Memetic Comp. (2015) 7:215–230
DOI 10.1007/s12293-015-0158-x

REGULAR RESEARCH PAPER

Accelerating Artificial Bee Colony algorithm with adaptive local search

Shimpi Singh Jadon · Jagdish Chand Bansal · Ritu Tiwari · Harish Sharma

Received: 25 March 2014 / Accepted: 5 February 2015 / Published online: 25 February 2015

Abstract Artificial Bee Colony (ABC) algorithm has been emerged as one of the latest Swarm Intelligence based algorithm. Though, ABC is a competitive algorithm as compared to many other optimization techniques, the drawbacks like preference on exploration at the cost of exploitation and skipping the true solution due to large step sizes, are also associated with it. In this paper, two modifications are proposed in the basic version of ABC to deal with these drawbacks: solution update strategy is modified by incorporating the role of fitness of the solutions and a local search based on greedy logarithmic decreasing step size is applied. The modified ABC is named as accelerating ABC with an adaptive local search (AABCLS). The former change is incorporated to guide to not so good solutions about the directions for position update, while the latter modification concentrates only on exploitation of the available information of the search space. To validate the performance of the proposed algorithm AABCLS, 30 benchmark optimization problems of different complexities are considered and results comparison section shows the clear superiority of the proposed modification over the Basic ABC and the other recent variants namely, Best-So-Far ABC (BSFABC), Gbest guided ABC (GABC), Opposition based levy flight ABC (OBLFABC) and Modified ABC (MABC).

S. S. Jadon · R. Tiwari
ABV-Indian Institute of Information Technology and Management, Gwalior, India
e-mail: shimpisingh2k6@gmail.com

R. Tiwari
e-mail: tiwariritu2@gmail.com

J. C. Bansal (✉)
South Asian University, New Delhi, India
e-mail: jcbansal@gmail.com

H. Sharma
Vardhaman Mahaveer Open University, Kota, India
e-mail: harish.sharma0107@gmail.com

Keywords Artificial Bee Colony · Memetic algorithm · Optimization · Exploration–exploitation · Swarm Intelligence

1 Introduction

Swarm Intelligence is one of the recent outcome of the research in the field of Nature inspired algorithms. The collaboration among social insects while searching for food and creation of the intelligent structures is known as Swarm Intelligence. Ant colony optimization (ACO) [12], particle swarm optimization (PSO) [27], bacterial foraging optimization (BFO) [39] are some examples of swarm intelligence based optimization techniques. The work presented in the articles [12,27,40,53] proved its efficiency and potential to deal with non-linear, non convex and discrete optimization problems. Karaboga [24] contributed the recent addition to this category, known as Artificial Bee Colony (ABC) optimization algorithm. The ABC algorithm mimics the foraging behavior of honey bees while they search for food. ABC is a simple and population based optimization algorithm. Here the population consists of possible solutions in terms of food sources for honey bees. A food source's fitness is measured in terms of nectar amount which the food source contains. The swarm updating in ABC is due to two processes namely, the variation process and the selection process which are responsible for exploration and exploitation, respectively. It is observed that the position update equation of ABC algorithm is good at exploration but poor at exploitation [56] i.e, ABC has not a proper balance between exploration and exploitation. Therefore these drawbacks require a modification in position update equation and/or a local search approach to be implemented in ABC. These drawbacks have also been addressed in earlier research. To enhance the exploitation, Gao and Liu [16] improved position update equation of ABC such that the bee searches only in neighborhood of the previous iteration's best solution. Banharnsakun et al. [3] proposed the best-so-far selection in ABC algorithm and incorporated three major changes: The best-so-far method, an adjustable search radius, and an objective-value-based comparison in ABC. To solve constrained optimization problems, Karaboga and Akay [26] used Deb's rules consisting of three simple heuristic rules and a probabilistic selection scheme in ABC algorithm. Karaboga [24] examined and suggested that the $limit$ should be taken as $SN \times D$, where, SN is the population size and D is the dimension of the problem and coefficient ϕ_{ij} in position update equation should be adopted in the range of $[-1, 1]$. Further, Kang et al. [22] introduced exploitation phase in ABC using Rosenbrock's rotational direction method and named modified ABC as Rosenbrock ABC (RABC).

In this paper two modifications in ABC are proposed. First, position update equation of basic ABC is modified based on the fitness such that the solutions having relatively low fitness search near the global best solution of the current swarm only and other solutions having better fitness include information of the global best solution and a random solution in their position update procedure. Second and an effective modification is the incorporation of a local search strategy in ABC based on greedy and logarithm decreasing range of step size. This local search strategy is inspired from logarithmic decreasing inertia weight strategy proposed in [17]. Our proposed modified ABC variant is named as Accelerating ABC with an adaptive local search (AABCLS).

The rest of the paper is organized as follows: Sect. 2 summarizes different local search strategies which have been incorporated in various optimization algorithms. ABC algorithm is explained in Sect. 3. Section 4 presents the modified ABC called AABCLS algorithm. The experimental results and discussions are presented in Sect. 5. Finally, the conclusion is drawn in Sect. 6.

2 Brief review on local search strategies

In the field of optimization, local search strategy is an interesting and efficient help for population based optimization methods while solving complex problems [37]. The population based optimization algorithm hybridized with local improvement strategy is termed as Memetic Algorithm (MA), initially by Moscato [32]. The traditional examples of population-based methods are Genetic Algorithms, Genetic Programming and other Evolutionary Algorithms, on the other hand Tabu Search and Simulated Annealing are two prominent local search representatives. The global search is expected to detect the global optimum while the obvious role of local search algorithm must be to converge to the closest local optimum. Local search strategies can be stochastic/deterministic, single/multi agent based, steepest descent/greedy approach based procedures [34]. Therefore, in order to maintain the proper balance between exploration and exploitation in an algorithmic process, a local search strategy is highly required to be incorporated within the basic population based algorithm.

Nowadays, researchers are continuously working to enhance the exploitation capability of the population based algorithms by hybridizing the various local search strategies within these [4,8,21,31,33,38,45,47,54]. Further, Memetic Algorithms are also applied to solve complex optimization problems like continuous optimization [36,38], flow shop scheduling [21], machine learning [7,20,44], multiobjective optimization [18,28,50], combinatorial optimization [21,42,51], bioinformatics [15,43], scheduling and routing [6], etc.

In the past, few more efforts have been made to incorporate a local search within ABC. Thammano and Phu-ang [52] proposed a hybrid ABC algorithm for solving the flexible job shop scheduling problem in which they incorporated five different local search strategies at different phases of the ABC algorithm to accomplish different purposes. To initialize the individual solution, the harmony search algorithm is considered. To search another solution in the neighborhood of a particular solution, the iterated local search technique, the scatter search technique, and the filter and fan algorithm are used. Finally, the simulated annealing algorithm is employed on a solution which is trapped in a local optimum. Sharma et al. [46] incorporated a power law-based local search strategy with DE. Fister et al. [14] developed a memetic ABC (MABC) which is hybridized with two local search heuristics: the Nelder–Mead algorithm (NMA) for exploration purpose and the random walk with direction exploitation ($RWDE$) for exploitation purpose. Here stochastic adaptive rule as specified by Neri [9] is applied for balancing the exploration and exploitation. Tarun et al. [48] developed an algorithm called improved local search in ABC using golden section search where golden section method is employed in onlooker phase. Ong and Keane [36] proposed various local search techniques during a Memetic Algorithm search in the sprit of Lamarckian learning. Further they also introduced different strategies for MAs control which choose one local search amongst the others at runtime for the exploitation of the existing solution. Nguyen et al. [35] presented a different probabilistic memetic framework that models MAs as a process involving the decision of embracing the separate actions of evolution or individual learning and analyzed 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.

A Hooke and Jeeves method [19] based local search is incorporated in basic ABC algorithm by Kang et al. [23] which is known as (HJABC) for numerical optimization. In the HJABC, the calculation of fitness function (fit_i) of basic ABC is also changed as in Eq. (1).

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

here NP is the total number of solutions, p_i is the position of the solution in the whole population after ranking, $SP \in [1.0, 2.0]$ is the selection pressure. Further, Mezura-Montes and Velez-Koeppel [30] introduced a Elitist ABC with two different local search strategies and which one out of both strategies will work at a time, is regulated through the count on function evaluations. Here, best solution in the current swarm is improved by generating a set of 1000 new food sources in its neighborhood. Kang et al. [22] proposed a Rosenbrock ABC (RABC) where exploitation phase is introduced in the ABC using Rosenbrock's rotational direction method.

3 Artificial Bee Colony (ABC) algorithm

The ABC algorithm is a population based Swarm Intelligence algorithm which is inspired by food foraging behavior of honey bees. In ABC, each solution is known as food source of honey bees whose fitness is determined in terms of the quality of the food source. Artificial Bee Colony is made of three groups of some bees: employed bees, onlooker bees and scout bees. The number of employed and onlooker bees is equal. The employed bees search the food source in the environment and store the information like the quality and the distance of the food source from the hive. Onlooker bees wait in the hive for employed bees and after collecting information from them, they start searching in neighborhood of those food sources which have better nectar. If any food source is abandoned then scout bee finds new food source randomly in the search space. While searching the solution of any optimization problem, ABC algorithm first initializes ABC parameters and swarm then it requires the repetitive iterations of the three phases, namely employed bee phase, onlooker bee phase and scout bee phase. The pseudo-code of the ABC algorithm is shown in Algorithm 1 [25]. Working procedure of each phase for ABC algorithm is explained in the Sects. 3.1–3.4.

