Generative Capacities of Grammars Codification for Evolution of NN Architectures

M. A. Guinea, G. Gutierrez, I. Galván, A. Sanchis, J. M. Molina
Sca-Lab. Departamento de Informática.
Universidad Carlos III de Madrid.
Avda. Universidad 30, 28911-Leganés (Madrid)-SPAIN.
e-mail: molina@ia.uc3m.es

Abstract- Designing the optimal architecture can be formulated as a search problem in the architectures space, where each point represents an architecture. The search space of all possible architectures is very large, and the task of finding the simplest architecture may be an arduous and mostly a random task. Methods based in indirect encoding have been used to reduce the chromosome length. In this work a new indirect encoding method is proposed and an analysis of the generative capacity of the method is presented.

I. INTRODUCTION

In the last years, many works have been centered toward automatic resolution of the design of neural networks architecture [1,2,3,4,5,6]. Two representation approaches exist to find the optimum net architecture using Genetic Algorithm (GA): one based on the complete representation of all the possible connections and other based on an indirect representation of the architecture. The first one is called Direct Encoding Method, and is based on the codification of the complete network (connections matrix) into the chromosome of the GA, and starts with Ash's work [6] and continues with other works, including [7, 8, 9, 10]. The direct representation is relatively simple and straightforward to implement. However, it does not scale well since large architectures require very large chromosomes to be represented. It is specially suitable for small and problem dependent particular architectures. In these cases, some unpredictable designs could be reached [9]. However, the capabilities of direct encoding for larger architectures are limited, remaining to be proven whether it can scale up to more complex tasks, because large architectures requires much larger chromosomes. This could end in a too huge space search that could make the method impossible in practice.

The second one is the called Indirect Encoding Method that consists on codifiying, not the complete network, but a compact representation of it [1]. In 1990, Kitano [9] presents a new method for designing neural networks by means of Genetic Algorithms, where neural networks are represented throught graph grammars, codified in the chromosomes of the individuals. The grammatical approach is not the only proposed representation method. Merril et al. [10] introduced a fractal representation for encoding the architectures, arguing that this representation is more related with biological ideas than constructive algorithms. Valls et al. [11] proposed a Multiagent System to find optimal architectures in Radial Basis Neural Networks.

An indirect scheme under a constructive algorithm, by the other hand, starts with a minimal architecture and new levels, neurones and connections are added, step by step, using some sets of rules. The rules and/or some initial conditions are codified into a chromosome of a GA. In this way, an evolution towards good constructive rules is achieved, and there is no evolution of architectures by themselves.

The solution proposed by Kitano was to encode networks as grammars, and let the GA to evolve grammars instead of networks architectures. Other works have considered some variations of Kitano's rule descriptions using recursive equations to model the growth of connection matrix [12]. In this case, the coefficients of some fixed equations were codified as chromosomes and evolved by a GA.

Kitano's approach is limited and has some lacks, mainly that for an NN of "n" neurons, an "m x n" matrix is needed, where "m" is the smallest power of 2 bigger than "n", and only the upper triangle of the "m x n" matrix is used, which also decreases the efficiency of the encoding. Also, in Kitano's method, the recursion is not included [5].

In this paper, one of the first objectives is presenting and testing a new approach has been developed that extends and improves Kitano's method to make a general and powerful method of designing NN, without restrictions in the matrix size and with recursion. This grammatical mechanism is integrated in a complete system, called GANET [13, 14]. This system is extended to incorporate in the chromosome the whole grammar without a predefined expansion method and the level applied to recursion. This extension will make the method more general, compact and powerful. In GANET, each

0-7803-7282-4/02/$10.00 ©2002 IEEE

chromosome codified a bidimensional grammar. By means of genetic operators, a population of grammars is obtained, except the first one that, as is usual [15], is randomly generated. These new individuals are evaluated through a fitness function. To calculate the fitness, the grammar codified in the individual chromosome is expanded, and a two-dimensional matrix (a word) is obtained. In a next step, this word is decodified as an NN matrix of connections. This matrix is transformed into an NN architecture that is trained. After training the NN, it is tested and an error value is obtained. With this error and some other relevant information about the NN (size, learning cycles, etc.), the fitness value of the considered individual is computed. The process is repeated until all the population is evaluated.

The second objective of this paper is analysed the capacity of GANET to generate any size of Neural Network. In this way a number of chromosomes have been defined randomly and a graphic representation of the matrix is presented and analysed.

