

Lévy flight artificial bee colony algorithm

Harish Sharma^{a,*}, Jagdish Chand Bansal^b, K. V. Arya^a and Xin-She Yang^c

^aDepartment of Information Communication and Technology, ABV-Indian Institute of Information Technology and Management, Gwalior, India; ^bDepartment of Mathematics, South Asian University, New Delhi, India; ^cSchool of Science and Technology, Middlesex University, London, UK

(Received 23 June 2012; accepted 21 April 2013)

Artificial bee colony (ABC) optimisation algorithm is a relatively simple and recent population-based probabilistic approach for global optimisation. 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 ABC, there is a high chance to skip the true solution due to its large step sizes. In order to balance between diversity and convergence in the ABC, a Lévy flight inspired search strategy is proposed and integrated with ABC. The proposed strategy is named as Lévy Flight ABC (LFABC) has both the local and global search capability simultaneously and can be achieved by tuning the Lévy flight parameters and thus automatically tuning the step sizes. In the LFABC, new solutions are generated around the best solution and it helps to enhance the exploitation capability of ABC. Furthermore, to improve the exploration capability, the numbers of scout bees are increased. The experiments on 20 test problems of different complexities and five real-world engineering optimisation problems show that the proposed strategy outperforms the basic ABC and recent variants of ABC, namely, Gbest-guided ABC, best-so-far ABC and modified ABC in most of the experiments.

Keywords: numerical optimisation; swarm intelligence; memetic algorithm; Lévy flight local search

1. Introduction

Swarm intelligence (SI) has become an emerging and interesting area in the field of nature-inspired computing that has been used to solve optimisation problems during the past decade. SI is largely based on the collective behaviour of social creatures. Swarm-based optimisation algorithms find solution by collaborative trial and error process. Social creatures utilise their ability of social learning to solve complex tasks. Peer to peer learning behaviour of social colonies is the main driving force behind the development of many efficient swarm-based optimisation algorithms. Researchers have analysed such behaviours and designed algorithms that can be used to solve nonlinear, nonconvex or discrete optimisation problems. Previous research works (Dorigo & Di Caro, 1999; Kennedy & Eberhart, 1995; Price, Storn, & Lampinen, 2005; Vesterstrom & Thomsen, 2004) have shown that algorithms based on SI have great potential to find solutions of real-world optimisation problems. The algorithms that have emerged in recent years include ant colony optimisation (ACO) (Dorigo & Di Caro, 1999), particle swarm optimisation (PSO) (Kennedy & Eberhart, 1995), bacterial foraging optimisation (BFO) (Passino, 2002), etc.

Artificial bee colony (ABC) optimisation algorithm introduced by Karaboga (2005) is a recent addition in this category. This algorithm is inspired by the behaviour of

honey bees when seeking a quality food source. Like any other population-based optimisation 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 (Karaboga & Akay, 2009). It can be observed that the solution search equation of ABC algorithm is good at exploration but poor at exploitation (Zhu & Kwong, 2010). Therefore, to maintain the proper balance between exploration and exploitation behaviour of ABC, it is highly required to develop a local search (LS) approach in the basic ABC to exploit the search region. In this paper, a LS strategy inspired from Lévy flight random walk is proposed and incorporated with ABC. The proposed strategy is used for finding the global optima of a unimodal and/or multimodal functions by iteratively reducing the step size

*Corresponding author. Email: harish.sharma0107@gmail.com

in updating process of the candidate solution in the search space within which the optima is known to exist. Furthermore, to improve the diversity of the algorithm, numbers of scout bees are increased. The proposed strategy is compared to recent variants of ABC, named, gbest-guided ABC (GABC) algorithm (Zhu & Kwong, 2010), best-so-far ABC (BSFABC) (Banharnsakun, Achalakul, & Sirinaovakul, 2011) and modified ABC (MABC) (Akay & Karaboga, 2012).

Rest of the paper is organised as follows. Section 2 describes a brief review on memetic approach. Basic ABC is explained in Section 3. Lévy flight search strategy (LFSS) is proposed and described in Section 4. In Section 5, LFSS is incorporated with ABC. In Section 6, performance of the proposed strategy is analysed over test problems. In Section 7, five real-world engineering optimisation problems are solved using proposed strategy. Finally, in Section 8, paper is concluded.

2. Brief review on memetic approach

In the field of optimisation, memetic computing is an interesting approach to solve the complex problems (Ong, Lim, & Chen, 2010). Memetic is synonymous to *memes* which can be described as ‘instructions for carrying out behaviour, stored in brains’ (Blackmore, 1999). Memetic computing is defined as ‘... a paradigm that uses the notion of *memes* as units of information encoded in computational representations for the purpose of problem solving’ (Ong et al., 2010). Memetic computing can be seen then as a subject which studies complex structures composed of simple modules (*memes*), which interact and evolve adapting to the problem in order to solve it (Neri, 2012). A good survey on memetic computing can be found in Ong et al. (2010), Neri (2012), and Chen, Ong, Lim, and Tan (2011). Memetic algorithms can be seen as an aspect of the realisation or condition-based subset of memetic computing (Chen et al., 2011). The term ‘memetic algorithm’ (MA) was first presented by Moscato (1989) as a population-based algorithm having local improvement strategy for search of solution. MAs are hybrid search methods that are based on the population-based search framework (Eiben & Smith, 2003; Fogel & Michalewicz, 1997) and neighbourhood-based LS framework (Hoos & Stützle, 2005). Popular examples of population-based methods include genetic algorithms (GAs) and other evolutionary algorithms, while tabu search and simulated annealing are two prominent LS representatives. The main role of MA in evolutionary computing is to provide a LS to establish exploitation of the search space. LS algorithms can be categorised as (Neri, 2012)

- stochastic or deterministic behaviour,
- single solution or multisolution-based search,
- steepest descent or greedy-approach-based selection.

An LS is thought of as an algorithmic structure converging to the closest local optimum, while the global search should have the potential of detecting the global optimum. Therefore, to maintain a proper balance between exploration and exploitation behaviour of an algorithm, it is highly required to incorporate an LS approach in the basic population-based algorithm to exploit the search region.

Generally, population-based search algorithms like GA (Goldberg & Holland, 1988), evolution strategy (Beyer & Schwefel, 2002), differential evolution (DE) (Price et al., 2005), ACO (Dorigo & Di Caro, 1999), PSO (Kennedy, 2006), artificial immune system (Dasgupta, 2006), ABC (Karaboga, 2005), etc. are stochastic in nature (Yang, 2010b). In recent years, researchers hybridised the LS procedures with the population-based algorithms to improve the exploitation capability of the population-based algorithms (Caponio, Neri, & Tirronen, 2009; Ishibuchi, Yoshida, & Murata, 2003; Mininno & Neri, 2010; Neri & Tirronen, 2009; Ong, Nair, & Keane, 2003; Valenzuela & Smith, 2002; Wang, Wang, & Yang, 2009). Furthermore, MAs have been successfully applied to solve a wide range of complex optimisation problems like multiobjective optimisation (Goh, Ong, & Tan, 2009; Knowles, Corne, & Deb, 2008), continuous optimisation (Ong & Keane, 2004; Ong et al., 2003), combinatorial optimisation (Ishibuchi et al., 2003; Repoussis, Tarrantis, & Ioannou, 2009; Tang, Mei, & Yao, 2009), bioinformatics (Gallo, Carballido, & Ponzoni, 2009; Richer, Goëffon, & Hao, 2009), flow shop scheduling (Ishibuchi et al., 2003), scheduling and routing (Brest, Zumer, & Maucec, 2006), machine learning (Caponio, Cascella, Neri, Salvatore, & Sumner, 2007; Ishibuchi & Yamamoto, 2004; Ruiz-Torrubiano & Suárez, 2010), etc.

Ong and Keane (2004) introduced strategies for MAs control that decide at runtime which LS method is to be chosen for the local refinement of the solution. Furthermore, they proposed multiple LS procedures during an MA search in the spirit of Lamarckian learning. Furthermore, Ong, Lim, Zhu, and Wong (2006) described a classification of *memes* adaptation in adaptive MAs on the basis of the mechanism used and the level of historical knowledge on the *memes* employed. Then, the asymptotic convergence properties of the adaptive MAs are analysed according to the classification. Nguyen, Ong, and Lim (2009) presented a novel probabilistic memetic framework that models MAs as a process involving the decision of embracing the separate actions of evolution or individual learning and analysed the probability of each process in locating the global optimum. Furthermore, the framework balances evolution and individual learning by governing the learning intensity of each individual according to the theoretical upper bound derived while the search progresses.

In past, very few efforts have been done to incorporate an LS with ABC. Kang, Li, Ma, and Li (2011) proposed a Hooke Jeeves ABC (HJABC) algorithm for

numerical optimisation. In HJABC, authors incorporated an LS technique which is based on HJ method (Hooke & Jeeves, 1961) with the basic ABC. Furthermore, Mezura-Montes and Velez-Koeppel (2010) introduced a variant of the basic ABC named Elitist ABC. In their work, the authors integrated two LS strategies. The first LS 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 neighbourhood. The other LS 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.

Fister, Fister, and Zumer (2012) proposed a memetic ABC for large-scale global optimisation. In their proposed work, ABC is hybridised with two LS heuristics: the Nelder–Mead algorithm (Rao & Rao, 2009) and the random walk with direction exploitation (Rao & Rao, 2009). The former is attended more towards exploration, while the latter more towards exploitation of the search space. The stochastic adaptive rule as specified by Neri (Cotta & Neri, 2012) is applied for balancing the exploration and exploitation.

Kang et al. (2011) presented a novel hybrid HJABC algorithm with intensification search based on the HJ pattern search and the ABC. In the HJABC, two modifications are proposed, one is the fitness (fit_i) calculation function of basic ABC is changed and calculated by Equation (1) and another is that an HJ LS is incorporated with the basic ABC

$$fit_i = 2 - SP + \frac{2(SP - 1)(p_i - 1)}{NP - 1}, \quad (1)$$

here p_i is the position of the solution in the whole population after ranking, $SP \in [1.0, 2.0]$ is the selection pressure. A medium value of $SP = 1.5$ can be a good choice and NP is the number of solutions.

