Generative Capacities of Cellular Automata Codification for Evolution of NN Codification

Germán Gutiérrez, Inés M. Galván, José M. Molina, and Araceli Sanchis

SCALAB. Departamento de Informática. Universidad Carlos III de Madrid.
Avda. Universidad 30, 28911-Leganés (Madrid)-SPAIN.
{ggutierr,igalvan, masm}@inf.uc3m.es,molina@ia.uc3m.es

Abstract. Automatic methods for designing artificial neural nets are desired to avoid the laborious and erratically human expert's job. Evolutionary computation has been used as a search technique to find appropriate NN architectures. Direct and indirect encoding methods are used to codify the net architecture into the chromosome. A reformulation of an indirect encoding method, based on two bi-dimensional cellular automata, and its generative capacity are presented.

1 Introduction

The architecture design is a fundamental step in the successful application of Artificial Neural Networks (ANN), and it is unfortunately still a human experts job. Most of the methods are based on evolutionary computation paradigms, Evolutionary Artificial Neural Networks (EANN). A wide review of using evolutionary techniques to evolve different aspects of neural networks can be find in (Yao, 1999).

The interest of this paper is focused on the design of Feedforward Neural Networks (FNN) architectures using genetic algorithms. There are two main representation approaches for codification of FNN in the chromosome to find the optimal FNN architecture. One, based on the complete representation of all the possible connections, direct encoding, relatively simple and straightforward to implement. But large architectures, for complex tasks, requires much larger chromosomes (Miller et al., 89; Fogel, 1990; Alba et al., 1993). Other based on an indirect representation of the architecture, indirect encoding schemes. Those schemes consists of codifying, not the complete network, but a compact representation of it, avoiding the scalability problem and reducing the length of the genotype. (Kitano, 1990; Gruau, 1992;Molina et al., 2000).

In this work, an indirect constructive encoding scheme, based on cellular automata (Wolfram, 1998), is reformulated. Two bi-dimensional cellular automata are used to generate FNN architectures proposed. It is inspired on the idea that only a few seeds for the initial configuration of cellular automata can produce a wide variety of FNN architectures, generative capacity. And this generative capacity, the search space of NN covered by cellular encoding, is shown too.

2 Description of Cellular System

The global system is composed of three different modules: the Genetic Algorithm Module, the Cellular Module and the Neural Network Module (Fig 1). All the modules are related to make a general procedure. The cellular module is composed of two bi-dimensional cellular systems and takes charge of generating FNN architectures. Initial configurations of cellular systems are given by several seeds and the rules of the systems are applied to generate final configurations, which correspond to a FNN architecture. The generated FNN is trained and relevant information about the goodness of FNN is used as the fitness value for the genetic module. The genetic algorithm module takes charge of generating the positions of the seeds (codified in the chromosome) in the two-dimensional grid of cellular systems, which determine initial configurations of cellular systems. In [Gutierrez et. al., 2001] a detailed description of Neural Network and Genetic Algorithm Modules can be found. In the next section, the new formulation of the Cellular Module is presented.

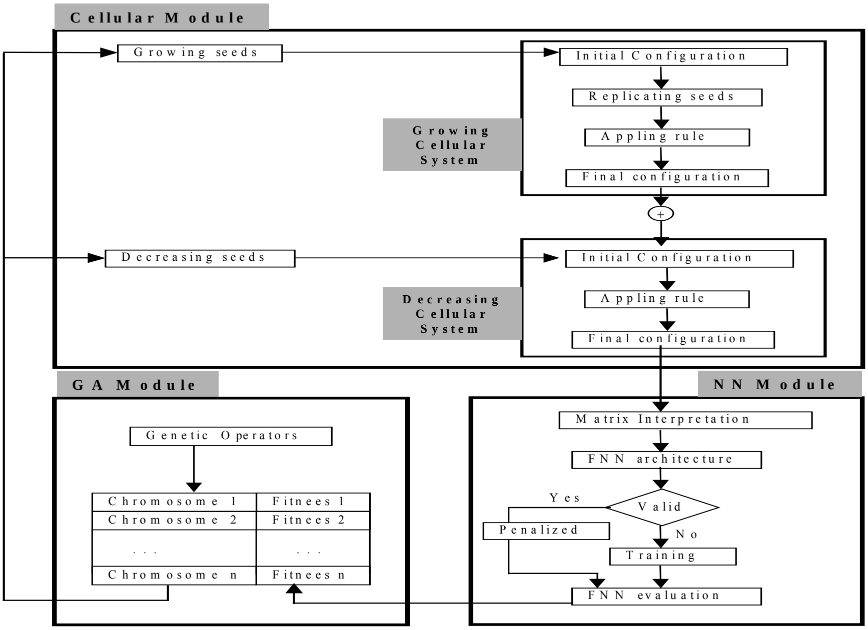

Fig. 1. System's architecture and modules relationship.