II. BIDIMENSIONAL GRAMMARS

In this work, Context Free Chomsky Grammars, G2, have been employed [16, 17], particularly bidimensional or graph ones [18]. These grammars are defined as a quintuple, G = {ΣT, ΣN, S, P, EM}, where ΣT is the alphabet of terminals, ΣN is the alphabet of non-terminals, S is the start symbol, S ∈ ΣN, P is the production or rules set and EM is the Expansion Method. This kind of grammars generates words that are matrix of terminal symbols.

In order to carry out the derivation process in a bidimensional context free grammar and to generate words, sometimes it is necessary to insert a row, a column, both row and column or neither. The position of the space inserted depends on the position of the symbol and the expansion method. For generating a word useful for Ganet system, a special derivation mechanism has been developed. This mechanism keeps the rectangularity of words during the derivation, inserting an auxiliary symbol that produces, when necessary, the space required before producing the desired derivation. For the Up and Right expansion method, the inserted row will be placed over the symbol and the inserted column will be placed at the right. In the production rule of Figure 1, the insertion of the row and column is made in first place, and the non-terminal symbol (in Figure 1 is the symbol B) is derived later.

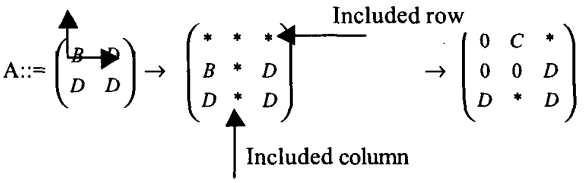

Figure 1: Derivation of non-terminal symbol B.

Then, the derivation sequence of all the non-terminal symbols is carried out "through levels". Firstly all the non-terminal symbols of the same level (that belong to the same previous derivation process) are derived and, secondly, the non-terminal symbols of next levels. In the example of Figure 1, the symbol D (that belongs to the first level) is derived before the derivation of the symbol C (that belongs to the second level). This derivation order assures that the number of auxiliary symbols is minimum. Finally, to stop the derivation process (when using recursive rules) two parameters ought to be specified: how many times a recursive rule is applied (maximum value of recursion, MNR) and the maximum number of levels with non-terminal symbols (number of levels, NL).

III. BIDIMENSIONAL WORD INTERPRETATION

The architecture selected is a Multi Layer Perceptron (MLP), with backpropagation, forward connections, and with one hidden layer. A hidden layer allows any solution without loosing generality, because is proven that any problem could be solved with only one hidden layer if there is not restrictions in the number of neurons in this layer. The method to obtain the MLP from the word obtained in the previous module is as follows:

1. 0 substitutes auxiliary symbols. Figure 2 shows this step in an example of word.

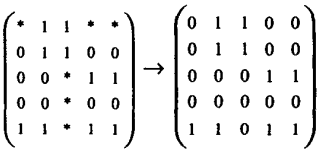
Figure 2: Transformation of auxiliary symbols in 0's.

2. The matrix is divided in three zones, one for each layer. The first zone is the input layer, the second one is the hidden layer and the last one the output layer. The first and the third layer have a constant size defined by the problem. The second zone, the hidden layer, has a variable size and depends on the word. The three zones appear in rows and column, as it is shown in Figure 3.

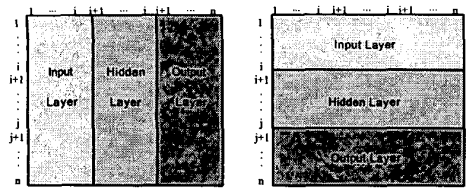

Figure 3: Matrix zones in neural module of Ganet system.

Then, the first i components of the matrix will correspond with input neurons, the following (n-j) neurons with the hidden layer and the last j with the output layer. Values of i and j were fixed through the problem to solve and will be kept constant during all the process. However, n is not a constant value, so the number of neurons in the hidden layer depends on the concrete grammar applied and this concrete grammar is learned by Ganet system. The final number of neurons in each layer is the number of neurons that are connected to other neurons. This number is obtained analysing the connections that appear in the matrix of Figure 2 following the division of Figure 3 at it is showed in Figure 4.

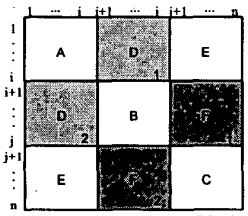

Figure 4: Conections between neurons considering three layers in the Neural Network.

