Expedited Artificial Bee Colony Algorithm

Shimpi Singh Jadon, Jagdish Chand Bansal, Ritu Tiwari and Harish Sharma

Abstract Artificial Bee Colony (ABC) is one of the latest and emerging swarm intelligence algorithms. Though, there are some areas where ABC works better than other optimization techniques but, the drawbacks like stucking at local optima and preferring exploration at the cost of exploitation, are also associated with it. This paper uses position update equation in ABC as in Gbest-guided ABC (GABC) and attempts to improve ABC algorithm by balancing its exploration and exploitation capabilities. The proposed algorithm is named as Expedited Artificial Bee Colony (EABC). We altered the onlooker bee phase of ABC by forcing the individual bee to take positive direction towards the random bee if this selected random bee has better fitness than the current bee and if it is not the case then the current bee will move in reverse direction. In this way, ABC colony members will not follow only global best bee but also a random bee which has better fitness than the current bee which is going to be modified. So the mentioned drawbacks of the ABC may be resolved. To analyze the performance of the proposed modification, 14 unbiased benchmark optimization functions have been considered and experimental results reflect its superiority over the Basic ABC and GABC.

Keywords Artificial bee colony · Swarm intelligence · Optimization · Gbest artificial bee colony

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
Government Engineering College, Jhalawar, India
e-mail: harish.sharma0107@gmail.com

M. Pant et al. (eds.), *Proceedings of the Third International Conference on Soft Computing for Problem Solving*, Advances in Intelligent Systems and Computing 259,
DOI: 10.1007/978-81-322-1768-8_68,

1 Introduction

Swarm Intelligence is an emerging category in nature inspired algorithms group in current decade. Collaborative trail and error method is the main engine behind the Swarm Intelligence which enables the algorithmic procedure to find the solution. Researchers analyzed such collaboration between the social insects while searching food for them and created the structures known as Swarm Intelligence based algorithms like ant colony optimization (ACO) [6], particle swarm optimization (PSO) [14], bacterial foraging optimization (BFO) [15]. The work presented in the articles [6, 14, 16, 17] under the mentioned category proved their efficiency and potential to deal with non linear, non convex and discrete optimization problems etc. Karaboga [10] 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 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 selection process which are responsible for exploration and exploitation respectively. However the ABC achieves a good solution at a significantly faster rate but, like the other optimization algorithms, it is also weak in refining the already explored search space, mainly due to less diversity in later search. 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 [11] and laking to balance between exploration and exploitation [19]. Therefore these drawbacks require a enhanced and more efficient ABC to deal with. Here we proposes a modification in ABC called as Expedited ABC (EABC) to overcome the mentioned drawbacks. Researchers are also working to enhance the capabilities of ABC. To enhance the exploitation, Gao et al. [8] improved position update equation of ABC such that the bee searches only in neighborhood of the previous iteration's best solution. Banharnsakun et al. [2] 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 [12] used Deb's rules consisting of three simple heuristic rules and a probabilistic selection scheme in ABC algorithm. Karaboga [10] 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. [9] introduced exploitation phase in ABC using Rosenbrock's rotational direction method and named modified ABC as Rosenbrock ABC (RABC).

Rest of the paper is organized as follows: ABC is explained in Sect. 2. In Sect. 3, proposed modified ABC (EABC) is explained. In Sect. 4, performance of the proposed strategy is analyzed. Finally, in Sect. 5, paper is concluded.

2 Artificial Bee Colony (ABC) Algorithm

Artificial Bee Colony is made of three groups 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 food sources with better nectar. If any food source is abandoned then scout bee finds new food source randomly in search space. ABC is a population-based iterative search procedure. 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 direction 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 Bee 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 $k \neq i \cdot \ \phi_{ij}$ is a random number between $[-1, 1]$. 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 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 its fitness it selects a solution with a probability $prob_i$. Here $prob_i$ can be calculated as function of fitness (there may be some other):

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

here $fitness_i$ is the fitness value of the ith solution and *maxfit* is the maximum fitness amongst all the solutions. 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 same as in initialization phase 1.1.

