Int J Syst Assur Eng Manag
DOI 10.1007/s13198-014-0286-6

ORIGINAL ARTICLE

Artificial bee colony algorithm with global and local neighborhoods

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

Received: 14 May 2014 / Revised: 18 July 2014

Abstract Artificial Bee Colony (ABC) is a well known population based efficient algorithm for global optimization. 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 slow convergence are also associated with it. In this article, basic ABC algorithm is studied by modifying its position update equation using the differential evolution with global and local neighborhoods like concept of food sources' neighborhoods. Neighborhood of each colony member includes 10 % members from the whole colony based on the index-graph of solution vectors. The proposed ABC is named as ABC with Global and Local Neighborhoods (ABCGLN) which concentrates to set a trade off between the exploration and exploitation and therefore increases the convergence rate of ABC. To validate the performance of proposed algorithm, ABCGLN is tested over 24 benchmark optimization functions and compared with standard ABC as well as its recent popular variants namely, Gbest guided ABC, Best-So-Far ABC and Modified ABC. Intensive statistical analyses of the results shows that ABCGLN is significantly better and takes on an average half number of function evaluations as compared to other considered algorithms.

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 · 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. Collaborative trial and error method is the main concept behind the swarm intelligence which enables the algorithmic procedure to find the solution. Researchers are analyzing such collaboration among the social insects while searching food for them and creating the intelligent structures known as Swarm Intelligence. Spider monkey optimization (SMO) (Bansal et al. 2013), Ant colony optimization (ACO) (Dorigo and Di Caro 1999), Particle swarm optimization (PSO) (Kennedy and Eberhart 1995), Bacterial foraging optimization (BFO) (Passino 2002) are some examples of swarm intelligence based techniques. The work presented in the articles (Dorigo and Di Caro 1999; Kennedy and Eberhart 1995; Price et al. 2005; Vesterstrom and Thomsen 2004) proved its efficiency and potential to deal with non linear, non convex and discrete optimization problems.

Karaboga (2005) contributed the recent addition to this category known as Artificial bee colony (ABC) optimization algorithm. ABC is a simple but effective algorithm for both continuous and discrete optimization problems. The ABC algorithm has emerged as one of the popular tool in the machine intelligence and has successfully been tested on almost all domains of science and engineering like electronics engineering (Kavian et al. 2012; Chidambaram and Lopes 2009), electrical engineering (Jones and Bouffet 2008; Nayak et al. 2009; Sulaiman et al. 2012), computer science engineering (Lei et al. 2010; Karaboga and Cetinkaya 2011; Lam et al. 2012), mechanical engineering (Pawar et al. 2008; Xu and Duan 2010; Banharnsakun et al. 2012), civil engineering (Li et al. 2011; Mandal et al. 2012; Akay and Karaboga 2012), medical pattern classification and clustering problems (Akay et al. 2008) and mathematical graph problems (Xing et al. 2007; Singh 2009; Yeh and Hsieh 2011). Many of the recent modifications and applications of ABC algorithm can be studied in Bansal et al. (2013).

The ABC algorithm mimics the foraging behavior of honey bees while searching food for them. ABC is a simple and population based optimization algorithm. Here the population consists of possible solutions in terms of food sources for honey bees whose fitness is regulated 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. However the ABC achieves a good solution but, like the other optimization algorithms, it has also problem of premature convergence and stagnation. On the other part, it is also required to tune the ABC control parameters based on problem. Also literature says 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 (Karaboga and Akay 2009) and it is observed that the position update equation of ABC algorithm is inefficient to balance exploration and exploitation (Zhu and Kwong 2010). Therefore these drawbacks require a modification in position update equation of ABC in order to make it capable to balance exploration and exploitation. These drawbacks have also addressed in earlier research. To enhance the exploitation, Wei-feng Gao and Liu (2011) improved position update equation of ABC such that the bee searches only in neighborhood of the previous iteration's best solution. Anan Banharnsakun et al. (2011) 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, D. Karaboga and B. Akay (2011) used Deb's rules consisting of three simple heuristic rules and a probabilistic selection scheme in ABC algorithm. Dervis Karaboga (2005) 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. (2011) introduced exploitation phase in ABC using Rosenbrock's rotational direction method and named modified ABC as Rosenbrock ABC (RABC). Qingxian and Haijun proposed a new initialization scheme by making the initial group symmetrical and to increase the ABC's convergence, they applied Boltzmann selection mechanism in place of roulette wheel selection (Haijun and Qingxian 2008). Tsai et al. (2009) proposed Interactive ABC (IABC) in which author applied the roulette wheel selection to select onlooker bees and he also used Newtonian law of universal gravitation to enhance the exploitation in onlooker bee phase. Zhu et al. (2010) also modified the position update equation of basic ABC by incorporated best individual member information in its position update equation. Baykasoglu et al. (2007) solved the generalized assignment problem by hybriding the ABC algorithm with shift neighborhood searches and greedy randomized adaptive search heuristic. Furthermore, Bahriye Akay and Dervis Karaboga (2010) also presented an improved ABC to solve real-parameter optimization problems and they also analyzed the effects of the perturbation rate, the scaling factor *limit* parameter on real-parameter optimization. Some other important improved versions of ABC can be found in Bansal et al. (2013); Sharma et al. (2013); Bansal et al. (2013); Jadon et al. (2014).

Inspired from the article "Differential evolution using a neighborhood-based mutation operator (DEGL) "(Das et al. 2009), we propose a new position update equation for employed bees in ABC which linearly incorporate two components namely local and global components in its swarm updating process in order to set a trade off between the exploration and exploitation capabilities of ABC. Here, local component is responsible for explorative moves and has better chance of locating the minima of test function. On the other hand, global component is responsible for exploitive moves and rapidly converges to a minimum of the test function.

Rest of the paper is organized as follows. Next Sect. 2 explains standard ABC algorithm. In Sect. 3, proposed modified ABC (ABCGLN) is described. Section 4 details about the benchmark mathematical optimization functions used in this article and also statistical analysis to compare the performance of the proposed strategy with respect to other recent ABC variants. Finally, in Sect. 5, paper is concluded.

2 Artificial Bee Colony(ABC) algorithm

The ABC algorithm is a population based recent swarm intelligence based 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 group of bees: employed bees, onlooker bees and scout bees. The number of employed and onlooker bees is equal. The employed bees searches 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 that food sources which are having better nectar. If any food source is abandoned then scout bee finds new food source randomly in 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. In ABC, first initialization of the solutions is done as:

2.1 Initialization of the swarm

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:

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

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.

