A Novel Hybrid Artificial Bee Colony with Monarch Butterfly Optimization for Global Optimization Problems

Waheed Ali H. M. Ghanem[1, 2, 3] and Aman Jantan[1]

[1]School of Computer Science, Universiti Sains Malaysia, Pulau Pinang, Malaysia
[2]Faculty of Education-Saber, University of Aden, Aden, Yemen
[3]Faculty of Engineering, University of Aden, Aden, Yemen

Email: waheed.ghanem@gmail.com
Email: aman@cs.usm.my

Abstract. This article introduces a novel hybrid approach between two of the meta-heuristic algorithms to solve global optimization problems. The proposed hybrid algorithm uses the butterfly adjusting operator in Monarch Butterfly Optimization (MBO) algorithm as a mutation operator to replace the employee phase of the Artificial Bee Colony (ABC) algorithm. The novel Hybrid ABC/MBO (HAM) algorithm addresses the issues of trapping in local optimal solutions, slow convergence, and low precision by improving the balance between the characteristics of exploration and exploitation. The proposed HAM algorithm is validated on eight benchmark functions, and is compared with ABC and MBO algorithms. The experimental results show that the HAM algorithm is clearly superior to both the standard ABC and MBO algorithms.

Keywords: Artificial bee colony algorithm; Monarch butterfly optimization algorithm; Global Optimization problem; Computation Intelligence.

1 Introduction

There are a lot of problems in the real world that involve a set of potential solutions, from which the one with the best quality is termed as the optimal solution, and the method of searching for such a solution is known as mathematical optimization. The quality of solutions is represented by the ability to maximize or minimize a certain function, called the objective function, while the pool of possible solutions that can satisfy the required objective is called the search space. One can traverse all possible solutions, examine the result of the objective function in each case, and select the best solution. However, many real problems are intractable using this exhaustive search strategy. In these problems, the search space expands exponentially with the input size, and exact optimization algorithms are impractical. The historical alternative in such situations is to resort to heuristics, similar to simple rules of thumb that humans would utilize in a search process. Heuristic algorithms implement such heuristics to explore the otherwise prohibitively large search space, but they do not guarantee finding the actual optimal solution, since not all areas of the space are examined. However, a close solution to the optimal is returned, which is "good enough" for the problem at hand.

adfa, p. 1, 2011.

The next step would be to generalize those heuristics in higher level algorithmic frameworks that are problem independent, and that provide strategies to develop heuristic optimization algorithms. The latter are known as metaheuristics [1]. Early metaheuristics were based on the concept of evolution, where the best solutions among a set of candidate solutions are selected in successive iterations, and new solution are generated by applying genetic operators such as crossover and mutation to the parent solutions.

Similar to and including evolutionary algorithms, many metaheuristics were based on a metaphor, inspired by some physical or biological processes. Many recent metaheuristics mimic the biological swarms in performing their activities; in particular, the important tasks of foraging, preying and migration. Popular examples of developed metaheuristic algorithms in this category include Particle Swarm Optimization (PSO) [2], which is inspired by the movement of swarms of birds or fishes; Ant Colony Optimization (ACO) [3, 4], which is inspired by the foraging behavior of ants, where ants looking for food sources in parallel employ the concept of pheromone to indicate the quality of the found solutions; and Artificial Bee Colony (ABC) algorithm, inspired by the intelligent foraging behavior of honey bees [5, 6].

The idea of deriving metaheuristics from natural-based metaphors proved so appealing that much more of such algorithms have been, and continue to be developed. A few more examples include Cuckoo Search (CS) [7, 8], Biogeography-Based Optimization (BBO) [9], Animal Migration Optimization (AMO) [10], Chicken Swarm Optimization (CSO) [11], Grey Wolf Optimization (GWO) [12], Krill Herd (KH) [13], and Monarch Butterfly Optimization (MBO) [14], which is inspired by the migration behavior of monarch butterfly. The Bat Algorithm (BA) [15] also belongs to the metaheuristics that are based on animal behavior; inspired by the echolocation behavior of bats in nature. On the other hand, several metaphor-based metaheuristics are derived from physical phenomena such as Simulated Annealing (SA) [16] which is inspired by the annealing process of a crystalline solid.