2.5 Main Steps of the ABC Algorithm

The pseudo-code of the ABC is shown in Algorithm 1 [11].

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.

In 2010, Zhu and Kwong [19] proposed Gbest-guided ABC (GABC) algorithm, an improvement in ABC algorithm by introducing global best (gbest) solution of the current population into the position update equation to improve the exploitation. This proposed version GABC is inspired by PSO [14], which uses the global best (gbest) solution to guide the search. The modified the position update equation of ABC as follows:

$$v_{ij} = x_{ij} + \phi_{ij}(x_{ij} - x_{kj}) + \psi_{ij}(y_j - x_{ij}) \tag{4}$$

The only difference in the modified Eq. (4) is the third term on the right-hand side, which is called gbest term. Here y_j is the jth dimension of the global best solution, ψ_{ij} is a uniform random number in [0, C], where C is a non negative constant. According to Eq. (4), the gbest term tries to drive the search towards the global best solution, therefore, responsible for the exploitation. Note that setting the parameter C in (4) is a crucial task as to balance the exploration and exploitation of the solution search. In this paper ABC uses Eq. (4) of GABC as its position update equation and which is modified further to enhance its search capability.

3 Expedited Artificial Bee Colony

Karaboga and Akay [11] shows inefficient balance between exploration and exploitation of the search space in ABC while experimenting its different variants. Recently Zhu and kwong [19] proposed a modified version of ABC namely Gbest-guided Artificial Bee Colony algorithm (GABC) which incorporated information of the bee having highest fitness in the swarm within the position update equation. As a result of which, ABC may suffer from premature convergence as in early iterations the whole population gets influence to best solution (which is not necessarily a optimum) in the current swarm. At the same time, we can not lose this best solution information also, which tries to guide the population towards the better one. So in order to make balance between losing and adopting the best solution information, we modify the difference term in position update equation of ABC, which adopts the directional difference between the current bee and a random bee in the swarm, through taking positive scaled value of this term in that bee's direction which is better between current and random bee. In this way, we adopt information of both the swarm best and best between current and a random bee and modified the position update equation inspired from PSO [14] which does not give more weightage to global best, but a equal weightage to personal best also. After analyzing these concerns, we propose following modification in **onlooker bee phase** of ABC. The proposed modification can be mathematically seen as follows:

if ($prob_{current(i)} > prob_{random(k)}$)

$$v_{ij} = x_{ij} + rand[0, 0.5](x_{ij} - x_{kj}) + \psi_{ij}(y_j - x_{ij}) \tag{5}$$

else

$$v_{ij} = x_{ij} + rand[0, 0.5](x_{kj} - x_{ij}) + \psi_{ij}(y_j - x_{ij}) \tag{6}$$

here $prob_{current(i)}$ and $prob_{random(k)}$ are the probabilities calculated in Eq. (3) of the current bee ith and random bee kth respectively. $rand[0, 0.5]$ is a random number between 0 and 0.5. Basically in ABC, we follow positive direction towards best bee in current swarm and random direction towards randomly selected kth bee, but in proposed EABC we follow positive direction towards gbest and also towards random bee if it has better fitness than the current bee otherwise we follow negative direction towards random bee. Here, in both Eqs. (5) and (6) we take random number between [0, 0.5] instead of [0, 1] so that this term may not takeover the gbest influence over the search. Therefore instead of completely following gbest bee in colony, we add some better random direction to current bee also to ensure convergence not prematurely.

4 Experiments and Results

4.1 Test Problems Under Consideration

To validate the performance, we compared the proposed strategy with basic ABC and GABC over 14 unbiased benchmark optimization functions which are given in Table 1. For a successful run, the minimum error criteria is fixed as in last colon of Table 1 i.e. an algorithm is considered successful if it finds the error less than acceptable error in a specified maximum function evaluations.