2.2 Employed bees phase

In this phase, ith candidate's position is updated using following equation:

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

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, the position with better fitness between the newly generated and old one is selected.

2.3 Onlooker bees phase

In this phase, employed bees share the information associated with their food sources 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 its fitness they select solutions with probability $prob_i$. Here $prob_i$ can be calculated as function of fitness (there may be some other):

$$prob_i(G) = \frac{0.9 \times fitness_i}{maxfit} + 0.1, \tag{3}$$

here $fitness_i$ is the fitness value of the ith solution and $maxfit$ is the maximum fitness amongst all the solutions. Based on this probability, onlooker selects a solution and modifies it using the same Eq. (2) 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 position in its memory and forgets the old one.

2.4 Scout bees phase

If for a predetermined number of cycles, any bee's position is not getting updated then that food source is taken to be abandoned and this bee becomes scout bee. In this phase, the abandoned food source is replaced by a randomly chosen food source within the search space. In ABC, the number of cycles after which a particular food source becomes abandoned is known as *limit* and is a crucial control parameter. In this phase the abandoned food source x_i is replaced by a randomly chosen food source within the search space using the Eq. (1) as in initialization phase.

2.5 Main steps of the ABC algorithm

The pseudo-code of the ABC is shown in Algorithm 1 (Karaboga and Akay 2009).

Algorithm 1 Artificial Bee Colony Algorithm:

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

3 Artificial bee colony algorithm with global and local neighborhoods

For an efficient optimization algorithm, two concepts exploration and exploitation should be well balanced which are actually contradictory to each other in nature. Exploration means the ability of exploring diverse regions of the search space and exploitation means refining already explored regions. 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 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 whole ABC process. Further, It is also suggested in literature (Karaboga and Akay 2009; Zhu and Kwong 2010) 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. Karaboga and Akay (2009) also analyzed the various variants of ABC and found that the ABC in its current form shows poor performance and remains inefficient to balance the exploration and exploitation capabilities of the search space. Consequently, convergence speed of ABC is also deteriorated.

In above context, we are proposing a modified ABC by incorporating local and global neighborhoods concept in its position update equation of employed bee phase. This modified ABC is now onward called as ABCGLN. The proposed modified position update equation is the inspiration from DEGL of Das et al. (2009). In DEGL, author proposed a very strong and promising DE version which includes two neighborhood models; the local neighborhood model and the global neighborhood model. In local neighborhood model each member is updated through best solution found so far in its sub population (i.e, local neighborhood), while in the global neighborhood model each member takes the advantage of best solution found so far in the whole population. We adopted both neighborhood strategies of DEGL in the proposed ABCGLN algorithm.

In ABCGLN, the neighborhood and neighborhood structure of any bee are crucial parts which need to be focussed. It should be clear here that the neighborhood structure of any solution is static and has been defined on the set of indices of the solutions. Neighborhood of a solution is the set of other solutions to which it is connected. For global neighborhood model, neighborhood structure is the star topology i.e, each solution is connected to each other solution and therefore cardinality of neighborhood of any solution in this model is equal to the number of food sources. On the other hand, for local neighborhood model, ring topology is adopted as neighborhood structure and cardinality of neighborhood is considered as 10 % of the number of food sources i.e, neighborhood of each solution includes 10 % solutions based on their indices as 5 % from the forward side and 5 % from the backward side. Here, it should be noticed that solutions belonging to a local neighborhood are not necessarily local in the sense of their geographical nearness or similar fitness values but, the overlapping neighborhoods based on the indices of the swarm members have been considered in ABC as shown in Fig. 1. For example, if population size is 60 then local neighborhood of $20th$ indexed solution will include solutions from indexed $17th$ to $23th$, similarly $21st$ indexed solution's neighborhood includes solutions from indexed $18th$ to $24th$ (see Fig. 1). In ABCGLN, for each employed bee, two components a local and a global component are created. The local component for a bee is created based on best solution position in that bee's local neighborhood and two other solutions selected randomly from this neighborhood while global component is created based on best solution of whole swarm and two other solutions selected randomly from the swarm i.e, from the global neighborhood. The mathematical equations for these components are as follows:

$$\begin{aligned} L_{ij} &= x_{ij} + (prob_i)(x_{lj} - x_{ij}) + \phi_{ij}(x_{r1j} - x_{r2j}), \\ G_{ij} &= x_{ij} + (1 - prob_i)(x_{gj} - x_{ij}) + \phi_{ij}(x_{R1j} - x_{R2j}) \end{aligned} \tag{4}$$

where, each j represents the jth dimension of the each position, L_i and G_i are the local and global components respectively. x_l and x_g are respectively the best positions in local and global neighborhoods of ith solution. x_{r1}, x_{r2} are the two neighbors chosen randomly from the local neighborhood of ith solution such that $r1 \neq r2 \neq i$. x_{R1}, x_{R2} are the two neighbors chosen randomly from the global neighborhood i.e, from whole swarm such that $R1 \neq R2 \neq i$. ϕ_{ij} is random number in $[-1, 1]$ and $prob_i$ is the probability of ith solution as in Eq. (3). Here in both equations the third term i.e, difference between two randomly selected neighbors is added so that individual member should not follow only the best position in order to prevent the swarm from getting trapped in local minima. Also second term in both the equations is multiplied by their probabilities which is a function of fitness. It is clear that solution with better fitness will give more weightage to its local best than the global best in order to prevent the swarm to converge too quickly as global best may be a local optima in initial stages. On the other hand the solution with low fitness will give priority to global best more than the local best as there surroundings are not good enough so they follow better directions. Finally both the local and the global components together guide the new direction for employed bees as follows: for *ith* employed bee, the new position is,

$$v_{ij} = (prob_i) \times G_{ij} + (1 - prob_i) \times L_{ij} \quad (5)$$

Now based on greedy selection, employed bee selects one between the old and new positions. In this way, the involvement of weighted linear sum of both local and global best components, is expected to balance the exploration and exploitation capabilities of ABC algorithm as the local best components of respective bees explore the search space or tries to identify the most promising search space regions, while the global best component will exploit the identified search space. The pseudo code of implementation of this whole neighborhood concept in employed bee phase of ABC is explained in Algorithm 2. Algorithm 2 introduced a new parameter cr. For each dimension of employed bee, a random number between 0 and 1 is generated and if it is less than cr value then that particular dimension will be changed. Actually, in basic ABC, only one random dimension of employed bee is changed for employed in order to generate neighborhood solution but in proposed algorithm ABCGLN, the number of dimensions to be changed are selected based on probability cr. In ABCGLN, cr is set to a low value 0.3 based on its sensitive analysis carried out in Fig. 2.

