Revisiting Sevilla Carpets: A New Tool for the P-Lingua Era

David Orellana-Martín, Carmen Graciani, Miguel Ángel Martínez-del-Amor, Agustín Riscos-Núñez, Luis Valencia-Cabrera

Research Group on Natural Computing
Department of Computer Science and Artificial Intelligence
University of Sevilla
dorelmar@gmail.com, {cgdiaz, mdelamor, ariscosn, lvalencia}@us.es

Summary. Sevilla Carpets have already been used to compare different solutions of the Subset Sum problem: either designed in the framework of P systems with active membranes (both in the case of membrane division and membrane creation), and also another one in the framework of tissue-like P systems with cell division.

Recently, the degree of parallelism and other descriptive complexity details have been found to be relevant when designing parallel simulators running on GPUs.

We present here a new way to use the information provided by Sevilla carpets, and a script that allows to generate them automatically from P-Lingua files.

1 Introduction

P systems are massively parallel computing devices, in the sense that their evolution eventually involves a great number of symbol-objects, membranes and rules. Furthermore, if we work with models where the number of membranes can increase along the computation, via creation or division of membranes, then it becomes specially difficult to describe the complexity of the computational process. Such models have actually been investigated largely in the literature, as their ability to generate an exponential number of membranes in polynomial time (making use of their intrinsic parallelism) makes them powerful tools for solving **NP**-complete problems. Indeed, several efficient solutions to these type of problems have been presented (see, e.g. [6, 16, 17, 18] or [19]).

The complexity in *time* (number of cellular steps) of the solutions obtained in this way is polynomial, but it is clear that time is not the unique variable that we need to consider in order to evaluate the complexity of such processes. This fact has been observed previously in the literature of P systems. The first paper related to this issue was [2], where G. Ciobanu, Gh. Păun and Gh. Ştefănescu presented a new way to describe the complexity of a computation in a P system,

the so-called *Sevilla Carpet*, which is an extension of the notion of Szilard language from grammars to the case when several rules are used at the same time.

In [8], the problem was revisited, introducing new parameters for the study of the descriptive complexity of P systems. Besides, several examples of a graphical representation were provided, and the utility of these parameters for comparing different solutions to a given problem was discussed. In that paper two different solutions of the Subset Sum problem, running on the same instance, were compared by using these parameters.

Sevilla Carpets have been adapted to tissue-like models, in order to describe the complexity of their computations (see [4]). There exists also an extension of the definition of Sevilla carpets to a four-dimensional manifold (see [10]) which can be used for a more verbose description of the complexity of a computation of a P system. The graphical representation of this four-dimensional manifold is carried out via projections on three-dimensional spaces.

Comparing two cellular designs that solve the same problem is not an easy task, as there are many ingredients to be taken into account. Note that given two Sevilla Carpets corresponding to P systems from different models designed to solve a decision problem, we can obtain detailed information about two single computations, but this is not enough to compare the efficiency of the two models in general.

Nonetheless, the numerical parameters obtained from these two Sevilla Carpets can give us some hints to compare the corresponding designs of solutions to the problem.

The paper is organized as follows. First, we recall the definition of the Sevilla carpets, together with a list of parameters associated with them. Then, we describe the tool that allows to generate Sevilla carpets automatically from P-Lingua files. Several examples are displayed, and we conclude the paper by providing an overview on the future directions of this ongoing work.

2 Sevilla Carpets

As pointed out in the introduction, the evolution of a P system is usually a too complex process to be evaluated only by the classical parameters from computational complexity measure, *time* and *space*. For instance, we are often interested in other types of descriptive complexity information: size of the alphabet, number of membranes (initially in the system or obtained during the computation), number of rules, etc. Another interesting parameter, especially when running software simulations, is the number of elementary operations (applications of rules) that are performed during the computation.