Furthermore, Kang, Li, and Ma (2011) described a Rosenbrock ABC (RABC) that combines Rosenbrock's rotational direction method with ABC for accurate numerical optimisation. In RABC, exploitation phase is introduced in the ABC using Rosenbrock's rotational direction method.

3. Artificial bee colony (ABC) algorithm

The ABC algorithm is relatively recent SI-based algorithm. The algorithm is inspired by the intelligent food foraging behaviour 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 numbers of employed bees are equal to the onlooker bees. The em-

ployed bees are the bees which searches the food source and gather the information about the quality of the food source. Onlooker bees which stay in the hive 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 food sources. Similar to the other population-based algorithms, ABC solution search process is an iterative process. After, initialisation 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 phases is described as follows.

3.1. Initialisation of the swarm

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

$$x_{ij} = x_{minj} + \text{rand}[0, 1](x_{maxj} - x_{minj}), \quad (2)$$

here x_{minj} and x_{maxj} are bounds of x_i in j th direction and $\text{rand}[0, 1]$ is a uniformly distributed random number in the range $[0, 1]$.

3.2. Employed bee phase

In the 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 i th candidate in this phase is

$$v_{ij} = x_{ij} + \phi_{ij}(x_{ij} - x_{kj}), \quad (3)$$

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

3.3. 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 analyse 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):

$$\text{prob}_i = \frac{\text{fitness}_i}{\sum_{i=1}^{\text{SN}} \text{fitness}_i}, \quad (4)$$

here 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 memorises the new position and forgets the old one.

3.4. Scout bees phase

If the position of a food source is not updated up to 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:

$$x_{ij} = x_{\min j} + \text{rand}[0, 1](x_{\max j} - x_{\min j}), \quad \text{for } j \in \{1, 2, \dots, D\}, \quad (5)$$

where $x_{\min j}$ and $x_{\max j}$ are bounds of x_i in j th direction.

3.5. 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. (Karaboga & Akay, 2009).

Algorithm 1. Artificial bee colony algorithm

```

Initialise 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: memorise the best food source found so far.
end while
Output the best solution found so far.

```

4. Lévy flight inspired search strategy

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

In this paper, we are proposing an LS strategy inspired by Lévy flight random walk and named LFSS. In past, the flight behaviour of many animals and insects has been analysed in various studies which exhibit the important properties of Lévy flights (Brown, Liebovitch, & Glendon, 2007; Pavlyukevich, 2007; Reynolds & Frye, 2007; Yang & Deb, 2010). Furthermore, this flight behaviour has been applied to optimisation and search algorithms, and the reported results show its importance in the field of solution search algorithms (Pavlyukevich, 2007; Reynolds & Frye, 2007; Shlesinger, 2006; Shlesinger, Zaslavsky, & Frisch, 1995). Recently, Yang proposed a new metaheuristic algorithm by combining Lévy flights with the search strategy via the firefly algorithm (Yang, 2010a).

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 Equation (6):

$$L(s) \sim |s|^{-1-\beta}, \text{ where } \beta (0 < \beta \leq 2) \text{ is an index and } s \text{ is the step length.} \quad (6)$$

In this paper, a Mantegna algorithm (Yang, 2010b) 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 Mantegna’s algorithm, the step length s can be calculated by

$$s = \frac{u}{|v|^{1/\beta}}, \quad (7)$$

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

$$u \sim N(0, \sigma_u^2), \quad v \sim N(0, \sigma_v^2), \quad (8)$$

where

$$\sigma_u = \left\{ \frac{\Gamma(1 + \beta) \sin(\pi\beta/2)}{\beta\Gamma[(1 + \beta)/2] 2^{(\beta-1)/2}} \right\}^{1/\beta}, \quad \sigma_v = 1. \quad (9)$$

This distribution (for s) obeys the expected Lévy distribution for $|s| \geq |s_0|$, where s_0 is the smallest step length (Yang, 2010b). Here $\Gamma(\cdot)$ is the Gamma function and calculated as follows:

$$\Gamma(1 + \beta) = \int_0^\infty t^\beta e^{-t} dt. \quad (10)$$

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:

$$\text{step_size}(t) = 0.001 \times s(t) \times SLC, \quad (11)$$

here t is the iteration counter for LS strategy, $s(t)$ is calculated using Lévy distribution as shown in Equation (7) and SLC is the social learning component of the global search algorithm.

In Lévy flights, the step sizes are too aggressive, that is, they may generate new solutions often outside the domain or on boundary. Since, the LS algorithms can be seen as 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 Equation (11) to reduce the step size. The solution update equation of an i th individual based on the proposed LS strategy is given in Equation (12):

$$x'_{ij}(t+1) = x_{ij}(t) + \text{step_size}(t) \times U(0, 1), \quad (12)$$

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 $\text{step_size}(t) \times U(0, 1)$ is the actual random walks or flights drawn from Lévy distribution.

The pseudo-code of the proposed LFSS is shown in Algorithm 2. In Algorithm 2, ϵ determines the termination of LS.

5. Lévy flight artificial bee colony

Exploration and exploitation are the two important characteristics of the population-based optimisation algorithms such as GA (Goldberg & Holland, 1988), PSO (Kennedy & Eberhart, 1995), DE (Storn & Price, 1997), BFO (Passino, 2002) and so on. In these optimisation algorithms, the exploration refers to the ability to investigate the various unknown regions in the solution space to discover the global

Algorithm 2 Lévy flight search strategy

```

Input optimisation function  $\text{Min } f(x)$  and  $\beta$ ;
Select an individual  $x_i$  in the swarm which is going to
modify its position;
Initialise  $t = 1$  and  $\sigma_v = 1$ ;
Compute  $\sigma_u$  using Equation (9);
while ( $t < \epsilon$ ) do
    Compute step_size using Equation (11);
    Generate a new solution  $x'_i$  using Equation (12);
    Calculate  $f(x'_i)$ ;
    if  $f(x'_i) < f(x_i)$  then
         $x_i = x'_i$ ;
    end if
     $t = t + 1$ ;
end while

```

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 optimisation performance, the two abilities should be well balanced. Karaboga and Akay (2009) (Karaboga & Akay, 2009) tested different variants of ABC for global optimisation 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 cannot be done manually.

The exploitation capability can be enhanced by incorporation of an LS algorithm with the ABC algorithm. Therefore, in this paper, to balance the diversity and convergence ability of ABC, the following four modifications are proposed.

- (1) To enhance the exploitation capability of ABC, LFSS (described in Section 4) 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 Lévy flight search

algorithm, in case of large step sizes, can search within the area that is jumped by the basic ABC.

- (2) In the basic ABC, the food sources are updated, as shown in equation (3). Inspired by PSO (Kennedy & Eberhart, 1995) and GABC (Zhu & Kwong, 2010) 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 Equation (3) is modified as follows (Zhu & Kwong, 2010):

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

here, ψ_{ij} is a uniform random number in $[0, C]$, where C is a non-negative constant. For detailed description refer to Zhu and Kwong (2010).

- (3) In the basic ABC, food sources are randomly initialised by the scout bees 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, in this paper, the scout bees randomly initialise the abandoned food sources by using current interval in the swarm which is, as the search does progress, increasingly smaller than the corresponding initial range. Now the following equation is used to update a food source x_i in the scout bee phase:

$$x_{ij} = a_j + \text{rand}[0, 1](b_j - a_j),$$

here, $[a_j, b_j]$ is the shrunken search interval in the j th direction.

- (4) To enhance the exploration capability, the numbers of scout bees are increased. This modification avoids situation of stagnation of the algorithm. Therefore, in this paper, all the bees who crosses the limit boundary are treated as the scout bees. However, only if this modification has been done in basic ABC, then it may make ABC relatively less stable as the scout bees do not use the previous knowledge for generating new food solutions and hence previous learning has been lost. But in the proposed strategy, first modification give more chance to best solution to update itself. Second modification uses the experience of global best solution and hence improves the convergence and third modification retains the acquired experience of the swarm about the search area, hence helps in exploitation. Therefore, in the first three modifications, better solutions get more chance in search process and minimise the threat of less stability while taking advantage of fourth modification in exploration of the search space.

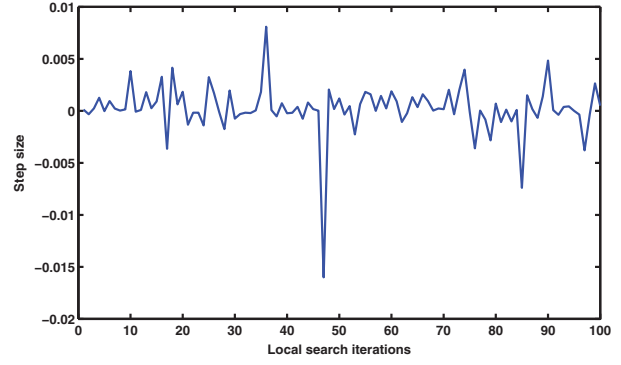


Figure 1. Size of Lévy flights in two dimension search space for f_{11} .

As described in the proposed first modification, a Lévy flight random walk inspired LS is incorporated with the basic ABC to improve the exploitation capability. In the proposed LS strategy, step size is calculated as shown in Equation (13)

$$\text{step_size}(t) = 0.001 \times s(t) \times (x_{bestj}(t) - x_{kj}(t)), \quad (13)$$

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 LS strategy, is given in Equation (14):

$$x'_{bestj}(t+1) = x_{bestj}(t) + \text{step_size}(t) \times U(0, 1). \quad (14)$$

The proposed strategy in ABC is hereby, named as Lévy Flight ABC (LFABC). In LFSS, only the best particle of the current swarm updates itself in its neighbourhood. Figures 1, 2 and 3 show an example of the Lévy flight random walk used to update an individual in two dimension search space for Goldstein–Price function (f_{11}),

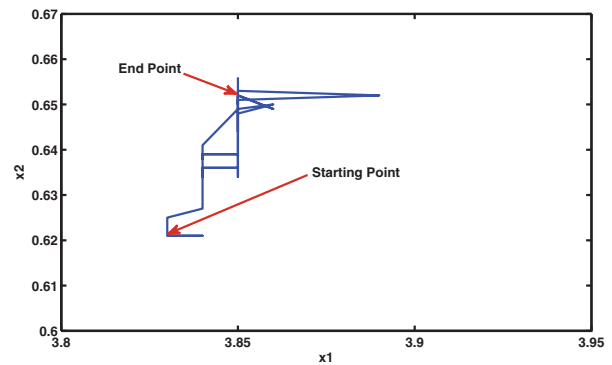


Figure 2. Best solution movement in the swarm during LFSS in two dimension search space for f_{11} .

