Suitability of Using Self-Organizing Neural Networks in Configuring P-System Communications Architectures

Abraham Gutiérrez, Soledad Delgado, and Luis Fernández

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

Abstract. Nowadays, it is possible to find out different viable architectures that implements P Systems in a distributed cluster of processors. These proposed architectures have reached a certain compromise between the massively parallelism character of the system and the evolution step times. They 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. For a given P-system and K processors, there exists a great volume of possible distributions of membranes over these. The main disadvantage related with these architectures is focused in the selection of the distribution of membranes that minimizes the external communications between them and maximizes the parallelism grade. In this paper, we suggest the use of Self-Organizing Neural Networks (SONN) with growing capability to help in this selection process for a given P-system.

1 Introduction

Possibilities offered by Natural Computation and, specifically P-Systems, for solving NP-problems, have made researchers concentrate their work towards HW and SW implementations of this new computational model. Transition P-Systems were introduced by Păun [1], and were inspired by "*basic features of biological membranes*". One membrane defines a region where there are a series of chemical components (*multisets*) that are able to go through chemical reactions (*evolution rules*) to produce other elements. Inside the region delimited by a membrane can be placed other membranes defining a complex hierarchical structure that can be represented as a tree. Generated products by chemical reactions can remain in the same region or can go to another region crossing a membrane.

A P-System is a computational device which is an abstract representation of a particular membrane structure. Each region is populated by a *multiset* of symbols. These *multisets* are materialized as strings of symbols. In addition, each region is associated with a set of rewriting rules. These rules are applied to the *multisets* (strings of symbols) of certain compartments and, consequently, change the system's configuration. The system configurations are determined by the membrane structure and *multisets* present inside membranes. The rules are applied simultaneously by observing the so-called *maximal parallelism* principle, that is the rules are selected in such a way that only "optimal" output is yielded. When it is not possible to apply any rule, the

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 437–444, 2009.

P-System halts. A designated compartment, called the output compartment, contains the output of the computation, which is equal to the cardinality of the *multiset* contained in it.

In the formal Transition P-Systems model can be distinguished two phases in each evolution step: rules application and communication. Once application rules phase is finished, then it begins communication phase, where those generated *multisets* travel through membranes towards their destination in case it is another region. These systems carry out computations through transitions between two consecutive configurations, what turn them into a computational model with the same capabilities as Turing machines.

Power of this model lies in the fact that the evolution process is massively parallel in application rules phases as well as in communication phase. The challenge for researchers is to achieve hardware and/or software implementations of P systems respecting the massively parallelism in both phases.

Nowadays, it is possible to find out at least three different viable architectures that implements P Systems in a distributed cluster of processors: P2P [2], Hierarchical P2P [3] and Master-Slave [4]. These proposed architectures have reached a certain compromise between the massively parallelism character of the system and evolution step times. In particular, they have focused in the second phase of an evolution step and have obtained good results in the throughput in the external communication among processors and parallelization levels of the system. 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. These solutions avoid communication collisions, and reduce the number and length for communication among membranes. All this facts allows obtaining a better step evolution time than in others suggested architectures congested quickly by the network collisions when the number of membranes grows. The main disadvantage related with these architectures is the great volume of possible combinations of membrane distributions over a number of processors that can vary from one to the number of membranes. If one processor is used, all of membranes run in the same processor, so external communications is reduced to zero but parallelization level disappears (all internal communications are sequential). On the other side, if each membrane runs in one processor, the better parallelization level is obtained but the increase of external communication produces network congestion and the worst evolution step times. The best solution is the balanced one where internal and external communications remain equilibrated.

In this paper, 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 this P System.

2 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 and another group of external communications to interchange information among pairs of membranes located in different processors. Therefore, the number of external communications against the previous model will always be smaller. Moreover, this is an important fact because the run time to carry out the internal communications will be negligible.

For example, the *22* external communications performed by an architecture with a membrane located in each processor (figure 1.a) have been reduced to 10 in the architecture that has located *3* membranes in *4* processors (Figure 1.b).