A possible way to describe the complexity of the evolution of a P system is by means of Sevilla carpets. They were presented in [2] as an extension of the Szilard language, which consists of all strings of rule labels describing correct derivations in a given grammar (see e.g., [11, 14] or [20]). The original framework for Szilard

language is the Chomsky hierarchy of grammars, where only one rule is used in each derivation step and, therefore, a derivation can be represented in a natural way as the string of labels corresponding to the used rules (the rule labelling is supposed to be one-to-one, we refer again to [2] for details).

Sevilla carpets are a Szilard-way to describe a computation in a P system, capturing via a bi-dimensional writing the fact that in each evolution step a P system can use not just a single rule, but a multiset of them. More precisely, the (Sevilla) carpet associated with a computation of a P system is a table with the time on the horizontal axis and the rules explicitly mentioned along the vertical axis; then, for each rule, in each step, a piece of information is given.

Ciobanu, Păun and Ştefănescu propose five variants for the Sevilla Carpets:

1. Specifying in each time unit for each membrane whether at least one rule was used in its region or not;
2. Specifying in each time unit for each rule whether it was used or not;
3. Mentioning in each time unit the number of applications of each rule; this is 0 when the rule is not used and can be arbitrarily large when the rules are dealing with arbitrarily large multisets;
4. We can also distinguish three cases: that a rule cannot be used, that a rule can be used but it is not because of the nondeterministic choice and that a rule is actually used;
5. A further possibility is to assign a cost to each rule, and to multiply the number of times a rule is used with its cost.

In what follows, we shall focus on the third variant (studied in [8]), that is, in each cell of the table we specify the number of applications of the corresponding rule in the considered step. Note that there is a huge amount of data contained in a Sevilla carpet, describing a computation of a P system, but these data are presented in a rough way, just a listing of which rules were applied at each step and how many times.

In order to facilitate reading the whole table just in one glance, we can obtain a three-dimensional representation of it in a natural way, expressing the numbers in each cell over a third axis (see Figure 1).

However, such a three-dimensional picture may not provide significant information by itself, specially in the case of comparing several carpets. In order to be able to "evaluate" the massive amount of information contained in the table of the Sevilla carpet, we need to extract some figures or statistics. The first natural parameters related with Sevilla carpets were defined in [2]: the sum of all the cells in the table (*weight*) and the total amount of cells in the table (*surface*). It is clear that the values of the weight and the surface of a Sevilla carpet give a general intuition on the complexity of the underlying computation. On one hand, the weight measures up the total number of applications of rules along the computation, which corresponds to the intuitive notion of "cost". On the other hand, the surface tells us about the $space \times time$ complexity of the system that is carrying out the computation, as the number of rows is the number of rules of the system

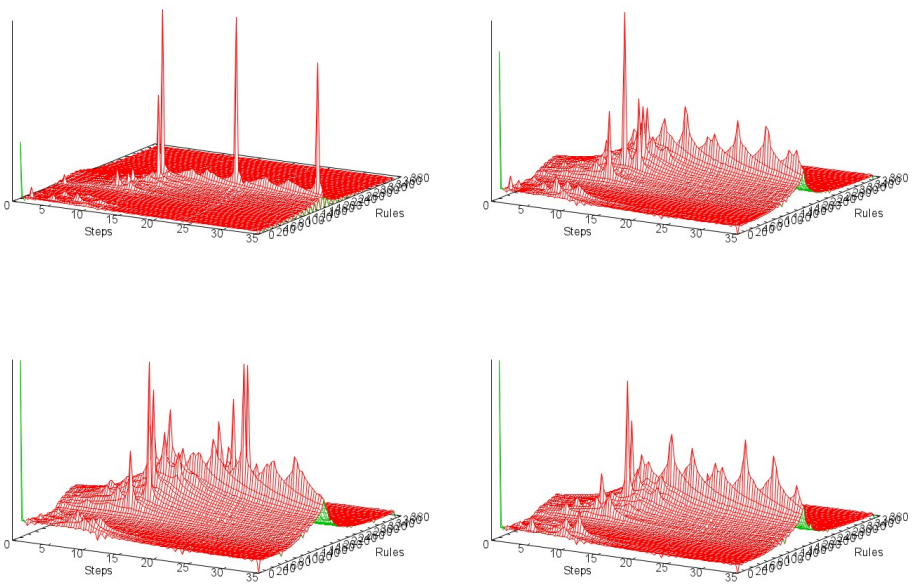

Fig. 1. Sevilla Carpets associated with a solution to SAT using membrane division (running on four different instances)

