Distributed models in P-Systems architectures to reduce computation time

Miguel Ángel Peña, Ginés Bravo, Luis Fernando de Mingo

Abstract—**Membrane systems are computational equivalent to Turing machines. However, their distributed and massively parallel nature obtains polynomial solutions opposite to traditional non-polynomial ones. At this point, it is very important to develop dedicated hardware and software implementations exploiting those two membrane systems features. Dealing with distributed implementations of P systems, the bottleneck communication problem has arisen. When the number of membranes grows up, the network get congested. The purpose of distributed architectures is to reach a compromise between the massively parallel character of the system and the needed evolution step time to transit from one configuration of the system to the next one, solving the bottleneck communication problem. The goal of this paper is twofold. Firstly, to survey in a systematic and uniform way the main results regarding the way membranes can be placed on processors in order to get a software/hardware simulation of P-Systems in a distributed environment. Secondly, we improve some results about the membrane dissolution problem, prove that it is connected, and discuss the possibility of simulating this property in the distributed model. All this yields an improvement in the system parallelism implementation since it gets an increment of the parallelism of the external communication among processors. Proposed ideas improve previous architectures to tackle the communication bottleneck problem, such as reduction of the total time of an evolution step, increase of the number of membranes that could run on a processor and reduction of the number of processors.**

Index Terms—**Distributed Communication, Membrane Computing, Membrame Dissolution, P-Systems Architectures, Computational Models**

I. Preliminaries

NATURAL sciences, and especially biology, represented a rich source of modelling paradigms. Well-defined areas of artificial intelligence (genetic algorithms, neural networks), mathematics, and theoretical computer science (L systems, DNA computing [1]) are massively influenced by the behaviour of various biological entities and phenomena. In the last decades or so, new emerging fields of so-called natural computing identify new (unconventional) computational paradigms in different forms. There are attempts to define and investigate new mathematical or theoretical models inspired by nature, as well as investigations into defining programming paradigms that implement computational approaches suggested by biochemical phenomena.

Membrane computing, inspired in *"basic features of biological membranes"*, was introduced by Gheorge Paŭn [2] to solve *NP*-Complete problems in polynomial time. As original model –Transition P System– as remaining models emerging from its; they are an abstract representation of hierarchical structure and non-deterministic behavior of biological membranes. A membrane is a region compounds by other membranes and chemicals (objects) that uses chemical reactions (evolution rules) generated another chemicals. Each membrane has a permeability capacity that enable chemicals (objects) to move between membranes (communication). Chemicals reactions produced in membranes can dissolve it. This process implies that contained object and membranes to become part of parent membrane.

Universality results have been obtained [3], [4], [5] for a number of variants of population P-systems. The following different rules are considered: transformation rules for modifying the objects that are present inside the cells, communication rules for moving objects from a cell to another one, cell division rules for introducing new cells in the system, cell differentiation rules for changing the types of the cells, and cell death rules for removing cells from the system. As well as this, bond-making rules are considered that are used to modify the links between the existing cells (i.e., the set of edges in the graph) at the end of each step of evolution performed by means of the aforementioned rules. In other words, a population P-system in [6], [7] is basically defined as an evolution- communication P-system but with the important difference that the structure of the system is not rigid and it is represented as an arbitrary graph. In particular, bond making rules are able to influence cell capability of moving objects from a place to another one by varying the set of edges in the underlying graph.

In base to this behaviour, P-System are systems that can be executed *on-line* that is, in vitro or simulated, using hardware implementations (Petreska [8], Fernandez [9] or Martinez [10]), using software simulations(Suzuki [11] o Arroyo [12]) or even in a real cluster of processors (Ciobanu [13] o Syropoulos [14]). Currently, researchers focused on simulations by distributed software, to alleviate the sequential nature of processors, to obtain lower running time.

II. P System definition

The first definition of a P System was published by Păun [2], who defined a Transition P System as:

M.A. Peña is with Dpto. Inteligencia Artificial, Facultad de Informática, Universitad Politécnica de Madrid, Campus de Montegancedo, 28660 Madrid, Spain, m.pena@upm.es

G. Bravo is with Dpto. Organizacón y Estructura de la Información, Escuela Universitaria de Informática, Universitad Politécnica de Madrid, Crta. De Valencia km. 7, 28031 Madrid, Spain, gines@eui.upm.es

L.F. de Mingo is with Dpto. Organizacón y Estructura de la Información, Escuela Universitaria de Informática, Universitad Politécnica de Madrid, Crta. De Valencia km. 7, 28031 Madrid, Spain, lfmingo@eui.upm.es

Definition 1: A Transition P System is $\Pi = (V, \mu, \omega_1, \ldots, \omega_n, (R_1, \rho_1), \ldots, (R_n, \rho_n), i_0)$, where:

- V is an alphabet (composed of objects),
- μ is the membrane structure with n membranes
- ω_i are the multiset of symbols for the membrane i.
- R_i are the evolution rules for the membrane i. A rule is a sorted pair (u, v) where u is a string over V, and $v = v'$ or $v = v'\delta$ and v' is a string over $V_{TAR} = V\ x\ TAR$ with $TAR = here,\ out\ U\ \{in_j | 1 \leq j \leq n\}$. δ is a special symbol not included in V, and represents the dissolution of the membrane
- P_i are the priority of rules for the membrane i.
- i_0 indicates a membrane, which is the system output membrane or skin membrane.

