International Journal of Applied Information Systems (IJAIS) – ISSN : 2249-0868
Foundation of Computer Science FCS, New York, USA
Volume 10 – No.4, January 2016 – www.ijais.org

Experimental Comparison between Differential Evolution and Artificial Bee Colony Algorithm: A Case Study with Continuous Optimization Problems

Mohammad Shafiul Alam
Ahsanullah University of Science and Technology
Dhaka-1208, Bangladesh

Raiyan Yousuf
Ahsanullah University of Science and Technology
Dhaka-1208, Bangladesh

Faria Alam
Ahsanullah University of Science and Technology
Dhaka-1208, Bangladesh

Hossain Shaikh Saadi
Ahsanullah University of Science and Technology
Dhaka-1208, Bangladesh

ABSTRACT
This paper conducts an experimental comparison between two recently introduced meta-heuristic algorithms, which are the Differential Evolution (DE) and the Artificial Bee Colony (ABC) algorithm. Both these algorithms are very prominent and significant to represent the broader family of algorithms to which they belong, i.e., the Evolutionary and Swarm Intelligence algorithm families. Both DE and ABC have been successfully employed to numerous and diverse problems from the fields of mathematics, science and engineering. DE is an evolutionary algorithm that computes the vector differences between randomly picked candidate solution vectors and uses these differences to produce new, improved candidate solutions to advance its evolutionary search and optimization process. The ABC is a swarm intelligent algorithm that mimics the candidate solutions as a swarm of bees that forage across a search space for continuously better quality food sources (i.e., candidate solutions). The aim and focus of this paper is to present a side-by-side comparison of these two evolutionary and swarm intelligence algorithms on a common set of continuous benchmark problems to achieve a better understanding of their strengths, weaknesses and characteristics. The experimental results show that ABC is more explorative and can consistently avoid the local optima to locate the neighborhood of the global minimum, while DE is more exploitative to achieve an excellent level of fine tuning, but at the risk of premature convergence because of its lack of explorative characteristics.

Keywords
Evolutionary algorithm, swarm intelligence, artificial bee colony algorithm, differential evolution, continuous optimization

1. INTRODUCTION
Recently the research field of heuristic and meta-heuristic algorithms has observed the emergence of several successful evolutionary algorithms (EAs) and swarm intelligence based algorithms (SIAs). Both EAs and SIAs are bio-inspired meta-heuristic algorithms that are based on the Darwinian theory of evolution and the behavior of intelligent, automated, self-organized swarms found in nature, such as bee swarm, ant colony, bird flock and fish school. EAs and SIAs have been successfully applied on complex mathematical and engineering problems, such as continuous optimization [1]–[3], combinatorial optimization [4], multi-criteria optimization [5], process analysis and control [6], engineering design [7], planning of digital IIR filters [8], PID controllers [9], artificial intelligence [10] and so on [11].

Both EAs and SIAs are usually resistant against premature convergence. This is because the pool of individuals or swarm members that act as candidate solutions can usually preserve sufficient amount of diversity and this diversity is necessary to continue the explorations around the locally optimal points. However, the opposite scenario has also been observed sometimes ([12]–[14]) when the pool of candidate solutions did completely lose diversity and the optimization procedure got stuck around some locally optimal points. Such a scenario is known as 'premature convergence' in the literature of EAs and SIAs. The primary reason behind premature convergence is using less amount of exploration and more exploitation by the algorithm. But increasing explorations without restrictions and at the cost of decreasing exploitations usually lead to disappointingly slow convergence speed. This is why a balance between the explorative and exploitative properties of the algorithm is always desired for good results and satisfactory convergence speed, especially for complex, high dimensional, multimodal functions that have many local optima along each search dimension and the number of local optima increasing exponentially with the number of dimensions.