and the number of columns is the number of cellular steps that the computation performs.

2.1 Parameters for the Descriptive Complexity

The following parameters have been proposed in the literature:

- **Weight:** It is defined in [2] as the sum of all the elements in the carpet, i.e., as the total number of applications of rules along the computation. The weight measures up the total number of applications of rules along the computation, which corresponds to the intuitive notion of "cost" of the computation.
- **Surface:** It is the multiplication of the number of steps by the total number of the rules used by the P system, was also introduced in [2]. It can be considered as the *potential size* (*space* $\times$ *time* bounds) of the computation. From a computational point of view we are not only interested on P systems which halt in a small number of steps, but in P systems which use a small amount of resources. The *surface* measures the resources used in the design of the P system. Graphically, it represents the surface where the Sevilla Carpet lies on.

- **Height:** Introduced in [8], the height of a Sevilla carpet captures the intuition of a peak in the computation, and it is defined as the maximum number of simultaneous (in one step) applications of any rule all over the computation. Graphically, it represents the highest point reached by the Sevilla carpet.
- **Average Weight:** It is calculated by dividing the *weight* to the *surface* of the Sevilla Carpet. This concept provides a relation between both parameters which gives an index on how the P system exploits its massive parallelism. This parameter was also introduced in [8].
- **Variance:** It is calculated as the sum of the squared differences between the elements of the carpet and the average weight, divided by the surface.
  This parameter was introduced in [10], and it indicates if the points are near or far from the average. That is, a high variance value indicates that there is a very large number of applications of rules performed in a few steps in the computation, while in the rest of the steps the activity can be considered low (see the peaks and valleys in Figure 1). On the other hand, a low variance value leads to think that the *work load* is more balanced.

One of the motivations of this paper is to facilitate the use of the information provided by the Sevilla carpets in the context of GPU-based simulators for P systems (see e.g. [1, 12, 13]). It has been observed that the speed-up obtained by such parallel simulations (with respect to standard sequential simulators) highly depends on how distributed the rule applications are during the simulated computation. Informally speaking, the underlying intuition agrees with the observation from [9]:

> *a bad design of a P system consists of a P system which does not exploit its parallelism, that is, working as a sequential machine: in each step only one object evolve in one membrane whereas the remaining objects do not evolve. On the other hand, a good design consists of a P system in which a huge amount of objects are evolving simultaneously in all membranes. If both P systems perform the same task, it is obvious that the second one is a better design that the first one.*

More precisely, the notion of *GP-systems* (GPU-oriented P systems) is introduced in [12], and also some specific parameters related to the performance of GPU simulations:

- **Density of objects per membrane:** general purpose parallel simulators usually save threads for the whole alphabet, so the more different objects are in the membrane, the higher thread usage.
- **Rule intensity:** some designs include rules associated with auxiliary objects which are only applied once in the whole computation (e.g. counters or synchronization routines). This cannot be parallelized.
- **Communication among membranes:** the skin is executed on the CPU, and every time we need to communicate objects through PCI express bus, this process slows down the process.

