Evolutionary Computation Applied to the Automatic Design of Artificial Neural Networks and Associative Memories

Humberto Sossa, Beatriz A. Garro, Juan Villegas, Gustavo Olague, and Carlos Avilés

Abstract. In this paper we describe how evolutionary computation can be used to automatically design artificial neural networks (ANNs) and associative memories (AMs). In the case of ANNs, Particle Swarm Optimization (PSO), Differential Evolution (DE), and Artificial Bee Colony (ABC) algorithms are used, while Genetic Programming is adopted for AMs. The derived ANNs and AMs are tested with several examples of well-known databases.

1 Introduction

If we want that a machine efficiently interacts with its environment, it is necessary that the so called pattern recognition problem is appropriately solved. Lots of approaches to face this problem have been reported in literature. One of the most popular one is the artificial neural network based approach. It consists on combining the individual capacities of many small processors (programs) in such a way that a set of patterns under study is correctly classified or restored.

An artificial neural network (ANN) can be seen as a set of highly interconnected processors. The processors can be electronic devices or computer programs. From

Humberto Sossa · Beatriz A. Garro
CIC-IPN, Juan de Dios Batiz S/N, Col. Nva. Industrial Vallejo, Mexico City, Mexico
e-mail: `hsossa@cic.ipn.mx`, `beatriz.auroragl@gmail.com`

Juan Villegas · Carlos Avilés
UAM-Azcapotzalco, Av. San Pablo Xalpa 180. Azcapotzalco, Mexico City, Mexico
e-mail: `jvillegas@gmail.com`, `caviles@correo.azc.uam.mx`

Gustavo Olague
CICESE, Carretera Ensenada-Tijuana 3918Zona Playitas, Ensenada, B. C., Mexico
e-mail: `gustavo.olague@gmail.com`

O. Schütze et al. (Eds.): EVOLVE – A Bridge between Probability, AISC 175, pp. 285–297.
springerlink.com

now on, these processors will be called nodes or units. These units can be the nodes of a graph. The edges of this graph determine the interconnections among the nodes. These represent the synaptic connections between the nodes, and are supposed to be similar to the synaptic connections between biological neurons of a brain.

Associative memories, in the other hand, are special cases of ANNs. They have several interesting properties that make them preferable than ANNs, for some problems.

In this paper, we briefly describe how bio-inspired and evolutionary based techniques can be efficiently used for the automatic design of ANNs and AMs. The rest of the paper is organized as follows. Section 2 is focused to explain the generalities of ANNs and AMs. Section 3 is oriented to explain the generalities about how three bio-inspired techniques: Particle Swarm Intelligence (PSO), Differential Evolution (DE) and Artificial Bee Colony (ABC) have been used with success in the design of ANNs to classify patterns. Section 4, in the other hand, is dedicated to provide the details of how Genetic Programming can be used to synthesize AMs for pattern classification as well as pattern restoration. Section 5 is devoted to present some of the obtained results. A discussion of the results is also given in this section. Finally, Section 6 is oriented for the conclusions and directions for further research.

2 Basics on Artificial Neural Networks and Associative Memories

In this section we present the most relevant concepts and definitions concerning artificial neural networks.

2.1 Basics on Artificial Neural Networks (ANNs)

An ANN is an interconnected set of simple processing elements, units or nodes, whose functionality is vaguely based on the animal neuron. The processing ability of the net is stoked in the connections (weights) among the units. These values are obtained by means of an adapting or learning process from a learning set [7].

An ANN performs a mapping between the input vector X and an output vector Y, by the consecutive application of two operations. The first operation computes at each node the alignment between the input vector X and the auxiliary weighting vector W. The second operation takes the result of the first operation and computes the mapping. The two equations that govern the functionality of an individual processing unit j net without bias b_j are the following:

$$a = \sum_{i=1}^{n} w_i x_i, \tag{1}$$

$$y_j = f(a), \tag{2}$$

where the x_i are the inputs to the neuron and the w_i are its synaptic weights. In matrix form:

$$a = W \cdot X, \tag{3}$$

where now $W = (w_1, w_2, \ldots, w_n)^T$ and $X = (x_1, x_2, \ldots, x_n)^T$.

As we can see the neuron performs the dot product between the weight vector W and the input vector X. The output function $f(a)$ of the neuron is usually non-linear. In the case of the threshold logic unit proposed by McCulloch and Pitts [15] is the hard limit function or in the case of the Perceptron [18] is the sigmoid function [7], [19].