The ABCGLN is composed of three phases: employed bee phase, onlooker bee phase and scout bee phase. Only the employed bee phase is changed and other phases; onlooker and scout bee remain same as in basic ABC. Here the new position search strategy (explained in Eqs. (4) and (5)) is proposed for employed bees. The pseudo-code of the proposed ABCGLN algorithm is shown in Algorithm 3.

Algorithm 3 ABC with Global and Local neighborhoods (ABCGLN):

```
Initialize the population and control parameters;
while Termination criteria is not satisfied do
  Memorize the best food source found so far;
  Calculate prob_i using Equation (3) for each bee;
  Employed bee phase: apply Algorithm (2) to generate new food sources for each employed bee.
  Onlooker bees phase: use Equation (2) to update the food sources for particular bees selected based on prob_i;
  Scout bee phase: use Equation (1) to determine the new food sources for exhausted food sources.
end while
Return the best solution found so far.
```

4 Experimental results and discussion

4.1 Test problems under consideration

To validate the effectiveness of the proposed algorithm ABCGLN, 24 mathematical optimization problems (f_1 to f_{24}) of different characteristics and complexities are taken into consideration (listed in Table 1). These all problems are continuous in nature. Test problems $f_1 - f_{17}$ and $f_{22} - f_{24}$ are taken from Ali et al. (2005) and test problems $f_{18} - f_{21}$ are taken from Suganthan et al. (2005) with the associated offset values. Table 1 includes various kind of unimodal, multimodal, separable and non separable test problems. A unimodal function $f(x)$ has a single extremum (minimum or maximum in the range specified for x) while, a function having more than one peaks in the search space i.e., local extremum, is said multimodal. Algorithms are tested on this type of functions to check their ability to coming out of local minima.

Algorithm 2 Proposed employed bee phase for ABCGLN:

```
Input employed bee i, prob_i and best solution g found so far in whole swarm;
Find best solution l in local neighborhood of i^th bee;
Select two neighbors x_r1, x_r2 randomly from the local neighborhood of i^th bee such that r1 ≠ r2 ≠ i;
Select two neighbors x_R1, x_R2 randomly from whole swarm such that R1 ≠ R2 ≠ i;
for (each dimension j) do
  if (rand < cr) then { # rand is a uniform random number between 0 and 1}
    Generate local component L_ij and global component G_ij using Equation (4);
    Generate new position v_ij using Equation (5);
  else
    v_ij = x_ij ;
  end if
end for
Return better of old position x_ij and new position v_ij for employed bee i;
```

4.2 Experimental setting

The results obtained from the proposed ABCGLN are stored in the form of success rate, average number of function evaluations, standard deviation of the fitness and mean error. Results for test problems (Table 1) are also obtained from the basic ABC and recent variants of ABC named Gbest-guided ABC (GABC) (Zhu and Kwong 2010), Best-So-Far ABC (BSFABC) (Banharnsakun et al. 2011) and Modified ABC (MABC) (Akay and Karaboga 2010) 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 $NP = 50$ (Diwold et al. 2011; El-Abd 2011) and Number of food sources $SN = NP/2$,
- $\phi_{ij} = rand[-1, 1]$ and limit = dimension × number of *food sources* $= D \times SN$ (Karaboga and Akay 2011; Akay and Karaboga 2010),
- The terminating criteria: Either acceptable error (mentioned in Table 1) meets or maximum number of function evaluations (which is set to be 200000) is reached,

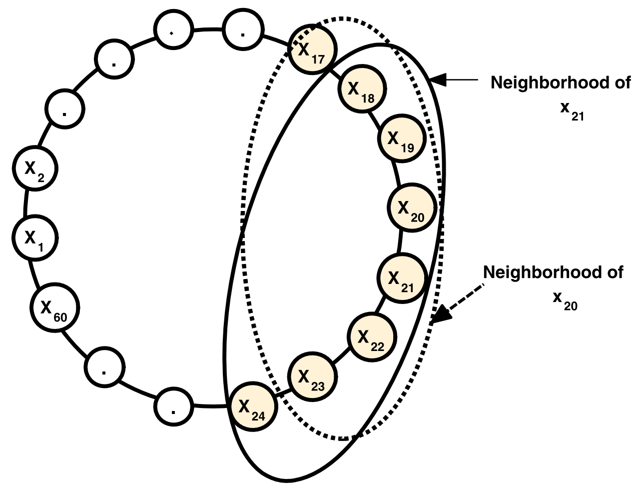


Fig. 1 Local neighborhood topology for radius 3

times, algorithm achieved the function optima with acceptable error in 100 runs, *AFE* is the average number of function evaluations of 100 runs by the algorithm to reach at the termination criteria and *ME* is the mean of absolute difference between exact and obtained solutions over 100 runs. Mathematically *ME* and *AFE* are defined as:

$$ME = \frac{\sum_{i=1}^{100} |\text{ Exact solution} - \text{Obtained solution } | \text{ for run i}}{100}$$

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

- The parameter *cr* in Algorithm 2 is set to 0.3 based on its sensitive analysis in range $[0.1, 1]$ as explained in the Fig. 2. Figure 2 shows a graph between *cr* and the sum of the successful runs for all the considered problems. It is clear that $c_r = 0.3$ provides the highest success rate.
- Parameter settings for the other considered algorithms ABC, GABC, BSFABC and MABC are adopted from their original articles.

4.3 Results analysis of experiments

Tables 2, 3 and 4 present the numerical results comparison of all considered algorithms for benchmark problems $f_1 - f_{24}$ with the experimental settings shown in Sect. 4.2. Tables 2 , 3 and 4 show the results of the proposed and other considered algorithms in terms of success rate (SR), average number of function evaluations (AFE) and mean error (ME) respectively. Here *SR* represents the number of

It can be seen from Table 2 that ABCGLN is more reliable as it has achieved higher or equal success rate than all the considered algorithms on all functions except f_3. It can be observed from Table 3 that most of the time, ABCGLN is more efficient as it took less number of function evaluations on 23 out of 24 test problems than the ABC and other considered modified ABC algorithms. Table 4 shows a clear superiority of ABCGLN in terms of accuracy as ABCGLN's mean error is less than the mean error achieved by the other algorithms on 19 out of 24 test functions. While comparing all three factors simultaneously then it can be analyzed that ABCGLN costs less than the considered algorithms over 19 test problems ($f_1 - f_2, f_4 - f_7, f_9 - f_{10}, f_{12} - f_{13}, f_{15} - f_{20}$ and $f_{22} - f_{24}$) out of 24 test functions. As these functions include unimodel, multimodel, separable, non separable, lower and higher dimensional functions, it can be stated that ABCGLN balances the exploration and exploitation capabilities efficiently on most of the functions. This should be

