P-System Communications Architectures Configuration based on Growing Self-Organizing Maps

Abraham Gutiérrez, Soledad Delgado, Sandra Gómez

Natural Computing Group - Universidad Politécnica de Madrid
Carretera de Valencia Km. 7, Madrid - 28031, Spain
{abraham,sole,sgomez}@eui.upm.es

Abstract. The different viable architectures that implement P-Systems (membrane systems) over distributed cluster of processors have a major drawback: the distribution of these architectures in a balanced tree of processors that can minimize external communications and maximize the parallelism grade. For a given P-System and K processors, there exists a great volume of possible distributions of membranes over these. In a recent paper the feasibility of using Self-Organizing Neural Networks (SONN) with growing capability to help in the selection process of a distribution for a given P-System has been demonstrated, although the nature of two-dimensional patterns used in the study limited the possibility of defining more flexible degrees of communication, making more difficult to locate the best distribution. In this paper the capacity of Growing Cell Structure (GCS) model of projecting high-dimensional spaces in bi-dimensional graphs is explored.

Keywords: Natural Computation, SONN Networks, GCS Networks.

I. INTRODUCTION

P-Systems, introduced by Păun [1], are a class of distributed, massively parallel and non-deterministic systems. This model has become, during last years, a powerful framework for developing new ideas in theoretical computation and connecting the Biology with Computer Science. Possibilities offered by P-Systems for solving NP-problems, in lineal time and of course lineal resources, have made researchers concentrate their work towards HW and SW implementations of this new computational model.

Nowadays, it is possible to find different viable architectures that implements P-Systems in a distributed cluster of processors [2]. These proposed architectures have reached a certain compromise between the massively parallelism character of the system and evolution step times. These architectures are based in the distribution of several membranes in each processor, the use of proxies to control the communication between membranes and mainly, the suitable distribution of the architecture in a balanced tree of processors. All this facts allows obtaining a better evolution step time than in others suggested architectures congested quickly by the network collisions when the number of membranes grows. The main problem in these architectures is to find the proper distribution of the membranes between processors and the definition of a network topology that minimizes communication between processors without reducing the system parallelization.

In a previous paper [3], we suggest the use of Self-Organizing Neural Networks (SONN) with growing capability, based in Fritzke work [5], to help in the search and selection of the balanced distribution for a given P-system, with the purpose of obtaining as a final objective the reduction of the run times of each step of evolution in a P-System, although the nature of two-dimensional patterns used in the study limited the possibility of defining more flexible degrees of communication, making more difficult to locate the best distribution. In this paper the capacity of Growing Cell Structure (GCS) model of projecting high-dimensional spaces in bi-dimensional graphs is explored. Specifically, we propose a more flexible definition of the internal and external communications degrees that occur in the processors, which basically needs the use of vectors with a dimension greater than two. In this case, it is necessary to work with the concept of topographic maps of the output layer of the GCS network to generate two-dimensional graphics that can be used to explore the high-dimensional input space. The proposed experiments have presented the opportunity of evaluating what kind of information about membrane-processor-communication is most appropriate in finding the best distribution when using the vector projections provided by the GCS network.

II. P SYSTEM COMMUNICATION ARCHITECTURES

The viable architectures that implements P-Systems in a distributed cluster of processors are based on the following:

Membranes distribution: In each processor, K membranes are located that will evolve, at worst, sequentially. The value of K is determined by the relation between the number of membranes M and processors P, where $K \geq 1$. The benefit obtained is that the number of the external communications decreases. The total number of communications splits in two classes: a group of internal communications for pairs of

membranes located in the same processor (the run time to carry out the internal communications will be negligible) and another group of external communications to interchange information among pairs of membranes located in different processors.

Proxy for processor: When a membrane wants to communicate with another one located at a different processor, the first one uses a proxy (programs or device located in the processor that carries out an action in representation of another), instead of doing it directly. This intermediate element located between the bus and the membranes concentrates and reduces the information that must be reported.

Tree topology of processors: The benefit obtained with the tree topology of processor is that it minimizes the total number of external communications made as the proxies interchange information only with its direct predecessor and its direct successors, and therefore the total number of external communications for P processors in each evolution step is $2(P-1)$.

