Cell-like Versus Tissue-like P Systems by Means of Sevilla Carpets

Daniel Díaz-Pernil[1], Pilar Gallego-Ortiz[2], Miguel A. Gutiérrez-Naranjo[2], Mario J. Pérez-Jiménez[2], Agustín Riscos-Núñez[2]

[1] Research Group on Computational Topology and Applied Mathematics
Departament of Applied Mathematics I
University of Sevilla, Avda. Reina Mercedes s/n, 41012, Sevilla, Spain
sbdani@us.es

[2]Research Group on Natural Computing
Department of Computer Science and Artificial Intelligence
University of Sevilla, Avda. Reina Mercedes s/n, 41012, Sevilla, Spain
pigaor@gmail.com; {magutier,marper,ariscosn}@us.es

Summary. Sevilla Carpets are a handy tool for comparing computations performed by different systems solving the same problem. Such Sevilla Carpets provide on one hand quantitative information through parameters such as Weight, Surface and Average weight, and on the other hand they also provide a fast glimpse on the complexity of the computation thanks to their graphical representation.

Up to now, Sevilla Carpets were only used on Cell-like P systems. In this paper we present a first comparison by means of Sevilla Carpets of the computations of three P systems (designed within different models), all of them solving the same instance of the Subset Sum problem. Two of these solutions use Cell-like P systems with active membranes, while the third one uses Tissue-like P systems with cell division.

1 Introduction

Comparing two cellular designs that solve the same problem is not an easy task, as there are many ingredients to be taken into account. Moreover, in the case of P systems where the number of membranes increases along the computation, the problem of describing the complexity of the computational process becomes specially hard. The complexity in *time* (number of parallel cellular steps) of these solutions is polynomial, but it is clear that the time is not the unique variable that we need to consider in order to evaluate the complexity of the process. This fact has been observed previously in the literature of P systems. The first paper related to this problem was [1], where Ciobanu, Păun and Ştefănescu presented a new way to describe the complexity of a computation in a P system. The so-called

Sevilla Carpet was introduced as an extension of the notion of Szilard language from grammars to the case when several rules are used at the same time.

In [4], 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.

In this paper we adapt Sevilla Carpets to tissue-like models, in order to describe the complexity of the computations.

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. In Section 2 we recall the definition of tissue-like P systems with cell division. Section 3 shows a solution of the Subset Sum problem in the model above presented. In Section 4 we revisit the definition of Sevilla Carpets and its associated parameters. Section 5 shows a comparison among two different solutions to the Subset Sum problem in the framework of P systems with active membranes and one solution designed with tissue-like P systems with cell division, all of them running on the same instance. Some final remarks are also provided.

2 Tissue-like P Systems with Cell Division

Tissue-like P systems with cell division is a well-established P system model presented by Gh. Păun *et al.* in [8]. In this section we briefly recall its main features. The biological inspiration for this model is that alive tissues are not *static* network of cells, since cells are duplicated via mitosis in a natural way.

Formally, a *tissue-like P system with cell division* of degree $q \geq 1$ is a tuple of the form

$$\Pi = (\Gamma, \mathcal{E}, w_1, \ldots, w_q, \mathcal{R}, i_0),$$

where:

1. Γ is a finite *alphabet*, whose symbols will be called *objects*.
2. $\mathcal{E} \subseteq \Gamma$.
3. $w_1, \ldots, w_q$ are strings over Γ representing the multisets of objects associated with the cells in the initial configuration.
4. $\mathcal{R}$ is a finite set of rules of the following form:
   (a) *Communication rules*: $(i, u/v, j)$, for $i, j \in \{0, 1, 2, \ldots, q\}, i \neq j$, $u, v \in \Gamma^*$.

(b) *Division rules*: $[a]_i \rightarrow [b]_i[c]_i$, where $i \in \{1, 2, \ldots, q\}$ and $a, b, c \in \Gamma$.

5. $i_0 \in \{0, 1, 2, \ldots, q\}$.

A tissue-like P system with cell division of degree $q \geq 1$ can be seen as a set of q cells (each one consisting of an elementary membrane) labeled by $1, 2, \ldots, q$. We will use 0 to refer to the label of the environment, and i_0 denotes the output region (which can be the region inside a cell or the environment).

The communication rules determine a virtual graph, where the nodes are the cells and the edges indicate if it is possible for pairs of cells to communicate directly. This is a dynamical graph, because new nodes can appear produced by the application of division rules.