4.2 Experimental Setting

To prove the efficiency of proposed *EABC*, it is compared with *ABC* and *GABC*. To test *EABC*, *ABC* and *GABC* over considered problems, following experimental setting is adopted:

Parameter setting for ABC:

- Colony size $NP = 50$ [5, 7],
- $\phi_{ij} = rand[-1, 1]$ and $\psi_{ij} = \text{rand}[0, 1.5]$ [19],
- Number of food sources $SN = NP/2$,
- $limit = D \times SN$ [1, 13],
- The number of simulations/run = 100.

Table 1 Test problems

Test problem	Objective function	SS	OV	D	AE
Beale function	$f_1(x) = [1.5 - x_1(1 - x_2)]^2 + [2.25 - x_1(1 - x_2^2)]^2 + [2.625 - x_1(1 - x_2^3)]^2$	$[-4.5, 4.5]$	$f(3, 0.5) = 0$	2	1.0*E*−05
Colville function	$f_2(x) = 100[x_2 - x_1^2]^2 + (1 - x_1)^2 + 90(x_4 - x_3^2)^2 + (1 - x_3)^2 + 10.1[(x_2 - 1)^2 + (x_4 - 1)^2] + 19.8(x_2 - 1)(x_4 - 1)$	$[-10, 10]$	$f(\mathbf{1}) = 0$	4	1.0*E*−05
Branins's function	$f_3(x) = a(x_2 - bx_1^2 + cx_1 - d)^2 + e(1 - f)\cos x_1 + e$	$-5 \le x_1 \le 10, 0 \le x\ 2 \le 15$	$f(-\pi, 12.275) = 0.3979$	2	1.0*E* −05
Kowalik function	$f_4(x) = \sum_{i=1}^{11}\left[a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4}\right]^2$	$[-5, 5]$	$f(0.1928, 0.1908, 0.1231, 0.1357) = 3.07E{-}04$	4	1.0*E*−05
Shifted Rosenbrock	$f_5(x) = \sum_{i=1}^{D-1}(100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2) + f_{bias},\ z = x - o + 1,\ x = [x_1, x_2, \ldots, x_D],\ o = [o_1, o_2, \ldots, o_D]$	$[-100, 100]$	$f(o) = f_{bias} = 390$	10	1.0*E* −01
Shifted sphere	$f_6(x) = \sum_{i=1}^{D} z_i^2 + f_{bias},\ z = x - o,\ x = [x_1, x_2, \ldots, x_D],\ o = [o_1, o_2, \ldots, o_D]$	$[-100, 100]$	$f(o) = f_{bias} = -450$	10	1.0*E*−05
Shifted rastrigin	$f_7(x) = \sum_{i=1}^{D}(z_i^2 - 10\cos(2\pi z_i) + 10) + f_{bias}\mathrm{z} = (\mathrm{x} - \mathrm{o}),\ \mathrm{x} = (x_1, x_2, \ldots, x_D),\ \mathrm{o} = (o_1, o_2, \ldots, o_D)$	$[-5, 5]$	$f(o) = f_{bias} = -330$	10	1.0*E*−02
Shifted schwefel	$f_8(x) = \sum_{i=1}^{D}\left(\sum_{j=1}^{i} z_j\right)^2 + f_{bias},\ z = x - o,\ x = [x_1, x_2, \ldots, x_D],\ o = [o_1, o_2, \ldots, o_D]$	$[-100, 100]$	$f(o) = f_{bias} = -450$	10	1.0*E*−05
Shifted griewank	$f_9(x) = \sum_{i=1}^{D}\frac{z_i^2}{4000} - \prod_{i=1}^{D}\cos\left(\frac{z_i}{\sqrt{i}}\right) + 1 + f_{bias},\ z = (x - o),\ x = [x_1, x_2, \ldots, x_D],\ o = [o_1, o_2, \ldots, o_D]$	$[-600, 600]$	$f(o) = f_{bias} = -180$	10	1.0*E*−05

