# Power law-based local search in artificial bee colony

Harish Sharma*, Jagdish Chand Bansal and K.V. Arya

ABV-Indian Institute of Information Technology and Management Gwalior,
Morena Link Road, Gwalior,
Madhya Pradesh-474015, India
E-mail: harish.sharma0107@gmail.com
E-mail: jcbansal@gmail.com
E-mail: kvarya@gmail.com
*Corresponding author

**Abstract:** Artificial bee colony (ABC) optimisation algorithm is relatively a simple and recent population-based probabilistic approach for global optimisation. ABC has been outperformed over some nature inspired algorithms (NIAs) when tested over benchmark as well as real world optimisation problems. 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, there is an enough chance to skip the true solution due to large step sizes. In order to balance the diversity and convergence capability of the ABC, in this paper, a power law-based local search strategy is proposed and integrated with ABC. The proposed strategy is named as power law-based local search in ABC (*PLABC*). In the *PLABC*, new solutions are generated around the best solution and it helps to enhance the exploitation capability of ABC. Further, to improve the exploration capability, numbers of scout bees are increased. The experiments on 24 test problems of different complexities show that the proposed strategy outperforms the basic ABC and recent variants of ABC, namely, Gbest guided ABC (*GABC*), best-so-far ABC (*BSFABC*) and modified ABC in most of the experiments.

**Keywords:** numerical optimisation; swarm intelligence; memetic algorithm; power law-based local search; PLLS; local search.

**Reference** to this paper should be made as follows: Sharma, H., Bansal, J.C. and Arya, K.V. (2014) 'Power law-based local search in artificial bee colony', *Int. J. Artificial Intelligence and Soft Computing*, Vol. 4, Nos. 2/3, pp.164–194.

**Biographical notes:** Harish Sharma received his BTech and MTech degrees in Computer Engineering from Goverment Engineering College, Kota and Rajasthan Technical University, Rajasthan in 2003 and 2009, respectively. He is currently a Research Scholar at ABV – Indian Institute of Information Technology and Management, Gwalior, India.

Jagdish Chand Bansal is an Assistant Professor at ABV-Indian Institute of Information Technology and Management Gwalior. He obtained his PhD in Mathematics from IIT Roorkee. 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.

Karm Veer Arya is working as an Associate Professor at ABV-Indian Institute of Information Technology and Management, Gwalior, India. He obtained his PhD in Computer Science and Engineering from Indian Institute of Technology Kanpur, India. He has more than 20 years of experience teaching undergraduate and postgraduate classes. He has published more than 75 journal and conference papers in the areas of information security, image processing, biometrics, wireless ad hoc networks and soft computing.

## 1 Introduction

Swarm intelligence has become an emerging and interesting area in the field of nature inspired techniques that is used to solve optimisation problems during the past decade. It is based on the collective behaviour of social creatures. Swarm-based optimisation algorithms find solution by collaborative trial and error process. Social creatures utilises 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 non-linear, non-convex or discrete optimisation problems. Previous research (Dorigo and Di Caro, 1999; Kennedy and Eberhart, 1995; Price et al., 2005; Vesterstrom and Thomsen, 2004) have shown that algorithms based on swarm intelligence 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 and Di Caro, 1999), particle swarm optimisation (PSO) (Kennedy and 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 (NIAs).

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 and Akay, 2009). It can be observed that the solution search equation of ABC algorithm is good at exploration but poor at exploitation (Zhu and 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 based on power law-based control parameter, is proposed and incorporated with ABC. In the proposed strategy, the step sizes are controlled by a parameter which is a power law function of iteration counter. The proposed strategy is used for finding the global optima of a unimodal and/or multimodal functions by iteratively reducing the step size in updating process of the candidate solution in the search space within which the optima is known to exist. Further, to improve the diversity of the algorithm, numbers of scout bees are increased. Further, the strategy proposed in this paper is also compared to recent variants of ABC, named, Gbest guided artificial bee colony (*GABC*) algorithm (Zhu and Kwong, 2010), best-so-far artificial bee colony (*BSFABC*) (Banharnsakun et al., 2011) and modified artificial bee colony (*MABC*) (Akay and 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. Power law-based local search (PLLS) is proposed and described in Section 4. In Section 5, PLLS is incorporated with ABC. In Section 6, performance of the proposed strategy is analysed. Finally, in Section 7, 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 et al., 2010). Memetic is synonymous to *memes* which can be described as "instructions for carrying out behavior, stored in brains" (Susan, 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 et al., 2012). A good survey on memetic computing can be found in Ong et al. (2010), Neri et al. (2012), and Chen et al. (2011). Memetic algorithms (MAs) can be seen as an aspect of the realisation or condition-based subset of memetic computing (Chen et al., 2011). The term '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 (Fogel and Michalewicz, 1997; Eiben and Smith, 2003) and neighbourhood-based LS framework (Hoos and Stützle, 2005). Popular examples of population-based methods include genetic algorithms (GAs) and other evolutionary algorithms while Tabu search and simulated annealing (SA) 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 et al., 2012):

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

A 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 the proper balance between exploration and exploitation behaviour of an algorithm, it is highly required to incorporate a LS approach in the basic population-based algorithm to exploit the search region.

Generally, population-based search algorithms like GA (Goldberg, 1989), evolution strategy (ES) (Beyer and Schwefel, 2002), differential evolution (DE) (Price et al., 2005), ACO (Dorigo and Di Caro, 1999), PSO (Kennedy, 2006), artificial immune system

(Dasgupta, 2006), ABC (Karaboga, 2005), etc., are stochastic in nature (Yang, 2010). In recent years, researchers hybridised the LS procedures with the population-based algorithms to improve the exploitation capability of the population-based algorithms (Neri and Tirronen, 2009; Caponio et al., 2009; Mininno and Neri, 2010; Wang et al., 2009; Valenzuela and Smith, 2002; Ishibuchi et al., 2003; Ong et al., 2003). Further, MAs have been successfully applied to solve a wide range of complex optimisation problems like multi-objective optimisation (Knowles et al., 2008; Goh et al., 2009), continuous optimisation (Ong et al., 2003; Ong and Keane, 2004), combinatorial optimisation (Ishibuchi et al., 2003; Tang et al., 2009; Repoussis et al., 2009), bioinformatics (Richer et al., 2009; Gallo et al., 2009), flow shop scheduling (Ishibuchi et al., 2003), scheduling and routing (Brest et al., 2006), machine learning (Ishibuchi and Yamamoto, 2004; Caponio et al., 2007; Ruiz-Torrubiano and 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. Further, they proposed multiple LS procedures during a MA search in the sprit of Lamarckian learning. Further, Ong et al. (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 et al. (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. Further, 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 a LS with ABC. Kang et al. (2011b) proposed a Hooke Jeeves artificial bee colony (HJABC) algorithm for numerical optimisation. In HJABC, authors incorporated a LS technique which is based on Hooke Jeeves (HJ) method (Hooke and Jeeves, 1961) with the basic ABC. Further, 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 1,000 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 et al. (2012) proposed a memetic ABC for large-scale global optimisation. In the proposed approach, ABC is hybridised with two LS heuristics: the Nelder-Mead algorithm (NMA) (Rao and Rao, 2009) and the random walk with direction exploitation (RWDE) (Rao and 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 Cotta and Neri (2012) is applied for balancing the exploration and exploitation.

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

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

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.

Further Kang et al. (2011a) described a Rosenbrock artificial bee colony (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.

Sharma et al. (2012) introduced group social learning in ABC algorithm in which they proposed structured swarm-based learning to balance the exploration and exploitation in the swarm.

## 3 ABC algorithm

The ABC algorithm is relatively recent swarm intelligence-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 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 which 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, 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 phase 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 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 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:

$$x_{ij} = x_{\min j} + rand[0, 1]\left(x_{\max j} - x_{\min j}\right) \tag{2}$$

here $x_{\min j}$ and $x_{\max j}$ are bounds of $x_i$ in $j^{th}$ direction and $rand[0, 1]$ 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 updates equation for $i^{th}$ candidate in this phase is

$$v_{ij} = x_{ij} + \phi_{ij}\left(x_{ij} - x_{kj}\right) \tag{3}$$

here $k \in \{1, 2,\ldots,SN\}$ and $j \in \{1, 2,\ldots,D\}$ are randomly chosen indices. $k$ must be different from $i$. $\phi_{ij}$ 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):

$$prob_i = \frac{fitness_i}{\sum_{i=1}^{SN} fitness_i} \tag{4}$$

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:

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

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 and Akay, 2009).

**Algorithm 1** ABC 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.

## 4 Power law-based local search

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.