Algorithm 1 Artificial Bee Colony Algorithm

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

3.1 Initialization of the Swarm

The first step in ABC is to initialize the solutions (food sources). If D is the number of variables in the optimization problem then each food source $x_i (i = 1, 2, \ldots, SN)$ is a D-dimensional vector among the SN food sources and is generated using a uniform distribution as:

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

here x_i represents the ith food source in the swarm, x_{minj} and x_{maxj} are bounds of x_i in jth dimension and $rand[0, 1]$ is a uniformly distributed random number in the range [0, 1]. After initialization phase ABC requires the cycle of the three phases, namely employed bee phase, onlooker bee phase and scout bee phase to be executed.

3.2 Employed bee phase

In this phase, ith food source's position is updated using following Equation:

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

here $k \in \{1, 2, \ldots, SN\}$ and $j \in \{1, 2, \ldots, D\}$ are randomly chosen indices and $k \neq i$. ϕ_{ij} is a random number in the range [−1,1]. After generating new position v_i, the position with better fitness between the newly generated v_i and the original x_i, is selected.

3.3 Onlooker bees phase

In this phase, employed bees share the information associated with its food source like quality (nectar) and position of the food source with the onlooker bees in the hive. Onlooker bees evaluate the available information about the food source and based on the food source's fitness, the onlooker bees select a food source with a probability $prob_i$. Here $prob_i$ can be calculated as function of fitness (there may be some other):

$$prob_i(G) = \frac{0.9 \times fitness_i}{maxfit} + 0.1, \quad (4)$$

here $fitness_i$ is the fitness value of the ith solution (food source) and $maxfit$ is the maximum fitness amongst all the food sources. Based on this probability, onlooker selects a food source and modifies it using the same Eq. (3) as in employed bee phase. Again by applying greedy selection, if the fitness is higher than the previous one, the onlooker bee stores the new food source in its memory and forgets the old one.

3.4 Scout bees phase

If a bee's food source is not updated for a fixed number of iterations, then that food source is considered to be abandoned and the corresponding bee becomes a scout bee. In this phase, the abandoned food source is replaced by a randomly chosen food source within the search space using the Eq. (2) as in initialization phase. In ABC, the number of iterations after which a particular food source becomes abandoned is known as $limit$ and is a crucial control parameter.

4 Accelerating Artificial Bee Colony algorithm with an adaptive local search

In this section, the proposed Accelerating ABC Algorithm with an adaptive local search (AABCLS) is introduced in detail. To be specific, in order to better guide the bees during the searching process, the existing ABC frame-work is modified. Here two modifications in ABC are proposed: first, the position update equation for solutions is modified and second, a new self adaptive local search strategy is incorporated with ABC. An algorithm which establishes a proper balance between exploration and exploitation capabilities, is considered as an efficient algorithm. In other words, an algorithm is regarded as reliable and widely applicable if it can balance exploration and exploitation during the search process. ABC algorithm achieves a good solution at a significantly faster rate, but it is weak in refining the already explored search space. Moreover, basic ABC itself has some drawbacks, like stop proceeding toward the global optimum even though the population has not converged to a local optimum [25]. These drawbacks make ABC a candidate to modify so that it can search solution more efficiently. The motivation for the proposed modifications in ABC can be described as follows:

1. **Why modification is required in position update equation?** In ABC, at any instance, a solution is updated through information flow from other solutions of the swarm. This position updating process uses a linear combination of current position of the potential solution which is going to be updated and position of a randomly selected solution as step size with a random coefficient $\phi_{ij} \in$ [-1, 1]. This process plays an important role to decide the quality of the new solution. If the current solution is far from randomly selected solution and absolute value of ϕ_{ij} is also high then the change will be large enough to jump the true optima. On the other hand, small change will decrease the convergence rate of the whole ABC process. Further, It is also suggested in literatures [25,56] that basic ABC itself has some drawbacks, like stop proceeding toward the global optimum even though the population has not converged to a local optimum and it is observed that the position update equation of ABC algorithm is good at exploration but poor at exploitation. Karaboga and Akay [25] also analyzed the various variants of ABC and found that the ABC shows poor performance and remains inefficient to balance the exploration and exploitation capabilities of the search space. Therefore, the amount of change in the solution (say, step size) should be taken care of to balance the exploration and exploitation capabilities of the ABC algorithm. But this balance can not be done manually, as it consists of random coefficient ϕ_{ij}.
2. **Why an adaptive local search is incorporated with ABC?** As mentioned earlier, if ϕ_{ij} and difference between randomly selected solution and current solution is high in position update equation of ABC then there will be sufficient chance that modified solution jump the global optima. In this situation, some local search strategy can help the search procedure. During the iterations, local search algorithm illustrates very strong exploitation capability [54]. Therefore, the exploitation capability of ABC algorithm may be enhanced by incorporating a local search strategy with ABC algorithm. In this way, the exploration and exploitation capability of ABC algorithm could be balanced as the global search capability of the ABC algorithm explores the search space or tries to identify the most promising search space regions, while the local search strategy will exploit the identified search space.

Hence, in this paper, the basic ABC is modified in two ways:

– In the position update equation of basic ABC (refer Eq. 3), individual candidate modifies its position by moving towards (or away from) a randomly selected solution. This randomly selected solution has equal chance to be good or bad, so there is no guaranty that new candidate position will be better than the last one. This scheme improves the exploration at the cost of exploitation. Hence, the solutions should not be allowed to follow blindly a randomly selected solution. Instead of this, it

may be a better strategy to compel less fit solutions to follow the best ever found solution through which the exploitation capability of ABC may be improved. Further, comparatively high fit solutions may be encouraged to use the information from best as well as a randomly selected individual to avoid premature convergence and stagnation. In this way, the higher fit solutions will be updated through a weighted sum of best and randomly selected solution and hence will not converge too quickly. In this strategy, to classify less and higher fit solutions, a probability $prob_i$ (refer Eq. 4), which is a function of fitness, is applied.
In the proposed modification, if the value of $prob_i$ is <0.5 then the ith solution is considered less fit solution than the other solutions in the swarm. So instead of searching around a randomly selected solution, this class of solutions (solutions for which $prob_i < 0.5$) moves towards the best solution found so far in the swarm. For these solutions, the random solution x_k in the position update equation of basic ABC is replaced by the best solution (x_{best}) found so far and assigned a positive weight ψ_{ij} to it in the interval $(0, C)$ as explained in Eq. (5), where C is a positive constant. For detailed description of C refer to [56]. On the other hand, relatively fit solutions (solutions for which $prob_i \geq 0.5$) are not allowed to blindly follow the direction of best solution in the swarm as it may be a local optimum and solutions may prematurely converge to it. Therefore, the position update equation of these solutions also includes the learning component of the basic ABC. The proposed position update equation is explained as below:

$$v_{ij} = \begin{cases} x_{ij} + \psi_{ij}(x_{bestj} - x_{ij}), & \text{if } prob_i < 0.5. \\ x_{ij} + \phi_{ij}(x_{ij} - x_{kj}) & \\ \quad +\psi_{ij}(x_{bestj} - x_{ij}), & \text{otherwise.} \end{cases} \quad (5)$$

– In the second modification, a self adaptive local search strategy is proposed and incorporated with the ABC. The proposed local search strategy is inspired from logarithmic decreasing inertia weight scheme [17]. In the proposed local search strategy, the required step size (i.e.,$\phi(x_{current} - x_{random})$) to update an individual is reduced self adaptively to exploit the search area in the proximity of the best solution. Thus, the proposed strategy is named as self adaptive local search (SALS). In SALS, the step size is reduced as a logarithmic function of iteration counter as shown in Eqs. (7) and (8). In SALS, the random component (ϕ_{ij}) of basic ABC algorithm is optimized to direct the best solution to update its position. This process can be seen as an optimization problem solver which minimizes the unimodal continuous objective function $f(\phi_{ij})$ in the direction provided by the best solution x_{best} over the variables w_1, w_2 in the interval $[-1, 1]$ or simply it optimizes the following mathematical optimization problem:

$$\min f(\phi) \text{ in } [-1, 1]; \quad (6)$$

SALS process starts with initial range ($w_1 = -1, w_2 = 1$) and generates two points in this interval by diminishing the edges of the range through a greedy way using the Eqs. (7) and (8). At a time, either of these Equations is executed depends which $f(w_1)$ or $f(w_2)$ has better fitness. If w_1 provides better fitness then edge w_2 of the range shrinks towards w_1 otherwise w_1 shifts itself near to w_2. The detailed implementation of SALS can be seen in Algorithm 2.

$$w_1 = w_1 + (w_2 - w_1) \times log(1 + t/maxiter), \quad (7)$$

$$w_2 = w_2 - (w_2 - w_1) \times log(1 + t/maxiter), \quad (8)$$

where t and $maxiter$ are the current iteration counter and maximum allowable iterations in the local search algorithm respectively.

Algorithm 2 Self Adaptive Local Search (SALS) Strategy:

```
Input: optimization function $Minf(x)$, the best solution $x_{best}$;
Initialize termination parameters $\epsilon$, maximum number of iteration
counter $maxiter$ and variables $w_1 = -1, w_2 = 1$ and $itercount = 1$;
while ($|w_1 - w_2| > \epsilon$ and $itercount \leq maxiter$) do
  Generate two new solutions $x_{new1}$ and $x_{new2}$ from $x_{best}$ by using
  $w_1$ and $w_2$, respectively using Algorithm 3;
  Calculate $f(x_{new1})$ and $f(x_{new2})$;
  if $f(x_{new1}) < f(x_{new2})$ then
    $w_2 = w_2 - (w_2 - w_1) \times log(1 + itercount/maxiter)$;
    if $f(x_{new1}) < f(x_{best})$ then
      $x_{best} = x_{new1}$;
    end if
  else
    $w_1 = w_1 + (w_2 - w_1) \times log(1 + itercount/maxiter)$;
    if $f(x_{new2}) < f(x_{best})$ then
      $x_{best} = x_{new2}$;
    end if
  end if
  Set $itercount = itercount + 1$;
end while
```

Algorithm 2 terminates when either iteration counter exceeds the maximum iterations allowable in local search or absolute difference between w_1 and w_2 falls below a user defined parameter ϵ. In Algorithm 3, D is the dimension of the problem, $U(0, 1)$ is a uniformly distributed random number in the range $(0, 1)$, p_r is a perturbation rate and is used to control the amount of disturbance in the best solution x_{best} and x_k is a randomly selected solution from the population. For details, see the parameter settings in Sect. 5.1.

