MEMBRANES DISTRIBUTION USING GENETIC ALGORITHMS

Miguel Ángel Peña, Juan Castellanos

Abstract: Membrane computing is an area of natural computing, which solves NP-complete problems simulating permeability of live cells membranes. Different researchers have developed architectures to distribute membranes in clusters. They have studied, at theoretical level, the system behavior and the minimum time it would take to executing. In this paper proposes the use of genetic algorithms to distribute membranes in processors, thanks to their evolving capacities, they achieve distributions better than random distribution. Theoretical results are compared with a set of examples, noting improvement that genetic algorithms produce on these systems and how architectures are beneficial from execution viewpoint.

Keywords: Distributed Communication, Membrane Computing, Membrane Dissolution, P-Systems Architectures, Genetic Algorithm

ACM Classification Keywords: F.1.2 Modes of Computation, I.6.1 Simulation Theory, H.1.1 Systems and Information Theory, C.2.4 Distributed Systems

Conference topic: Distributed and Telecommunication Systems

Introduction

Membrane computing, inspired in "*basic features of biological membranes*", was introduced by Gheorge Paun [Paun, 2000] 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 enables 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.

In base to this behaviour, P-System are systems that can be executed in vitro or simulated, using hardware implementations ([Petreska, 2004], [Fernandez, 2005] or [Martinez, 2007]), using software simulations ([Suzuki, 2000] or [Arroyo, 2003]) or even two researcher using a real cluster of processors ([Ciobanu, 2004] and [Syropoulos, 2004]). Currently, researchers focused on simulations by distributed software, to alleviate the sequential nature of processors, to obtain lower running time. Must be distinguish two main steps in this system evolution: applying rules (with proper selection of it), and object communication between membranes. To apply rules, [Frutos, 2009] proposed to create decision trees to determine possible rules to apply according to membrane context, and [Gil, 2009] proposed algorithms to distribute objects among rules and its application.

Objects communication between membranes depends on used distributed architecture. Since Ciobanu detected that network congestion produce higher response time, future studies have focused on searching architectures to eliminate network collision.

For each architecture, its author presents a study about time it would takes in evolving a P System, using the membranes number and communication and application time of the cluster used, like input parameters. Nevertheless, is not considering P System topology, like one of these parameters and so it is not given a formula to membranes distribution into different cluster processor. Due to internal properties of existing architectures and diversities of tree topologies possible on the system –even with the same membranes number- it is not exists some algorithm able to indicate what is the better distribution of P-System on elements in the processor network that will be used for its evolution.

Genetic algorithms, based on a idea that genes of better adapted individuals tends to survivor, are a good option to seek better membranes distributions on processors of clusters, since in the algorithm execution will be finding better distributions will reduce the time evolution of the system. This paper pretends to show how genetic algorithms improve results of evolution global time, with better membranes distributions on processors. It also pretends to study the improvement percentage of this technique and possible differences between theoretical times of each architecture with real times obtaining after a realistic distribution, like are showed by some genetic algorithms. By last, it will compared different architectures to see if the differences found in the theory are also maintained when make practical distributions.

P System definition

The first definition of a P System was published by Paun [Paun, 2000], who defined a Transition P System as:

Definition: 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 $V_{TAR} = V \times TAR$ with $TAR = \{here, out\} \cup \{in_j \mid 1 \leq j \leq n\}$. δ is a special symbol no include in V, and represent the dissolution of the membrane.

ρ_1 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 exist in the P-System that is indicated in its consequent.

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. 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. Transferring of multiset of 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:
    a. Objects do not assigned to any rule at the non-deterministic distribution.
    b. Objects created in the membrane whose target identifier, is the membrane itself.
    c. Objects created in daughter membranes whose target identifier was *out*.
    d. Objects coming from mother membrane, whose target identifier was in_j.
    e. Objects coming from dissolution of daughter membranes of membrane *j* and all objects whose their destiny was dissolved *j* daughter membrane.