The LS algorithms can be seen as a population-based stochastic algorithms, where main task is to exploit the available knowledge about a problem. Therefore, steps sizes play an important role in exploiting the identified region. Hence, in this paper, a LS strategy, based on power law, is proposed and named *PLLS*. In the proposed strategy, the step sizes, require to update an individual, is iteratively decreased to exploit the search area in the vicinity of the best candidate solution. In the proposed search strategy, the step sizes are forced to decrease using a parameter $u$ which is a power law function of iteration counter. The position update equation of an $i^{\text{th}}$ individual is shown in equation (6):

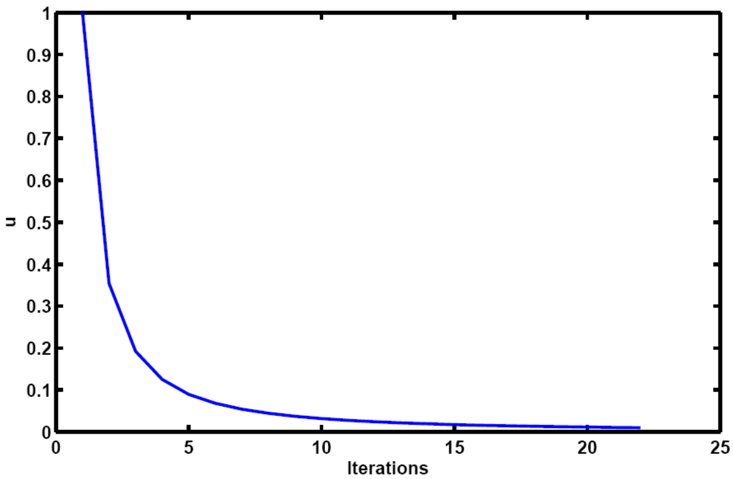
$$x_{ij}(t+1) = x_{ij}(t) + \alpha\beta \operatorname{sign}\left( rand[0, 1] - \frac{1}{2} \right) \times u(t), \tag{6}$$

here $\alpha$ is the step size control parameter and $\beta$ is the social learning component which depends upon the global search algorithm. The $\operatorname{sign}\left( rand[0, 1] - \frac{1}{2} \right)$ essentially provides a random sign or direction while $u$ is a parameter which is a power law function of iteration counter ($t$) and computed using equation (7):

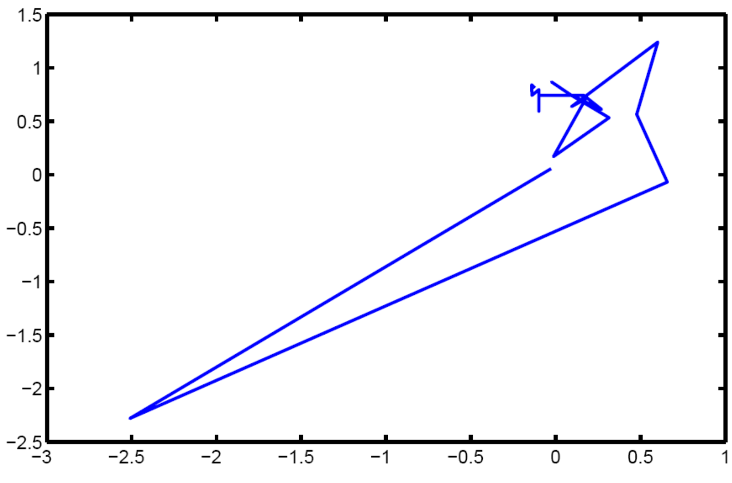
$$u(t) = t^{-\lambda}, (1 < \lambda \leq 3), \tag{7}$$

Here $t$ is the iteration counter. Due to equation (7), as the iteration counter increases, $u$ decreases results the decrease in step length. In this way, $u$ calculated through equation (7), helps to exploit the search space through iterations. The step size to update a candidate solution is $\left(\beta \times \alpha \text{sign}\left(rand[0,1] - \frac{1}{2}\right) \times u\right)$ a random walk process with a power-law distribution with decreasing step-length and having a heavy tail. Figure 1 shows a power law graph, being used to demonstrate $u$ and an example of the *PLLS* random walk used to update an individual, using ABC as a global search algorithm ($\beta = (x_{ij} - x_{kj})$), in two dimension search space for *Beale function* ($f_9$). It is clear from Figure 1(b) that the steps sizes provided by *PLLS* are stochastic and decreasing in nature and therefore *PLLS* is expected to provide a better exploitation process in basic ABC.

**Figure 1** (a) Power-law graph, being used to demonstrate $u$ (b) Position's of an individual based on *PLLS* in two dimension search space (see online version for colours)




(a)



(b)

The pseudo-code of the proposed PLLS is shown in Algorithm 2. In Algorithm 2, $\epsilon$ determines the termination of LS.

**Algorithm 2** PLLS strategy

```
Input optimization function Minf(x), α and λ;
Select the best solution x_best in the swarm;
Initialize a counter t = 1;
while (u > ϵ) do
    Compute u(t) = t^(−λ);
    Generate a new solution x_new using u(t) and α by equation (6).
    Calculate f(x_new).
    if f(x_new) < f(x_best) then
        x_best = x_new;
    end if
    t = t + 1;
end while
```

## 5 Power law-based local search in artificial bee colony

Exploration and exploitation are the two important characteristics of the population-based optimisation algorithms such as GA (Goldberg, 1989), PSO (Kennedy and Eberhart, 1995), DE (Storn and 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 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) 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 $\phi_{ij}$, 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 a LS algorithm with the ABC algorithm. Therefore, in this paper, to balance the diversity and convergence ability of ABC, four modifications are proposed:

1. To enhance the exploitation capability of ABC, *PLLS* strategy (described in Section 3) 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 *PLLS* strategy, in case of large step sizes, can search within the area that is jumped by the basic ABC.
2. To enhance the exploration capability, the number 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.
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 $x_i$:

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

   here $[a, b]$ is the shrunken search space.
4. In the basic ABC, the food sources are updated, as shown in equation (3), based on the random step size. Inspired by PSO (Kennedy and Eberhart, 1995) and GABC (Zhu and 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 and Kwong, 2010):

   $$x_{ij}(t+1) = x_{ij}(t) + \phi_{ij}\left(x_{ij}(t) - x_{kj}(t)\right) + \psi_{ij}\left(x_{bestj}(t) - x_{ij}(t)\right),$$

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

In this paper, the *PLLS* strategy is incorporate with the basic ABC to improve the exploitation capability. In the proposed LS strategy, the position update equation of an $i^{\text{th}}$ food source is shown in equation (8).

$$x_{ij}(t+1) = x_{ij}(t) + \left(x_{ij}(t) - x_{kj}(t)\right)\alpha \operatorname{sign}\left(rand[0,1] - \frac{1}{2}\right) \times u(t), \tag{8}$$

here, symbols have their usual meanings, $\beta = (x_{ij} - x_{kj})$ is the social learning component of the ABC algorithm and $i^{\text{th}}$ solution is the best solution in the current swarm. The proposed strategy in ABC is hereby, named as power law-based local search in artificial bee colony (*PLABC*). The pseudo-code of the proposed *PLLS* strategy with ABC is shown in Algorithm 3. In *PLLS*, only the best particle of the current swarm updates itself in its neighbourhood.

**Algorithm 3** PLLS strategy with ABC

Input optimization function *Minf*($x$), $\alpha$ and $\lambda$;

Select the best solution $x_{best}$ in the swarm;

Initialize a counter $t = 1$;
**while** ($u > \epsilon$) **do**
  Compute $u(t) = t^{-\lambda}$;
  Generate a new solution $x_{new}$ using $u(t)$ and $\alpha$ by Algorithm 4.
  Calculate $f(x_{new})$.
  **if** $f(x_{new}) < f(x_{best})$ **then**
    $x_{best} = x_{new}$;
  **end if**
  $t = t + 1$;
**end while**

In Algorithms 3 and 4, $\epsilon$ 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 5.2 for details of these parameter settings.

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

**Algorithm 4** New solution generation

Input $u$, $\alpha$ and best solution $x_{best}$;
**for** $j = 1$ to $D$ **do**
  **if** $U(0, 1) > p_r$ **then**

$$x_{newj} = x_{bestj} + \left(x_{bestj} - x_{kj}\right) \times \alpha \operatorname{sign}\left(rand[0,1] - \frac{1}{2}\right) \times u;$$

  **else**
    $x_{newj} = x_{bestj}$;
  **end if**
**end for**
Return $x_{new}$

**Algorithm 5** Power law-based local search in artificial bee colony

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 *PLLS* strategy using Algorithm 3.
**end while**
Print best solution.