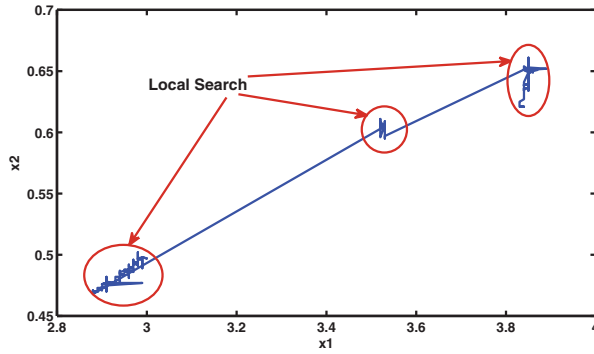


Figure 3. Best solution movement using LFABC, in two dimension search space for f_{11} .

refer Table 1. The pseudo-code of the proposed LFSS with ABC is shown in Algorithm 3.

Algorithm 3 Lévy flight search strategy with ABC

Input optimisation function $\text{Min } f(x)$ and β ;
 Select the best solution x_{best} in the swarm;
 Initialise $t = 1$ and $\sigma_v = 1$;
 Compute σ_u using Equation (9);
while ($t < \epsilon$) **do**
 Compute step_size using Equation (11);
 Generate a new solution x'_{best} using Algorithm 4;
 Calculate $f(x'_{\text{best}})$;
 if $f(x'_{\text{best}}) < f(x_{\text{best}})$ **then**
 $x_{\text{best}} = x'_{\text{best}}$;
 end if
 $t = t + 1$;
end while

Algorithm 4 New solution generation

Input the best solution x_{best} and s ;
for $j = 1$ to D **do**
 if $U(0, 1) > p_r$ **then**
 $x'_{\text{best},j} = x_{\text{best},j} + 0.001 \times s \times (x_{\text{best},j} - x_{kj}) \times U(0, 1)$;
 else
 $x'_{\text{best},j} = x_{\text{best},j}$;
 end if
end for
 Return x'_{best}

In Algorithms 3 and 4, ϵ is the termination criteria of the proposed LS. 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 Section 6.2 for details of these parameter settings.

The proposed LFABC consists of four phases: employed bee phase, onlooker bee phase, scout bee phase and LFSS. The pseudo-code of the LFABC algorithm is shown in Algorithm 5.

Algorithm 5 Lévy Flight ABC

Initialise 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 search strategy (LFSS) phase using Algorithm 3.
end while
 Print best solution.

6. Experimental results and discussion

6.1. Test problems under consideration

In order to analyse the performance of LFABC, 20 different global optimisation problems (f_1 to f_{20}) are selected (listed in Table 1). These are continuous optimisation problems and have different degrees of complexity and multimodality. Test problems f_1 to f_5 and f_{11} to f_{20} are taken from Ali, Khompatraporn, and Zabinsky (2005) and test problems f_6 to f_{10} are taken from Suganthan et al. (2005) with the associated offset values.

6.2. Experimental setting

To prove the efficiency of LFABC, it is compared with ABC and recent variants of ABC named GABC (Zhu & Kwong, 2010), BSFABC (Banharnsakun et al., 2011) and MABC (Akay & Karaboga, 2012). To test LFABC, ABC, GABC, BSFABC and MABC over considered problems, following experimental setting is adopted.

- Colony size $NP = 50$ (Diwold, Aderhold, Scheidler, & Middendorf, 2011; El-Abd, 2011).
- $\phi_{ij} = \text{rand}[-1, 1]$.
- Number of food sources $SN = NP/2$.
- $\text{limit} = D \times SN$ (Akay & Karaboga, 2012; Karaboga & Basturk, 2007).
- The stopping criteria is either maximum number of function evaluations (which is set to be 200,000) is reached or the acceptable error (mentioned in Table 1) has been achieved.
- The number of simulations/run = 100.
- $C = 1.5$ (Zhu & Kwong, 2010),

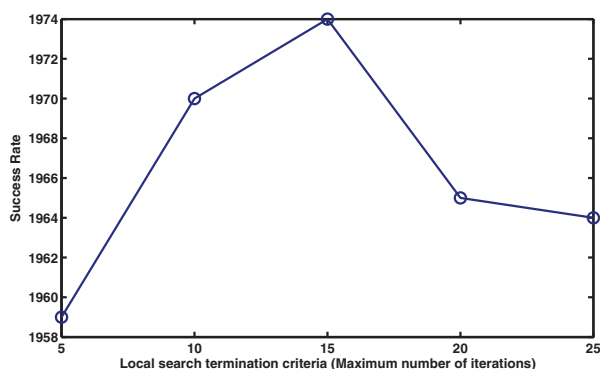
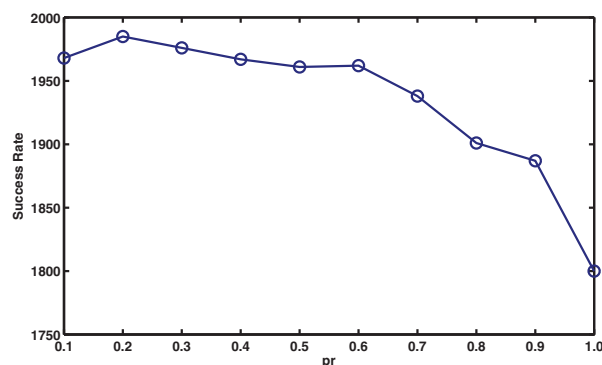
Table 1. Test problems.

Test problem	Objective function	Search range	Optimum value	D	Acceptable error
Neumaier 3 problem (NF3)	$f_1(x) = \sum_{i=1}^D (x_i - 1)^2 - \sum_{i=2}^D x_i x_{i-1}$	$[-D^2, D^2]$	$f(\vec{0}) = -(D \times (D + 4)(D - 1))/6.0$	10	1.0E - 01
Beale function	$f_2(x) = [1.5 - x_1(1 - x_2)]^2 + [2.25 - x_1(1 - x_2^2)]^2 + [2.625 - x_1(1 - x_2^3)]^2$	$[-4.5, 4.5]$	$f(3, 0.5) = 0$	2	1.0E - 05
Colville function	$f_3(x) = 100[x_2 - x_1^2]^2 + (1 - x_1)^2 + 90(x_4 - x_3^2)^2 + (1 - x_3)^2 + 10.1[(x_2 - 1)^2 + (x_4 - 1)^2] + 19.8(x_2 - 1)(x_4 - 1)$	$[-10, 10]$	$f(\vec{1}) = 0$	4	1.0E - 05
Branins's function	$f_4(x) = a(x_2 - bx_1^2 + cx_1 - d)^2 + e(1 - f) \cos x_1 + e$	$x_1 \in [-5, 10], x_2 \in [0, 15]$	$f(-\pi, 12.275) = 0.3979$	2	1.0E - 05
Kowalik function	$f_5(x) = \sum_{i=1}^{11} \left[a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4} \right]^2$	$[-5, 5]$	$f(0.1928, 0.1908, 0.1231, 0.1357) = 3.07E - 04$	4	1.0E - 05
Shifted Rosenbrock	$f_6(x) = \sum_{i=1}^{D-1} (100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2) + f_{\text{bias}}, z = x - o + 1, x = [x_1, x_2, \dots, x_D], o = [o_1, o_2, \dots, o_D]$	$[-100, 100]$	$f(o) = f_{\text{bias}} = 390$	10	1.0E - 01
Shifted sphere	$f_7(x) = \sum_{i=1}^D z_i^2 + f_{\text{bias}}, z = x - o, x = [x_1, x_2, \dots, x_D], o = [o_1, o_2, \dots, o_D]$	$[-100, 100]$	$f(o) = f_{\text{bias}} = -450$	10	1.0E - 05
Shifted Rastrigin	$f_8(x) = \sum_{i=1}^D (z_i^2 - 10 \cos(2\pi z_i) + 10) + f_{\text{bias}}, z = (x - o), x = (x_1, x_2, \dots, x_D), o = (o_1, o_2, \dots, o_D)$	$[-5, 5]$	$f(o) = f_{\text{bias}} = -330$	10	1.0E - 02
Shifted Griewank	$f_9(x) = \sum_{i=1}^D \frac{z_i^2}{4000} - \prod_{i=1}^D \cos(\frac{z_i}{\sqrt{i}}) + 1 + f_{\text{bias}}, z = (x - o), x = [x_1, x_2, \dots, x_D], o = [o_1, o_2, \dots, o_D]$	$[-600, 600]$	$f(o) = f_{\text{bias}} = -180$	10	1.0E - 05
Shifted Ackley	$f_{10}(x) = -20 \exp(-0.2 \sqrt{\frac{1}{D} \sum_{i=1}^D z_i^2}) - \exp(\frac{1}{D} \sum_{i=1}^D \cos(2\pi z_i)) + 20 + e + f_{\text{bias}}, z = (x - o), x = (x_1, x_2, \dots, x_D), o = (o_1, o_2, \dots, o_D)$	$[-32, 32]$	$f(o) = f_{\text{bias}} = -140$	10	1.0E - 05
Goldstein-Price	$f_{11}(x) = (1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2))(30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2))$	$[-2, 2]$	$f(0, -1) = 3$	2	1.0E - 14
Six-hump camel back	$f_{12}(x) = (4 - 2.1x_1^2 + x_1^4/3)x_2^2 + x_1x_2 + (-4 + 4x_2^2)x_2^2$	$[-5, 5]$	$f(-0.0898, 0.7126) = -1.0316$	2	1.0E - 05
Easom's function	$f_{13}(x) = -\cos x_1 \cos x_2 e^{((-(x_1 - \pi)^2 - (x_2 - \pi)^2))}$	$[-10, 10]$	$f(\pi, \pi) = -1$	2	1.0E - 13
Dekkers and Aarts	$f_{14}(x) = 10^5 x_1^2 + x_2^2 - (x_1^2 + x_2^2)^2 + 10^{-5}(x_1^2 + x_2^2)^4$	$[-20, 20]$	$f(0, 15) = f(0, -15) = -24777$	2	5.0E - 01