The strings $w_1, \ldots, w_q$ describe the multisets of objects initially placed in the q cells of the system. We interpret that $\mathcal{E} \subseteq \Gamma$ is the set of objects placed in the environment, each one of them in an arbitrary large amount of copies.

The communication rule $(i, u/v, j)$ can be applied over two cells i and j such that u is contained in cell i and v is contained in cell j. The application of this rule means that the objects of the multisets represented by u and v are interchanged between the two cells.

The division rule $[a]_i \rightarrow [b]_i[c]_i$ is applied over a cell i containing object a. The application of this rule divides this cell into two new cells with the same label. All the objects in the original cell are replicated and copied in each of the new cells, with the exception of the object a, which is replaced by the object b in the first one and by c in the other one.

Rules are used as usual in the framework of membrane computing, that is, in a maximally parallel way (a universal clock is considered). In one step, each object in a membrane can only be used for one rule (non-deterministically chosen when there are several possibilities), but any object which can participate in a rule of any form must do it, i.e, in each step we apply a maximal set of rules. This way of applying rules has only one restriction when a cell is divided, the division rule is the only one which is applied for that cell in that step; the objects inside that cell cannot be communicated in that step.

The main features of this model, from the computational point of view, are that cells have not polarizations (the contrary holds in the cell-like model of P systems with active membranes); the cells obtained by division have the same labels as the original cell and if a cell is divided, its interaction with other cells or with the environment is blocked during the mitosis process. In some sense, this means that while a cell is dividing it closes the communication channels with other cells and with the environment.

2.1 Recognizer Tissue-like P Systems with Cell Division

Complexity classes within Membrane Computing have been usually studied in the framework of *decision problems*. Let us recall that a decision problem is a pair (I_X, θ_X) where I_X is a language over a finite alphabet (whose elements are called *instances*) and θ_X is a total boolean function over I_X.

In order to study the computational efficiency for solving **NP**-complete decision problems, a special class of tissue P systems with cell division is introduced in [8]: *recognizer tissue P systems*. The key idea of such recognizer systems is the same one as from recognizer P systems with cell-like structure.

Recognizer cell-like P systems were introduced in [10] and they are the natural framework to study and solve decision problems within Membrane Computing, since deciding whether an instance of a given problem has an affirmative or negative answer is equivalent to deciding if a string belongs or not to the language associated with the problem.

In the literature, recognizer cell-like P systems are associated with P systems with *input* in a natural way. The data encoding to an instance of the decision problem has to be provided to the P system in order to compute the appropriate answer. This is done by codifying each instance as a multiset placed in an *input membrane*. The output of the computation (`yes` or `no`) is sent to the environment, in the last step of the computation. In this way, cell-like P systems with input and external output are devices which can be seen as black boxes, in the sense that the user provides the data before the computation starts, and then waits *outside* the P system until it sends to the environment the output in the last step of the computation.

A recognizer tissue-like P system with cell division of degree $q \geq 1$ is a tuple

$$\Pi = (\Gamma, \Sigma, \mathcal{E}, w_1, \ldots, w_q, \mathcal{R}, i_{in}, i_0)$$

where

- $(\Gamma, \mathcal{E}, w_1, \ldots, w_q, \mathcal{R}, i_0)$ is a tissue-like P system with cell division of degree $q \geq 1$ (as defined in the previous section), $i_0 = env$ and $w_1, \ldots, w_q$ strings over $\Gamma \setminus \Sigma$.
- The working alphabet Γ has two distinguished objects `yes` and `no`, present in at least one copy in some initial multisets $w_1, \ldots, w_q$, but not present in $\mathcal{E}$.
- Σ is an (input) alphabet strictly contained in Γ.
- $i_{in} \in \{1, \ldots, q\}$ is the input cell.
- All computations halt.
- If $\mathcal{C}$ is a computation of Π, then either the object `yes` or the object `no` (but not both) must have been released into the environment, and only in the last step of the computation.

The computations of the system Π with input $w \in \Sigma^*$ start from a configuration of the form $(w_1, w_2, \ldots, w_{i_{in}} w, \ldots, w_q; \mathcal{E})$, that is, after adding the multiset w to the contents of the input cell i_{in}. We say that the multiset w is *recognized* by Π if and only if the object `yes` is sent to the environment, in the last step of the corresponding computation. We say that $\mathcal{C}$ is an accepting computation (respectively, rejecting computation) if the object `yes` (respectively, `no`) appears in the environment associated to the corresponding halting configuration of $\mathcal{C}$.