2.2 Projections of Sevilla Carpets

The graphical representation in 3D of the Sevilla carpet of a computation provides an intuitive representation of the computational effort associated with each rule in each step. Nevertheless, sometimes it is better to have a more concise representation of this effort. In these cases we can consider the *projections* of the Sevilla carpet. These *projections* are obtained in two different ways:

- By considering the whole number of applications of rules for each step. If the application of a rule has an associated *cost*, this projection will give information about the whole cost of each step.
- By considering for a given rule the whole number of steps in which it has been applied. This provides information about the *utility* of a rule: if we have designed a solution of a problem where several rules are used a low number of times along the computation, we can consider to replace these rules by another rule with the same function.

3 Tool description

In this paper we present a tool that automatically generates Sevilla carpets. The script receives the description of a P system in P-Lingua syntax (a plain text file with .pli extension, we refer to [5, 21] for details), and produces a jpg image of a 3D representation of the Sevilla Carpet associated to (one computation of) the given P system.

In this first version, we have used *python* as programming language, and *gnuplot* for producing the graphical output.

The tool works as follows: first, pLinguaCore library computes a single computation, then the python script parses the results and generates a matrix with the numerical data corresponding to the points of the Carpet. Finally, gnuplot is called and it renders the matrix to a 3D graph that represents the Sevilla carpet associated with this computation.

Note that this represents a notable improvement from previous works, where the process was done manually, after processing the output of simulators for P systems with active membranes written in Prolog ([3, 7]). Now it became a much faster and simpler process, we avoid noise in data due to mistakes when manually building the matrix, and we can cover all the models of P systems included in the P-Lingua framework.

3.1 The algorithm

The parser is designed to work on a text file generated by pLinguaCore library containing verbose information about a computation.

More precisely, the file describes the sequence of configurations, and a list of which rules were applied at each step, and how many times. The format used in such file is as follows:

STEP k:
Rules selected for MEMBRANE ID: x, Label: y, Charge: z
n * #r q
...

where:

- k represents the current step.
- r is the label (usually a number) of the applied rule.
- q is the rule itself.
- n represents the number of times that rule r is applied in step k.

We have thus all the necessary information to generate a matrix M with the data to be plotted.

The parser reads the whole file, paying attention to the lines that have a "Step" statement, or an applied rule, and ignoring the rest.

Whenever a *Step* is found, we move to the next row of the matrix, and whenever a *rule* line is read, we apply the next:

$$M_{current\ step-1,r-1} = M_{current\ step-1,r-1} + n$$

If the rule r, isn't applied in the step k, then we can do:

$$M_{k-1,r-1} = 0$$

From this matrix, we can successfully obtain the x axis (*steps*, the rows of the matrix), the y axis (*rules*, columns of the matrix) and the z axis (*number of times*, values of the matrix).

We can then, with *gnuplot* generate a graph with the data obtained, and create the Sevilla Carpet.

```
STEP: 3

Rules selected for MEMBRANE ID: 1, Label: 2, Charge: 0
1 * #109 d{1}[]'2 --> [d{2}]
1 * #112 [s{1,1} --> s{1,2}]'2

Rules selected for MEMBRANE ID: 2, Label: 2, Charge: 0
1 * #109 d{1}[]'2 --> [d{2}]
1 * #113 [s{2,1} --> s{2,2}]'2
```

Fig. 2. A few lines extracted from a computation file

In the example shown in Figure 2, for step 3 we get the following values in the table: $(3, 109, 2)$, $(3, 112, 1)$, $(3, 113, 1)$, and $(3, r, 0)$ for any other rule label r.

4 Examples

We have done some examples, using .pli files corresponding to solutions to SAT, KNAPSACK, PARTITION and SUBSETSUM designed by means of families of P systems with active membranes.