Table 1 Benchmark functions used in experiments

Test problem	Objective function	Search range	Optimum value	D	C	AE
Sphere	$f_1(x) = \sum_{i=1}^{D} x_i^2$	[−5.12 5.12]	$f(\boldsymbol{0}) = 0$	30	US	$1.0E-05$
Griewank	$f_2(x) = 1 + \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D}\cos(\frac{x_i}{\sqrt{i}})$	[−600 600]	$f(\boldsymbol{0}) = 0$	30	MN	$1.0E-05$
Rosenbrock	$f_3(x) = \sum_{i=1}^{D}(100(x_{i+1} - x_i{}^2)^2 + (x_i - 1)^2)$	[−30 30]	$f(\boldsymbol{1}) = 0$	30	UN	$1.0E-02$
Ackley	$f_4(x) = -20 + e + exp\left(-\frac{0.2}{D}\sqrt{\sum_{i=1}^{D} x_i{}^3}\right)$ $-exp(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)x_i)$	[−1 1]	$f(\boldsymbol{0}) = 0$	30	MN	$1.0E-05$
Cosine Mixture	$f_5(x) = \sum_{i=1}^{D} x_i{}^2 - 0.1(\sum_{i=1}^{D}\cos 5\pi x_i) + 0.1D$	[−1 1]	$f(\boldsymbol{0}) = -D \times 0.1$	30	MS	$1.0E-05$
Exponential	$f_6(x) = -(exp(-0.5\sum_{i=1}^{D} x_i{}^2)) + 1$	[−1 1]	$f(\boldsymbol{0}) = -1$	30	MN	$1.0E-05$
Zakharov	$f_7(x) = \sum_{i=1}^{D} x_i{}^2 + (\sum_{i=1}^{D}\frac{ix_i}{2})^2 + (\sum_{i=1}^{D}\frac{ix_1}{2})^4$	[−5.12 5.12]	$f(\boldsymbol{0}) = 0$	30	MN	$1.0E-02$
Brown3	$f_8(x) = \sum_{i=1}^{D-1}(x_i{}^{2^{(x_{i+1})^2+1}} + x_{i+1}{}^{2^{x_i{}^2+1}})$	[−1 4]	$f(\boldsymbol{0}) = 0$	30	UN	$1.0E-05$
Schewel	$f_9(x) = \sum_{i=1}^{D}\lvert x_i\rvert + \prod_{i=1}^{D}\lvert x_i\rvert$	[−10 10]	$f(\boldsymbol{0}) = 0$	30	UN	$1.0E-05$
Salomon Problem	$f_{10}(x) = 1 - cos\left(2\pi\sqrt{\sum_{i=1}^{D} x_i^2}\right) + 0.1\left(\sqrt{\sum_{i=1}^{D} x_i^2}\right)$	[−100 100]	$f(\boldsymbol{0}) = 0$	30	MN	$1.0E-01$
Powel Sum of different powers	$f_{11}(x) = \sum_{i=1}^{D}\lvert x_i\rvert^{i+1}$	[−1 1]	$f(\boldsymbol{0}) = 0$	30	US	$1.0E-05$
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 5]	$f(\boldsymbol{0}) = -D + 1$	10	MN	$1.0E-05$
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	UN	$1.0E-01$
Levy montalvo 2	$f_{14}(x) = 0.1(\sin^2(3\pi x_1) + \sum_{i=1}^{D-1}(x_i - 1)^2 \times (1 + \sin^2(3\pi x_{i+1})) + (x_D - 1)^2(1 + \sin^2(2\pi x_D))$	[−5 5]	$f(\boldsymbol{1}) = 0$	30	MN	$1.0E-05$
Beale	$f_{15}(x) = [1.5 - x_1(1 - x_2)]^2 + [2.25 - x_1(1 - x_2^2)]^2 + [2.625 - x_1(1 - x_2^3)]^2$	[−4.5 4.5]	$f(3, 0.5) = 0$	2	UN	$1.0E-05$
Colville	$f_{16}(x) = 100[x_2 - x_1^2]^2 + (1 - x_1)^2 + 90(x_4 - x_3^2)^2 + (1 - x_3)^2 + 10.1[(x_2 - 1)^2 + (x_4 - 1)^2] + 19.8(x_2 - 1)(x_4 - 1)$	[−10 10]	$f(\boldsymbol{1}) = 0$	4	MN	$1.0E-05$
Kowalik	$f_{17}(x) = \sum_{i=1}^{11}[a_i - \frac{x_1(b_i^2+b_ix_2)}{b_i^2+b_ix_3+x_4}]^2$	[−5 5]	$f(0.192833, 0.190836, 0.123117, 0.135766) = 0.000307486$	4	MN	$1.0E-05$
Shifted Rosenbrock	$f_{18}(x) = \sum_{i=1}^{D-1}(100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2) + f_{bias}$, $z = x - o + 1$, $x = [x_1, x_2, \ldots x_D]$, $o = [o_1, o_2, \ldots o_D]$	[−100 100]	$f(o) = f_{bias} = 390$	10	MN	$1.0E-01$
Shifted Sphere	$f_{19}(x) = \sum_{i=1}^{D} z_i^2 + f_{bias}$, $z = x - o$, $x = [x_1, x_2, \ldots x_D]$, $o = [o_1, o_2, \ldots o_D]$	[−100 100]	$f(o) = f_{bias} = -450$	10	US	$1.0E-05$
Shifted Griewank	$f_{20}(x) = \sum_{i=1}^{D}\frac{z_i^2}{4000} - \prod_{i=1}^{D}\cos\left(\frac{z_i}{\sqrt{i}}\right) + 1 + f_{bias}$, $z = (x - o)$, $x = [x_1, x_2, \ldots x_D]$, $o = [o_1, o_2, \ldots o_D]$	[−600 600]	$f(o) = f_{bias} = -180$	10	MN	$1.0E-05$
Shifted Ackley	$f_{21}(x) = -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} z_i^2}\right) - \exp(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi z_i)) + 20 + e + f_{bias}$, $z = (x - o)$, $x = (x_1, x_2, \ldots\ldots x_D)$, $o = (o_1, o_2, \ldots\ldots o_D)$	[−32 32]	$f(o) = f_{bias} = -140$	10	MN	$1.0E-05$
Goldstein-Price	$f_{22}(x) = (1 + (x_1 + x_2 + 1)^2 \cdot (19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)) \cdot (30 + (2x_1 - 3x_2)^2 \cdot (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2))$	[−2 2]	$f(0, -1) = 3$	2	MN	$1.0E-14$
Meyer and Roth	$f_{23}(x) = \sum_{i=1}^{5}(\frac{x_1x_3t_i}{1+x_1t_i+x_2v_i} - y_i)^2$	[−10 10]	$f(3.13, 15.16, 0.78) = 0.4 \times 10^{-4}$	3	UN	$1.0E-03$
Moved axis parallel hyper-ellipsoid	$f_{24}(x) = \sum_{i=1}^{D} 5i \times x_i^2$	[−5.12 5.12]	$f(x) = 0; x(i) = 5 \times i, i = 1 : D$	30	MS	$1.0E-15$

