{ij} = x_{minj} + rand[0, 1](x_{maxj} - x_{minj}) \tag{1}$$

where x_{minj} and x_{maxj} are bounds of x_i in jth direction and $rand[0, 1]$ is a uniformly distributed random number in the range [0, 1]

2.2 Employed bee phase

In employed bee phase, employed bees modify the current solution (food source) based on the information of individual experience and the fitness value of the new solution. If the fitness value of the new solution is higher than that of the old solution, the bee updates her position with the new one

and discards the old one. The position update equation for ith candidate in this phase is

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

where $k \in \{1, 2, \ldots, SN\}$ and $j \in \{1, 2, \ldots, D\}$ are randomly chosen indices. k must be different from i. ϕ_{ij} is a random number between $[-1, 1]$.

2.3 Onlooker bees phase

After completion of the employed bees phase, the onlooker bees phase starts. In onlooker bees phase, all the employed bees share the new fitness information (nectar) of the new solutions (food sources) and their position information with the onlooker bees in the hive. Onlooker bees analyze the available information and select a solution with a probability $prob_i$ related to its fitness. The probability $prob_i$ may be calculated using following expression (there may be some other but must be a function of fitness):

$$prob_i = \frac{fitness_i}{\sum_{i=1}^{SN} fitness_i} \tag{3}$$

where $fitness_i$ is the fitness value of the solution i. As in the case of the employed bee, it produces a modification on the position in its memory and checks the fitness of the candidate source. If the fitness is higher than that of the previous one, the bee memorizes the new position and forgets the old one.

2.4 Scout bees phase

If the position of a food source is not updated up to a predetermined number of cycles, then the food source is assumed to be abandoned and scout bees phase starts. In this phase the bee associated with the abandoned food source becomes scout bee and the food source is replaced by a randomly chosen food source within the search space. In ABC, predetermined number of cycles is a crucial control parameter which is called $limit$ for abandonment.

Assume that the abandoned source is x_i. The scout bee replaces this food source by a randomly chosen food source which is generated as follows

$$x_{ij} = x_{minj} + rand[0, 1](x_{maxj} - x_{minj}), \quad \text{for } j \in \{1, 2, \ldots, D\} \tag{4}$$

where x_{minj} and x_{maxj} are bounds of x_i in jth direction.

2.5 Main steps of the ABC algorithm

Based on the above explanation, it is clear that there are three control parameters in ABC search process: The number of food sources SN (equal to number of onlooker or employed bees), the value of $limit$ and the maximum number of iterations. The pseudo-code of the ABC is shown in Algorithm 1 [14]:

Algorithm 1 Artificial Bee Colony Algorithm:

```
Initialize the parameters;
while Termination criteria is not satisfied do
  Step 1: Employed bee phase for generating new food sources.
  Step 2: Onlooker bees phase for updating the food sources
          depending on their nectar amounts.
  Step 3: Scout bee phase for discovering the new food sources
          in place of abandoned food sources.
  Step 4: Memorize the best food source found so far.
end while
Output the best solution found so far.
```

3 Lévy flight local search

Local search algorithms can be seen as a population based stochastic algorithms, where main task is to exploit the available knowledge about a problem. Generally, in local search algorithms some or all individuals in the population are improved by some local search method. Local search algorithms are basically designed to incorporate a local search strategy between iterations of a population based search algorithm. In this way, the population based global search algorithms are hybridized with local search algorithms and the hybridized algorithms named as memetic algorithms. In memetic algorithms, the global search capability of the main algorithm explore the search space, trying to identify the most promising search space regions while the local search part scrutinizes the surroundings of some initial solutions, exploiting it in this way.

In this paper, we are proposing a local search strategy inspired by Lévy Flight random walk and named Lévy Flight local search (LFLS). In past, the flight behavior of many animals and insects has been analyzed in various studies which exhibit the important properties of Lévy flights [4,21,25,35]. Further, this flight behavior has been applied to optimization and search algorithms, and the reported results show its importance in the field of solution search algorithms [21,25,27,28]. Recently, Xin-She Yang proposed a new metaheuristic algorithm by combining Lévy flights with the search strategy via the Firefly Algorithm [36].

The Lévy Flight is a random walk in which the steps are defined in terms of the step-lengths, which have a certain probability distribution. The random step lengths are drawn from a Lévy distribution which is defined in Eq. (5):

$$L(s) \sim |s|^{-1-\beta}, \quad \text{where } \beta(0 < \beta \leq 2) \text{ is an index and } s \text{ is the step length.} \tag{5}$$

In this paper, a Mantegna algorithm [37] for a symmetric Lévy stable distribution is used for generating random step sizes. Here 'symmetric' means that the step size may be positive or negative.

In Mantega's algorithm, the step length s can be calculated by

$$s = \frac{u}{|v|^{1/\beta}}, \tag{6}$$

where, u and v are drawn from normal distributions. That is

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

where,

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

This distribution (for s) obeys the expected lévy distribution for $|s| \geq |s_0|$, where s_0 is the smallest step length [37]. Here $\Gamma(.)$ is the Gamma function and calculated as follows:

$$\Gamma(1+\beta) = \int_0^\infty t^\beta e^{-t} dt. \tag{9}$$

In a special case when β is an integer, then we have $\Gamma(1+\beta) = \beta!$.

In the proposed strategy, the step sizes are generated using lévy distribution to exploit the search area and calculated as follows:

$$step\_size(t) = 0.001 \times s(t) \times SLC, \tag{10}$$

here t is the iteration counter for local search strategy, $s(t)$ is calculated using lévy distribution as shown in Eq. (6) and SLC is the social learning component of the global search algorithm.

In Levy flights, the step sizes are too aggressive, that is, they may generate new solutions often outside the domain or on boundary. Since, the local search algorithms can be seen as a population based stochastic algorithms, where main task is to exploit the available knowledge about a problem and steps sizes play an important role in exploiting the identified region. Therefore, 0.001 multiplier is used in Eq. (10) to reduce the step size. The solution update equation of an ith individual, based on the proposed local search strategy is given in Eq. (11):

$$x'_{ij}(t+1) = x_{ij}(t) + step\_size(t) \times U(0,1), \tag{11}$$

here x_{ij} is the individual which is going to modify its position, $U(0, 1)$ is a uniformly distributed random number between 0 and 1 and $step\_size(t) \times U(0, 1)$ is the actual random walks or flights drawn from lévy distribution. The pseudo-code of the proposed LFLS is shown in Algorithm 2. In Algorithm 2, ϵ determines the termination of local search.

