2005 IEEE/PES Transmission and Distribution
Conference & Exhibition: Asia and Pacific
Dalian, China

A Novel Back-propagation Neural Network Training Algorithm Designed by an Ant Colony Optimization

Jeng-Bin Li and Yun-Kung Chung

***Abstract*-- This article presents a new back-propagation neural network (BPN) training algorithm performed with an ant colony optimization (ACO) to get the optimal connection weights of the BPN. The concentration of pheromone laid by the artificial ants moving on the connection path is the key factor of the optimal weight determination. This is the metaphor that the optimal weights make the "total length of neuron-to-neuron connections traversed by all artificial ants," which is defined by the BPN output values and the target values of the proposed ant-based BPN, be the shortest. The generalization ability of the proposed ant-based BPN was checked by means of approximating two functions, a statistic standard normal distribution N(0, 1) and a Sin(x) function, and demonstrated by comparing it to that of a standard BPN. Surprisingly, the superiority of the ant-based BPN's generalization ability in terms of both the statistical determination coefficient and the means square error is beyond the expectation.**

***Index Terms*-- artificial neural network, ant colony optimization, back-propagation network, curve fitting, learning algorithm, function approximation, swarm intelligence.**

I. Introduction

AN artificial neural network (ANN) has been acknowledged as an intelligent universal mechanism of dealing with function approximation, pattern recognition, process estimation and prediction, optimization design and other real applications. Back-propagation network (BPN) is one of such ANN mechanisms. Essentially, the ANN's universality of solving the above problems origins from its capability of learning objects and features existed in a known or an unknown circumstance. The known circumstance results in the necessity of a supervised learning; the unknown one requires an unsupervised learning. BPN is a kind of a supervised ANN, which will be investigated in this paper.

Normally, a gradient descent optimization technique is used to train BPN in which the mean square errors (MSE) between the values output from the neurons in BPN output layer and the target values existed in the known circumstance are minimized by iteratively adjusting the connection strength (or memory) in the form of weights. This standard BPN training technique, however, has the problem of trapping into a pseudo-optimal point on the MSE surface, where the point could make BPN be insufficient for its accurate learning in a practical application. Certain efforts have been made to conquer this problem by applying non-differential optimization techniques, e.g., genetic algorithm (GA), tabu search (TS) and simulated annealing (SA). Application of the technique to the BPN learning is a mushrooming research area. The latest representative works along this respect were contributed by [1] – [7].

Ant colony optimization (ACO) is one of the above non-differential optimization techniques. In natural environment, a colony of real ants holds their swarm intelligence and ability to find the shortest route between their colony and a food source. The swarm intelligence will be applied to find and optimize a route length in a graph or network. Dorigo and Gambardella [8] and Dorigo, Maniezzo and Colorni [9] first quantified this shortest route discovery behavior of real ants and presented an essential foundation for developing an ACO that has been successfully applied to a number of benchmark combinatorial optimization problems, such as the traveling salesman and quadratic assignment problems; meanwhile, they also illustrated that the ACO's solvability outperforms GA's. This truly enables ACO to be applied to a range of combinatorial optimization problems, provided problem specific formulations can be developed.

In this paper, ACO attempts to tackle the specific pseudo-optimization problem in a standard BPN training. The pseudo-optimal points can be avoidable because ACO is with the following facts occurred in its global search: (1) multi-starting points which contain possible initialized feasible points which possibly get to a (multiple) global optimal solution(s), (2) probabilistic mechanism which highly possibly makes converging points jump out of the sub-optimization trap, and (3) natural (non-differential) decent convergence which progresses along with multiple non-gradient asymptotic paths to the optimal points on the high-dimensional discontinuous weight surface. Also, these facts can result in a greater function approximation and system generalization capabilities thus further facilitate practical applications.

This work was supported in part by the National Science Foudation, Taiwan, under NSC90-2212-E-155-013.

Jeng-Bin Li is a graduate student with the Industrial Engineering Department, Yuan-Ze University, Chung Li, Taiwan.

Yun-Kung Chung is an associate professor with the Industrial Engineering Department, Yuan-Ze University, Chung Li, Taiwan. (e-mail: ieychung@saturn.yzu.edu.tw).

0-7803-9114-4/05/$20.00 ©2005 IEEE.

