Revista Mexicana de Física
ISSN: 0035-001X
rmf@ciencias.unam.mx
Sociedad Mexicana de Física A.C.
México

Villegas-Cortez, J.; Sossa, J.H.; Avilés-Cruz, C.; Olague, G.
Evolutionary Associative Memories Through Genetic Programming
Revista Mexicana de Física, vol. 57, núm. 2, abril, 2011, pp. 110-116
Sociedad Mexicana de Física A.C.
Distrito Federal, México

Available in: http://www.redalyc.org/articulo.oa?id=57019378003

How to cite
Complete issue
More information about this article
Journal's homepage in redalyc.org

Scientific Information System
Network of Scientific Journals from Latin America, the Caribbean, Spain and Portugal
Non-profit academic project, developed under the open access initiative

Evolutionary Associative Memories Through Genetic Programming

J. Villegas-Cortez
Universidad Autónoma Metropolitana - Azcapotzalco, Departamento de Electrónica,
Av. San Pablo 180 Col. Reynosa, México, D.F., 02200, México,
e-mail: juanvc@correo.azc.uam.mx

J.H. Sossa
Centro de Investigación en Computación del Instituto Politécnico Nacional,
Av. Juan de Dios Bátiz, esquina con Miguel Othón de Mendizábal,
Col. Nueva Industrial Vallejo. México D.F. 07738, México,
e-mail: hsossa@cic.ipn.mx

C. Avilés-Cruz
Universidad Autónoma Metropolitana - Azcapotzalco, Departamento de Electrónica,
Av. San Pablo 180 Col. Reynosa, México D.F., 02200, México.

G. Olague
Centro de Investiación Científica y de Educación Superior de Ensenada CICESE,
Km. 107 Carretera Tijuana-Ensenada, Ensenada, 22860, B.C. México.

Recibido el 23 de octubre de 2009; aceptado el 10 de enero de 2010

Associative Memories (AMs) are useful devices designed to recall output patterns from input patterns. Each input-output pair forms an association. Thus, AMs store associations among pairs of patterns. An important feature is that since its origins AMs have been manually designed. This way, during the last 50 years about 26 different models and variations have been reported. In this paper, we illustrate how new models of AMs can be automatically generated through Genetic Programming (GP) based methodology. In particular, GP provides a way to successfully facilitate the search for an AM in the form of a computer program. The efficiency of the proposal was conducted by means of two tests based on binary and real-valued patterns. The experimental results show that it is possible to automatically generate AMs that achieve good results for the selected pattern recognition problems. This opens a new research area that allows, for the first time, synthesizing new AMs to solve specific problems.

Keywords: Computer science and technology; neural engineering; image quality; contrast; resolution; noise; image analysis.

Las memorias asociativas (AMs) son estructuras matemáticas específicamente diseñadas para recuperar patrones de entrada con patrones de salida. Cada par asociado (entrada-salida) forma una asociación, es así que la AM almacena las asociaciones entre los pares. Desde sus orígenes las AMs han sido diseñadas manualmente, y durante los últimos 50 años se han reportado un aproximado de 26 modelos de AMs con sus variantes. En este trabajo mostramos un nuevo modelo de AMs que es generado de forma automática por medio de Programación Genética. Este trabajo abre una nueva área de investigación que permite por primera vez sintetizar nuevas AMs para resolver problemas específicos. Para probar la eficiencia de nuestra propuesta la hemos aplicado para los casos de patrones en valores binarios y reales. Los experimentos muestran que es posible la generación automática de AMs para alcanzar buenos resultados para algunos problemas comunes del área de reconocimiento de patrones.

Descriptores: Ciencias de la computación y tecnología; ingeniería neuronal; calidad de imagen; contraste; resolución; ruido; análisis de imágenes.

PACS: 89.20.Ff; 87.80.Xs; 87.57.Ce; 87.57.Nk

1. Introduction

Associative Memories (AMs) are simple and useful devices designed to recall output patterns in terms of input patterns by means of simple operations. AMs are considered as part of Artificial Neural Networks (ANN), but instead of using complex structures and operators based on transcendental or trigonometric functions, AMs work in two simple phases using two elemental operators, one for association and one for recall. During the association phase the AM is built in terms of the associations $(X, Y)^k$, for k an integer. Every pattern association is carried out by the application of simple operations such as: additions, multiplications, maximums, minimums, and others. Associative Memory $\boldsymbol{M}$ is represented by a matrix, which is formed from all the pattern associations. The corresponding components m_{ij} can be seen as the synapses of a simple neural network. Operator $\boldsymbol{M}$ is generated from a preset of finite known associations called the fundamental set; this association set is represented as $\{(X^k; Y^k)|k = 1, \ldots, p\}$, where p is the number of associations. If $(X^k = Y^k)\forall k = 1, \ldots, p$, then $\boldsymbol{M}$ is considered auto-associative, otherwise it is hetero-associative. If a distorted version of a pattern, denoted as $\widetilde{X}$, is fed to $\boldsymbol{M}$ and the obtained output is exactly Y^k, then recalling is considered as perfect. The simplicity of AMs mod-