(continued)

Table 1. (Continued)

Test problem	Objective function	Search range	Optimum value	D	Acceptable error
Hosaki problem	$f_{15} = (1 - 8x_1 + 7x_1^2 - 7/3x_1^3 + 1/4x_1^4)x_2^2 \exp(-x_2)$	$x_1 \in [0, 5], x_2 \in [0, 6]$	-2.3458	2	1.0E - 6
McCormick	$f_{16}(x) = \sin(x_1 + x_2) + (x_1 - x_2)^2 - \frac{3}{2}x_1 + \frac{5}{2}x_2 + 1$	$x_1 \in [-1.5, 4], x_2 \in [-3, 3]$	$f(-0.547, -1.547) = -1.9133$	30	1.0E - 04
Meyer and Roth	$f_{17}(x) = \sum_{i=1}^5 \left(\frac{x_1 x_3 f_i}{1 + x_1 f_i + x_2 v_i} - y_i \right)^2$	$[-10, 10]$	$f(3.13, 15.16, 0.78) = 0.4E - 04$	3	1.0E - 03
Shubert	$f_{18}(x) = -\sum_{i=1}^5 i \cos((i + 1)x_1 + 1) \sum_{i=1}^5 i \cos((i + 1)x_2 + 1)$	$[-10, 10]$	$f(7.0835, 4.8580) = -186.7309$	2	1.0E - 05
Sinusoidal	$f_{19}(x) = -[A \prod_{i=1}^n \sin(x_i - z) + \prod_{i=1}^n \sin(B(x_i - z))], A = 2.5, B = 5, z = 30$	$[0, 180]$	$f(90 + z) = -(A + 1)$	10	1.0E - 02
Moved axis parallel hyper-ellipsoid	$f_{20}(x) = \sum_{i=1}^D 5i x_i^2$	$[-5.12, 5.12]$	$f(x) = 0; x(i) = 5i, i = 1:D$	30	1.0E - 15

Figure 4. Effect of LFSS termination criteria (ϵ) on SR.Figure 5. Effect of parameter p_r on SR.

- The value of $\beta = 2$ is to be set based on the empirical experiments.
- To set termination criteria of LFSS, the performance of LFABC is measured for considered test problems on different values of ϵ and results are analysed in Figure 4. It is clear from Figure 4 that $\epsilon = 15$ gives better results. Therefore, termination criteria is set to be $\epsilon = 15$.
- Parameter settings for the algorithms GABC, BSFABC and MABC are similar to their original research papers.
- In order to investigate the effect of the parameter p_r , described by Algorithm 4 on the performance of LFABC, its sensitivity with respect to different values of p_r in the range $[0.1, 1]$ is examined in Figure 5. It can be observed from Figure 5 that the test prob-

lems 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 this paper.

6.3. Results comparison

Numerical results with experimental setting of Subsection 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 LFABC outperforms in terms of reliability, efficiency and accuracy as compared to the basic ABC, GABC, BSFABC and MABC. Some more intensive analyses based on acceleration rate (AR) (Rahnamayan, Tizhoosh, & Salama, 2008), performance indices (PI) and

Table 2. Comparison of the results of test problems.

Test function	Algorithm	SD	ME	AFE	SR
f_1	ABC	8.36E-01	9.66E-01	197,813.12	4
	LFABC	6.84E-02	1.07E-01	39,650.81	95
	GABC	1.58E+00	1.16E+00	191,063.41	11
	BSFABC	5.19E+00	4.18E+00	200,026.1	0
	MABC	6.65E-03	9.91E-02	127,828.88	98
f_2	ABC	1.55E-06	8.74E-06	16,402.52	100
	LFABC	2.84E-06	7.52E-06	3746.11	100
	GABC	3.02E-06	5.28E-06	9553.01	100
	BSFABC	2.31E-05	1.26E-05	43,260.44	96
	MABC	2.76E-06	5.12E-06	9974.54	100
f_3	ABC	1.15E-01	1.62E-01	194,852.27	4
	LFABC	1.29E-03	9.19E-03	65,107.64	100
	GABC	1.39E-02	1.72E-02	161,149.56	42
	BSFABC	3.47E-02	2.38E-02	156,594.82	42
	MABC	1.05E-02	1.42E-02	98,919.26	82
f_4	ABC	5.98E-06	5.13E-06	2020.3	100
	LFABC	7.12E-06	6.34E-06	14,867.54	93
	GABC	6.50E-06	5.56E-06	17,019.79	92
	BSFABC	6.92E-06	6.03E-06	33,561.7	84
	MABC	6.95E-06	5.87E-06	24,764.92	89
f_5	ABC	7.33E-05	1.70E-04	181,870.15	21
	LFABC	1.79E-04	1.37E-04	61,386.26	95
	GABC	2.69E-05	8.42E-05	81,335.87	95
	BSFABC	6.27E-05	1.30E-04	140,290.92	59
	MABC	7.05E-05	2.08E-04	148,042.35	58
f_6	ABC	1.61E+00	9.45E-01	173,849.01	23
	LFABC	7.64E-01	2.53E-01	66,632.89	95
	GABC	1.16E-01	9.64E-02	110,835.85	93
	BSFABC	4.21E+00	2.64E+00	183,435.91	17
	MABC	9.14E-01	6.63E-01	128,771.75	54
f_7	ABC	2.39E-06	7.25E-06	9014.5	100
	LFABC	2.36E-06	7.27E-06	6203.32	100
	GABC	2.24E-06	6.86E-06	5545	100
	BSFABC	2.32E-06	7.09E-06	18,064.5	100
	MABC	1.66E-06	7.72E-06	8671	100
f_8	ABC	9.61E+00	8.68E+01	200,011.85	0
	LFABC	2.07E+01	1.30E+02	200,026.27	0
	GABC	9.87E+00	8.46E+01	200,008.55	0
	BSFABC	1.76E+01	1.20E+02	200,036.17	0
	MABC	1.13E+01	8.26E+01	200,014.08	0
f_9	ABC	2.54E-03	8.42E-04	69,495.97	90
	LFABC	7.35E-04	8.01E-05	40,382.88	100
	GABC	2.97E-06	5.48E-06	40,280.38	100
	BSFABC	5.83E-03	4.02E-03	105,148.43	64
	MABC	1.03E-03	1.54E-04	78,393.47	98
f_{10}	ABC	1.84E-06	7.86E-06	16,615.5	100
	LFABC	1.34E-06	8.66E-06	10,934.63	100
	GABC	1.28E-06	8.55E-06	9376.5	100
	BSFABC	1.74E-06	8.21E-06	31,209	100
	MABC	1.00E-06	8.85E-06	14,268.06	100
f_{11}	ABC	6.06E-06	1.18E-06	107,740.64	61
	LFABC	4.40E-15	5.22E-15	4468.6	100
	GABC	4.33E-15	4.63E-15	4017.73	100
	BSFABC	4.89E-15	6.67E-15	13,388.95	100
	MABC	4.31E-15	5.04E-15	12,893.26	100

(continued)

Table 2. (Continued).

Test function	Algorithm	SD	ME	AFE	SR
f_{12}	ABC	1.07E-05	1.24E-05	982.52	100
	LFABC	1.51E-05	1.33E-05	70,344.57	65
	GABC	1.44E-05	1.69E-05	100,300.68	50
	BSFABC	1.39E-05	1.86E-05	118,329.65	41
	MABC	1.51E-05	1.65E-05	98,785	51
f_{13}	ABC	5.27E-05	1.98E-05	197,359.89	4
	LFABC	3.28E-14	5.60E-14	14,065.55	100
	GABC	3.48E-12	3.97E-13	51,043.52	99
	BSFABC	3.23E-14	4.83E-14	4680.58	100
	MABC	2.51E-03	1.29E-03	200,025.34	0
f_{14}	ABC	4.95E-03	4.89E-01	1424.57	100
	LFABC	5.68E-03	4.91E-01	687.8	100
	GABC	5.02E-03	4.89E-01	778	100
	BSFABC	5.34E-03	4.91E-01	2775.72	100
	MABC	5.85E-03	4.91E-01	2326.44	100
f_{15}	ABC	5.91E-06	5.50E-06	656.5	100
	LFABC	6.16E-06	5.45E-06	12,378.82	94
	GABC	6.47E-06	5.81E-06	18,342.24	91
	BSFABC	6.53E-06	6.58E-06	38,545.38	81
	MABC	6.35E-06	5.53E-06	16,954.62	92
f_{16}	ABC	7.00E-06	8.89E-05	1244.56	100
	LFABC	6.96E-06	9.04E-05	587.42	100
	GABC	6.00E-06	8.74E-05	612.5	100
	BSFABC	6.57E-06	8.79E-05	989.56	100
	MABC	7.01E-06	8.86E-05	1711.38	100
f_{17}	ABC	2.84E-06	1.95E-03	31,280.19	100
	LFABC	3.10E-06	1.95E-03	3418.07	100
	GABC	3.18E-06	1.95E-03	5088.67	100
	BSFABC	2.99E-06	1.95E-03	19,162.9	100
	MABC	2.80E-06	1.95E-03	8565.87	100
f_{18}	ABC	5.89E-06	5.19E-06	4572.09	100
	LFABC	5.83E-06	5.16E-06	1619.34	100
	GABC	5.71E-06	5.02E-06	2467.47	100
	BSFABC	5.17E-06	4.37E-06	9081.59	100
	MABC	5.98E-06	5.21E-06	27,782.89	100
f_{19}	ABC	1.89E-03	7.64E-03	51,845.51	100
	LFABC	1.67E-03	8.35E-03	22,030.31	100
	GABC	2.13E-03	7.70E-03	47,747.58	100
	BSFABC	2.05E-03	7.75E-03	64,507.74	100
	MABC	8.94E-02	6.21E-01	200,033.69	0
f_{20}	ABC	1.47E-16	8.20E-16	59,699	100
	LFABC	1.09E-16	8.75E-16	44,903	100
	GABC	9.98E-17	8.66E-16	48,738.5	100
	BSFABC	2.37E-16	7.17E-16	71,124	100
	MABC	hline 7.11E-17	9.03E-16	59,554	100

boxplots have been carried out for results of ABC and its variants.