Algorithm 2 Lévy Flight Local Search Strategy:

```
Input optimization function Minf(x) and β;
Select an individual x_i in the swarm which is going to modify
its position ;
Initialize t = 1 and σ_v = 1;
Compute σ_u using equation (8);
while (t < ε) do
   Compute step_size using equation (10);
   Generate a new solution x'_i using equation (11);
   Calculate f(x'_i);
   if f(x'_i) < f(x_i) then
      x_i = x'_i;
   end if
   t = t + 1;
end while
Return the x_i;
```

4 Opposition based learning (OBL)

Nature inspired algorithms (NIAs) start searching of the global optima randomly and all the individual randomly initialized in the given search space. Further, the individuals update their positions based on self intelligence and social learning and move towards the optimum solution or required solution. NIAs are iterative and stochastic in nature and solution search process terminates when some predefined criteria is satisfied. The performance of the algorithms are measured in terms of computational complexity which is directly proportionate to the number of objective function evaluations to be optimized. Researchers are continuously working to improve the performance of NIAs in terms of computational complexity. Rahnamayan et al. [24] proposed opposition based differential evolution (OBDE) to improve the convergence rate of DE which was based on the theory of opposite-based learning (OBL). The concept of OBL was introduced by Tizhoosh [31]. The same concept can also be applied in ABC to improve the convergence speed. If the randomized initialized solutions are near to the global optima then the required solution can be found in less computational efforts but if the solutions are very far form the optima then it takes high computational cost or may be some time infeasible to track the required solution. As suggested by Tizhoosh [31] and further by Rahnamayan et al. [24], the computational cost can be reduced by applying the concept of OBL. In OBL we consider the better solutions as well as their opposite counter part solutions and combined them, then we apply greedy approach to find the fittest solutions among them. We judge the solution on the basis of its fitness in respect to global optima. Let x be a solution and $\tilde{x}$ is its opposite solution in the search space. On the basis of probability theory, we can say that there is 50 % chance that the opposite solution may be fitter. The concept of OBL can be defined as follows [31]:

Definition 1 *(Opposite Number)* Let $x \in \Re$ be a real number defined on a certain interval: $x \in [a, b]$. The opposite number $\tilde{x}$ is defined as follows

$$\tilde{x} = a + b - x$$

For the higher dimensions, the above definition can be extended as follows [24,31]:

Definition 2 *(Opposite Point)* Let $P(x_1, x_2, \ldots, x_D)$ be a point in a D-dimensional coordinate system with $x_1, \ldots, x_D \in \Re$ and $x_i \in [a_i, b_i]$. The opposite point $\tilde{P}$ is completely defined by its coordinates $x_1, \ldots, x_D$ where

$$\tilde{x_i} = a_i + b_i - x_i, \quad i = 1, \ldots, D.$$

On the basis of the Definition 2, an Opposition Based Optimization Method ($OBOM$) has been developed [24]. The pseudo code of the $OBOM$ is shown in Algorithm 3:

Algorithm 3 Opposition Based Optimization Method (OBOM):

```
Input optimization function f(x) and Swarm P of size N_p;
Generate a new swarm OP having N_p new opposite solutions as follows:
for i = 1 to N_p do
   for j = 1 to D do
      OP_ij = a_j + b_j − P_ij;
   end for
end for
Combine the swarm OP and P .
Calculate fitness of the swarm {OP, P} using optimization function f;
Return the fittest N_p solutions.
```

5 Opposition based lévy flight ABC (OBLFABC)

Exploration and exploitation are the two important characteristics of the population-based optimization algorithms such as GA [10], PSO [16], DE [29], BFO [20] and so on. In these optimization algorithms, the exploration refers to the ability to investigate the various unknown regions in the solution space to discover the global optimum. While, the exploitation refers to the ability to apply the knowledge of the previous good solutions to find better solutions. In practice, the exploration and exploitation contradict with each other, and in order to achieve better optimization performance, the two abilities should be well balanced. Dervis Karaboga and Bahriye Akay [14] tested different variants of ABC for global optimization and found that the ABC shows poor performance and remains inefficient in exploring the search space. In ABC, any potential solution updates itself using the information provided by a randomly selected potential solution within the current swarm. In this process, a step size which is a linear combination of a random number $\phi_{ij} \in [-1, 1]$, current solution and a randomly selected solution are used. Now the quality of the updated solution highly depends upon this step size. If the step size is too large, which may occur if the difference of current solution and randomly selected solution is large with high absolute value of ϕ_{ij}, then updated solution can surpass the true solution and if this step size is too small then the convergence rate of ABC may significantly decrease. A proper balance of this step size can balance the exploration and exploitation capability of the ABC simultaneously. But, since this step size consists of random component so the balance can not be done manually. Therefore, in this paper, to balance the diversity and convergence ability of ABC, three modifications are proposed:

1. To enhance the exploitation capability of ABC, Lévy Flight Local Search (LFLS) is incorporated with the basic ABC. In this way, the situation of skipping true solution can be avoided while maintaining the speed of convergence. The LFLS strategy, in case of large step sizes, can search within the area that is jumped by the basic ABC.
2. Inspired by the $OBDE$ [24], to balance the diversity and convergence capability of ABC, the OBL strategy is incorporated with the basic ABC. This modification avoids situation of stagnation of the algorithm and speed up the convergence to global optima.
3. In the basic ABC, the food sources are updated, as shown in Eq. (2), based on the random step size. Inspired by PSO [16] and Gbest-guided ABC (GABC) [38] algorithms which, in order to improve the exploitation, take advantage of the information of the global best solution to guide the search of candidate solutions, the solution search equation described by Eq. (2) is modified as follows [38]:

$$v_{ij} = x_{ij} + \phi_{ij}(x_{ij} - x_{kj}) + \psi_{ij}(x_{bestj} - x_{ij}),$$

here, ψ_{ij} is a uniform random number in $[0, C]$, where C is a nonnegative constant. For details description refer to [38].

As described in the proposed first modification, a Lévy Flight random walk inspired local search is incorporated with the basic ABC to improve the exploitation capability. In the proposed local search strategy, step size is calculated as shown in Eq. (12).

$$step\_size(t) = 0.001 \times s(t) \times (x_{bestj}(t) - x_{kj}(t)), \quad (12)$$

here, symbols have their usual meanings, $SLC = (x_{bestj} - x_{kj})$ is the social learning component of the ABC algorithm in which x_{best} is the best solution in the current swarm and x_k is the randomly selected solution within swarm and $x_k \neq x_{best}$. The solution update equation of the best individual within the current swarm, based on the proposed local search strategy is given in Eq. (13):

$$x'_{bestj}(t+1) = x_{bestj}(t) + step\_size(t) \times U(0, 1), \quad (13)$$

In LFLS, only the best particle of the current swarm updates itself in its neighborhood. The pseudo-code of the

proposed Lévy Flight search strategy with ABC is shown in Algorithm 4 and

Algorithm 4 Lévy Flight Search Strategy with ABC:

```
Input optimization function Minf(x) and β;
Select the best solution x_best in the swarm;
Initialize t = 1 and σ_v = 1;
Compute σ_u using equation (8);
while (t < ϵ) do
  Compute step_size(t) using equation (12);
  for j = 1 to D do
    if U(0,1) > p_r then
      x'_bestj = x_bestj + step_size(t) × U(0,1);
    else
      x'_bestj = x_bestj;
    end if
  end for
  Calculate f(x'_best);
  if f(x'_best) < f(x_best) then
    x_best = x'_best;
  end if
  t = t + 1;
end while
Return the x_best;
```

In Algorithm 4, ϵ is the termination criteria of the proposed local search, p_r is a perturbation rate (a number between 0 and 1) which controls the amount of perturbation in the best solution, $U(0, 1)$ is a uniform distributed random number between 0 and 1, D is the dimension of the problem and x_k is a randomly selected solution within swarm. See Sect. 6.2 for details of these parameter settings.