Algorithm 3 New solution generation:

```
Input: $w$ and best solution $x_{best}$ ;
for $j = 1$ to $D$ do
  if $U(0, 1) < p_r$ then
    $x_{newj} = x_{bestj} + w(x_{bestj} - x_{kj})$;
  else
    $x_{newj} = x_{bestj}$;
  end if
end for
Return $x_{new}$
```

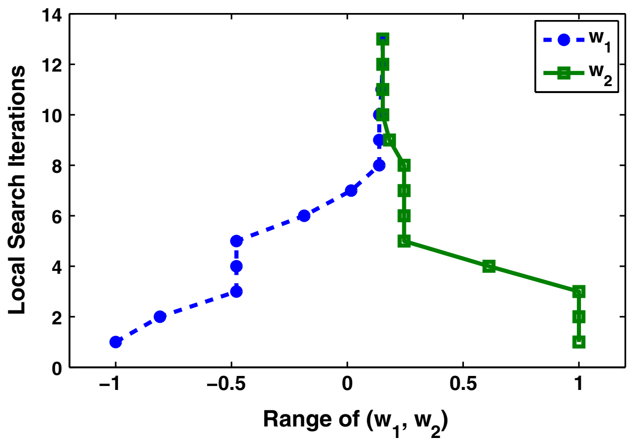


Fig. 1 Changes in the range $[w_1, w_2]$ during SALS in two dimension search space for f_{26}, refer Table 1

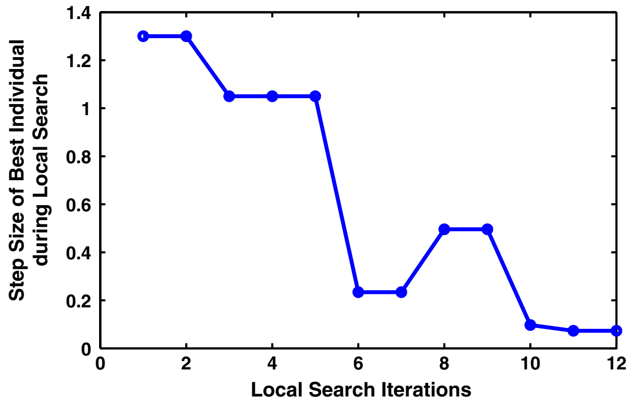


Fig. 2 Variation in step size of best individual during SALS in two dimension search space for f_{26}, refer Table 1

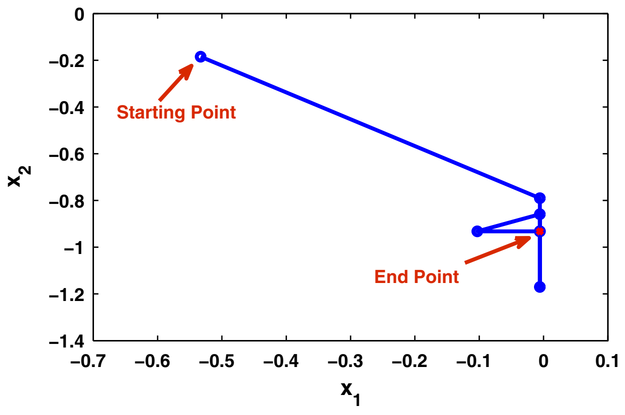


Fig. 3 Best solution movement during SALS in two dimension search space for f_{26}, refer Table 1

As the modifications are proposed to improve the convergence speed of ABC, while maintaining the diversity in the swarm. Therefore, the proposed strategy is named as "Accelerating ABC with an adaptive Local Search" (AABCLS).

It is clear from Algorithm 2 that only the best individual of the current swarm updates itself in its neighborhood. Further analyses of the proposed local search may be done through Figs. 1, 2 and 3. Figure 1 presents the variation of the range $[w_1, w_2]$ during the local search process. It is clear from this figure that the given range is iteratively reducing based on the movement of the best individual. Figure 2 shows the variations in the step size of the best individual during the local search. Figure 3 shows the movement of best individual during the local search in two dimension search space for *Goldstein–Price* function (f_{26}), (refer Table 1).

The AABCLS is composed of four phases: employed bee phase, onlooker bee phase, scout bee phase and self adaptive local search phase. The scout bee phase is same as it was in basic ABC. The employed bee phase and onlooker bee phase are also same as in basic ABC except the position update equation. The position update equations of these phases have been replaced with the proposed position update Eq. (5). The last phase, namely self adaptive local search phase is executed after the completion of scout bee phase. The pseudo-code of the proposed AABCLS algorithm is shown in Algorithm 4.

Algorithm 4 Accelerating ABC using adaptive local search (AABCLS):

```
Initialize the population and control parameters;
while Termination criteria is not satisfied do
  Step 1: Employed bee phase to generate new food sources for each bee using proposed position update equation (5).
  Step 2: Onlooker bees phase to update the food sources using Equation (5) based on their profitability.
  Step 3: Scout bee phase to determine the new food sources for exhausted food sources.
  Step 4: Apply self adaptive local search strategy phase to exploit the best solution found so far using Algorithm 2
end while
Return the best solution.
```

5 Experimental results and discussion

In this section, 30 benchmark functions which are used to investigate the algorithms' searching performance are described. The set of all the benchmark functions used, includes uni-model, multi-model, separable and non separable problems. Next, the detail of the simulation settings for all involved ABC's recent popular variants Best-So-Far ABC

(BSFABC) [3], Gbest-guided ABC (GABC) [56], Opposition Based Lévy Flight ABC (OBLFABC) [45] and Modified ABC (MABC) [1] are provided. Finally, the experimental results after analyzing and comparing the performance of the proposed algorithm AABCLS with ABC algorithm and its recent variants through distinct statistical tests are presented.

Table 1 includes 30 mathematical optimization problems (f_1 to f_{30}) of different characteristics and complexities. These all problems are continuous in nature. Test problems $f_1 - f_{21}$ and $f_{26} - f_{30}$ have been taken from [2] and test problems $f_{22} - f_{25}$ have been taken from [49] with the associated offset values. Authors decided to use this set of problems because it is versatile enough in nature which includes almost all type of problems like unimodel—multimodel, separable—nonseparable, scalable—non scalable and biased—unbiased optimization problems and if any algorithm can solve these problems of different characteristics then that algorithm may be considered as an efficient algorithm.

5.1 Experimental setting

The results obtained from the proposed AABCLS are stored in the form of success rate (SR), average number of function evaluations (AFE), standard deviation (SD) of the fitness and mean error (ME). Here *SR* represents the number of times, algorithm achieved the function optima with acceptable error in 100 runs i.e, if an algorithm is applied 100 times to solve a problem then SR of the algorithm is the number of times it finds the optimum solution or a solution with acceptable error defined in Table 1 for that problem and *AFE* is the average number of function evaluations called in 100 runs by the algorithm to reach at the termination criteria. Results for these test problems (Table 1) are also obtained from the basic ABC and its recent variants of ABC namely, BSFABC, GABC, OBLFABC and MABC for the comparison purpose. The following parameter setting is adopted while implementing our proposed and other considered algorithms to solve the problems:

- The number of simulations/run =100,
- Colony size $NP = 50$ [11,13] and Number of food sources $SN = NP/2$,
- All random numbers are generated using uniform probability distribution,
- $\phi_{ij} = rand[-1, 1]$ and limit=Dimension×Number of food sources=$D \times SN$ [1,26],
- $C = 1.5$ [56],
- The terminating criteria: Either acceptable error (Table 1) meets or maximum number of function evaluations (which is set to be 200000) is reached,
- The proposed local search in ABC runs either 10 times (based on empirical experiment) for each iteration or $|w_1 - w_2| \le \epsilon$ (here ϵ is set to 0.001) whichever comes earlier in algorithm.
- The parameter p_r in Algorithm 3 is set to 0.4 based on its sensitive analysis in range [0.1, 1] as explained in the Fig. 4. Figure 4 shows a graph between p_r and the sum of the successful runs for all the considered problems. It is clear that $p_r = 0.4$ provides the highest success rate.
- Parameter settings for the other considered algorithms ABC, GABC, BSFABC, OBLFABC and MABC are adopted from their original articles.

5.2 Results analysis of experiments

Table 2 presents the numerical results for benchmark problems of Table 1 with the experimental settings shown in Sect. 5.1. Table 2 shows the results of the proposed and other considered algorithms in terms of ME, SD, AFE and SR. Here *SR* represents the number of times, algorithm achieved the function optima with acceptable error in 100 runs and *AFE* is the average number of function evaluations called in 100 runs by the algorithm to reach at the termination criteria. Mathematically *AFE* is defined as:

$$AFE = \frac{\sum_{i=1}^{100} \text{Number of function evaluations to meet the termination criteria for run } i}{100}$$

It can be observed from Table 2 that AABCLS outperforms the considered algorithms most of the times in terms of accuracy, reliability and efficiency. Some other statistical tests like box-plots, the Mann–Whitney U rank sum test, acceleration rate (AR) [41], and performance indices [5] have also been done in order to analyze the algorithms output more intensively.

