International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2013): 4.438

Explorative Artificial Bee Colony Algorithm: A Novel Swarm Intelligence Based Algorithm for Continuous Function Optimization

Shifat Sharmin Shapla[1], H. M. Zabir Haque[2], Mohammad Shafiul Alam[3]

[1, 2]Stamford University Bangladesh, 51, Siddeswari Road, Dhaka-1217, Bangladesh

[3]Ahsanullah University of Science and Technology, Dhaka-1208, Bangladesh

Abstract: ***The Artificial Bee Colony (ABC) algorithm is a recently introduced swarm intelligence based algorithm that has been successfully employed to numerous scientific and engineering problems. However, ABC sometimes suffers from premature convergence and fitness stagnation, which usually originates from the lack of explorative search capability of its perturbation operator. This paper introduces Explorative ABC (EABC) — a novel variant of the basic ABC algorithm that modifies its exploitative perturbation operation in a more explorative way. EABC not only introduces more randomness during the perturbation operation of ABC, but also customizes the degree of exploitations and explorations at the individual solution level, separately for each candidate solution of the bee population. Besides, EABC introduces a crossover operation and a number of additional alternations of the basic ABC algorithm to assist its explorative perturbation operation. EABC is evaluated with several benchmark problems on numerical function optimization and the results are compared with the basic ABC algorithm. The experimental results demonstrate that EABC often performs better optimization than the basic ABC algorithm on most of the benchmark problems, which indicates the effectiveness of its explorative perturbation operation.***

Keywords: Artificial bee colony algorithm, perturbation, exploration and exploitation, continuous function optimization.

1. Introduction

The Artificial Bee Colony (ABC) algorithm [1] is a recently introduced swarm intelligence based algorithm inspired by the intelligent food foraging behavior of the honey bees found in nature. Since its advent [2], ABC and its variants are often successfully employed to wide and diverse range of problems, such as numeric optimization [3], discrete optimization [4], multi-objective optimization [5], industrial process control [6], structural design [7], design of digital IIR filters [8], PID controller [9], machine learning [10] and so on [11]. In comparison to other greedy and local search based algorithms, ABC is more resilient against premature convergence and fitness stagnation, because the population of candidate solutions can maintain some amount of diversity that is necessary to continue search space explorations avoiding the locally optimal points. However, it is still possible (e.g., [12] – [14]) that the evolving population of candidate solutions loses its diversity and explorative search capability too soon. This leads the candidate solutions to prematurely get trapped around the local optima of the search space. Aside from premature convergence, another problem that is faced by the ABC algorithm is fitness stagnation, where all the candidate solutions fail to improve their fitness values for indefinitely prolonged iterations, for no apparent reason and even without any premature convergence around the locally optimal points. The risk of premature convergence and fitness stagnation usually rises with reduced explorations and increased exploitations. But, increasing the explorations may lead to unacceptably slow convergence speed. So an adaptive and balanced mix of explorations and exploitations is often necessary for good results and sufficient convergence speed of the algorithm, especially for complex, high dimensional, multimodal problems with many locally optimal points.

There exist a number of research works (e.g., [15] – [34]) that attempt to alter the explorative and/or exploitative properties of the basic ABC algorithm. However, most of them focus on altering the selection operation only. In the literature, not much has been reported to improve the basic, non-adaptive and fixed perturbation operator of ABC. The proposed algorithm — Explorative ABC (EABC) alters the perturbation operation of ABC, as well as incorporates few more basic schemes to increase the degree of explorations of the ABC algorithm, as well as to bring a balance between its degree of exploitations and explorations. EABC searches with more randomness across the search space while the original ABC algorithm mostly searches towards and around the best candidate solutions. Besides, the number of parameters that are perturbed by EABC is gradually self-adapted, cycle (i.e., generation) by cycle, separately for each candidate solution, while ABC always uses a fixed, small (hence, exploitative) perturbation rate. The objective of EABC is to introduce more randomness during perturbations for more search space explorations, and to customize the degree of explorations and exploitations at the individual candidate solution level, by adapting the perturbation rate separately for each candidate solution of the bee population.

The rest of this paper is organized as follows. Section 2 describes the basic ABC algorithm in details. Section 3 presents a few improved ABC-variants and explains how EABC is significantly different from them. Section 4 describes EABC in details. Section 5 provides details of the benchmark problems and compares the results of ABC and EABC. Finally, section 6 concludes the paper by leaving a few suggestions for further research with EABC.