(continued)

Table 1 (continued)

Test problem	Objective function	SS	OV	D	AE
Ackley	$f_10(x) = -20 + e + exp\left(-\frac{0.2}{D}\sqrt{\sum_{i=1}^{D} x_i^3}\right)$	$[-1, 1]$	$f(\mathbf{0}) = 0$	30	$1.0E{-}05$
Goldstein-price	$f_{11}(x) = (1 + (x_1 + x_2 + 1)^2 \cdot (19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)) \cdot (30 + (2x_1 - 3x_2)^2 \cdot (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2))$	$[-2, 2]$	$f(0, -1) = 3$	2	$1.0E{-}14$
Easom's function	$f_{12}(x) = -\cos x_1 \cos x_2 e^{((-(x_1-\Pi)^2-(x_2-\Pi)^2))}$	$[-10, 10]$	$f(\pi, \pi) = -1$	2	$1.0E{-}13$
Meyer and Roth	$f_{13}(x) = \sum_{i=1}^{5}\left(\frac{x_1x_3t_i}{1+x_1t_i+x_2v_i} - y_i\right)^2$	$[-10, 10]$	$f(3.13, 15.16, 0.78) = 0.4\ E{-}04$	3	$1.0E{-}03$
Shubert	$f_{14}(x) = -\sum_{i=1}^{5} i\cos((i+1)x_1 + 1) \sum_{i=1}^{5} i\cos((i+1)x_2 + 1)$	$[-10, 10]$	$f(7.0835, 4.8580) = -186.7309$	2	$1.0E{-}05$

SS Search space; *OV* Optimum value; *D* Dimension; *AE* Acceptable error

4.3 Results Comparison

With experimental setting of Sect. 4.2, Table 2 reports the numerical results in terms of standard deviation (*SD*), mean error (*ME*), average function evaluations (*AFE*), and success rate (*SR*). To minimize the effect of the algorithmic stochastic nature, the reported function evaluations is averaged over 100 runs. Results in Table 2 reflects that most of the time *EABC* outperforms in terms of reliability, efficiency and accuracy as compare to the *ABC* and *GABC*. For more intensive analyses of the results, the performance indices and boxplots have been carried out.

EABC, *ABC* and *GABC* are compared through *SR*, *AFE* and *ME* in Table 2. Here signs +/− indicate that for that particular function, *EABC* is better/worst than the considered algorithms, respectively. If *EABC* has more success rate *SR* than the other algorithm then it is said to be better and if *EABC* has equal success rate to other then comparison is done in order of preference of average function evaluation (*AFE*), mean error (*ME*). Total Outcome of comparison is mentioned in Table 3. The bottom line of Table 3 represents that of *EABC* is better than *ABC* in all 14 functions and better than *GABC* in 11 functions out of 14 functions and hence shows the superiority of *EABC*.

We also did boxplot analyses [18] to compare all the considered algorithms in terms of consolidated performance as it can efficiently represent the empirical distribution of data graphically. The boxplots for *EABC*, *ABC* and *GABC* are shown in Fig. 1. It is clear from this figure that *EABC* is better than the considered algorithms as interquartile range and median are comparatively low.

Table 3 shows the performance of *EABC* in the order of preference of *SR*, *AFE* and *ME*. Now giving weighted importance to these parameters, performance indices (*PI*) are calculated [4]. The values of *PI* for the *EABC*, *ABC* and *GABC* are calculated by using following equations:

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

where

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

- Sr^i = Successful simulations/runs of ith problem.
- Tr^i = Total simulations of ith problem.
- Mf^i = Minimum of average number of function evaluations used to obtain the required solution of ith problem.