Different researchers have reduced these theoretical microsteps to only two general steps, in function of number of involved membranes, without losing the characteristics of original system. In particular, the steps are: rule selection and application, and communication between membranes. The first step, which occurs on each

membrane individually and without the need of it knows about the rest of P System is divided into rules selection and the subsequent application. [Frutos, 2009] proposes the rules selection based on decision tree, and [Gil, 2009] proposed an algorithm for non-deterministic assignation of objects in the selected rules, thus like application of these rules. On step of communication between membranes, like its name describe, are involved several membranes, and thus, in clustered deployments is necessary to know the network topology that forms the P-System, and that form the different processors. Because that, the communication depends on the used architecture.

P-System Architectures

The first architecture designed specifically to membranes distribution in diverse processors of a network or a cluster was proposed by Tejedor [Tejedor, 2008]. This architecture eliminates collisions previously found in experiments when they don't take into account the topology of P System [Ciobanu, 2004]. The architecture named "partially parallel evolution with partially parallel communication" originally and subsequently known like Peer-to-Peer (P2P) it distributed several membranes in each one of processors that used.

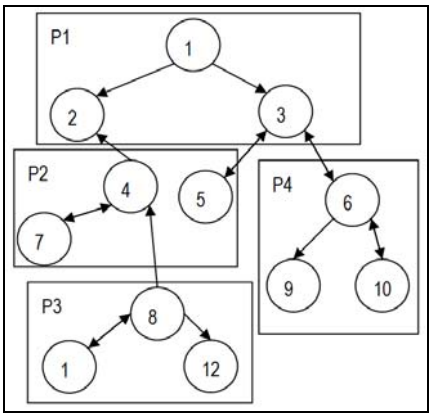


Fig. 1 P System example with 12 membranes distributed on 4 processors.

Rules selection and application step is made in parallel on each processor. So, all processors, at the same time, execute sequentially the rules selection and application on each one of its membranes. In the next step (communication between membranes), each processors, through its proxy communicates with the rest of adjacent processors according to tree topology used on membranes distribution. The communication order is determinate by topology, and in each moment only one processor is communicating. This order is observed on figure 2(a). Tejedor suggest that to know the evolution time of a P-System is necessary make a uniform distribution of membranes in processors, and this time is:

$$T_{P2P} = \left\lceil \sqrt{\frac{2MT_{com}}{T_{apl}}} \right\rceil T_{apl} + 2\left(\left\lceil \sqrt{\frac{MT_{apl}}{2T_{com}}} \right\rceil - 1 \right) T_{com} \quad (1)$$

Where T_{apl} is the time that take each membrane in its application step (considering constant for all membranes), and T_{com} is the necessary time for each processor to communicate with other. M is the membranes number.

Based on the same idea of use the tree topology and the proxy like communicator element, Bravo proposes an improvement over this architecture using the superposition of communications, so in every moment it can communicate various processors without collisions. This architecture is known like Hierarchical Peer-to-Peer (HP2P). This topology was further enhanced [Pena, 2011] to allow membranes can be dissolve and with only one message can getting lower times. The order communication is showed on figure 2(b), and the needed time for each evolution is:

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

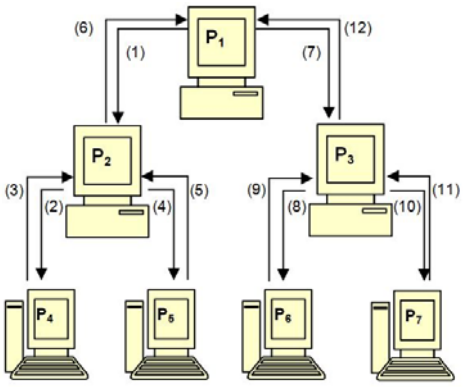
Where *A* is amplitude or processor number of children and *L* is number of processors tree levels, taking root like *L=1*. *L* and *A* values are determined by T_{com} y T_{apl} to minimize the function.