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

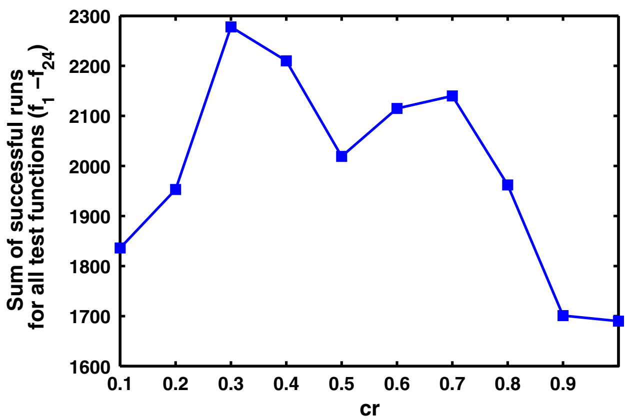


Fig. 2 Effect of parameter *cr*

noticed that ABCGLN performed worst than all considered algorithms on unimodel nonseparable test function f_3. Some other statistical tests like the Mann-Whitney U rank sum test, acceleration rate (AR) (Rahnamayan et al. 2008), boxplots and performance indices (Bansal and Sharma 2012) have also been done in next subsection to analyze the algorithms output more intensively.

4.4 Statistical analysis

Since the empirical distribution of results can efficiently be represented by boxplot (Williamson et al. 1989) , the boxplots for success rate (SR), average number of function evaluations (AFE) and mean error (ME) for ABCGLN and considered algorithms have been represented in Fig. 3 (a), (b), (c). Figures 3 (a), (b), (c) show that ABCGLN is cost effective in terms of SR, AFE and ME as the interquartile range and median are very low for ABCGLN. It can be observed from Fig. 3(a) that box of ABCGLN for SR is not being seen, as in almost all cases, ABCGLN achieved the 100 % success rate so there is negligible variation which boxplot is unable to represent and hence is showing a constant line segment instead of variation box. Boxplots for average number of function evaluations in Fig. 3(b) shows clear superiority of ABCGLN over other considered algorithms as the interquartile range and median for ABCGLN are very low. Boxplots for mean error in Fig. 3(c) represents that ABCGLN and MABC achieved almost equal and less average error than the other algorithms.

Now, it is clear from boxplot for AFE in Figure 3(b) that results of ABCGLN differs from the other algorithms. But to check, significant difference between ABCGLN's and other algorithm's output or the difference is due to some randomness, we applied another statistical test namely, the Mann-Whitney U rank sum (Mann and Whitney 1947) on AFE. We have chosen this non parametric test to compare the performance of the algorithms as boxplots of Fig. 3 (b) show that average number of function evaluations used by the considered algorithms to solve the different problems are not normally distributed. The test is performed at 5 % level of significance ($\alpha = 0.05$) between our proposed ABCGLN and each other considered algorithm.

The results of the Mann-Whitney U rank sum test for the average function evaluations of 100 simulations is presented in Table 5. First we checked that two data sets are significantly different or not through Mann-Whitney U rank sum test. 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 we compared the average number of function evaluations. And we use signs '+' and '−' for the case where ABCGLN takes less or more average number of function evaluations than the other algorithms, respectively. Therefore in Table 5, '+' indicates that ABCGLN is significantly better and '−' shows that ABCGLN is significantly worse, while '=' means that performance of both algorithms is equivalent. Since Table 5 contains 91 '+' signs out of 96 comparisons, it can be concluded that ABCGLN performance is significantly cost effective than other considered algorithms over test problems of Table 1.

Further, to compare the considered algorithms by giving weighted importance to SR, AFE and ME, performance indices (*PIs*) are calculated (Bansal and Sharma 2012). The values of *PI* for the ABCGLN and other considered algorithms, are calculated using following equations:

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

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

- k_1, k_2 and k_3 are the weights assigned to SR, AFE and ME respectively, where $k_1 + k_2 + k_3 = 1$ and $0 \le k_1, k_2, k_3 \le 1$.
- Sr^i = Successful simulations/runs of *ith* problem.
- Tr^i = Total simulations of *ith* problem.
- Mf^i = Minimum of mean number of function evaluations performed to achieve the required solution of *ith* problem.
- Af^i = Average number of function evaluations performed to achieve the required solution of *ith* problem.
- Mo^i = Minimum of mean error obtained for the *ith* problem.
- Ao^i = Mean Error achieved by an algorithm for the *ith* problem.
- N_p = Total number of optimization problems evaluated.

To calculate the *PI*s, equal weights are assigned to two out of SR, AFE and ME variables and weight of the remaining

Table 2 Success Rate (SR) achieved by the algorithms in 100 runs, *TP*: Test Problems

TP	ABC	BSFABC	GABC	MABC	ABCGLN	TP	ABC	BSFABC	GABC	MABC	ABCGLN
f_1	100	100	100	100	100	f_{13}	3	9	23	30	89
f_2	97	100	100	100	100	f_{14}	100	100	100	100	100
f_3	22	24	25	10	9	f_{15}	96	98	100	100	100
f_4	100	100	100	100	100	f_{16}	1	31	33	39	93
f_5	100	100	100	100	100	f_{17}	21	57	88	27	94
f_6	100	100	100	100	100	f_{18}	21	24	56	34	100
f_7	0	0	0	0	100	f_{19}	100	100	100	100	100
f_8	100	100	100	100	100	f_{20}	83	82	98	92	98
f_9	100	100	100	100	100	f_{21}	100	100	100	100	100
f_{10}	57	63	97	99	100	f_{22}	53	57	53	54	100
f_{11}	100	100	100	100	100	f_{23}	100	99	100	100	100
f_{12}	84	84	98	100	100	f_{24}	99	100	100	100	100

Table 3 Average number of function evolutions (AFE) done by the algorithms in 100 runs, *TP*: Test Problems