Further, the opposition based learning (OBL) strategy, explained in Sect. 4, is incorporated with the ABC to balance the diversity and the convergence capability. The OBL is applied to the current swarm of the ABC after the scout bee phase in the solution search process on the basis of probability named as jumping rate (j_r) [24]. Unlike the opposition based optimization method ($OBOM$) explained in Algorithm 3, opposition based solutions are generated dynamically. In the $OBOM$, evolution of opposition based solutions are in the static range (solution search space). Therefore, there is a chance to jump outside of the already shrunken search space and the knowledge of the current reduced space (converged swarm) would be lost. Hence, Instead of using predefined search range ($[a_j, b_j]$), the solutions are generated by using current interval in the swarm which is, as the search does progress, increasingly smaller than the corresponding initial range. Therefore, in OBOM, the opposition based solutions are generated using following equation:

$$OP_{ij} = {MIN_J}^P + {MAX_j}^P - P_{ij},$$

here, $[{MIN_J}^P, {MAX_j}^P]$ is the current interval in the swarm.

Therefore, $LFLS$ and $OBOM$ are incorporated with the basic ABC to speed up the evolutionary process (search process). The proposed algorithm is named as Opposition Based Lévy Flight ABC (OBLFABC). Pseudo-code of the OBLFABC is shown in Algorithm 5:

Algorithm 5 Opposition Based Lévy Flight ABC (OBLFABC):

```
Initialize the parameters;
while Termination criteria do
  Step 1: Employed bee phase for generating new food sources.
  Step 2: Onlooker bees phase for updating the food sources depending on their
          nectar amounts.
  Step 3: Scout bee phase for discovering the new food sources in place of
          abandoned food sources.
  Step 4: Apply Lévy Flight Local Search strategy (LFLS) using Algorithm 4
  if j_r > rand(0,1) then
    Apply the Opposition based optimization method (OBOM) using Algorithm 3;
  end if
end while
Print best solution.
```

6 Experimental results and discussion

6.1 Test problems under consideration

In order to analyze the performance of OBLFABC, 14 un-biased different global optimization problems (f_1 to f_{14}) are selected (listed in Table 1). These are continuous optimization problems and have different degrees of complexity and multimodality. Test problems f_1 to f_5 and f_{11} to f_{14} are taken from [2] while test problems f_6 to f_{10} are taken from [30] with the associated offset values. Further, to see the robustness of the proposed strategy, five well known engineering optimization problems (f_{15} to f_{19}) which are described as follows, have been solved.

Compression Spring (f_{15}): The considered first engineering optimization problem is compression spring problem [19, 26]. This problem minimizes the weight of a compression spring, subject to constraints of minimum deflection, shear stress, surge frequency, and limits on outside diameter and on design variables. There are three design variables: the wire diameter x_1, the mean coil diameter x_2, and the number of active coils x_3. This is a simplified version of a more difficult problem. The mathematical formulation of this problem is:

$x_1 \in \{1, \ldots, 70\}$ granularity 1

$x_2 \in [0.6; 3]$

$x_3 \in [0.207; 0.5]$ granularity 0.001

and four constraints

$$g_1 := \frac{8C_f F_{max} x_2}{\pi x_3^3} - S \leq 0$$

$$g_2 := l_f - l_{max} \leq 0$$

$$g_3 := \sigma_p - \sigma_{pm} \leq 0$$

$$g_4 := \sigma_w - \frac{F_m ax - F_p}{K} \leq 0$$

with

$$C_f = 1 + 0.75\frac{x_3}{x_2 - x_3} + 0.615\frac{x_3}{x_2}$$

$$F_{max} = 1{,}000$$

$$S = 189{,}000$$

Table 1 Test problems

Test problem	Objective function	Search range	Optimum value	D	Acceptable error
Beale function	$f_1(x) = [1.5 - x_1(1 - x_2)]^2 + [2.25 - x_1(1 - x_2^2)]^2 + [2.625 - x_1(1 - x_2^3)]^2$	$[-4.5, 4.5]$	$f(3, 0.5) = 0$	2	1.0E−05
Colville function	$f_2(x) = 100[x_2 - x_1^2]^2 + (1 - x_1)^2 + 90(x_4 - x_3^2)^2 + (1 - x_3)^2 + 10.1[(x_2 - 1)^2 + (x_4 - 1)^2] + 19.8(x_2 - 1)(x_4 - 1)$	$[-10, 10]$	$f(\mathbf{1}) = 0$	4	1.0E−05
Braninss function	$f_3(x) = a(x_2 - bx_1^2 + cx_1 - d)^2 + e(1 - f)\cos x_1 + e$	$x_1 \in [-5, 10], x_2 \in [0, 15]$	$f(-\pi, 12.275) = 0.3979$	2	1.0E−05
Kowalik function	$f_4(x) = \sum_{i=1}^{11} \left[a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4} \right]^2$	$[-5, 5]$	$f(0.1928, 0.1908, 0.1231, 0.1357) = 3.07\text{E}{-04}$	4	1.0E−05
2D tripod	$f_5(x) = p(x_2)(1 + p(x_1)) + \lvert(x_1 + 50p(x_2)(1 - 2p(x_1)))\rvert + \lvert(x_2 + 50(1 - 2p(x_2)))\rvert$	$[-100, 100]$	$f(0, -50) = 0$	2	1.0E−04
Shifted rosenbrock	$f_6(x) = \sum_{i=1}^{D-1}(100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2) + f_{bias}, z = x - o + 1, x = [x_1, x_2, ...x_D], o = [o_1, o_2, ...o_D]$	$[-100, 100]$	$f(o) = f_{bias} = 390$	10	1.0E−01
Shifted sphere	$f_7(x) = \sum_{i=1}^{D} z_i^2 + f_{bias}, z = x - o, x = [x_1, x_2, ...x_D], o = [o_1, o_2, ...o_D]$	$[-100, 100]$	$f(o) = f_{bias} = -450$	10	1.0E−05
Shifted rastrigin	$f_8(x) = \sum_{i=1}^{D}(z_i^2 - 10\cos(2\pi z_i) + 10) + f_{bias}$ $z = (x - o), x = (x_1, x_2, ...x_D), o = (o_1, o_2, ...o_D)$	$[-5, 5]$	$f(o) = f_{bias} = -330$	10	1.0E−02
Shifted griewank	$f_9(x) = \sum_{i=1}^{D} \frac{z_i^2}{4000} - \prod_{i=1}^{D} \cos(\frac{z_i}{\sqrt{i}}) + 1 + f_{bias}, z = (x - o), x = [x_1, x_2, ...x_D], o = [o_1, o_2, ...o_D]$	$[-600, 600]$	$f(o) = f_{bias} = -180$	10	1.0E−05
Shifted ackley	$f_{10}(x) = -20\exp(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} z_i^2}) - \exp(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi z_i)) + 20 + e + f_{bias}, z = (x - o), x = (x_1, x_2, ...x_D), o = (o_1, o_2, ...o_D)$	$[-32, 32]$	$f(o) = f_{bias} = -140$	10	1.0E−05
Goldstein-price	$f_{11}(x) = (1 + (x_1 + x_2 + 1)^2 \cdot (19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2))(30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2))$	$[-2, 2]$	$f(0, -1) = 3$	2	1.0E−14
McCormick	$f_{12}(x) = \sin(x_1 + x_2) + (x_1 - x_2)^2 - \frac{3}{2}x_1 + \frac{5}{2}x_2 + 1$	$x_1 \in [-1.5, 4], x_2 \in [-3, 3]$	$f(-0.547, -1.547) = -1.9133$	30	1.0E−04
Meyer and roth	$f_{13}(x) = \sum_{i=1}^{5}\left(\frac{x_1 x_3 t_i}{1 + x_1 t_i + x_2 v_i} - y_i\right)^2$	$[-10, 10]$	$f(3.13, 15.16, 0.78) = 0.4\text{E}{-04}$	3	1.0E−03
Shubert	$f_{14}(x) = -\sum_{i=1}^{5} i\cos((i+1)x_1+1)\sum_{i=1}^{5} i\cos((i+1)x_2 + 1)$	$[-10, 10]$	$f(7.0835, 4.8580) = -186.7309$	2	1.0E−05