2. The Basic Artificial Bee Colony (ABC) Algorithm

Honey bees in a colonyshow remarkable self-organization and co-ordination skills in their food foraging behavior. Bees have to forage over a vast area in search of good sources of food. After an initial exploration stage, more bees are employed to collect honey from the more profitable food sources whereas fewer bees are assigned to the less worthy food sources. Some scout bees are also assigned for exploration to find newer food sources. If the quality of a food source declines after some exploitation, this information is also shared with other bees so that fewer bees are now attracted to this source. After the quality of a food source falls below some threshold, the bees assigned to it abandon it. The foraging process is initiated by scout bees that start searching for flower patches suitable as food sources. Quality is usually measured as a combination of some values, such as quantity and density of sugar, ease of access, distance from the colony etc. After they return to the hive, those scout bees that found a patch with quality above some threshold, deposit their nectar and then go to the 'dance floor' to perform a dance known as the 'waggle dance'. This dance plays the key role to communicate information among the bees about the food sources. The waggle dance contains three pieces of information: i) the quality of the flower patch of this dancing bee, ii) the distance of the flower patch from the hive, iii) the direction from the hive that you have to travel in order to reach the flower. The 'onlooker' bees, waiting around the dance floor, observe the waggle dances of these 'employed' bees that have found good food sources and pick any one of them to become its 'follower' and collect nectar from its flower patch. The better a flower patch as a food source, the bigger is the number of follower bees along with its employed bee. However, if the patch is no longer good enough, it will not be advertised in the next waggle dance and the bees recruited for it as employed or follower bees will choose either to follow some other employed bee or start working as a scout bee to randomly explore the search space for finding new food source.

The ABC algorithm mimics the food foraging behavior of the honey bees with these three groups of bees: employed bees, onlookers and scouts. A bee working to forage a food source (i.e. solution) previously visited by itself and searching only around its vicinity is called an employed bee. Employed bees perform waggle dance to propagate information of its food source to other bees. A bee waiting around the dance floor to choose any of the employed bees to follow is called an onlooker. A bee randomly searching a search space for finding a new food source is called a scout. For every food source, there is only one employed bee and a number of follower bees. The scout bee, after finding a good food source also becomes an employed bee. In the basic ABC algorithm implementation, half of the colony is employed bees and the other half is the onlookers. Number of food sources (i.e., solutions) is equal to the number of employed bees. An employed bee whose food source is exhausted (i.e. solution not improved after several attempts) becomes a scout. The detailed pseudo code is given below.

Step 1) Generate an initial population of N individuals. Each individual is a food source (i.e. solution) and has D attributes, where D is the dimensionality of the problem.

Step 2) Evaluate the fitness of each individual.

Step 3) Each employed beesearches in the neighborhood of its current position to find a better food source. For each employed bee, generate a new solution, $\boldsymbol{v}_i$around its current position, $\boldsymbol{x}_i$ using (1).

$$\boldsymbol{v}_{ij} = \boldsymbol{x}_{ij} + \varphi_{ij} (\boldsymbol{x}_{ij} - \boldsymbol{x}_{kj}) \quad (1)$$

Here, $k \in \{1, 2, \ldots, N_{emp}\}$ and $j \in \{1, 2, \ldots, D\}$ are randomly chosen indices. Nemp is the number of employed bees. Φ_{ij} is a uniform random number generated from the range [-1, 1].

Step 4) Compute the fitness of both $\boldsymbol{x}_i$ and $\boldsymbol{v}_i$. Apply greedy selection scheme to choose the betterone.

Step 5) Calculate the selection probability, P_i for each solution, $\boldsymbol{x}_i$and normalize the probability value by (2).

$$P_i = fit_i \Big/ \sum_{k=1}^{N} fit_k \quad (2)$$

Step 6) Assign each onlooker bee to a solution, $\boldsymbol{x}_i$ at random with probability proportional to P_i.

Step 7) Produce new food positions (i.e. solutions), $\boldsymbol{v}_i$ for each onlooker bee using the employed bee$\boldsymbol{x}_i$by using (1).