Each zone in Figure 4 corresponds to a different type of connection: Zone 'A', connections between neurons of the input layer, Zone 'B', connections between neurons of the hidden layer.
Zone 'C': connections between neurons of the output layer, Zone 'D' (D1 AND D2), connections between neurons of the input and hidden layers, Zone 'E', connections between neurons of the input and output layers, Zone 'F' (F1 AND F2): connections between neurons of the output and hidden layers.

In the proposed method, only zones D and F (Figure 4) are considered. The position (x, y) with a value of 0 in the matrix means that x and y neurons are not connected, a 1 value will mean that a connection exists. In order to evaluate the existence of a connection between two neurons a 1 value must appear in the two subzones (D1 and D2, F1 and F2). The logical AND operator has been used because the meaning of D1 (F1) is the same than the D2 (F2). An example is shown in Figure 5.

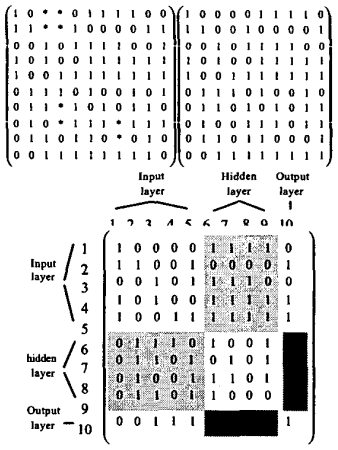

Figure 5: Example of interpretation of a matrix to obtain a Neural Network.

The NN interpreted from Figure 5 is shown in Figure 6.

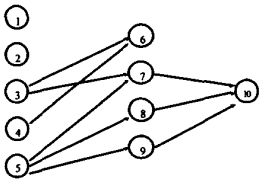
Figure 6: Neural Network obtained from Figure 5.

IV. CODIFICATION

Ganet works with a population of grammars, so the chromosome codifies in binary a bidimensional grammar. The codification represents all the elements of a bidimensional grammar: (Σ_T, Σ_N, S, P), the expansion method and the level of recursion.

The first part of the chromosome represents the production rules. The right side of the production rules is represented in the chromosome, as shown in Figure7 , where T represents terminal symbols and NT the non terminal ones. The genotype of Figure 7 represents grammars with four different NT symbols, and has been used in this work.


Figure 7: Structure of the genotype.

Each part of genotype shown in Figure7 is the right part of a production rule, in the same sequence that appears in the grammar definition, as shown in Figure 8. The first four rules contain non-terminal symbols and the last ones are composed only by terminal symbols. Finally, the codified chromosome needs two bits for representing non-terminal symbols (four non-terminal symbols) and one bit for terminal ones (two terminal symbols, 0 and 1).

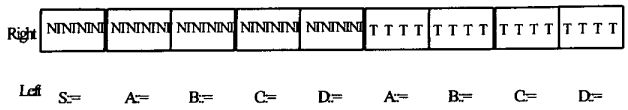

Figure 8: Production rules represented in the genotype.

In this work four non terminal symbols have been used because the first production rule in these graph grammars (S ::= anything) has four non terminals in the right part. More non-terminal symbols could be considered only increasing the size of the genotype. The final part of chromosome (the one shown in Figure 8 is followed by four binary digits) represents the Expansion Mode and the number of levels. In this way the whole grammar has been codified. Depend on the value of number of levels different sizes of NN are obtained.

TABLE I. GENE VALUES, CORRESPONDENT LEVEL OF RECURSION AND SIZE OF NN.

Gene Value	Number of Levels	Number of Neurones
000	1	-
001	2	$2^{2+1}= 8$
010	3	$2^{3+1}= 16$
011	4	$2^{4+1}= 32$
100	5	$2^{5+1}= 64$
101	6	$2^{6+1}= 128$
110	7	$2^{7+1}= 256$
111	8	$2^{8+1}= 512$

V. GENERATIVE CAPACITY

The proposed system should be a general method of generating NNs, no including bias due to the codification neither the derivation process and the space of the NN architectures is completely covered. For discarding these undesirable influences, starting from a randomly generated population of chromosomes the obtained matrix should be distributed along the matrix space. The generated matrixes are squared, from the 8x8 dimension to a maximal of 512 x 512. A matrix dimension is computed as $2^{\text{number of levels}+1} \times 2^{\text{number of levels}+1}$. So, 2262144 different matrixes of 512 x 512 dimension could be generated, 265536 of 256 x 256 dimension, etc, causing an extremely huge population. A representative sample from this population should be extracted. The proposed equation to estimate the size of the sample in an infinite sample is Equation 1.

$$n = \frac{1{,}96^2 * 50^2}{e^2} \quad (1)$$

Where: e is the maximum allowed error and n: sample size

For obtaining a confidence interval of 95% an e = 5% will be used Aplying Equation 1, a 400 individuals sample is obtained. Finally a 1000 individuals sample has been used to obtain optimal results.