Definition 1. *We say that a decision problem $X = (I_X, \theta_X)$ is solvable in polynomial time by a family $\mathbf{\Pi} = \{\Pi(n) \mid n \in \mathbb{N}\}$ of recognizer tissue-like P systems with cell division if the following holds:*

- *The family $\mathbf{\Pi}$ is* polynomially uniform *by Turing machines, that is, there exists a deterministic Turing machine working in polynomial time which constructs the system $\Pi(n)$ from $n \in \mathbb{N}$.*
- *There exists a pair (cod, s) of polynomial-time computable functions over I_X (called a polynomial encoding of I_X in $\mathbf{\Pi}$) such that:*
  - *for each instance $u \in I_X$, $s(u)$ is a natural number and $cod(u)$ is an input multiset of the system $\Pi(s(u))$;*
  - *the family $\mathbf{\Pi}$ is* polynomially bounded *with regard to (X, cod, s), that is, there exists a polynomial function p, such that for each $u \in I_X$ every computation of $\Pi(s(u))$ with input $cod(u)$ is halting and, moreover, it performs at most $p(|u|)$ steps;*
  - *the family $\mathbf{\Pi}$ is* sound *with regard to (X, cod, s), that is, for each $u \in I_X$, if there exists an accepting computation of $\Pi(s(u))$ with input $cod(u)$, then $\theta_X(u) = 1$;*
  - *the family $\mathbf{\Pi}$ is* complete *with regard to (X, cod, s), that is, for each $u \in I_X$, if $\theta_X(u) = 1$, then every computation of $\Pi(s(u))$ with input $cod(u)$ is an accepting one.*

In the above definition we have defined every P system $\Pi(n)$ to be *confluent*, in the following sense: every computation of a system with the *same* input multiset must always give the *same* answer.

3 A Solution for the Subset Sum Problem

For the study of the Sevilla Carpet in tissue-like P systems with Cell division we take the computation of one P systems of the family presented in [2]. In such a paper a uniform family of tissue-like P systems with cell division solving the Subset Sum problem was presented.

The Subset Sum problem is the following one: *Given a finite set A, a weight function, $w : A \to \mathbb{N}$, and a constant $k \in \mathbb{N}$, determine whether or not there exists a subset $B \subseteq A$ such that $w(B) = k$.*

The Subset Sum problem was solved in a linear time by a family of recognizer tissue P systems with cell division. The resolution was addressed via a brute force algorithm.

A tuple $(n, (w_1, \ldots, w_n), k)$ was used to represent an instance of the problem, where n stands for the size of $A = \{a_1, \ldots, a_n\}$, $w_i = w(a_i)$, and k is the constant given as input for the problem.

Let $A = \{a_1, \ldots, a_n\}$ be a finite set, $w : A \longrightarrow \mathbb{N}$ a weight function with $n = |A|$ and $k \in \mathbb{N}$. Let $g : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ be a *function* defined by $g(n, k) = ((n + k)(n + k + 1/2)) + n$. This function is primitive recursive and bijective between $\mathbb{N}^2$ and

$\mathbb{N}$ and computable in polynomial time. Let us denote by $u = (n, (w_1, \ldots, w_n), k)$, where $w_i = w(a_i)$, $1 \leq i \leq n$, the given instance of the problem. We define the polynomially computable function $s(u) = g(n, k)$.

We will provide a family of tissue P systems where each P system solves all the instances of the SUBSET SUM problem with the same size. The weight function w of the concrete instance will be provided via an input multiset determined via the function $cod(u) = \{\{v_i^j : w(a_i) = j \wedge 1 \leq i \leq n\}\} \cup \{\{q^k\}\}$, where v_i^j (i.e., j copies of object v_i) represents that j is the weight of the element a_i.

Next, we will provide a family of recognizer tissue P systems with cell division which solve the SUBSET SUM problem in linear time. For each $(n, k) \in \mathbb{N}^2$ we will consider the system $\Pi(n, k) = (\Gamma, \Sigma, \omega_1, \omega_2, \mathcal{R}, \mathcal{E}, i_{in}, i_0)$, where