Here are the results of some of the computations:

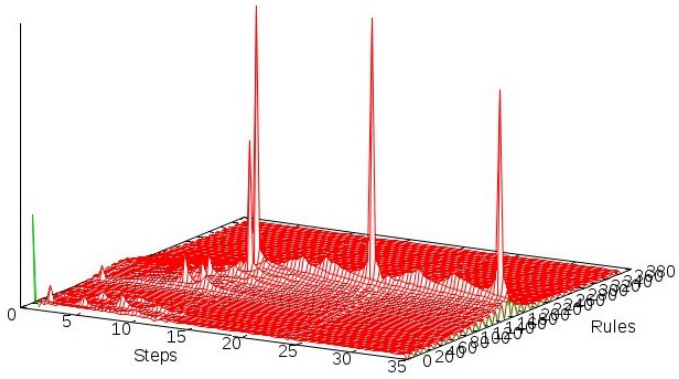

Fig. 3. SAT simulation

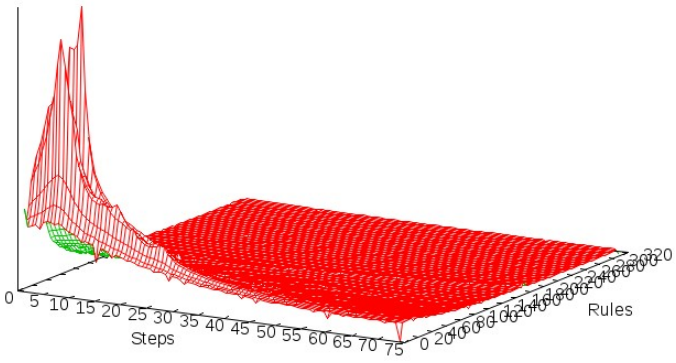

Fig. 4. KNAPSACK simulation

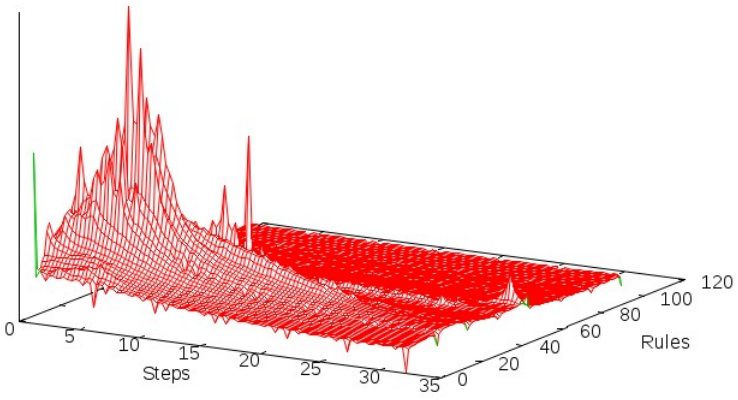

Fig. 5. PARTITION simulation

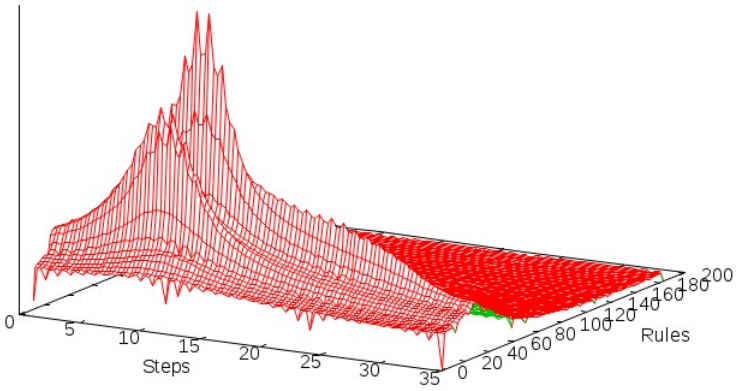

Fig. 6. SUBSETSUM simulation

5 Final Remarks and Future Work

We would like to remark that the information extracted from a Sevilla carpet (and from its associated parameters) should always be interpreted being aware that it only refers to one computation. Nevertheless, we still believe that it is a useful instrument that can be used for example as a guide to find possible refinements on the definition of the simulated P system (e.g. in order to "translate" it into an equivalent GP system), or as an assistant for designing ad-hoc simulators for particular families of P systems (e.g. solutions to hard problems using active membranes)[1].