LFABC, 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 summarised in Table 3. In Table 3,

‘+’ indicates that the LFABC 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 LFABC over ABC, GABC, BSFABC and MABC.

Furthermore, we compare the convergence speed of the considered algorithms by measuring the AFEs. Smaller AFEs means higher convergence speed. In order to minimise the effect of the stochastic nature of the algorithms,

Table 3. Summary of Table 2 outcome.

Function	LFABC vs ABC	LFABC vs GABC	LFABC vs BSFABC	LFABC vs 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}	+	+	+	+
f_{20}	+	+	+	+
Total number of + sign	16	15	18	18

the reported function evaluations for each test problem are the average over 100 runs. In order to compare convergence speeds, we use the which is defined as follows, based on the AFEs for the two algorithms ALGO and LFABC:

$$AR = \frac{AFE_{ALGO}}{AFE_{LFABC}}, \quad (15)$$

Table 4. Acceleration rate (AR) of LFABC compare to the basic ABC, GABC, BSFABC and MABC.

Test problems	ABC	GABC	BSFABC	MABC
f_1	4.988879672	4.818650867	5.044691395	3.22386554
f_2	4.378547346	2.550114652	11.54809656	2.662639378
f_3	2.992771202	2.475125193	2.405168119	1.519318777
f_4	0.135886636	1.144761675	2.257380844	1.665703943
f_5	2.962717553	1.32498494	2.285379823	2.411652868
f_6	2.609057029	1.663380502	2.752933424	1.932555379
f_7	1.453173462	0.893876182	2.912069666	1.397799888
f_8	0.999927909	0.999911412	1.000049493	0.999939058
f_9	1.720926541	0.997461796	2.603787298	1.941255057
f_{10}	1.519530153	0.857505009	2.854143213	1.304850736
f_{11}	24.11060287	0.899102627	2.996229244	2.885301884
f_{12}	0.013967247	1.425848221	1.682143341	1.404301711
f_{13}	14.0314378	3.628974338	0.332769071	14.22093981
f_{14}	2.071198023	1.131142774	4.035649898	3.382436755
f_{15}	0.053034134	1.481743817	3.113816987	1.369647511
f_{16}	2.118688502	1.042695176	1.684586837	2.91338395
f_{17}	9.151418783	1.48875535	5.606350952	2.506054586
f_{18}	2.823428063	1.523750417	5.608204577	17.15692196
f_{19}	2.353371786	2.167358517	2.928135827	9.079930786
f_{20}	1.329510278	1.085417455	1.58394762	1.326281095

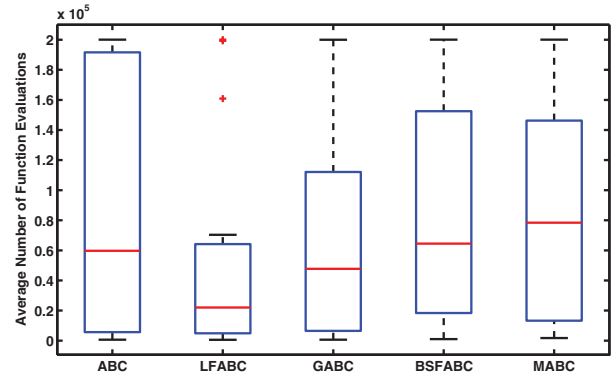


Figure 6. Boxplots graphs for AFE.

where $ALGO \in \{ABC, GABC, BSFABC, MABC\}$ and $AR > 1$ means LFABC is faster. In order to investigate the AR of the proposed algorithm compared to the basic ABC and its variants, results of Table 2 are analysed and the value of AR is calculated using Equation (15). Table 4 shows a clear comparison between LFABC and ABC, LFABC and GABC, LFABC and BSFABC, and LFABC and MABC in terms of AR. It is clear from Table 4 that convergence speed of LFABC 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 (Williamson, Parker, & Kendrick, 1989). The boxplots for ABC, LFABC, GABC, BSFABC and MABC are shown in Figure 6. It is clear from this figure that LFABC

is better than the considered algorithms as interquartile range and median are comparatively low.

Furthermore, to compare the considered algorithms, by giving weighted importance to the SR, the ME and the average number of function evaluations, PI are calculated (Bansal & Sharma, 2012). The values of PI for the ABC, LFABC, GABC, BSFABC and MABC are calculated by using following equations:

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

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

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

- Sr^i = successful simulations/runs of i th problem.
- Tr^i = total simulations of i th problem.
- Mf^i = minimum of average number of function evaluations used for obtaining the required solution of i th problem.
- Af^i = average number of function evaluations used for obtaining the required solution of i th problem.
- Mo^i = minimum of ME obtained for the i th problem.
- Ao^i = ME obtained by an algorithm for the i th problem.
- N_p = total number of optimisation problems evaluated.

The weights assigned to the SR, the average number of function evaluations and the ME 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 variables vary from 0 to 1 as given in Bansal and Sharma (2012). 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, LFABC, GABC, BSFABC and MABC are shown in Figure 7(a)–(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 ME are given equal weights. PIs of the considered algorithms are superimposed in Figure 7(a) for comparison of the performance. It is observed that PI of LFABC are higher than the considered algorithms. In case (2), equal weights are assigned to the SR and ME and in case (3), equal weights are assigned to the SR and average number

of function evaluations. It is clear from Figure 7(b) and 7(c) that the algorithms perform same as in case (1).

7. Applications of LFABC to engineering optimisation problems

To see the robustness of the proposed strategy, five well-known engineering optimisation problems, namely, pressure vessel (confinement method) (Wang, Gao, & Ovaska, 2008), Lennard-Jones (Clerc, 2012), parameter estimation for frequency-modulated (FM) sound waves (Das & Suganthan, 2010), compression spring (Onwubolu & Babu, 2004; Sandgren, 1990) and welded beam design optimisation problem (Mahdavi, Fesanghary, & Damangir, 2007; Ragsdell & Phillips, 1976) are solved. The considered engineering optimisation problems are described as follows.

7.1. Pressure vessel design

The pressure vessel design is to minimise the total cost of the material, forming and welding of a cylindrical vessel (Wang et al., 2008). 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 optimisation problem is as follows:

$$E_1(\vec{X}) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 \\ + 3.1611x_1^2x_4 + 19.84x_1^2x_3,$$

subject to

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

The search boundaries for the variables are $1.125 \leq x_1 \leq 12.5$, $0.625 \leq x_2 \leq 12.5$, $1.0E - 8 \leq x_3 \leq 240$ and $1.0E - 8 \leq x_4 \leq 240$. The known global optimum solution is $f(1.125, 0.625, 55.8592, 57.7315) = 7197.729$ (Wang et al., 2008). A algorithm is said to be successful if it finds error less than $1.0E - 5$.

7.2. Lennard-Jones

It is a potential energy minimisation problem 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

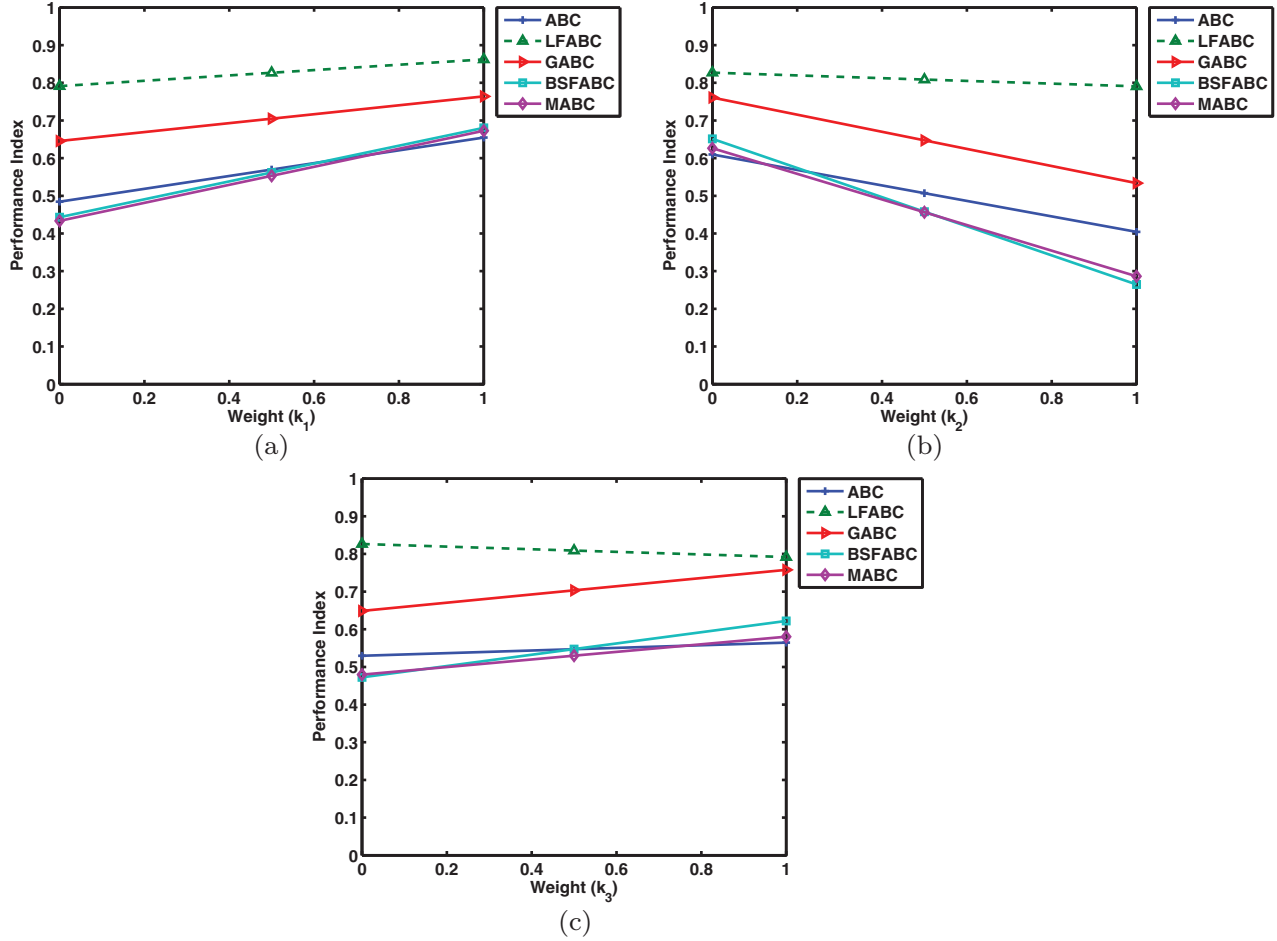


Figure 7. Performance index for test problems: (a) for case (1), (b) for case (2) and (c) for case (3).