The way the set of neurons are interconnected determines the ANN architecture. The neurons in an ANN can be connected feedforward, sometimes they can admit side connections, even feedback.

Three elements characterize the functionality of an ANN: its architecture (the way its nodes are interconnected), the values of its weights, and the transfer functions that determine the kind of output of the net.

In Section 3 we will see how for a given ANN to automatically select each of these components, and this by means of bio-inspired techniques.

2.2 Basics on Associative Memories (AMs)

An AM is a mapping used to associate patterns from two different spaces. Mathematically, an AM, M is a mapping that allows restoring or recalling a pattern y^k, $k = 1, \ldots, p$, given an input pattern x^k, $k = 1, \ldots, p$. In general y^k is of dimension m, while x^k is of dimension n.

Both x^k and y^k can be seen as vectors as follows: $x^k = (x_1^k, \ldots, x_n^k)^T$ and $y^k = (y_1^k, \ldots, y_n^k)^T$. Thus:

$$x^k \to M \to y^k. \tag{4}$$

If for all k, $x^k = y^k$, the memory operates in auto-associative way, otherwise it works as a hetero-associative operator. For each k, $(x^k, y^k)_{k=1}^p$ is called an association. The whole set of associations is called the fundamental set of associations.

Examples of AMs are the Linear Associator (LA) [1]-[10], the Lernmatrix (LM) [20], and the morphological associative memory (MAM) [17]. Both the LA and the LM operate in the hetero-associative way, while the MAM can operate in auto and hetero-associative fashions. The LA and the LM operate with binary-valued vectors. MAMs can operate both with binary or real-valued vectors.

To operate an AM two phases are required, one of construction or designing and one of testing or retrieval.

Generally, in the design of an AM two operators are required: an internal operator and an external operator. Internal operator O_I is used to derive a partial codification of the set of patterns. It acts on each association: $(x^k, y^k)_{k=1}^p$. It gives, as a result a part of the AM. External operator O_E, on the other hand, combines the partial results obtained by O_I, and gives, as a result, the total mapping M.

As an illustrative example, let us take the case of the LA. In a first step, the LA takes each association $(x^k, y^k)_{k=1}^p$ and produces the partial codification:

$$M^k = y^k (x^k)^T \tag{5}$$

It then takes the k partial matrices and produces the final mapping:

$$M = M^1 + \ldots + M^p = \sum_{k=1}^{p} y^k (x^k)^T. \tag{6}$$

As can be seen from this example, the following two operations are needed to get the LA: a product between each two vectors: y^k and x^k, to get matrix M^k, and a sum between matrices M^k to the final M.

Recalling of a given pattern y^k, through a designed M is given as follows:

$$y^k = M \cdot x^k. \tag{7}$$

In this case, recalling demands only one operation, a multiplication between the LA and the input vector.

Necessary conditions for correct recall of each y^k is that the all the x^k are orthonormal. This is a very restrictive condition but it allows visualizing the necessary operations to operate the memory.

3 Automatic Synthesis of ANNs

The designing of an ANN normally involves 1) the automatic adjustment of the synaptic weights between the different neurons of the ANN, 2) the selection of the corresponding architecture of the ANN, and 3) the selection also of the transfer function of the neurons is also, sometimes, a matter.

Several methods to adjust the weights of the ANN, once its architecture has been selected, can be found in the literature. Probably, the most known in the case of arrangement of Perceptrons is the back-propagation rule (BP) [19]. It is based on gradient decent principle and, if no convenient actions are taken into account BP, generally falls into a local minimum providing a non optimal solution. Since the point of view of pattern classification this could be interpreted as a deficient learning of the ANN, and/or a bad generalization capacity.

Since several years many scientists have used evolutionary and bio-inspired techniques to evolve: 1) the synaptic weights of the ANN, 2) its architecture, or 3) both the synaptic weights and its architecture. For a good review on the subject refer, for example, to [28]-[29]. In [2], [3], [4], [5] and [6], the authors show how bio-inspired techniques such as PSO, DE and ABC can be used to automatically select the architecture of and ANN, tune its weights and even to chose the transfer function for each neuron. In this section we give the generalities of this proposal. Related work concerning the training of spiking neurons by means bio-inspired techniques can be found in [22], [23], [24] and [25].