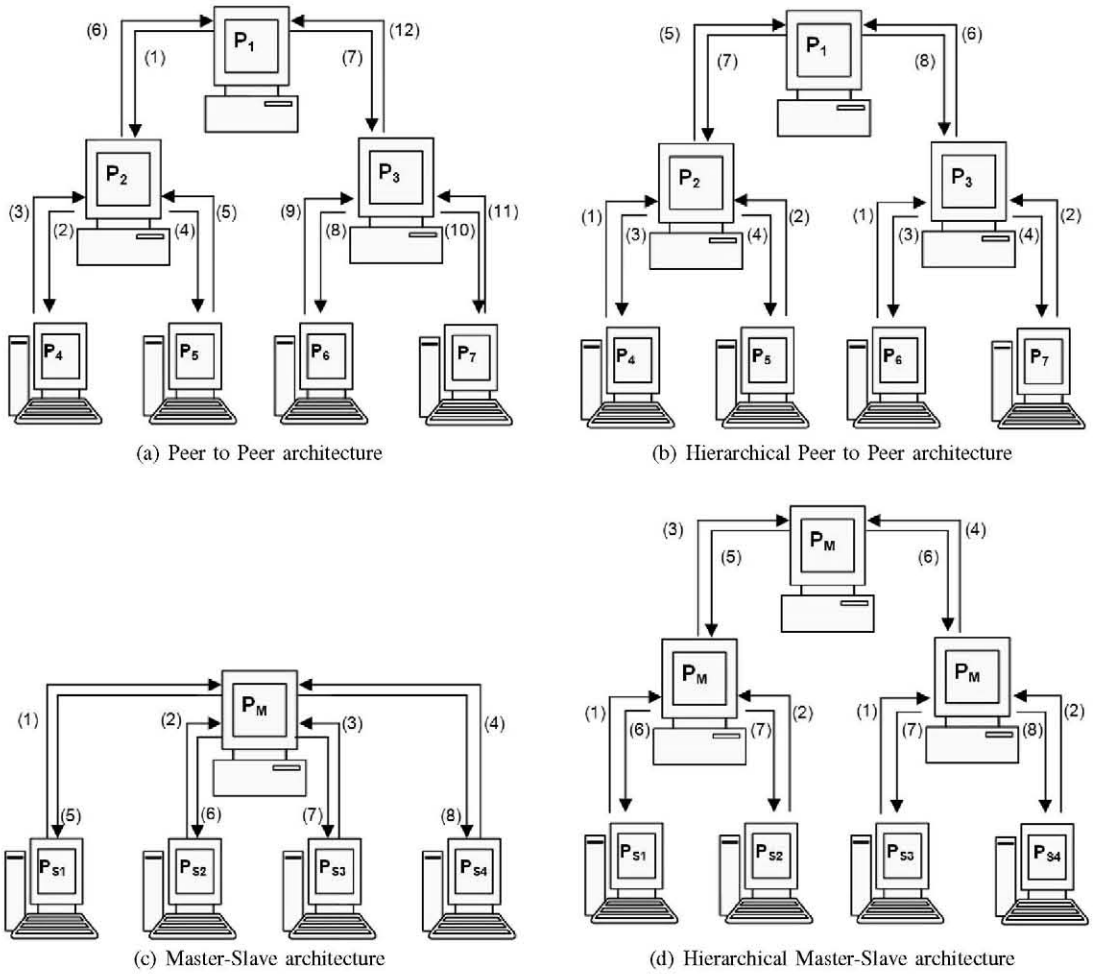
Dynamics of P-System is made through configurations. The following phases are performed on each configuration of a membrane in parallel and no deterministic way:

1) To determine the set of useful rules: On this micro-step all evolution rules of the membrane are evaluated in order to determine which are useful. A rule is useful when all membranes in its consequent exists in the P-System.
2) To identify the set of applicable rules: It will be necessary to evaluate all the evolution rules of the membrane to identify those that meet the following constraint: its predecessor is contained in the multiset of objects of the region.
3) To build the set of actives rules: Intersection of two previous sets are the input group to this micro-step. Each one of the rules belonging to the set must be processed, to determine which meets the condition of active rule [15], [16]. To determine if a rule meets such condition, it is necessary check if there is not another rule with higher priority that belongs to the useful and applicable rules set.
4) Non deterministic distribution of objects of the region between its active rules and application: In this micro-step, copies of present objects in the multiset of the region are distributed between active evolution rules. Copies of objects that are assigned to each rule, match with those of the multiset that results from scalar product of a number between minimum and maximum bound of applicability of those rule and its predecessor. This distribution process is made on a non-deterministic way. Moreover, at the end of it, objects no assigned to any rule forms a multiset and they will be characterized because they do not contained to any predecessor of rules. The result of the distribution is a multiset of actives rules, where multiplicity of each rule defines the times that would be applied, and therefore, indirectly through its predecessors, objects are assigned to the multiset of the region. Objects used are eliminated and generate new objects that are destined for a membrane.
5) Output of multiset generated objects or communication phase: In this micro-step, the new objects generated on the previous micro-step whose target membranes was in or out, must be transferred to its corresponding membranes. Each membrane, will have unused objects in its application, together with those that result for the applying of rules and that have this membrane as destiny.
6) Membranes dissolution: This action means that membranes who have applied some rule with dissolution capacity, they must send containing objects at this time to their nearest antecessor.
7) Composition of new objects multiset: At finish this last micro-step, on each membrane it should generate a new multiset of objects that will be used in the next step of evolution. In this way, multiset of objects of delimitated region by a membrane identified as j will be formed by:
   - Objects do not assigned to any rule at the non-deterministic distribution.
   - Objects created in the membrane whose target identifier, is the membrane itself.
   - Objects created in daughter membranes whose target identifier was out.
   - Objects coming from mother membrane, whose target identifier was in_j.
   - Objects coming from dissolution of daughter membranes of membrane j and all objects whose their destiny was dissolved j daughter membrane.