$X = (X_1, X_2, \dots, X_N)$, and we have then

$$E_2(\vec{X}) = \sum_{i=1}^{N-1} \sum_{j=i+1}^N \left(\frac{1}{\|X_i - X_j\|^{2\alpha}} - \frac{1}{\|X_i - X_j\|^\alpha} \right).$$

In this study, $N = 5$, $\alpha = 6$ and the search space is $[-2, 2]$ (Clerc, 2012).

7.3. Frequency-modulated (FM) sound wave

Frequency-modulated (FM) sound wave synthesis has an important role in several modern music systems. The parameter optimisation of an FM synthesiser is a six-dimensional optimisation problem where the vector to be optimised is $\vec{X} = \{a_1, w_1, a_2, w_2, a_3, w_3\}$ of the sound wave given in Equation (16). 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 $f(\vec{X}_{sol}) = 0$. This problem has been tackled using GAs in Refs [1,2]. The expressions for the estimated sound and the target sound waves are given as

$$y(t) = a_1 \sin(w_1 t \theta + a_2 \sin(w_2 t \theta + a_3 \sin(w_3 t \theta))), \quad (16)$$

$$y_0(t) = (1.0) \sin((5.0)t\theta - (1.5) \sin((4.8)t\theta) + (2.0) \sin((4.9)t\theta)), \quad (17)$$

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:

$$E_3(\vec{X}) = \sum_{i=0}^{100} (y(t) - y_0(t))^2.$$

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

7.4. Compression spring

The considered fourth engineering optimisation application is compression spring problem (Onwubolu & Babu, 2004;

Table 5. Comparison of the results of test problems.

Test function	Algorithm	SD	ME	AFE	SR
E_1	ABC	1.21E+01	1.78E+01	200,022.32	0
	LFABC	1.62E+00	1.03E+00	199,313.43	3
	GABC	4.04E+00	6.04E+00	200,023.52	0
	BSFABC	2.23E+01	2.73E+01	200,038.25	0
	MABC	8.64E+00	1.47E+01	200,025.66	0
E_2	ABC	1.32E-04	8.65E-04	73,667.4	99
	LFABC	1.16E-04	9.13E-04	29,437.45	98
	GABC	7.03E-04	1.15E-03	112,503.53	73
	BSFABC	3.54E-04	9.74E-04	126,621.9	81
	MABC	1.65E-01	4.49E-01	140,965.5	25
E_3	ABC	5.23E+00	5.99E+00	200,033.41	0
	LFABC	5.22E+00	3.99E+00	160,861.01	45
	GABC	4.86E+00	3.69E+00	191,679.01	11
	BSFABC	4.86E+00	1.02E+01	200,031.93	0
	MABC	3.05E+00	2.81E+00	200,018.79	0
E_4	ABC	1.17E-02	1.36E-02	187,602.32	10
	LFABC	1.27E-03	1.37E-03	147,049.12	24
	GABC	9.50E-03	8.64E-03	189,543.56	11
	BSFABC	3.08E-03	3.02E-02	200,031.13	0
	MABC	6.59E-03	5.28E-03	181,705.01	15
E_5	ABC	8.75E-02	2.52E-01	200,017.84	1
	LFABC	5.07E-03	9.38E-02	38,992.28	100
	GABC	9.22E-03	9.91E-02	116,903.66	68
	BSFABC	5.12E-03	9.46E-02	53,885.62	98
	MABC	4.91E-03	9.36E-02	32,049.47	100

Sandgren, 1990). This problem minimises 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

$$\begin{aligned} x_1 &\in \{1, \dots, 70\} \text{ granularity } 1, \\ x_2 &\in [0.6, 3], \\ x_3 &\in [0.207, 0.5] \text{ granularity } 0.001, \end{aligned}$$

and four constraints

$$\begin{aligned} g_1 &= \frac{8C_f F_{\max} x_2}{\pi x_3^3} - S \leq 0, \\ g_2 &= l_f - l_{\max} \leq 0, \\ g_3 &= \sigma_p - \sigma_{pm} \leq 0, \\ g_4 &= \sigma_w - \frac{F_{\max}}{K} F_p K \leq 0, \end{aligned}$$

with

$$\begin{aligned} C_f &= 1 + 0.75 \frac{x_3}{x_2 - x_3} + 0.615 \frac{x_3}{x_2}, \\ F_{\max} &= 1000, \end{aligned}$$

$$S = 189,000,$$

$$l_f = \frac{F_{\max}}{K} + 1.05(x_1 + 2)x_3,$$

$$l_{\max} = 14,$$

$$\sigma_p = \frac{F_p}{K},$$

$$\sigma_{pm} = 6,$$

$$F_p = 300,$$

$$K = 11.5 \times 10^6 \frac{x_3^4}{8x_1 x_2^3},$$

$$\sigma_w = 1.25,$$

and the function to be minimised is

$$E_4(\vec{X}) = \pi^2 \frac{x_2 x_3^2 (x_1 + 2)}{4}.$$

The best known solution is $f(7, 1.386599591, 0.292) = 2.6254$. Acceptable error for this problem is $1.0E - 04$.

7.5. Welded beam design optimisation problem

The problem is to design a welded beam for minimum cost, subject to some constraints (Mahdavi et al., 2007; Ragsdell & Phillips, 1976). 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

Table 6. Summary of Table 5 outcome.

Function	LFABC vs ABC	LFABC vs GABC	LFABC vs BSFABC	LFABC vs MABC
E_1	+	+	+	+
E_2	—	+	+	+
E_3	+	+	+	+
E_4	+	+	+	+
E_5	+	+	+	—

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:

$$E_5(\vec{x}) = 1.10471x_1^2x_2 + 0.04811x_3x_4(14.0 + x_2),$$

subject to

$$g_1(\vec{x}) = \tau(\vec{x}) - \tau_{\max} \leq 0,$$

$$g_2(\vec{x}) = \sigma(\vec{x}) - \sigma_{\max} \leq 0,$$

$$g_3(\vec{x}) = x_1 - x_4 \leq 0,$$

$$g_4(\vec{x}) = \delta(\vec{x}) - \delta_{\max} \leq 0,$$

$$g_5(\vec{x}) = P - P_c(\vec{x}) \leq 0,$$

$$0.125 \leq x_1 \leq 5, \quad 0.1 \leq x_2, x_3 \leq 10 \text{ and } 0.1 \leq x_4 \leq 5,$$

where

$$\tau(\vec{x}) = \sqrt{\tau'^2 - \tau'\tau''\frac{x_2}{R} + \tau''^2},$$

$$\tau' = \frac{P}{\sqrt{2}x_1x_2}, \tau'' = \frac{MR}{J}, M = P\left(L + \frac{x_2}{2}\right),$$