els is a result of a great development, which has been carried out during the last 50 years; for some examples, refer to [6,22,24,14,15,25,23,31]. Most of these models have several limitations: limited storage capacity, difficulty to deal with more than one type of pattern (binary, integer or real-valued), lack of robustness to different kinds of noises (additive, subtractive, mixed, Gaussian, etc.); and above all, most of these models work only for the purpose for which they were specially designed and not for any other [32]. All these models have been manually developed. The process of designing one model could take one or two years.

A first attempt to automatically generate AMs has been reported in [16]. This method was conceived for the case of bidirectional associative memories (BAM). On the other hand, the closest work to the approach presented in this paper is concerned with the automatic design of Artificial Neural Networks [13], which uses some kind of genetic programming (GP) for this purpose. Another reported work applies Particle Swarm Optimization [4]. Bearing in mind the relevant results obtained with GP techniques versus Genetic Algorithms [21], in this work we describe a GP-based methodology to automatically generate AMs. GP has been successfully applied in a huge range of areas. In particular, we can mention some related to computer vision [1,29,10,12,27,28,35,36,20,26].

In this paper, we deal with the problem of automatically generating AMs through GP; previous results can be found in [33]. We improve our initial technique and provide a better description of the problems involved in the recall of binary and real-valued patterns.

The rest of the paper is organized as follows. In Sec. 2 we provide a brief description of our first approach fully described in Ref 33. In Sec. 3 we present our new co-evolutionary based-GP methodology for the automatic generation of AM through GP. In Sec. 4 we show the experimental results for some representative databases. Finally, in Sec. 5 we present the conclusions and suggestions for further research.

2. Automatic design of AMs through GP

The proposed model for generating AMs using evolutionary algorithms rests on the following two-stage rule. (1) Evolution starts as a simple ANN with a pre-established topology that is evolved from more than one connection [37]. (2) Later, evolution of architecture is developed with the aim of generating different topological structures. Common AM models fix the connection structure and consider the full aspects of every design such as the capacity of memory recall, as well as noise suppression. Our proposal is based on the linear associator concept introduced in [7], in order to consider the general association aspects between pattern sets, but bearing in mind the local association parts focused on the connectivity of every synapse that is carried out between the vector pattern components. Taking into account our first work, we present a very important modification using the co-evolutionary paradigm. As we will see, this allows modeling the two processes in AM designing: association and recall, with the idea of preserving the local pattern aspects.