About those steps, many researchers have developed algorithms to reduce time it takes to evolve P-System. In some cases, many steps were grouped to reach a reduction of time. In particular, Frutos [17] proposed to create decision trees to determine possible rules to apply according to membrane context (micro-steps 1-3). Tejedor [18] proposed an algorithm to distribute objects among rules and its application (micro-step 4). Objects communication between membranes depends on used distributed architecture. Since Ciobanu [13] detected that network congestion produce higher response time, future studies have focused on searching architectures to eliminate network collision. Then, we can group micro-steps on base to membranes permeability: steps executed inside membrane (1 to 4 steps) called applying rules, and inter-membrane steps (5 to 7 steps) called object communication between membranes, and depend on the used architecture.

III. P-Systems Architectures

The implementation of P system in digital hardware device is being carried out from the point of view of Hardware as well as Software. Most of the solutions have been focused, mainly, in the first phase of the P system evolution describing digital circuits or software architectures/designs that have allowed the application of the defined evolution rules inside the membranes. The phase of multisets membranes communication has not been contemplated or it has simply been performed by shared memory, except Syropoulos [14] and Ciobanu [13] that in their distributed implementations of P systems use Java Remote Method Invocation (RMI) and the Message Passing Interface (MPI) respectively, on a cluster of PC connected by Ethernet. These last authors do not carry out a detailed analysis of the importance of the time used during communication phase in the total time of P system evolution, although Ciobanu states that "the response time of the program



(a) Peer to Peer architecture

(b) Hierarchical Peer to Peer architecture

(c) Master-Slave architecture

(d) Hierarchical Master-Slave architecture

Fig. 1. Different distributed P-System architectures including the object communication order.

has been acceptable. There are however executions that could take a rather long time due to unexpected network congestion [13].

The first distributed architecture to implement P-Systems that eliminate collisions is proposed like *"partially parallel evolution with partially parallel communication"* [19]; namely Peer-to-Peer ($P2P$) as opposed to other architectures. In this architecture (Figure 1(a)), processors are connected with a tree topology following the inner scheme in P-System structure: k membranes are placed in every processor P and sequentially executed applying rules. On communication step, following a post order way of processor tree (Figure 1(a), proxies situated on each processor send generated objects in its membranes, using only one message, to proxies of processors that are target membranes. Total time used in each evolution of P-System is:

$$T = KT_{apl} + 2(P-1)T_{com} \tag{1}$$

Based on the same idea and same tree topology, Bravo in [20] adds parallelism to transmission of objects, so simultaneously multiple processors can be communicating with others processors with no collision at all (Figure 1(b)). This architecture cited like Hierarchical Peer-to- Peer ($HP2P$) reduces, in a notorious way, time used in evolution steps:

$$T = \frac{M(A-1)T_{apl}}{A^L - 1} + 2T_{com}A(L-1) \tag{2}$$

Where A is amplitude or processor number of children and L is number of processors tree levels, taking root like L=1.

Third architecture reduces time and eliminates complicated restrictions of tree topology of previous models, it is Master-Slave architecture (MS) [21]. With this architecture, also based on unique simultaneously communication between proxies (Figure 1(c)), can make the communication step while other processors are in their communication step. Time used by every evolution phase is:

$$T = kT_{apl} + T_{com}(P_s + 1) \tag{3}$$

Communications on Master-Slave architecture are sequentials and it produces that time of this step is linear. To address these issues arises Hierarchical Master-Slave architecture (HMS) [22] groups advantages of $HP2P$ architecture with MS architecture (Figure 1(d)), obtaining a time of:

$$T = \frac{MT_{apl}}{A^{L-1}} + T_{com}(2L + A + A^{L-2} - 4) \tag{4}$$

IV. Behavior of distributed architectures

All this architectures are designed to reduce P-System running time, considering two execution steps of it. In addition to this stages, in different types of P-System may exists membrane dissolution or membrane division. There are to make changes to include these new steps in these architectures. $P2P$ and MS architectures may work with only the addition of a improvement proxy that permit dissolution. $HP2P$ and HMS architectures have to modify message flow to permit, through intermediate processors, communications among nodes even though it is not connected directly. This new communication sequence are shown in figures 3(a) (to H$P2P$) and 3(b) (to HMS).

$HP2P$ architecture starts its communication on root processor, which communicates with its children sequentially. Each one of these can starts communication with its children, when received father communication (root). In the same way, children communicate with it corresponding children. Reverse process, from leaves to root, is similar considering the restrictions that one processor only can have one communication to avoid collisions. When root received the last communication from it child, starts next evolution on application phase.

HMS architecture starts with communication with all slaves processors. Communications of all slaves are made in sequential, because there are many slaves processor connected to same master. But, two slaves of different master can starts to communicate in parallel, because there are not collisions. With the same idea, master can communicate with its masters-father, and so, to reach master-root. When it have received from whole its masters-children, made its process and sequentially communicate to whole masters-children. Each one of them, after receive the communication, it can communicate with its children (similarly to make to $HP2P$ architecture). When a master has terminated to communicate with its slaves, these can execute the application step of its next evolution phase, and it starts communication step when finished application step, avoiding collisions.