The aforementioned metaheuristics are classified as stochastic optimization techniques. To avoid searching the whole solution space, they include a randomization component to explore new solution areas. Though these random operators are essential, they can introduce two types of problems. First, if the randomization is too strong, the metaheuristic algorithm might keep moving between candidate solutions, loosely examining each localized region and failing to exploit promising solutions and find the best solution. Second, if the search process is too localized, exploiting the first found good solutions very well but failing to explore more regions, the algorithm might indeed miss the real optimal solution (called the global optimum), and trap into some local optima.

The perfect balance between exploitation and exploration is essential to all metaheuristics. In fact, it is whether and how this balance is achieved that distinguishes most metaheuristics from each other, and forms a source of new attempts to improve existing algorithms, possibly by hybridizing ideas from more than one metaheuristic strategy [18].

In this paper, we follow this path and introduce a new hybrid metaheuristic that augments the popular ABC algorithm with a feature from the MBO algorithm so as to make the correct balance between randomization of local search and global search.

The rest of this article is organized as follows. Section 2 describes the proposed HAM method, while Section 3 explains the setup of experimental evaluation. Section 4 presents and discusses the obtained results, and finally Section 5 concludes the paper.

2 Hybrid Algorithm Based on Artificial Bee Colony and Monarch Butterfly Optimization

This section introduces the (HAM) algorithm, which is based on the standard ABC [5,6] and MBO [14] algorithms. The ABC algorithm was proposed by Karaboga for optimizing numerical problems in 2005, and several developments were based on this algorithm [19, 20, and 21]. The MBO algorithm was proposed by Gai-Ge, Suash and Zhihua in 2015. It is a new nature-inspired metaheuristic optimization algorithm that works by simplifying and idealizing the migration behavior of monarch butterfly individuals between two distinct lands, namely (northern USA (Land1) and southern Canada (Lnad2)). For more details about the two algorithm please refer to [5, 14].

The most important factors in metaheuristic algorithms are the exploitation and exploration search mechanisms. A good metaheuristic algorithm has the ability to strike a balance between these two mechanisms, thereby enhancing the solving of low and high-dimensional optimization problems. The exploitation mechanism is based on the present knowledge to seek better solutions, while the exploration mechanism is based on fully searching the problem space for an optimal solution.

In general, by analyzing the standard MBO algorithm, we notice that it has the ability to explore the search space very effectively; however, it has a poor ability to exploit the search space due to the occasional use of Levy flight by the updating operators, which leads to large random steps or moves. On the other side, we notice that the ABC algorithm has the ability to explore the search space well, but has better ability in finding local optima through the *employee* and *onlooker* phases, which are considered local search processes. ABC is mostly based on selecting the solutions that improve the local search. Global search, on the other hand, is implemented in the ABC algorithm by the scout phase, which results in reducing the convergence speed during the search process.

The core idea of the new hybrid approach is based on two improvements; firstly, to modify the butterfly adjusting operator in the MBO algorithm in order to improve the exploitation versus exploration balance, by increasing the search diversity and counterbalance the shortfall of ABC algorithm in global search efficacy. The modified version of the operator is show in algorithm 1. The second improvement is to integrate the modified butterfly adjusting operator from MBO in place of the first phase in the standard ABC algorithm (the employee phase). The improved operator is named as “employee bee adjusting operator” and the resulting modified phase is called the “employee bee adjusting phase”.

The employee bee adjusting phase is very simple and is used to update all the solutions in the bee population, where each solution is a D-dimensional vector. In the Initialization phase, we need to define all the variables that would be defined in the standard ABC algorithm and assign them suitable values. The HAM algorithm adopts all parameters from the original ABC algorithm, and adds three new control parameters: *limit1*, *limit2* and the maximum walk step parameter S_{max}; these three parameters are used in the employee bee adjusting phase.

```
Algorithm1: Employee bee adjusting phase
Begin
For i = 1 to SN do
  Calculate the walk step dx by Equation (1);
  Calculate the weighting factor by Equation (2);
  For j = 1 to D do
    If rand ≥ limit1 then
      Generate the j^th element by Equation (3);
    Else
      Randomly select a food Source (r) by Equation (4);
      If rand < limit2 then
        Generate the j^th element by Equation (5);
      Else
        Generate the j^th element by Equation (6);
        If rand < BAR then
            Generate the j^th element by Equation (7);
        End if
      End if
    End if
  End for j
Evaluate the fitness value of the candidate solution x_i.
Apply a greedy selection process between x_i and x_best
If solution x_i does not improve, trial_i = trial_i + 1,
Otherwise trial_i = 0.
End for i
End
```