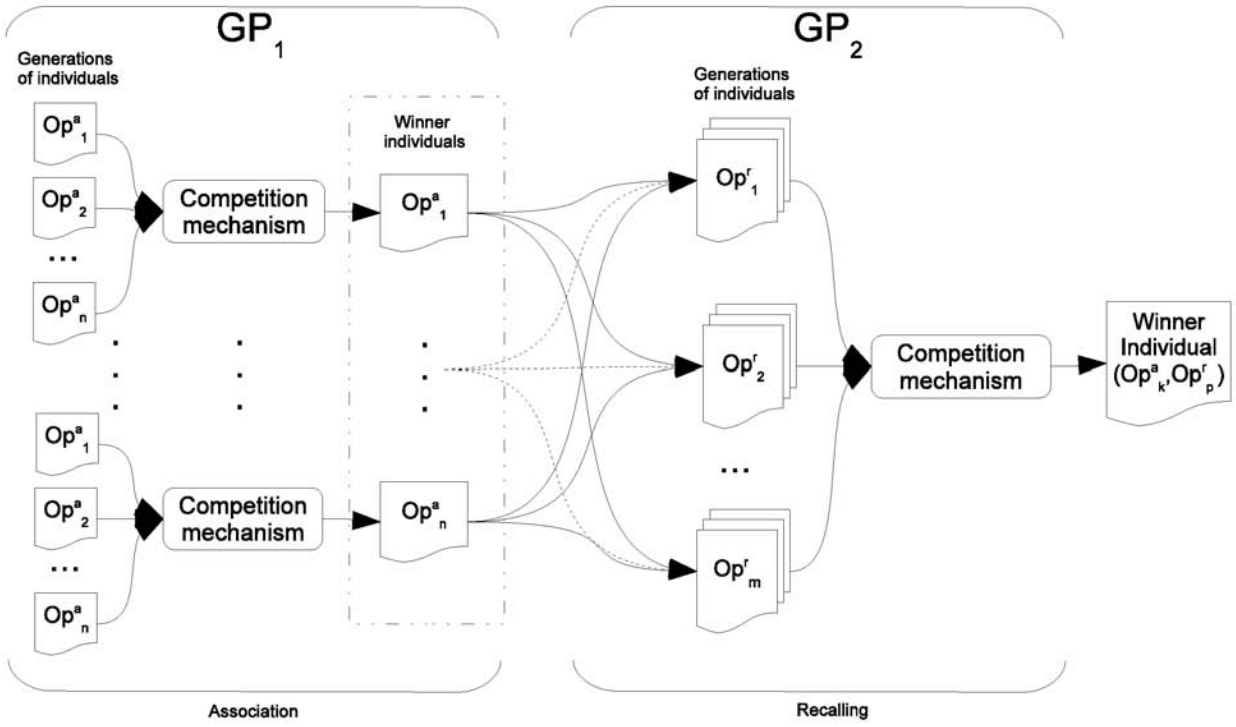

FIGURE 1. Framework of our cooperative co-evolutionary process.

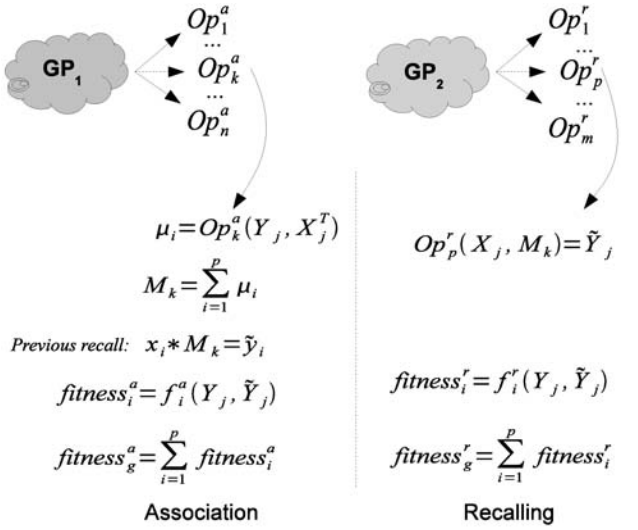

FIGURE 2. Proposed co-evolutionary model for the GP-development of AMs.

Our proposal outlines the evolutionary process in three steps and two phases. First, we define the function operators used during association; second, we define the function operators and terminal elements for recalling, considering the fundamental aspects of the association matrix previously built; and thirdly, we implement the co-evolutionary approach [2] by joining two evolved individuals to create a new one.

3. Design of AMs

Our proposed GP based co-evolutionary model comprises two populations, one for association, and one for recalling. This is a first contribution with respect to previous works that considered only one evolutionary process [33]. In this paper, we consider two sequential processes instead of only one bearing in mind the cooperative co-evolutionary paradigm, we consider as more adequate to reach our goal of developing a simulated correlation between species. Thus, population represents a separate part of the problem, in such a way that every species cooperates to have a global solution in two considered phases. Hence, every individual is a possible solution for the problem in the pre-set common environment. The basic idea of co-evolution consists in using the concept of dividing and conquering the problem (system) into many sub-problems (sub-systems), and evolving them separately, then later combining them for implementing the entire solution [11].

Figure 1 shows the framework for implementing the cooperative co-evolutionary process. Here, the solutions (appropriate AMs fitting one pattern set with a perfect recall) come from two search spaces. The first process (GP_1) is a clustered process developed to produce individuals for the association stage of the AM, where every individual identified by each cluster is the winner of one evolutionary process. Then, all winner individuals participate in the solution of the second process (GP_2, the recalling one), by recombining their learned features. After the whole process is completed we get an individual global winner.

Fitness function of the whole process is implemented as the minimum of all the individual pairs after the second phase. The components of our new model are defined in terms of both functions and terminal sets per every population for each GP evolutionary process, see Fig. 2. Thus, the co-evolutionary model is composed of the following aspects:

- Op_k^a represents the evolutionary operator used in each pattern association. Its representation is generated as an individual in the form of a tree. This correspond to the encoded genotype. The local association matrix μ_i is integrated by the Op_k^a operator acting on the components of each vector of the local association $\{Y_j, X_j^T\}$.

- M_k describes the general association matrix. It is taken as the sum of all local associations and provides the cumulative knowledge inspired by the perceptron principle.

- T_a is the *Terminal Set* for the association stage. It consists of a set of nodes taking the vector entries, such as $T_a = \{x_j, y_j\}$. These nodes belong to each pattern-vector $\{X\}$ and $\{Y\}$; this is in order to have a local encoded correlation input.