Third architecture that eliminates the restriction that processors form a tree is known like Master-Slave (MS) [Bravo, 2007b]. This architecture, where communication also is made by proxies (Figure 2(c)), each evolution step needs:

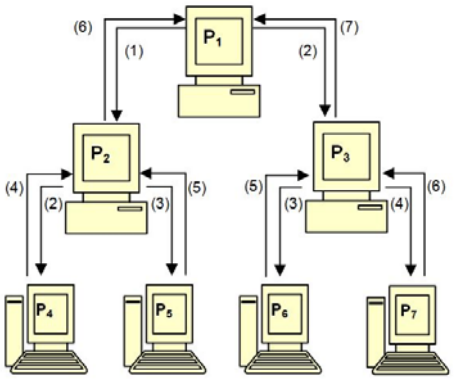
$$T_{MS} = \left\lceil \sqrt{\frac{MT_{com}}{T_{apl}}} \right\rceil T_{apl} + \left(\left\lceil \sqrt{\frac{MT_{apl}}{T_{com}}} \right\rceil + 1 \right) T_{com} \tag{3}$$

Unifying characteristics of the HP2P and MS architectures, it arises an architecture [Bravo, 2008] that uses the parallelism of HP2P architecture with elimination of restriction to make a distribution on a tree shape showed on MS architecture. Later, with introduction of membranes dissolution in P-System evolution, it was observed that behavior of HP2P and HMS is similar (figure 2(d)), but HMS besides not having a distribution in tree form, allows the overlap of steps. The time needed for each evolution depends if steps overlapping is total or partial:

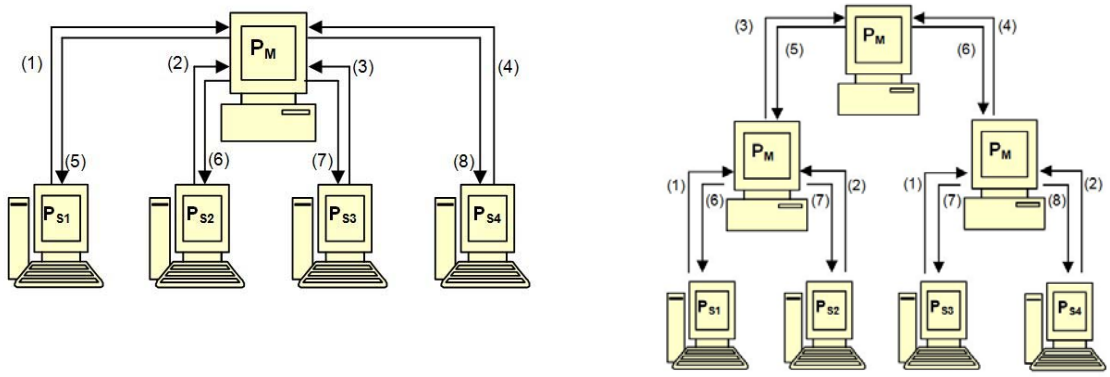
$$T_{HMS} = \begin{cases} (\mathrm{AL} + \mathrm{L} - 2)T_{com} & if \quad \dfrac{MT_{apl}}{A^{L-1}} \le (A-1)T^{com} \\ \dfrac{MT_{apl}}{A^{L-1}} + (LA + L - A - 1)T_{com} & otherwise \end{cases} \tag{4}$$



a) Peer to Peer architecture



b) Hierarchical Peer to Peer architecture



c) Master-Slave architecture

d) Hierarchical Master-Slave architecture

Fig. 2. Different distributed P-System architectures including the object communication order

Genetic algorithm

Genetic algorithms (GAs) are adaptive methods that can be used to resolve problems of searching and optimization. They are based on a genetic process of live organisms. Over generations, natural populations evolve in line with the principles of natural selection and survival of the fittest, proposed by Darwin [Darwin, 1859]. In imitation of this process, genetic algorithms are capable to creating solutions to real world problems. The evolution of these solutions to optimal values of problem depends on an appropriate codification of it. Basic principles of genetic algorithms were established by Holland [Holland, 1975], and are good described on text of Goldberg [Goldberg, 1989] or Davis [Davis, 1991]. The abstract algorithm, shows in a global way how must be implemented a genetic algorithm (Figure 3) and the parts that compose it.

```
BEGIN AGA
   Make initial population at random.
   WHILE NOT stop DO
      BEGIN
         Select parents from the population.
         Produce children from the selected parents.
         Mutate the individuals.
         Extend the population adding the children to it.
         Reduce the extend population.
      END
   Output the best individual found.
END AGA
```

Fig. 3. The pseudocode of the Abstract Genetic Algorithm (AGA).