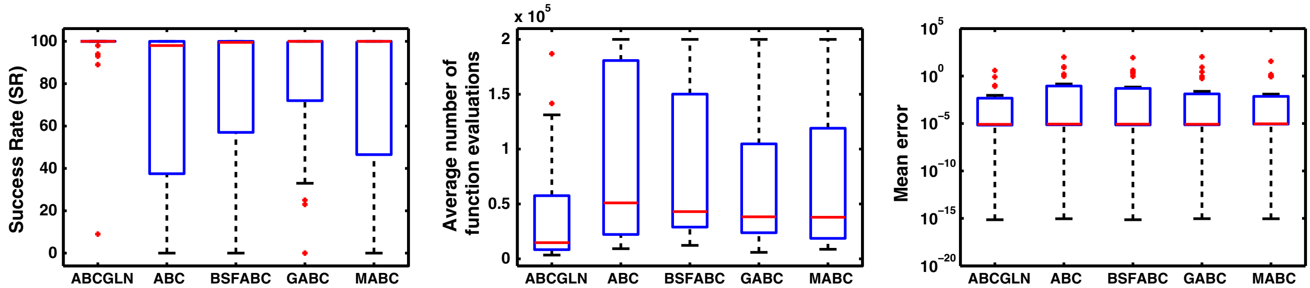
TP	ABC	BSFABC	GABC	MABC	ABCGLN	TP	ABC	BSFABC	GABC	MABC	ABCGLN
f_1	20639	29864	24937	23004	**9591**	f_{13}	198760	194001	186012	169918	**141626**
f_2	46004	43115	33900	42835	**24210**	f_{14}	28861	21968	26702	20921	**10012**
f_3	183117	180542	**175911**	189716	186909	f_{15}	28215	27917	14007	10818	**6347**
f_4	49107	42879	47858	43630	**18046**	f_{16}	199667	159666	160337	140006	**77760**
f_5	23016	31789	28572	22764	**9962**	f_{17}	179860	142527	99409	170962	**59392**
f_6	16836	16791	22576	16305	**7319**	f_{18}	181580	171251	110077	150959	**80523**
f_7	200000	200000	200000	200000	**131207**	f_{19}	9247	12203	8839	8664	**6073**
f_8	21271	31445	25837	22992	**10324**	f_{20}	88929	99271	38443	81323	**27653**
f_9	52841	41807	52705	32889	**18726**	f_{21}	17662	31160	16069	14217	**7961**
f_{10}	190143	157648	122536	26303	**11328**	f_{22}	126814	94450	96819	98001	**8697**
f_{11}	16096	14647	15017	9490	**3394**	f_{23}	28429	18816	5873	9134	**4439**
f_{12}	88007	100336	66012	70527	**55651**	f_{24}	72011	71508	38101	60152	**25071**

Bold values indicate the best value achieved among all algorithms

Table 4 Mean Error (ME) achieved by the algorithms in 100 runs, *TP*: Test Problems

TP	ABC	BSFABC	GABC	MABC	ABCGLN	TP	ABC	BSFABC	GABC	MABC	ABCGLN
f_1	8.17E−06	7.49E−06	8.26E−06	8.95E−06	**7.46E−06**	f_{13}	9.34E+00	4.12E+00	8.51E+00	1.06E+00	**1.02E−01**
f_2	2.28E−04	5.96E−06	6.92E−06	9.24E−06	**3.29E−06**	f_{14}	7.08E−06	**6.99E−06**	7.91E−06	9.27E−06	7.98E−06
f_3	**1.60E+00**	1.97E+00	2.75E+00	3.60E+01	3.85E+00	f_{15}	8.32E−06	2.05E−05	5.64E−06	5.23E−06	**5.02E−06**
f_4	8.28E−06	8.11E−06	9.06E−06	9.51E−06	**8.06E−06**	f_{16}	1.46E−01	3.07E−02	2.44E−02	1.25E−02	**7.37E−03**
f_5	7.22E−06	7.01E−06	7.98E−06	9.23E−06	**6.93E−06**	f_{17}	1.77E−04	1.34E−04	9.92E−05	1.95E−04	**1.28E−05**
f_6	7.46E−06	7.33E−06	8.31E−06	9.11E−06	**7.04E−06**	f_{18}	8.14E+00	2.72E+00	5.59E−01	8.54E−01	**8.45E−02**
f_7	9.89E+01	8.67E+01	1.06E+02	1.47E+00	**9.35E−03**	f_{19}	6.98E−06	7.09E−06	7.16E−06	8.34E−06	**6.70E−06**
f_8	7.63E−06	**7.15E−06**	8.17E−06	9.02E−06	8.88E−06	f_{20}	7.44E−04	4.04E−03	1.54E−04	6.22E−04	**1.53E−04**
f_9	9.05E−06	8.88E−06	9.22E−06	9.49E−06	**8.40E−06**	f_{21}	**7.63E−06**	8.16E−06	8.32E−06	9.06E−06	8.89E−06
f_{10}	9.77E−01	9.60E−01	9.34E−01	9.33E−01	**8.09E−01**	f_{22}	4.76E−07	4.45E−07	4.80E−07	4.67E−07	**1.71E−14**
f_{11}	**5.49E−06**	6.08E−06	5.50E−06	7.57E−06	5.96E−06	f_{23}	1.95E−03	1.96E−03	1.95E−03	1.95E−03	**1.94E−03**
f_{12}	3.25E−02	6.90E−02	7.39E−06	8.34E−06	**7.05E−06**	f_{24}	9.22E−16	7.29E−16	9.36E−16	9.23E−16	**7.18E−16**

Bold values indicate the best value achieved among all algorithms



(a) SR **(b)** AFE **(c)** ME

Fig. 3 Boxplots graph for considered algorithms on test problems $f_1 - f_{24}$

Table 5 Comparison of algorithms based on the Mann-Whitney U rank sum test at a $\alpha = 0.05$ significance level and mean number of function evaluations, TP: Test Problem

TP	Mann-Whitney U rank sum test with ABCGLN				TP	Mann-Whitney U rank sum test with ABCGLN			
	ABC	BSFABC	GABC	MABC		ABC	BSFABC	GABC	MABC
f_1	+	+	+	+	f_{13}	+	+	+	+
f_2	+	+	+	+	f_{14}	+	+	+	+
f_3	−	−	−	+	f_{15}	+	+	+	+
f_4	+	+	+	+	f_{16}	+	+	+	+
f_5	+	+	+	+	f_{17}	+	+	+	+
f_6	+	+	+	+	f_{18}	+	+	+	+
f_7	+	+	+	+	f_{19}	+	+	+	=
f_8	+	+	+	+	f_{20}	+	+	+	+
f_9	+	+	+	+	f_{21}	+	+	+	+
f_{10}	+	+	+	+	f_{22}	+	+	+	+
f_{11}	+	+	+	+	f_{23}	+	+	=	+
f_{12}	+	+	+	+	f_{24}	+	+	+	+

variable varies from 0 to 1 as given in Bansal and Sharma (2012). Following are the resultant cases:

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

The *PI* graphs corresponding to above cases (1), (2) and (3) for all the algorithms are shown in Fig. 4(a),(b) and (c) respectively where horizontal axis represents the weights k_1, k_2 and k_3 and vertical axis represents the *PI*. In case (1), equal weights are given to AFE and ME while weight to SR varies from 0 to 1. In case (2), SR and ME are assigned equal weights and weight to AFE varies in range 0 to 1, while in case (3), equal weights are assigned to SR and AFE and weight to ME varies in range $(0, 1)$. It can be observed from Fig. 4 that *PI* of ABCGLN is much higher than all the considered algorithms for all three cases.