In Algorithm 1, each employee bee is assigned to its food source and in turn generates a new one either by using Levy flight or through mutation operators, which are based on the two control parameters (*limit1* and *limit2*). These parameters are used to fine-tune the exploitation versus exportation by improving the global search diversity. The first step is to calculate a walk step "dx" for the *i*^th^ bee using the Levy flight in Equation 1, and calculate the weighting factor"$\propto$" by Equation 2, where S_{max} represents the max walk step that a bee individual can move in one step, and *t* is the current generation. Then, for each element *j* of the D dimensions, if (rand $\geq$ *limit1*), the algorithm uses Equation 3 to update the solution element:

$$dx_k = levy(x_j^t) \quad (1)$$

$$\propto = S_{max}/t^2 \quad (2)$$

$$x_{i,j}^{t+1} = x_{best,j}^t \quad (3)$$

where $x_{i,j}^{t+1}$ is the j^{th} element of solution x_i at generation *t*+1, which represents the location of the solution *i*, while $x_{best,j}^t$ is the j^{th} element of x_{best} at generation *t*, which represents the best location among the food sources so far with respect to the i^{th} bee. On the contrast, if (rand <

limit1) then another set of updates are performed. First, a random food source (equivalent to a random solution or bee) is selected from the current population using Equation 4. Then, depending on whether a randomly generated value is smaller than *limit2*, Equation 5 is used to update the solution elements, as follows:

$$r = round((SN * rand) + 0.5) \quad (4)$$

$$x_{i,j}^{t+1} = x_{r,j}^{t} + 0.5 * rand * (x_{worst,j}^{t} - x_{r2,j}^{t} - x_{best,j}^{t}) \quad (5)$$

where $x_{i,j}^{t+1}$ is the j^{th} element of solution x_i at generation *t*+1, which represents the location of the solution *i*, $x_{best,j}^{t}$ is the j^{th} element of x_{best} at generation *t*, which represents the best location among the food sources so far; $x_{worst,j}^{t}$ is the j^{th} element of x_{worst} at generation *t*, which represents the worst location among the food sources so far; and $x_{r,j}^{t}$ is the j^{th} element of x_r at generation *t*, which represents the location of the solution *r* calculated by Equation 4. The *t* in Equation 5 is the current generation number.
On the other hand, if the randomly generated value was bigger than *limit2*, the solution elements are updated by Equation 6, where $x_{i,j}^{t+1}$ is the j^{th} element of solution x_i at generation *t*+1, which represents the location of the solution *i*; $x_{best,j}^{t}$ is the j^{th} element of x_{best} at generation *t*, which represents the best location among the food sources so far; $x_{worst,j}^{t}$ is the j^{th} element of x_{worst} at generation *t*, which represents the worst location among the food sources so far, while $x_{r,j}^{t}$ is the j^{th} element of x_r at generation *t*, which represents the location of the solution *r* calculated by Equation 4.

$$x_{i,j}^{t+1} = x_{r,j}^{t} + 0.5 * rand * (x_{best,j}^{t} - x_{r3,j}^{t} \; - \; x_{worst,j}^{t}) \quad (6)$$

The Levy flight step from the MBO algorithm is adopted here with a smaller probability of execution to reduce its impact on the exploitation process. Assuming the execution path passed the test of *limit1* and *limit2* control parameters, yet another random check against the *BAR* parameter is performed, right after the update by Equation 6 to further change the value of $x_{i,j}^{t+1}$ occasionally by the amount $\propto \times (\mathrm{dx_k} - 0.5)$, as per Equation 7.

$$x_{i,j}^{t+1} = x_{i,j}^{t+1} + \propto \times (dx_k - 0.5) \quad (7)$$

Finally, the employee bee adjusting phase tests the boundary for the new solution to make sure the newly generated solution is within the allowed boundaries for the optimization problem at hand, and then evaluates the fitness value of the new solution in order to apply a greedy selection process between the new and the best solutions to select the better one. If the solution does not improve then a trial counter is increased by one. As for the onlooker bee and scout phases, the algorithm adopts their implementation from the original ABC algorithm without any change, which can be found in [22].