**Table 1** Test problems

| *Test problem* | *Search range* | *Optimum value* | *D* | *Acceptable error* |
|---|---|---|---|---|
| $f_1(x)=\sum_{i=1}^{D}x_i^2-0.1\left(\sum_{i=1}^{D}\cos 5\pi x_i\right)+0.1D$ | [–1, 1] | $f(\vec{0})=-0.1D$ | 30 | 1.0E-05 |
| $f_2(x)=-\left(\exp\left(-0.5\sum_{i=1}^{D}x_i^2\right)\right)+1$ | [–1, 1] | $f(\vec{0})=-1$ | 30 | 1.0E-05 |
| $f_3(x)=\sum_{i=1}^{D}x_i^2+\left(\sum_{i=1}^{D}\frac{ix_i}{2}\right)^2+\left(\sum_{i=1}^{D}\frac{ix_1}{2}\right)^4$ | [–5.12, 5.12] | $f(\vec{0})=0$ | 30 | 1.0E-02 |
| $f_4(x)=1-\cos\left(2\pi\sqrt{\sum_{i=1}^{D}x_i^2}\right)+0.1\left(\sum_{i=1}^{D}x_i^2\right)$ | [–100, 100] | $f(\vec{0})=0$ | 30 | 1.0E-01 |
| $f_5(x)=\sum_{i=1}^{n}ix_i^4+random[0,1)$ | [–1.28, 1.28] | $f(\vec{0})=0$ | 30 | 1.0 |
| $f_6(x)=-\sum_{i=1}^{D-1}\left(\exp\left(\frac{-\left(x_i^2+x_{i+1}^2+0.5x_ix_{i+1}\right)}{8}\right)\times I\right)$ where $I=\cos\left(4\sqrt{x_i^2+x_{i+1}^2+0.5x_ix_{i+1}}\right)$ | [–5, 5] | $f(\vec{0})=-D+1$ | 10 | 1.0E-05 |
| $f_7(x)=\sum_{i=1}^{D}(x_i-1)^2-\sum_{i=2}^{D}x_ix_{i-1}$ | [$-D^2$, $D^2$] | $f(\vec{0})=-(D(D+4)(D-1))/6.0$ | 10 | 1.0E-01 |

**Table 1** Test problems (continued)

| *Test problem* | *Search range* | *Optimum value* | *D* | *Acceptable error* |
|---|---|---|---|---|
| $f_8(x) = \sum_{i=1}^{D} \sum_{j=1}^{i} x_j^2$ | [–65.536, 65.536] | $f(\vec{0}) = 0$ | 30 | 1.0E-05 |
| $f_9(x) = \left[1.5 - x_1(1 - x_2)\right]^2 + \left[2.25 - x_1(1 - x_2^2)\right]^2 + \left[2.625 - x_1(1 - x_3^2)\right]^2$ | [–4.5, 4.5] | $f(3, 0.5) = 0$ | 2 | 1.0E-05 |
| $f_{10}(x) = 100(x_2 - x_2^1)^2 + (1 - x_1)^2 + 90(x_4 - x_2^3)^2 + (1 - x_3)^2$ $+ 10.1\left[(x_2 - 1)^2 + (x_4 - 1)^2\right] + 19.8(x_2 - 1)(x_4 - 1)$ | [–10, 10] | $f(\vec{1}) = 0$ | 4 | 1.0E-05 |
| $f_{11}(x) = \sum_{i=1}^{11} \left[a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4}\right]^2$ | [–5, 5] | $f(0.1928, 0.1908, 0.1231, 0.1357) = 3.07E\text{-}04$ | 4 | 1.0E-05 |
| $f_{12}(x) = \sum_{i=1}^{D-1} \left(100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2\right) + f_{bias}$, $z = x - o + 1, x = [x_1, x_2, \ldots, x_D], o = [o_1, o_2, \ldots, o_D]$ | [–100, 100] | $f(o) = f_{bias} = 390$ | 10 | 1.0E-01 |
| $f_{13}(x) = \sum_{i=1}^{D} z_i^2 + f_{bias}$, $z = x - o, x = [x_1, x_2, \ldots, x_D], o = [o_1, o_2, \ldots, o_D]$ | [–100, 100] | $f(o) = f_{bias} = -450$ | 10 | 1.0E-05 |

**Table 1** Test problems (continued)

| *Test problem* | *Search range* | *Optimum value* | *D* | *Acceptable error* |
|---|---|---|---|---|
| $f_{14}(x)=\sum_{i=1}^{D}\left(z_i^2-10\cos(2\pi z_i)+10\right)+f_{bias}$, $z=(x-o)$, $x=(x_1,x_2,\ldots,x_D)$, $o=(o_1,o_2,\ldots,o_D)$ | [–5, 5] | $f(o)=f_{bias}=-330$ | 10 | 1.0E-02 |
| $f_{15}(x)=\sum_{i=1}^{D}\left(\sum_{j=1}^{i}z_j\right)^2+f_{bias}$, $z=x-o$, $x=[x_1,x_2,\ldots,x_D]$, $o=[o_1,o_2,\ldots,o_D]$ | [–100, 100] | $f(o)=f_{biais}=-450$ | 10 | 1.0E-05 |
| $f_{16}(x)=\sum_{i=1}^{D}\frac{z_i^2}{4,000}-\prod_{i=1}^{D}\cos\left(\frac{z_i}{\sqrt{i}}\right)+1+f_{bias}$, $z=(x-o)$, $x=[x_1,x_2,\ldots,x_D]$, $o=[o_1,o_2,\ldots,o_D]$ | [–600, 600] | $f(o)=f_{bias}=-180$ | 10 | 1.0E-05 |
| $f_{17}(x)=-20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D}z_{2i}}\right)-\exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi z_i)\right)+20+e+f_{bias}$, $z=(x-o)$, $x=(x_1,x_2,\ldots,x_D)$, $o=(o_1,o_2,\ldots,o_D)$ | [–32, 32] | $f(o)=f_{bias}=-140$ | 10 | 1.0E-05 |

**Table 1** Test problems (continued)

| *Test problem* | *Search range* | *Optimum value* | *D* | *Acceptable error* |
|---|---|---|---|---|
| $f_{18}(x) = \left(1+(x_1+x_2+1)^2(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2)\right)$ $\left(30+(2x_1-3x_2)^2(18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2)\right)$ | [–2, 2] | $f(0, -1) = 3$ | 2 | 1.0E-14 |
| $f_{19}(x) = -\cos x_1 \cos x_2 e^{\left((-(x_1-\pi)^2-(x_2-\pi)^2)\right)}$ | [–10, 10] | $f(\pi, \pi) = -1$ | 2 | 1.0E-13 |
| $f_{20}(x) = 10^5 x_1^2 + x_2^2 - (x_1^2+x_2^2)^2 + 10^{-5}(x_1^2+x_2^2)^4$ | [–20, 20] | $f(0, 15) = f(0, -15) = -24{,}777$ | 2 | 5.0E-01 |
| $f_{21}(x) = \sin(x_1+x_2) + (x_1+x_2)^2 - \frac{3}{2}x_1 + \frac{5}{2}x_2 + 1$ | $-1.5 \le x_1 \le 4$, $-3 \le x_2 \le 3$ | $f(-0.547, -1.547) = -1.9133$ | 30 | 1.0E-04 |
| $f_{22}(x) = \sum_{i=1}^{5}\left(\frac{x_1x_3t_i}{1+x_1t_i+x_2v_i} - y_i\right)^2$ | [–10, 10] | $f(3.13, 15.16, 0.78) = 0.4\text{E-}04$ | 3 | 1.0E-03 |
| $f_{23}(x) = -\left(\sum_{i=1}^{5} i\cos((i+1)x_1+1)\right) - \sum_{i=1}^{5} i\cos((i+1)x_2+1)$ | [–10, 10] | $f(7.0835, 4.8580) = -186.7309$ | 2 | 1.0E-05 |
| $f_{24}(x) = \sum_{i=1}^{D} 5ix_i^2$ | [–5.12, 5.12] | $f(x) = 0; x(i) = 5i, i = 1 : D$ 30 | | 1.0E-15 |

## 6 Experimental results and discussion

### 6.1 Test problems under consideration

In order to analyse the performance of *PLABC*, 24 different global optimisation problems ($f_1$ to $f_{24}$) are selected (listed in Table 1). These are continuous optimisation problems and have different degrees of complexity and multimodality. Test problems $f_1$ to $f_{11}$ and $f_{18}$ to $f_{24}$ are taken from Ali et al. (2005) and test problems $f_{12}$ to $f_{17}$ are taken from Suganthan (2005) with the associated offset values.

### 6.2 Experimental setting

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

- Colony size $NP = 50$ (Diwold et al., 2011; El-Abd, 2011).
- $\phi_{ij} = rand[-1, 1]$.
- Number of food sources $SN = NP / 2$.
- $Limit = 1{,}500$ (Karaboga and Basturk, 2007; Akay and Karaboga, 2012).
- 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 and Kwong, 2010).
- The value of $\alpha = 2$ and $\lambda = 1.5$ are to be set based on the empirical experiments.
- Value of termination criteria in *PLLS* is set to be $\epsilon = 0.01$.
- 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 *PLABC*, its sensitivity with respect to different values of $p_r$ in the range [0.2, 1], is examined in the Figure 2. It can be observed from Figure 2 that the test problems are very sensitive towards $p_r$ and value 0.4 gives comparatively better results. Therefore, $p_r = 0.4$ is selected for the experiments in this paper.

### 6.3 Results comparison

Numerical results with experimental setting of Subsection 5.6 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 *PLABC* outperforms in terms of reliability, efficiency and accuracy as compare to the basic *ABC*, *GABC*, *BSFABC* and *MABC*. Some more intensive analyses based on performance indices and boxplots have been carried out for results of *ABC* and its variants.