Table 2 Comparison of the results of test problems

Test Function	Algorithm	SD	ME	AFEs	SR
f_1	ABC	1.66E−06	8.64E−06	16520.09	100
	GABC	3.05E−06	5.03E−06	9314.71	100
	EABC	2.77E−06	5.86E−06	5623.48	100
f_2	ABC	1.03E−01	1.67E−01	199254.48	1
	GABC	1.71E−02	1.95E−02	151300.35	46
	EABC	4.32E−03	8.79E−03	98278.88	85
f_3	ABC	6.83E−06	6.05E−06	1925.52	100
	GABC	6.54E−06	5.76E−06	1204.65	100
	EABC	6.35E−06	6.18E−06	816.13	100
f_4	ABC	7.33E−05	1.76E−04	180578.91	18
	GABC	2.15E−05	8.68E−05	90834.53	97
	EABC	1.49E−05	8.79E−05	63816.73	100
f_5	ABC	1.25E+00	8.32E−01	179015.68	18
	GABC	2.97E−02	8.76E−02	101517.9	95
	EABC	2.93E−02	8.84E−02	107569.72	92
f_6	ABC	2.62E−06	6.75E−06	9163.5	100
	GABC	2.45E−06	6.95E−06	5626	100
	EABC	2.09E−06	7.30E−06	5285	100
f_7	ABC	1.12E+01	8.76E+01	200000	0
	GABC	9.77E+00	8.63E+01	200000	0
	EABC	9.95E+00	8.76E+01	200000	0
f_8	ABC	3.40E+03	1.21E+04	200000	0
	GABC	3.31E+03	1.05E+04	200000	0
	EABC	3.00E+03	1.03E+04	200000	0
f_9	ABC	3.10E−03	1.41E−03	80078.81	82
	GABC	2.85E−06	5.25E−06	39423.63	100
	EABC	1.61E−03	3.75E−04	50717.76	95
f_{10}	ABC	1.75E−06	7.73E−06	16790.5	100
	GABC	1.64E−06	8.04E−06	9349.5	100
	EABC	1.32E−06	8.44E−06	8784	100
f_{11}	ABC	2.18E−06	4.97E−07	106142.69	65
	GABC	4.38E−15	4.80E−15	3965.99	100
	EABC	4.39E−15	4.67E−15	3093.09	100
f_{12}	ABC	6.19E−05	2.15E−05	195783.29	4
	GABC	2.98E−14	4.39E−14	48432.57	100
	EABC	2.80E−14	4.73E−14	25909.31	100
f_{13}	ABC	2.94E−06	1.95E−03	24185.13	100
	GABC	3.03E−06	1.95E−03	4770.87	100
	EABC	2.81E−06	1.95E−03	4311.85	100
f_{14}	ABC	5.47E−06	4.66E−06	4883.53	100
	GABC	5.88E−06	5.09E−06	2450.53	100
	EABC	5.46E−06	4.59E−06	2041.98	100

Table 3 Outcome of Table 2

TP	EABC Vs GABC	EABC Vs ABC
f_1	+	+
f_2	+	+
f_3	+	+
f_4	+	+
f_5	–	+
f_6	+	+
f_7	–	+
f_8	+	+
f_9	–	+
f_{10}	+	+
f_{11}	+	+
f_{12}	+	+
f_{13}	+	+
f_{14}	+	+
Total number of + sign	11	14