- $\Gamma = \Sigma(n) \cup \{A_i, B_i \mid 1 \leq i \leq n\}$
  $\cup \{a_i \mid 1 \leq i \leq n + \lceil \log n \rceil + \lceil \log(k+1) \rceil + 11\}$
  $\cup \{c_i \mid 1 \leq i \leq n+1\}$
  $\cup \{d_i \mid 1 \leq i \leq \lceil \log n \rceil + \lceil \log(k+1) \rceil + 4\}$
  $\cup \{e_i \mid 1 \leq i \leq \lceil \log n \rceil + 1\}$
  $\cup \{B_{ij} \mid 1 \leq i \leq n \wedge 1 \leq j \leq \lceil log(k+1) \rceil + 1\}$
  $\cup \{b, D, p, q, g_1, g_2, f_1, T, S, N, \texttt{yes}, \texttt{no}\}$
- $\Sigma = \{v_i \mid 1 \leq i \leq n\}$
- $\omega_1 = a_1\, b\, c_1\, \texttt{yes}\, \texttt{no}$
- $\omega_2 = DA_1 \cdots A_n$
- $\mathcal{R}$ is the following set of rules:
  1. *Division rules:*
     $r_{1,i} \equiv [A_i]_2 \rightarrow [B_i]_2[\lambda]_2$ for $i = 1, \ldots, n$
  2. *Communication rules:*
     $r_{2,i} \equiv (1, a_i/a_{i+1}, 0)$ for $i = 1, \ldots, n + \lceil \log n \rceil + \lceil \log(k+1) \rceil + 10$
     $r_{3,i} \equiv (1, c_i/c_{i+1}^2, 0)$ for $i = 1, \ldots, n$
     $r_4 \equiv (1, c_{n+1}/D, 2)$
     $r_5 \equiv (2, c_{n+1}/d_1 e_1, 0)$
     $r_{6,i} \equiv (2, e_i/e_{i+1}^2, 0)$ for $i = 1, \ldots, \lceil \log n \rceil$
     $r_{7,i} \equiv (2, d_i/d_{i+1}, 0)$ for $i = 1, \ldots, \lceil \log n \rceil + \lceil \log(k+1) \rceil + 3$
     $r_{8,i} \equiv (2, e_{\lceil \log n \rceil + 1} B_i / B_{i1}, 0)$ for $i = 1, \ldots, n$
     $r_{9,i,j} \equiv (2, B_{ij}/B_{ij+1}^2, 0)$ for $i = 1, \ldots, n$, $j = 1, \ldots, \lceil \log(k+1) \rceil$
     $r_{10,i} \equiv (2, B_{i\lceil \log(k+1) \rceil + 1} v_i / p, 0)$ for $i = 1, \ldots, n$
     $r_{11} \equiv (2, pq/\lambda, 0)$
     $r_{12} \equiv (2, d_{\lceil \log n \rceil + \lceil \log(k+1) \rceil + 4}/g_1 f_1, 0)$
     $r_{13} \equiv (2, f_1 p/\lambda, 0)$
     $r_{14} \equiv (2, f_1 q/\lambda, 0)$
     $r_{15} \equiv (2, g_1/g_2, 0)$
     $r_{16} \equiv (2, g_2 f_1/T, 0)$
     $r_{17} \equiv (2, T/\lambda, 1)$
     $r_{18} \equiv (1, bT/S, 0)$
     $r_{19} \equiv (1, S\texttt{yes}/\lambda, 0)$
     $r_{20} \equiv (1, a_{n+\lceil \log n \rceil + \lceil \log(k+1) \rceil + 11} b/N, 0)$

$$r_{21} \equiv (1, N\mathtt{no}/\lambda, 0)$$

- $\mathcal{E} = \Gamma - \{\mathtt{yes}, \mathtt{no}\}$
- $i_{in} = 2$, is the input cell
- $i_0 = env$, is the output cell

An overview of the computation and more details of the design can be read in [2].

4 Sevilla Carpets

Sevilla Carpets were presented in [1] 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., [6, 7] or [11]). The Szilard language is usually defined for grammars in the Chomsky hierarchy where only a single rule is used in each derivation step, so a derivation can be represented as the string of the labels of the rules used in the derivation (the labeling is supposed to be one-to-one). Sevilla Carpets are a Szilard-way to describe a computation in a P system. The main difference is that a multiset of rules can be used in each evolution step of a P system. In [1] a bidimensional writing is proposed to describe a computation of a P system. 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. Depending on the amount of information given to describe the evolution, 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.

They also propose two parameters (*weight* and *surface*) to study Sevilla Carpets. In [4] two new parameters (*height* and *average weight*) were proposed.

4.1 Parameters for the Descriptive Complexity

Many times we will not be interested only in the number of cellular steps of the computation, but also in other type of resources required to perform the computation. Specially if we want to implement *in silico* a P system, we need to be

careful with the number of times that a rule is applied, maybe with the number of membranes and/or the number of objects present in a given configuration.

In order to describe the complexity of the computation, the following parameters are proposed:

- **Weight:** It is defined in [1] as the sum of all the elements in the carpet, i.e., as the total number of applications of rules along the computation. The application of a rule has a cost and the weight measures the total 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. It can be considered as the *potential size* 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:** It is the maximum number of applications of any rule in a step along 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.