The next improvement to be added in the near future is to get the values of all parameters together with the graphical 3D representation. Some other customization possibilities can also be considered for the script, like adding options for which kind of Carpet is wanted.

Another interesting possibility is to plug a Sevilla carpet module (adapting the script presented here) into the pLinguaCore library, in such a way that the matrix can be generated on-the-fly as the computation is being simulated, thus avoiding parsing the output file. We are also considering to bring the idea of the script into MeCoSim as an extension. MeCoSim [15, 22] is a general purpose software tool to model, design, simulate, analyze and verify different types of models based on P systems (specially intended for PDP systems). It is a highly customizable tool, allowing to easily configure ad-hoc GUIs for each case study to be modeled. Therefore, it seems reasonable to offer a Sevilla carpet as one possible output that the user can be interested on, together with some other descriptive complexity details (for example, number of membranes generated during the computation).

In the case of probabilistic P systems, it might be interesting to extract several samples of computations and then use the average values in order to generate the Sevilla carpet and the associated parameters. Actually, MeCoSim already includes the possibility to run several computations when working with models for ecosystems designed using PDP systems, and then uses the statistical information gathered to plot the output graphics.

Acknowledgements

The authors acknowledge the support of the Project TIN2012-37434 of the Ministerio de Economía y Competitividad of Spain, cofinanced by FEDER funds.

[1] This work direction seems to be worth studying, in the context of the PMCGPU project [23].

References

1. Cecilia, J.M.; Garca, J.M.; Guerrero, G.D.; Martnez-del-Amor, M.A.; Prez-Jimnez, M.J.; Ujaldn, M. The GPU on the simulation of cellular computing models *Soft Computing*, **16** (2), 2012, 231–246.
2. Ciobanu, G.; Păun, Gh.; Ştefănescu, Gh. Sevilla Carpets Associated with P Systems, in M. Cavaliere, C. Martín–Vide and Gh. Păun (eds.), *Proceedings of the Brainstorming Week on Membrane Computing*, Tarragona, Spain, 2003, Report RGML 26/03, 135-140.
3. Cordón-Franco, C.; Gutiérrez-Naranjo, M.A.; Pérez-Jiménez, M.J.; Sancho-Caparrini, F. A Prolog simulator for deterministic P systems with active membranes, *New Generation Computing*, **22** (4), 2004, 349–363.
4. Daz Pernil, D., Gallego-Ortiz, P., Gutirrez Naranjo M.A., Prez Jimnez M.J., Riscos Nez A. Descriptional Complexity of Tissue-like P Systems with Cell Division. In C.S. Calude et al. (eds.) Unconventional Computation. Lecture Notes in Computer Science, Springer-Verlag, Berlin-Heidelberg, 5715 (2009), 168-178.
5. Daz Pernil, D.; Prez–Hurtado, I.; Prez-Jimnez, M.J.; Riscos-Nez, A. A P-lingua programming environment for Membrane Computing. *Lecture Notes in Computer Science*, **5391** (2009), 187–203.
6. Gutiérrez-Naranjo, M.A.; Pérez-Jiménez, M.J.; Riscos-Núñez, A. A Fast P System for Finding a Balanced 2-Partition, *Soft Computing*. Springer. To appear.
7. M. A. Gutiérrez-Naranjo, M. J. Pérez-Jiménez, A. Riscos-Núñez, A Simulator for Confluent P Systems. In M. A. Gutiérrez-Naranjo, A. Riscos-Núñez, F. J. Romero-Campero, D. Sburlan (eds.), *Third Brainstorming Week on Membrane Computing* Fénix Editora, Sevilla, 2005, 169–184.
8. M. A. Gutiérrez-Naranjo, M.J. Pérez-Jiménez, A. Riscos-Núñez, On Descriptive Complexity of P Systems. In: G. Mauri, Gh. Păun, M. J. Pérez-Jiménez, G. Rozenberg, A. Salomaa (eds.), *Membrane Computing*. LNCS **3365**, Springer-Verlag, 2005, 320–330.
9. M. A. Gutiérrez-Naranjo, M.J. Pérez-Jiménez, A. Riscos-Núñez. On the Degree of Parallelism in Membrane Systems. Theoretical Computer Science, 372 (2-3), (2007) 183–195.
10. M. A. Gutiérrez-Naranjo, M.J. Pérez-Jiménez, A. Riscos-Núñez. Multi-dimensional descriptional complexity of P systems. Journal of Automata, Languages and Combinatorics 12 (2007) 1/2, 167179. A preliminar version in Proceedings of the 7th International Workshop on Descriptional Complexity of Formal Systems, Como, Italy, June 30 - July 2, 2005, pp. 134-145.
11. Mäkinen, E. A Bibliography on Szilard Languages, Dept. of Computer and Information Sciences, University of Tampere, `http://www.cs.uta.fi/reports/pdf/Szilard.pdf`
12. Martnez-del-Amor, M. A. *Accelerating Membrane Systems Simulators using High Performance Computing with GPU* (PhD Thesis), 2013.
13. Martnez-del-Amor, M. A.; Prez-Hurtado, I.; Prez-Jimnez, M.J.; Cecilia J.M.; Guerrero, G.D.; Garca, J.M. Simulating active membrane systems using GPUs. In Gh. Păun et al (eds.) *10th Workshop on Membrane Computing*, 369–384.
14. Mateescu, A. and Salomaa, A. Aspects of Classical Language Theory, in G. Rozenberg and A. Salomaa (eds.), *Handbook of Formal Languages* (vol. 1), Springer-Verlag, Berlin Heidelberg, 1997.