3 Cellular Module

The cellular module (Fig 1) is composed by two bi-dimensional cellular system. "Growing cellular system" and "decreasing cellular system". The bi-dimensional cellular systems consist of a regular grid of cells. Each cell takes different values, denoted by a_{ij}., and the system is updated in discrete time steps according to some rule that depends on the value of sites in some neighbourhood around it. In this work, the neighbourhood structure is defined by the square region around a cell. The size of the grids, $Dim_x xDim_y$, for the cellular systems is previously fixed. Dim_x (rows) is equal to the number of input neurons N, plus the number of output neurons M, and Dim_y (columns) corresponds with the maximum number of hidden neurons. In the next, the initial configurations, possible values of each cell and the evolution rules of the growing and decreasing cellular systems are presented.

3.1 Growing Cellular System

The growing cellular system is designed in order to obtain FNN architectures with a large number of connections between the input and hidden layer and between hidden and output layer. The initial configuration of the growing cellular system is given by n seeds, ($s_1,s_2,...,s_n$), called " growing seeds" (GS). Each seed is defined by two coordinates which indicates the positions of the seed in the grid. That positions are provided by the genetic algorithm module. In order to apply the automata rule the first time each seed is replicated over its quadratic neighbourhood, in such a way that if a new seed has to be placed in a position previously occupied by another seed, the first one is replaced. Thus, the value a_{ij}. of a cell can take two possible values: $a_{ij}.=0$ when the cell is inactive and $a_{ij} =s_k$ if the cell contains the seed s_k.

The rule of the growing cellular system has been designed to allow the reproduction of growing seeds. The idea is to copy a particular growing seed s_k when a cell is inactive and there are at least three identical growing seeds in its neighbourhood. The rule of the growing automata cellular is defined in equation 1:

$$a_{i,j}^{(t+1)} = s_k \text{ if } a_{i,j}^{(t)} = 0 \text{ AND} \qquad (\text{eq } 1)$$

$$\begin{aligned}
&a_{i-1,j-1}^{(t)} = a_{i-1,j}^{(t)} = a_{i-1,j+1}^{(t)} = s_k \text{ OR } a_{i+11,j-1}^{(t)} = a_{i+1,j}^{(t)} = a_{i+1,j+1}^{(t)} = s_k \text{ OR}\\
&a_{i-1,j-1}^{(t)} = a_{i,j-1}^{(t)} = a_{i+1,j-1}^{(t)} = s_k \text{ OR } a_{i-1,j+1}^{(t)} = a_{i,j+1}^{(t)} = a_{i+1,j+1}^{(t)} = s_k \text{ OR}\\
&a_{i-1,j-1}^{(t)} = a_{i-1,j}^{(t)} = a_{i,j-1}^{(t)} = s_k \text{ OR } a_{i-1,j}^{(t)} = a_{i-1,j+1}^{(t)} = a_{i,j+1}^{(t)} = s_k \text{ OR}\\
&a_{i,j-1}^{(t)} = a_{i+1,j-1}^{(t)} = a_{i+1,j}^{(t)} = s_k \text{ OR } a_{i+1,j}^{(t)} = a_{i+1,j+1}^{(t)} = a_{i,j+1}^{(t)} = s_k
\end{aligned}$$

$$a_{i,j}^{(t+1)} = a_{i,j}^{(t)} \text{ in other case}$$

According to that rule, a seed s_k is reproduced when there are at least three identical growing seeds in its neighbourhood, which must be located in the same row, or in the same column or in the corner of the neighbourhood.

3.2 Decreasing Cellular System

Once the growing cellular system is expanded, most of the cells in the grid are occupied by growing seeds. If the presence of a growing seed is considered as the presence of a connection in the network, could be convenient to remove seeds in the grid in order to obtain a large variety of architectures. Hence, the decreasing cellular system is incorporated to remove connections. The initial configuration of the decreasing cellular system is given by the final configuration of the growing cellular system and by m seeds (d_1,... d_m), called "decreasing seeds" (DS). Each seed is defined also by two coordinates and they are provided by the genetic algorithm module. The value a_{ij} of a cell in this automata can be: $a_{ij} =0$ when the cell is inactive; $a_{ij} =s_k$ when the cell contains the growing seed s_k and $a_{ij} =d_r$ if the cell contains the decreasing seed d_r.