3.1 PSO, DE and ABC

PSO, DE and ABC are examples of searching techniques to solve optimization problems with many optima where most standard methods will fail. Generally speaking, a bio-inspired technique is a method inspired in a metaphor of nature that takes into account the partial but powerful abilities of each of the individuals to produce a global solution for a difficult problem. A typical problem to solve is searching for food. For example, the individuals of the colony of ants experiment hanger but they do not know where the food is. A subset of the ants goes in all directions inside their territory searching for the precious product. Ants communicate among themselves by so-called pheromones. Once an ant or a group of ants find the desired food, they communicate to the other. The information goes back as a chain to the nest, the collector ants then go for the food. Details about the functioning of PSO, DE and ABC can be found in [9], [21], and [8], respectively.

3.2 Garro's Proposal

The problem to be solved is stated as follows:

Given a set of input patterns $X = \{X^1, \dots, X^p\}$, $X^k \in \mathbb{R}^n$, and a set of desired patterns $D = \{d^1, \dots, d^p\}$, $d^k \in \mathbb{R}^m$, find an ANN represented by a matrix $W \in \mathbb{R}^{q\times(q+1)}$, such that a function defined as $\min(f(X,D,W))$ is minimized. In this case q is the maximum number of neurons MNN; it is defined as $q = 2(m+n)$.

Each individual ANN is codified as a matrix as follows:

$$\begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,NMN+1} & x_{1,NMN+2} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{NMN,1} & x_{NMN,2} & \cdots & x_{NMN,NMN+1} & x_{NMN,NMN+2} \end{bmatrix} \quad (8)$$

The matrix is composed by three parts: The topology (the first column of (1)), the transfer functions (the last column of (1)), and the synaptic weights (the submatrix of (1) without the first and last column).

The aptitude of an individual is computed by means of the MSE function:

$$F1 = \frac{1}{p \cdot m} \sum_{\xi=1}^{p} \sum_{j=1}^{m} (d_j^{\xi} - y_j^{\xi})^2. \tag{9}$$

This way, all the values of matrix W are codified so as to obtain the desired ANN. Moreover, each solution must be tested in order to evaluate its performance. For this, it is necessary to know the classification error (CER), this is to know how many patterns have been correctly classified and how many were incorrectly classified. Based on the winner-take-all technique the CER function can be computed as follows:

$$F2 = 1 - \frac{nwcp}{tnp} \tag{10}$$

In this case, $nwcp$ is the number of well classified patterns and tnp is the total number of patterns to be classified.

Additionally, if we want to minimize the number of connections of the ANN, we would also make use of the following function:

$$F3 = \frac{NC}{NmaxC}. \tag{11}$$

In this case NC is the number of connections of the ANN, while $NmaxC = \sum_{i=n}^{MNN} i$ is the maximum number of connections generated with MNN neurons. When functions $F1$, $F2$ and $F3$ are combined, we get the two functions to be optimized:

$$FF1 = F1 \cdot F2 \tag{12}$$

$$FF2 = F1 \cdot F3 \tag{13}$$

The six transfer functions used by Garro are the logsig (LS), tansig (TS), sin (S), radbas (RD), pureline (PL), and hardlim (HL). These functions were selected for they are the most popular and useful transfer functions in several kinds of problems.

In Section 5, we will see how these two functions can be used to automatically synthesize an ANN for a given classification problem.

4 Automatic Synthesis of AMs by Means of Genetic Programming

Until 2005 all the AMs models found in literature (more or less 30) have been produced by a human user. In 2009, in [26] and [27], the authors arrive to an original solution where, for the first time, they propose a methodology for the automatic synthesis of AMs for pattern restoration. In this section we provide the generalities of this proposal.

4.1 Genetic Programming

Genetic programming (GP) as proposed by J. R. Koza is an evolutionary algorithm-based methodology inspired by biological evolution to find computer programs that perform a user-defined task [11], [12], [13] and [14]. It is a specialization of genetic algorithms (GA) where each individual is a computer program. Therefore, GP is a machine learning technique used to optimize a population of computer programs according to a fitness function determined by a program's ability to perform a given computational task. The idea behind GP is to evolve computer programs represented in memory as tree structures. Basically, the way to modify the threes is carried out in two ways, either by crossing them or mutating them. This way we can evaluate the performance of the trees. At the end of the process we will have the winner tree or winner trees. To generate a solution, GP operates onto two sets (a terminal set, TS, and a function set, FS) and a fitness function FF. At the end of the evolving process GP delivers one or more solutions as programs that solve the problem.