$$l_f = \frac{F_{max}}{K} + 1.05(x_1 + 2)x_3$$
$$l_{max} = 14$$
$$\sigma_p = \frac{F_p}{K}$$
$$\sigma_{pm} = 6$$
$$F_p = 300$$
$$K = 11.5 \times 10^6 \frac{x_3^4}{8x_1x_2^3}$$
$$\sigma_w = 1.25$$

and the function to be minimized is

$$f_{15}(\mathbf{x}) = \pi^2 \frac{x_2 x_3^2 (x_1 + 2)}{4}$$

The best known solution is (7, 1.386599591, 0.292), which gives the fitness value $f^* = 2.6254214578$. Here, minimum error is fixed to be 10^{-10}, i.e. a run is said to be successful if it finds a fitness f so that $|f - f^*| \leq 10^{-10}$ in the maximum number of function evaluations.

Pressure Vessel design (f_{16}) The pressure vessel design is to minimize the total cost of the material, forming, and welding of a cylindrical vessel [33]. There are four design variables involved: x_1, (T_s, shell thickness), x_2 (T_h, spherical head thickness), x_3 (R, radius of cylindrical shell), and x_4 (L, shell length). The mathematical formulation of this typical constrained optimization problem is as follows:

$$f_{16}(\mathbf{x}) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1611x_1^2x_4 + 19.84x_1^2x_3$$

subject to

$$g_1(\mathbf{x}) = 0.0193x_3 - x_1$$
$$g_2(\mathbf{x}) = 0.00954x_3 - x_2$$
$$g_3(\mathbf{x}) = 750 \times 1728 - \pi x_3^2 \left(x_4 + \frac{4}{3}x_3\right)$$

The search boundaries for the variables are

$$1.125 \leq x_1 \leq 12.5,$$
$$0.625 \leq x_2 \leq 12.5,$$
$$1.0 \times 10^{-8} \leq x_3 \leq 240$$

and

$$1.0 \times 10^{-8} \leq x_4 \leq 240.$$

The best known global optimum solution is $f(1.125, 0.625, 55.8592, 57.7315) = 7197.729$ [33]. For a successful run, the minimum error criteria is fixed to be 1.0E−05 i.e. an algorithm is considered successful if it finds the error less than acceptable error in a specified maximum function evaluations.

Lennard-Jones (f_{17}) The function to minimize is a kind of potential energy of a set of N atoms. The position x_i of the atom i has three coordinates, and therefore the dimension of the search space is $3N$. In practice, the coordinates of a point x are the concatenation of the ones of the x_i. In short, we can write $x = (x_1, x_2, \ldots, x_N)$, and we have then

$$f_{17}(\mathbf{x}) = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \left(\frac{1}{\|x_i - x_j\|^{2\alpha}} - \frac{1}{\|x_i - x_j\|^{\alpha}} \right)$$

In this study $N = 5, \alpha = 6$, and the search space is $[-2, 2]$ [5].

Parameter Estimation for Frequency-Modulated (FM) [5] (f_{18}): Frequency-Modulated (FM) sound wave synthesis has an important role in several modern music systems. The parameter optimization of an FM synthesizer is a six dimensional optimization problem where the vector to be optimized is $\mathbf{x} = \{a_1, w_1, a_2, w_2, a_3, w_3\}$ of the sound wave given in Eq. (14). The problem is to generate a sound (1) similar to target (2). This problem is a highly complex multimodal one having strong epistasis, with minimum value $f(\mathbf{x}) = 0$. This problem has been tackled using genetic algorithms (GAs) in [1], [2].The expressions for the estimated sound and the target sound waves are given as:

$$y(t) = a_1 \sin(w_1 t\theta + a_2 \sin(w_2 t\theta + a_3 \sin(w_3 t\theta))) \quad (14)$$
$$y_0(t) = (1.0)\sin((5.0)t\theta - (1.5)\sin((4.8)t\theta + (2.0)\sin((4.9)t\theta))) \quad (15)$$

respectively where $\theta = 2\pi/100$ and the parameters are defined in the range [-6.4, 6.35]. The fitness function is the summation of square errors between the estimated wave (1) and the target wave (2) as follows:

$$f_{18}(\mathbf{x}) = \sum_{i=0}^{100} (y(t) - y_0(t))^2$$

Acceptable error for this problem is 1.0E−05, i.e. an algorithm is considered successful if it finds the error less than acceptable error in a given number of generations.

Welded beam design optimization problem (f_{19}) The problem is to design a welded beam for minimum cost, subject to some constraints [17,23]. The objective is to find the minimum fabricating cost of the welded beam subject to constraints on shear stress τ, bending stress σ, buckling load P_c, end deflection δ, and side constraint. There are four design variables: x_1, x_2, x_3 and x_4. The mathematical formulation of the objective function is described as follows:

$$f_{19}(\mathbf{x}) = 1.10471x_1^2x_2 + 0.04811x_3x_4(14.0 + x_2)$$

Subject to:

$$g_1(\mathbf{x}) = \tau(\mathbf{x}) - \tau_{max} \leq 0$$
$$g_2(\mathbf{x}) = \sigma(\mathbf{x}) - \sigma_{max} \leq 0$$
$$g_3(\mathbf{x}) = x_1 - x_4 \leq 0$$
$$g_4(\mathbf{x}) = \delta(\mathbf{x}) - \delta_{max} \leq 0$$
$$g_5(\mathbf{x}) = P - P_c(\mathbf{x}) \leq 0$$
$$0.125 \leq x_1 \leq 5, 0.1 \leq x_2, x_3 \leq 10 \text{ and } 0.1 \leq x_4 \leq 5$$

where

$$\tau(\mathbf{x}) = \sqrt{\tau'^2 - \tau'\tau''\frac{x_2}{R} + \tau''^2},$$
$$\tau' = \frac{P}{\sqrt{2}x_1x_2}, \quad \tau'' = \frac{MR}{J}, \quad M = P\left(L + \frac{x_2}{2}\right),$$
$$R = \sqrt{\frac{{x_2}^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2},$$
$$J = 2/\left(\sqrt{2}x_1x_2\left[\frac{{x_2}^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2\right]\right),$$
$$\sigma(\mathbf{x}) = \frac{6PL}{x_4{x_3}^2}, \quad \delta(\mathbf{x}) = \frac{6PL^3}{Ex_4{x_3}^2},$$
$$P_c(\mathbf{x}) = \frac{4.013Ex_3{x_4}^3}{6L^2}\left(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right),$$
$$P = 6{,}000\ lb, \quad L = 14\ in., \quad \delta_{max} = 0.25\ in.,$$
$$\sigma_{max} = 30{,}000\ psi, \quad \tau_{max} = 13600\ psi,$$
$$E = 30 \times 10^6\ psi, \quad G = 12 \times 10^6\ psi.$$

The best known solution is (0.205730, 3.470489, 9.036624, 0.205729) , which gives the function value 1.724852. Acceptable error for this problem is 1.0E−01.