3 Experimental Evaluation

In this section, we layout the experimental setup through which we have evaluated the proposed algorithm, HAM.

3.1 General setup

Hardware and software implementation.
All the experiments were conducted on a laptop with an Intel Core 5i processor running at 2.4 GHz, and 8 GB of RAM. The software implementation of the proposed HAM algorithm was based on the implementation of ABC and MBO. All software is compiled using MATLAB R2009b (V7.9.0.529) running under Windows 7.

Parameters.
For a fair comparison, we set all the common control parameters for all methods to the same values, including the population size SN, and the dimensionality of the search space D. The SN for our experiments in this work was set to 50 and the number of dimensions D was set to 10 for all methods. The parameters of all methods used in this work are presented below:

The variables of the proposed HAM algorithm have been set in all experiments as follows: S_{max} = 1.0, migration period *peri* = 1.2, the migration ratio *p* = 0.4167, *limit1* = 0.8 and *limit2* = 0.5. ABC parameter settings: The number of colony size is 50 employed bees and 50 onlooker bees because the colony size is 100; *limit* is set to 100. MBO parameter settings: There are many parameters for MBO method. In this work, we followed the setup in the original work of MBO [14], and set the butterfly adjusting rate *BAR* = 0.4167, max step S_{max} = 1.0, migration period *peri* = 1.2, the migration ratio *p* = 0.4167 and the population size NP is the same as the colony size, which is 50.

3.2 Benchmark Function

This paper uses a set of 8 test functions for global numerical optimization. These functions are listed in Table 1 alongside their respective equations and properties.

Table 1. Benchmark global numerical functions used for evaluating optimization methods

No.	Name	Equation	Low	Up
1	Sphere	$f(x) = \sum_{i=1}^{n} x_i^2$	-100	100
2	Schwefel 2.22	$f(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	-1.28	1.28
3	Schwefel 1.2	$f(x) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j\right)^2$	-5.12	5.12
4	Schwefel 2.21	$f(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	-600	600
5	Schwefel 2.26	$f(x) = -418.983 \sum_{i=1}^{n} \left[x_i \sin\left(\sqrt{\lvert x_i \rvert}\right)\right]$	-50	50
6	Rosenbrock	$f(x) = \sum_{i=1}^{n-1} \left[100\sqrt{\lvert x_i - x_i^2 \rvert} + (1 - x_i)^2\right]$	-100	100
7	Step	$f(x) = \sum_{i=1}^{n} \lfloor \lvert x_i \rvert \rfloor$	0	3.1416

8	Quartic	$f(x) = \sum_{i=1}^{n} ix_i^4 + rand[0,1)$	-5	10

4 Results

Table 2 lists the optimization results when applying the 8 optimization test functions to ABC, MBO and our HAM methods. The listed values are the optimal values of the objective function achieved by each algorithm after iterating over 50 generations. The *mean* values in the table are averaged over 20 runs (each run constitutes 50 iterations) and listed along the *standard deviation*. The *min* values, however, are the best results achieved by each algorithm at all. By the "best result" we mean the closest result to the actual optimal value of the function.

It is evident from Table 2 that the HAM algorithm can reach a better optimum on average; at least with respect to the set of benchmark functions used in the experiments (HAM has better average results in the case of 7 out of 8 test functions). For ease of recognition, the best average result is marked with bold font and shaded in a grey cell. The *min* values are bold font to identify the absolute best minimum achieved for each function. Note that this value is meaningful because it happened that the minimum achieved values by the algorithms for the selected benchmark functions are closest to the real optimum. With respect to the set of test functions used in our evaluations, HAM could achieve the best result in 6 out of 8 cases.

On another perspective, we also graphed the optimization process of each algorithm (for each benchmark function) as the value of the so-far best solution versus the current iteration, which shows the search path in terms of selected best solution per iteration. The curve of this kind is expected to decline overall at a slope that reflects the convergence speed of the algorithm (there is no degradation during the process of any included metaheuristic algorithm, as the best solution is either improved or kept unchanged at all iterations). Therefore, these graphs can be called the convergence plots of the algorithms. Because of the large number of plots, we include hereby representative samples of the convergence plots in Figure 1, which compares the convergence of HAM with the two most related metaheuristic techniques: ABC and MBO.