4.2 Villegas' Proposal

We have seen that to operate an AM two operators are required, one for designing the memory and for testing the memory. Let us design these operators as follows: D_O, for designing operator or codifying operator and D_T, for testing operator.

The general idea of the technique proposed by Villegas to automatically synthesize an AM by means of genetic programming is as follows, given a set of associations $(x^k, y^k)_{k=1}^{p}$:

1. Propose a set of initial Q solutions, this a set of couples of operators: $(D_O, D_T)^q$, $q = 1, \ldots, Q$, each one expressed as a tree in terms of the chosen function and terminal sets, F and T.
2. Test the different couples $(D_O, D_T)^q$ with the p associations $(x^k, y^k)_{k=1}^{p}$, and retain the best solutions according the chosen fitness function FF.
3. Evolve the couples.
4. Repeat Steps 2 and 3 until obtaining the set of the best solutions.

The result is a set of several evolved couples (D_O^*, D_T^*) that best satisfy fitness function FF. In the next section we will show examples of obtained couples for several pattern restoration examples.

5 Experimental Results

Here, we present several examples of the ANNs and AMs automatically obtained by the proposals explained in Sections 3 and 4. We provide also a discussion to complete explanation.

5.1 Examples of Synthetically Generated ANNs

The methodology described in Section 3 was applied to several well-known pattern recognition problems. The following pattern classification problems taken from the machine learning benchmark repository UCI were taken [16]: iris plant database, wine database and breast cancer database. Due to space limitations, we only show results concerning the application of ABC technique to the iris plant database. The iris plant database consists of 150 samples, described by four features: 1) length of the sepal, 2) width of the sepal, 3) length of the petal, and 4) width of the petal, all in cm. Ten experiments were performed for each of the three databases.

Figure 1 shows the evolution of the error for functions and for the iris database. Figure 2(a) shows one of the ANNs obtained by the proposed methodology. Figure 2(b) shows one of the ANNs obtained by the proposed methodology taking into account $F3$. Note the reduction of the connections. Figure 3 shows the percentages of recognition for the ten experiments. Note the in all ten experiments, the percentage of recognition maintains high.

Note also how reducing the number of connections does not dramatically affect the performance of the synthesized ANN. From this experiment we can conclude that the proposed methodology provides very promising results.

5.2 Examples of Synthetically Generated AMs

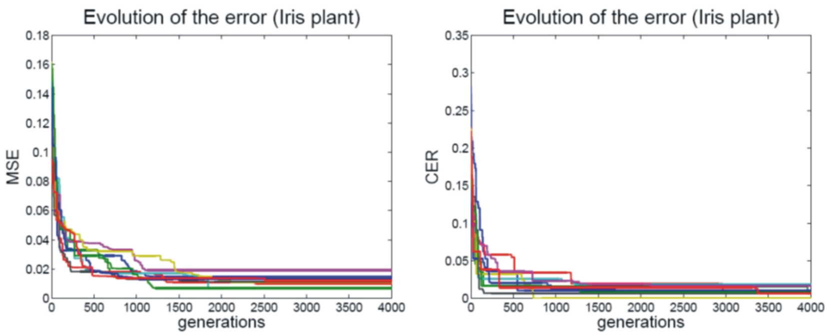
The methodology described in Section 4 was applied to several well-known pattern recognition problems. According to [26] and [27]:

For association:

- $O_{p_k^a}$ is the evolved operator for pattern association.
- M^k is the partial association matrix by applying operator $O_{p_k^a}$ to each association (x^k, y^k).
- M is the associative memory that comes up from the addition of all the M^k.
- $TS_a = \{x^k, y^k\}$ is the set of terminals for association.
- $FS_a = \{+, -, \min, \max, times\}$ is the set of functions for association.