To use a genetic algorithm on solution of a problem is needed define the characteristics and algorithms that it will use. For example, representation of individuals can be done in binary, integer or floating-point. To crossover can be used one-point crossover [Holland, 1975], n-point crossover or uniform crossover [Syswerda, 1991]...

Distribution using genetic algorithm

To use a genetic algorithm for membranes distribution, it must to decide representation to be used. It is a phenotype. As there are *m* membranes, are used *m* numbers, representing the *i-th* processor that goes into *i* membranes. Each of these numbers is an integer positive number.

It must distinguish two types of architectures, P2P and MS (HP2P and HMS are similar respectively, only changing the fitness function). To P2P distributions will be uses crossover and mutation oriented to tress and for MS are valid any evolutionary method (it will use one point crossover). On the same way, the initial population will be created on randomly way, but conserving the tree topology in case of P2P architecture.

It is used a method of proportional selection to objective function, forming a new population with new individuals, and 20% of more adapted of parents. Mutations always are performed with a 50% of probability that an individual mutates. Experimentally it has found that mutation increases the searching in the solutions space. It will stop on population $log_{1.01}(m)$. Population size will be $2m$.

With these parameters, has been proved suitability of genetic algorithms on set of 1000 P Systems of 100 membranes each one, and on 122 documented cases of 1000 membranes. Obtained results are showed on figures 4 and 5.