- F_a is the *Function Set* for the association stage. These functions have been defined by considering the solutions reported in literature in order to target the space of possible structures with the aim of producing individuals similar in performance to the reviewed AM models: $F_a = \{+, -, \wedge, \vee, \times\}$

  where $\wedge, \vee$, and $\times$ are the *minimum, maximum* and *multiplication* operators

- Op_p^r is the evolutionary operator for pattern recall. It comprises the input vector X_j, as well as the matrix association M_k generated by the previous evolutionary association operator.

- T_r is the *Terminal Set* for the recalling stage, T_r=$\{v, R_1, R_2, \ldots, R_m, M_k\}$; with $v \in X$, as the input vector, and R_i as the ith-row-vector belonging to M_k, which is the corresponding kth-AM.

- F_r represents the *Function Set* for the recalling stage. $F_r = \{+, -, \wedge, \vee, \otimes\}$; $\otimes$ is defined as the multiplication operator between vector components, $\otimes(X, Y)$=$[x_1 * y_1, x_2 * y_2, \ldots, x_n * y_n]$, and when the association matrix is considered is defined as follows $\otimes(X, M_k) = X * M_k$; thus it satisfies the dimensionality of the multiplication operator between matrices.

- $\tilde{Y}_j$ is the approximated Y_j pattern resulting from the application obtained with the operator rule Op_p^r in the recalling stage.

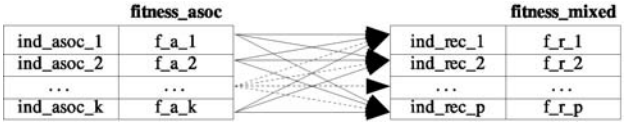

FIGURE 3. Fitness model for the GP-development of AMs.

TABLE I. Winner individuals (pairs) of the association rules (first column in competition), with its corresponding recalling rule (second column), and the global fitness attained in cooperation. The indexed operators in the first column are related to each association rule. Operator Op_1^a is related to Fig. 7(a); numbers two and three are related to Figs. 7(b) and 8. Operator Op_1^r in the second column is related to the operator depicted in Fig. 9(a), and so on.

Association rule	Recalling rule	Global Fitness
Competition		
$(Op_1^a, Op_2^a, Op_3^a) := Op_1^a$	Op_1^r	1
Competition		
$(Op_1^a, Op_2^a, Op_3^a) := Op_1^a$	Op_2^r	1
Competition		
$(Op_1^a, Op_2^a, Op_3^a) := Op_1^a$	Op_3^r	1

The fitness function which provides the qualification to every process as an essential evolutionary step in the whole process is requested. For our purposes the fitness value is applied at the end of both stages (association and recall). The fitness function covers the sum of all the local fitness (related to local associations), and all the recalling set (one recalling for every pattern relation), as shown in Fig. 2. For this evolutionary approach we considered the normalized correlation coefficient between the goal (Y) and the source processed pattern ($\tilde{Y}$) [18]. The fitness function f, known as *similarity* $(0 \leq f \leq 1)$, is defined as follows:

$$f = \frac{Y \cdot \tilde{Y}}{\sqrt{Y \cdot Y}\sqrt{\tilde{Y} \cdot \tilde{Y}}} \tag{1}$$

where Y and $\tilde{Y}$ are vectors of size $1 \times N$, and $Y \cdot \tilde{Y}$ is given by the following equation:

$$Y \cdot \tilde{Y} = \frac{1}{N} \sum_{j=1}^{N} Y(1,j) \cdot \tilde{Y}(1,j) \tag{2}$$

Function f tries to maximize the number of matching component vectors Y and $\tilde{Y}$. This seems a reasonable choice for the fitness function. The optimum is found when $f = 1$ corresponds to the matching of all pixels. The worst case takes place for $f = 0$, implying that not a single pixel matches.

Thus, we used Eq. (1) as our fitness function, and was applied in the evaluation of every generated individual μ_i. For the association stage, the training one, between the source set X, and the goal set Y, both conformed our *fundamental sets*. The fitness evaluation is carried out in three steps: First, the *association* between pattern sets X and Y using the first stage operator Op_k^a; second, we fix the *multiplication* operator for the recalling task just to have a first estimation of *multiplication* operator. After this we have one recalled pattern set, $\widehat{Y}$ and this local fitness is considered to have the winner individual for the first evolutionary process GP_1, which is taken into count for the n batch processes (see Fig. 1, the association block). Second, the $n-$winner individuals from the first process are considered for the second GP process, the recalling-one. At the end of this block, the fitness function is applied again to get the winner individual for every $m-$population as a local estimation to each operator Op_a^r. Here we wave a pair of operators as one individual (Op_k^a, Op_a^r), thirdly, we consider the global fitness for each pair of operators as the high-fitness grade for the solution pair. To follow the explanation, refer to Figs. 1, 2 and 3.