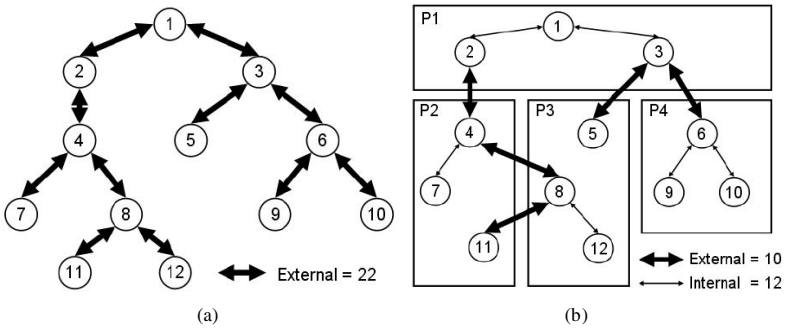

Fig. 1. (a) P system communications. **(b)** Communications with membranes distribution.

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. Therefore, the communications that use the common line (external communications to the processor) are carried out between proxies, not between membranes. This intermediate element located between the bus and the membranes concentrates the information in two stages:

a) *N multisets* of *N* membranes located in a processor that has a common father membrane in another processor, becoming integrated in a single *multiset* that is the one that will be sent.
b) The *S* communication packet of *L* length necessary to communicate between S pairs of membranes located in 2 different processors are reduced to one single packet of *S.L* length.

The benefit of using proxies in the communication among membranes against direct communication is double:

a) Due to the first stage previously described, the amount of information sent is smaller. This is produced by the fact that the *N* packet necessary to communicate *N* membranes with the same father, are transformed into a single packet of the length of a single *multiset*.

b) Due to the second stage, the number of external communications is smaller although packets are bigger. But, considering that the communication protocols penalize the transmission of small packets because to the data encapsulated processes and to the time safety intervals between future transmissions, it is better to send one packet of length equal *S.L* than *S* packets of length equal *L*.

Figure 2.a shows that if proxies are introduced in the processors, then the number of external communications is reduced to 8.

Tree topology of processors: In graph theory it is established that $P-1$ connections is the minimum number required to interconnect a connected graph of P processors. This restriction imposes on the graph a tree topology. 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 in each evolution step is $2(P-1)$.

Figure 2.b shows that external communications are reduced to 6 when a tree topology of processors is used to connect them.

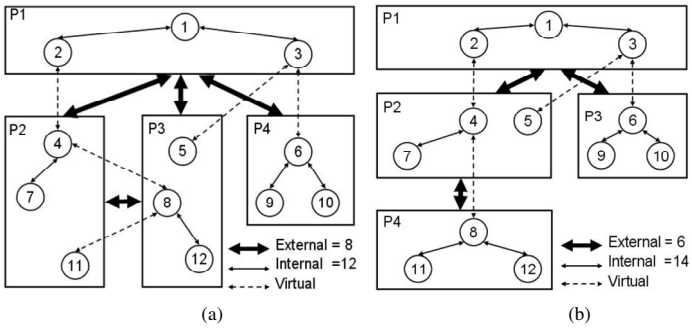

Fig. 2. (a) Communications with a proxy for processor. **(b)** Communications using a tree topology.

3 Fritzke's Self-Organizing Neural Networks (SONN)

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 [5] proposed first model of SOM, where the complete network structure had to be specified in advance and remained static during all the training process. When Kohonen's SOM is used, the choice of inappropriate parameters to define the architecture can degrade the posterior performance of the network. In 1994, B. Fritzke [6] proposed a SOM model called Growing Cell Structure (GCS) where this static structure limitation was eliminated. In addition, the flexibility that offers the possibility of inserting and removing neurons on the output layer of the network during the training phase causes that the GCS network receives

better value associated to the topology preserving term, understanding this like the grade that defines the quality of the simplified model that the network represents. There exist other models of SOM networks that usually offer better topology preserving grades than GCS, as GNG[7] and ESOM[8], where there no exist so strict structural organization of the output layer. Nevertheless, this feature causes that they cannot be used directly to generate two-dimensional graphs to show the relations of the input patterns, being necessary using dimension reduction algorithms (like Sammon's projection) to visualize the prototype nodes and their relations. With the purpose of exploiting this characteristic when the input patterns present more than two dimensions we decided to use GCS model.