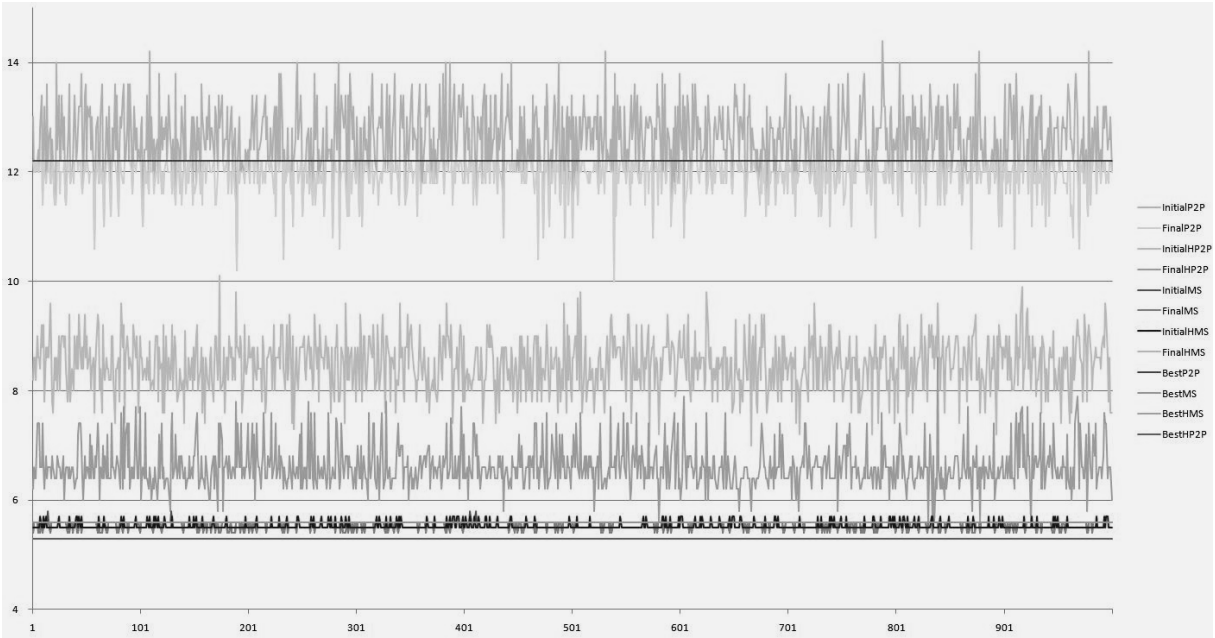


Fig. 4. Evolution times with 100 membranes

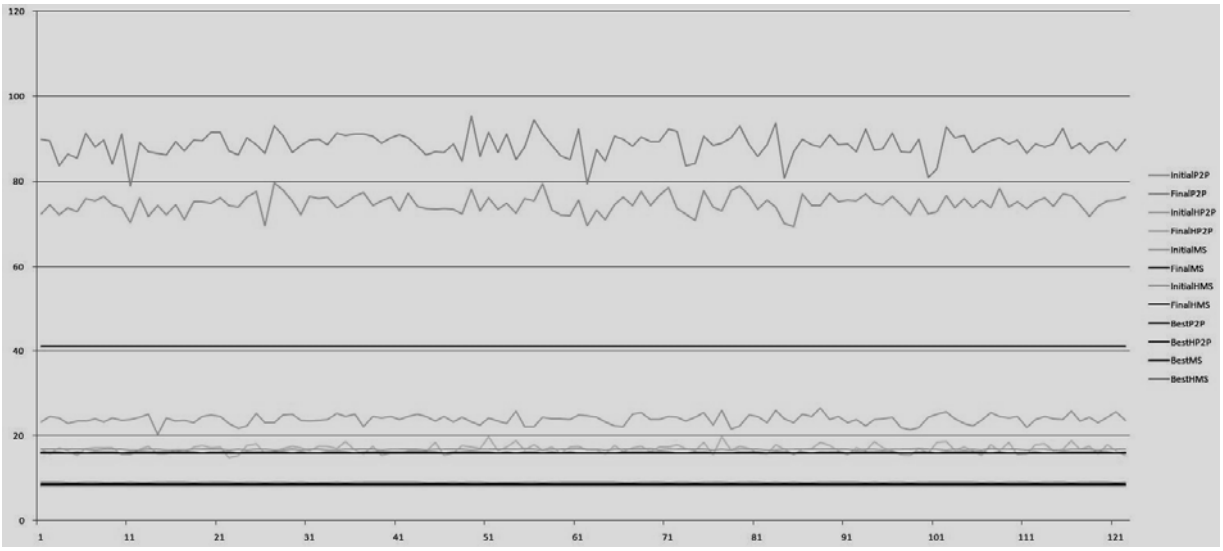


Fig. 5. Evolution times with 1000 membranes

On each graph is showed, to each P System (X axis), the needed time for its evolution (Y axis). Different lines showed: the best theoretical values for each architecture (BestP2P, BestHP2P, BestMS, BestHMS) and times used to make the evolution if is used the better solution of the initial population (initialP2P, initialHP2P, initialMS, initialHMS) or using the solution given by the algorithm (finalP2P, finalHP2P, finalHMS, finalHMS).

The first conclusion to be drawn is that there is a great difference between theoretical time of architectures and real times obtained after making the distribution. However, MS and HMS architectures time are similar because they are not penalized external communications.

It is very important pointed how genetic algorithms improve the evolution times of P System on 20%, given better distributions than the random distributions. And it is expected that with more evolutions of the genetic algorithm, the time improves even more.

Also it is needed mentioned how some solutions are even better that those theoretically calculated. It is showed in the P2P architecture for P System of 100 membranes. It is because in the theoretical calculus, rounding is applied because the processors number and the membranes number in each processor must be an integer, but in practice there are better solutions that used less processors or membranes on each processor getting better results.

Regarding to architectures, we see that the HMS architecture is not only better in theory, but there are distributions that achieve the same time and they need the least time.

Conclusion

Although, in theoretical studies shown four architectures for membranes distribution, none indicates how distributions must be done. This paper has shown the validity of genetic algorithms from random distributions, obtaining better distributions that approximates to optimal time for each architecture presented.

It has been analyzed results on architectures and was obtained that Hierarchical Master-Slave architecture is not only the best at theoretical level, but also it is easy to make membranes distributions that achieve the theoretical time; they are the best of the four architectures.

Bibliography

[Arroyo, 2003] Arroyo, F., Luengo, C., Baranda, A. V., and de Mingo, L. (2003). A software simulation of transition p systems in haskell. Membrane Computing, 2597:19-32.

[Bravo, 2007a] Bravo, G., Fernández, L., Arroyo, F., and Frutos, J. A. (2007). A hierarchical architecture with parallel communication for implementing p systems. In Kr. Markov, K. I. E., editor, Proceedings of the Fifth International Conference Information Research and Applications i.TECH 2007, volume 1, pages 168-174.