II. Back-propagation Neural Network

As presented schematically in Fig. 1, a typical BPN consists of a number of neurons organized into three layers. The first layer is called an input layer with one neuron for each variable or feature X_i of a pattern or a vector (X_1, X_2, …, X_n) in a training data set in the form of pairs of (X_i, $\hat{Y}_{N_o}$); $\hat{Y}_{N_o}$ is the external known feature used as a target value to supervise the BPN output response Y_{N_o} for monitoring whether the BPN learning process is optimal. Similarly, the third layer functions as an output layer consisting of neurons in which each Y_{N_o} represents for a specified feature meaning to be analyzed. In between, there is a hidden layer also formed by a number of neurons accountable for activating and learning the input features. Neurons in one layer are fully connected to neurons of a succeeding layer. The BPN propagates its input vectors, which can be any multivariate data series, to its output layer through the hidden layer, so that each connection strength quantified with a real number called a weight ($\boldsymbol{w}$) can be adapted (trained) to be optimal value by means of minimizing the MSE between Y_{N_o} and $\hat{Y}_{N_o}$; thus, the weights are thought of as the BPN's memorization of the input vectors.

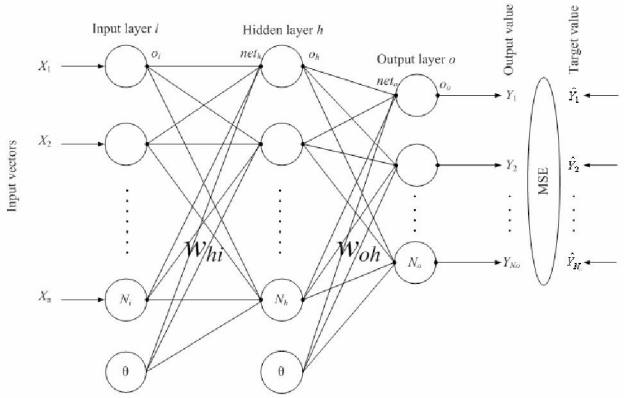


Fig. 1 A standard BPN scheme

Mathematically, net_h , the input to neuron h is a collected signal come from each input neuron i, is given by:

$$net_h = \sum_{i=1}^{N_i} w_{hi} o_i \quad , h = 1, 2, ..., \quad N_h \tag{1}$$

where w_{hi} is the weight associated with the connection from input neuron i to hidden neuron h; o_i is the input-layer's output value which is the same as its own input feature X_i. Unlike o_i, o_h, the neuron output of the hidden layer, is determined by both the activation function f and net_h which is the collected signal of neuron h. One common f is the sigmoid as follows:

$$o_h = f(net_h) = \frac{1}{1 + \exp[-(net_h + \theta)]} \tag{2}$$

where θ is a bias term or threshold value of neuron h responsible for accommodating non-zero offsets in the data. The computations of the input values net_o and output values o_o of the neurons in the output layer are similar with (1) and (2), respectively.

Once the output value o_o (=Y_o) estimated by the BPN has been obtained, the MSE can be calculated by:

$$\mathrm{MSE}(\boldsymbol{w}) = \sum_{o=1}^{No} \left(Y_o - \hat{Y}_o\right)^2 \tag{3}$$

Equation (3), the objective function of the training algorithm, is used to find a set of optimal weights (and biases) that permit the BPN to estimate its output values ($\boldsymbol{Y}$) as close as possible to the external known targets ($\hat{\boldsymbol{Y}}$), in order to minimize their error function MSE($\boldsymbol{w}$). The adaptation of the weights completes when the MSE($\boldsymbol{w}$) satisfies the previously regulated convergence criteria. At this point, the BPN is considered trained, and then the trained BPN may be used to evaluate its generalization or solution capability (solvability) by using other un-training sets, known as the validation set. Detailed BPN development can be found in [10] - [12].

III. Ant-based BPN Training Algorithm

In this section, the useful ant cooperative behavior is used to contribute the basis for developing an improvably supervised BPN training algorithm. Equation (3) is the "ant route distance" which is ACO's objective function to be minimized. To avoid an ant repeatedly passes the same place, each ant k has its own $tabu^k(t)$ at time t used to memorize where the ant has gone. The places stored in $tabu^k(t)$ will not go to again, until each ant finished with going through the entire route once. Two important parameters controlling an ant's intensively moving to a place are the ant's visibility η and the pheromone concentration τ. Detailed ACO definitions and mathematics will not be introduced here due to the limited paper length requirement. Readers interested in ACO can find its mathematic algorithm in [8], [9], [13] and [14]. The proposed ant-based BPN training algorithm simply is mathematically formulated as follows.