At this point a mechanism of representing the obtained matrixes is required. The values associated to a matrix are the size and the values of each position in the matrix. Both values are composed of two values each one, so a function that combines them into one value should be used. The proposed function is given in Equation 2 for the dimension and in Equation 3 and Equation 4 for the values in each position.

$$\log_{10}(i \times 2^j) \quad (2)$$

where i is the vertical dimension of the matrix and j the horizontal one.

$$\log_{10}(k \times 2^h) \quad (3)$$

where k and h indicates the position of the matrix with a 1 inside. This equation is applied to every position in the matrix that contains a 1. All values of one row are added and all the rows are added, following Equation 4.

$$\sum_{1}^{H}\sum_{1}^{K} \log_{10}(k*2^h)* X_{k,h} \quad (4)$$

Where h, k are the vertical and the horizontal dimensions of the matrix, respectively, k and h are the row and column of the matrix and $X_{k,h}$ is the value of the k,h position.

Graphical representations of these values can be found in Figure 9. In x-axis value related with the dimension is placed and in y-axis the calculated value for the correspondent positions. So, the greater the matrix the righted in the graph, and the most ones in the matrix the upper in the graph.

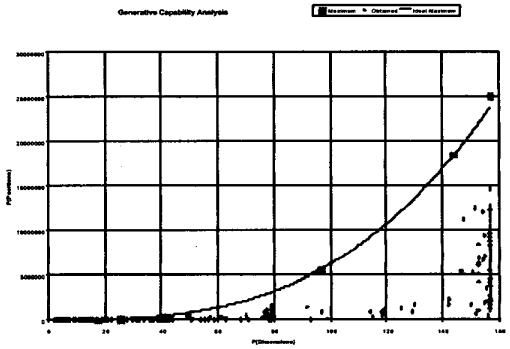
Figure 9: Graphical representation of the system's generative capacity.

Figure 10 contains the same information of figure 9 but the scale is different. The curve and the biggest dots of Figures 9 and 10 represent the function maximum values. Each possible generated value should be under the maximum values. Looking Figures 9 and 10 carefully allows to observe that there is a hole between 100 and 140 values. This is related with the fact that values near $2^n x\ 2^n$ (2x2, 4x4, 8x8, 16x16, 32x32, 64x64, 128x128, 256x256,

512x512) are usually obtained. From the 957 matrixes of the used sample, the correspondent dimensions are in Table II.

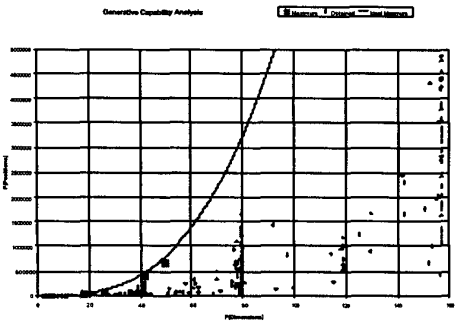

Figure 10: Representation of figure 9 in different scale.

In Table II, 632 matrixes are considered, adjusted to the 2^n x 2^n shape. These results throw the validity of the proposal because there are values along all the graph, the 2^n x 2^n shape matrixes are in large number and inside this kind of matrix the smallest are more frequent. This allows to conclude that with the consideration of the number of levels the system covers all the NN architectures space.

TABLE II: DISTRIBUTION OF MATRIXES IN THE SAMPLE.

matrixes	dimension	x value
113	4x4	1,80618
114	8x8	3,31133
92	16x16	6,0206
79	32x32	11,13811
68	64x64	21,0721
69	128x128	40,639049
56	256x256	79,471919
41	512x512	156,836628

VI. EXAMPLE

GANET system has been applied to determine the simplest feed-forward neural network able to approximate the logistic time series. The logistic map is given by Equation 5.

$$x_{t+1} = \lambda \cdot x_t \cdot (1 - x_t) \qquad (5)$$

Where λ values determine the behavior of the series. In these experiments a value of λ=3.97 and x(0) = 0.5 are used, and the map describes a strongly chaotic time series. The use of the logistic map has two main advantages, firstly, the chaotic behavior and its prediction is a non trivial task and secondly, the optimal NN to predict the map is known. As the value of the map at instant *t* depends only on the value of the map at instant *t-1*, the optimal network has to consider the input carrying the *t-1* signal. In order to increase the complexity of the problem and to test the ability of the method to generate good architectures, five inputs have been taken into account: x_{t-4}, x_{t-3}, x_{t-2}, x_{t-1}, x_t. Thus, the system must find the most relevant input variables in the dynamic behavior of the logistic time series, besides the minimum number of hidden neurons to solve the problem.