The problem of premature convergence is being addressed by both EA and SIA family algorithms. There exist several improved variations of many EAs and SIAs that try to avoid premature convergence, usually by modifying their selection and perturbation operators. This paper makes a careful study and comparison of two such improved EA and SIA variants, which are the Differential Evolution (DE) and the Artificial Bee Colony (ABC) algorithm. DE adopts an automatic adaptation of the degrees of explorations and exploitations by using the differences among the candidate solutions during their perturbations. These differences tend to be large during initial generations (i.e., iterations of the algorithm), but gradually smaller during the later generations, which ensures more explorations during early generations, followed by gradually increased exploitations and reduced explorations for later generations. In contrast, ABC adopts a more explicit and direct approach to handle the exploration vs. exploitation problem by partitioning the swarm of bees (i.e., candidate solutions) into three groups (i.e., 'employed', 'onlooker' and 'scout' bees) and explicitly dedicating some of them (i.e., the employed and scout bees) for explorations, while dedicating the others (i.e., onlooker bees) for exploitations. Thus, DE and ABC try to achieve the same goal (i.e., dynamic adaptation of

the degrees of explorations and exploitations), but using very different methodologies (i.e., automatic, indirect adaptation with no explicit control vs. explicit, direct division of work for explorations and exploitations). Since DE and ABC algorithms belong to different families of algorithms (i.e., EA and SIA), they are unlikely to be compared ever before on the same set of problems. The primary contribution of this paper is an experimental side-by-side comparison between DE and ABC with the aim to have valuable insight on their characteristics and strengths against premature convergence.

Following is the organization of the rest of this paper. Section 2 introduces the continuous optimization problem. Section 3 briefly describes the DE algorithm. Section 4 introduces the ABC algorithm in details. Section 5 presents the parameter settings and experimental setup of both DE and ABC and makes a comparison of their performance on seven complex, high dimensional multimodal functions. Finally, section 6 concludes the paper with a brief discussion on DE and ABC, followed by some suggestions as directions for further research with DE and ABC.

2. CONTINUOUS OPTIMIZATION PROBLEM

A continuous optimization problem can be formalized as follows.

$$\underset{\boldsymbol{x}}{\text{Minimize}}\ f(\boldsymbol{x});\ \text{subject to}\ \boldsymbol{x} \in S$$

Here, the goal is to find a vector $\boldsymbol{x}_{min}$ such that $f(\boldsymbol{x}_{min}) \leq f(\boldsymbol{x})$ for all $\boldsymbol{x} \in S$, where the search space S is a bounded subset of $\boldsymbol{R}^n$ and the objective function $f(.)$ is an n dimensional real valued function that is to be optimized over its parameter $\boldsymbol{x}$. Each element x_i of the vector $\boldsymbol{x}$ is a real valued variable, i.e., $\boldsymbol{x} = [x_1, x_2, \ldots, x_n]^T$

There exist many real world problems that can be formulated as optimization problems of its parameters or variables, most (or, all) of which assume values from the continuous domain. Thus the problem converts into the task of optimizing a continuous objective function. The generic term of 'continuous optimization' is often referred by many other different names, such as function optimization, numeric optimization and real parameter optimization. All of them essentially refer to the same undertaking of finding a globally optimum point (i.e., solution) across a real valued search space such that the point gives the best (minimum or maximum) value of the objective function. In the subsequent sections, this paper presents how the problem of continuous optimization is addressed by DE and ABC, which may also act as a model of how continuous optimization problems are dealt with the existing EAs and SIAs.

3. DIFFERENTIAL EVOLUTION

DE maintains a population of N vectors, each one being an n-dimensional real valued vector $\boldsymbol{x}_i = [x_{i1}, x_{i2}, \ldots, x_{in}] \in S$; where $S \subset \boldsymbol{R}^n$ and $i = 1, 2, \ldots, N$. This population of vectors (i.e., candidate solutions) gradually evolves, generation by generation, by the DE operators of mutation and crossover. During every generation, DE uses its mutation operation on every individual vector $\boldsymbol{x}_{i,G}$ (also called the target vector) to produce the mutated vector $\boldsymbol{v}_{i,G}$, then crossover operation on the target and mutated vectors to produce the trial vector $\boldsymbol{u}_{i,G}$, followed by the selection operation on the target and trial vectors to select the better one of them for the next generation.