Step 1. Initialize m ants, random pheromone (weight) τ_0 for each connection (i, h) and (h, o), $N_i \times N_h \times N_o$ neurons (see Fig. 1), $tabu^k(t)$, four scaling convergence parameters δ , ρ , α and β, and $t = 0$;
Place the m ant at the N_i neurons uniformly;
Input training vectors $\boldsymbol{X}$ used for each ant's carrying;
Set target vectors $\hat{\boldsymbol{Y}}$ used for supervising the ant-based BPN's output values;
For every connection (i, h) and (h, o), set an initial value $\tau_{ij}(t)$ (weight).

Step 2. Set $e = 1$; // the first element in $tabu^k(t)$ //
For k =1 to m do

Place the starting connection of the k^{th} ant in $tabu^k(t)$

Set $\eta_{ih}(t) = o_i$.

Step 3. Repeat until $tabu^k(t)$ is full.

Set $e = e + 1$;

For input neurons do

$$P_{ij}^k(t) = \begin{cases} \dfrac{[\tau_{ij}(t)]^\alpha [\eta_{ij}(t)]^\beta}{\sum_{h \notin Tabu^k(t)} [\tau_{ih}(t)]^\alpha [\eta_{ih}(t)]^\beta} & , j \notin Tabu^k(t), k = 1,2,...,m \\ 0 & , \text{otherwise} \end{cases};$$

Move the k^{th} ant to the connection (i, h);

Save connection (i, h) into $tabu^k(t)$;

For hidden neurons do

the similar activities as those done on the previous connection (i, h).

Step 4. For output neurons do

For $k = 1$ to m do

Compute MSE^k of the "route" described by k^{th} ant;

Sum up MSE^k;

IF the MSE (sum of MSE^k) is unchanged THEN

Save the optimal pheromone trial value of each connection as the optimal weights.

Go to Step 7.

Step 5. Update (Adapt) the pheromone trail of each connection (i, h) and (h, o).

Set $Q = 1$;

For $k = 1$ to m do

$$\Delta\tau_{ih}^k = \begin{cases} 1/MSE^k & \text{if ant } k \text{ uses connection } (i, h) \text{ on its route between time } t \text{ and } t+1 \\ 0 & \text{otherwise} \end{cases}$$

$$\Delta\tau_{ih} = \sum_{k=1}^{m} \Delta\tau_{ih}^k$$

Step 6. For each connection (i, h)

Compute $\tau_{ih}(t+1) = \rho \cdot \tau_{ih}(t) + (1-\rho)\Delta\tau_{ih}$

and do the same computation for each connection (h, o).

Reset $t = t + 1$;

Empty each $tabu^k(t)$.

Go to Step 2.

Step 7. Stop the ant-based BPN training.

IV. Experimental Study

Before checking the solvability of an ANN, some exercises should be performed in order to determine a better set of its training parameters; nevertheless, the sensitivity of the proposed ant-based BPN with respect to its parameters is not be investigated here but it will be done for future practical application. Two checking data sets were randomly generated from a standard normal distribution N(0, 1) and a function Sin(x) respectively, and were used to check the ant-based BPN's solvability by comparing it to the standard BPN's. Each of the both data sets was divided into two subsets, one for both BPNs' learning and another (unlearning data) for validating solvability of the two learned BPN generalizations. The ant-based BPN was designed with Matlab tool and the commercial software NeuralWare was used to run the standard BPN. Figure 2 shows the computer screen used to set the ant-based BPN parameters.

Fig. 2. The computer screen of setting ant-based BPN training parameters (in Chinese.)

The both kinds of BPN training began with initializing all the connection weights (pheromones) to random values between 0.1 and 0.2. Table I indicates the parameter setting and the experimental statistic results of the both BPNs. In the table, for the experiment of N(0,1) distribution curve fitting done with the ant-based BPN, (3-6-1) means the ant-based BPN's topological structure is with 3 input, 6 hidden and 1 output neurons, whilst (0.1, 0.2, 0.3, 0.5) means the corresponding values to (δ, ρ, α and β) parameters; but for the standard BPN in the table, (0.1, 0.01) represents for its value of setting (learning rate, moment rate). These meanings of the initialized parameters also are same used in the second comparison experiment for the Sin(x) function approximation.