Except for the first evolution of P System, HMS and $HP2P$ architectures require the same time to communicate (as shown in figures 3(a) and 3(b)). Knowing that communication time between two processors (only one way) is T_{com}, total communication time for each evolution of the P System is:

$$T = T_{com}(AL + L - 2) \tag{5}$$

In both architectures, communications number behaves same, but its application time is different, because in $HP2P$ architecture exists membranes in every processor; and in HMS only exists in processors slaves. Existing k membranes on each processor, to take T_{apl} to apply, application time would be:

$$T = kT_{apl} \tag{6}$$

On $HP2P$ architecture there will be P processors:

$$P = \frac{A^L - 1}{A - 1} \tag{7}$$

On HMS architecture there will be the same number of total processors, of whom, processors slaves and masters, will be:

$$P_s = A^{L-1} \tag{8}$$

$$P_m = \frac{A^{L-1} - 1}{A - 1} \tag{9}$$

Knowing total number of membranes M is results of multiply membranes of each processor by the processors number that contain membranes, total times of each execution step to $HP2P$ and HMS respectively are:

$$T_{HP2P} = \frac{T_{apl}M(A-1)}{A^L - 1} + (LA + L - 2)T_{com} \tag{10}$$

$$T_{HMS} = \frac{MT_{apl}}{A^{L-1}} + (LA + L - 2)T_{com} \tag{11}$$

Knowing the time lasting each evolution step, it would be determined what is the optimum value of A and L to obtain minimum time. Also knowing that A and L must be integers. Figures 4(a) and 4(b) shown the better combination of A and L to be the minimum time, according relationship between T_{apl} and T_{com}, and number of membranes.

V. Behavior of distributed architectures with overlapping stages

On initial proposal of Paun [2] about membrane performance, each showed micro-step are sequentially executed, although it exists parallelism inside each micro-step. Those micro-steps are executed in-parallel on all processors. However, in software and hardware implementations, recent studies have demonstrated some micro-steps can be grouped and executing in-parallel with new algorithms- so the total time is reduced. Figure 3(b), shows how while some processors are communicating, another are doing their application phase. Specifically, this figure shows how a processor when determine that has finished to communicate, it starts to applied rules. But even if finished the communication on subnet where it is connected, the communication phase is not finished, because it exists communication in other subnet. For example, the processors 3 to 6 ($P3 - P6$) subnet finished their communication on time 15 ($t = 15$). In this moment, processors 4, 5 and 6 can start their rules application phase, because their thought that communication phase is finished. But, in other subnets, there are processors still not finish their communication, like processor 7.

This idea, reducing communication time, overlap communication step and application step. If a processor does not wait to finish the communication on belonging subnet, and start its rules applying step on the moment it finished execution step, it could reduced more the evolution total time. In the same example, processor 4 could start its selection phase and rules application phase on $t = 13$. Figure 5 shows how the execution schedule is implemented with this overlapping steps

Simultaneously, between $t = 13$ and $t = 19$, it is producing communication over some processors and rules application over other ones. When a processor finished its communication phase and net is available for a new communication phase passed:

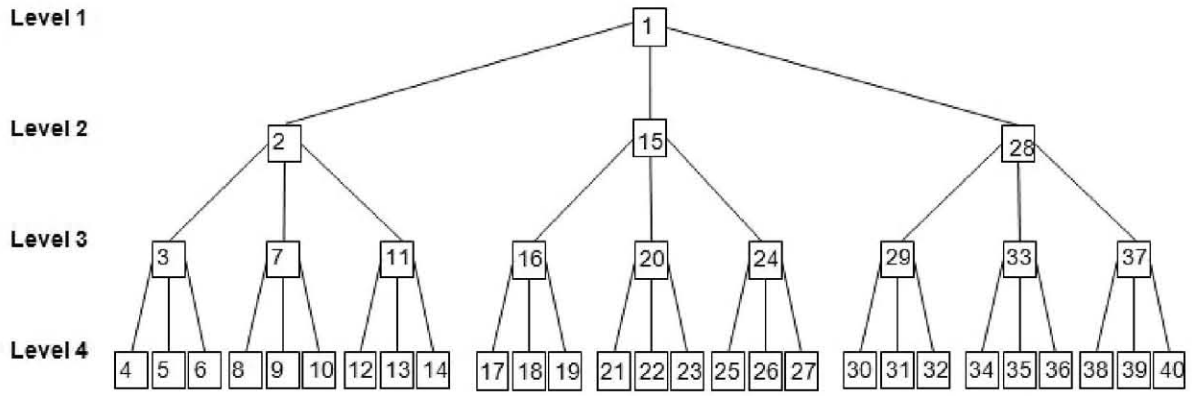
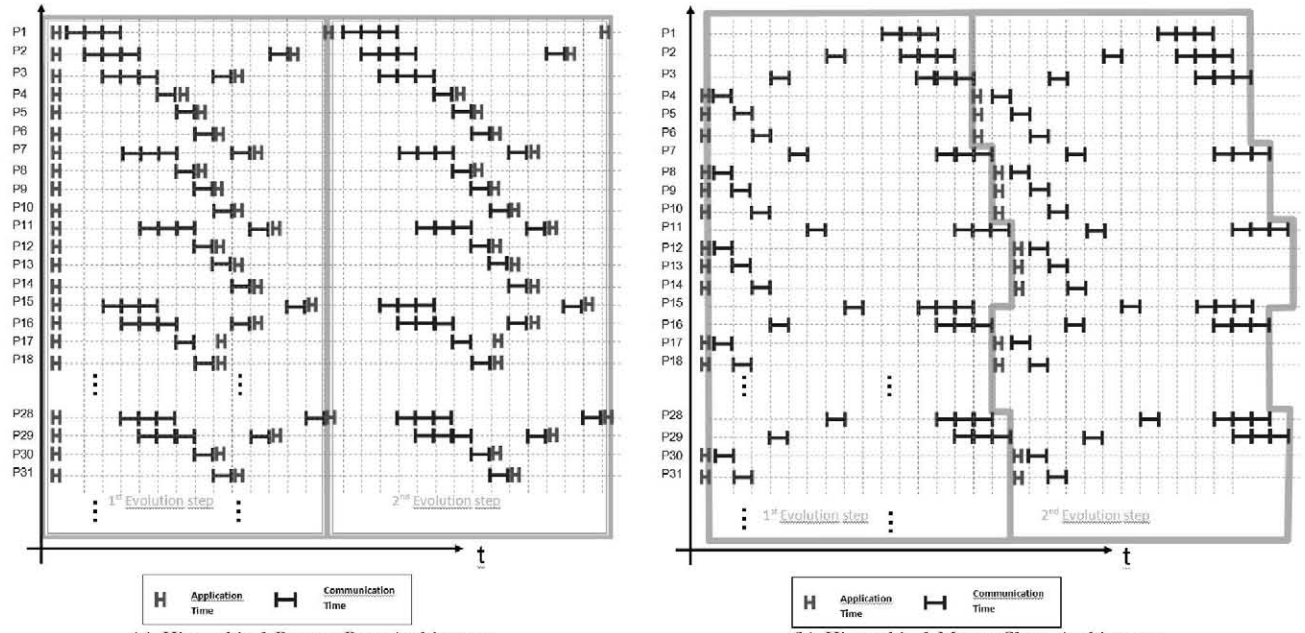