Step 8) Evaluate the fitness of each employed bee, $\boldsymbol{x}_i$and its produced onlooker bee, $\boldsymbol{v}_i$. Apply greedy selection scheme to keep the one with better fitness and discard the other.

Step 9) If a particular solution has not been improved over a number (say, 100) of cycles, then select it for abandonment. Replace it by placing a scout bee at a food source placed uniformly at random over the entire search space using (3), i.e., for$j = 1, 2, \ldots, D$

$$\boldsymbol{x}_{ij} = min_j + rand\,(0,1) * (max_j - min_j) \quad (3)$$

Step 10) Keep track of the best solution found so far.

Step 11) Check for termination. If the best solution found is acceptable or maximum number of iterations has elapsed, stop and return the best solution found so far. Otherwise go back to step 2 and repeat.

3. Existing Variants ofthe ABC Algorithm

There exist several recent works (e.g., [16] – [34]) that try to tweak the explorative and/or exploitative properties of the basic ABC algorithm. For example, the cooperative ABC (CABC) algorithm [16] decomposes the search space into a number of subspaces and enforces more explorations by employing different bee colonies to explore the different subspaces. Another explorative variant — ABC with diversity strategy (DABC) [17] tries to preserve sufficient amount of diversity among the candidate solutions by switching between two different perturbation schemes. Chaotic ABC (CHABC) [18] is another explorative ABC-variant that uses dynamic chaotic sequence generators, instead of random number generators, to improve the explorative characteristics of the basic ABC algorithm. The explorative search capacity of ABC may also be improved by intelligent organization of the locally optimal points [19] and using the information of the global best solution, as in the Gbest-guided ABC (GABC) [20] algorithm. The Hooke

Jeeves ABC (HJABC) [21] is another improved ABC-variant that hybridizes the Hooke Jeeves pattern search technique with the basic ABC algorithm. The elitist ABC (EABC) [22] is a hybrid ABC variant which hybridizes ABC with two different local search operators to intensify the exploitations around the best solutions. Quan and Shi [23] reported improvement of the convergence speed by introducing an exploitative search iteration operator based on the fixed point theorem of contractive mapping. Qingxian and Haijun [24] employed the Boltzmann selection scheme and introduced an improved initialization scheme to improve the convergence speed. The hybrid crossover based ABC (CbABC) [25] is an exploitative variant that strengthens the exploitation phase of ABC by using a crossover operation. Another ABC-variant — NABC [26] alters the search pattern of employed and onlooker bees by searching around neighborhood of best solutions. JA-ABC [27] tries to improve average fitness of the bee population by replacing the poor solutions with perturbations of the fittest solution, which makes it exploitative. Some other recently introduced ABC variants can be found in the [28] – [34], but each of them comes with some limitations, such as inadequate degree of explorations [28] – [30], poor exploitations [31], slower rate of convergence of the algorithm [32] and increased computational complexity [33], [34].

However, most of these ABC-variants try to improve either the explorative or the exploitative properties if the basic ABC algorithm, while very few of them (e.g., [17]) tries to achieve a proper balance between explorations and exploitations. The explorative enhancements are usually based on more explorative perturbation, selection and/or initialization (e.g., [18], [19]) or employing some technique to maintain more population diversity (e.g., [16], [17]), while the exploitative developments are usually based on increasing the local search operations around the best candidate solutions ([21] – [22], [26] – [27]). Another limitation of all these ABC-variants (e.g., [16] – [34]) is that they do not consider the individual explorative/exploitative needs of the candidate solutions; rather they treat all the candidate solutions equally, employing some population-wide uniform strategy, identically on all candidate solutions. The proposed algorithm — EABC differs from all these algorithms in both these aspects. First, EABC improves both explorations and exploitations by introducing more randomness during perturbations and by incorporating a crossover operation after perturbations. Second, EABC customizes explorations and exploitations separately for every candidate solution $\boldsymbol{x}_i$ of the bee population by introducing and separately maintaining a control parameter R_i for each $\boldsymbol{x}_i$. Third, EABC tries to bring a proper balance between explorations and exploitations by abandoning the exploitative employed bee phase and the explorative scout bee phase of the original ABC algorithm. All these modifications are explained in details in the following section.