TP Test problems

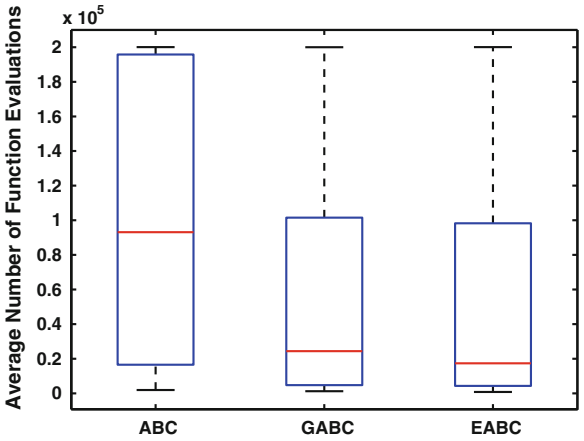

Fig. 1 Boxplots graphs for average function evaluation

- Af^i = Average number of function evaluations used to obtain the required solution of ith problem.
- Mo^i = Minimum of standard deviation got for the ith problem.
- Ao^i = Standard deviation obtained by an algorithm for the ith problem.
- N_p = Total number of optimization problems evaluated.
- k_1, k_2, k_3 = weights assigned to the success rate, the average number of function evaluations and the standard deviation respectively such that $k_1 + k_2 + k_3 = 1$ and $0 \leq k_1, k_2, k_3 \leq 1$.

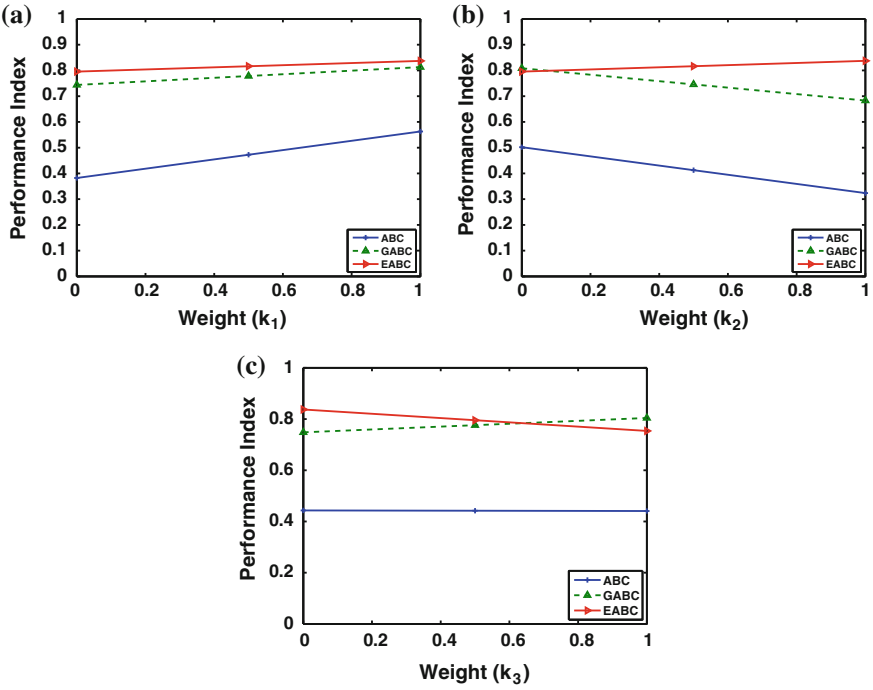

Fig. 2 Performance index for test problems; **a** for case (1), **b** for case (2) and **c** for case (3)

To calculate the *PI*s, equal weights are assigned to two variables while weight of the remaining variable vary from 0 to 1 as given in [3]. 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 *EABC*, *ABC* and *GABC* are shown in Fig. 2a–c respectively. In these figures, horizontal axis represents the weights k_1, k_2 and k_3 and while vertical axis represents the *PI*.

For case (1), *PIs* of the considered algorithms are superimposed in Fig. 2a by giving equal weights to average number of function evaluations and the standard deviation. It is observed that *PI* of *EABC* are higher than the considered algorithms. Similarly for case (2), equal weights are assigned to the success rate and standard deviation and for case (3), equal weights are assigned to the success rate and average function evaluations. It is clear from Fig. 2b, c that in these cases also, the *EABC* algorithm has better performance than the others.