Fig. 2. Distribution of processors in a tree with 4 depth levels and amplitude equal to 3.



(a) Hierarchical Peer to Peer Architecture

(b) Hierarchical Master-Slave Architecture

Fig. 3. Chronogram of different P-System architectures with 4 depth levels and amplitude equal to 3 (Figure 2).

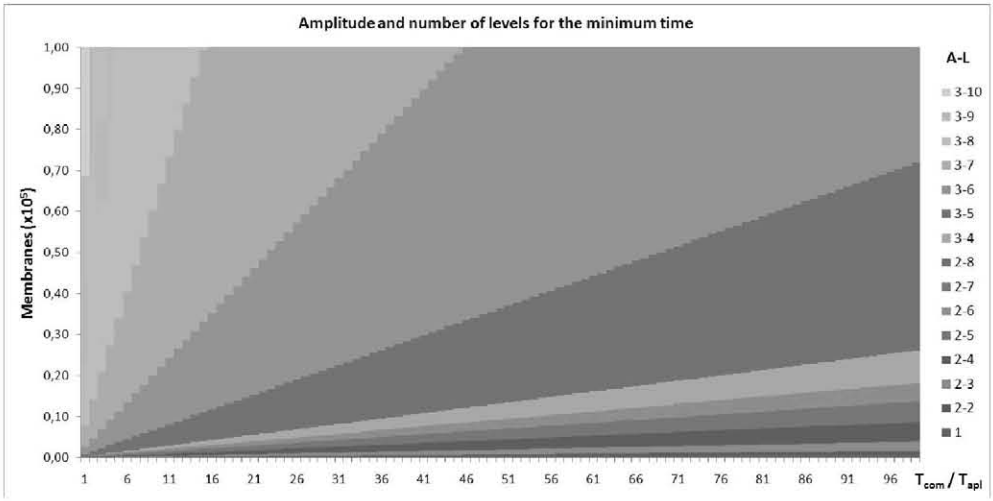
$$T = (A-1)T_{com} \tag{12}$$

It is to say, the time lasting to communicate the other slave-proccesors of subnet. On the example, the processor 4 can communicate again on $t = 15$ if finished its rules application step. This causes two types of possible distributions on Hierarchical Master-Slave: those have a totally overlapping communication and rules application and those have a partial overlapping. In the first group, the network utilization is maximized, it is not idle waiting that a processor finish its rules application step. Also, using fixed time intervals to communication, each processor can start the communication on its assigned time without considering the net state and with the certainty there is not collisions. If instead the rules application takes much longer and the overlapping is partially, it is necessary that the processor monitoring the net state to dont start its communication before the assigned time. Since there are two overlapping steps types, the time spent on each evolution step is determined by:

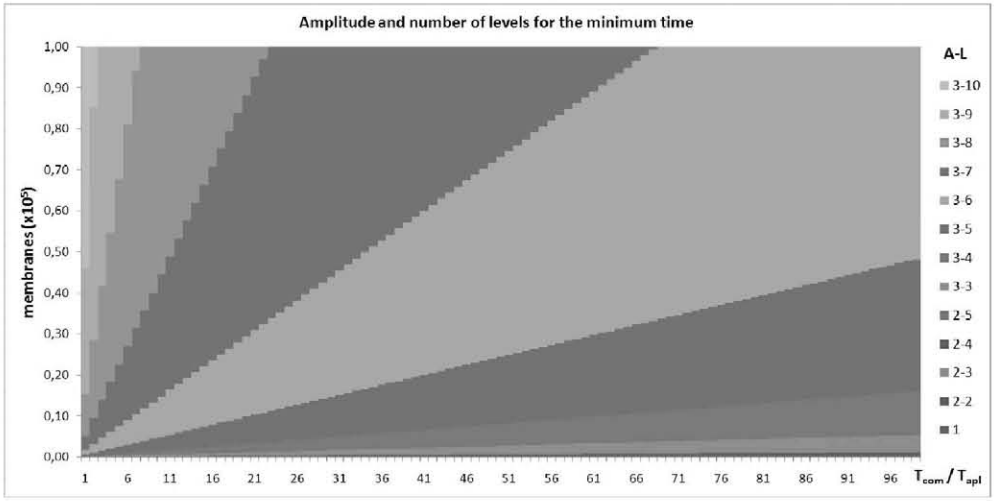
$$T_{HMS} = \begin{cases} (AL+L-2)T_{com} \text{ if } \frac{MT_{apl}}{A^{L-1}} \leq (A-1)T_{com} \\ \frac{MT_{apl}}{A^{L-1}} + (LA+L-A-1)T_{com} \text{ otherwise} \end{cases} \tag{13}$$