3.1. GP experiments and parameters

All the experiments were implemented on workstations with 64-bit architectures using Matlab with GPLab toolbox ver. 3.0 [17]. The co-evolutionary process was composed of two phases, association and recalling. They were performed in two batch processes n and m: GP1 and GP2, respectively. After 50 generations with 70 individuals per every evolutionary process, we got one evolutionary solution for our problem.

For the experiments we considered the GP parameters similar to those suggested in [8], rate values of 0.7 and 0.3 for the crossover and mutation operators respectively were set up. In order to initialize populations, a ramped-half-and-half initialization method for the mutation was used.

4. Results and analysis

In this section we present the set of experiments we have performed to test the efficiency of the proposed methodology. For this we have used two datasets. In the first case we show how the proposal works. In the second we show the application of the proposal to solve one classic pattern recognition problem. For the first experiment we used simple vector sets consisting of binary sets of digit characters from 0 to 9, within a matrix size of 7×5, as shown in Fig. 4.

For these simple images we considered the line vector per matrix where X and Y are the source and the goal vector sets, respectively, using an auto-associative relationship.

The application of our methodology generates a set of rules for association, and another set of rules for recall. Each of these rules is co-evolved with the aim of maximizing the global fitness. The winner is one pair set as shown in Fig. 5. The AM shown in Fig. 5 is only one example of pairs selected from a vast production of individuals, which fits to this kind of pattern perfectly.

The solution depicted in Fig. 5 shows that an important describing feature for this particular pattern set, is row number 7, which belongs to the association matrix M_k. It is the only significant one during recalling regarding the auto-

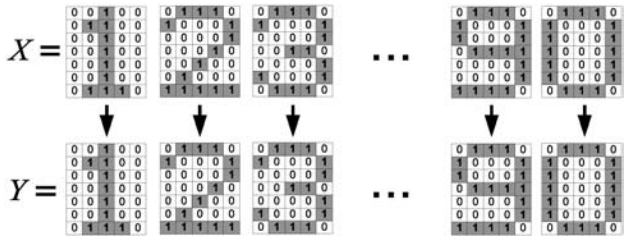

FIGURE 4. Matrices representing digit characters for the first AMs generation.

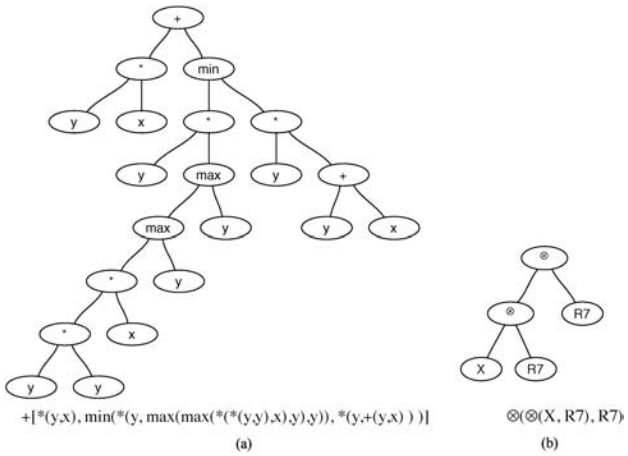

FIGURE 5. Evolutionary AM for the auto-associative case related to digit characters set. 5(a) Rule for the pattern association; 5(b) Rule for the pattern recall.

associative relationship. As can be seen this solution is generated by the corresponding association rule 5(a).

We then tested the AMs formed by the individuals shown in Fig. 5 adding mixed noise (salt and pepper) in order to evaluate its robustness. The mixed noise was applied from 10% to 100 % scale in steps of 10%, see Fig. 6. These noisy images were generated using Matlab. Remarkably, we got perfect recall: error zero, so our GP-generated AM had better behavior compared to the morphological and the alpha-beta models [15] and [34].

After this excellent result we decided to implement our methodology considering one association problem endowed with more complexity, real valued pattern associations which involved a bigger and more complex search space, which is closer to a real application.