The statistic performance comparisons of the two BPNs are also summarized in Table I. For the both generalization errors of the learned and unlearned data in the N(0, 1) fitting experiment, the ant-based BPN shows its superiority. Moreover, the generalization accuracy of unlearned data is lower than that of the learned data for the both BPNs, which means the solvability of the both BPNs has still been improved.

TABLE I
Statistical experimental result of comparing ABN to BPN

Model		ABN	BPN
Experimental result of normal distribution	Initialization	(3-6-1), (0.1, 0.2, 0.3, 0.5)	(3-6-1), (0.1, 0.01)
	Generalization error of learned data	0.09	0.21
	Generalization error of unlearned data	0.13	0.51
	R^2 of learned data	0.8898	0.8171
	R^2 of unlearned data	0.8861	0.7714
Experimental result of Function Sin(x) curve	Initialization	(2-10-1), (0.3, 0.5, 0.2, 0.1)	(2-10-1) (0.1, 0.01)
	Generalization error of learned data	0.11	0.36
	Generalization error of unlearned data	0.26	0.52
	R^2 of learned data	0.8403	0.8509
	R^2 of unlearned data	0.8488	0.8093

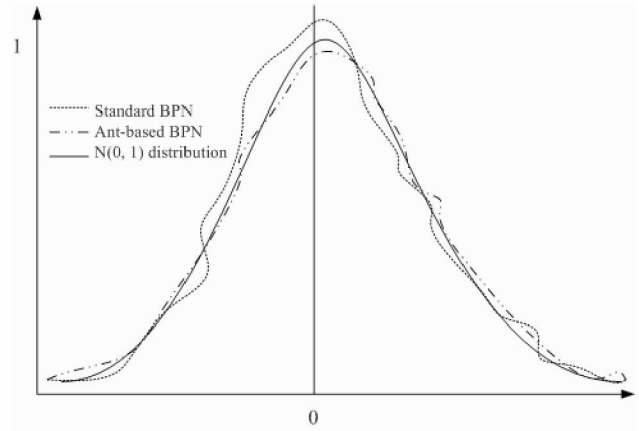


Fig. 3. Comparison of the trained ant-based BPN to the standard BPN for fitting the unlearned data of N(0, 1) distribution.

Another criterion commonly used to illustrate the adequacy of a fitted model is the determination coefficient, R^2. This quantity indicates what of the total variation in the generalized values the fitted model explains. Often, an experimenter will report R^2 and interpret the fitted result as percentage variation explained by the postulated model. Table I shows R^2 of the two fitted BPN models in the two experiments. The larger the coefficient is, the lower variation the BPN has. For the ant-based BPN in each of the both experiments, its R^2 of the learned and unlearned data fitting can be said the same, but this does not seem for the standard BPN, which its R^2 of the unlearned data fitting is worse than that of the learned data fitting. Paralleling the ant-based BPN with the standard one, the variation of the data fitted into the ant-based BPN is lower than that of standard BPN.