Mutation Operation:

For each individual (i.e., target vector) $\boldsymbol{x}_{i,G}$ during generation G, an associated mutated vector $\boldsymbol{v}_{i,G} = [v_{i1,G}, v_{i2,G}, \ldots, v_{in,G}]$ is produced by using any of the following five strategies.

Strategy DE/best/1:

$$\boldsymbol{v}_{i,G} = \boldsymbol{x}_{best,G} + F \,.\, (\boldsymbol{x}_{r_1,G} - \boldsymbol{x}_{r_2,G})$$

Strategy DE/rand/1:

$$\boldsymbol{v}_{i,G} = \boldsymbol{x}_{r_1,G} + F \,.\, (\boldsymbol{x}_{r_2,G} - \boldsymbol{x}_{r_3,G})$$

Strategy DE/best/2:

$$\boldsymbol{v}_{i,G} = \boldsymbol{x}_{best,G} + F \,.\, (\boldsymbol{x}_{r_1,G} - \boldsymbol{x}_{r_2,G}) + F \,.\, (\boldsymbol{x}_{r_3,G} - \boldsymbol{x}_{r_4,G})$$

Strategy DE/rand/2:

$$\boldsymbol{v}_{i,G} = \boldsymbol{x}_{r_1,G} + F \,.\, (\boldsymbol{x}_{r_2,G} - \boldsymbol{x}_{r_3,G}) + F \,.\, (\boldsymbol{x}_{r_4,G} - \boldsymbol{x}_{r_5,G})$$

Strategy DE/current-to-best/1:

$$\boldsymbol{v}_{i,G} = \boldsymbol{x}_{i,G} + F \,.\, (\boldsymbol{x}_{best,G} - \boldsymbol{x}_{i,G}) + F \,.\, (\boldsymbol{x}_{r_1,G} - \boldsymbol{x}_{r_2,G})$$

Here, the indices r_1, r_2, r_3, r_4, r_5 are random integers which are different from each other and generated uniformly at random from the range [1, N]. They are also different from i (i.e., the index of the current vector). The vector $\boldsymbol{x}_{best,G}$ is the best individual vector with highest fitness value in the current generation G, and F is a parameter of the algorithm that takes values from the range (0, 1+) which acts as a scaling factor for the vector differences.

Crossover operation:

After the mutation operation, as explained above, the crossover operation takes place between each pair of target vector $\boldsymbol{x}_{i,G}$ and the corresponding mutated vector $\boldsymbol{v}_{i,G}$ to produce a trial vector $\boldsymbol{u}_{i,G} = [u_{i1,G}, u_{i2,G}, \ldots, u_{in,G}]$.

for $j = 1, 2, \ldots, n$

$$u_{ij,G} = \begin{cases} v_{ij,G}, & \text{if } \left(rand_j[0,1] \leq CR \text{ or } j = j_{rand}\right) \\ x_{ij,G}, & \text{otherwise} \end{cases}$$

Here, the crossover rate CR is a user specified parameter of DE in the range of [0,1). The integer j_{rand} is randomly picked from [1,n] to ensure that the trial vector $\boldsymbol{u}_{i,G}$ is always different from the original target vector $\boldsymbol{x}_{i,G}$ by at least one parameter.