Further, to compare convergence speed of ABCGLN with respect to other algorithms, we calculated the acceleration rate (AR) based on the AFEs, which is defined below in Eq. (6). A smaller AFEs means higher convergence speed. To minimize the effect of algorithm's stochastic nature, we averaged the reported function evaluations over 100 runs for each test problem.

$$AR = \frac{AFE_{ALGO}}{AFE_{ABCGLN}}, \tag{6}$$

where, ALGO is any of the algorithms ABC, BSFABC, GABC and MABC. Table 6 shows a comparison between ABCGLN and other considered recent modified ABC algorithms in terms of *AR*. Here, $AR > 1$ means ABCGLN is faster i.e, in less number of function evaluations it achieved the required optimum with acceptable error. It is clear from the Table 6 that convergence speed of ABCGLN is much better than the other considered algorithms for all the test problems except functions f_3 as it took on an average half number of function evaluations as compare to other algorithms.

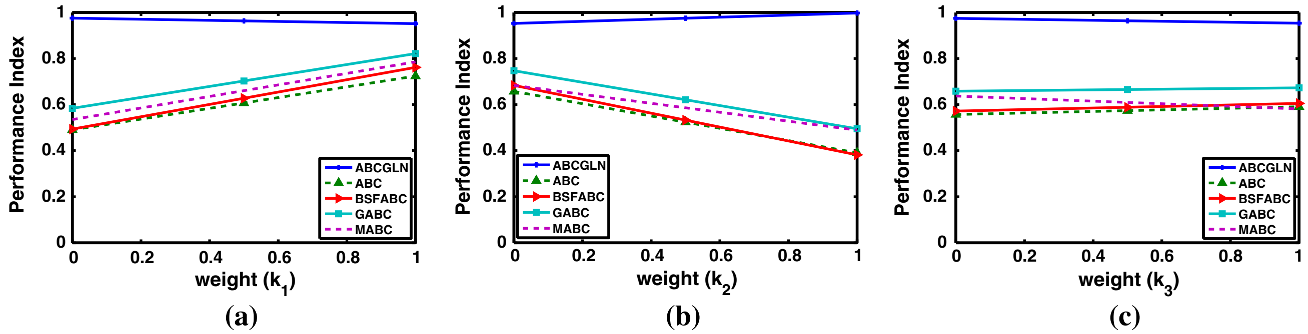


Fig. 4 Performance index for algorithms on test problems; (a) for case (1), (b) for case (2) and (c) for case (3)

Table 6 Acceleration Rate (AR) of ABCGLN as compared to other considered algorithms , TP: Test Problems

TP	ABCGLN vs				TP	ABCGLN vs			
	ABC	BSFABC	GABC	MABC		ABC	BSFABC	GABC	MABC
f_1	2.152	3.114	2.600	2.398	f_{13}	1.403	1.370	1.313	1.200
f_2	1.900	1.781	1.400	1.769	f_{14}	2.883	2.194	2.667	2.090
f_3	0.980	0.966	0.941	1.015	f_{15}	4.445	4.398	2.207	1.704
f_4	2.721	2.376	2.652	2.418	f_{16}	2.568	2.053	2.062	1.800
f_5	2.310	3.191	2.868	2.285	f_{17}	3.028	2.400	1.674	2.879
f_6	2.300	2.294	3.085	2.228	f_{18}	2.255	2.127	1.367	1.875
f_7	1.524	1.524	1.524	1.524	f_{19}	1.523	2.010	1.456	1.427
f_8	2.060	3.046	2.503	2.227	f_{20}	3.216	3.590	1.390	2.941
f_9	2.822	2.233	2.815	1.756	f_{21}	2.219	3.914	2.018	1.786
f_{10}	16.785	13.917	10.817	2.322	f_{22}	14.581	10.860	11.132	11.268
f_{11}	4.742	4.316	4.425	2.796	f_{23}	6.404	4.239	1.323	2.058
f_{12}	1.581	1.803	1.186	1.267	f_{24}	2.872	2.852	1.520	2.399

5 Conclusion

In this article, local and global neighborhood based position update strategy is incorporated in employed bee phase of ABC. In the proposed strategy, each employed bee gets updated using best solutions in its local and global neighborhoods as well as random members from these neighborhoods. Here, local neighborhood of any solution means the sub population around that bee based on index not based on search space or objective space while global neighborhood is the entire swarm for each solution. Therefore the exploration (due to presence of local neighborhoods) and exploitation (due to presence of global neighborhood) can be well balanced. The performance of the proposed algorithm ABCGLN has been extensively compared with other recent variants of ABC namely, GABC, BSFABC and MABC. Through the extensive and statistical experiments, it can be stated that the proposed strategy uplifted the ABC algorithm's performance in terms of robustness, accuracy and efficiency and hence ABCGLN can be considered as a promising alternative algorithm to solve a variety of optimization problems. In future, authors will apply proposed algorithm to solve some real world optimization problems.