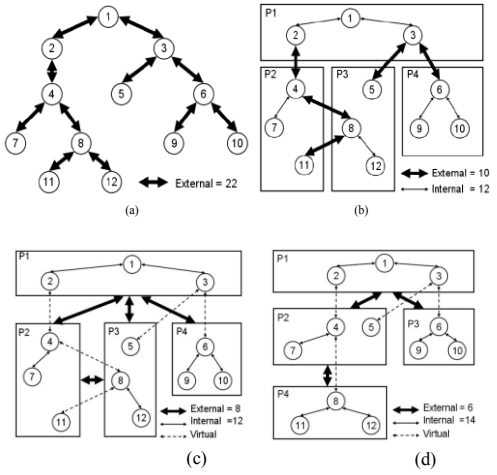

Fig.1. (a) P-System communications. (b) Communications with membranes distribution (c) Communications with a proxy for processor. (d) Communications using a tree topology

III. FRITZKE'S GROWING CELL STRUCTURES (GCS)

Self-Organizing Map (SOM) is an artificial neural network model with competitive and unsupervised training. SOM network has two main characteristics: it makes possible obtaining a simplified model of the training data (normally high-dimensional) and it has the capacity to project them on a two dimensional map that shows the existing relations among them. In 1982, Kohonen [4] proposed first model of SOM, where the complete network structure had to be specified in advance and remained static during all the training process. In 1994, B. Fritzke [5] proposed a SOM model called Growing Cell Structure (GCS) where this static structure limitation was eliminated.

GCS is a two-layer architecture network. Neurons located at the input layer are fully connected with those in the output one. These connections have associated a weight, w_{ij}, where i identify the input neuron and j the output one. There exist as many input neurons as dimension has the input vectors. Neurons in the output layer have neighbor connections between them presenting a topology formed by groups of basic k-dimensional hyper-tetrahedrons structures. In order to facilitate the visualization of the output layer, in this work a value of k=2 has been used.

Every output c unit has an n-dimensional synaptic vector $w_c = (w_{1c}, \ldots, w_{nc})$ associated. This vector can be seen as the position of c in the input vector space. Each time a new input pattern $e = (e1, \ldots, en)$ is processed, only one output neuron is activated, called the *best matching unit (bmu)*, that is the one with the synaptic vector that matches best with the input pattern. Formally:

$$S_{bmu} = \arg\min \|e - w_c\| \quad (1)$$

Thereby $\|\cdot\|$ denotes the Euclidean vector norm. By this the input vector space is partitioned into a set of regions, each consisting of the locations having a common nearest synaptic vector. This way, the set of all synaptic vectors of the output layer can be seen as a simplified model of the input vector space.

The training phase in GCS network adapts synaptic vectors looking for that each output neuron represents a group of similar input patterns. At the beginning of the training phase the output layer of the network has only three neurons interconnected via neighbor relations ($k=2$). During the training process a set of input patterns is presented to the network iteratively. In each adaptation step an input pattern is processed, the *bmu* is calculated and its synaptic vector and its topological neighbor's synaptic vectors are modified using equations 2 and 3 respectively (where $\varepsilon_b > \varepsilon_n$).

$$\Delta w_{bmu} = \varepsilon_b (e - w_{bmu}) \quad (2)$$

$$\Delta w_c = \varepsilon_n (e - w_c) \text{ (for all } c \text{ neigbor of } bmu) \quad (3)$$

After a fixed number of adaptation steps a new output unit is inserted and is connected to other cells in such a way that the triangular groups of neighbor units are guaranteed. Periodically superfluous neurons are removed in order to obtain better results when input space consists of several separate regions of positive probability density. A constant threshold, η, is used to eliminate those neurons with probability density below this value. The removal process ensures the triangular architecture of the output layer, but the output neighbor mesh can results broken in several sub-meshes. In this work the modification of the GCS training algorithm proposed in [6] has been used in order to achieve a better interpretation of the removal parameters.

In a trained network the output layer map can be seen as a projection of the input vector space in a bi-dimensional plane that exhibits the relations of the

input patterns. Printing the output layer map, called topographic map, data inherent knowledge can be discovered. As it has been exposed previously, when *k*=2 architecture factor is used, the GCS output layer is organized in groups of interconnected triangles. In spite of bi-dimensional nature of these meshes, it is not obvious how to embed this structure into the plane. In this paper the projection technique exposed in [7] has been used to generate the topographic map. Using this methodology, traditional Kohonen visualization methods can be implemented using GCS networks. The following GCS visualization methods have been used: distance map, Unified map (U-map), distance addition map, and component planes [7].