**Figure 2** Effect of parameter $p_r$ on success rate (see online version for colours)

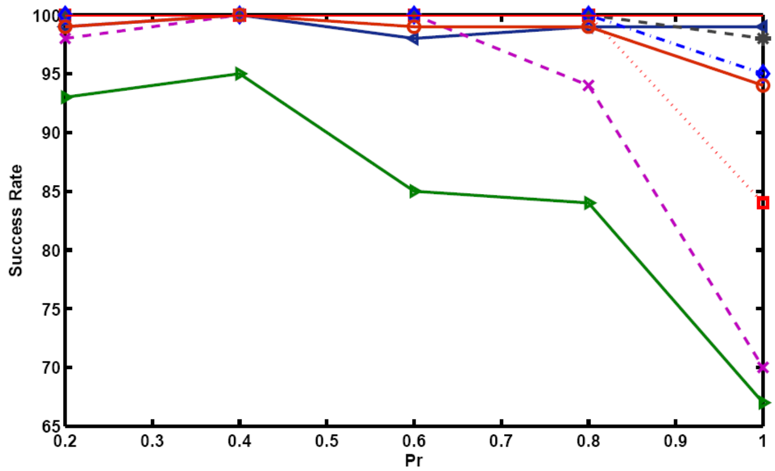



**Table 2** Comparison of the results of test problems

| *Test function* | *Algorithm* | *SD* | *ME* | *AFE* | *SR* |
|---|---|---|---|---|---|
| $f_1$ | ABC | 2.29E-06 | 7.78E-06 | 23,267.5 | 100 |
| | PLABC | 6.10E-07 | 9.43E-06 | 16,401.77 | 100 |
| | GABC | 2.21E-06 | 7.64E-06 | 15,414.5 | 100 |
| | BSFABC | 2.20E-06 | 7.21E-06 | 31,618.5 | 100 |
| | MABC | 7.63E-07 | 9.18E-06 | 22,889 | 100 |
| $f_2$ | ABC | 2.62E-06 | 7.26E-06 | 16,967 | 100 |
| | PLABC | 6.40E-07 | 9.41E-06 | 10,317.44 | 100 |
| | GABC | 1.53E-06 | 8.27E-06 | 11,728.5 | 100 |
| | BSFABC | 2.17E-06 | 7.63E-06 | 18,737 | 100 |
| | MABC | 7.37E-07 | 9.13E-06 | 16,736 | 100 |
| $f_3$ | ABC | 1.52E+01 | 9.73E+01 | 200,000 | 0 |
| | PLABC | 2.26E-02 | 8.07E-02 | 200,016.07 | 0 |
| | GABC | 1.89E+01 | 9.73E+01 | 200,000.01 | 0 |
| | BSFABC | 1.22E+01 | 8.49E+01 | 200,000 | 0 |
| | MABC | 1.02E-01 | 1.46E-01 | 200,005.52 | 0 |
| $f_4$ | ABC | 6.25E-02 | 9.56E-01 | 149,071.35 | 68 |
| | PLABC | 3.04E-02 | 9.34E-01 | 19,392.79 | 100 |
| | GABC | 3.38E-02 | 9.32E-01 | 75,922.34 | 98 |
| | BSFABC | 6.58E-02 | 9.53E-01 | 184,747.81 | 74 |
| | MABC | 3.46E-02 | 9.31E-01 | 27,739.5 | 100 |

**Table 2** Comparison of the results of test problems (continued)

| *Test function* | *Algorithm* | *SD* | *ME* | *AFE* | *SR* |
|---|---|---|---|---|---|
| $f_5$ | ABC | 5.63E-01 | 1.17E+01 | 200,038.43 | 0 |
| | PLABC | 4.18E-01 | 9.03E+00 | 200,029.42 | 0 |
| | GABC | 5.20E-01 | 1.05E+01 | 200,016.25 | 0 |
| | BSFABC | 5.03E-01 | 1.00E+01 | 200,031.51 | 0 |
| | MABC | 4.51E-01 | 9.86E+00 | 200,014.41 | 0 |
| $f_6$ | ABC | 1.06E-01 | 1.84E-02 | 83,315.84 | 93 |
| | PLABC | 7.82E-02 | 7.87E-03 | 57,398.94 | 99 |
| | GABC | 2.44E-06 | 6.92E-06 | 47,798.14 | 100 |
| | BSFABC | 3.00E-01 | 1.19E-01 | 127,844.86 | 78 |
| | MABC | 1.37E-06 | 8.53E-06 | 68,974.05 | 100 |
| $f_7$ | ABC | 7.34E-01 | 9.40E-01 | 199,149.67 | 1 |
| | PLABC | 1.47E-02 | 8.39E-02 | 16,118.72 | 100 |
| | GABC | 9.76E-01 | 1.10E+00 | 190,113.42 | 12 |
| | BSFABC | 5.52E+00 | 4.47E+00 | 200,022.66 | 0 |
| | MABC | 1.02E-01 | 1.15E-01 | 131,932.63 | 94 |
| $f_8$ | ABC | 1.96E-06 | 8.03E-06 | 27,967 | 100 |
| | PLABC | 6.23E-07 | 9.43E-06 | 23,540.95 | 100 |
| | GABC | 1.98E-06 | 8.00E-06 | 19,519.5 | 100 |
| | BSFABC | 2.40E-06 | 6.87E-06 | 49,294 | 100 |
| | MABC | 7.38E-07 | 9.11E-06 | 33,057.5 | 100 |
| $f_9$ | ABC | 1.40E-06 | 8.66E-06 | 16,391.49 | 100 |
| | PLABC | 2.97E-06 | 5.22E-06 | 981.26 | 100 |
| | GABC | 2.88E-06 | 5.31E-06 | 7,862.92 | 100 |
| | BSFABC | 4.20E-05 | 1.55E-05 | 47,809.44 | 95 |
| | MABC | 3.06E-06 | 4.73E-06 | 10,092.1 | 100 |
| $f_{10}$ | ABC | 1.10E-01 | 1.54E-01 | 200,023.98 | 0 |
| | PLABC | 2.15E-03 | 7.43E-03 | 13,343.06 | 100 |
| | GABC | 1.61E-02 | 1.95E-02 | 159,885.7 | 39 |
| | BSFABC | 2.19E-02 | 2.48E-02 | 158,412.54 | 42 |
| | MABC | 1.05E-02 | 1.50E-02 | 151,722.6 | 49 |
| $f_{11}$ | ABC | 7.30E-05 | 1.88E-04 | 186,761.63 | 15 |
| | PLABC | 8.33E-05 | 9.79E-05 | 19,205.54 | 99 |
| | GABC | 3.07E-05 | 8.64E-05 | 93,221.31 | 89 |
| | BSFABC | 8.11E-05 | 1.39E-04 | 147,317.48 | 57 |
| | MABC | 7.45E-05 | 2.02E-04 | 187,855.66 | 10 |

**Table 2** Comparison of the results of test problems (continued)

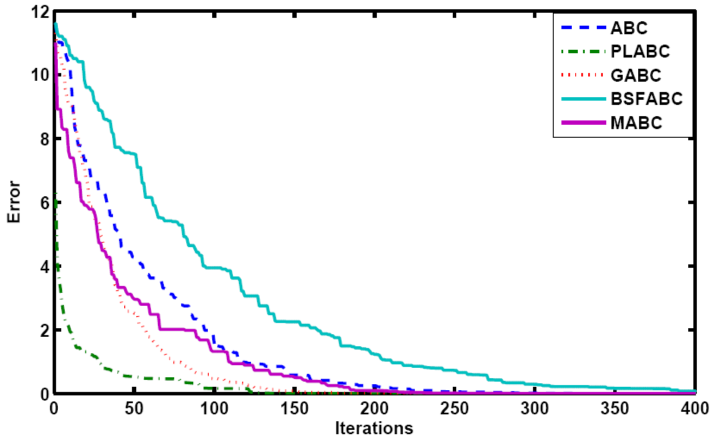
| *Test function* | *Algorithm* | *SD* | *ME* | *AFE* | *SR* |
|---|---|---|---|---|---|
| $f_{12}$ | ABC | 1.62E+00 | 7.69E-01 | 177,169.99 | 19 |
| | PLABC | 5.45E-01 | 1.73E-01 | 82,793.2 | 97 |
| | GABC | 6.10E-02 | 9.07E-02 | 110,189.39 | 92 |
| | BSFABC | 4.60E+00 | 2.48E+00 | 182,906.62 | 16 |
| | MABC | 8.83E-01 | 6.98E-01 | 171,598.96 | 30 |
| $f_{13}$ | ABC | 2.67E-06 | 6.77E-06 | 9,023.5 | 100 |
| | PLABC | 1.31E-06 | 8.68E-06 | 5,806.9 | 100 |
| | GABC | 2.19E-06 | 7.23E-06 | 5,518 | 100 |
| | BSFABC | 2.10E-06 | 7.29E-06 | 18,154 | 100 |
| | MABC | 1.64E-06 | 7.92E-06 | 8,651 | 100 |
| $f_{14}$ | ABC | 1.22E+01 | 8.72E+01 | 200,011.66 | 0 |
| | PLABC | 1.89E+01 | 1.25E+02 | 200,024.78 | 0 |
| | GABC | 1.10E+01 | 8.43E+01 | 200,007.19 | 0 |
| | BSFABC | 1.66E+01 | 1.25E+02 | 200,036.07 | 0 |
| | MABC | 9.92E+00 | 8.21E+01 | 200,015.17 | 0 |
| $f_{15}$ | ABC | 2.89E+03 | 1.15E+04 | 200,027.55 | 0 |
| | PLABC | 8.02E+03 | 2.17E+04 | 200,047.72 | 0 |
| | GABC | 2.73E+03 | 1.09E+04 | 200,015.94 | 0 |
| | BSFABC | 7.90E+03 | 2.85E+04 | 200,036.66 | 0 |
| | MABC | 3.01E+03 | 1.05E+04 | 200,020.3 | 0 |
| $f_{16}$ | ABC | 3.00E-03 | 1.09E-03 | 74,162.28 | 88 |
| | PLABC | 1.56E-03 | 2.28E-04 | 61,308.12 | 98 |
| | GABC | 3.01E-06 | 4.91E-06 | 38,682.29 | 100 |
| | BSFABC | 6.30E-03 | 4.83E-03 | 111,954.86 | 58 |
| | MABC | 1.89E-03 | 5.24E-04 | 88,708.91 | 92 |
| $f_{17}$ | ABC | 2.04E-06 | 7.51E-06 | 16,516.5 | 100 |
| | PLABC | 7.97E-07 | 9.21E-06 | 10,383.55 | 100 |
| | GABC | 1.41E-06 | 8.31E-06 | 9,327 | 100 |
| | BSFABC | 1.58E-06 | 8.13E-06 | 31,237 | 100 |
| | MABC | 8.87E-07 | 8.99E-06 | 14,197.54 | 100 |
| $f_{18}$ | ABC | 8.31E-06 | 1.38E-06 | 93,469.31 | 72 |
| | PLABC | 4.42E-15 | 5.59E-15 | 4,490.89 | 100 |
| | GABC | 4.67E-15 | 5.24E-15 | 3,910.98 | 100 |
| | BSFABC | 4.76E-15 | 6.56E-15 | 13,124.91 | 100 |
| | MABC | 4.30E-15 | 4.68E-15 | 11,969.73 | 100 |