We applied our methodology to a very-well known database problem, the Iris Plant classification database [30], which considers 4 features, 3 classes and 150 instances. The resulting AMs suitable for this problem, in real valued patterns, are shown in Figs. 7 and 9. All the experiments were implemented in the auto-associative case. We implemented our model looking for three-individuals to be used as the association rules (association stage). Later the co-evolutionary process was executed to find three-individuals in order to

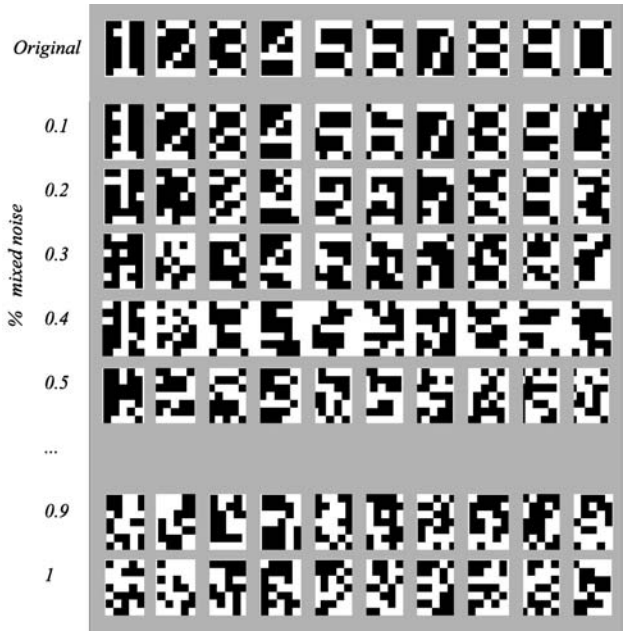

FIGURE 6. Digit characters represented as image matrices. Without noise in the first row, and after adding some mixed noise to the original images.

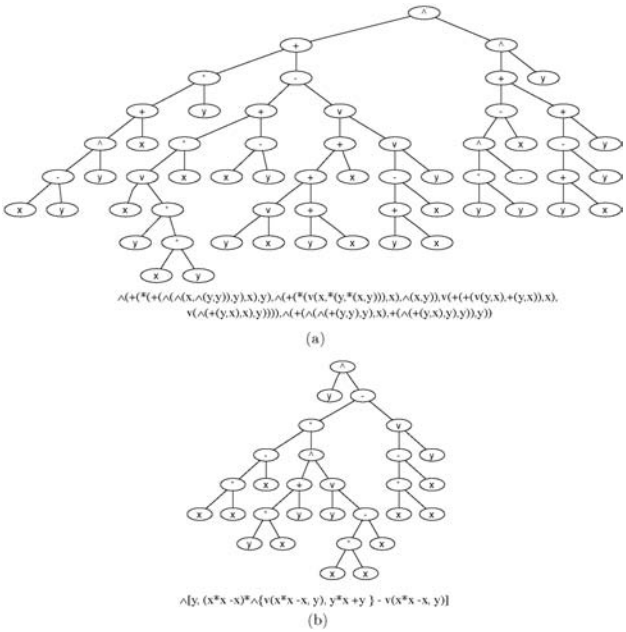

FIGURE 7. Evolutionary AM for Iris Plant database, three different rules for pattern association (considering the individual en Fig. 8 too), where x and y are the input and target patterns components.

recall the patterns. By co-evolving each individual of the association stage and by following the proposed cooperative co-evolutionary model we wanted to achieve the desired solution. Each individual has a tag per every evolutionary process, and the global fitness for every winner pair is shown in Table I.

As we can observe, according to Figs. 7, 9 and Table I, the evolutionary process produces miscellaneous solutions,

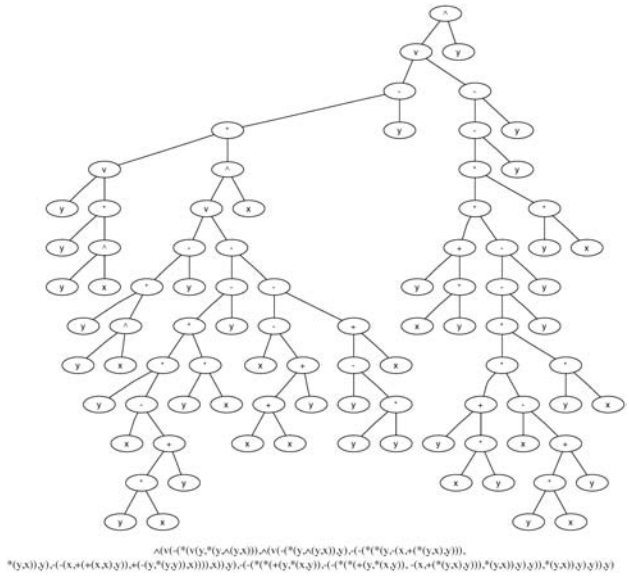

FIGURE 8. Evolutionary AM for Iris Plant database, longest individual.