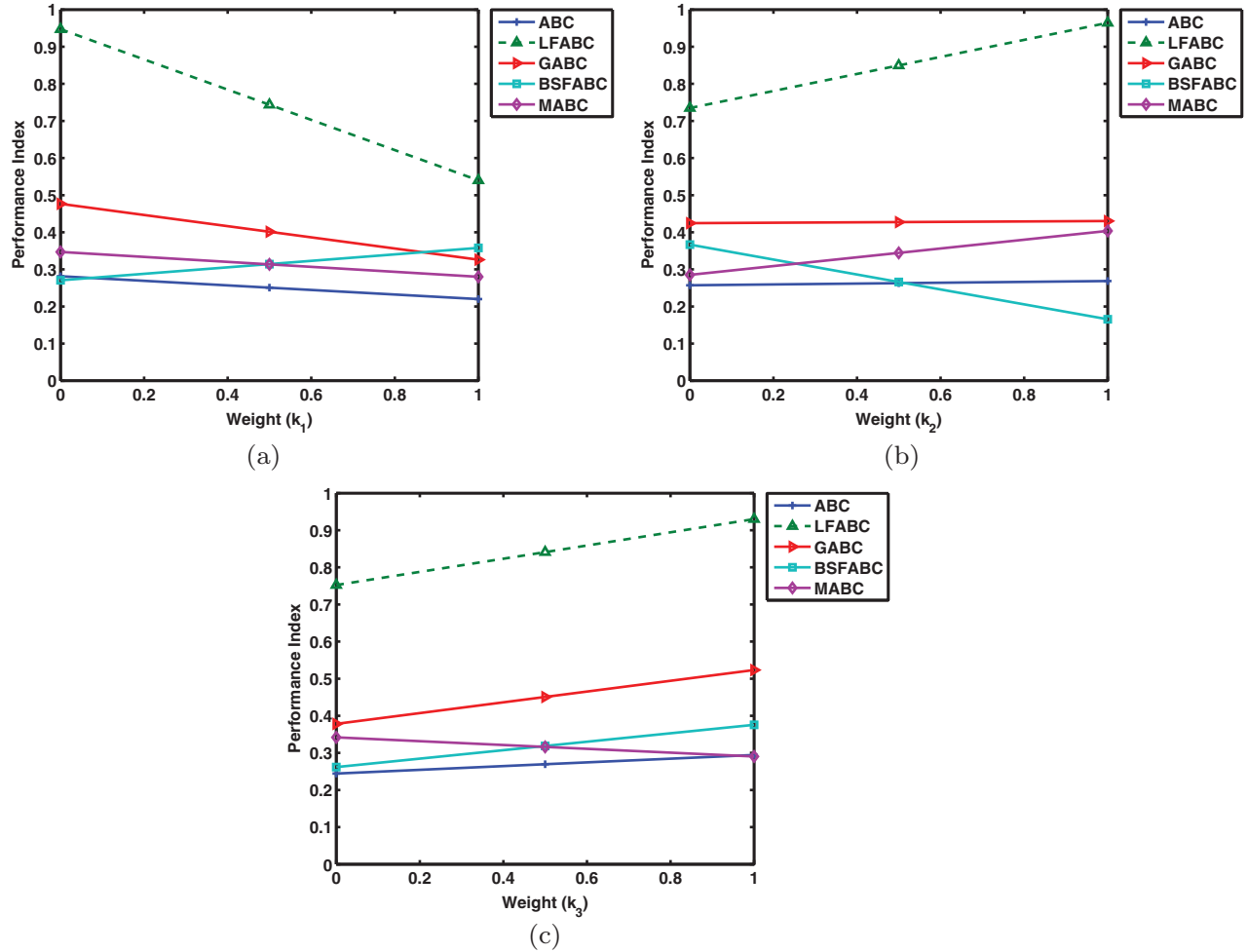


Figure 8. Performance index for engineering optimisation problems: (a) for case (1), (b) for case (2) and (c) for case (3).

$$\begin{aligned}
 R &= \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2}, \\
 J &= 2 / \left(\sqrt{2} x_1 x_2 \left[\frac{x_2^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2 \right] \right), \\
 \sigma(\vec{x}) &= \frac{6PL}{x_4 x_3^2}, \delta(\vec{x}) = \frac{6PL^3}{E x_4 x_3^2}, \\
 P_c(\vec{x}) &= \frac{4.013 E x_3 x_4^3}{6L^2} \left(1 - \frac{x_3}{2L} \sqrt{\frac{E}{4G}} \right), \\
 P &= 6000 \text{ lb}, L = 14 \text{ in.}, \delta_{\max} = 0.25 \text{ in.}, \\
 \sigma_{\max} &= 30,000 \text{ psi},
 \end{aligned}$$

$$\tau_{\max} = 13600 \text{ psi}, E = 30 \times 10^6 \text{ psi}, G = 12 \times 10^6 \text{ psi}.$$

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 $1.0E - 01$.

7.6. Experimental results

To solve the constraint optimisation problems (E_1 , E_4 and E_5), a penalty function approach is used in the experiments. In this approach, the search is modified by converting the original problem into an unconstrained optimisation problem by adding a penalty term in case of constraints violation as shown below

$$f(x) = f(x) + \beta,$$

where $f(x)$ is the original function value and β is the penalty term which is set to 10^3 .

Table 5 shows the experimental results of the considered algorithms on the engineering optimisation problems. It is clear from Table 5 that the LFABC strategy performs better than the considered algorithms.

Furthermore, the algorithms are compared through SR, ME and AFE. On the basis of results shown in Table 5, the results of comparison are given in Table 6. It is clear from Table 6 that the performance of LFABC is better or comparable to the considered algorithms.

The algorithms are also compared on the basis of PI. The PI are calculated same as described in Section 6.3 and the results for each case are shown in Figure 8. It is observed from Figure 8 that the inclusion of the LFSS approach enhances the performance of ABC compared to the basic ABC and its recent variants.

8. Conclusion

ABC can be efficient, but it has a drawback that may lead to incorrect optimal solutions. In this paper, a Lévy flight random walk inspired search strategy is proposed and incorporated with ABC. This Lévy flight based strategy can

carry out both local and global search simultaneously with a focus on more efficient LS. Therefore, it can be more efficient than any standard Gaussian random walks. The proposed modified ABC is named as LFABC. In the proposed LS, new solutions are generated in the neighbourhood of the best solution depending upon a newly introduced parameter, perturbation rate. Furthermore, the proposed algorithm has been extensively compared with other recent variants of ABC, namely, GABC, BSFABC and MABC and with the help of experiments over test problems and engineering optimisation problems. Our simulation results have shown that the LFABC can outperform other algorithms considered in our tests in terms of reliability, efficiency and accuracy.

Disclosure statement

No potential conflict of interest was reported by the authors.

Notes on contributors



Harish Sharma is working as an associate professor at Rajasthan Technical University. He has more than 10 year of teaching experience. In past, he has served as an assistant professor at Government Engineering College Jhalawar and Vardhman Mahaveer Open University, Kota. He received his BTech, MTech degree in Computer Engg. from Government Engineering College, Kota and Rajasthan Technical University, Rajasthan in 2003 and 2009, respectively. Dr Sharma has obtained his PhD in information communication and technology from ABV-Indian Institute of Information Technology and Management, Gwalior, India. His area of research is nature inspired algorithms and soft computing.



Jagdish Chand Bansal is an assistant professor at South Asian University New Delhi. Dr Bansal has obtained his PhD in mathematics from IIT Roorkee. Before joining SAU New Delhi, he has worked as an assistant professor at ABV-Indian Institute of Information Technology and Management Gwalior and BITS Pilani. He is the editor in chief of "International Journal of Swarm Intelligence (IJSI)" published by Inderscience. His primary area of interest is nature inspired optimisation techniques. He has published more than 40 research papers in various international journals/conferences.




Karm Veer Arya is working as an associate professor at ABV-Indian Institute of Information Technology and Management, Gwalior, India. He earned PhD degree in computer science and engineering from Indian Institute of Technology (IIT Kanpur), Kanpur, India. He has more than 20 years of experience to teach the undergraduate and postgraduate classes. He has published more than 75 journal and conference papers in the area of information security, image processing, biometrics, wireless ad hoc networks and soft computing.



Xin-She Yang obtained his DPhil in applied mathematics from the University of Oxford. He then worked at Cambridge University and National Physical Laboratory (UK) as a senior research scientist. Now he is a research professor/reader at Middlesex University London, an adjunct professor at Reykjavik University (Iceland) and a guest professor at Xi'an Polytechnic University (China). He is the IEEE CIS Chair for the task force on business intelligence and knowledge management, Director of International Consortium for Optimisation and Modelling in Science and Industry (iCOMSI), and the editor-in-chief of *International Journal of Mathematical Modelling and Numerical Optimisation* (IJMMNO).

ORCID

Jagdish Chand Bansal  <http://orcid.org/0000-0001-9029-5129>

References

- Akay, B., & Karaboga, D. (2012). A modified artificial bee colony algorithm for real-parameter optimization. *Information Sciences*, 192, 120–142.
- Ali, M., Khompatraporn, C., & Zabinsky, Z. (2005). A numerical evaluation of several stochastic algorithms on selected continuous global optimization test problems. *Journal of Global Optimization*, 31(4), 635–672.
- Banharnsakun, A., Achalakul, T., & Sirinaovakul, B. (2011). The best-so-far selection in artificial bee colony algorithm. *Applied Soft Computing*, 11(2), 2888–2901.
- Bansal, J.C., & Sharma, H. (2012). Cognitive learning in differential evolution and its application to model order reduction problem for single-input single-output systems. *Memetic Computing*, 4(3), 209–229.
- Beyer, H., & Schwefel, H. (2002). Evolution strategies – a comprehensive introduction. *Natural computing*, Springer, 1(1), 3–52.
- Blackmore, S. (1999). *The meme machine*. Oxford: Oxford University Press.
- Brest, J., Zumer, V., & Maucec, M.S. (2006, July). Self-adaptive differential evolution algorithm in constrained real-parameter optimization. In *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on* (pp. 215–222). IEEE.
- Brown, C., Liebovitch, L., & Glendon, R. (2007). Lévy flights in Dobe Ju/'hoansi foraging patterns. *Human Ecology*, 35(1), 129–138.
- Caponio, A., Cascella, G., Neri, F., Salvatore, N., & Sumner, M. (2007). A fast adaptive memetic algorithm for online and offline control design of PMSM drives. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 37(1), 28–41.
- Caponio, A., Neri, F., & Tirronen, V. (2009). Super-fit control adaptation in memetic differential evolution frameworks. *Soft Computing – A Fusion of Foundations, Methodologies and Applications*, 13(8), 811–831.
- Chen, X., Ong, Y., Lim, M., & Tan, K. (2011). A multi-facet survey on memetic computation. *IEEE Transactions on Evolutionary Computation*, 15(5), 591–607.
- Clerc, M. (2012). List based PSO for real problems. Retrieved from <http://clerc.maurice.free.fr/psolist/ListBasedPSO/ListBasedPSO28PSOsite29.pdf>
- Cotta, C., & Neri, F. (2012). Memetic algorithms in continuous optimization. *Handbook of Memetic Algorithms*, 1, 121–134.
- Das, S., & Suganthan, P. (2010). *Problem definitions and evaluation criteria for CEC 2011 competition on testing evolutionary algorithms on real world optimization problems* (Tech. Rep.). Kolkata: Jadavpur University and Singapore: Nanyang Technological University.
- Dasgupta, D. (2006). Advances in artificial immune systems. *IEEE Computational Intelligence Magazine*, 1(4), 40–49.
- Diwold, K., Aderhold, A., Scheidler, A., & Middendorf, M. (2011). Performance evaluation of artificial bee colony optimization and new selection schemes. *Memetic Computing*, 3(3), 149–162.
- Dorigo, M., & Di Caro, G. (1999). Ant colony optimization: A new meta-heuristic. In *Proceedings of the 1999 Congress on Evolutionary Computation, 1999. CEC 99* (Vol. 2, pp. 28–29). Washington, DC: IEEE.
- Eiben, A., & Smith, J. (2003). *Introduction to evolutionary computing*. Berlin: Springer Verlag.
- El-Abd, M. (2011). Performance assessment of foraging algorithms vs. evolutionary algorithms. *Information Sciences*, 182(1), 243–263.
- Fister, I., Fister, I., Jr., & Zumer, J.B. (2012). Memetic artificial bee colony algorithm for large-scale global optimization. In *Evolutionary Computation (CEC), 2012 IEEE Congress on* (pp. 1–8). IEEE.
- Fogel, D., & Michalewicz, Z. (1997). *Handbook of evolutionary computation*. Abingdon: Taylor & Francis.
- Gallo, C., Carballido, J., & Ponzoni, I. (2009). Bihea: A hybrid evolutionary approach for microarray biclustering. *Advances in Bioinformatics and Computational Biology*, 1, 36–47.
- Goh, C., Ong, Y., & Tan, K. (2009). *Multi-objective memetic algorithms* (Vol. 171). Berlin: Springer Verlag.
- Goldberg, D.E., & Holland, J.H. (1988). Genetic algorithms and machine learning. *Machine Learning*, 3(2), 95–99.
- Hooke, R., & Jeeves, T. (1961). 'Direct search' solution of numerical and statistical problems. *Journal of the ACM (JACM)*, 8(2), 212–229.
- Hoos, H., & Stützle, T. (2005). *Stochastic local search: Foundations and applications*. Burlington, MA: Morgan Kaufmann.
- Ishibuchi, H., & Yamamoto, T. (2004). Fuzzy rule selection by multi-objective genetic local search algorithms and rule evaluation measures in data mining. *Fuzzy Sets and Systems*, 141(1), 59–88.
- Ishibuchi, H., Yoshida, T., & Murata, T. (2003). Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling. *IEEE Transactions on Evolutionary Computation*, 7(2), 204–223.
- Kang, F., Li, J., & Ma, Z. (2011). Rosenbrock artificial bee colony algorithm for accurate global optimization of numerical functions. *Information Sciences*, 181(16), 3508–3531.
- Kang, F., Li, J., Ma, Z., & Li, H. (2011). Artificial bee colony algorithm with local search for numerical optimization. *Journal of Software*, 6(3), 490–497.
- Karaboga, D. (2005). *An idea based on honey bee swarm for numerical optimization* (Tech. Rep. TR06). Erciyes Univ. Press.
- Karaboga, D., & Akay, B. (2009). A comparative study of artificial bee colony algorithm. *Applied Mathematics and Computation*, 214(1), 108–132.
- Karaboga, D., & Basturk, B. (2007). Artificial bee colony (ABC) optimization algorithm for solving constrained optimization problems. *Foundations of Fuzzy Logic and Soft Computing*, 1, 789–798.
- Kennedy, J. (2006). Swarm intelligence. *Handbook of Nature-Inspired and Innovative Computing*, 2, 187–219.
- Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. In *IEEE International Conference on Neural Networks, 1995:*