Selection operation:

DE employs a greedy selection procedure between each pair of target vector and trial vector. The fitness value of each trial vector $\boldsymbol{u}_{i,G}$ is computed and compared with the fitness value of the corresponding target vector $\boldsymbol{x}_{i,G}$. If the trial vector $\boldsymbol{u}_{i,G}$ has smaller or equal function value (for a minimization problem) than the corresponding target vector $\boldsymbol{x}_{i,G}$, then $\boldsymbol{u}_{i,G}$ will replace the original vector $\boldsymbol{x}_{i,G}$ and be included into the population for the next generation. Otherwise, the original target vector $\boldsymbol{x}_{i,G}$ will be kept and $\boldsymbol{u}_{i,G}$ will be discarded.

4. ARTIFICIAL BEE COLONY (ABC) ALGORITHM

Bees have to forage over a vast area in search of good food sources (i.e., nectar). After an initial exploration stage, more and more bees are employed to collect honey from more profitable food sources whereas fewer bees are assigned to the less worthy sources. After returning the hive, each bee goes to the 'dance floor' and performs a special dance known as the 'waggle dance' to share the information of the food source it

has found. The 'onlooker' bees, waiting around the dance floor, observe the waggle dances of the 'employed' bees and pick any of them to follow and collect nectar from the vicinity of its food source. Some scout bees are also assigned for random explorations of the search space to find new food sources. The basic ABC algorithm mimics the food foraging behavior of honey bees with the same three groups of bees — employed, onlooker and scout bees. A bee working to forage a particular food source (i.e., candidate solution) and searching only around its vicinity is called an employed bee. Onlooker bees randomly pick and follow any of the employed bees. The probability of picking an employed bee is proportional to the quality of its food source. Scout bees can perform random explorations of the search space to find new food sources. If the employed and onlooker bees, even after limit attempts, fail to find a better food position around a particular food $\boldsymbol{x}_i$, then $\boldsymbol{x}_i$ is abandoned and replaced by initiating a scout bee and its food source is placed uniformly at random across the search space. In the original implementation of the ABC algorithm, half of the colony is employed bees, the other half is onlooker bees, and scout bees are created on demand only when a food source fails to improve with several attempts. Each cycle (i.e., iteration) of ABC consists of foraging by the employed bees, then foraging by the onlookers, followed by placement of the scout bees (if necessary). Each of these stages is briefly described below.

Foraging by employed bees: Suppose, an employed bee is currently positioned at a food source position $\boldsymbol{x}_i$. During this stage, each employed bee searches in the vicinity of its current position $\boldsymbol{x}_i$ to produce new trial food source $\boldsymbol{v}_i$ using (1), where $j \in \{1, 2, \ldots, D\}$ and $k \in \{1, 2, \ldots, SN\}$ are randomly picked indices, D is dimensionality of the problem, SN is the number of food positions and φ_{ij} is a uniform random value ~ [-1, 1].

$$v_{ij} = x_{ij} + \varphi_{ij} (x_{kj} - x_{ij}) \tag{1}$$

Thus, the new solution $\boldsymbol{v}_i$ is produced from $\boldsymbol{x}_i$ by perturbing its randomly picked j-th parameter and using the information of $\boldsymbol{x}_i$ and another randomly picked solution $\boldsymbol{x}_k$. If $\boldsymbol{v}_i$ has better 'fitness' than the old food position $\boldsymbol{x}_i$, then $\boldsymbol{x}_i$ is replaced by $\boldsymbol{v}_i$. For the problem of function optimization, where f is the function to be minimized, ABC computes the 'fitness' of a candidate solution $\boldsymbol{x}_i$ by using (2).

$$fitness(\boldsymbol{x}_i) = \begin{cases} \dfrac{1}{1+f(\boldsymbol{x}_i)}; & \text{if } f(\boldsymbol{x}_i) \geq 0 \\ 1+\left|f(\boldsymbol{x}_i)\right| & \text{otherwise} \end{cases} \tag{2}$$

Foraging by onlooker bees: During this stage, each onlooker bee randomly picks an employed bee to follow and forages only around the vicinity of its food source. The probability w_i that the employed bee with food source $\boldsymbol{x}_i$ would be picked by an onlooker bee is computed using (3), which makes the probability w_i to be proportional to fitness ($\boldsymbol{x}_i$).