About these two options, and for every case, it must be analyzed what of these reduce the total time of evolution. Figure 6 shows the better combination of processors to minimize the evolution time, in function of number of membranes and T_{apl} T_{com} ratio. In particular, it shows values of amplitude and depth. Comparing with evolution without overlapping, significantly increases the amplitude because at the time other processor communicate, makes rules application.

This advance made on Hierarchical Master-Slave cannot be doing on Hierarchical Peer-to-Peer architecture, due to root processor also have membranes and it is the first and the last



(a) $HP2P$ Architecture



(b) HMS Architecture

Fig. 4. Amplitude and depth for the minimum time in P-Systems Architectures.

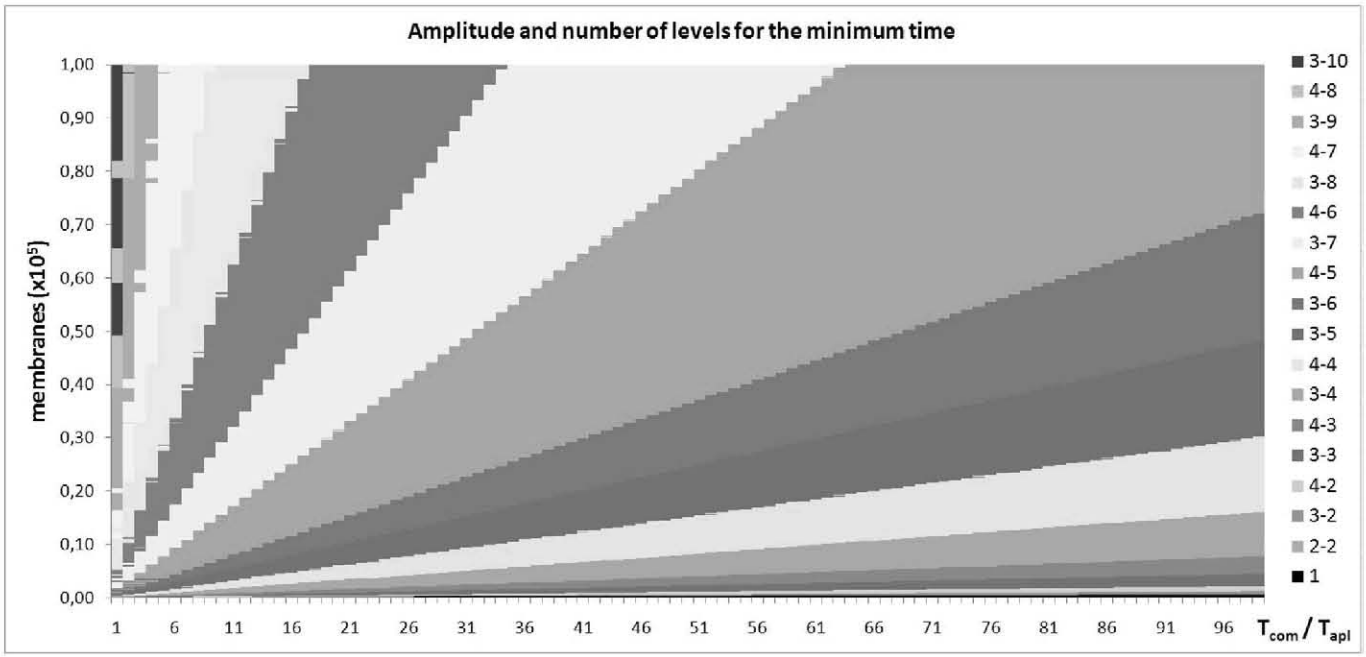


Fig. 6. Amplitude and depth for the minimum time in Hierarchical Master-Slave Architecture with overlapping stages

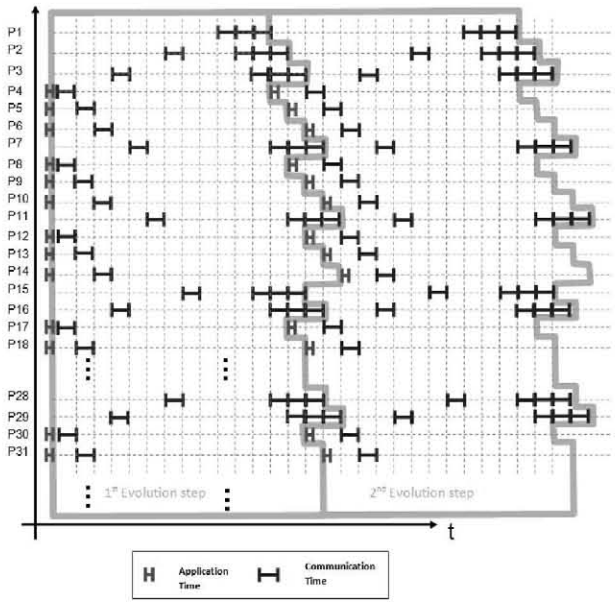


Fig. 5. Chronogram of Hierarchical Master-Slave Architecture with overlapping stages with 4 depth levels and amplitude equal to 3 (Figure 2).

that communicate. Although it cannot exist overlapping on this processor, but in other ones can. If only root membrane on this processor, it can be other membranes distribution, that reach a overlapping on the others processors and reducing total time.