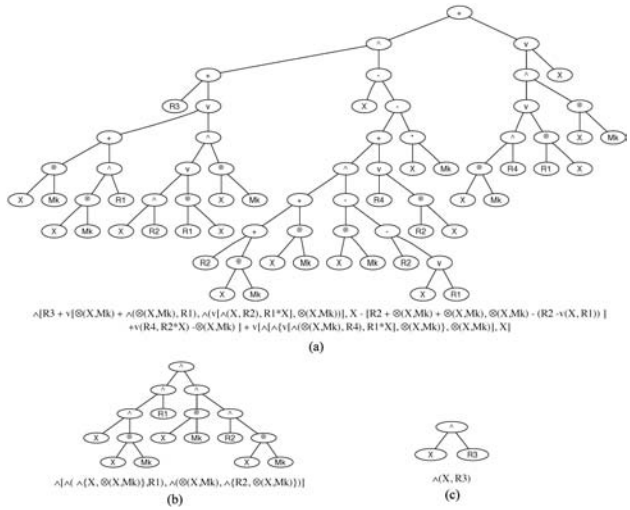

FIGURE 9. Evolutionary AM for Iris Plant database, three different rules for pattern recall, where Mk is the association matrix.

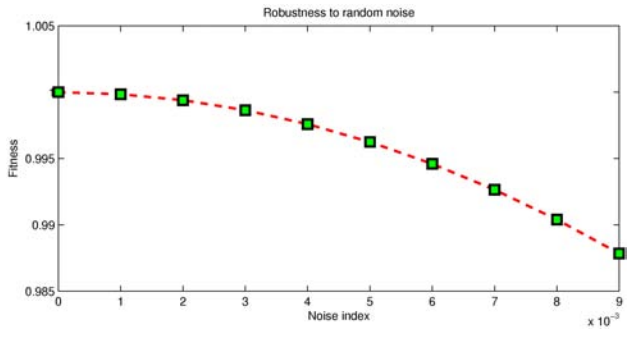

FIGURE 10. Robustness to noisy patterns for the three AM pairs.

from a complex one (Fig. 8) to just a simple one rule for the recalling stage, see Fig. 9(c). According to Table I, the association rules, see Figs. 7(b) and 8, are not good enough for the association stage. All the three-evolutionary rules obtained for the recalling stage achieved the higher fitness through the first association rule, Fig. 7(a).

Another important aspect revealed through this methodology are the main synapses or connections used. For this example the winner rule for the recalling stage shows the rows numbers 1, 2 and 4, which belong to their association matrix M_k, generated by the association rule $O_{p_1}^{a}$ (Fig. 9(a)). These rows are the most significant during the recalling phase, which are related to the sepal length, sepal width and petal width. So, the new AM shows only three of the four features as the most relevant from the Iris Plant database [30].

We tested this new AM by adding random noise to the input pattern set X; regarding the Iris database the noise scale has only a small amount of noise, from 0.01 to 0.09 %; hence, the resulting fitness is depicted in Fig. 10. We observed that the recalling rate decreased slowly as noise increased.

5. Conclusions

In this paper we improved our first reported approach where we originally proposed the GP-based methodology for the synthesis of AMs. We extensively described the inherent problems that were solved to achieve our final goal [33]. With the new methodology we obtained several AMs that perfectly fit each pattern set. The time necessary to generate such solutions varies from hours to days, depending on the computational time necessary to synthesize ten or more models instead of the traditional approach that takes years of research performed by experts. This methodology is similar to the development of ANN considering genetic algorithms. Our methodology has one very important advantage, the possibility to develop AMs specially designed for specific pattern sets. The resulting new AMs are produced by the cost effective GP that searches for optimized solutions. In particular, these can be implemented in order to have several solutions per pattern applications employing a less time. At the moment we are working on improving our methodology with the use of new fitness functions and metric spaces, according to the kind of pattern sets for challenging real world applications.

Acknowledgements

The Authors thank SIP-IPN for the economical support under grant number 20100468. Authors thank the European Union, the European Commission and CONACYT for the economical support. This paper has been prepared by economical support of the European Commission under grant FONCICYT 93829. The content of this paper is an exclusive responsibility of the CIC-IPN and it cannot be considered that it reflects the position of the European Union. We thank also the reviewers for their comments for the improvement of this paper.