GCS is a two-layer architecture network (Fig 3). 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, so output units are connected forming triangle groups.

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:

$$S_{bmu} = \arg\min \left\| e - w_c \right\| . \tag{1}$$

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 1 and 2 respectively (where $\varepsilon_b > \varepsilon_n$).

$$\Delta w_{bmu} = \varepsilon_b \left(e - w_{bmu} \right) . \tag{2}$$

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

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. The place where new unit is inserted is determined using two different criteria: "*looking for the unknown probability distribution of the input patterns*" (LUPD) or "*looking for equalize accumulated error measure*" (LEAE) [6]. Periodically superfluous neurons are removed in order to obtain better results when input

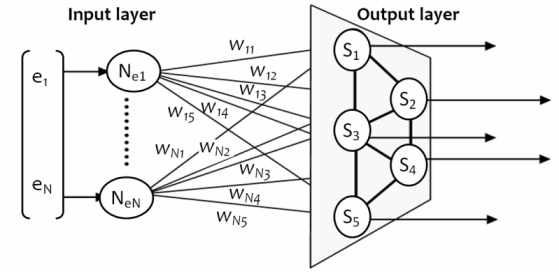

Fig. 3. GCS network topology with $k=2$, N input neurons and 5 output units that exhibit neighbors connections in groups of figures of triangles

space consists of several separate regions of positive probability density. An output neuron can be considered superfluous if it has a synaptic vector in a region with very low probability density (a region without any training pattern). 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 [9] 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 data inherent knowledge can be discovered.

4 Experiments and Results

To test the system we have selected thirty P-System models generally used in the literature of P-System. Different membrane distributions between different number of processors have been generated for every P-System, observing how the resulting communications of the distribution affect to the parallelization grade. For each data set associated to a concrete P-System diverse GCS networks have trained 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.

For space reasons in this section only the results of one of the multiple GCS trained networks is showed, in particular the one trained with the P-System of the fig. 1.a, with distributions that uses four processors. First of all, 15 feasible combinations of 12 membranes have been generated in 4 processors that have resulted in 180 bi-dimensional labeled vectors. Each vector maintains the volume of internal and external communications for a concrete membrane-processor-distribution and it has associate a label that identifies these three elements. With this patterns a GCS network has been trained, with LEAE insertion criterion, $\varepsilon_b = 0.06$, $\varepsilon_n = 0.002$, $\mu = 0.006$, and concluding when at least 6 isolated clusters of output units are obtained. After the training phase the output units of the network has been marked with the union of the labels of all those input patterns that fall inside its Voronoi region. Figure 4 shows the scattergram of this GCS network, where the position of each output unit is determined by the two components of its synaptic vector. X-axis coordinates indicates the internal