**Table 2** Comparison of the results of test problems (continued)

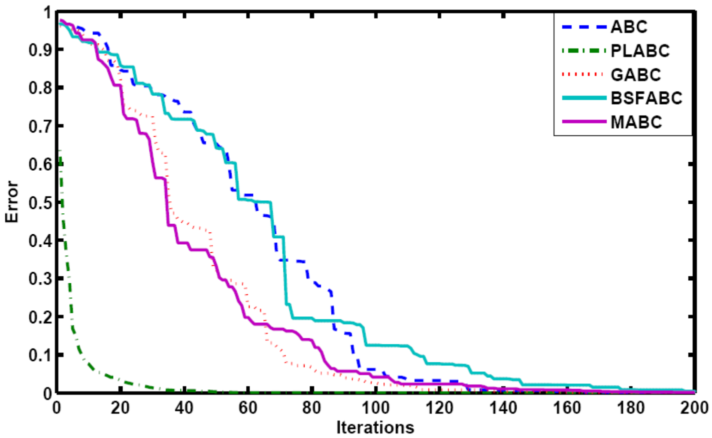
| *Test function* | *Algorithm* | *SD* | *ME* | *AFE* | *SR* |
|---|---|---|---|---|---|
| $f_{19}$ | ABC | 8.48E-05 | 2.67E-05 | 184,517.49 | 15 |
| | PLABC | 2.86E-14 | 5.04E-14 | 13,912.06 | 100 |
| | GABC | 1.34E-13 | 5.74E-14 | 45,418.68 | 99 |
| | BSFABC | 3.17E-14 | 4.43E-14 | 4,682.16 | 100 |
| | MABC | 1.25E-03 | 7.87E-04 | 200,026.53 | 0 |
| $f_{20}$ | ABC | 5.43E-03 | 4.90E-01 | 1,394.02 | 100 |
| | PLABC | 5.94E-03 | 4.90E-01 | 674.29 | 100 |
| | GABC | 5.21E-03 | 4.89E-01 | 736 | 100 |
| | BSFABC | 5.20E-03 | 4.92E-01 | 2,768.27 | 100 |
| | MABC | 5.20E-03 | 4.90E-01 | 2,315.54 | 100 |
| $f_{21}$ | ABC | 6.65E-06 | 8.93E-05 | 1,179.03 | 100 |
| | PLABC | 6.81E-06 | 9.01E-05 | 311.5 | 100 |
| | GABC | 6.58E-06 | 8.78E-05 | 611.5 | 100 |
| | BSFABC | 6.81E-06 | 8.80E-05 | 1,014.51 | 100 |
| | MABC | 6.62E-06 | 8.93E-05 | 1,745.22 | 100 |
| $f_{22}$ | ABC | 2.91E-06 | 1.95E-03 | 23,897.62 | 100 |
| | PLABC | 2.89E-06 | 1.95E-03 | 2,403.49 | 100 |
| | GABC | 2.86E-06 | 1.95E-03 | 4,497.41 | 100 |
| | BSFABC | 2.88E-06 | 1.95E-03 | 16,033.86 | 100 |
| | MABC | 2.56E-06 | 1.95E-03 | 8,535.54 | 100 |
| $f_{23}$ | ABC | 5.90E-06 | 5.30E-06 | 4,930.84 | 100 |
| | PLABC | 5.53E-06 | 5.13E-06 | 2,292.61 | 100 |
| | GABC | 5.77E-06 | 5.14E-06 | 2,495.11 | 100 |
| | BSFABC | 5.86E-06 | 5.17E-06 | 9,367.23 | 100 |
| | MABC | 5.50E-06 | 4.82E-06 | 30,951.69 | 100 |
| $f_{24}$ | ABC | 1.62E-16 | 8.06E-16 | 59,873 | 100 |
| | PLABC | 3.81E-17 | 9.60E-16 | 45,858.92 | 100 |
| | GABC | 1.08E-16 | 8.73E-16 | 38,699.5 | 100 |
| | BSFABC | 2.39E-16 | 7.14E-16 | 71,431.5 | 100 |
| | MABC | 7.83E-17 | 9.18E-16 | 59,690 | 100 |

Figure 3 shows the convergence characteristics in terms of the error of the median run of each algorithm for functions on which *ABC*, *PLABC*, *GABC*, *BSFABC* and *MABC* algorithms achieved 100% success rate within the specified maximum function evaluations (to carry out fair comparison of convergence rate). It can be observed that the convergence of *PLABC* is relatively better than *ABC*, *GABC*, *BSFABC* and *MABC*.

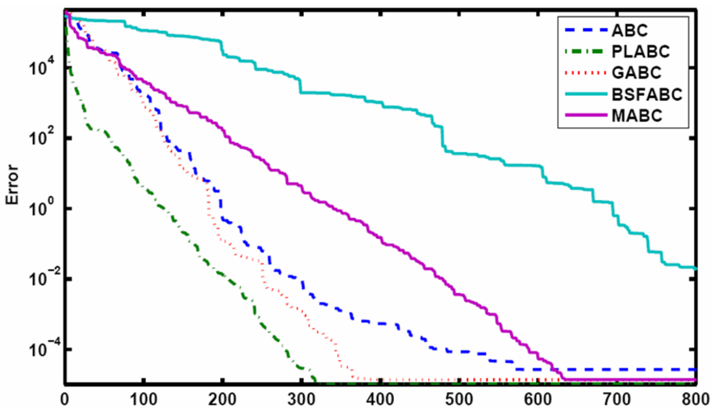
**Figure 3** Convergence characteristics of *ABC*, *PLABC*, *GABC*, *BSFABC* and *MABC* for functions (a) $f_1$, (b) $f_2$, (c) $f_8$, (d) $f_{13}$, (e) $f_{17}$, (f) $f_{20}$, (g) $f_{21}$, (h) $f_{22}$, (i) $f_{23}$, (j) $f_{24}$ (see online version for colours)




(a)

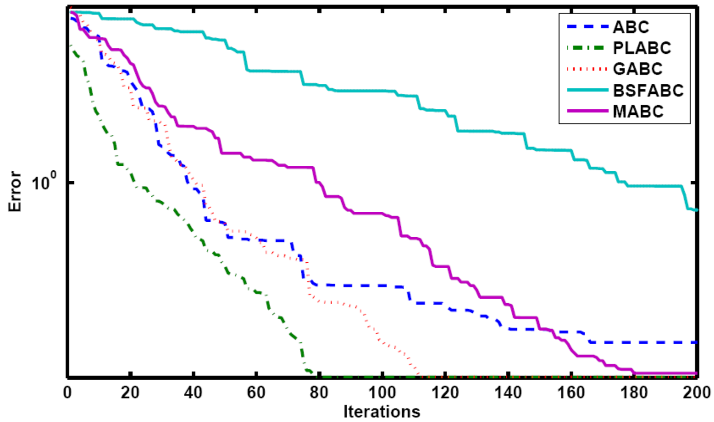



(b)

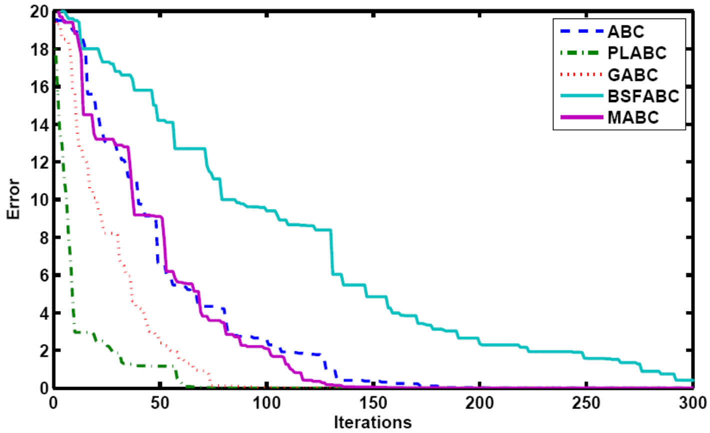



(c)