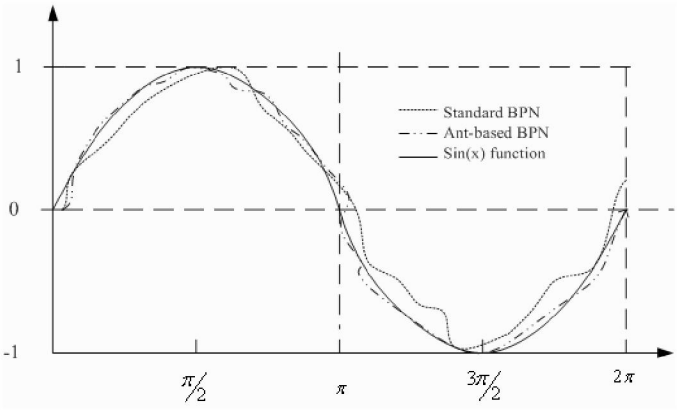
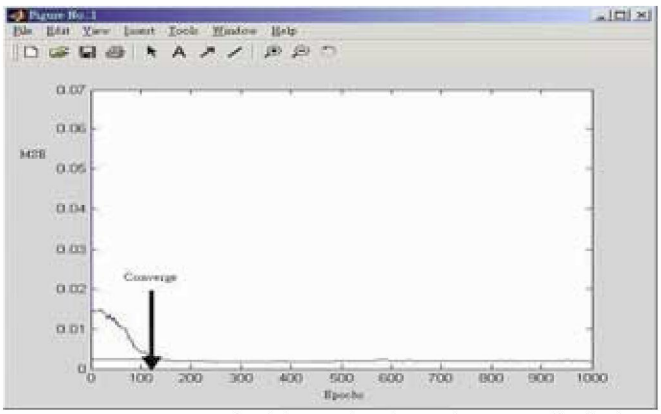
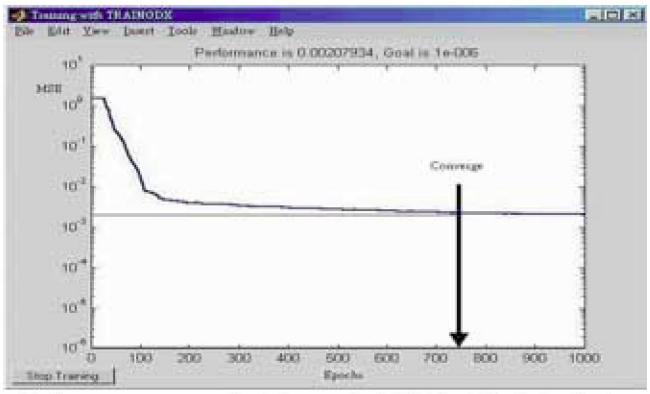


Fig. 4. Comparison of the trained ant-based BPN to the standard BPN for fitting the unlearned data of a Sin(x) curve



(a) Convergence curve of training ant-based BPN for N(0, 1) distribution



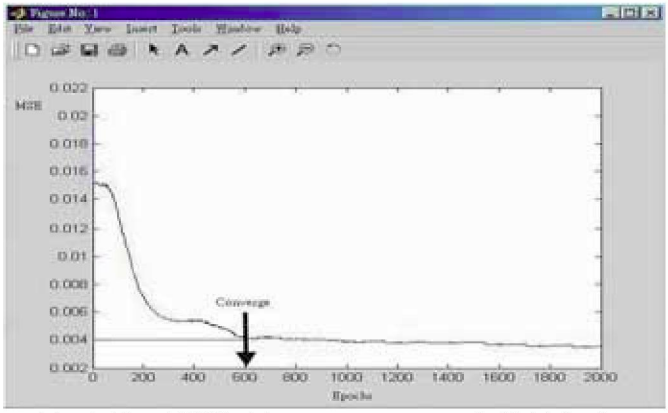
(b) Convergence curve of training standard BPN for N(0, 1) distribution

Fig. 5. Convergence curve of training the two kinds of BPN with N(0,1) distribution

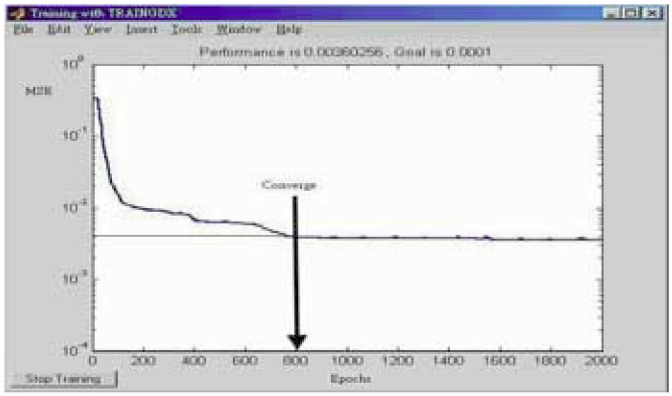
Fig. 3 expresses the solvability of both the trained ant-based BPN and the trained standard BPN used for fitting unlearned data of N(0, 1), and the figure for comparing the unlearning data used to fit Sin(x) function is shown in Fig. 4. These two figures show the similar compared conclusion with

each other; namely, the function approximation ability of the ant-based BPN is better than that of the standard BPN for the two functions. The figures drawn with the trained data used to approximate the same two functions are not shown here because their comparisons are similar with those of the untrained data fitting the functions.