Table 2. The *min*, *mean* and *standard deviation* of test function values found by ABC, MBO and the proposed HAM algorithms, averaged over 20 experimental runs. The Dimensions set to 10.

ABC			MBO			HAM		
Best	Mean	Std. dev	Best	Mean	Std. dev	Best	Mean	Std. dev
4.13E-04	1.01E-02	9.37E-03	5.14E-04	8.67E-01	2.73E+00	**3.57E-05**	**9.37E-05**	**6.24E-05**
1.07E-01	2.64E-01	1.21E-01	3.79E-02	2.18E+00	4.08E+00	**7.14E-03**	**1.70E-02**	**8.60E-03**
6.66E+02	2.11E+03	1.00E+03	7.61E-03	4.18E+03	3.20E+03	**6.72E-03**	**2.97E+00**	**1.16E+01**
9.89E+00	2.14E+01	7.13E+00	2.35E-02	1.68E+01	1.57E+01	**5.84E-03**	**1.22E-02**	**4.64E-03**
2.45E+02	6.47E+02	1.89E+02	1.29E-04	1.22E+03	6.79E+02	1.13E+03	1.55E+03	**1.85E+02**
2.31E+01	1.78E+02	2.33E+02	8.91E+00	7.14E+04	1.54E+05	**8.20E+00**	**8.56E+00**	**1.24E-01**
3.44E+00	5.30E+00	1.85E+00	6.52E-06	1.29E+03	2.43E+03	2.51E+00	**2.52E+00**	**6.70E-03**
2.02E+00	2.97E+00	4.55E-01	1.83E+00	3.31E+00	1.34E+00	**1.54E+00**	**2.19E+00**	**2.43E-01**

Figures 1 (a-d) show that the HAM algorithm enjoys not only a superior overall performance in terms of the quality of the found optimal solution, but also a faster convergence especially in the earlier stages. Although the starting points of the algorithms are close to each other in the plots of the four testing functions in the figure, the proposed HAM method does not trap into a quick local optimum, unlike the original ABC and MBO algorithms for example.

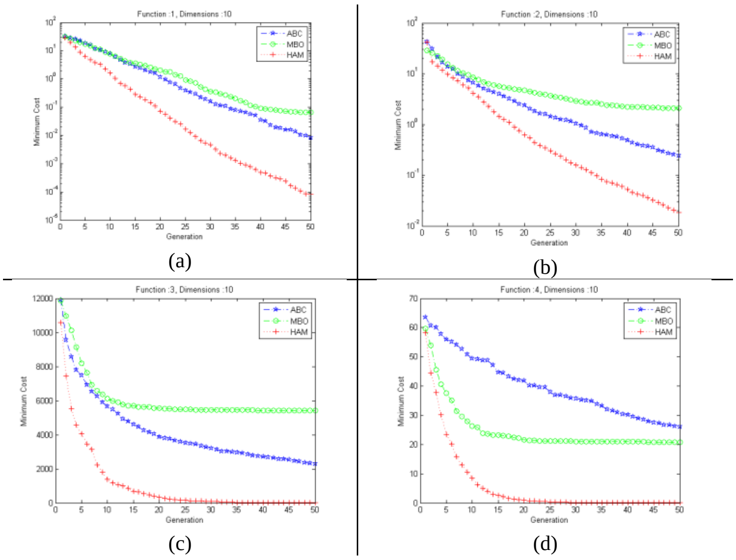

Fig. 1. Performance of ABC, MBO and HAM algorithms for (a) F1, (b) F2, (c) F3 and (d) F4 benchmark functions.

5 Conclusion

In this article, a Hybrid algorithm of Artificial Bee Colony and Monarch Butterfly Optimization algorithms (HAM) was proposed for solving numerical optimization problems. This method is based on a modified version of the *adjusting operator* in MBO algorithm, integrated as a first phase in the standard ABC algorithm, in place of the *employee bee phase*. In the HAM method, the improved diversification of MBO was used to augment the good intensification ability of ABC to find better global solutions and increase the convergence speed. As a future work, we plan to use the new method in training artificial neural networks (ANN) for various purposes, and to extend the method for solving multi-objective optimization problems

Acknowledgments.
This research was funded by Universiti Sains Malaysia under USM Fellowship 2016 [APEX (1002/CIPS/ ATSG4001)], Also partially supported by the Fundamental Rsearch Grant Scheme (FRGS) for “Content Based Analysis Framework for Better Email Forensic and Cyber Investigation” [203/PKOMP/6711426].