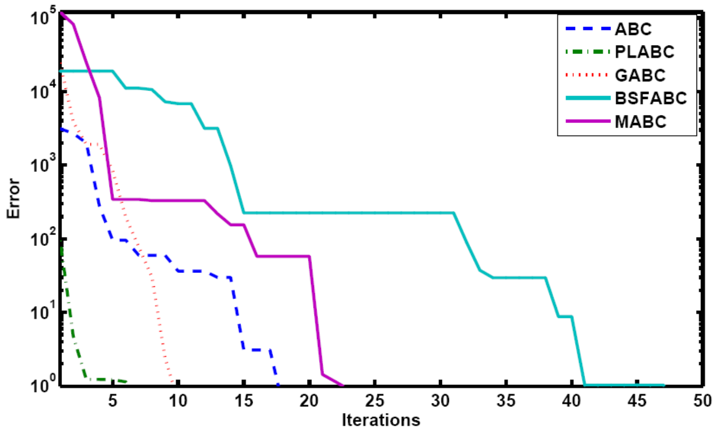
**Figure 3** Convergence characteristics of *ABC*, *PLABC*, *GABC*, *BSFABC* and *MABC* for functions (a) $f_1$, (b) $f_2$, (c) $f_8$, (d) $f_{13}$, (e) $f_{17}$, (f) $f_{20}$, (g) $f_{21}$, (h) $f_{22}$, (i) $f_{23}$, (j) $f_{24}$ (continued) (see online version for colours)




(d)

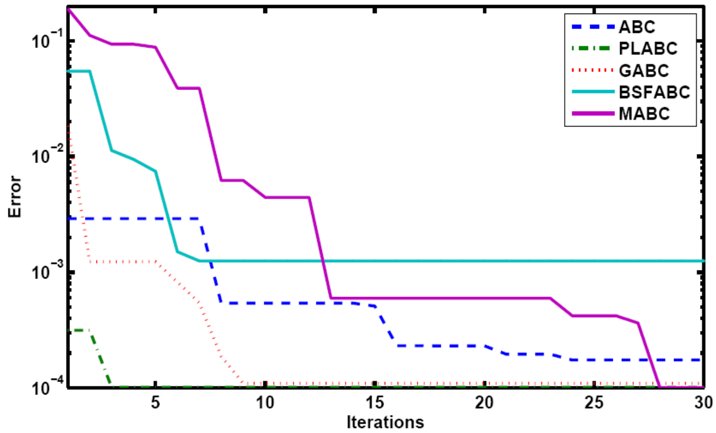



(e)

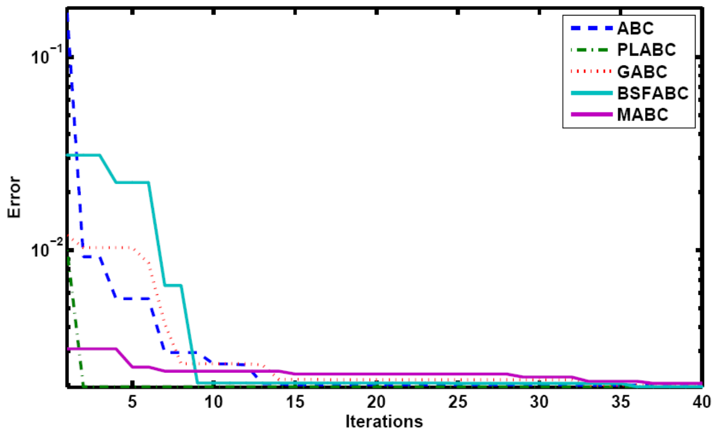



(f)

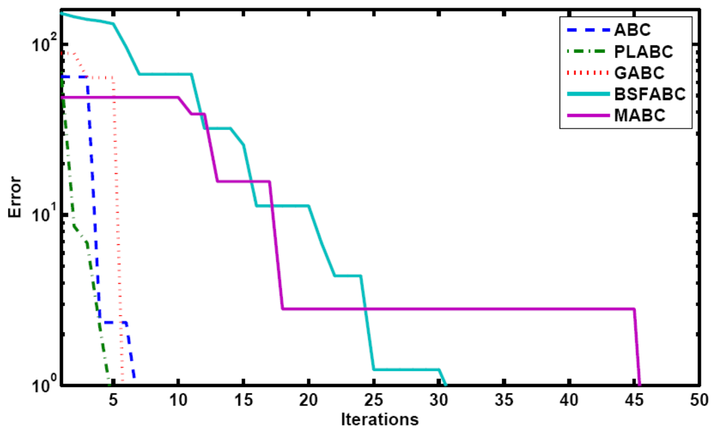
**Figure 3** Convergence characteristics of *ABC*, *PLABC*, *GABC*, *BSFABC* and *MABC* for functions (a) $f_1$, (b) $f_2$, (c) $f_8$, (d) $f_{13}$, (e) $f_{17}$, (f) $f_{20}$, (g) $f_{21}$, (h) $f_{22}$, (i) $f_{23}$, (j) $f_{24}$ (continued) (see online version for colours)




(g)

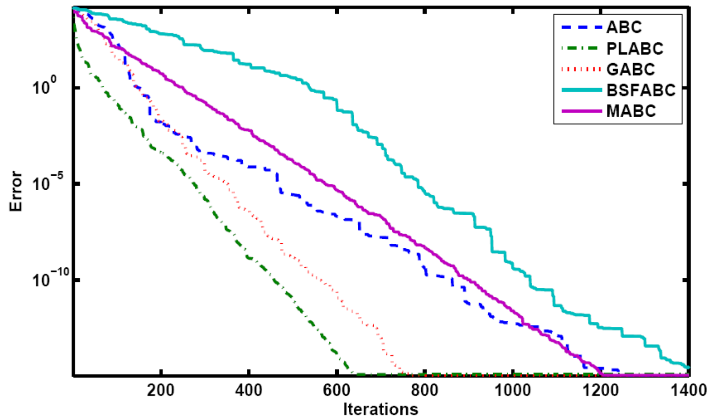



(h)




(i)

**Figure 3** Convergence characteristics of *ABC*, *PLABC*, *GABC*, *BSFABC* and *MABC* for functions (a) $f_1$, (b) $f_2$, (c) $f_8$, (d) $f_{13}$, (e) $f_{17}$, (f) $f_{20}$, (g) $f_{21}$, (h) $f_{22}$, (i) $f_{23}$, (j) $f_{24}$ (continued) (see online version for colours)




(j)

*PLABC*, *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 PLABC 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 *PLABC* over *ABC*, *BSFABC*, *MABC*.

**Table 3** Summary of Table 2 outcome

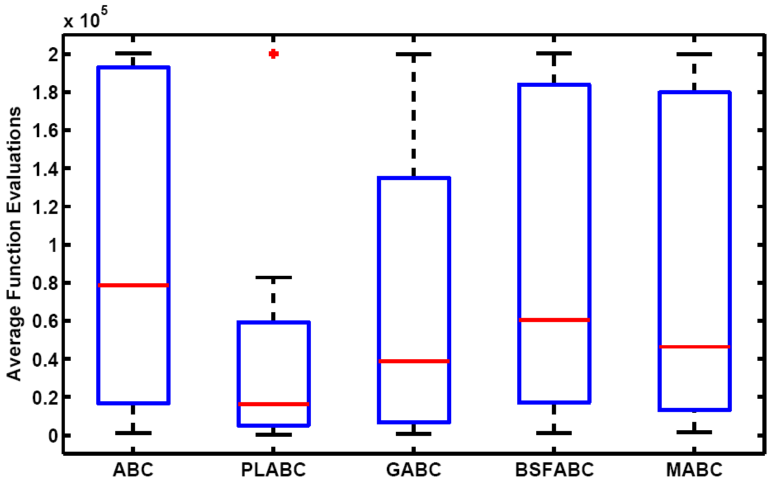
| *Function* | *PLABC vs. ABC* | *PLABC vs. GABC* | *PLABC vs. BSFABC* | *PLABC vs. MABC* |
|---|---|---|---|---|
| $f_1$ | + | – | + | + |
| $f_2$ | + | + | + | + |
| $f_3$ | + | + | + | + |
| $f_4$ | + | + | + | + |
| $f_5$ | + | + | + | + |
| $f_6$ | + | – | + | – |
| $f_7$ | + | + | + | + |
| $f_8$ | + | – | + | + |
| $f_9$ | + | + | + | + |
| $f_{10}$ | + | + | + | + |
| $f_{11}$ | + | + | + | + |
| $f_{12}$ | + | + | + | + |
| $f_{13}$ | + | – | + | + |
| $f_{14}$ | – | – | – | – |

**Table 3** Summary of Table 2 outcome (continued)

| *Function* | *PLABC vs. ABC* | *PLABC vs. GABC* | *PLABC vs. BSFABC* | *PLABC vs. MABC* |
|---|---|---|---|---|
| $f_{15}$ | – | – | + | – |
| $f_{16}$ | + | – | + | + |
| $f_{17}$ | + | – | + | + |
| $f_{18}$ | + | – | + | + |
| $f_{19}$ | + | + | – | + |
| $f_{20}$ | + | + | + | + |
| $f_{21}$ | + | + | + | + |
| $f_{22}$ | + | + | + | + |
| $f_{23}$ | + | + | + | + |
| $f_{24}$ | + | – | + | + |
| Total number of + sign | 22 | 14 | 22 | 21 |

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 et al., 1989). The boxplots for *ABC*, *PLABC*, *GABC*, *BSFABC* and *MABC* are shown in Figure 4. It is clear from this figure that *PLABC* is better than the considered algorithms as interquartile range and median are comparatively low.