6.2 Experimental setting

To prove the efficiency of $OBLFABC$, it is compared with ABC and recent variants of ABC named Gbest-guided ABC ($GABC$) [38], Best-So-Far ABC ($BSFABC$) [3] and Modified ABC ($MABC$) [1]. To test $OBLFABC$, ABC, $GABC$, $BSFABC$ and $MABC$ over considered problems, following experimental setting is adopted:

- Colony size $NP = 50$ [7,9],
- $\phi_{ij} = rand[-1, 1]$,
- Number of food sources $SN = NP/2$,
- $limit = 1{,}500$ [1,15],
- The stopping criteria is either maximum number of function evaluations (which is set to be 200000) is reached or the acceptable error (mentioned in Table 1) has been achieved,
- The number of simulations/run =100,
- $C = 1.5$ [38],
- The value of $\beta = 2$ is to be set based on the empirical experiments.
- To set termination criteria of LFLS, the performance of OBLFABC is measured for considered test problems on different values of ϵ and results are analyzed in Fig. 1a. It is clear from Fig. 1a that $\epsilon = 15$ gives better results. Therefore, termination criteria is set to be $\epsilon = 15$.

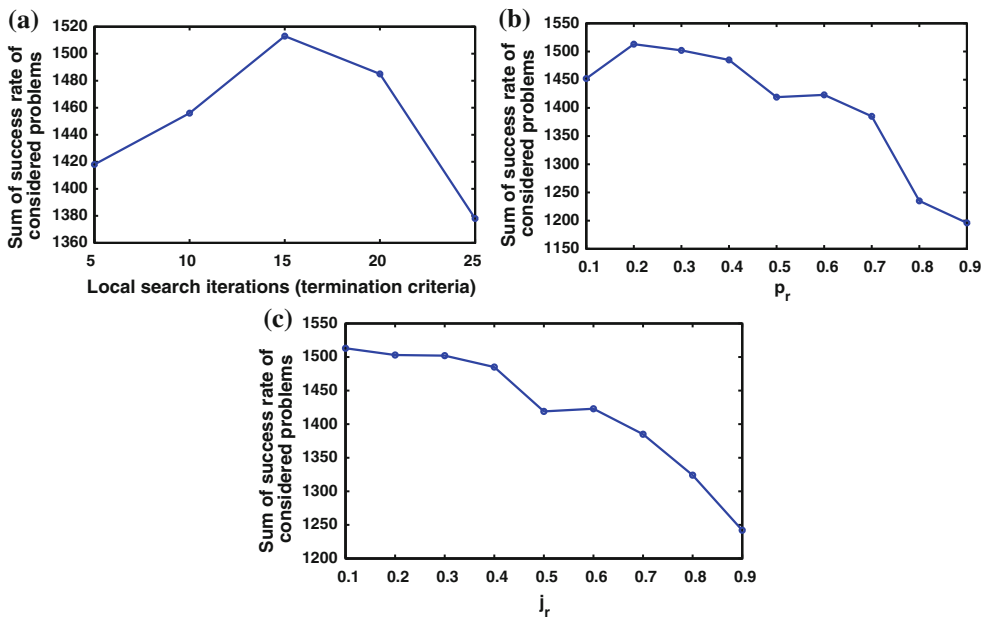

Fig. 1 **a** Effect of LFLS termination criteria ϵ on success rate, **b** effect of parameter p_r on success rate and **c** effect of parameter j_r on success rate

Table 2 Comparison of the results of test problems

Test function	Algorithm	SD	ME	AFE	SR
f_1	OBLFABC	2.87E−06	7.55E−06	7,465.64	100
	ABC	1.66E−06	8.64E−06	16,520.09	100
	GABC	3.05E−06	5.03E−06	9,314.71	100
	BSFABC	5.64E−05	1.98E−05	47,522.01	95
	MABC	2.68E−06	5.47E−06	10,350.53	100
f_2	OBLFABC	1.23E−02	1.46E−02	115,073.86	68
	ABC	1.03E−01	1.67E−01	199,254.48	1
	GABC	1.71E−02	1.95E−02	151,300.35	46
	BSFABC	1.61E−02	1.86E−02	153,393.46	47
	MABC	8.26E−03	1.25E−02	147,787.15	52
f_3	OBLFABC	6.67E−06	5.93E−06	988.51	100
	ABC	6.83E−06	6.05E−06	1,925.52	100
	GABC	6.54E−06	5.76E−06	1,204.65	100
	BSFABC	6.99E−06	5.87E−06	25,770.82	88
	MABC	6.49E−06	5.71E−06	28,716.34	87
f_4	OBLFABC	1.21E−05	9.47E−05	56,381.43	100
	ABC	7.33E−05	1.76E−04	180,578.91	18
	GABC	2.15E−05	8.68E−05	90,834.53	97
	BSFABC	7.57E−05	1.41E−04	147,931.24	50
	MABC	8.02E−05	2.02E−04	187,320.13	13
f_5	OBLFABC	2.53E−05	6.79E−05	5,184.77	100
	ABC	2.69E−05	6.42E−05	8,771.65	100
	GABC	2.58E−05	6.34E−05	7,305.93	100
	BSFABC	1.55E−03	2.49E−04	8,453.82	97
	MABC	1.49E−04	1.06E−04	67,336.12	88
f_6	OBLFABC	7.15E+00	1.44E+00	75,645.49	85
	ABC	1.05E+00	6.36E−01	176,098.02	23
	GABC	1.60E−02	8.45E−02	99,219.48	99
	BSFABC	3.79E+00	2.34E+00	179,970.99	19
	MABC	9.19E−01	6.99E−01	180,961.73	23
f_7	OBLFABC	2.64E−06	7.31E−06	6,693.71	100
	ABC	2.42E−06	7.16E−06	9,013.5	100
	GABC	2.08E−06	6.83E−06	5,585.5	100
	BSFABC	2.18E−06	7.44E−06	18,122	100
	MABC	1.61E−06	8.23E−06	8,702	100
f_8	OBLFABC	1.07E+01	9.13E+01	200,034.54	0
	ABC	1.21E+01	8.91E+01	200,011.71	0
	GABC	9.24E+00	8.56E+01	200,006.8	0
	BSFABC	1.77E+01	1.20E+02	200,036.53	0
	MABC	1.15E+01	8.00E+01	200,015.14	0
f_9	OBLFABC	3.73E−03	1.71E−03	81,595.29	82
	ABC	2.21E−03	6.95E−04	61,650.9	90
	GABC	7.35E−04	7.88E−05	38,328.96	99
	BSFABC	6.34E−03	4.76E−03	115,441.96	58
	MABC	2.21E−03	6.24E−04	85,853.52	92
f_{10}	OBLFABC	1.63E−06	8.27E−06	11,675.15	100
	ABC	1.80E−06	7.90E−06	16,767	100
	GABC	1.37E−06	8.31E−06	9,366	100