15. Prez-Hurtado, I.; Valencia, L.; Prez-Jimnez, M.J.; Colomer, M.A.; Riscos-Nez, A. MeCoSim: A general purpose software tool for simulating biological phenomena by means of P Systems. In K. Li, Z. Tang, R. Li, A.K. Nagar, R. Thamburaj (eds.) *Proceedings 2010 IEEE Fifth International Conference on Bio-inpired Computing: Theories and Applications (BIC-TA 2010)*, IEEE Press, Volume 1, September 23-26, 2010, Changsha, China, ISBN 978-1-4244-6439-5, pp. 637–643.
16. Pérez-Jiménez, M.J.; Riscos-Núñez, A. Solving the Subset Sum Problem by Active Membranes, *New Generation Computing*, Vol. 23, num. 4 (2005), 367-384.
17. Pérez-Jiménez, M.J.; Riscos-Núñez, A. A Linear Solution for the Knapsack Problem Using Active Membranes, in C. Martín-Vide, G. Mauri, Gh. Păun, G. Rozenberg and A. Salomaa (eds.),*Membrane Computing*. Lecture Notes in Computer Science, **2933**, 2004, 250–268.
18. Pérez-Jiménez, M.J.; Romero-Jiménez, A.; Sancho-Caparrini, F. A Polynomial Complexity Class in P systems Using Membrane Division, in E. Csuhaj–Varjú, C. Kintala, D. Wotschke, and Gy. Vaszyl (eds.), *Proceedings of the 5th Workshop on Descriptional Complexity of Formal Systems*, Budapest, Hungary, 2003, 284–294.
19. Pérez-Jiménez, M.J.; Romero-Jiménez, A.; Sancho-Caparrini, F. Solving VALIDITY Problem by Active Membranes with Input, in M. Cavaliere, C. Martín-Vide, Gh. Păun (eds), *Proceedings of the Brainstorming Week on Membrane Computing*, Tarragona, Spain, 2003, Report RGML 26/03, 279–290.
20. Salomaa, A. *Formal Languages*, Academic Press, New York, 1973.
21. `http://www.p-lingua.org`
22. `http://www.p-lingua.org/mecosim/`
23. `http://sourceforge.net/p/pmcgpu/`