**Figure 4** Boxplots graphs for average function evaluation (see online version for colours)




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 (Bansal and Sharma, 2012). The values of *PI* for the *ABC*, *PLABC*, *GABC*, *BSFABC*, and *MABC* are calculated by using following equations:

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

where

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

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

$Sr^i$ = successful simulations/runs of $i^{\text{th}}$ problem

$Tr^i$ = total simulations of $i^{\text{th}}$ problem

$Mf^i$ = minimum of average number of function evaluations used for obtaining the required solution of $i^{\text{th}}$ problem

$Af^i$ = average number of function evaluations used for obtaining the required solution of $i^{\text{th}}$ problem

$Mo^i$ = minimum of mean error obtained for the $i^{\text{th}}$ problem

$Ao^i$ = mean error obtained by an algorithm for the $i^{\text{th}}$ problem

$N_p$ = total number of optimisation 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 *PI*s, equal weights are assigned to two variables while weight of the remaining variable vary from 0 to 1 as given in Bansal and Sharma (2012). Following are the resultant cases:

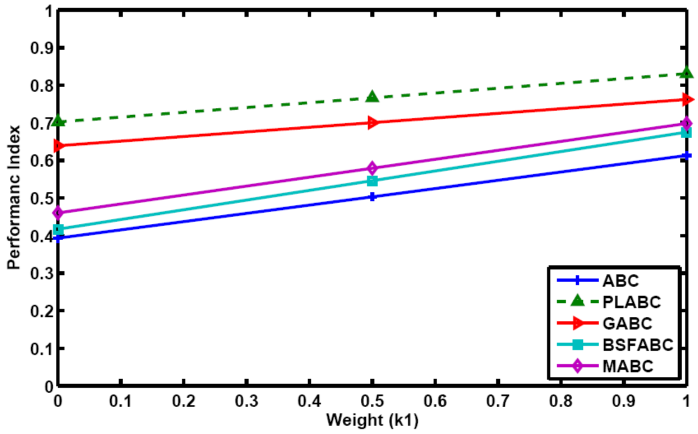
1 $k_1 = W, k_2 = k_3 = \dfrac{1-W}{2}, 0 \leq W \leq 1$

2 $k_2 = W, k_1 = k_3 = \dfrac{1-W}{2}, 0 \leq W \leq 1$

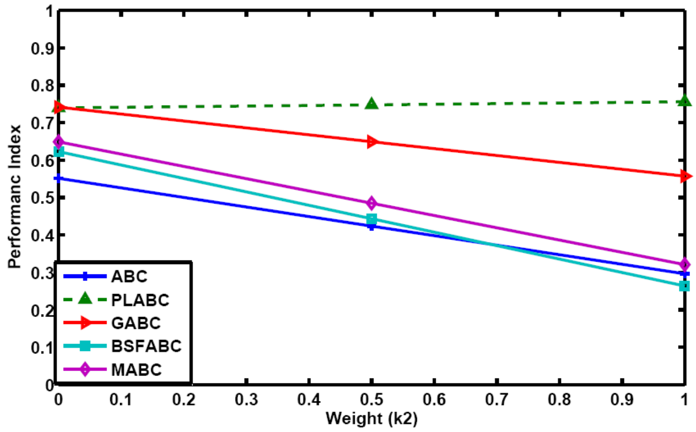
3 $k_3 = W, k_1 = k_2 = \dfrac{1-W}{2}, 0 \leq W \leq 1.$

The graphs corresponding to each of the cases 1, 2 and 3 for *ABC*, *PLABC*, *GABC*, *BSFABC*, and *MABC* are shown in Figures 5(a), 5(b), and 5(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.

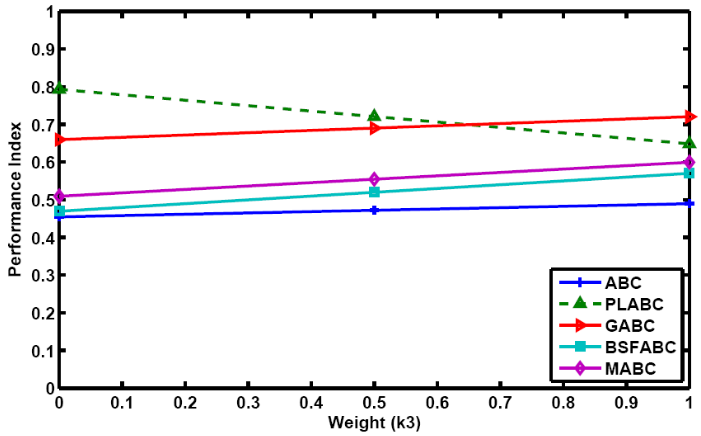
**Figure 5** Performance index for test problems; (a) for case 1, (b) for case 2 and (c) for case 3 (see online version for colours)




(a)




(b)




(c)

In case 1, average number of function evaluations and the mean error are given equal weights. *PI*s of the considered algorithms are superimposed in Figure 5(a) for comparison of the performance. It is observed that *PIs* of *PLABC* 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 number of function evaluations. It is clear from Figure 5(b) and Figure 5(c) that the algorithms perform same as in case 1.

## 7 Conclusions

In this paper, a *PLLS* strategy is proposed and incorporated with ABC. The so obtained *ABC* is named as *PLABC*. In the proposed LS, new solutions are generated in the neighbourhood of the best solution depending upon a newly introduced parameter, perturbation rate. Further, the proposed algorithm is compared to the recent variants of ABC, namely, *GABC*, *BSFABC* and *MABC* and with the help of experiments over test problems, it is shown that the *PLABC* outperforms other algorithms under consideration in terms of reliability, efficiency and accuracy.

## References

Akay, B. and Karaboga, D. (2012) 'A modified artificial bee colony algorithm for real-parameter optimization', *Information Sciences*, Vol. 192, pp.120–142, Elsevier.

Ali, M.M., Khompatraporn, C. and Zabinsky, Z.B. (2005) 'A numerical evaluation of several stochastic algorithms on selected continuous global optimization test problems', *Journal of Global Optimization*, Vol. 31, No. 4, pp.635–672.

Banharnsakun, A., Achalakul, T. and Sirinaovakul, B. (2011) 'The best-so-far selection in artificial bee colony algorithm', *Applied Soft Computing*, Vol. 11, No. 2, pp.2888–2901.

Bansal, J.C. and Sharma, H. (2012) 'Cognitive learning in differential evolution and its application to model order reduction problem for single-input single-output systems', *Memetic Computing*, September, Vol. 4, No. 3, pp.209–229.

Beyer, H.G. and Schwefel, H.P. (2002) 'Evolution strategies – a comprehensive introduction', *Natural Computing*, Vol. 1, No. 1, pp.3–52, Springer.

Brest, J., Zumer, V. and Maucec, M.S. (2006) 'Self-adaptive differential evolution algorithm in constrained real-parameter optimization', *IEEE Congress on Evolutionary Computation, 2006, CEC 2006*, pp.215–222, IEEE.

Caponio, A., Cascella, G.L., Neri, F., Salvatore, N. and 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*, Vol. 37, No. 1, pp.28–41.

Caponio, A., Neri, F. and Tirronen, V. (2009) 'Super-fit control adaptation in memetic differential evolution frameworks', *Soft Computing – A Fusion of Foundations, Methodologies and Applications*, Vol. 13, No. 8, pp.811–831.

Chen, X., Ong, Y.S., Lim, M.H. and Tan, K.C. (2011) 'A multi-facet survey on memetic computation', *IEEE Transactions on Evolutionary Computation*, Vol. 15, No. 5, pp.591–607.

Cotta, C. and Neri, F. (2012) 'Memetic algorithms in continuous optimization', *Handbook of Memetic Algorithms*, *Studies in Computational Intelligence*, Vol. 379, pp.121–134, Springer, Berlin, Germany.

Dasgupta, D. (2006) 'Advances in artificial immune systems', *Computational Intelligence Magazine*, Vol. 1, No. 4, pp.40–49, IEEE.

Diwold, K., Aderhold, A., Scheidler, A. and Middendorf, M. (2011) 'Performance evaluation of artificial bee colony optimization and new selection schemes', *Memetic Computing*, Vol. 3, No. 3, pp.149–162, Springer.