4. The Explorative ABC (EABC) Algorithm

There exist several aspects on which the proposed variant, EABC is significantly different from the original ABC algorithm. Firstly, EABC uses the following perturbation operation (4), which is different from the perturbation eq. (1) of the original ABC algorithm. For each employed bee $\boldsymbol{x}_{i,G}$ of the current generation G, EABC first produces a perturbed vector $\boldsymbol{w}_{i,G+1}$by using (4).

$$\boldsymbol{w}_{i,G+1} = \boldsymbol{x}_{r1,G} + \varphi_{i,G+1}\,(\boldsymbol{x}_{r2,G} - \boldsymbol{x}_{r3,G}) \quad (4)$$

Herer$_1$, r_2, $r_3$are randomly chosen indices from [1, N] that are mutually different and also different from the current index i. To be more precise, $r_1 \neq r_2 \neq r_3 \neq i$. The $\varphi_{i,G+1}$is a uniform, random variable that picks its values randomly from [0, 2]. The details of how the values of $\varphi_{i,G+1}$ are produced randomly are explained in a subsequent paragraph. If we compare the perturbation eq. (4) of EABC with the previously used eq. (1) by ABC, we can observe that EABC employs relatively higher degree of randomness in (4) by using three random indices r_1, r_2, $r_3$in order to ensure higher degree of search space explorations.

Secondly, EABC introduces a crossover operation to better combine the information of the original candidate solution $\boldsymbol{x}_{i,G}$ and perturbed candidate solution $\boldsymbol{w}_{i,G+1}$ to produce the new candidate solution $\boldsymbol{v}_{i,G+1}$. Each parameter of $\boldsymbol{v}_{i,G+1}$ is selected at random, either from $\boldsymbol{x}_{i,G}$ (with probability $R_{i,G+1}$) or from $\boldsymbol{w}_{i,G+1}$ (with probability $1 - R_{i,G+1}$). However, EABC also ensures that at least one parameter is selected from the perturbed vector $\boldsymbol{w}_{i,G+1}$; otherwise the new candidate solution $\boldsymbol{v}_{i,G+1}$ would be identical to its parent candidate solution $\boldsymbol{x}_{i,G}$. The details of how the value of $R_{i,G+1}$ is produced and maintained are explained in a subsequent paragraph. After producing $\boldsymbol{v}_{i,G+1}$, EABC employs a greedy selection between the original candidate solution $\boldsymbol{x}_{i,G}$ and the new candidate solution $\boldsymbol{v}_{i,G+1}$, similar to the step (4) of the original ABC algorithm.

Thirdly, to customize the degree of explorations and exploitations at the individual solution level, EABC maintains the scaling factor φ_i and crossover rate R_i, separately for every candidate solution $\boldsymbol{x}_i$. The values of φ_i and R_i are gradually adapted, generation by generation, separately for each candidate solution, by using the following Eqs. (5) and (6).

$$\varphi_{i,G+1} = \begin{cases} \varphi_{min} + \text{rand}(0,1) * (\varphi_{max} - \varphi_{min}); & \text{if rand}(0,1) \leq p_1 \\ \varphi_{i,G} & \text{otherwise} \end{cases} \quad (5)$$

$$R_{i,G+1} = \begin{cases} R_{min} + \text{rand}(0,1) * (R_{max} - R_{min}); & \text{if rand}(0,1) \leq p_2 \\ R_{i,G} & \text{otherwise} \end{cases} \quad (6)$$

Both the above equations put proper emphasis not only on search space explorations by innovation (i.e., trying new values for $\varphi_i$and R_i, with probability p_1 and p_2, respectively), but also on search space exploitations by inheritance (i.e., using the same value of parent $\boldsymbol{x}_i$, with probability = 1–p_1 and 1–p_2 for φ_i and R_i, respectively). Together they try to balance between explorations and exploitations, by using a suitable value of all these parameters — p_1 and p_2, φ_{min} and φ_{max}, R_{min} and R_{max}. EABC uses the following parameter values: $p_1 = p_2 = 0.1$, $\varphi_{min} = 0.1$, $\varphi_{max} = 0.9$, $R_{min} = 0$ and $R_{max} = 1.0$. Fourthly, with all the above explorative and exploitative measures, we have found (with some trial and error) that the necessity of using separate employed bee phase and scout bee phase (as used by the original ABC algorithm) does not exist anymore for the proposed EABC algorithm. So, EABC

has abandoned both the onlookerbee and scout bee phases (i.e., steps5–9) of the original ABC algorithm.