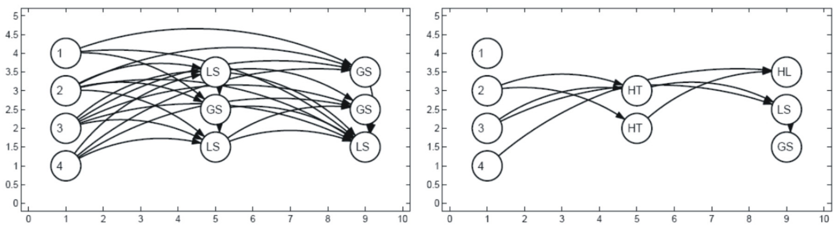
For recalling:

- $O_{p_k^r}$ is the evolved operator for pattern recalling or classification.
- $TS_r = \{v, row1, row2, \ldots, rowm, M^k\}$ is the set of terminals for pattern recalling or classification. v is the input vector, $row\,j$ is the j-th row of matrix M.
- $FS_a = \{+, -, \min, \max, mytimesm\}$ is the set of functions for pattern recalling or classification.
- $mytimesm$ operation produces a vector and is defined as: $mytimesm(x^k, y^k) = [x_1y_1, \ldots, x_ny_n]$.
- $\hat{y}^k$ is the recalled by applying operator $O_{p_k^r}$ to input vector $x(\tilde{x})$. $\tilde{x}$ is a distorted version of x.



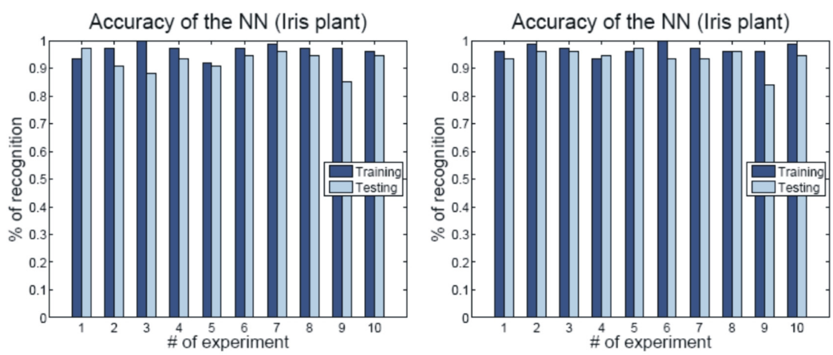
(a) (b)

Fig. 1 Evolution of the error for the ten experiments for the Iris plant problem. (a) Evolution of $FF1$ using MSE function. (b) Evolution of $FF2$ using CER function. Figure taken from [5].



(a) (b)

Fig. 2 Two different ANNs designs for the Iris plant problem. (a) ANN designed by the ABC algorithm without taking into account $F3$ function. (b) ANN designed by the ABC algorithm taking into account $F3$ function. Figure taken from [5].



(a) (b)

Fig. 3 Percentage of recognition for the Iris problem and the ten experiments during the training and testing stage for each fitness function. (a) Percentage of recognition minimizing the $FF1$ function. (b) Percentage of recognition minimizing the $FF2$ function. Figure taken from [5].

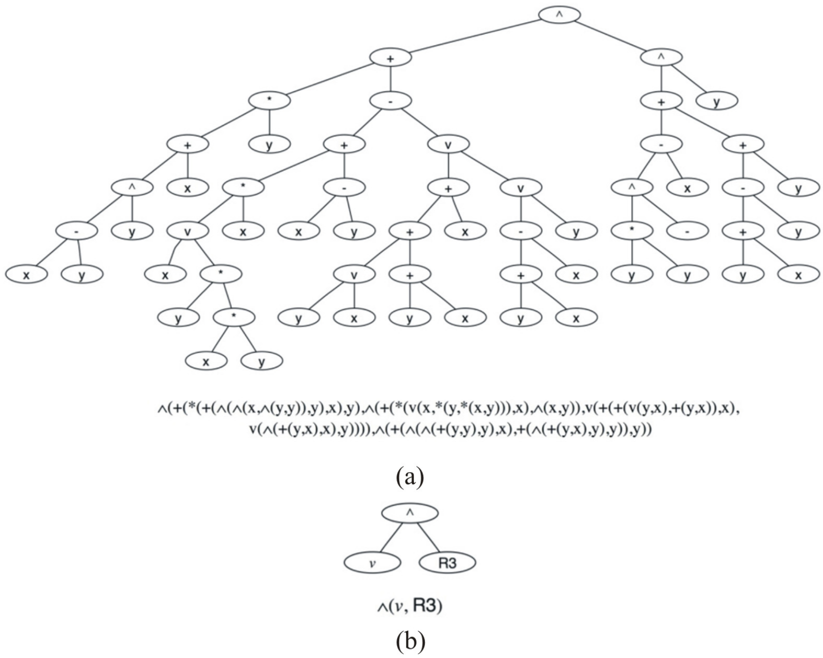


Fig. 4 A couple of association and recalling operators automatically derived by the proposal for the Iris database problem. Taken from [27].