Dorigo, M. and Di Caro, G. (1999) 'Ant colony optimization: a new meta-heuristic', *Proceedings of the 1999 Congress on Evolutionary Computation, 1999, CEC'99*, Vol. 2, IEEE.

Eiben, A.E. and Smith, J.E. (2003) *Introduction to Evolutionary Computing*, Springer Verlag, Heidelberg, Germany.

El-Abd, M. (2011) 'Performance assessment of foraging algorithms vs. evolutionary algorithms', *Information Sciences*, Vol. 182, No. 1, pp.243–263.

Fister, I., Fister, I., Jr., Brest, J. and Žumer, V. (2012) 'Memetic artificial bee colony algorithm for large-scale global optimization', *Proceedings: 2012 IEEE Congress on Evolutionary Computation (CEC)*, pp.1–8, IEEE, Arxiv preprint arXiv:1206.1074.

Fogel, D.B. and Michalewicz, Z. (1997) *Handbook of Evolutionary Computation*, Taylor & Francis, New York.

Gallo, C., Carballido, J. and Ponzoni, I. (2009) 'Bihea: a hybrid evolutionary approach for microarray biclustering', *Advances in Bioinformatics and Computational Biology*, pp.36–47.

Goh, C.K., Ong, Y.S. and Tan, K.C. (2009) *Multi-objective Memetic Algorithms*, Vol. 171, Springer Verlag, Heidelberg, Germany, Vol. 171.

Goldberg, D.E. (1989) *Genetic Algorithms in Search, Optimization, and Machine Learning*.

Hooke, R. and Jeeves, T.A. (1961) ''Direct Search' solution of numerical and statistical problems', *Journal of the ACM (JACM)*, Vol. 8, No. 2, pp.212–229.

Hoos, H.H. and Stützle, T. (2005) *Stochastic Local Search: Foundations and Applications*, Morgan Kaufmann, Amsterdam.

Ishibuchi, H. and Yamamoto, T. (2004) 'Fuzzy rule selection by multi-objective genetic local search algorithms and rule evaluation measures in data mining', *Fuzzy Sets and Systems*, Vol. 141, No. 1, pp.59–88.

Ishibuchi, H., Yoshida, T. and Murata, T. (2003) 'Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling', *IEEE Transactions on Evolutionary Computation*, Vol. 7, No. 2, pp.204–223.

Kang, F., Li, J. and Ma, Z. (2011a) 'Rosenbrock artificial bee colony algorithm for accurate global optimization of numerical functions', *Information Sciences*, Vol. 181, No. 16, pp.3508–3531.

Kang, F., Li, J., Ma, Z. and Li, H. (2011b) 'Artificial bee colony algorithm with local search for numerical optimization', *Journal of Software*, Vol. 6, No. 3, pp.490–497.

Karaboga, D. (2005) *An Idea Based on Honey Bee Swarm for Numerical Optimization*, Techn. Rep. TR06, Erciyes Univ. Press, Erciyes.

Karaboga, D. and Akay, B. (2009) 'A comparative study of artificial bee colony algorithm', *Applied Mathematics and Computation*, Vol. 214, No. 1, pp.108–132.

Karaboga, D. and Basturk, B. (2007) 'Artificial bee colony (ABC) optimization algorithm for solving constrained optimization problems', *Foundations of Fuzzy Logic and Soft Computing*, pp.789–798.

Kennedy, J. (2006) 'Swarm intelligence', *Handbook of Nature-Inspired and Innovative Computing*, pp.187–219.

Kennedy, J. and Eberhart, R. (1995) 'Particle swarm optimization', *Proceedings of the IEEE International Conference on Neural Networks, 1995*, IEEE, Vol. 4, pp.1942–1948.

Knowles, J., Corne, D. and Deb, K. (2008) *Multiobjective Problem Solving from Nature: From Concepts to Applications (Natural Computing Series)*, Springer, Berlin, Germany.

Mezura-Montes, E. and Velez-Koeppel, R.E. (2010) 'Elitist artificial bee colony for constrained real-parameter optimization', *2010 Congress on Evolutionary Computation (CEC2010)*, IEEE Service Center, Barcelona, Spain, pp.2068–2075.

Mininno, E. and Neri, F. (2010) 'A memetic differential evolution approach in noisy optimization', *Memetic Computing*, Vol. 2, No. 2, pp.111–135.

Moscato, P. (1989) 'On evolution, search, optimization, genetic algorithms and martial arts: towards memetic algorithms', *Caltech Concurrent Computation Program*, C3P Report, 826:1989.

Neri, F. and Tirronen, V. (2009) 'Scale factor local search in differential evolution', *Memetic Computing*, Vol. 1, No. 2, pp.153–171, Springer.

Neri, F., Cotta, C. and Moscato, P. (Eds.) (2012) *Handbook of Memetic Algorithms*, Studies in Computational Intelligence, Vol. 379, Springer.

Nguyen, Q.H., Ong, Y.S. and Lim, M.H. (2009) 'A probabilistic memetic framework', *IEEE Transactions on Evolutionary Computation*, Vol. 13, No. 3, pp.604–623.

Ong, Y.S. and Keane, A.J. (2004) 'Meta-lamarckian learning in memetic algorithms', *IEEE Transactions on Evolutionary Computation*, Vol. 8, No. 2, pp.99–110.

Ong, Y.S., Lim, M. and Chen, X. (2010) 'Memetic computation – past, present & future [research frontier]', *Computational Intelligence Magazine*, Vol. 5, No. 2, pp.24–31, IEEE.

Ong, Y.S., Lim, M.H., Zhu, N. and Wong, K.W. (2006) 'Classification of adaptive memetic algorithms: a comparative study', *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, Vol. 36, No. 1, pp.141–152.

Ong, Y.S., Nair, P.B. and Keane, A.J. (2003) 'Evolutionary optimization of computationally expensive problems via surrogate modeling', *AIAA Journal*, Vol. 41, No. 4, pp.687–696.

Passino, K.M. (2002) 'Biomimicry of bacterial foraging for distributed optimization and control', *Control Systems Magazine*, Vol. 22, No. 3, pp.52–67, IEEE.

Price, K.V., Storn, R.M. and Lampinen, J.A. (2005) *Differential Evolution: A Practical Approach to Global Optimization*, Springer Verlag, Berlin, Germany.

Rao, S.S. and Rao, S.S. (2009) *Engineering Optimization: Theory and Practice*, John Wiley & Sons, New York.

Repoussis, P.P., Tarantilis, C.D. and Ioannou, G. (2009) 'Arc-guided evolutionary algorithm for the vehicle routing problem with time windows', *IEEE Transactions on Evolutionary Computation*, Vol. 13, No. 3, pp.624–647.

Richer, J.M., Göeffon, A. and Hao, J.K. (2009) 'A memetic algorithm for phylogenetic reconstruction with maximum parsimony', *Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics*, Vol. 5483, pp.164–175, Springer, Berlin Heidelberg.

Ruiz-Torrubiano, R. and Suárez, A. (2010) 'Hybrid approaches and dimensionality reduction for portfolio selection with cardinality constraints', *Computational Intelligence Magazine*, Vol. 5, No. 2, pp.92–107, IEEE.

Sharma, H., Verma, A. and Bansal, J. (2012) 'Group social learning in artificial bee colony optimization algorithm', *Proceedings of the International Conference on Soft Computing for Problem Solving (SocProS 2011)*, 20–22 December 2011, Springer, pp.441–451.

Storn, R. and Price, K. (1997) 'Differential evolution-a simple and efficient adaptive scheme for global optimization over continuous spaces', *Journal of Global Optimization*, Vol. 11, No. 4, pp.341–359.

Suganthan, P.N., Hansen, N., Liang, J.J., Deb, K., Chen, Y.P., Auger, A. and Tiwari, S. (2005) 'Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization', *CEC 2005*.

Susan, J. (1999) *The Meme Machine*, Oxford University Press, New York.

Tang, K., Mei, Y. and Yao, X. (2009) 'Memetic algorithm with extended neighborhood search for capacitated arc routing problems', *IEEE Transactions on Evolutionary Computation*, Vol. 13, No. 5, pp.1151–1166.

Valenzuela, J. and Smith, A.E. (2002) 'A seeded memetic algorithm for large unit commitment problems', *Journal of Heuristics*, Vol. 8, No. 2, pp.173–195.

Vesterstrom, J. and Thomsen, R. (2004) 'A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems', *Congress on Evolutionary Computation, 2004, CEC2004*, IEEE, Vol. 2, pp.1980–1987.

Wang, H., Wang, D. and Yang, S. (2009) 'A memetic algorithm with adaptive hill climbing strategy for dynamic optimization problems', *Soft Computing – A Fusion of Foundations, Methodologies and Applications*, Vol. 13, No. 8, pp.763–780.

Williamson, D.F., Parker, R.A. and Kendrick, J.S. (1989) 'The box plot: a simple visual method to interpret data', *Annals of Internal Medicine*, Vol. 110, No. 11, 916p.

Yang, X.S. (2010) *Nature-inspired Metaheuristic Algorithms*, Luniver Press, UK.

Zhu, G. and Kwong, S. (2010) 'Gbest-guided artificial bee colony algorithm for numerical function optimization', *Applied Mathematics and Computation*, Vol. 217, No. 7, pp.3166–3173.