5. Experimental Studies

To evaluate the performance of the proposed EABC algorithm, we have used a standard benchmark suite on continuous function optimization problems, consisting of sixhigh dimensional functions [1], [2], [18]. Table 1 presents a brief overview on each of these benchmark functions. More details on each benchmark function can be found in [1]. All the benchmark functions we used are multimodal functions. To optimize a multimodal function, the search algorithm must possess both exploitative and explorative characteristics so that it can explore the locally optimal points without being trapped around any of them. Some of the multimodal functions can have hundreds of local minima, even when the dimensionality is just two or three. The number of local optima usually increases exponentially with the number of dimensions, which makes their optimization extremely difficult. For example, the Ackley function f_3 has one narrow global minimum basin, but with exponentially many minor local minima. The Griewank function f_4 has a component creating linkage among the variables, which complicates the search by perturbing any subset of the variables. Any technique that tries to optimize each variable separately without considering the others will fail with this function. The difficulty for the Schwefel function f_1 arises from its deep local minima which are far from the single global minimum. All these multimodal functions have exponentially many local minima and the number of local minima increases exponentially with the high dimensionality (i.e., $D = 30$), making them extremely difficult for any algorithm to be explored and optimized without being trapped around the locally optimal points of the search space.

Table 1: The six continuous benchmark functions used in our experimental studies. Here, *D*: dimensionality of the function, *S*: search space, f_{min}: function value at the global minimum, *C*: function characteristics with the values — *M*: Multimodal, *S*: Separable, *N*: Non-separable.

No	Function	*C*	*D*	*S*	f_{min}
f_1	Schwefel 2.26	*MS*	30	$[-500, 500]^D$	-12569.5
f_2	Rastrigin	*MS*	30	$[-5.12, 5.12]^D$	0
f_3	Ackley	*MN*	30	$[-32, 32]^D$	0
f_4	Griewank	*MN*	30	$[-600, 600]^D$	0
f_5	Penalized	*MN*	30	$[-50, 50]^D$	0
f_6	Penalized2	*MN*	30	$[-50, 50]^D$	0

Table 2: Performance of the proposed algorithm EABC, compared to the basic ABC algorithm on the benchmark functions. Results are averaged over 50 independent runs. Better performance on each function is marked with boldface font. In case the performance difference is not significant by *t*-Test with at least 99% level of confidence (i.e., $\alpha = 0.99$), it is marked by "Similar" at the rightmost column.

No	f_{min}	ABC		EABC		Better Performance (*t*-Test with $\alpha = 0.99$)
		Mean Error	Std. Dev.	Mean Error	Std. Dev.	
f_1	-12569.5	1.86e+01	5.21e+00	**1.034–02**	**3.66–03**	EABC
f_2	0	7.15 e–16	6.44e–17	**2.33e–31**	**3.58e–32**	EABC
f_3	0	8.97 e–12	1.15e–12	**1.12e–14**	**4.05e–15**	EABC
f_4	0	6.39e–16	9.25e–17	**1.09e–30**	**2.99e–31**	EABC
f_5	0	**7.03e–16**	**1.86e–16**	7.39e–14	1.76e–14	ABC
f_6	0	2.61e–03	8.36e–17	2.61e–03	8.22e–17	Similar

Table 2 presents the results of EABC with the basic ABC [1] algorithm. The common parameters of both the algorithms are set as — population size $N = 100$, maximum number of generations $MGN = 1500$ and $limit = 100$. The other parameters of EABC are set as: $p_1 = p_2 = 0.1$, $\varphi_{min} = 0.1$, $\varphi_{max} = 0.9$, $R_{min} = 0$ and $R_{max} = 1.0$. All the initial values of φ_i and R_i are set to 0.5 and 0.9, respectively for all the candidate solutions $\boldsymbol{x}_i$ of the initial bee population. All these values are selected after some initial experiments, and not meant to be optimal. Each algorithm has made 50 independent runs on each function. The mean and standard deviation of the error values (i.e., the difference between best found solution and the global minimum) found from the different runs are reported in Table 2. Our observations are summarized in the following few points.