The rule of the decreasing cellular system is designed to remove growing seeds in the grid. A growing seed s_k is removed when two contiguous neighbouring cells contain identical growing seeds and another neighbouring cell contain a decreasing seed. The rule of the decreasing cellular system is defined as:

$$
\begin{aligned}
a_{i,j}^{(t+1)} = d_r \text{ if } & (a_{i,j}^{(t)} = a_{i-1,j-1}^{(t)} = a_{i,j-1}^{(t)} = s_k \text{ AND } a_{i-1,j}^{(t)} = d_r) \text{ OR} \\
& (a_{i,j}^{(t)} = a_{i-1,j-1}^{(t)} = a_{i-1,j}^{(t)} = s_k \text{ AND } a_{i,j-1}^{(t)} = d_r) \text{ OR} \\
& (a_{i,j}^{(t)} = a_{i-1,j+1}^{(t)} = a_{i,j+1}^{(t)} = s_k \text{ AND } a_{i-1,j}^{(t)} = d_r) \text{ OR} \\
& (a_{i,j}^{(t)} = a_{i-1,j}^{(t)} = a_{i-1,j+1}^{(t)} = s_k \text{ AND } a_{i,j+1}^{(t)} = d_r) \text{ OR} \\
& (a_{i,j}^{(t)} = a_{i,j-1}^{(t)} = a_{i+1,j-1}^{(t)} = s_k \text{ AND } a_{i+1,j}^{(t)} = d_r) \text{ OR} \\
& (a_{i,j}^{(t)} = a_{i,j+1}^{(t)} = a_{i+1,j+1}^{(t)} = s_k \text{ AND } a_{i+1,j}^{(t)} = d_r) \text{ OR} \\
a_{i,j}^{(t+1)} = a_{i,j}^{(t)} \text{ in other case} &
\end{aligned}
\qquad \text{(eq 2)}
$$

Similar rules could be used, but the design must enforce that not all growing seed in the grid are removed.

3.3 Evolving Growing and Decreasing Cellular System

In order to evolve and to combine both growing and decreasing cellular systems, a special procedure that allows the convergence toward a final configuration is proposed (see Fig 1):

1. All cells in the grid are set to the inactive state and the growing seeds provided by the genetic module are located in the grid. The growing seeds are replicated over their quadratic neighbourhood.
2. The rule of the growing cellular system is applied until no more rule conditions could be fired and a final configuration is reached.
3. The decreasing seeds are placed in the grid.
4. The rule of the decreasing cellular system is applied until the final configuration is reached.
5. A binary matrix M is finally obtained, replacing the growing seeds by an 1 and the decreasing seeds or inactive cells by a 0. That matrix will be used by the neural network module to obtain a FNN architecture.

4 Experimental Results

In this paper, the cellular approach has been tested for the parity problem. The fitness function provided to the genetic algorithm module is the inverse of computational effort, equation 3 (a). Where "c" is the number of connections in the FNN architecture and "t_c" the number of training cycles carried out. If the network doesn't reach the defined error, it is trained a maximum of cycles and the fitness value associated is given by the equation 3 (b), where "$e_{reached}$" is the error reached and "e_{fixed}" the error previously fixed.

The parity problem is a mapping problem where the domain set consists of all distinct N-bit binary vectors and the results of the mapping indicates whether the sum of N components of the binary vector is odd o even. In most current studies (Sontag, 1992) shown that a sufficient number of hidden units for the network is (N/2) +1 if N is even and (N+1)/2 if N is odd. In this work parity seven has been considered as a study case. Hence, the network will have 7 input neurons and 1 output. Thus, the size of the grid would be 8x64.

The number of growing and decreasing seeds has been modified and the architectures obtained with the cellular approach for the different number of seeds after 100 generations have four hidden neurons and most of them are fully connected. Only in one case (5-5), the architecture is not fully connected, the first input neurons is only connected to one hidden neuron, without connections to the rest of hidden neurons. All of then obtain percentage of train and test errors around 90% and 80 %, respectively. When the direct encoding is used to find the optimal architecture, the length of the chromosome is 343 (7inputs x 49 hidden) and more complex architectures are obtained. After 300 generations the architecture has 48 hidden neurons and 48% of connectivity.