[Bravo, 2007b] Bravo, G., Fernández, L., Arroyo, F., and Tejedor, J. (2007b). Master-slave distributed architecture for membrane systems implementation. In Aggarwal, A., editor, Proceedings of the 8th WSEAS International Conference on Evolutionary Computing - Volume 8, pages 326-332, Stevens Point, Wisconsin, USA. World Scientific and Engineering Academy and Society (WSEAS).

[Bravo, 2008] Bravo, G., Fernández, L., Arroyo, F., and Peña, M. A. (2008). Hierarchical master-slave architecture for membrane systems implementation. In Sugisaka, M. and Tanaka, H., editors, Proceedings of the 13-th International Symposium on Artificial Life and Robotics (AROB 2008), pages 485-490, Beppu, Japan.

[Ciobanu 2004] Ciobanu, G. and Guo, W. Y. (2004). P systems running on a cluster of computers. Membrane Computing, 2933:123-139.

[Darwin, 1859] Darwin, C. (1859). On the Origin of Species by Means of Natural Selection. John Murray, London.

[Davis, 1991] Davis, L. (1991). Handbook of Genetic Algorithms. Van Nostrand Reinhold.

[Fernandez, 2005] Fernandez, L., Martinez, V. J., Arroyo, F., and Mingo, L. F. (2005). A hardware circuit for selecting active rules in transition p systems. Seventh International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, Proceedings, 1:415-418.

[Frutos, 2009] Frutos, J. d., Fernandez, L., and Arroyo, F. (2009). Decision trees for obtaining active rules in transition p systems. In Gheorghe Paun, Mario J. Perez-Jimenez, A. R.-N. n., editor, Tenth Workshop on Membrane Computing (WMC10), pages 210-217.

[Gil, 2009] Gil, F. J., Tejedor, J., and Fernandez, L. (2009). Fast lineal algorithm for active rules. International Journal "INFORMATION THEORIES & APPLICATIONS", 16(3):222-232.

[Goldberg, 1989] Goldberg, D. E. (1989). Genetic Algorithms in Search, Optimization & Machine Learning. Addison-Wesley, Reading, MA.

[Holland, 1975] Holland, J. H. (1975). Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor, MI.

[Martinez, 2007] Martinez, V., Arroyo, F., Gutierrez, A., and Fernandez, L. (2007). 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, 1:343-349.

[Paun, 2000] Paun, G. (2000). Computing with membranes. Journal of Computer and System Sciences, 61(1):108-143.

[Pena, 2011] Pena, M.A., Bravo, G., de Mingo, L.F. Membrane Dissolution in Distributed Architectures of P-Systems. Recent researches in applied computer and applied computational science, pages 229-234.

[Petreska, 2004] Petreska, B. and Teuscher, C. (2004). A reconfigurable hardware membrane system. Membrane Computing, 2933:269-285.

[Suzuki, 2000] Suzuki, Y. and Tanaka, H. (2000). On a lisp implementation of a class of p systems. In Romanian Journal of Information Science and Technology, volume 3, pages 173-186.

[Syropoulos, 2004] Syropoulos, A., Mamatas, E. G., Allilomes, P. C., and Sotiriades, K. T. (2004). A distributed simulation of transition p systems. Membrane Computing, 2933:357-368.

[Syswerda, 1991] Syswerda, G. (1991). Schedule optimization using genetic algorithms. In Davis, L., editor, Handbook of Genetic Algorithms, pages 332-349, New York. Van Nostrand Reinhold.

[Tejedor, 2008] Tejedor, J., Feranndez, L., Arroyo, F., and Bravo, G. (2008). An architecture for attacking the communication bottleneck in p systems. Artificial Life and Robotics, 12:236-240.

Authors' Information

Miguel Angel Peña – Dept. Inteligencia Artificial, Facultad de Informática, Universidad Politécnica de Madrid, Campus de Montegancedo, 28660 Madrid, Spain; e-mail: m.pena@fi.upm.es

Juan B. Castellanos – Dept. Inteligencia Artificial, Facultad de Informática, Universidad Politécnica de Madrid, Campus de Montegancedo, 28660 Madrid, Spain; e-mail: jcastellanos@fi.upm.es