- Out of the six functions $f_1 – f_6$, EABC performs better than ABC on as many as four functions, perform equally well on one, while ABC performs better only on the remaining one function. Thus the overall performance of EABC is significantly better than ABC.
- For all these functions, EABC reaches very close to the global minimum value (i.e., mean error ≈ 0), while the basic ABC algorithm fails to reach sufficiently close to the global minimum for one functions (i.e., f_1).
- The performance of EABC is very consistent, i.e., EABC regularly reaches very close to the global minimum, which is demonstrated by the very low standard deviation values of the errors of EABC.

In summary, EABC is more effective than the original ABC algorithm on almost all of these complex, high dimensional multimodal functions, which indicate the effectiveness of the proposed explorative techniques employed by EABC.

6. Conclusion

This paper introduces EABC — an explorative variant of the basic ABC algorithm and evaluates its performance on several standard continuous benchmark functions. Results indicate that EABC can perform better than ABC on most of these functions. There might be several possible ways to further improve EABC. Firstly, EABC uses a simple strategy to adapt the scaling factor and crossover rate for each

candidate solution. Some more sophisticated scheme, such as some scheme parameterized by the current maturity of the search process, may improve the algorithm further. Secondly, EABC puts more emphasis to increase the degree of explorations, rather than exploitations. Putting somewhat more emphasis on exploitations, especially around the best-so-far candidate solutions, may further improve the results. Thirdly, the quality of the final solution might be improved further by using an efficient local searcher after the execution of EABC is over. Finally, EABC has been applied only on the benchmark continuous optimization problems. It would be interesting to study how well EABC can perform on many other existing problems, especially the discrete and real world ones.