5.3 Statistical analysis

Algorithms ABC, GABC, BSFABC, OBLFABC, MABC and AABCLS are compared based on SR, AFE, and ME. First SR of all these algorithms is compared and if it is not possible to distinguish the performance of algorithms based on SR then comparison is made on the basis of AFE. ME is used for comparison if the comparison is not possible on the basis of SR and AFE both. From the results shown in Table 2, it is clear that AABCLS costs less on 27 test functions ($f_1 - f_3, f_5 - f_{11}, f_{13} - f_{28}, f_{30}$) among all the considered algorithms. As these functions include unimodel, multimodel, separable, non separable, lower and

Table 1 Test problems

Fun.	Test problem	Objective function	Search range	Optimum value	D	C	Acceptable error
f_1	Sphere	$f_1(x)=\sum_{i=1}^{D} x_i^2$	$[-5.12$ to $5.12]$	$f(\mathbf{0})=0$	30	U, S	$1.0E{-}05$
f_2	De Jong f4	$f_2(x)=\sum_{i=1}^{D} i.(x_i)^4$	$[-5.12$ to $5.12]$	$f(\mathbf{0})=0$	30	U, S	1.0E−05
f_3	Griewank	$f_3(x)=1+\frac{1}{4000}\sum_{i=1}^{D} x_i^2-\prod_{i=1}^{D}\cos(\frac{x_i}{\sqrt{i}})$	$[-600$ to $600]$	$f(\mathbf{0})=0$	30	M, NS	$1.0E{-}05$
f_4	Rosenbrock	$f_4(x)=\sum_{i=1}^{D-1}(100(x_{i+1}-{x_i}^2)^2+(x_i-1)^2)$	$[-30$ to $30]$	$f(\mathbf{1})=0$	30	U, NS	$1.0E{-}02$
f_5	Ackley	$f_5(x)=-20+e+exp(-\frac{0.2}{D}\sqrt{\sum_{i=1}^{D}{x_i}^3})$ $-exp(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)x_i)$	$[-1$ to $1]$	$f(\mathbf{0})=0$	30	M, NS	$1.0E{-}05$
f_6	Alpine	$f_6(x)=\sum_{i=1}^{D}\lvert x_i\sin x_i+0.1x_i\rvert$	$[-10$ to $10]$	$f(\mathbf{0})=0$	30	M, S	$1.0E{-}05$
f_7	Michalewicz	$f_7(x)=-\sum_{i=1}^{D}\sin x_i(\sin(\frac{i.{x_i}^2}{\pi})^{20})$	$[0,\ \pi]$	f_{min} =-9.66015	10	M, S	$1.0E{-}05$
f_8	Salomon problem	$f_8(x)=1-cos(2\pi\sqrt{\sum_{i=1}^{D}x_i^2})+0.1(\sqrt{\sum_{i=1}^{D}x_i^2})$	$[-100$ to $100]$	$f(\mathbf{0})=0$	30	M, S	$1.0E{-}01$
f_9	Axis parallel hyper-ellipsoid	$f_9(x)=\sum_{i=1}^{D} i x_i^2$	$[-5.12$ to $5.12]$	$f(\mathbf{0})=0$	30	U, S	$1.0E{-}05$
f_{10}	Sum of different powers	$f_{10}(x)=\sum_{i=1}^{D}\lvert x_i\rvert^{i+1}$	$[-1$ to $1]$	$f(\mathbf{0})=0$	30	U, S	$1.0E{-}05$
f_{11}	Step function	$f_{11}(x)=\sum_{i=1}^{D}(\lfloor x_i+0.5\rfloor)^2$	$[-100$ to $100]$	f(-0.5 $\le x\le 0.5)=0$	30	U, S	$1.0E{-}05$
f_{12}	Inverted cosine wave	$f_{12}(x)=-\sum_{i=1}^{D-1}\left(\exp\left(\frac{-(x_i^2+x_{i+1}^2+0.5x_ix_{i+1})}{8}\right)\times \mathrm{I}\right)$ where, $\mathrm{I}\ =cos\left(4\sqrt{x_i^2+x_{i+1}^2+0.5x_ix_{i+1}}\right)$	$[-5$ to $5]$	$f(\mathbf{0})=-D+1$	10	M, NS	$1.0E{-}05$
f_{13}	Neumaier 3 problem (NF3)	$f_{13}(x)=\sum_{i=1}^{D}(x_i-1)^2-\sum_{i=2}^{D}x_ix_{i-1}$	$[-D^2\ D^2]$	$f_{min}=-\frac{(D(D+4)(D-1))}{6}$	10	U, NS	$1.0E{-}01$
f_{14}	Rotated hyper-ellipsoid	$f_{14}(x)=\sum_{i=1}^{D}\sum_{j=1}^{i}x_j^2$	$[-65.536$ to $65.536]$	$f(\mathbf{0})=0$	30	U, S	$1.0E{-}05$
f_{15}	Levy montalvo 1	$f_{15}(x)=\frac{\pi}{D}(10\sin^2(\pi y_1)+\sum_{i=1}^{D-1}(y_i-1)^2\times(1+10\sin^2(\pi y_{i+1}))+(y_D-1)^2)$, where $y_i=1+\frac{1}{4}(x_i+1)$	$[-10$ to $10]$	$f(\mathbf{-1})=0$	30	M, NS	$1.0E{-}05$
f_{16}	Levy montalvo 2	$f_{16}(x)=0.1(\sin^2(3\pi x_1)+\sum_{i=1}^{D-1}(x_i-1)^2\times(1+\sin^2(3\pi x_{i+1}))+(x_D-1)^2(1+\sin^2(2\pi x_D))$	$[-5$ to $5]$	$f(\mathbf{1})=0$	30	M, NS	$1.0E{-}05$
f_{17}	Ellipsoidal	$f_{17}(x)=\sum_{i=1}^{D}(x_i-i)^2$	[−D to D]	$f(1,2,3,\ldots,D)=0$	30	U, S	$1.0E{-}05$
f_{18}	Beale	$f_{18}(x)=[1.5-x_1(1-x_2)]^2+[2.25-x_1(1-x_2^2)]^2+[2.625-x_1(1-x_2^3)]^2$	$[-4.5$ to $4.5]$	$f(3,0.5)=0$	2	U, NS	$1.0E{-}05$
f_{19}	Colville	$f_{19}(x)=100[x_2-x_1^2]^2+(1-x_1)^2+90(x_4-x_3^2)^2+(1-x_3)^2+10.1[(x_2-1)^2+(x_4-1)^2]+19.8(x_2-1)(x_4-1)$	$[-10$ to $10]$	$f(\mathbf{1})=0$	4	M, NS	$1.0E{-}05$
f_{20}	Branins's function	$f_{20}(x)=a(x_2-bx_1^2+cx_1-d)^2+e(1-f)\cos x_1+e$	$-5\le x_1\le 10, 0\le x_2\le 15$	$f(-\pi,12.275)=0.3979$	2	U, NS	$1.0E{-}05$

Table 1 continued

Fun.	Test problem	Objective function	Search range	Optimum value	D	C	Acceptable error
f_{21}	Kowalik	$f_{21}(x)=\sum_{i=1}^{11}[a_i-\frac{x_1(b_i^2+b_ix_2)}{b_i^2+b_ix_3+x_4}]^2$	[−5 to 5]	$f(0.192833, 0.190836, 0.123117, 0.135766) = 0.000307486$	4	M, NS	1.0E−05
f_{22}	Shifted Rosenbrock	$f_{22}(x)=\sum_{i=1}^{D-1}(100(z_i^2-z_{i+1})^2+(z_i-1)^2)+f_{bias}, z=x-o+1, x=[x_1,x_2,\dots x_D], o=[o_1,o_2,\dots o_D]$	[−100 to 100]	$f(o)=f_{bias}=390$	10	U, NS	1.0E−01
f_{23}	Shifted sphere	$f_{23}(x)=\sum_{i=1}^{D}z_i^2+f_{bias}, z=x-o, x=[x_1,x_2,\dots x_D], o=[o_1,o_2,\dots o_D]$	[−100 to 100]	$f(o)=f_{bias}=-450$	10	M, S	1.0E−05
f_{24}	Shifted Griewank	$f_{24}(x)=\sum_{i=1}^{D}\frac{z_i^2}{4000}-\prod_{i=1}^{D}\cos(\frac{z_i}{\sqrt{i}})+1+f_{bias}, z=(x-o), x=[x_1,x_2,\dots x_D], o=[o_1,o_2,\dots o_D]$	[−600 to 600]	$f(o)=f_{bias}=-180$	10	M, NS	1.0E−05
f_{25}	Shifted Ackley	$f_{25}(x)=-20\exp(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D}z_i^2})-\exp(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi z_i))+20+e+f_{bias}, z=(x-o), x=(x_1,x_2,\dots\dots\dots x_D), o=(o_1,o_2,\dots\dots\dots o_D)$	[−32 to 32]	$f(o)=f_{bias}=-140$	10	M, S	1.0E−05
f_{26}	Goldstein–Price	$f_{26}(x)=(1+(x_1+x_2+1)^2\times(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2))\times(30+(2x_1-3x_2)^2\times(18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2))$	[−2 to 2]	$f(0,-1)=3$	2	U, NS	1.0E−14
f_{27}	Six-hump camel back	$f_{27}(x)=(4-2.1x_1^2+x_1^4/3)x_1^2+x_1x_2+(-4+4x_2^2)x_2^2$	[−5 to 5]	$f(-0.0898, 0.7126)=-1.0316$	2	M, NS	1.0E−05
f_{28}	Meyer and Roth	$f_{28}(x)=\sum_{i=1}^{5}(\frac{x_1x_3t_i}{1+x_1t_i+x_2v_i}-y_i)^2$	[−10 to 10]	$f(3.13, 15.16, 0.78)=0.4\times10^{-4}$	3	U, NS	1.0E−03
f_{29}	Sinusoidal	$f_{29}(x)=-[A\prod_{i=1}^{D}\sin(x_i-z)+\prod_{i=1}^{D}sin(B(x_i-z))], A=2.5, B=5, z=30$	[0–180]	$f(\mathbf{90+z})=-(A+1)$	10	M, NS	1.00E−02
f_{30}	Moved axis parallel hyper-ellipsoid	$f_{30}(x)=\sum_{i=1}^{D}5i\times x_i^2$	[−5.12 to 5.12]	$f(x)=0; x(i)=5\times i, i=1:D$	30	M, S	1.0E−15

D dimensions, *C* characteristic, *U* unimodal, *M* multimodal, *S* separable, *N* non-separable, *AE* acceptable error (cont.)