Table 2 continued

Test function	Algorithm	SD	ME	AFE	SR
	BSFABC	1.35E−06	8.39E−06	31,224	100
	MABC	9.96E−07	8.93E−06	14,189.06	100
f_{11}	OBLFABC	4.71E−15	5.68E−15	4,296.23	100
	ABC	5.16E−06	1.04E−06	109,879.46	62
	GABC	4.37E−15	4.87E−15	3,956.05	100
	BSFABC	4.90E−15	6.62E−15	14,031.79	100
	MABC	4.11E−15	4.73E−15	14,228.59	100
f_{12}	OBLFABC	6.96E−06	8.91E−05	662.79	100
	ABC	6.67E−06	8.92E−05	1,166.5	100
	GABC	6.45E−06	8.79E−05	622	100
	BSFABC	6.34E−06	8.91E−05	958.51	100
	MABC	6.15E−06	8.95E−05	1,702.28	100
f_{13}	OBLFABC	3.03E−06	1.95E−03	5,229.73	100
	ABC	2.89E−06	1.94E−03	24,476.88	100
	GABC	2.74E−06	1.95E−03	5,127.73	100
	BSFABC	2.98E−06	1.94E−03	15,703.99	100
	MABC	2.79E−06	1.95E−03	9,019.7	100
f_{14}	OBLFABC	5.89E−06	5.27E−06	2,255.04	100
	ABC	5.34E−06	4.86E−06	4,752.21	100
	GABC	5.72E−06	5.07E−06	2,550.57	100
	BSFABC	5.94E−06	5.27E−06	9,036.83	100
	MABC	5.60E−06	4.83E−06	33,268.91	100
f_{15}	OBLFABC	7.94E−03	5.21E−03	126,520.99	58
	ABC	1.22E−02	1.42E−02	189,423.57	10
	GABC	1.08E−02	1.09E−02	188,220.68	12
	BSFABC	5.73E−03	2.93E−02	196,442.59	2
	MABC	6.65E−03	4.98E−03	174,943.41	25
f_{16}	OBLFABC	2.43E+00	1.51E+00	200,030.62	0
	ABC	1.07E+01	1.69E+01	200,024.49	0
	GABC	4.47E+00	6.84E+00	200,023.29	0
	BSFABC	2.19E+01	2.66E+01	200,036.19	0
	MABC	8.63E+00	1.51E+01	200,025.82	0
f_{17}	OBLFABC	1.05E−04	8.91E−04	22,077.03	100
	ABC	1.24E−04	8.77E−04	70,189.49	100
	GABC	5.26E−04	1.11E−03	112,535.45	69
	BSFABC	6.13E−04	9.46E−04	154,352.33	89
	MABC	1.51E−01	4.66E−01	200,031.91	0
f_{18}	OBLFABC	3.81E+00	1.68E+00	158,797.2	44
	ABC	5.08E+00	6.01E+00	200,032.9	0
	GABC	4.81E+00	3.10E+00	184,843.53	14
	BSFABC	4.44E+00	1.01E+01	198,887.41	1
	MABC	3.06E+00	2.85E+00	200,022.49	0
f_{19}	OBLFABC	1.98E−02	1.06E−01	116,907.25	71
	ABC	8.72E−02	2.47E−01	197,869.95	2
	GABC	1.09E−02	9.95E−02	120,011.32	69
	BSFABC	4.79E−03	9.57E−02	55,029.62	100
	MABC	5.18E−03	9.37E−02	33,617.76	100

Table 3 Summary of Table 2 outcome

Test problems	OBLFABC versus ABC	OBLFABC versus GABC	OBLFABC versus BSFABC	OBLFABC versus MABC
f_1	+	+	+	+
f_2	+	+	+	+
f_3	+	+	+	+
f_4	+	+	+	+
f_5	+	+	+	+
f_6	+	−	+	+
f_7	+	−	+	+
f_8	+	−	+	+
f_9	−	−	+	−
f_{10}	+	−	+	+
f_{11}	+	+	+	+
f_{12}	+	−	+	+
f_{13}	+	−	+	+
f_{14}	+	−	+	+
f_{15}	+	+	+	+
f_{16}	+	+	+	+
f_{17}	+	+	+	+
f_{18}	+	+	+	+
f_{19}	+	+	−	−
Total number of + sign	18	11	18	17

- In order to investigate the effect of the parameter p_r, described by algorithm 4 on the performance of $OBLFABC$, its sensitivity with respect to different values of p_r in the range [0.1, 1], is examined in the Fig. 1b. It can be observed from Fig. 1b that the test problems are very sensitive towards p_r and value 0.2 gives comparatively better results. Therefore $p_r = 0.2$ is selected for the experiments.
- In order to find out the optimal value of the jumping rate constant j_r for all the considered test problems, $OBLFABC$ is executed for different values of j_r in the range [0.1, 0.9]. The sensitivity analysis of j_r is shown in Fig. 1c. It is clear from Fig. 1c that $j_r = 0.1$ gives comparatively better results. Hence, for the experiments, $j_r = 0.1$ is selected.
- Parameter settings for the algorithms GABC, BSFABC and MABC are similar to their original research papers.

6.3 Results comparison

Numerical results with experimental setting of Subsect. 6.2 are given in Table 2. In Table 2, standard deviation (SD), mean error (ME), average function evaluations (AFE) and success rate (SR) are reported. Table 2 shows that most of the time OBLFABC outperforms in terms of reliability, efficiency and accuracy as compare to the basic ABC, GABC, BSFABC and MABC. Some more intensive analyses based on acceleration rate (AR), performance indices and boxplots have been carried out for results of ABC and its variants.

$OBLFABC$, ABC, $GABC$, $BSFABC$, and $MABC$ are compared through SR, ME and AFE in Table 2. First SR is compared for all these algorithms and if it is not possible to distinguish the algorithms based on SR then comparison is made on the basis of AFE. ME is used for comparison if it is not possible on the basis of SR and AFE both. Outcome of this comparison is summarized in Table 3. In Table 3, '+' indicates that the $OBLFABC$ is better than the considered algorithms and '−' indicates that the algorithm is not better or the difference is very small. The last row of Table 3, establishes the superiority of $OBLFABC$ over ABC, $BSFABC$, $MABC$.

Further, we compare the convergence speed of the considered algorithms by measuring the average function evaluations (AFEs). A smaller AFEs means higher convergence speed. In order to minimize the effect of the stochastic nature of the algorithms, the reported function evaluations for each test problem is the average over 100 runs. In order to compare convergence speeds, we use the acceleration rate (AR) which is defined as follows, based on the AFEs for the two algorithms $ALGO$ and OBLFABC:

$$AR = \frac{AFE_{ALGO}}{AFE_{OBLFABC}}, \tag{16}$$

Table 4 Acceleration Rate (AR) of $OBLFABC$ compare to the basic ABC, $GABC$, $BSFABC$ and $MABC$