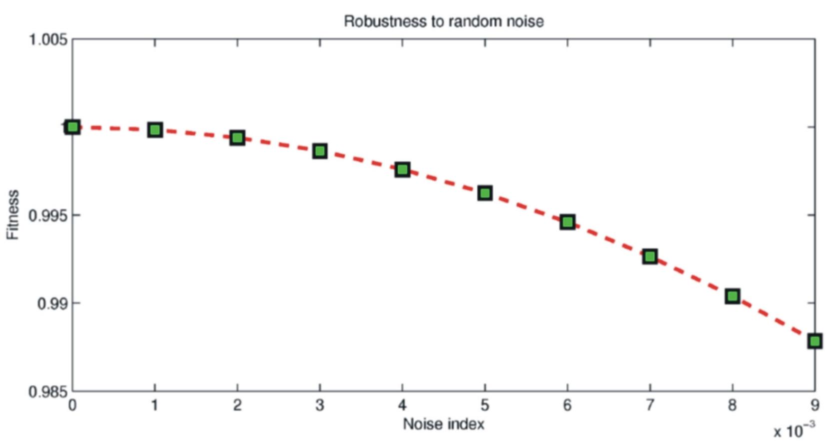


Fig. 5 Original images of the first ten digits and noisy versions of them, used to test the proposal. Taken from [27].

The fitness function used to test the efficiency of the evolved operators is:

$$f = \frac{y\tilde{y}}{\sqrt{yy} \cdot \sqrt{\tilde{y}\tilde{y}}} \quad (14)$$

The proposed methodology was tested with several pattern classification examples: Database of numbers and the same databases used in Section 3. Due to space limitations, only results with the Iris Plant classification database already used in Section 3.

We implemented our model looking for several pairs of association and recalling operators. One of these pairs is shown in Fig. 4. We tested this AM by adding random noise to the input pattern set as shown in Fig 5. In this case the noise was added from 0.01 to 0.09% in steps of 0.01. While the fundamental set was correctly recalled, the recalling rate decreased slowly as noise increased. One can note the complexity of the operator for association, compared with other associations proposed by humans. Note however the simplicity of the recalling operator for this example.

6 Conclusions and Directions for Further Research

We have seen that it is possible to automatically synthesize ANNs and AMs for pattern classification and pattern restoration porpoises. In the case of ANNs we have made used of bio-inspired techniques such as PSO, DE and ABC; while for AMs we have used GP. In both cases we very nice and promising results have been obtained. We have tested the proposed techniques with several reported benchmarks with satisfactory results.

Venues for further research are the following: 1) Automatic design of radial base networks, 2) Automatic design of morphological neural networks. 3) Automatic design of spiking neural networks. 4) Automatic design of bidirectional associative memories, and 5) Simplification of associative memory operators.

Acknowledgements. H. Sossa thanks SIP-IPN and CONACYT for the economical support under grants SIP 20111016, SIP 20121311 and CONACYT 155014 to develop the reported investigations. We also thank CIC-IPN, CICESE and UAM-Azcapotzalco for the support to undertake this research.

References

1. Anderson, J.A.: A simple neural network generating an interactive memory. Mathematical Biosciences 14, 197–220 (1972)
2. Garro, B.A., Sossa, H., Vázquez, R.A.: Design of Artificial Neural Networks using a Modified Particle Swarm Optimization Algorithm. In: International Joint Conference on Neural Networks (IJCNN 2009), Atlanta, GE, USA, June 14-19, pp. 938–945 (2009)