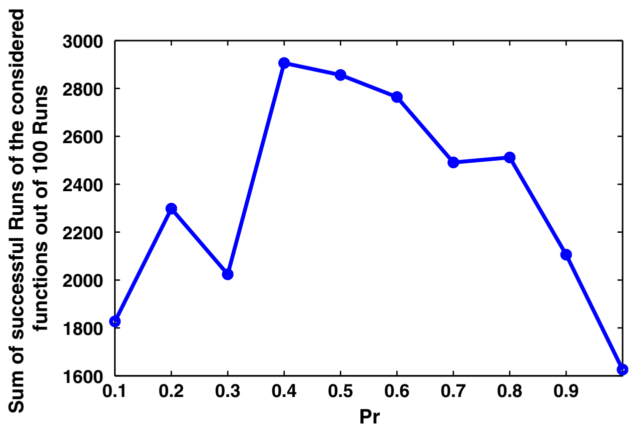


Fig. 4 Effect of parameter p_r on success rate

higher dimensional functions, it can be stated that AABCLS balances the exploration and exploitation capabilities efficiently. ABC, GABC, BSFABC and OBLFABC outperforms AABCLS over test function f_4, in which the global minimum is inside a long, narrow, parabolic shaped flat valley. The cost of OBLFABC is lower for four test functions (f_4, f_{12} and f_{29}) than the AABCLS, ABC, BSFABC and MABC which are multimodel functions. When AABCLS is compared with each of the considered algorithms individually, it is better than ABC, BSFABC and GABC over 29 test functions and it is better than MABC over all test functions of mixed characteristics. It means that when the results of all functions are evaluated together, the AABCLS algorithm is the cost effective algorithm for most of the functions. Now if AABCLS algorithm is compared based on mean error only then one can see from Table 2 that AABCLS is achieving less error on 21 out of 30 functions than all the other considered algorithms. ABC and BSFABC are good on 9 functions, OBLFABC is good on 7 functions and GABC is good on 5 functions while MABC is better on single function f_{15} than the proposed AABCLS algorithm with respect to mean error.

Since boxplot [55] can efficiently represent the empirical distribution of results, the boxplots for AFE and ME for all algorithms AABCLS, ABC, GABC, BSFABC, OBLFABC and MABC have been represented in Fig. 5. Figure 5a shows that AABCLS is cost effective in terms of function evaluations as interquartile range and median of AFE are very low for AABCLS. Boxplot Fig. 5b shows that AABCLS and GABC are competitive in terms of ME as interquartile range for both the algorithms are very low and less than the other considered algorithms. Though, it is clear from box plots that AABCLS is cost effective than ABC, BSFABC, GABC, OBLFABC and MABC i.e., AABCLS's result differs from the other, now to check, whether there exists any significant difference between algorithm's output or this difference is due to some randomness, another statistical test is required. It can be observed from boxplots of Fig. 5 that

Table 2 Comparison of the results of test problems (cont.)

TP	Algorithm	ME	SD	*AFE*	*SR*
f_1	AABCLS	7.97E−06	9.85E−07	11,403	100
	ABC	8.17E−06	2.02E−06	20,534	100
	BSFABC	7.49E−06	2.15E−06	30,159	100
	GABC	8.27E−06	1.81E−06	24,348	100
	OBLFABC	8.51E−06	1.69E−06	17,487	100
	MABC	8.95E−06	9.48E−07	22,954	100
f_2	AABCLS	5.51E−06	1.36E−06	5875	100
	ABC	4.90E−06	3.11E−06	10,175	100
	BSFABC	5.31E−06	3.12E−06	23,626	100
	GABC	5.51E−06	2.72E−06	8825	100
	OBLFABC	7.44E−06	2.66E−06	11,113	100
	MABC	8.63E−06	1.27E−06	21,835	100
f_3	AABCLS	5.55E−06	1.03E−06	32,183	100
	ABC	2.28E−04	1.26E−03	46,189	97
	BSFABC	5.96E−06	2.79E−06	43,426	100
	GABC	6.33E−06	2.99E−06	33,817	99
	OBLFABC	2.05E−04	1.42E−03	43,085	98
	MABC	9.24E−06	5.42E−07	43,003	100
f_4	AABCLS	4.57E+00	3.43E+01	187,214	12
	ABC	1.60E+00	3.08E+00	183,018	22
	BSFABC	1.97E+00	3.31E+00	179,903	23
	GABC	4.66E+00	1.15E+01	176,318	24
	OBLFABC	1.17E+00	9.98E+00	121,289	76
	MABC	3.60E+01	2.41E+01	190,001	10
f_5	AABCLS	8.54E−06	5.05E−07	31,915	100
	ABC	8.28E−06	1.56E−06	49,216	100
	BSFABC	8.11E−06	1.67E−06	42,936	100
	GABC	8.56E−06	1.37E−06	47,580	100
	OBLFABC	8.99E−06	1.06E−06	37,696	100
	MABC	9.51E−06	4.31E−07	43,009	100
f_6	AABCLS	7.05E−06	1.49E−06	44,755	100
	ABC	8.21E−06	2.27E−06	97,527	100
	BSFABC	7.43E−06	3.82E−06	116,412	98
	GABC	8.23E−06	2.09E−06	58,458	99
	OBLFABC	1.17E−05	3.23E−05	80,614	96
	MABC	1.01E−03	7.61E−04	178,806	14
f_7	AABCLS	4.98E−06	3.79E−06	16,077	100
	ABC	3.66E−05	3.74E−06	127,214	70
	BSFABC	3.63E−06	3.76E−06	46,437	90
	GABC	4.53E−06	3.56E−06	32,200	92
	OBLFABC	5.18E−06	3.54E−06	21,120	100
	MABC	6.70E−06	3.28E−06	46,368	89
f_8	AABCLS	9.22E−01	3.04E−02	23,230	100
	ABC	9.77E−01	6.61E−01	187,957	56
	BSFABC	9.60E−01	7.68E−01	159,019	62
	GABC	9.40E−01	3.39E−02	122,315	96
	OBLFABC	9.36E−01	3.43E−02	82,241	93
	MABC	9.33E−01	3.82E−02	28,327	99

Table 2 continued

TP	Algorithm	ME	SD	*AFE*	*SR*
f_9	AABCLS	6.17E−06	6.53E−07	13,270	100
	ABC	7.90E−06	1.99E−06	42,765	91
	BSFABC	6.90E−06	2.38E−06	36,045	94
	GABC	8.22E−06	1.58E−06	15,918	100
	OBLFABC	8.75E−06	1.67E−06	18,924	100
	MABC	9.22E−06	6.91E−07	25,772	96
f_{10}	AABCLS	6.25E−06	1.68E−06	3504	100
	ABC	5.49E−06	2.90E−06	16,818	100
	BSFABC	6.08E−06	2.64E−06	14,996	100
	GABC	5.97E−06	2.85E−06	14,786	100
	OBLFABC	6.05E−06	3.08E−06	7649	100
	MABC	7.57E−06	1.90E−06	9589	100
f_{11}	AABCLS	0.00E+00	0.00E+00	8186	100
	ABC	4.80E−06	1.89E−06	41,501	94
	BSFABC	1.11E−06	1.77E−06	37,072	99
	GABC	1.21E−06	1.68E−06	19,034	98
	OBLFABC	0.00E+00	0.00E+00	11,360	100
	MABC	1.22E−06	1.08E−06	15,844	98
f_{12}	AABCLS	7.26E−06	1.88E−06	34,012	100
	ABC	2.39E−02	1.34E−01	87,099	88
	BSFABC	6.20E−02	1.88E−01	98,972	89
	GABC	7.29E−06	2.38E−05	65,061	99
	OBLFABC	8.24E−06	1.91E−06	23,760	100
	MABC	8.08E−06	1.64E−06	66,422	100
f_{13}	AABCLS	9.06E−02	1.16E−02	13,081	100
	ABC	9.15E+00	6.63E+00	196,658	5
	BSFABC	4.42E+00	5.86E+00	193,009	7
	GABC	1.40E+00	1.85E+00	185,140	23
	OBLFABC	8.94E−02	1.31E−02	19,108	100
	MABC	1.01E+00	2.08E+00	145,806	32
f_{14}	AABCLS	5.97E−06	3.06E−07	14,995	100
	ABC	7.75E−06	2.20E−06	47,958	93
	BSFABC	6.90E−06	2.51E−06	48,981	89
	GABC	8.24E−06	1.68E−06	19,599	93
	OBLFABC	8.74E−06	1.68E−06	22,469	100
	MABC	9.23E−06	6.90E−07	33,065	91
f_{15}	AABCLS	9.11E−06	7.40E−07	10,707	100
	ABC	6.98E−06	2.24E−06	26,605	100
	BSFABC	6.99E−06	2.23E−06	19,608	100
	GABC	8.08E−06	1.85E−06	13,118	100
	OBLFABC	8.44E−06	1.85E−06	15,171	100
	MABC	9.06E−06	7.90E−07	22,738	100
f_{16}	AABCLS	9.11E−06	7.56E−07	11,889	100
	ABC	7.15E−06	2.46E−06	28,936	100
	BSFABC	6.96E−06	2.31E−06	21,837	100
	GABC	7.63E−06	2.13E−06	24,401	100
	OBLFABC	8.24E−06	1.92E−06	17,116	100
	MABC	9.16E−06	7.37E−07	21,028	100