In this work, a fitness evaluation has been developed in order to avoid the definition of a function relating the testing error and size of the NN as it could be used in [13, 14]. Then, the fitness function used in this work is given by the inverse (multiply by 10^6 to scale the value) of the number of calculations to adapt the weights of the neural network, when the error is below 0.008. When the error is up to 0.008 a maximum value of calculations is used (10^{10}). The fitness function is compiled in Equation 6.

$$f = \begin{cases} \frac{10^6}{N} & if \quad error < 0.008 \\ \frac{10^6}{10^{10}} & if \quad error \geq 0.008 \end{cases} \qquad (6)$$

where f is the fitness function, N is the number of calculations and *error* is the training error of the net.

The experimental parameters are as follows: a maximum of 15,000 learning cycles, 200 generations of GA, population size of 50 individuals, parent population of 35 individuals, elitism percentage 10% and is increased a 2% each 40 generations, mutation 1%, 95 patterns (100 iterations of logistic series with x_0=0.5 and λ=3.97 with six decimals) and learning factor α=0.9. The number of experiments executed is 13. In Figure 11, the evolution of the average value (over 13 executions) of the fitness function is shown. Figure 12 and 13 depicts the average of error and size.

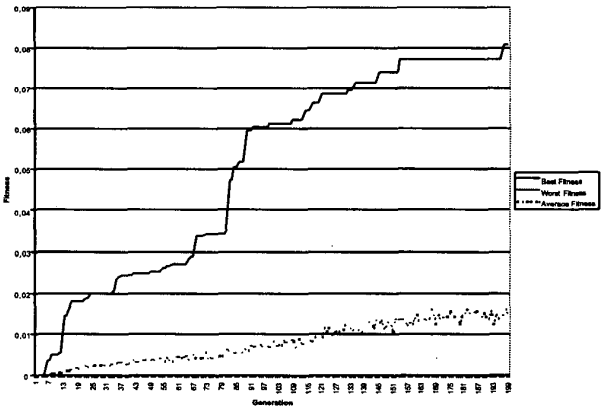

Figure 11: Evolution of fitness average.

The evolution of fitness of Figure 11 shows the usual behavior of an evolutionary computation technique that maximizes an objective function. In Figure 12, the tendency of the error is a decreasing curve in best, worst and mean for the population. The noise of the curve is the result of randomly initialization procedure of initial weights. For each initialization, the error of the NN is different. The best individual is the individual with the best fitness value, and it could be seen in Figure 12 is not

the best at the beginning of the evolution, but at the end the error is under the limit of the fitness function. The evolution of the NN size is also a decreasing curve, see Figure 13. The behavior of the best individual shows that at the beginning, the evolutionary process minimizes the error but not the size, when the error is under the threshold of 0.008 the evolutionary process minimizes the size of the net.

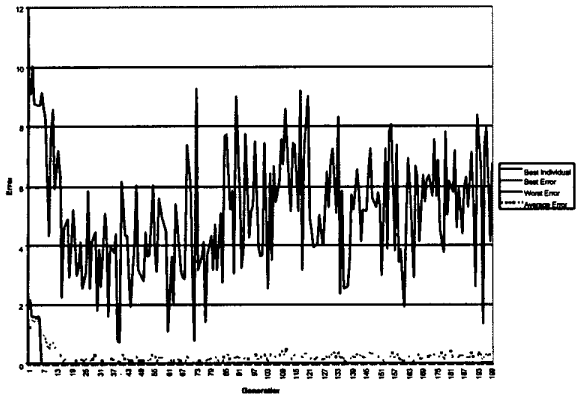

Figure 12: Evolution of error average.