References

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

Akay B, Karaboga D (2012) Artificial bee colony algorithm for large-scale problems and engineering design optimization. J Intell Manuf 23(4):1001

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

Banharnsakun A, Achalakul T, Sirinaovakul B (2011) The best-so-far selection in artificial bee colony algorithm. Appl Soft Comput 11(2):2888–2901

Banharnsakun A, Sirinaovakul B, Achalakul T (2012) Job shop scheduling with the best-so-far abc. Eng Appl Artif Intell 25(3):583–593

Bansal Jagdish Chand, Sharma Harish, Arya KV, Nagar Atulya (2013) Memetic search in artificial bee colony algorithm. Soft Comput 17(10):1911–1928

Bansal Jagdish Chand, Sharma Harish, Jadon Shimpi Singh (2013) Artificial bee colony algorithm: a survey. Int J Adv Intell Paradig 5(1):123–159

Bansal JC, Sharma H, Jadon SS, Clerc M (2013) Spider monkey optimization algorithm for numerical optimization. Memet Comput 1–17

Bansal Jagdish Chand, Sharma Harish, Nagar Atulya, Arya KV (2013) Balanced artificial bee colony algorithm. Int J Artif IntellSoft Comput 3(3):222–243

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 1–21

Baykasoglu A, Ozbakir L, Tapkan P (2007) Artificial bee colony algorithm and its application to generalized assignment problem. Swarm Intell 113–144

Chidambaram C, Lopes HS (2009) A new approach for template matching in digital images using an artificial bee colony algorithm. In Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on, pages 146–151. IEEE

Akay B, Karaboga D, Ozturk C (2008) Training neural networks with abc optimization algorithm on medical pattern classification. In: International conference on multivariate statistical modelling and high dimensional data mining (Kayseri, TURKEY), June 19–23

Das S, Abraham A, Chakraborty UK, Konar A (2009) Differential evolution using a neighborhood-based mutation operator. Evolut Comput IEEE Trans 13(3):526–553

Das Swagatam, Abraham Ajith, Chakraborty Uday K, Konar Amit (2009) Differential evolution using a neighborhood-based mutation operator. Evolut Comput IEEE Trans 13(3):526–553

Diwold K, Aderhold A, Scheidler A, Middendorf M (2011) Performance evaluation of artificial bee colony optimization and new selection schemes. Memet Comput 1–14

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, volume 2. IEEE

El-Abd M (2011) Performance assessment of foraging algorithms vs. evolutionary algorithms. Inf Sci 182(1):243–263

Haijun D, Qingxian F (2008) Bee colony algorithm for the function optimization. Science paper online, Aug 2008

Gao W, Liu S (2011) A modified artificial bee colony algorithm. Comput Oper Res 39(3):687–697

Jadon S, Bansal J C, Tiwari R, Sharma H (2014) Expedited artificial bee colony algorithm. In: Proceedings of the 3rd international conference on soft computing for problem solving, 787–800. Springer 2014

Jones KO, Bouffet A (2008) Comparison of bees algorithm, ant colony optimisation and particle swarm optimisation for pid controller tuning. In Proceedings of the 9th international conference on computer systems and technologies and workshop for PhD students in computing, pages IIIA-9. ACM

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

Karaboga D (2005) An idea based on honey bee swarm for numerical optimization. Technology Report TR06, Erciyes Univercity Press, Erciyes

Karaboga D, Akay B (2009) A comparative study of artificial bee colony algorithm. Appl Math Comput 214(1):108–132

Karaboga Dervis, Akay Bahriye (2011) A modified artificial bee colony (abc) algorithm for constrained optimization problems. Appl Soft Comput 11(3):3021–3031

Karaboga N, Cetinkaya MB (2011) A novel and efficient algorithm for adaptive filtering: artificial bee colony algorithm. Turk J Electr Eng Comput Sci 19:175–190

Kavian YS, Rashedi A, Mahani A, Ghassemlooy Z (2012) Routing and wavelength assignment in optical networks using artificial bee colony algorithm. Optik-Int J Light Electr Opt

Kennedy J, Eberhart R (1995) Particle swarm optimization. In neural networks, 1995. In: Proceedings IEEE international conference on, EEE, vol 4, p 1942–1948

Xing F, Fenglei L, Haijun D (2007) The parameter improvement of bee colony algorithm in tsp problem. Science paper online, Nov 2007

Lam SSB, Raju ML, Ch S, Srivastav PR et al (2012) Automated generation of independent paths and test suite optimization using artificial bee colony. Procedia Eng 30:191–200

Lei X, Huang X, Zhang A (2010) Improved artificial bee colony algorithm and its application in data clustering. In Bio-Inspired computing: theories and applications (BIC-TA), 2010 IEEE 5th international conference on, EEE, pp 514–521

Li HJ, Li JJ, Kang F (2011) Artificial bee colony algorithm for reliability analysis of engineering structures. Adv Mater Res 163:3103–3109

Mandal SK, Chan FTS, Tiwari MK (2012) Leak detection of pipeline: an integrated approach of rough set theory and artificial bee colony trained svm. Expert Syst Appl 39(3):3071–3080

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

Nayak SK, Krishnanand KR, Panigrahi BK, Rout PK (2009) Application of artificial bee colony to economic load dispatch problem with ramp rate limits and prohibited operating zones. In Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on, pages 1237–1242. IEEE

Passino KM (2002) Biomimicry of bacterial foraging for distributed optimization and control. Control Syst Mag IEEE 22(3):52–67

Pawar P, Rao R, Shankar R (2008) Multi-objective optimization of electro-chemical machining process parameters using artificial bee colony (abc) algorithm. Advances in mechanical engineering (AME-2008), Surat, India

Price KV, Storn RM, Lampinen JA (2005) Differential evolution: a practical approach to global optimization. Springer, New York

Rahnamayan S, Tizhoosh HR, Salama MMA (2008) Opposition-based differential evolution. Evolut Comput IEEE Trans 12(1):64–79

Sharma Harish, Bansal Jagdish Chand, Arya KV (2013) Opposition based lévy flight artificial bee colony. Memet Comput 5(3):213–227

Singh A (2009) An artificial bee colony algorithm for the leaf-constrained minimum spanning tree problem. Appl Soft Comput 9(2):625–631

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

Sulaiman MH, Mustafa MW, Shareef H, Abd Khalid SN (2012) An application of artificial bee colony algorithm with least squares support vector machine for real and reactive power tracing in deregulated power system. Int J Electr Power Energy Syst 37(1):67–77

Tsai PW, Pan JS, Liao BY, Chu SC (2009) Enhanced artificial bee colony optimization. Int J Innov Comput Inf Control 5(12):5081–5092

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

Williamson DF, Parker RA, Kendrick JS (1989) The box plot: a simple visual method to interpret data. Ann Int Med 110(11):916

Xu C, Duan H (2010) Artificial bee colony (abc) optimized edge potential function (epf) approach to target recognition for low-altitude aircraft. Pattern Recognit Lett 31(13):1759–1772

Yeh WC, Hsieh TJ (2011) Solving reliability redundancy allocation problems using an artificial bee colony algorithm. Comput Oper Res 38(11):1465–1473

Zhu G, Kwong S (2010) Gbest-guided artificial bee colony algorithm for numerical function optimization. Appl MatH Comput 217(7):3166–3173