Table 2 continued

TP	Algorithm	ME	SD	*AFE*	*SR*
f_{17}	AABCLS	7.16E−06	8.75E−07	14,345	100
	ABC	7.92E−06	2.29E−06	44,094	98
	BSFABC	7.30E−06	2.29E−06	41,464	100
	GABC	7.93E−06	2.14E−06	16,637	100
	OBLFABC	8.86E−06	1.62E−06	19,368	100
	MABC	9.18E−06	6.17E−07	26,831	100
f_{18}	AABCLS	4.74E−06	3.16E−06	1665	100
	ABC	8.75E−06	1.37E−06	28,268	94
	BSFABC	1.63E−05	3.72E−05	28,477	100
	GABC	5.39E−06	2.83E−06	14,114	100
	OBLFABC	7.58E−06	2.95E−06	5888	100
	MABC	5.24E−06	2.88E−06	10,176	100
f_{19}	AABCLS	8.55E−03	1.85E−03	16,411	100
	ABC	1.48E−01	9.72E−01	200,000	0
	BSFABC	2.35E−02	2.53E−02	156,114	32
	GABC	1.66E−02	1.41E−02	160,936	31
	OBLFABC	1.32E−02	9.35E−03	106,158	76
	MABC	1.36E−02	1.06E−02	142,163	36
f_{20}	AABCLS	5.27E−06	6.31E−06	693	100
	ABC	5.17E−06	6.84E−06	2049	100
	BSFABC	5.77E−06	6.79E−06	19,731	92
	GABC	5.90E−06	6.77E−06	21,104	89
	OBLFABC	6.16E−06	6.94E−06	1021	100
	MABC	6.58E−06	7.49E−06	18,653	93
f_{21}	AABCLS	8.28E−05	1.63E−05	37,599	100
	ABC	1.71E−04	7.68E−05	180,050	20
	BSFABC	1.30E−04	6.04E−05	147,033	52
	GABC	8.39E−05	1.95E−05	90,996	87
	OBLFABC	9.82E−05	2.30E−05	63,607	99
	MABC	1.98E−04	7.60E−05	176,081	24
f_{22}	AABCLS	9.18E−02	2.20E−02	97,472	98
	ABC	1.22E+00	4.98E+00	178,336	19
	BSFABC	2.07E+00	3.33E+00	171,110	21
	GABC	4.63E−01	1.54E+00	104,323	58
	OBLFABC	9.84E−02	7.26E−02	72,577	86
	MABC	9.60E−01	1.50E+00	160,528	32
f_{23}	AABCLS	5.95E−06	1.48E−06	4621	100
	ABC	7.21E−06	2.34E−06	9232	100
	BSFABC	6.65E−06	2.63E−06	14,233	100
	GABC	7.52E−06	2.05E−06	8488	100
	OBLFABC	7.60E−06	2.28E−06	6648	100
	MABC	8.19E−06	1.53E−06	8661	100
f_{24}	AABCLS	5.47E−06	3.04E−06	20,317	100
	ABC	1.27E−04	3.06E−03	88,275	83
	BSFABC	4.54E−03	5.84E−03	99,854	80
	GABC	5.61E−06	2.86E−06	36,712	100
	OBLFABC	2.22E−03	4.29E−03	90,433	77
	MABC	5.51E−04	1.99E−03	81,135	92

Table 2 continued

TP	Algorithm	ME	SD	AFE	SR
f_{25}	AABCLS	8.67E−06	6.33E−07	9966	100
	ABC	7.66E−06	1.91E−06	17,627	100
	BSFABC	7.88E−06	1.68E−06	31,261	100
	GABC	8.14E−06	1.37E−06	15,326	100
	OBLFABC	8.58E−06	1.53E−06	11,783	100
	MABC	8.87E−06	9.26E−07	14,437	100
f_{26}	AABCLS	1.70E−14	1.78E−14	3485	100
	ABC	2.16E−06	1.05E−05	120,927	54
	BSFABC	3.89E−07	4.78E−08	82,091	59
	GABC	4.73E−07	1.88E−08	96,611	53
	OBLFABC	1.66E−14	2.14E−14	4169	100
	MABC	5.12E−07	4.87E−08	109,023	51
f_{27}	AABCLS	1.15E−05	1.17E−05	474	100
	ABC	1.20E−05	1.07E−05	978	100
	BSFABC	1.72E−04	1.52E−05	99,419	49
	GABC	1.21E−04	1.14E−05	120,426	40
	OBLFABC	1.39E−05	1.39E−05	626	100
	MABC	1.57E−04	1.46E−05	92,860	54
f_{28}	AABCLS	1.95E−03	2.77E−06	3727	100
	ABC	1.95E−03	2.61E−06	26,936	100
	BSFABC	1.95E−03	3.02E−06	15,034	100
	GABC	1.95E−03	2.84E−06	5645	100
	OBLFABC	1.95E−03	2.56E−06	6725	100
	MABC	1.95E−03	2.79E−06	9115	100
f_{29}	AABCLS	7.35E−03	1.20E−03	24,405	100
	ABC	7.70E−03	1.94E−03	53,169	100
	BSFABC	7.45E−03	1.90E−03	60,906	100
	GABC	7.55E−03	2.22E−03	46,118	100
	OBLFABC	8.49E−03	1.48E−03	23,057	100
	MABC	6.17E−01	1.01E−01	200,000	0
f_{30}	AABCLS	7.24E−16	6.03E−17	39,289	100
	ABC	9.40E−16	6.42E−17	72,891	98
	BSFABC	7.35E−16	2.36E−16	71,216	100
	GABC	9.30E−16	8.16E−17	39,719	100
	OBLFABC	9.35E−16	7.65E−17	59,040	100
	MABC	9.26E−16	6.71E−17	59,746	100

average number of function evaluations used by the considered algorithms and mean error achieved by the algorithms to solve the different problems are not normally distributed, so a non-parametric statistical test is required to compare the performance of the algorithms. The Mann-Whitney U rank sum [29], a non-parametric test, is well established test for comparison among non-Gaussian data. In this paper, this test is performed on average number of function evaluations and ME at 5 % level of significance ($\alpha = 0.05$) between AABCLS–ABC, AABCLS–BSFABC, AABCLS–GABC, AABCLS–OBLFABC and AABCLS–MABC.

Tables 3 and 4 show the results of the Mann-Whitney U rank sum test for the average number of function evaluations and ME of 100 simulations. First the significant difference is observed by Mann–Whitney U rank sum test i.e., whether the two data sets are significantly different or not. If significant difference is not seen (i.e., the null hypothesis is accepted) then sign '=' appears and when significant difference is observed i.e., the null hypothesis is rejected then compare the AFE. The signs '+' and '−' are used for the case where AABCLS takes less or more average number of function evaluations than the other algorithms, respectively. Similarly, for mean error, if significant difference is observed then compare the ME and the signs '+' and '−' are used for the case where AABCLS achieves less or more mean error. Therefore in Tables 3 and 4, '+' shows that AABCLS is significantly better and '−' shows that AABCLS is significantly worse. As Tables 3 and 4 include 138 '+' signs for AFE case and 107 '+' signs for ME case out of 150 comparisons. Therefore, it can be concluded that the results of AABCLS is significantly cost effective than ABC, BSFABC, GABC, OBLFABC and MABC over considered test problems.

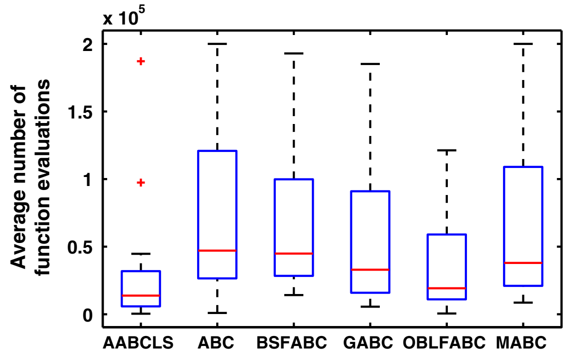
Further, the convergence speeds of the considered algorithms are compared by measuring the AFEs. A smaller AFEs means higher convergence speed. In order to minimize the effect of the stochastic nature of the algorithms, the reported function evaluations for each test problem is averaged over 100 runs.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

$$AR = \frac{AFE_{ALGO}}{AFE_{AABCLS}}, \tag{9}$$

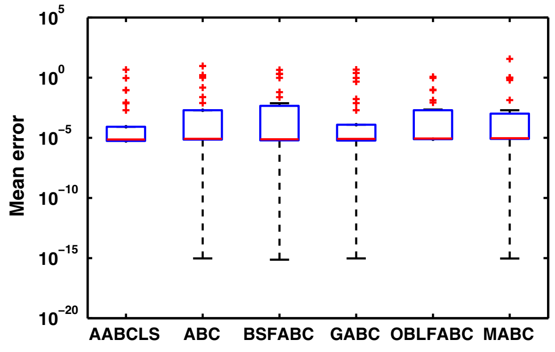
where, ALGO∈ {ABC, BSFABC, GABC, OBLFABC and MABC} and $AR > 1$ means AABCLS is faster. In order to investigate the AR of the proposed algorithm as compare to the considered algorithms, results of Table 2 are analyzed and the value of AR is calculated using Eq. (9). Table 5 shows a comparison between AABCLS–ABC, AABCLS–BSFABC, AABCLS–GABC, AABCLS–OBLFABC, and AABCLS–MABC in terms of AR. It is clear from the Table 5 that convergence speed of AABCLS is better than considered algorithms for most of the functions.