$$w_i = \frac{fitness(x_i)}{\sum_{n=1}^{SN} fitness(x_n)} \tag{3}$$

Like the employed bees, each onlooker bee also employs (1) to produce trial food source $\boldsymbol{v}_i$ in the vicinity of its current food source position $\boldsymbol{x}_i$. If $\boldsymbol{v}_i$ has better fitness than $\boldsymbol{x}_i$, then $\boldsymbol{x}_i$ is replaced by $\boldsymbol{v}_i$. Otherwise, $\boldsymbol{v}_i$ is discarded.

Placement of Scout bees: A scout bee is created only when a particular food source $\boldsymbol{x}_i$ failed to be improved over the last '*limit*' iterations. The bee employed to $\boldsymbol{x}_i$ now becomes a scout bee and its food source is replaced randomly across the search space using (4), where $j = 1, 2, \ldots, D$ and $[min_j, max_j]$ is the search space along the j-th dimension.

$$x_{ij} = min_j + rand\,(0,1) * (max_j - min_j) \tag{4}$$

The detailed pseudo code of the ABC algorithm is presented in details in the following few points.

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 bee searches 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).

Step 4) Compute the fitness of both $\boldsymbol{x}_i$ and $\boldsymbol{v}_i$ by using (2). Apply greedy selection scheme to choose the better one and discard the other.

Step 5) Calculate the selection probability, w_i for each solution, $\boldsymbol{x}_i$ and normalize the probability value by using (3).

Step 6) Assign each onlooker bee to an employed bee, $\boldsymbol{x}_i$ at random with probability proportional to w_i.

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

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

Step 9) If a particular solution has not been improved over the past '*limit*' cycles (say, *limit* = 100 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 by using (4).

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.

5. EXPERIMENTAL STUDIES

In order to compare the performance of ABC and DE, this paper uses a standard benchmark suite of continuous optimization problems, consisting of seven complex, high dimensional, multi-modal functions [1]–[3], [15], [16]. An overview of the continuous objective functions is presented in Table 1. For optimizing any multimodal function, DE and ABC must exhibit both exploitative and explorative properties, because the algorithm has to explore many locally optimal points without getting trapped around any of them. Most of the multimodal functions have numerous local optima, even when the dimensionality is just two or three. Increasing the number of dimensions cause the number of local optima to grow exponentially. 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 for 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 their high dimensionality (i.e., $D = 30$), making their optimization extremely difficult for any algorithm

Table 1: The seven continuous benchmark functions used in experimental studies. Here, *D*: dimensionality of the function, *S*: search space, f_{min}: function value at the global minimum, *C*: function characteristics with the following 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	Rosenbrock	*MS*	30	$[-30, 30]^D$	0
f_6	Penalized	*MN*	30	$[-50, 50]^D$	0
f_7	Penalized2	*MN*	30	$[-50, 50]^D$	0

Table 2: Performance comparison of ABC and two DE variants on seven benchmark functions. The values indicate the error (i.e., difference between global minimum and the minimum possible function value found by ABC and DE variants) on the different functions. The best performance (i.e., minimum error) on each function is marked with boldface font

No	f_{min}	Generations	DE/best/1/exp	DE/rand/1/exp	ABC	Best Performance by
f_1	–12569.5	9000	7.69e+03	1.34e–02	**7.28e–11**	ABC
f_2	0	5000	**1.48e–201**	2.31e–64	6.12e–16	DE/best/1/exp
f_3	0	1500	20.01	20.02	**1.22e–11**	ABC
f_4	0	2000	**5.78e–82**	6.48e–27	7.31e–16	DE/best/1/exp
f_5	0	20000	9.03e–29	**0**	2.77e–02	DE/rand/1/exp
f_6	0	1500	**7.39e–14**	7.40e–14	1.22e–11	DE/best/1/exp
f_7	0	1500	2.61e–03	2.61e–03	**6.95e–16**	ABC

Table 2 presents the results of ABC and DE on the seven benchmark functions. The common parameter of both the algorithms is the population/swarm size *N*, which is set to 100. The no. of maximum generations is different for the different functions, as shown in Table 2. The *F* and *CR* parameters of DE are set to 0.5 and 0.9, respectively. The *limit* parameter of ABC is set to 100. All these values are selected after some initial experiments, and not meant to be optimum. The important observations on the results in Table 2 are summarized in the following few points.