References

[1] D. Karaboga and B. Basturk, On the performance of artificial bee colony (ABC) algorithm, *Applied Soft Computing***8** (1) (2008) 687–697.
[2] D. Karaboga, An idea based on honey bee swarm for numerical optimization, Erciyes University, Kayseri, Turkey, *Technical Report-TR06*, 2005.
[3] D. Karaboga and B. Akay, A comparative study of artificial bee colony algorithm, *Applied Mathematics and Computation***214** (1) (2009) 108–132.
[4] S. Sobti and P. Singla, Solving travelling salesman problem using bee colony based approach, *International Journal of Engineering Research and Technology***2** (6) (2013) 186–189.
[5] K. Naidu, H. Mokhlis and A.H.A. Bakar, Multiobjective optimization using weighted sum Artificial Bee Colony algorithm for Load Frequency Control, *International Journal of Electrical Power and Energy Systems***55** (2) (2014) 657–667.
[6] R. Mukherjee, D. Goswami and S. Chakraborty, Parametric optimization of Nd:YAG laser beam machining process using artificial bee colony algorithm, *Journal of Industrial Engineering*, vol. 2013, Article ID 570250, 15 pages, 2013. DOI: 10.1155/2013/570250.
[7] H. Garg, Solving structural engineering design optimization problems using an artificial bee colony algorithm, *Journal of Industrial and Management Optimization*,**10** (3) (2014) 777–794.
[8] Z. Zhao, D. Yin and Y. Jiang, Improved bee colony algorithm based on knowledge strategy for digital filter design, *International Journal of Computer Applications*,**47** (2) (2013) 241–248.
[9] A. Mishra, A. Khanna, N. Singh and V. Mishra, Speed control of DC motor using bee colony optimization, *Universal Journal of Electrical and Electronic Engineering* **1** (3) (2013) 68–75.
[10] A. Karegowda and M. Darshan, Optimizing feed forward neural network connection weights using artificial bee colony algorithm, *International Journal of Advanced Research in Computer Science and Software Engineering* **3** (7) (2013) 452–454.
[11] A. Bolaji, A. Khader, M. Betar and M. Awadallah, Bee colony algorithm, its variants and applications: A survey, *Journal of Theoretical and Applied Technology* **47** (2)(2013) 434–459.
[12] T. Park and K. R. Ryu, A Dual population genetic algorithm for adaptive diversity control, *IEEE Trans. Evolutionary Computation* **14** (6) (2010) 865–884.
[13] R. K. Ursem, Diversity guided evolutionary algorithms, in *Proc. 7th Int. Conf. Parallel Problem Solving from Nature* (*PPSN*), 2002, pp. 462–474.
[14] J. Lampinen and I. Zelinka, On stagnation of the differential evolution algorithm, in *Proc. 6th Int. Mendel Conf. Soft Computing*, Brno, Czech Republic, 2000, pp. 76–83.
[15] V. Tereshko, A. Loengarov, "Collective Decision-Making in Honey Bee Foraging Dynamics", *Comput. Inf. Sys. J.*, vol. 9, no. 3, pp. 1–7, 2005.
[16] M. Abd, A cooperative approach to the artificial bee colony algorithm, in *Proc. IEEE Congress on Evolutionary Computation* (*CEC*), 2010, pp. 1–5.
[17] W. Lee and W. Cai, A novel artificial bee colony algorithm with diversity strategy, in *Proc. 7th Int. Conf. Natural Computation*, 2011, pp. 1441–1444.
[18] B. Wu and S. Fan, Improved artificial bee colony algorithm with chaos, in *Computer Science for Environmental Engineering and Eco-Informatics, Part I, Communications in Computer and Information Science*, eds. Y. Yu, Z. Yu and J. Zhao, vol. 158, 2011, pp. 51–56.
[19] L. Fenglei, D. Haijun and F. Xing, The parameter improvement of bee colony algorithm in TSP problem, *Science Paper Online*, Nov. 2007.
[20] G. Zhu and S. Kwong, Gbest-guided artificial bee colony algorithm for numerical function optimization, *Applied Mathematics and Computation***217** (7) (2010) 3166–3173.
[21] F. Kang, J. Li, Z. Ma and H. Li, Artificial bee colony algorithm with local search for numerical optimization, *Journal of Software* **6** (3) (2011) 490–497.
[22] E. Montes and R. Koeppel, Elitist artificial bee colony for constrained real-parameter optimization, in *Proc. IEEE Congress on Evolutionary Computation*, 2010, pp. 1–8.
[23] H. Quan and X. Shi, On the analysis of performance of the improved artificial bee colony algorithm, in *Proc.4th Int. Conf. Natural Computation* (*ICNC*), 2008, pp. 654–658.
[24] F. Qingxian and D. Haijun, Bee colony algorithm for the function optimization, *Science Paper Online*, Aug. 2008.
[25] S. Kumar, V. Sharma and R. Kumari, A novel crossover based artificial bee colony algorithm for optimization, *International Journal of Computer Applications***82** (8) (2013) 18–25.
[26] Y. Xu, P. Fan and L. Yuan, A simple and efficient artificial bee colony algorithm, *Mathematical Problems in Engineering*, vol. 2013, Article ID 526315, 9 pages, 2013. DOI: 10.1155/2013/526315.
[27] N. Sulaiman, J. Saleh and A. Abro, A modified artificial bee colony (JA-ABC) optimization algorithm, in *Proc. International Conference on Applied Mathematics and Computational Methods in Engineering* (*AMCME*), 2013, pp. 74–79.
[28] A. Abro and J. Saleh, Enhanced global-best artificial bee colony optimization algorithm, in *Proc. 6th European Symposium on Computer Modeling and Simulation*, 2012, pp. 95–100.

[29] W. Gao, S. Liu and L. Huang, A global best bee colony algorithm for global optimization, *Journal of Computational and Applied Mathematics***236** (11) (2012) pp. 2741–2753.
[30] W. Gao and S. Liu, A modified artificial bee colony algorithm, *Computers and Operations Research***39** (3) (2012) pp. 687–697.
[31] 20 W. Gao and S. Liu, Improved artificial bee colony algorithm for global optimization, *Information Processing Letters***111** (17) (2011) pp. 871–882.
[32] G. Zhu, S. Kwong, Gbest-guided bee colony algorithm for numerical optimization, *Applied Mathematics and Computation***217** (7) (2010) pp. 3166–3173.
[33] A. Abro and J. Saleh, An enhanced artificial bee colony optimization algorithm, *Recent Advances in Systems Science and Mathematical Modelling*, ed. D.S. Nikos Mastorakis, ValeriuPrepelita, 2012: WSEAS Press.
[34] A.Banharnsakun, T. Achalakul and B. Sirinaovakul, The best-so-far selection in artificial bee colony algorithm, *Applied Soft Computing***11** (2) (2011) pp. 2888–2901.