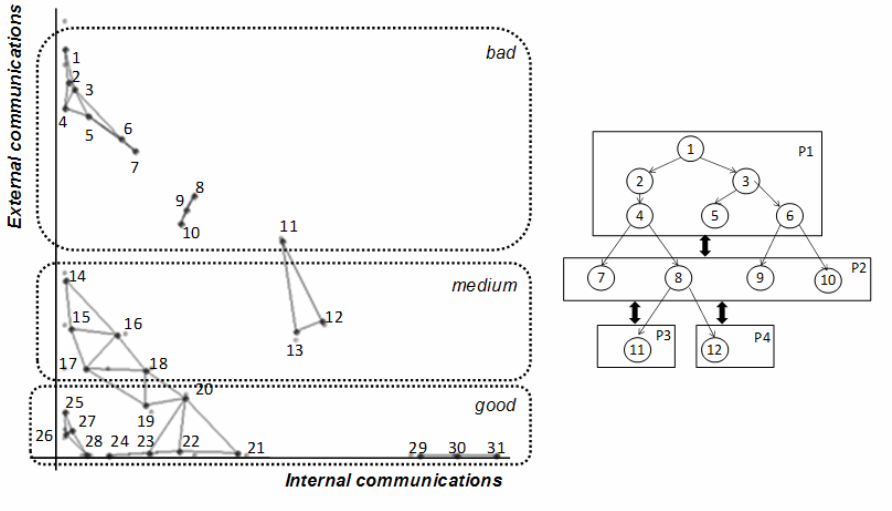

Fig. 4. Left: Scattergram of a GCS network. Points represent neurons and lines between them neighbor connections. Units are grouped in *bad*, *medium* and *good* classes on the basis of the external communication degree. Right: Distribution of membranes C_{13}.

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 (with a volume of 183 for internal communications and 83 for external communications).

5 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 [10][11].

Given the good results obtained in the experiments, as future extensions the possibility of working with vectors of greater dimension is considered, what will allow to fit the search of suitable balanced P-System, for example generating a single vector for each membrane distribution or working with fuzzy values for determining the

degree of internal and external communications of a processor. This will require working with new visualizations of the output layer of the GCS network, such as those exposed in [12].

References

1. Păun, G.: Computing with membranes. Journal of Computer and System Sciences, 61 (2000), and Turku Center for Computer Science-TUCS Report No 208 (1998)
2. Tejedor, A., Fernandez, L., Arroyo, F., Bravo, G.: An architecture for attacking the bottleneck communication in P Systems. In: Sugisaka, M., Tanaka, H. (eds.) Proceedings of the 12th Int. Symposium on Artificial Life and Robotics, Beppu, Oita, Japan, pp. 500–505 (2007)
3. Bravo, G., Fernández, L., Arroyo, F., et al.: A hierarchical architecture with parallel communication for implementing P-Systems. In: ITA 2007 Xth Joint International Scientific Events on Informatics, Varna, Bulgaria (2007)
4. Bravo, G., Fernández, L., Arroyo, F., et al.: Master-Slave Distributed Architecture for Membrane Systems Implementation. In: 8th WSEAS Int. Conf. on Evolutionary Computing (EC 2007), Vancouver, Canada (2007)
5. Kohonen, T.: Self-Organized Formation of Topologically Correct Feature Maps. Biological Cybernetics 4359–4369 (1982)
6. Fritzke, B.: Growing Cell Structures – A Self-organizing network for Unsupervised and Supervised learning. Neural Networks 7(1), 1441–1460 (1994)
7. Fritzke, B.: A growing neural gas network learns topologies. Advances in neural information processing systems 7, 625–632 (1995)
8. Deng, D., Kasabov, N.: On-line pattern analysis by evolving self-organising maps. Neurocomputing 51, 87–103 (2003)
9. Delgado, S., Gonzalo, C., Martínez, E., Arquero, A.: Improvement of Self-Organizing Maps with Growing Capability for Goodness Evaluation of Multispectral Training Patterns. In: IEEE International Geoscience and Remote Sensing Symposium, vol. 1, pp. 564–567 (2004)
10. Gutiérrez, A., Fernández, L., Arroyo, F., Alonso, S.: Hardware and Software Architecture for Implementing Membrane Systems: A Case of Study to Transition P Systems. In: Garzon, M.H., Yan, H. (eds.) DNA 2007. LNCS, vol. 4848, pp. 211–220. Springer, Heidelberg (2008)
11. Gutierrez, A., Fernández, L., Arroyo, F., Alonso, S.: Suitability of Using Microcontrollers in Implementing new P System Communications Architectures. In: AROB 2008, XIII International Symposium on Artificial Life and Robotics, Oita, JAPAN, January 31-February 2 (2008)
12. Delgado, S., Gonzalo, C., Martinez, E., Arquero, A.: Visualizing High-Dimensional Input Data with Growing Self-Organizing Maps. In: Sandoval, F., Prieto, A.G., Cabestany, J., Graña, M. (eds.) IWANN 2007. LNCS, vol. 4507, pp. 580–587. Springer, Heidelberg (2007)