The convergence trend of training both the ant-based BPN and the standard BPN with N(0, 1) distribution is shown in Fig. 5. Apparently, the ant-based BPN training time in terms of epochs is much shorter than the standard one. The number of epochs of the ant-based BPN training was around 120 times, but the standard BPN needed about 750 times. For the Sin(x) function training, the ant-based BPN also was obviously faster than the standard BPN, 600 epochs versus 800 epochs, as shown in Fig. 6.



(a) Ant-based BPN training convergence curve of a Sin(x) function



(b) Standard BPN training convergence curve of a Sin(x) function

Fig. 6. Convergence curve of training the two kinds of BPN with a Sin(x) function

V. Conclusions

This paper used a standard ACO to find the shortest route in terms of the distance (MSE) between the output values and the target values of BPN to present a novel full-connected BPN training algorithm. The proposed ant-based BPN experimented the BPN training time (epochs) requirement and its ants' pheromones provided the final optimal weights used to check the solvability of the both learned and unlearned data generalization (i.e., function approximation). The results of paralleling the ant-based BPN with the standard BPN showed that the ant-based BPN was successful to achieve: (1) to shorten the BPN training time, (2) to reduce the BPN calculating complexity, and (3) to increase the BPN data generalization accuracy.

Even though this paper experimented only two functions for the empirical generalization comparison, it can be seen that by taking advantage of the ACO global search for BPN training, many if not all problems concerned with BPN can be conquered. More practical results did not be reported yet, but the ant-based BPN indeed motivates us to pursue this path. Improvements on the ant-based BPN convergence are still possible as well as a comparative study with the more other ANN trainings to understand the benefits of the ant-based BPN has to be done in the future.

VI. References

[1] D. T. Pham and D. Karaboga, *Intelligent optimization techniques, genetic algorithms, tabu search, simulated annealing, and neural networks*, Berlin, Springer, 2000.

[2] Salcedo-Sanz, Sancho; Bousoño-Calzón, Carlos, "A portable and scalable algorithm for a class of constrained combinatorial optimization problems," *Computers and Operations Research*, Vol. 32, pp. 2671-2687, Oct. 2005.

[3] K., Adem and K. Dervis, "Training recurrent neural networks by using parallel tabu search algorithm based on crossover operation," *Engineering Applications of Artificial Intelligence*, Vol. 17, pp. 529-542, Aug., 2004.

[4] T. Tambouratzis, "A simulated annealing artificial neural network implementation of the n-queens problem," *International Journal of Intelligent Systems*, Vol.12, pp. 739-751, Oct. 1997.

[5] Kim, K-J. and Cho, S-B., "Prediction of colon cancer using an evolutionary neural network," *Neurocomputing*, Vol.61, pp. 361-379, Oct. 2004.

[6] Sexton, R. S., Alidaee, B., Dorsey, R. E. and Johnson, J. D., "Global optimization for artificial neural networks: A tabu search application," *European Journal of Operational Research*, Vol.106, pp. 570-584, April, 1998.

[7] Karaboga, D.and Kalinli, A. "Training recurrent neural networks for dynamic system identification using parallel tabu search algorithm," in *Proceedings of the 1997 IEEE International Symposium on Intelligent Control*, pp.113 – 118.

[8] Dorigo, M., and Gambardella, L. M. "Ant colonies for the traveling salesman problem," *BioSystems,* Vol. 43, pp. 73–8, 1997.

[9] Dorigo, M., Maniezzo, V., and Colorni, A., "The ant system: optimization by a colony of cooperating ants," *IEEE Trans. Syst. Man Cybern.,* Vol.26, pp.29–42, 1996.

[10] Christopher M. Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press, 2004.

[11] Simon Haykin, *Neural Networks: A Comprehensive Foundation*, 2nd Ed., Macmillan College Publishing, New York, 2001.

[12] Martí, R. and El-Fallahi, Abdellah, "Multilayer neural networks: an experimental evaluation of on-line training methods," *Computers and Operations Research*, Vol. 31, pp. 1491-1513, August, 2004.

[13] Stutzle, T. and Hoos, H. H., "MAX-MIN ant system," *Future Generation Comput. Systems, Vol.*16, pp.889–914, 2000.

[14] H.M. Botee and E. Bonabeau, "Evolving ant colony optimization", *Advances in Complex Systems,* vol. 1, no. 2/3, pp.149-159, 1999.