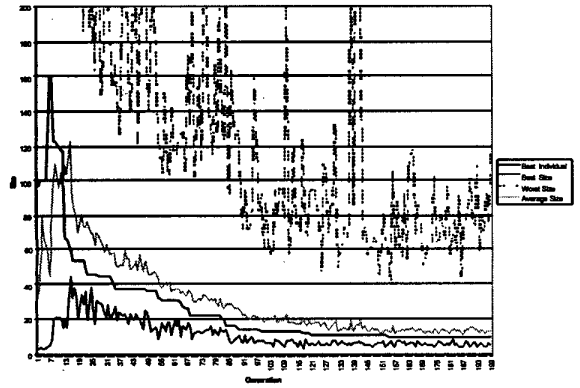

Figure13: Evolution of size average.

VII. CONCLUSIONS

The proposed method is a successful extension of Kitano's works [9]. An automatic procedure to design neural networks architectures has been developed by means of the two-dimensional grammar evolution. The considered grammars allow overcoming Kitano's restrictions that affect to the grammar contents, the architecture of NN's and the genetic operators. In this work these three restrictions are eliminated. The grammars include recursion. The number of NN nodes (constant in Kitano's work, determined through the word size and fixed also in the genotype) is evolved by GANET systems in an optimal way (the size of the final word is variable). In GANET the genetic operators, in particular crossover, are generic and are not necessary to redefine them as happens in Kitano's work. Moreover, this generalization of Kitano's work extends the capability of the automatic procedure without loosing the quality of the obtained nets and without dismissing the results of GA.

VIII. REFERENCES

[1] S. Harp, Samad T. and Guha A. Towards the Genetic Synthesis of Neural Networks. Proceedings of the third International Conference on Genetic Algorithms and their applications, pp 360-369, San Mateo, CA, USA, 1989.

[2] G.F. Miller, P.M. Todd and S.U. Hegde. Designing neural networks using genetic algorithms. In Proc. of the third international conference on genetic algorithms and their applications, pp 379-384, San Mateo, CA, USA, 1989.

[3] S. Harp, Samad T. and Guha A. Designing Application-Specific Neural Networks using the Genetic Algorithm, Advances in Neural Information Processing Systems, vol2, 447-454, 1990.

[4] F. Gruau. Genetic Synthesis of Boolean Neural Networks with a Cell Rewriting Developmental Process. Proc. of COGANN-92 International Workshop on Combinations of Genetic Algorithms and Neural Networks, pp. 55-74, IEEE Computer Society Press, 1990.

[5] F. Gruau. "Neural Network Synthesis Using Cellular Encoding and the Genetic Algorithm". PhD Thesis, Ecole Normale Supèrieure de Lyon, (1994).

[6] T. Ash. Dynamic Node Creation in Backpropagation Networks ICS Report 8901, The Institute for Cognitive Science, University of California, San Diego (Saiensu-sh, 1988), 1988.

[7] F. Gruau. Automatic Definition of Modular Neural Networks. Adaptive Behaviour, vol. 2, 3, 151-183, 1995.

[8] D.B. Fogel, Fogel L.J. and Porto V.W. Evolving Neural Network, Biological Cybernetics, 63, 487-493, 1990.

[9] H. Kitano. Designing Neural Networks using Genetic Algorithms with Graph Generation System, Complex Systems, 4, 461-476, 1990.

[10] J.W.L. Merril and R.F. Port. Fractally configured Neural Networks. Neural Networks, 4, 53-60, 1991.

[11] Valls J.M., Galván I.M. and Molina J. M. (2000),"Multiagent System for designing optimal Radial Basis Neural Networks", Information Processing and Management of Uncertainty in Knowledge Based Systems. Spain.

[12] Hartz J., Krough A. and R.G. Palmer (1991). "Introduction to the Theory of Neural Computation". Addison-Wesley.

[13] Molina, J. M., Galván, I., Isasi, P., Sanchis, A. Grammars and Cellular Automata for Evolving Neural Networks Architectures, IEEE International Conference on Systems, Man and Cybernetics. pp. 2497-2502. USA 2000.

[14] M. A. Guinea, A. Sanchis, J. M. Molina. "Characteristics of Grammars Codification for Evolution of NN Architectures", IASTED International Conference Artificial Intelligence and Applications (AIA 2001), pp 72-76. España, Septiembre, 2001.

[15] Goldberg D.E. (1989). Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley, New York

[16] Chomsky N. (1959)."On Certain Formal Properties of Grammars", Information and Control 2,137-167.

[17] Hopcroft J.E. and Ullman J.D. (1979). Introduction to Automata Theory, Languages and Computation, Addison-Wesley.

[18] Ehring, Nagel, Rozemberg and Rosenfeld (1987)."Graph Grammars and their Applications to Computer Science", Lecture Notes in Computer Science.