Test problems	ABC	GABC	BSFABC	MABC
f_1	2.212816316	1.247677359	6.365430157	1.386422329
f_2	1.731535555	1.314810766	1.33300004	1.284280809
f_3	1.947901387	1.218652315	26.07036853	29.05012595
f_4	3.202808265	1.611071766	2.623758213	3.32237281
f_5	1.691810823	1.409113615	1.630510129	12.98729163
f_6	2.327938123	1.311637746	2.379137077	2.392234223
f_7	1.346562669	0.834440094	2.707317765	1.300026443
f_8	0.99988587	0.999861324	1.000009948	0.999903017
f_9	0.755569347	0.469744761	1.414811566	1.052187203
f_{10}	1.436127159	0.802216674	2.674398188	1.215321431
f_{11}	1.497171102	1.487663668	1.552648221	1.382722424
f_{12}	25.57578621	0.920818951	3.266070485	3.311878088
f_{13}	1.759984309	0.938457128	1.446174505	2.568354984
f_{14}	4.680333402	0.980496125	3.002829974	1.724697068
f_{15}	2.107372818	1.131053108	4.00739233	14.7531352
f_{16}	0.999969355	0.999963356	1.000027846	0.999976004
f_{17}	3.17929948	5.097399877	6.991535093	9.060634968
f_{18}	1.259675234	1.164022602	1.252461693	1.259609678
f_{19}	1.692537888	1.026551561	0.470711782	0.28755924

where, $ALGO \in \{$ ABC, GABC, BSFABC, MABC$\}$ and $AR > 1$ means OBLFABC is faster. In order to investigate the AR of the proposed algorithm compare to the basic ABC and its variants, results of Table 2 are analyzed and the value of AR is calculated using Eq. (16). Table 4 shows a clear comparison between $OBLFABC$ and ABC, OBLFABC and GABC, OBLFABC and BSFABC, and OBLFABC and MABC in terms of AR. It is clear from the Table 4 that convergence speed of OBLFABC is faster among all the considered algorithms.

For the purpose of comparison in terms of consolidated performance, boxplot analyses have been carried out for all the considered algorithms. The empirical distribution of data is efficiently represented graphically by the boxplot analysis tool [34]. The boxplots for ABC, OBLFABC, GABC, BSFABC and MABC are shown in Fig. 2. It is clear from this figure that OBLFABC is better than the considered algorithms as interquartile range and median are comparatively low.

Further, to compare the considered algorithms, by giving weighted importance to the success rate, the mean error and the average number of function evaluations, performance indices (PI) are calculated [6]. The values of PI for the ABC, OBLFABC, GABC, BSFABC, and MABC are calculated by using following equations:

$$PI = \frac{1}{N_p}\sum_{i=1}^{N_p}(k_1\alpha_1^i + k_2\alpha_2^i + k_3\alpha_3^i)$$

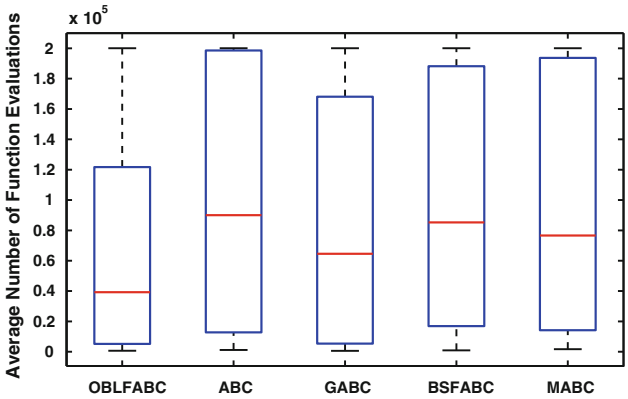

Fig. 2 *Boxplots* graphs for average function evaluation

where $\alpha_1^i = \frac{Sr^i}{Tr^i}$; $\alpha_2^i = \begin{cases} \frac{Mf^i}{Af^i}, & \text{if } Sr^i > 0. \\ 0, & \text{if } Sr^i = 0. \end{cases}$; and $\alpha_3^i = \frac{Mo^i}{Ao^i}$

$i = 1, 2, \ldots, N_p$

- Sr^i = Successful simulations/runs of ith problem.
- Tr^i = Total simulations of ith problem.
- Mf^i = Minimum of average number of function evaluations used for obtaining the required solution of ith problem.
- Af^i = Average number of function evaluations used for obtaining the required solution of ith problem.
- Mo^i = Minimum of mean error obtained for the ith problem.

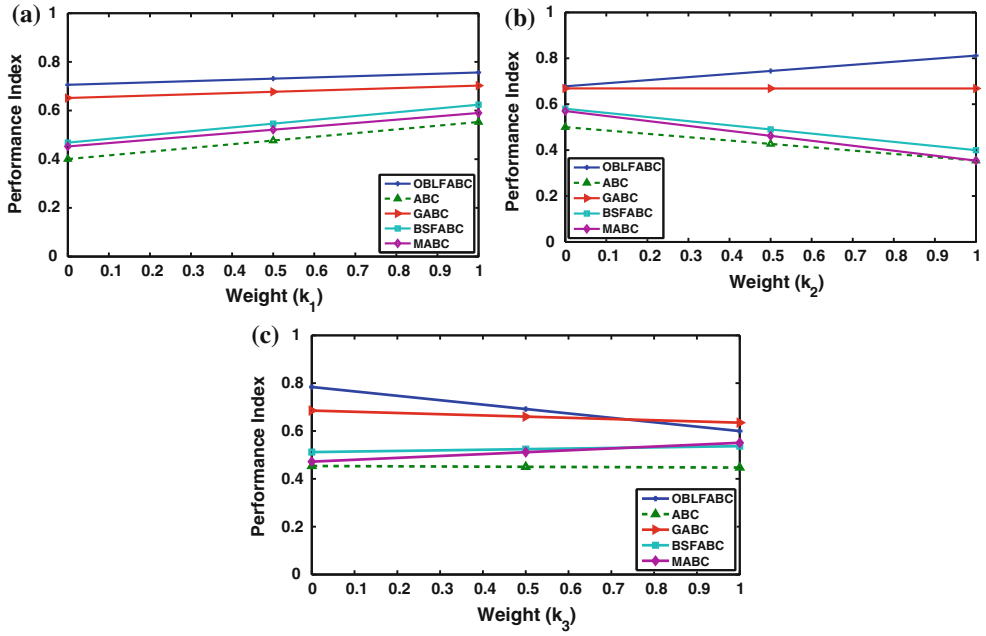

Fig. 3 Performance index for test problems; **a** for case (1), **b** for case (2) and **c** for case (3)

– Ao^i = Mean error obtained by an algorithm for the ith problem.
– N_p = Total number of optimization problems evaluated.

The weights assigned to the success rate, the average number of function evaluations and the mean error are represented by k_1, k_2 and k_3 respectively where $k_1 + k_2 + k_3 = 1$ and $0 \leq k_1, k_2, k_3 \leq 1$. To calculate the PIs, equal weights are assigned to two variables while weight of the remaining variable vary from 0 to 1 as given in [6]. Following are the resultant cases:

1. $k_1 = W, k_2 = k_3 = \frac{1-W}{2}, 0 \leq W \leq 1;$
2. $k_2 = W, k_1 = k_3 = \frac{1-W}{2}, 0 \leq W \leq 1;$
3. $k_3 = W, k_1 = k_2 = \frac{1-W}{2}, 0 \leq W \leq 1$

The graphs corresponding to each of the cases (1), (2) and (3) for ABC, OBLFABC, GABC, BSFABC, and MABC are shown in Figs. 3a–c respectively. In these figures the weights k_1, k_2 and k_3 are represented by horizontal axis while the PI is represented by the vertical axis.