REFERENCE

1. Sörensen, K. and F.W. Glover, *Metaheuristics*, in *Encyclopedia of Operations Research and Management Science*. 2013, Springer. p. 960-970.
2. Eberhart, R.C. and J. Kennedy. *A new optimizer using particle swarm theory*. in *Proceedings of the sixth international symposium on micro machine and human science*. 1995. New York, NY.
3. Dorigo, M., M. Birattari, and T. Stützle, *Ant colony optimization.* Computational Intelligence Magazine, IEEE, 2006. 1(4): p. 28-39.
4. Dorigo, M., V. Maniezzo, and A. Colorni, *Ant system: optimization by a colony of cooperating agents.* Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on, 1996. 26(1): p. 29-41.
5. Karaboga, D., *An idea based on honey bee swarm for numerical optimization*. 2005, Technical report-tr06, Erciyes university, engineering faculty, computer engineering department.
6. Karaboga, D. and B. Basturk, *A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm.* Journal of global optimization, 2007. 39(3): p. 459-471.
7. Yang, X.-S., *Nature-inspired metaheuristic algorithms*. 2010: Luniver press.
8. Yang, X.-S. and S. Deb. *Cuckoo search via Lévy flights*. in *Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on*. 2009. IEEE.
9. Simon, D., *Biogeography-based optimization.* Evolutionary Computation, IEEE Transactions on, 2008. 12(6): p. 702-713.
10. Li, X., J. Zhang, and M. Yin, *Animal migration optimization: an optimization algorithm inspired by animal migration behavior.* Neural Computing and Applications, 2014. 24(7-8): p. 1867-1877.
11. Meng, X., et al., *A new bio-inspired algorithm: chicken swarm optimization*, in *Advances in swarm intelligence*. 2014, Springer. p. 86-94.
12. Mirjalili, S., S.M. Mirjalili, and A. Lewis, *Grey wolf optimizer.* Advances in Engineering Software, 2014. 69: p. 46-61.
13. Gandomi, A.H. and A.H. Alavi, *Krill herd: a new bio-inspired optimization algorithm.* Communications in Nonlinear Science and Numerical Simulation, 2012. 17(12): p. 4831-4845.
14. Wang, G.-G., S. Deb, and Z. Cui, *Monarch butterfly optimization.* Neural Computing and Applications, 2015: p. 1-20.
15. Yang, X.-S., *A new metaheuristic bat-inspired algorithm*, in *Nature inspired cooperative strategies for optimization (NICSO 2010)*. 2010, Springer. p. 65-74.
16. Kirkpatrick, S. and M.P. Vecchi, *Optimization by simmulated annealing.* science, 1983. 220(4598): p. 671-680.
17. Gandomi, Amir Hossein, Xin-She Yang, SiamakTalatahari, and Amir Hossein Alavi, eds. Metaheuristic applications in structures and infrastructures. Newnes, 2013.
18. Črepinšek, Matej, Shih-Hsi Liu, and Marjan Mernik. "Exploration and exploitation in evolutionary algorithms: A survey." ACM Computing Surveys (CSUR) 45, no. 3 (2013): 35.
19. Ghanem, Waheed Ali HM, and Aman Jantan. "Novel Multi-Objective Artificial Bee Colony Optimization for Wrapper Based Feature Selection in Intrusion Detection." International Journal of Advances in Soft Computing & Its Applications 8, no. 1 (2016).
20. Karaboga, Dervis, and Celal Ozturk. "Neural networks training by artificial bee colony algorithm on pattern classification." Neural Network World 19, no. 3 (2009): 279.
21. Ghanem, Waheed Ali HM, and Aman Jantan. "using hybrid artificial bee colony algorithm and particle swarm optimization for training feed-forward neural networks." Journal of Theoretical and Applied Information Technology 67, no. 3 (2014).
22. Bolaji, A. L. A., khader, A. T., Al-Betar, M. A., & Awadallah, M. A. (2013). Artificial bee colony algorithm, its variants and applications: A survey. Journal of Theoretical & Applied Information Technology, 47(2).