- Out of the seven functions $f_1 – f_7$, ABC outperforms both the DE variants on three functions (i.e., f_1, f_3, f_7), while DE performs better on the remaining four.
- For all the seven functions, ABC has reached sufficiently close to the global minimum (error ≈ 0), while the DE variants have failed to do so on as many as three functions (f_1, f_3 and f_7).
- The overall performance of the algorithms can be compared based on their mean absolute error (MAE) over the seven functions. The MAE of ABC is only 3.96e–03, which is much smaller than both DE/best/1/exp (MAE = 1.10e+03) and DE/rand/1/exp (MAE = 2.86e+00). Thus, the overall performance of ABC is better than its DE counterparts.
- From the viewpoint of exploration vs. exploitation phenomenon, the DE variants are much more exploitative than ABC. This becomes apparent from their extremely low error values for some functions (e.g., f_2, f_4 and f_5). But such intense exploitations come at the cost of unsatisfactory explorations, causing DE to fail for some functions (e.g., f_1 and f_3) by prematurely converging around the locally optimal points of the search space. In contrast, ABC is sufficiently explorative to locate the neighborhood of the global minimum for all these

functions, though it may not perform as intense fine tuning as DE on some functions (e.g., f_2, f_4 and f_5).

To summarize the experimental findings, ABC is more explorative than DE, and hence it can locate the neighborhood of the global minimum for all the seven functions, without showing any sign of premature convergence. In contrast, DE is more exploitative, hence it can perform excellent fine tuning once it reaches near the global minimum, but it may also converge prematurely because of its lack of explorative properties (e.g., for f_1 and f_3).

6. CONCLUSION

This paper presents an experimental evaluation and comparison of two different meta-heuristic algorithms — one from the evolutionary algorithm family (i.e., DE) and the other from swarm intelligence family (i.e., ABC) using several benchmark problems on continuous optimization. Results indicate that ABC is more robust against premature convergence because of its more explorative design (i.e., explicit explorations by employed and scout bees), while DE is better for fine tuning and exploitations, but with the risk of premature convergence because of its lack of explorations.

There might be several possible future research directions based on DE and ABC. Firstly, since DE is more exploitative than ABC, it might be possible to hybridize them together into a new, single algorithm. This new algorithm may deploy ABC during the early generations when more explorations are desired, but later may switch to more exploitation by employing the DE variants, especially the DE/best/1/exp variant which is extremely exploitative. Secondly, both DE and ABC might be compared on easier unimodal and low dimensional functions to gain further insights on their strengths and weaknesses. Thirdly, the possibility of improving the final solution quality might be investigated by using an efficient local searcher (or, the DE/best/1/exp variant). This may make their performance even better. Finally, this paper employs DE and ABC only on continuous benchmark functions. It might be interesting to observe and compare their performance on many other existing problems, especially the discrete and real world problems.

7. 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] X. Yao, Y. Liu and G. Lin, "Evolutionary programming made faster", *IEEE Transactions on Evolutionary Computation* **3** (2) (1999) 82–102.

[4] S. Sobti and P. Singla, Solving travelling salesman problem using bee colony based approach, *Int. Journal of Engg. 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] T. Bäck and H.–P. Schwefel, "An overview of evolutionary algorithms for parameter optimization", *Evolutionary Computation* **1** (1) (1993) 1–23.

[16] 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.