For the generative capacity of the method 10000 chromosomes are randomly generated, with 7 GS and 7 DS. And the nets obtained, indicating how much hidden nodes and connections has each one, are shown in Fig 2. For a FNN, with “H” hidden nodes, N inputs and M outputs there is a maximum (*H(N+M)*) and a minimum (*H*) number of connections, and it is displayed in Fig 2. The nets obtained cover the search space of FNN on the whole.

$$F = \frac{1}{(c \cdot t_c)} \text{ (a)} \qquad F = \frac{1}{c \cdot t_c} \cdot \left(e_{fixed} / e_{reached}\right) \text{ (b)} \qquad \text{(eq. 3)}$$

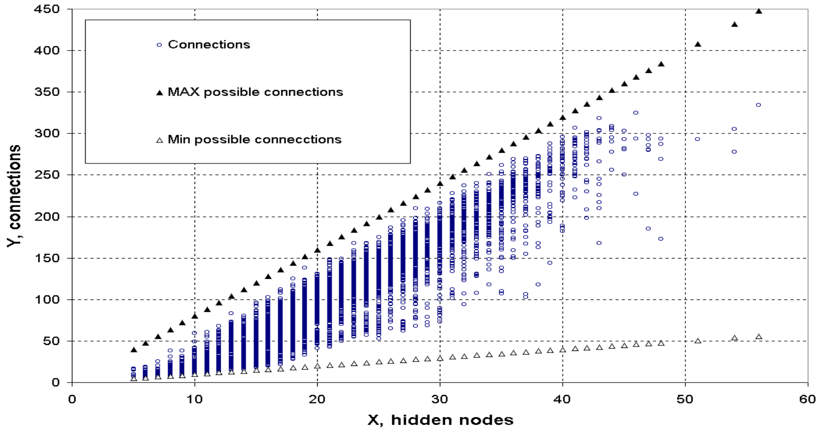

Fig. 2. Generative capacity for parity problem. A point represents a NN with X hidden nodes and Y connections

5 Conclusions and Future Work

Cellular automata are good candidates for non-direct codification's. The final representation has a reduced size and could be controlled by the number of seeds used. The results shown that the cellular scheme presented in this paper is able to find appropriate FNN architectures, and the nets obtained are independent of how many seeds (GS and DS) are placed in the CA. In addition, the number of generations over the population is less when the indirect encoding approach is used instead of direct codifications.

In future works not any individual in the population will have the same number of growing and decreasing seeds, i.e. a seed in the chromosome could be a growing seed or decreasing seed. Besides, some issues about Neural Network Module and fitness function used, i.e. how punish the nets to increase the search, will be studied in future works.

References

[Alba 1993] Alba, E., Aldana, J. F., and Troya, J. M.: Fully automatic RNA design: a genetic approach. X. Yao: Evolutionary Artificial Neural Networks 45 In Proc. of Int'l Workshop on Artificial Neural Networks (IWRNA'93), pages 399–404. Springer-Verlag, 1993. Lecture Notes in Computer Science, Vol. 686.

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

[Gruau, 1992] Gruau F. "Genetic Synthesis of Boolean Neural Networks with a Cell Rewriting Developmental Process". Proceedings of COGANN-92 International Workshop on Combinations of Genetic Algorithms and Neural Networks, pp. 55–74, IEEE Computer Society Press, 1992.

[Gutierrez et. al., 2001] G. Gutiérrez1, P. Isasi, J. M. Molina, A. Sanchís and I. M. Galván. Evolutionary Cellular Configurations for Designing Feed-Forward Neural Networks Architectures. Connectionist Models of neurons, Learning Processes and Artificial Intelligence. 6th International Work-Conference on Artificial Neural Networks, IWANN 2001. Proceedings, Part I. LNCS 2084. J. Mira, A Prieto (Eds.) Springer.

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

[Miller et al., 1989] Geoffrey F. Miller, Peter M. Todd, and Shailesh U. Hegde. Designing Neural Networks using Genetic Algorithms. In J. David Schaffer, editor, Proceedings of the Third International Conference on Genetic Algorithms, pages 379–384, San Mateo, California, 1989. Philips Laboratories, Morgan Kaufman Publishers, Inc.

[Molina et al., 2000] Molina, J. M., Torresano, A., Galván, I. M., Isasi, P., Sanchis, A.: Evolution of Context-free Grammars for Designing Optimal Neural Networks Architectures. GECCO 2000, Workshop on Evolutionary Computation in the Development of ANN. USA. Julio, 2000

[Sontag, 92] E. D. Sontag. Feedforward nets for interpolation and classification. Journal of Computer and System Sciences, 45, 1992.

[Wolfram, 1988] S. Wolfram. Theory and applications of cellular automata. World Scientific, Singapore, 1988.

[Yao, 1999] Yao, X. 1999. Evolving artificial neural networks, Proceedings of the IEEE vol. 87, no. 9, p.1423–1447.