VI. Results and Future Work

Concerning membrane dissolution and membrane division steps, they modify the existing architectures behavior. The communications of this architecture must take into account such changes. Hierarchical Peer to Peer Architecture and Hierarchical Master-Slave must consider this behaviour. However, the total time and the spatial distribution change because in $HP2P$ there are membranes on all processors and in HMS only on slaves. Moreover, this change reduces the execution time.

To achieve a minimum time, it should consider other distributions different than when they took into account in the dissolution of membranes. With this new configuration the number of children of a processor increases to 3 with less than 1000 membranes. The optimal configuration of the number of children and the depth tree of processors, depending on the number of membranes and the ratio of the time it takes to apply membranes and that take in communicating processors.

With the overlapping steps, while ones processor making communication steps, other ones can make rules applying step. With this overlapping, the needed time to rules application is eliminated, taking the P-System evolution only the time used to communicate the processors where it is hosted.

A. Networks of Evolutionary Processors

Previous ideas can be taken into account in new distributed symbolic models such as Networks of Evolutionary Processors.

A network of evolutionary processors [23], [24] of size n is $\Gamma = (V, N_1, N_2, \cdots, N_n, G)$, where V is an alphabet and for each $1 \leq i \leq n$, $N_i = (M_i, A_i, PI_i, PO_i)$ is the i-th evolutionary node processor of the network. The parameters of every processor are:

- M_i is a finite set of evolution rules of one of the following forms only
  - $a \to b, a, b \in V$ (substitution rules)
  - $a \to \epsilon, a \in V$ (deletion rules)
  - $\epsilon \to a, a \in V$ (insertion rules)

  More clearly, the set of evolution rules of any processor contains either substitution or deletion or insertion rules.
- A_i is a finite set of strings over V. The set A_i is the set of initial strings in the i-th node. Actually, in what follows, we consider that each string appearing in any node at any step has an arbitrarily large number of copies in that node, so that we shall identify multisets by their supports.
- PI_i and PO_i are subsets of V^* representing the input and the output filter, respectively. These filters are defined by the membership condition, namely a string $w \in V^*$ can pass the input filter (the output filter) if $w \in PI_i (w \in PO_i)$.

the network. The edges of G, that is the elements of E, are given in the form of sets of two nodes. The complete graph with n vertices is denoted by K_n. By a configuration (state) of an *NEP* as above we mean an n-tuple $C = (L_1, L_2, \cdots ., L_n)$, with $L_i \subseteq V^*$ for all $1 \leq i \leq n$. A configuration represents the sets of strings (remember that each string appears in an arbitrarily large number of copies) which are present in any node at a given moment; clearly the initial configuration of the network is $C_0 = (A_1, A_2, \cdots, A_n)$.

A configuration can change either by an evolutionary step or by a communicating step [23]. When changing by an evolutionary step, each component L_i of the configuration is changed in accordance with the evolutionary rules associated with the node i. When changing by a communication step, each node processor N_i sends all copies of the strings it has which are able to pass its output filter to all the node processors connected to N_i and receives all copies of the strings sent by any node processor connected with N_i providing that they can pass its input filter.

More important results are:

- Each recursively enumerable language can be generated by a complete NEP of size 5. [23]
- Each recursively enumerable language can be generated by a star NEP of size 5. [23]
- The bounded PCP can be solved by an NEP in size and time linearly bounded by the product of K and the length of the longest string of the two Post lists. [24]

In fact, NEPs are P-System with a graph topology, but they have a different behaviour. In this case, distribution of processors instead of membranes must be accomplished in order to reduce computation time when simulating in a cluster environment. Proposed architectures could be use to optimise such processors distribution.

ACKNOWLEDGMENT

This work has been partially supported by the Spanish Research Projects: TRA2010-15645. COMUNICACIONES EN MALLA PARA VEHICULOS E INFRAESTRUCTURAS INTELIGENTES (Mesh communication with intelligent vehicles). (2010) and TEC2010-21303-C04-02. ESTRUCTURAS RESONANTES PARA APLICACIONES DE SEAL FOTONICA DE BANDA ANCHA. (2010).

REFERENCES

[1] R. Freund, "An integrating view on dna computing and membrane computing," in *Proceedings of the 9th WSEAS International Conference on EVOLUTIONARY COMPUTING(EC08)*. World Scientific and Engineering Acad and Soc, 2008, pp. 15–20.

[2] G. Paun, "Computing with membranes," *Journal of Computer and System Sciences*, vol. 61, no. 1, pp. 108–143, Aug. 2000.

[3] M. Madhu and K. Krithivasan, "Improved results about universality of P systems," *Bulletin of the EATCS*, no. 76, pp. 162–168, February 2002.

[4] A. Rodriguez-Patón, "On the universality of P systems with membrane creation," *Bulletin of the EATCS*, no. 74, pp. 229–234, June 2001.

[5] R. Freund, L. Kari, M. Oswald, and P. Sosík, "Computationally universal P systems without priorities: two catalysts are sufficient," *Theoretical Computer Science*, 2004, in press.

[6] F. Bernardini, M. Gheorghe, and N. Krasnogor, "Population P systems and quorum sensing in bacteria," *Theoretical Computer Science*, 2006, ?