- Proceedings* (Vol. 4, pp. 1942–1948). University of Western Australia, Perth, Western Australia: IEEE.
- Knowles, J., Corne, D., & Deb, K. (2008). *Multiobjective problem solving from nature: From concepts to applications* (Natural computing series). Berlin: Springer.
- Mahdavi, M., Fesanghary, M., & Damangir, E. (2007). An improved harmony search algorithm for solving optimization problems. *Applied Mathematics and Computation*, 188(2), 1567–1579.
- Mezura-Montes, E., & Velez-Koeppel, R. (2010). Elitist artificial bee colony for constrained real-parameter optimization. In *2010 Congress on Evolutionary Computation (CEC)* (pp. 2068–2075). Barcelona: IEEE Service Center.
- Mininno, E., & Neri, F. (2010). A memetic differential evolution approach in noisy optimization. *Memetic Computing*, 2(2), 111–135.
- Moscato, P. (1989). *On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms*. Caltech concurrent computation program (C3P Report No. 826). Pasadena, CA: California Institute of Technology.
- Neri, F., Cotta, C., & Moscato, P. (Eds.). (2012). *Handbook of memetic algorithms: Studies in computational intelligence* (Vol. 379). Berlin: Springer.
- Neri, F., & Tirronen, V. (2009). Scale factor local search in differential evolution. *Memetic Computing Journal*, 1(2), 153–171.
- Nguyen, Q., Ong, Y., & Lim, M. (2009). A probabilistic memetic framework. *IEEE Transactions on Evolutionary Computation*, 13(3), 604–623.
- Ong, Y., & Keane, A. (2004). Meta-Lamarckian learning in memetic algorithms. *IEEE Transactions on Evolutionary Computation*, 8(2), 99–110.
- Ong, Y., Lim, M., & Chen, X. (2010). Research frontier: Memetic computation – past, present & future. *IEEE Computational Intelligence Magazine*, 5(2), 24–31.
- Ong, Y., Lim, M., Zhu, N., & Wong, K. (2006). Classification of adaptive memetic algorithms: A comparative study. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 36(1), 141–152.
- Ong, Y., Nair, P., & Keane, A. (2003). Evolutionary optimization of computationally expensive problems via surrogate modeling. *AIAA journal*, 41(4), 687–696.
- Onwubolu, G., & Babu, B. (2004). *New optimization techniques in engineering*. Berlin: Springer Verlag.
- Passino, K. (2002). Biomimicry of bacterial foraging for distributed optimization and control. *Control Systems Magazine, IEEE*, 22(3), 52–67.
- Pavlyukevich, I. (2007). Lévy flights, non-local search and simulated annealing. *Journal of Computational Physics*, 226(2), 1830–1844.
- Price, K., Storn, R., & Lampinen, J. (2005). *Differential evolution: a practical approach to global optimization*. Berlin: Springer Verlag.
- Ragsdell, K., & Phillips, D. (1976). Optimal design of a class of welded structures using geometric programming. *ASME Journal of Engineering for Industries*, 98(3), 1021–1025.
- Rahnamayan, S., Tizhoosh, H., & Salama, M. (2008). Opposition-based differential evolution. *IEEE Transactions on Evolutionary Computation*, 12(1), 64–79.
- Rao, S., & Rao, S. (2009). *Engineering optimization: Theory and practice*. Manhattan, NY: Wiley.
- Repoussis, P., Tarantilis, C., & Ioannou, G. (2009). Arc-guided evolutionary algorithm for the vehicle routing problem with time windows. *IEEE Transactions on Evolutionary Computation*, 13(3), 624–647.
- Reynolds, A., & Frye, M. (2007). Free-flight odor tracking in *Drosophila* is consistent with an optimal intermittent scale-free search. *PLoS One*, 2(4), e354.
- Richer, J., Goëffon, A., & Hao, J. (2009). A memetic algorithm for phylogenetic reconstruction with maximum parsimony. *Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics*, 1, 164–175.
- Ruiz-Torrubiano, R., & Suárez, A. (2010). Hybrid approaches and dimensionality reduction for portfolio selection with cardinality constraints. *IEEE Computational Intelligence Magazine*, 5(2), 92–107.
- Sandgren, E. (1990). Nonlinear integer and discrete programming in mechanical design optimization. *Journal of Mechanical Design*, 112, 223.
- Shlesinger, M. (2006). Mathematical physics: Search research. *Nature*, 443(7109), 281–282.
- Shlesinger, M.F., Zaslavsky, G.M., & Frisch, U. (1995). Lévy flights and related topics in physics. In *Proceedings of the International Workshop Held at Nice, Levy flights and related topics in Physics*, France, 27–30 June 1994 (Vol. 450, pp. 87–102). Springer.
- Storn, R., & Price, K. (1997). Differential evolution – a simple and efficient adaptive scheme for global optimization over continuous spaces. *Journal of Global Optimization*, 11, 341–359.
- Suganthan, P., Hansen, N., Liang, J., Deb, K., Chen, Y., Auger, A., & Tiwari, S. (2005). Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization. In *CEC 2005* (pp. 1–50). Kanpur: Kanpur Genetic Algorithms Laboratory (KanGAL), Indian Institute of Technology.
- Tang, K., Mei, Y., & Yao, X. (2009). Memetic algorithm with extended neighborhood search for capacitated arc routing problems. *IEEE Transactions on Evolutionary Computation*, 13(5), 1151–1166.
- Valenzuela, J., & Smith, A. (2002). A seeded memetic algorithm for large unit commitment problems. *Journal of Heuristics*, 8(2), 173–195.
- Vesterstrom, J., & Thomsen, R. (2004). A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems. In *Congress on Evolutionary Computation, 2004. CEC2004* (Vol. 2, pp. 1980–1987). Portland: IEEE.
- Wang, H., Wang, D., & Yang, S. (2009). A memetic algorithm with adaptive hill climbing strategy for dynamic optimization problems. *Soft Computing – A Fusion of Foundations, Methodologies and Applications*, 13(8), 763–780.
- Wang, X., Gao, X., & Ovaska, S. (2008). A simulated annealing-based immune optimization method. In *Proceedings of the International and Interdisciplinary Conference on Adaptive Knowledge Representation and Reasoning* (pp. 41–47). Espoo: Porvoo.
- Williamson, D., Parker, R., & Kendrick, J. (1989). The box plot: A simple visual method to interpret data. *Annals of internal medicine*, 110(11), 916–918.
- Yang, X. (2010a). Firefly algorithm, Levy flights and global optimization. *Research and Development in Intelligent Systems XXVI*, 1, 209–218.
- Yang, X. (2010b). *Nature-inspired metaheuristic algorithms* (2nd ed.). Luniver Press. United Kingdom, Springer-Verlag London Ltd, London, Conference held in Granada, Spain.
- Yang, X.S., & Deb, S. (2010). Eagle strategy using Lévy walk and firefly algorithms for stochastic optimization. In *Nature Inspired Cooperative Strategies for Optimization (NICSO 2010): Studies in Computational Intelligence* (Vol. 284, pp. 101–111). Berlin: Springer Verlag.
- Zhu, G., & Kwong, S. (2010). Gbest-guided artificial bee colony algorithm for numerical function optimization. *Applied Mathematics and Computation*, 217(7), 3166–3173.