Further, to compare the considered algorithms by giving weighted importance to SR, AFE and ME, performance indices (PIs) are calculated [5]. The values of PI for the AABCLS, ABC, BSFABC, GABC, OBLFABC and MABC are calculated using following Equations:

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



(a) AFE



(b) ME

Fig. 5 *Boxplots graph* for average number of function evaluations and mean error

Table 3 Comparison based on average number of function evaluations and the Mann–Whitney U rank sum test at a $\alpha = 0.05$ significance level

TP	Mann–Whitney U rank sum test with AABCLS					TP	Mann–Whitney U rank sum test with AABCLS				
	ABC	BSFABC	GABC	OBLFABC	MABC		ABC	BSFABC	GABC	OBLFABC	MABC
f_1	+	+	+	+	+	f_{16}	+	+	+	+	+
f_2	+	+	+	+	+	f_{17}	+	+	+	+	+
f_3	+	+	=	+	+	f_{18}	+	+	+	+	+
f_4	–	–	–	–	+	f_{19}	+	+	+	+	+
f_5	+	+	+	+	+	f_{20}	+	+	+	=	+
f_6	+	+	+	+	+	f_{21}	+	+	+	+	+
f_7	+	+	+	+	+	f_{22}	+	+	+	–	+
f_8	+	+	+	+	+	f_{23}	+	+	+	+	+
f_9	+	+	+	+	+	f_{24}	+	+	+	+	+
f_{10}	+	+	+	+	+	f_{25}	+	+	+	+	+
f_{11}	+	+	+	+	+	f_{26}	+	+	+	=	+
f_{12}	+	+	+	–	+	f_{27}	+	+	+	=	+
f_{13}	+	+	+	+	+	f_{28}	+	+	+	+	+
f_{14}	+	+	+	+	+	f_{29}	+	+	+	–	+
f_{15}	+	+	+	+	+	f_{30}	+	+	=	+	+

TP test problem, '+' indicates AABCLS is significantly better, '–' indicates AABCLS is worse and '=' indicates that there is no significant difference)

Where $\alpha_1^i = \frac{Sr^i}{Tr^i}$; $\alpha_2^i = \begin{cases} \frac{Mf^i}{Af^i}, & \text{if } Sr^i > 0. \\ 0, & \text{if } Sr^i = 0. \end{cases}$; and $\alpha_3^i = \frac{Mo^i}{Ao^i}$ $i = 1, 2, ..., N_p$

- Sr^i = Successful simulations/runs of i^{th} problem.
- Tr^i = Total simulations of i^{th} problem.
- Mf^i = Minimum of AFE used for obtaining the required solution of i^{th} problem.
- Af^i = AFE 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 optimization problems evaluated.

The weights assigned to SR, AFE and 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 variable vary from 0 to 1 as given in [10]. 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 the considered algorithms are shown in Fig. 6a–c respectively. In these figures the weights k_1, k_2 and k_3 are

Table 4 Comparison based on mean error and the Mann–Whitney U rank sum test at a $\alpha = 0.05$ significance level

TP	Mann–Whitney U rank sum test with AABCLS					TP	Mann–Whitney U rank sum test with AABCLS				
	ABC	BSFABC	GABC	OBLFABC	MABC		ABC	BSFABC	GABC	OBLFABC	MABC
f_1	+	−	+	+	+	f_{16}	−	−	−	−	=
f_2	−	−	=	+	+	f_{17}	+	+	+	+	+
f_3	+	+	+	+	+	f_{18}	+	+	+	+	+
f_4	−	−	=	−	+	f_{19}	+	+	+	+	+
f_5	=	−	=	+	+	f_{20}	=	+	+	+	+
f_6	+	+	+	+	+	f_{21}	+	+	+	+	+
f_7	+	−	−	+	+	f_{22}	+	+	+	+	+
f_8	+	+	+	+	+	f_{23}	+	+	+	+	+
f_9	+	+	+	+	+	f_{24}	+	+	+	+	+
f_{10}	−	=	−	=	+	f_{25}	−	−	−	=	+
f_{11}	+	+	+	=	+	f_{26}	+	+	+	=	+
f_{12}	+	+	=	+	+	f_{27}	=	+	+	+	+
f_{13}	+	+	+	=	+	f_{28}	=	=	=	=	=
f_{14}	+	+	+	+	+	f_{29}	+	+	+	+	+
f_{15}	−	−	−	−	=	f_{30}	+	=	+	+	+

TP test problem, '+' indicates AABCLS is significantly better, '−' indicates AABCLS is worse and '=' indicates that there is no significant difference)

Table 5 Acceleration rate (AR) of AABCLS as compared to the ABC, BSFABC, GABC, OBLFABC and MABC, TP: test problems

TP	ABC	BSFABC	GABC	OBLFABC	MABC
f_1	1.801	2.645	2.135	1.534	2.013
f_2	1.732	4.021	1.502	1.891	3.717
f_3	1.435	1.349	1.051	1.339	1.336
f_4	0.978	0.961	0.942	0.648	1.015
f_5	1.542	1.345	1.491	1.181	1.348
f_6	2.179	2.601	1.306	1.801	3.995
f_7	7.913	2.888	2.003	1.314	2.884
f_8	8.091	6.845	5.265	3.540	1.219
f_9	3.223	2.716	1.200	1.426	1.942
f_{10}	4.800	4.280	4.220	2.183	2.737
f_{11}	5.070	4.529	2.325	1.388	1.936
f_{12}	2.561	2.910	1.913	0.699	1.953
f_{13}	15.034	14.755	14.154	1.461	11.147
f_{14}	3.198	3.267	1.307	1.498	2.205
f_{15}	2.485	1.831	1.225	1.417	2.124
f_{16}	2.434	1.837	2.052	1.440	1.769
f_{17}	3.074	2.890	1.160	1.350	1.870
f_{18}	16.974	17.100	8.475	3.535	6.110
f_{19}	12.187	9.513	9.807	6.469	8.663
f_{20}	2.868	27.621	29.544	1.429	26.112
f_{21}	4.789	3.911	2.420	1.692	4.683
f_{22}	1.830	1.755	1.070	0.745	1.647
f_{23}	1.998	3.080	1.837	1.439	1.874
f_{24}	4.345	4.915	1.807	4.451	3.994
f_{25}	1.769	3.137	1.538	1.182	1.449
f_{26}	34.700	23.556	27.723	1.196	31.284
f_{27}	2.061	209.577	253.860	1.319	195.750
f_{28}	7.227	4.034	1.515	1.804	2.446
f_{29}	2.179	2.496	1.890	0.945	8.195
f_{30}	1.855	1.813	1.011	1.503	1.521

represented by horizontal axis while the PI is represented by the vertical axis.

In case (1), AFE and ME are given equal weights. PIs of the considered algorithms are superimposed in Fig. 6a for comparison of the performance. It is observed that PI of AABCLS is higher than the considered algorithms. In case (2), equal weights are assigned to SR and ME and in case (3), equal weights are assigned to SR and AFE. It is clear from Fig. 6b and c that the algorithms perform same as in case (1).

6 Conclusion

ABC is a simple algorithm having very less parameters with drawbacks like premature convergence and poor in exploitation. In order to develop an ABC algorithm with better exploitation and exploration capabilities, this article proposed a modified position update equation for ABC in which individuals update their respective positions in guidance of global best individual on the basis of fitness. Fur-

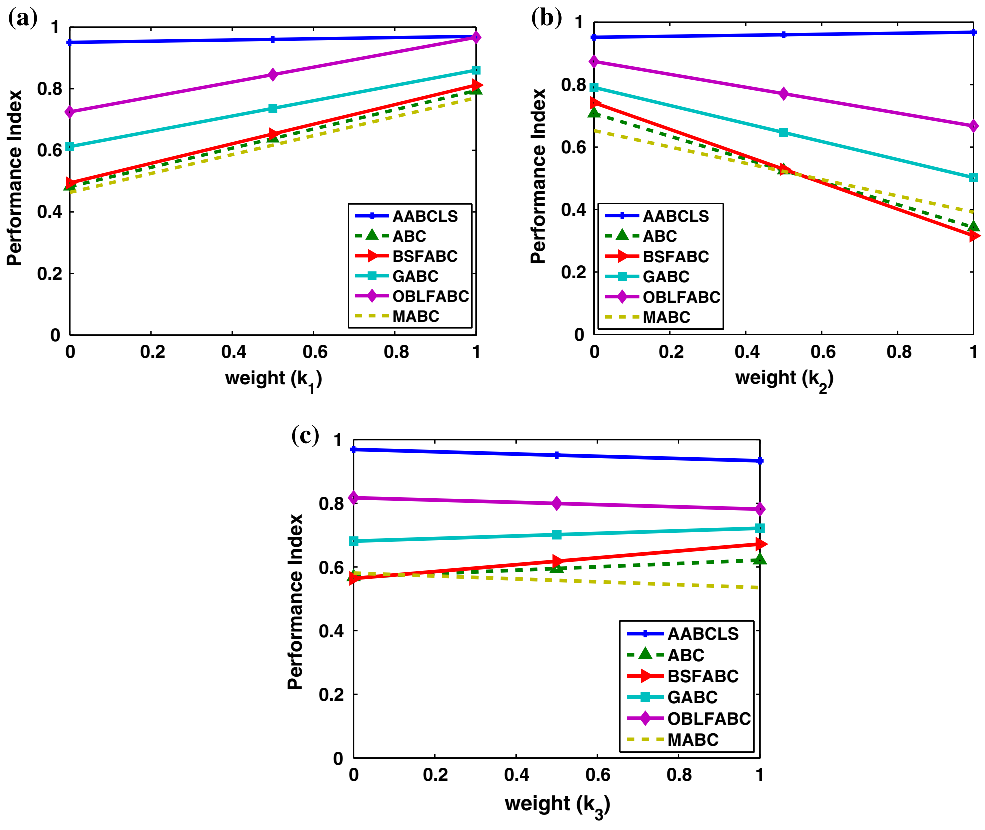


Fig. 6 Performance index for test problems; **a** for case (1), **b** for case (2) and **c** for case (3)

ther, a self adaptive local search strategy which is based on greedy logarithmic decreasing step size, is proposed and incorporated with ABC to enhance the exploitation capability. The proposed algorithm has been extensively compared with other recent variants of ABC namely, BSFABC, GABC, OBLFABC, and MABC. Based on various computational and statistical analyses, it is found that AABCLS achieves better success rate in less number of function evaluations with less ME on most of the problems considered. Through the extensive experiments, it can be stated that the proposed algorithm is a competitive algorithm to solve the continuous optimization problems. In future, work will be extended to solve constrained and real world problems.