[7] F. Bernardini and M. Gheorghe, "Population P systems," *Journal of Universal Computer Science*, vol. 10, no. 5, pp. 509–539, May 2004.

[8] B. Petreska and C. Teuscher, "A reconfigurable hardware membrane system," *Membrane Computing*, vol. 2933, pp. 269–285, 2004.

[9] L. Fernandez, V. J. Martinez, F. Arroyo, and L. F. Mingo, "A hardware circuit for selecting active rules in transition p systems," *Seventh International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, Proceedings*, vol. 1, pp. 415–418, 2005.

[10] V. Martinez, F. Arroyo, A. Gutierrez, and L. Fernandez, "Hardware implementation of a bounded algorithm for application of rules in a transition p-system," *SYNASC 2006: Eighth International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, Proceedings*, vol. 1, pp. 343–349, 2007.

[11] Y. Suzuki and H. Tanaka, "On a lisp implementation of a class of p systems," in *Romanian Journal of Information Science and Technology*, vol. 3, no. 2, 2000, pp. 173–186.

[12] F. Arroyo, C. Luengo, A. V. Baranda, and L. de Mingo, "A software simulation of transition p systems in haskell," *Membrane Computing*, vol. 2597, pp. 19–32, 2003.

[13] G. Ciobanu and W. Y. Guo, "P systems running on a cluster of computers," *Membrane Computing*, vol. 2933, pp. 123–139, 2004.

[14] A. Syropoulos, E. G. Mamatas, P. C. Allilomes, and K. T. Sotiriades, "A distributed simulation of transition p systems," *Membrane Computing*, vol. 2933, pp. 357–368, 2004.

[15] R. Freund and M. Oswald, "P systems with antiport rules for evolution rules," *WSEAS TRANSACTIONS on SYSTEMS*, vol. 2, no. 3, pp. 866–873, Apr 2004.

[16] M. Oswald, "Computations with 1-deterministic p systems using antiport/symport rules for evolution rules," *WSEAS TRANSACTIONS on BIOLOGY and BIOMEDICINE*, vol. 2, no. 1, pp. 280–286, Apr 2004.

[17] J. d. Frutos, L. Fernández, and F. Arroyo, "Decision trees for obtaining active rules in transition p systems," in *Tenth Workshop on Membrane Computing (WMC10)*, A. R.-N. n. Gheorghe Paun, Mario J. Pérez-Jiménez, Ed., 2009, pp. 210–217.

[18] J. A. Tejedor, L. Fernández, F. Arroyo, and A. Gutiérrez, "Algorithm of active rules elimination for application of evolution rules," in *Proceedings of the 8th Conference on 8th WSEAS International Conference on Evolutionary Computing - Volume 8*. Stevens Point, Wisconsin, USA: World Scientific and Engineering Academy and Society (WSEAS), 2007, pp. 259–267. [Online]. Available: http://portal.acm.org/citation.cfm?id=1347992.1347998

[19] J. Tejedor, L. Fernndez, F. Arroyo, and G. Bravo, "An architecture for attacking the communication bottleneck in p systems," *Artificial Life and Robotics*, vol. 12, pp. 236–240, 2008. [Online]. Available: http://dx.doi.org/10.1007/s10015-007-0474-4

[20] G. Bravo, L. Fernández, F. Arroyo, and J. A. Frutos, "A hierarchical architecture with parallel communication for implementing p systems," in *Proceedings of the Fifth International Conference Information Research and Applications i.TECH 2007*, K. I. E. Kr. Markov, Ed., vol. 1, June 2007, pp. 168–174.

[21] G. Bravo, L. Fernández, F. Arroyo, and J. Tejedor, "Master-slave distributed architecture for membrane systems implementation," in *Proceedings of the 8th Conference on 8th WSEAS International Conference on Evolutionary Computing - Volume 8*, A. Aggarwal, Ed. Stevens Point, Wisconsin, USA: World Scientific and Engineering Academy and Society (WSEAS), 2007, pp. 326–332. [Online]. Available: http://portal.acm.org/citation.cfm?id=1347992.1348009

[22] G. Bravo, L. Fernández, F. Arroyo, and M. A. Peña, "Hierarchical master-slave architecture for membrane systems implementation," in *Proceedings of the 13th International Symposium on Artificial Life and Robotics (AROB 2008)*, M. Sugisaka and H. Tanaka, Eds., Beppu, Japan, January 31 - February 2 2008, pp. 485–490.

[23] J. Castellanos, C. Martín-Vide, V. Mitrana, and J. Sempere, "Networks of evolutionary processors," *Acta Informática*, vol. 39, pp. 517–529, 2003.

[24] ——, "Solving np-complete problems with networks of evolutionary processors," *Lecture Notes in Computer Science*, vol. 2084, pp. 621–628, 2001.

Miguel Angel Peña Ph.D. student on Artificial Intelligence. He has participated in several funded project with different companies in the field of Software Development.

Gines Bravo Professor at the Universidad Politécnica de Madrid and a Ph.D. student on Artificial Intelligence. His research interests lie in the area of bio-computing, in particular in the theoretical parts of DNA computing and membrane systems, but also in the area of neural networks. He has published some papers concerning P-Systems.

Luis Fernando de Mingo Full professor at the Universidad Politécnica de Madrid since 1998 and Ph.D. on Artificial Intelligence. His research interests lie in the area of learning models, in particular in the theoretical parts of Neural Networks and Pattern Recognition, but also in the area of Networks of Evolutionary Processors. He has participated in several INTAS projects and some local projects. Member of the editorial board of IJITA, and international reviewer of KDS, ITECH conferences.