IV. EXPERIMENTS AND RESULTS

Several experiments have been carried out to validate GCS networks as a tool to help in discovering the best membrane distribution over a set of processor in a concrete P-System. In particular, thirty P-System models generally used in the literature of P-System have been employed in this study. Several membrane distributions between a concrete number of processors (calculated using eq. 5) have been generated for every P-System, observing how the resulting communications of the distribution affect to the parallelization grade. A great volume of GCS networks have trained using this information with the intention of visualize the output layer and establish the optimal distribution, which will be the one that balances the degrees of external communications and parallelization.

The minimal execution time of a complete P-System [2] is expressed in the formula:

$$T_{\min} = 2\sqrt{2\,M\,T_{apl}\,T_{com}} - 2T_{com}. \qquad (4)$$

Where T_{apl} is the maximum time used by the slowest membrane in applying its rules, T_{com} is the maximum time used by the slowest membrane for communication, and M is the number of membranes. The optimal number of processors that minimizes the execution time of a complete P-System is calculated by:

$$P_{opt} = \sqrt{\frac{M\,T_{apl}}{2T_{com}}}. \qquad (5)$$

For each P-System three families of data have been generated. The first consists of one vector for membrane-processor-distribution, where each one maintains the degree of internal and external communications that generates one specific membrane in a concrete processor for one of the combinations. In this case vectors are bi-dimensional, and for a particular combination there exists as many vectors as membranes has the P-System. In this first family of data each vector is labeled with a string that identifies *i* membrane, *j* processor, and *k* combination ($M_iP_jC_k$). In the second group of data a vector by combination has been generated, whose dimension corresponds with the double of existing membranes. For each membrane the degrees of internal and external communications are included in the corresponding vector. In this case each pattern is labeled only with the *k* combination that represents (C_k). Finally, the third group of data consists of a vector for processor-distribution, which includes six components that maintain different levels (high, medium and low) of internal and external communication obtained using fuzzy logic. Each vector is labeled to identify the *j* processor and the *k* combination (P_jC_k).

The goal we seek training a GCS network with any of these three groups of data is to obtain an arrangement of the vectors that allows us to analyze graphically, using topographic maps, the volume of internal and external communications that take place for each combination, been able to determine the best that balances the degree of parallelization and external communications. After the training phase the output units of the GCS network can be labeled with the union of the tags of all those input patterns that fall inside its Voronoi region. For space reasons this section only shows some of the experiments and results obtained for a particular P-System (Fig. 1a) which consists of 12 membranes. Fifteen feasible combinations, which fulfill the condition of tree topology of processors, of 12 membranes have been generated for 4 processors (the optimal number calculated by eq. 6). The volume of communications produced by membranes is not homogeneous. Using the 15 combinations three families of data previously described have been created. For the first group, each pair of internal-external communication of a membrane forms a vector (altogether there are 180 vectors). For the second group there are 15 vectors of 24 dimensions. Finally, the last family of data has been generated using fuzzy logic values to define 15 vectors of 6 dimensions for each processor.

Using these three sets of data several GCS networks have been trained. The common learning parameters used in all the experiments are: LEAE insertion criterion, ε_b = 0.06 (*bmu* adaptation rate), ε_n = 0.002 (*bmu* neighbor adaptation rate), λ=1 epoch (number of iterations to insert a new unit). The only varying factor involving experiments is related with the concluding condition of the training process, fulfilled when the output layer gets a specific number of output units or when at least an explicit number of isolated clusters of output units are obtained. For the first, a removal threshold μ = 0 has been used, and for the last a value of μ=0.0006.

Figure 2 shows the scattergram of a GCS network, trained with the first group of bi-dimensional data (concluded when at least 6 isolated clusters of output units are obtained), where the position of each output unit is determined by the two components of its synaptic vector (lines between points represents the

output layer neighbor connections). X-axis coordinates indicates the internal degree of communications and Y-axis the external one. We have grouped the neurons into three classes: bad (from 1 to 11), medium (from 14 to 18) and good (from 19 to 28). Within each of these three groups we have ordered neurons from highest to lowest level of external communication and for those with a similar value, from highest to lowest level of internal communication. Based on this information has been determined that the best distribution is the C13, that contains 1 bad neuron, 4 medium neurons and 7 good neurons. Moreover, this distribution has one of the best ratios of communication.