5 Conclusion

In this paper, ABC algorithm is modified through alteration in position update equation in onlooker bee phase to balance the exploration and exploitation capabilities of ABC. Instead focusing only on the global best bee, we also emphasize on a random bee with better fitness in the colony and the so obtained modified ABC is named as Expedited ABC (*EABC*). To analyze the proposed algorithm, it is compared to *ABC* and a recent variant *GABC* and with the help of experiments over 14 unbiased benchmark optimization functions, it can be concluded that the *EABC* outperforms to the considered algorithms in terms of reliability, efficiency and accuracy.

References

1. Akay, B., Karaboga, D.: A modified artificial bee colony algorithm for real-parameter optimization. Inf. Sci. (2010). doi:10.1016/j.ins.2010.07.015
2. Banharnsakun, A., Achalakul, T., Sirinaovakul, B.: The best-so-far selection in artificial bee colony algorithm. Appl. Soft Comput. **11**(2), 2888–2901 (2011)
3. Bansal, J.C., Sharma, H., Arya, K.V., Nagar, A.: Memetic search in artificial bee colony algorithm. Soft Comput. **17**(10), 1–18 (2013)
4. Bansal, J.C., Sharma, H.: Cognitive learning in differential evolution and its application to model order reduction problem for single-input single-output systems. Memetic Comput. **4**, 1–21 (2012)
5. Diwold, K., Aderhold, A., Scheidler, A., Middendorf, M.: Performance evaluation of artificial bee colony optimization and new selection schemes. Memetic Comput. **3**, 1–14 (2011)
6. Dorigo, M., Di Caro, G.: Ant colony optimization: a new meta-heuristic. In: Proceedings of the 1999 Congress on Evolutionary Computation (CEC 99), vol. 2. IEEE (1999)
7. El-Abd, M.: Performance assessment of foraging algorithms vs. evolutionary algorithms. Inf. Sci. **182**(1), 243–263 (2011)
8. Gao, W., Liu, S.: A modified artificial bee colony algorithm. Comput. Oper. Res. **39**(3), 687–697 (2011)
9. Kang, F., Li, J., Ma, Z.: Rosenbrock artificial bee colony algorithm for accurate global optimization of numerical functions. Inf. Sci. **181**(16), 3508–3531 (2011)
10. Karaboga, D.: An idea based on honey bee swarm for numerical optimization. Technical report TR06, Erciyes University Press, Erciyes (2005)
11. Karaboga, D., Akay, B.: A comparative study of artificial bee colony algorithm. Appl. Math. Comput. **214**(1), 108–132 (2009)
12. Karaboga, D., Akay, B.: A modified artificial bee colony (abc) algorithm for constrained optimization problems. Appl. Soft Comput. **11**(3), 3021–3031 (2011)
13. Karaboga, D., Basturk, B.: Artificial bee colony (ABC) optimization algorithm for solving constrained optimization problems. Foundations of Fuzzy Logic and Soft Computing, pp. 789–798. Springer, Berlin (2007)
14. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of the IEEE International Conference on Neural Networks, vol. 4, pp. 1942–1948. IEEE (1995)
15. Passino, K.M.: Biomimicry of bacterial foraging for distributed optimization and control. Control Syst. Mag. IEEE **22**(3), 52–67 (2002)
16. Price, K.V., Storn, R.M., Lampinen, J.A.: Differential evolution: a practical approach to global optimization. Springer, Berlin (2005)

17. Vesterstrom, J., Thomsen, R.: A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems. In: Congress on Evolutionary Computation (CEC2004), vol. 2, pp. 1980–1987. IEEE (2004)
18. Williamson, D.F., Parker, R.A., Kendrick, J.S.: The box plot: a simple visual method to interpret data. Ann. Intern. Med. **110**(11), 916 (1989)
19. Zhu, G., Kwong, S.: Gbest-guided artificial bee colony algorithm for numerical function optimization. Appl. Math. Comput. **217**(7), 3166–3173 (2010)