3. Garro, B.A., Sossa, H., Vázquez, R.A.: Design of Artificial Neural Networks Using Differential Evolution Algorithm. In: Wong, K.W., Mendis, B.S.U., Bouzerdoum, A. (eds.) ICONIP 2010, Part II. LNCS, vol. 6444, pp. 201–208. Springer, Heidelberg (2010)
4. Garro, B.A., Sossa, H., Vázquez, R.A.: Evolving Neural Networks: A Comparison between Differential Evolution and Particle Swarm Optimization. In: Tan, Y., Shi, Y., Chai, Y., Wang, G. (eds.) ICSI 2011, Part I. LNCS, vol. 6728, pp. 447–454. Springer, Heidelberg (2011)
5. Garro, B.A., Sossa, H., Vázquez, R.A.: Artificial Neural Network Synthesis by means of Artificial Bee Colony (ABC) Algorithm. In: CEC 2011, New Orleans, June 5-8 (2011)
6. Garro, B.A., Sossa, H., Vázquez, R.A.: Back-Propagation vs Particle Swarm Optimization Algorithm: which Algorithm is better to adjust the Synaptic Weights of a Feed-Forward ANN? International Journal of Artificial Intelligence 7(11), 208–218 (2011)
7. Gurney, K.: An Introduction to Neural Networks. Taylor and Francis Group (1997)
8. Karaboga, D., Basturk, B.: A powerful and efficient algorithm for numerical function optimization: Artificial Bee Colony (ABC) algorithm. Journal of Global Optimization 39(3), 459–471 (2007)
9. Kennedy, J., Eberhart, R.: Particle Swarm Optimization. In: Proceedings of IEEE International Conference on Neural Networks, vol. IV, pp. 1942–1948 (1995)
10. Kohonen, T.: Correlation matrix memories. IEEE Transactions on Computers C-21(4), 353–359 (1972)
11. Koza, J.R.: Genetic Programming: A Paradigm for Genetically Breeding Populations of Computer Programs to Solve Problems, Stanford University Computer Science Department technical report (1990)
12. Koza, J.R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press (1992)
13. Koza, J.R.: Genetic Programming II: Automatic Discovery of Reusable Programs. MIT Press (1994)
14. Koza, J.R., Bennett, F.H., Andre, D., Keane, M.A.: Genetic Programming III: Darwinian Invention and Problem Solving. Morgan Kaufmann (1999)
15. McCulloch, W., Pitts, W.: A logical calculus of the ideas immanent in nervous activity. Bulletin of Mathematical Biophysics 7, 115–133 (1943)
16. Murphy, P.M., Aha, D.W.: UCI Repository of machine learning databases. University of California, Department of Information and Computer Science, Irvine, CA, US., Technical Report (1994)
17. Ritter, G., Dìaz, J.: Morphological associative memories. IEEE Transactions on Neural Networks 9(2), 281–293 (1998)
18. Rosenblatt, F.: The Perceptron-a perceiving and recognizing automaton. Report 85-460-1, Cornell Aeronautical Laboratory (1957)
19. Rojas, R.: Neural networks - A systematic introduction. Chapter 7: The back propagation algorithm (1996)
20. Steinbuch, K.: Die Lernmatrix. Kybernetik 1(1), 36–45 (1961)
21. Storn, R., Price, K.: Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization 11, 341–359 (1997)
22. Vazquez, R.A.: Izhikevich Neuron Model and its Application in Pattern Recognition. Australian Journal of Intelligent Information Processing Systems 11(1), 53–60 (2010)
23. Vázquez, R.A.: Pattern Recognition Using Spiking Neurons and Firing Rates. In: Kuri-Morales, A., Simari, G.R. (eds.) IBERAMIA 2010. LNCS, vol. 6433, pp. 423–432. Springer, Heidelberg (2010)
24. Vazquez, R.A.: A computational approach for modelling the biological olfactory system during an odour discrimination task using spiking neuron. BMC Neuroscience 12(supp.1), 360 (2011)

25. Vázquez, R.A., Garro, B.A.: Training Spiking Neurons by Means of Particle Swarm Optimization. In: Tan, Y., Shi, Y., Chai, Y., Wang, G. (eds.) ICSI 2011, Part I. LNCS, vol. 6728, pp. 242–249. Springer, Heidelberg (2011)
26. Villegas, J., Sossa, H., Avilés, C., Olague, G.: Automatic Synthesis of Associative Memories by genetic Programming, a First Approach. Research in Computing Science 42, 91–102 (2009)
27. Villegas, J., Sossa, H., Avilés, C., Olague, G.: Automatic Synthesis of Associative Memories through Genetic Programming: a co-evolutionary approach. Revista Mexicana de Fsica 57(2), 110–116 (2011)
28. Yao, X.: A review of evolutionary artificial neural networks. Int. J. Intell. Syst. 8(4), 539–567 (1993)
29. Yao, X.: Evolutionary artificial neural networks. In: Kent, A., Williams, J.G. (eds.) Encyclopedia of Computer Science and Technology, vol. 33, pp. 137–170. Marcel Dekker, New York (1995)