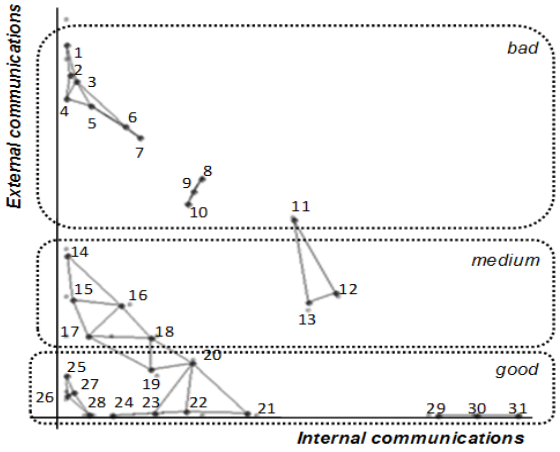

Fig.2. Scattergram of the GCS network trained with bi-dimensional patterns. Points represent neurons and lines between them neighbor connections.

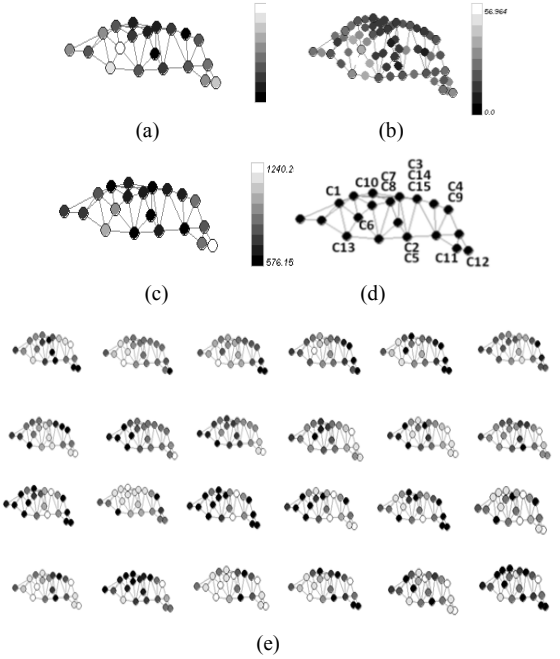

Fig.3. GCS network trained with 24-dimensional vectors. a) Distance map. b) U-map. c) Distance addition map. d) Labeled topographic map. e) Component planes.

Figure 3 includes some of the topographic maps of the GCS network trained with the 24-dimensional data. In this experiment, the learning concluding condition was determined when the output layer got 20 output units. Distance map, U-map, Distance addition map and labeled topographic map show certain grade of clustering associated with the patterns C2, C3, C5, C7, C8, C10, C14 and C15, C11 with C12, C4 with C9, and finally, C1, C6 and C13 seems to be isolated. Component planes in the first and second rows show the internal and external communications associated to the membranes 1 to 12 from left to right, respectively, where black color represents low values. The high dimension of the vectors make no easy to find the combination that better minimizes the degree of external and internal communications. The C13 combination that results the better one in the previous experiment, continues showing good characteristics, although the C1 or C6 is perhaps better.

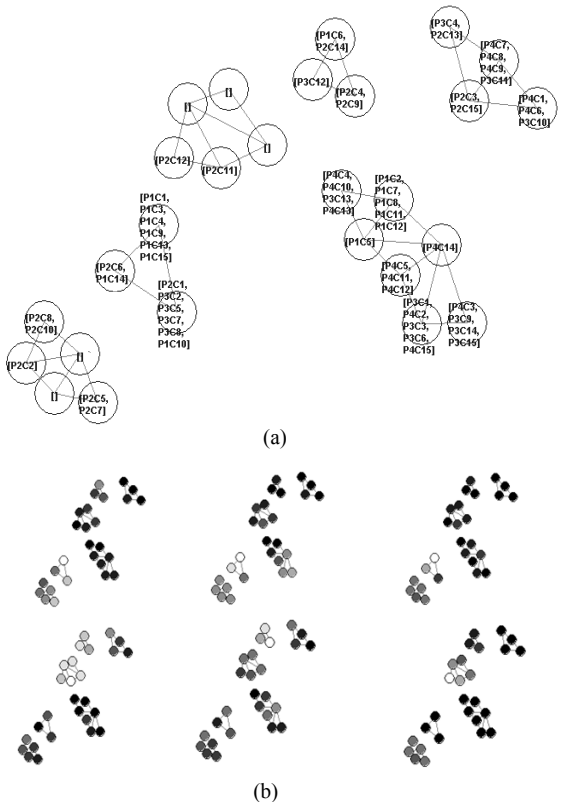

Fig.4. GCS network trained with 6-dimensional vectors. a) Labeled topographic map. b) Component planes.