1. S. Cagnoni, E. Lutton, and G. Olague. *Genetic and Evolutionary Computation for Image Processing and Analysis, of EURASIP Book Series on Signal Processing and Communications Series* vol. 8 (Hindawi Publishing Corporation, 2008).
2. Goh, Chi-Keong and Tan, and Kay Chen, *IEEE Transactions on Evolutionary Computation* **13** (2009) 103.
3. N. Barricelli, *Methodos* (1954) 45.
4. B.A. Garro, H. Sossa, and R.A. Vazquez, *International Joint Conference on Neural Networks* (IJCNN 2009, Atlanta, GE, USA.) p. 938.
5. J.H. Holland, *Adaptation in natural and artificial systems* (University of Michigan Press, 1975).
6. J.J. Hopfield, *Proceedings of the National Academy of Sciences* **79** (1982) 2554.
7. T. Kohonen, *IEEE Transactions on Computers* **C-21** (1972) 353.
8. J.R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection* (MIT Press. 1992).
9. R. Forsyth, *Kybernetes* **10** (1981) 159.
10. B. Hernández, G. Olague, R. Hammoud, L. Trujillo, and E. Romero, *Comput. Vis. Image Underst* **106** (2007) 258.
11. Potter and A. Mitchell, and Jong., Kenneth A. De. *Evolutionary Computation* **8** (2000) 1.
12. C. Perez, and G. Olague, *Learning Invariant Region Descriptor Operators with Genetic Programming and the F-Measure* (International Conference on Pattern Recognition. ICPR., 2008).
13. D. Rivero, J. Rabuñal, and Alejandro Pazos, *Automatic Desing of ANNs by Means of GP for Data Mining Tasks: Iris Flower Classification Problem.* ICANNGA 07. Part I, Springer-Verlag LNCS 4431: (2007) 276.
14. G.X. Ritter *et al.*, *IEEE Transactions on Neural Networks* **9** (1998) 281.
15. G.X. Ritter *et al.*, *International Journal of Mathematical Imaging and Vision* **19** (2003) 95.
16. Shi Guoyong, *International Journal of Systems Science* **28** (1997) 133.
17. S. Silva and J. Almeida, *GPLAB- A Genetic Programming Toolbox for MATLAB. In Gregersen L (ed), Proceedings of the Nordic MATLAB Conference* (2003) 278.
18. M. Quintana, R. Poli, and E. Claridge, *Genetic Programming and Evolvable Machines* **7** (2006) 81.
19. I. Rechenberg, *Evolutionsstrategie Optimierung technischer Systeme nach Prinzipien der biologischen Evolution* (PhD thesis, Reprinted by Fromman-Holzboog. Berlin, Germany, 1973).
20. A.R. Silva Lavalle, *Un Método de Algoritmos Genéticos para Optimización de Memorias Asociativas Morfológicas* (Tésis. Univesidad de Puerto Rico. 2006).
21. K. Seo and S. Hyun, *Applications of Evolutionary Computation, LNCS* **6024** (2010) 352.
22. H. Sossa, R. Barrón and R.A. Vázquez, *New associative memories for recall real-valued patterns. LNCS, 3287* (2004) 195.
23. H. Sossa and R. Barrón, *Rev. Mex. Fí.* **53** (2007) 10.
24. K. Steinbuch and Die Lernmatrix, *Biological Cybernetics* **1** (1961) 36.
25. P. Sussner, *Journal of mathematical imaging and vision* **19** (2003) 81.
26. K. Trist, V. Ciesielski, and P. Barile, *Arts and Technology* **262** (2010) 255.
27. L. Trujillo and G. Olague, *Using Evolution to Learn How to Perform Interest Point Detection ICPR.* (2006) pp. 211.
28. L. Trujillo and G. Olague, *Evolutionary Computation. MIT Press* **16** (2008) 483.
29. G. Olague and C. Puente, *Honeybees as an Intelligent based Approach for 3D Reconstruction* (International Conference on Pattern Recognition. Hong Kong, China. 2006).
30. A. Asuncion and D.J. Newman, *UCI Machine Learning Repository* (2007).
31. R.A. Vázquez and H. Sossa, *A New Model of Associative Memories Network. Third International Workshop on Artificial Networks and Intelligent Information Processing* (ANNIP, Angers, France, May 2007) p. 9.
32. R.A. Vázquez and H. Sossa, *Hetero-Associative Memories for Voice Signal and Image Processing* (CIARP '08: Proceedings of the 13th Iberoamerican Congress on Pattern Recognition, 2008) p. 659.
33. J. Villegas-Cortez, H. Sossa, C. Aviles, and G. Olague, *Advances in Computer Science and Engineering* **42** (2009) 91.
34. C. Yáñez M. and J.L. Diaz de León, Ph. D. Thesis abstract. *Computación y Sistemas* **6** (2003) 300.
35. M. Zhang, P. Andreae, and M. Pritchard, *Applications of Evolutionary Computing* (2003) 455.
36. U. Bhowan, M. Zhang, and M. Johnston, *Genetic Programming for Classification with Unbalanced Data* (2010) 1.
37. X. Yao, *Proceedings of the IEEE* **87** (1999) 1423.