References

1. Akay B, Karaboga D (2010) A modified artificial bee colony algorithm for real-parameter optimization. Inf Sci. doi:10.1016/j.ins.2010.07.015
2. Ali MM, Khompatraporn C, Zabinsky ZB (2005) A numerical evaluation of several stochastic algorithms on selected continuous global optimization test problems. J Glob Optim 31(4):635–672
3. Banharnsakun A, Achalakul T, Sirinaovakul B (2011) The best-so-far selection in artificial bee colony algorithm. Appl Soft Comput 11(2):2888–2901
4. Chand Bansal Jagdish, Harish Sharma, Atulya Nagar (2013) Memetic search in artificial bee colony algorithm. Soft Comput 17(10):1911–1928
5. Bansal JC, Sharma H (2012) Cognitive learning in differential evolution and its application to model order reduction problem for single-input single-output systems. Memet Comput 4(3):209–229
6. Brest J, Zumer V, Maucec MS (2006) Self-adaptive differential evolution algorithm in constrained real-parameter optimization. In: IEEE congress on evolutionary computation, 2006. CEC 2006, pp 215–222, IEEE
7. Caponio A, Cascella GL, Neri F, Salvatore N, Sumner M (2007) A fast adaptive memetic algorithm for online and offline control design of pmsm drives. Syst Man Cybern Part B: Cybern IEEE Trans 37(1):28–41
8. Caponio A, Neri F, Tirronen V (2009) Super-fit control adaptation in memetic differential evolution frameworks. Soft Comput-Fusion Found Methodol Appl 13(8):811–831
9. Cotta C, Neri F (2012) Memetic algorithms in continuous optimization. Handbook of memetic algorithms, pp 121–134
10. Deep K, Thakur M (2007) A new crossover operator for real coded genetic algorithms. Appl Math Comput 188(1):895–911
11. Diwold K, Aderhold A, Scheidler A, Middendorf M (2011) Performance evaluation of artificial bee colony optimization and new selection schemes. Memet Comput 3(3):149–162
12. Dorigo M, Di Caro G (1999) Ant colony optimization: a new meta-heuristic. In Evolutionary computation, 1999. CEC 99. In: Proceedings of the 1999 congress on, vol 2, IEEE
13. El-Abd M (2011) Performance assessment of foraging algorithms vs. evolutionary algorithms. Inf Sci 182(1):243–263
14. Fister I, Fister Jr I, Brest J, Žumer V (2012) Memetic artificial bee colony algorithm for large-scale global optimization. Arxiv preprint arXiv:1206.1074

15. Gallo C, Carballido J, Ponzoni I (2009) Bihea: a hybrid evolutionary approach for microarray biclustering. In: Guimarães KS, Panchenko A, Przytycka TM (eds) Advances in bioinformatics and computational biology. Springer, Berlin, Heidelberg, pp 36–47
16. Gao W, Liu S (2011) A modified artificial bee colony algorithm. Comput Oper Res 39(3):687–697
17. Gao Y, An X, Liu J (2008) A particle swarm optimization algorithm with logarithm decreasing inertia weight and chaos mutation. In: Computational intelligence and security, 2008. CIS'08. International conference on, vol 1, pp 61–65, IEEE
18. Goh CK, Ong YS, Tan KC (2009) Multi-objective memetic algorithms, vol 171. Springer Verlag, Berlin
19. Hooke R, Jeeves TA (1961) "Direct search" solution of numerical and statistical problems. J ACM (JACM) 8(2):212–229
20. Ishibuchi H, Yamamoto T (2004) Fuzzy rule selection by multi-objective genetic local search algorithms and rule evaluation measures in data mining. Fuzzy Sets Syst 141(1):59–88
21. Ishibuchi H, Yoshida T, Murata T (2003) Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling. IEEE Trans Evol Comput 7(2):204–223
22. Kang F, Li J, Ma Z (2011) Rosenbrock artificial bee colony algorithm for accurate global optimization of numerical functions. Inf Sci 181(16):3508–3531
23. Kang F, Li J, Ma Z, Li H (2011) Artificial bee colony algorithm with local search for numerical optimization. J Softw 6(3):490–497
24. Karaboga D (2005) An idea based on honey bee swarm for numerical optimization. Technical report TR06. Erciyes University Press, Erciyes
25. Karaboga D, Akay B (2009) A comparative study of artificial bee colony algorithm. Appl Math Comput 214(1):108–132
26. Dervis Karaboga, Bahriye Akay (2011) A modified artificial bee colony (abc) algorithm for constrained optimization problems. Appl Soft Comput 11(3):3021–3031
27. Kennedy J, Eberhart R (1995) Particle swarm optimization. In: Neural networks, 1995. In: Proceedings, IEEE international conference on, vol 4, pp 1942–1948, IEEE
28. Knowles J, Corne D, Deb K (2008) Multiobjective problem solving from nature: from concepts to applications (Natural computing series). Springer, Berlin
29. Mann HB, Whitney DR (1947) On a test of whether one of two random variables is stochastically larger than the other. Ann Math Stat 18(1):50–60
30. Mezura-Montes E, Velez-Koeppel RE (2010) Elitist artificial bee colony for constrained real-parameter optimization. In: 2010 Congress on evolutionary computation (CEC'2010). IEEE Service Center, Barcelona, Spain, pp 2068–2075
31. Mininno E, Neri F (2010) A memetic differential evolution approach in noisy optimization. Memet Comput 2(2):111–135
32. Moscato P (1989) On evolution, search, optimization, genetic algorithms and martial arts: towards memetic algorithms. Caltech concurrent computation program, C3P. Report 826:1989
33. Neri F, Tirronen V (2009) Scale factor local search in differential evolution. Memet Comput Springer 1(2):153–171
34. Neri F, Cotta C, Moscato P (eds) (2012) Handbook of memetic algorithms. Springer, Studies in computational intelligence, vol 379
35. Nguyen QH, Ong YS, Lim MH (2009) A probabilistic memetic framework. IEEE Trans Evol Comput 13(3):604–623
36. Ong YS, Keane AJ (2004) Meta-Lamarckian learning in memetic algorithms. IEEE Trans Evol Comput 8(2):99–110
37. Ong YS, Lim M, Chen X (2010) Memetic computation-past, present and future [research frontier]. Comput Intell Mag IEEE 5(2):24–31
38. Ong YS, Nair PB, Keane AJ (2003) Evolutionary optimization of computationally expensive problems via surrogate modeling. AIAA J 41(4):687–696
39. Passino KM (2002) Biomimicry of bacterial foraging for distributed optimization and control. Control Syst Mag IEEE 22(3):52–67
40. Price KV, Storn RM, Lampinen JA (2005) Differential evolution: a practical approach to global optimization. Springer Verlag, Berlin
41. Rahnamayan S, Tizhoosh HR, Salama MMA (2008) Opposition-based differential evolution. Evol Comput IEEE Trans 12(1):64–79
42. Repoussis PP, Tarantilis CD, Ioannou G (2009) Arc-guided evolutionary algorithm for the vehicle routing problem with time windows. Evol Comput IEEE Trans 13(3):624–647
43. Richer JM, Goëffon A, Hao JK (2009) A memetic algorithm for phylogenetic reconstruction with maximum parsimony. Evolutionary computation, machine learning and data mining in bioinformatics, pp 164–175
44. Ruiz-Torrubiano R, Suárez A (2010) Hybrid approaches and dimensionality reduction for portfolio selection with cardinality constraints. Comput Intell Mag IEEE 5(2):92–107
45. Sharma Harish, Bansal Jagdish Chand, Arya KV (2013) Opposition based lévy flight artificial bee colony. Memet Comput 5(3):213–227
46. Sharma Harish, Bansal Jagdish Chand, Arya KV (2013) Power law-based local search in differential evolution. Int J Comput Intell Stud 2(2):90–112
47. Sharma H, Jadon SS, Bansal JC, Arya KV (2013) Lèvy flight based local search in differential evolution. In: Swarm, evolutionary, and memetic computing, pp 248–259. Springer
48. Sharma TK, Pant M, Singh VP (2012) Improved local search in artificial bee colony using golden section search. arXiv preprint arXiv:1210.6128
49. Suganthan PN, Hansen N, Liang JJ, Deb K, Chen YP, Auger A, Tiwari S (2005) Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization. In: CEC 2005
50. Tan KC, Khor EF, Lee TH (2006) Multiobjective evolutionary algorithms and applications: algorithms and applications. Springer Science & Business Media
51. Tang K, Mei Y, Yao X (2009) Memetic algorithm with extended neighborhood search for capacitated arc routing problems. IEEE Trans Evol Comput 13(5):1151–1166
52. Arit Thammano, Ajchara Phu-ang (2013) A hybrid artificial bee colony algorithm with local search for flexible job-shop scheduling problem. Procedia Comput Sci 20:96–101
53. Vesterstrom J, Thomsen R (2004) A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems. In: Evolutionary computation, 2004. CEC2004. Congress on, vol 2, pp 1980–1987, IEEE
54. Wang H, Wang D, Yang S (2009) A memetic algorithm with adaptive hill climbing strategy for dynamic optimization problems. Soft Comput-Fusion Found Methodol Appl 13(8):763–780
55. Williamson DF, Parker RA, Kendrick JS (1989) The box plot: a simple visual method to interpret data. Ann Intern Med 110(11):916
56. Zhu G, Kwong S (2010) Gbest-guided artificial bee colony algorithm for numerical function optimization. Appl Math Comput 217(7):3166–3173