Finally, Figure 4 shows the information of the GCS network trained with the third set of data (concluded when at least 6 isolated clusters of output units are obtained). In this network only 6 component planes exist, simplifying its visual analysis. Component planes in the first row show the high, medium and low internal communications and the high, medium and low external communication appear in the second one,

where black color represents low values. The two clusters in the upper left are the worst external communications high, medium and low show (although the grades for internal communications are good). The two clusters below on the left offer good levels of external communications, but they have the worst internal communications levels. The two remaining clusters get the best ratios by minimizing both types of communications. Each of the 15 combinations consists of 4 patterns (one per processor). The combination that has more patterns in the two best clusters is the C13 again (it has 3 out of 4).

With the first set of data the problem that arises in searching the best distribution is the dispersion of the vectors that compose a concrete combination of membranes by processor. Although this example is relatively simple, with only 12 membranes, it lets patent the complication to determine which the best distribution is. With respect to the second data set, although it solves the previous problem because there only exists a single vector by combination, the high dimension of the input space makes difficult the component plane analysis. The third data set raises a commitment between the first two sets, presenting so many vectors by combination as processors exist. The volume of processors is not usually high, because it would increase the volume of external communications, making easy to find in the graphs all the patterns associated to a concrete combination. On the other hand, the dimension of these vectors facilitates the analysis of the component planes.

V. CONCLUSIONS

GCS networks have demonstrated to be a useful tool to P-System in the searching of membrane balanced distributions. Although the example that has been used to document the methodology has a small volume of membranes, the feature of simplified model associated to GCS networks allows working with high volumes of membranes where the distribution possibilities go off.

The analysis of the information of a GCS network could be automated for feeding a system of automatic membrane distribution over processors. In particular, this tool is being adapted to be used in the distributed system of membranes based on microcontrollers exposed in [8][9].

The experiments have showed that the input patterns family that better display a P-System communication information is the third, where for a given combination four patterns are generated to describe a fuzzy characterization of the internal and external communications produced in a particular processor for a specific combination.

VI. REFERENCES

[1] Gh.Păun (1998), Computing with membranes. Journal of Computer and System Sciences, 61 (2000), and Turku Center for Computer Science-TUCS Report nº 208.

[2] A. Tejedor, L. Fernandez, F. Arroyo, et al (2007), An architecture for attacking the bottleneck communication in P Systems. LNCS, ISSN 1433-5298, Artificial Life and Robotics vol. 12, Springer Japan, pp. 236-240.

[3] A. Gutiérrez, S. Delgado y L. Fernandez (2009), Suitability of Using Self-Organizing Neural Networks in Configuring P-System Communications Architectures, LNCS vol. 5507, ICONIP 2008, Part II, Springer-Verlag Berlin Heidelberg, pp. 437–444.

[4] T. Kohonen (1982), Self-Organized Formation of Topologically Correct Feature Maps. Biological Cybernetics, Vol. 4359--69.

[5] B. Fritzke (1994), Growing Cell Structures – A Self-organizing network for Unsupervised and Supervised learning. Neural Networks, Vol. 7, no.1. 1441--60.

[6] S. Delgado, C. Gonzalo, E. Martinez and A. Arquero (2004), Improvement of Self-Organizing Maps with Growing Capability for Goodness Evaluation of Multispectral Training Patterns. Geoscience and Remote Sensing Symposium, IEEE International. Vol. 1, 564-567.

[7] S. Delgado, C. Gonzalo, E. Martinez and A. Arquero (2007), Visualizing High-Dimensional Input Data with Growing Self-Organizing Maps. LNCS, vol. 4507, Springer, Berlin, 580-587.

[8] A. Gutierrez, L. Fernández, F. Arroyo, et al (2008), Hardware and Software Architecture for Implementing Membrane Systems: A Case of Study to Transition P Systems. LNCS, Springer, Berlin, DNA Computing 211-220.

[9] A. Gutierrez, L. Fernández, F. Arroyo, et al (2008), Suitability of Using Microcontrollers in Implementing new P System Communications Architectures. LNCS, ISSN 1433-5298, Artificial Life and Robotics vol. 13, Springer Japan, pp. 102-106.