In case (1), average number of function evaluations and the mean error are given equal weights. PIs of the considered algorithms are superimposed in Fig. 3a for comparison of the performance. It is observed that PI of OBLFABC are higher than the considered algorithms. In case (2), equal weights are assigned to the success rate and mean error and in case (3), equal weights are assigned to the success rate and average function evaluations. It is clear from Fig. 3b, c that, the algorithms perform same as in case (1).

7 Conclusion

In this paper, a new local search strategy based on the Lévy Flight random walk is proposed for finding the new solutions around the best solution. In this search strategy, new solutions are generated in the neighborhood of the best solution depending upon perturbation rate. The proposed local search strategy along with opposition based learning (OBL) has been employed to improve the convergence of ABC. The proposed Lévy flight search strategy is used to exploit the search space whereas OBL is used to introduce opposition-based swarm generation to speed up the convergence. Further, inspired by PSO and GABC, a modified position update equation is used to generate the solutions in the search process. By embedding these three modifications within ABC, a new variant of ABC is proposed and named as OBLFABC. Further, the proposed algorithm is compared with the basic ABC, GABC, BSFABC, and MABC through the help of experiments over test problems and shown that the OBLFABC outperforms to the considered algorithms in terms of reliability, efficiency and accuracy.

References

1. Akay B, Karaboga D (2010) A modified artificial bee colony algorithm for real-parameter optimization. Inf Sci, doi:10.1016/j.ins.2010.07.015
2. Ali MM, Khompatraporn C, Zabinsky ZB (2005) A numerical evaluation of several stochastic algorithms on selected continuous global optimization test problems. J Glob Optim 31(4):635–672

3. Banharnsakun A, Achalakul T, Sirinaovakul B (2011) The best-so-far selection in artificial bee colony algorithm. Appl Soft Comput 11(2):2888–2901
4. Brown CT, Liebovitch LS, Glendon R (2007) Lévy flights in dobe ju/hoansi foraging patterns. Hum Ecol 35(1):129–138
5. Das S, Suganthan PN (2010) Problem definitions and evaluation criteria for CEC 2011 competition on testing evolutionary algorithms on real world optimization problems. Jadavpur University, Kolkata, India, and Nangyang Technological University, Singapore, Tech. Rep
6. Thakur M, Deep K (2007) A new crossover operator for real coded genetic algorithms. Appl Math Comput 188(1):895–911
7. Diwold K, Aderhold A, Scheidler A, Middendorf M (2011) Performance evaluation of artificial bee colony optimization and new selection schemes. Memet Comput, 1–14
8. Dorigo M, Di Caro G (1999) Ant colony optimization: a new meta-heuristic. In: Evolutionary computation, 1999. CEC 99. Proceedings of the 1999 congress on, 2. IEEE
9. El-Abd M (2011) Performance assessment of foraging algorithms vs. evolutionary algorithms. Inf Sci 182(1):243–263
10. Goldberg DE (1989) Genetic algorithms in search, optimization, and machine learning. Addison-Wesley, Reading
11. Hooke R, Jeeves TA (1961) "Direct search" solution of numerical and statistical problems. J ACM (JACM) 8(2):212–229
12. Kang F, Li J, Ma Z, Li H (2011) Artificial bee colony algorithm with local search for numerical optimization. J Softw 6(3):490–497
13. Karaboga D (2005) An idea based on honey bee swarm for numerical optimization. Techn. Rep. TR06, Erciyes University Press, Erciyes
14. Karaboga D, Akay B (2009) A comparative study of artificial bee colony algorithm. Appl Math Comput 214(1):108–132
15. Karaboga D, Akay B (2011) A modified artificial bee colony (ABC) algorithm for constrained optimization problems. Appl Soft Comput 11(3):3021–3031
16. Kennedy J, Eberhart R (1995) Particle swarm optimization. In: Neural networks, 1995. Proceedings., IEEE international conference on, 4, pp 1942–1948. IEEE
17. Mahdavi M, Fesanghary M, Damangir E (2007) An improved harmony search algorithm for solving optimization problems. Appl Math Comput 188(2):1567–1579
18. Mezura-Montes E, Velez-Koeppel RE, (2010) Elitist artificial bee colony for constrained real-parameter optimization. In: (2010) Congress on evolutionary computation (CEC 2010). IEEE Service Center, Barcelona, Spain, pp 2068–2075
19. Onwubolu GC, Babu BV (2004) New optimization techniques in engineering. Springer, Berlin
20. Passino KM (2002) Biomimicry of bacterial foraging for distributed optimization and control. IEEE Control Syst Mag 22(3):52–67
21. Pavlyukevich I (2007) Lévy flights, non-local search and simulated annealing. J Comput Phys 226(2):1830–1844
22. Price KV, Storn RM, Lampinen JA (2005) Differential evolution: a practical approach to global optimization. Springer, Berlin
23. Ragsdell KM, Phillips DT (1976) Optimal design of a class of welded structures using geometric programming. ASME J Eng Ind 98(3):1021–1025
24. Rahnamayan S, Tizhoosh HR, Salama MMA (2008) Opposition-based differential evolution. IEEE Trans Evol Comput 12(1):64–79
25. Reynolds AM, Frye MA (2007) Free-flight odor tracking in drosophila is consistent with an optimal intermittent scale-free search. PLoS One 2(4):e354
26. Sandgren E (1990) Nonlinear integer and discrete programming in mechanical design optimization. J Mech Des 112:223
27. Shlesinger MF (2006) Mathematical physics: search research. Nature 443(7109):281–282
28. Shlesinger MF, Zaslavsky GM, Frisch U (1995) Lévy flights and related topics in physics. In Levy flights and related topics in, Physics, vol 450
29. Storn R, Price K (1997) Differential evolution-a simple and efficient adaptive scheme for global optimization over continuous spaces. J Glob Optim 11:341–359
30. Suganthan PN, Hansen N, Liang JJ, Deb K, Chen YP, Auger A, Tiwari S (2005) Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization. In: CEC 2005
31. Tizhoosh HR (2005) Opposition-based learning: a new scheme for machine intelligence. In: Computational intelligence for modelling, control and automation, 2005 and international conference on intelligent agents, web technologies and internet commerce, international conference on, 1, pp 695–701. IEEE
32. Vesterstrom J, Thomsen R (2004) A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems. In: Evolutionary computation, 2004. CEC2004. Congress on, 2, pp 1980–1987. IEEE
33. Wang X, Gao XZ, Ovaska SJ (2008) A simulated annealing-based immune optimization method. In: Proceedings of the international and interdisciplinary conference on adaptive knowledge representation and reasoning, porvoo, Finland, pp 41–47
34. Williamson DF, Parker RA, Kendrick JS (1989) The box plot: a simple visual method to interpret data. Ann Intern Med 110(11):916
35. Yang XS, Deb S (2010) Eagle strategy using lévy walk and firefly algorithms for stochastic optimization. In: Nature inspired cooperative strategies for optimization (NICSO 2010), Springer, vol 284 of studies in, Computational Intelligence, pp 101–111
36. Yang XS (2010) Firefly algorithm, levy flights and global optimization. Research and Development in Intelligent Systems XXVI, pp 209–218
37. Yang XS (2010) Nature-inspired metaheuristic algorithms, 2nd edn. Luniver Press, Beckington
38. Zhu G, Kwong S (2010) Gbest-guided artificial bee colony algorithm for numerical function optimization. Appl Math Comput 217(7):3166–3173