5 Comparing the Solutions

In [4], two uniform families of P systems with active membranes solving the Subset Sum were presented. In both solutions, a tuple $(n, (w_1, \ldots, w_n), k)$ is used to represent an instance of the problem, where n stands for the size of $A = \{a_1, \ldots, a_n\}$, $w_i = w(a_i)$, and k is the constant given as input for the problem. Both solutions are based on a brute force algorithm implemented in the framework of P systems with active membranes. The idea of the design can be divided into several stages:

- *Generation stage*: for every subset of A, a membrane is generated via membrane division.
- *Weight calculation stage*: in each membrane the weight of the associated subset is calculated. This stage will take place in parallel with the previous one.
- *Checking stage*: in each membrane it is checked whether or not the weight of its associated subset is exactly k. This stage cannot start in a membrane before the previous ones are over in that membrane.
- *Output stage*: when the previous stage has been completed in all membranes, the system sends out the answer to the environment.

The first family can be found in [9]. Let us recall that the instance $u = (n, (w_1, \ldots, w_n), k)$ is processed by the P system $\Pi_1(\langle n, k\rangle)$ with input the multiset $x_1^{w_1} x_2^{w_2} \ldots x_n^{w_n}$.

This design depends on the two constants that are given as input in the problem: n and k. It consists on $5n + 5k + 18$ evolution rules, and if an appropriate input multiset is introduced inside membrane e before starting the computation, the system will stop and output an answer in $2n + 2k + 6$ steps (if the answer is No) or in $2n + 2k + 5$ steps (if the answer is Yes).

The second family is inspired in the previous one. Some modifications were made following the design presented in [3]. In this solution the instance $u = (n, (w_1, \ldots, w_n), k)$ is processed by the P system $\Pi_2(n)$ with input the multiset $x_1^{w_1} x_2^{w_2} \ldots x_n^{w_n}$.

The above design depends only on one of the constants that are given as input in the problem: n. It is quite similar to the previous one, the difference lies in the checking stage and the answer stage. In this case we avoid the use of counters that require knowing the constant k.

The number of evolution rules is $5n + 41$, and the number of steps of the computation depends on the concrete instance that we need to solve, but it is linearly bounded.

We compare these solutions with the solution described in Section 3. As pointed out in [2], the number of rules for a set $A = \{a_1 \ldots, a_n\}$ of size n is ...and the number of steps is ... if the answer is `Yes` and ... if the answer is `No`.

5.1 Descriptive Complexity

We present some detailed statistics about the previous designs, trying to compare them on a more general basis than just looking the number of steps that the computation performs. Following this scheme, we present the Sevilla Carpets associated with the computations of the three different solutions to the Subset Sum problem working on the same instance: $u = (5, (3, 5, 3, 2, 5), 9)$. That is, $n = 5$, $k = 9$ and the list of weights is $w_1 = 3, w_2 = 5, w_3 = 3, w_4 = 2, w_5 = 5$.

The P system $\Pi_1(\langle 5, 9\rangle)$ has 88 evolution rules, and all of them are applied with the exception of the rules: $[q_{19}]_e^- \to [\ ]_e^0 Yes$, $[q_3]_e^- \to [\ ]_e^- \#$, $[q_9]_e^- \to [\ ]_e^- \#$ and $[Yes]_s^- \to [\ ]_s^0 Yes$. The P system $\Pi_1(5, 9)$ stops at step 33 and sends an object No to the environment.

The weight of the Sevilla Carpet (the total number of rule applications along the computation) is 2179. The height of the Sevilla Carpet (the maximal number of times that a rule is applied in one evolution step) is 82 and it is reached at Step 9. The surface of the Sevilla Carpet is 2904. The average weight of the Sevilla Carpet is 0.749656

The P system $\Pi_2(5)$ has 65 evolution rules, and all of them are applied with the exception of the rules: $[q_3]_e^0 \to [\ ]_e^+ Yes$ and $[Yes]_s^- \to [\ ]_s^0 Yes$. The P system $\Pi_2(5)$ stops at step 38 and sends an object No to the environment.

The weight of the Sevilla Carpet is 3368. The height of the Sevilla Carpet is 108 and it is reached at step 10. The surface of the Sevilla Carpet is 2470. The average weight of the Sevilla Carpet is 1.36275

Finally, the solution with tissue-like P systems with cell division has 88 rules and 84 of them are applied in this computation. The P system stops at step 24.

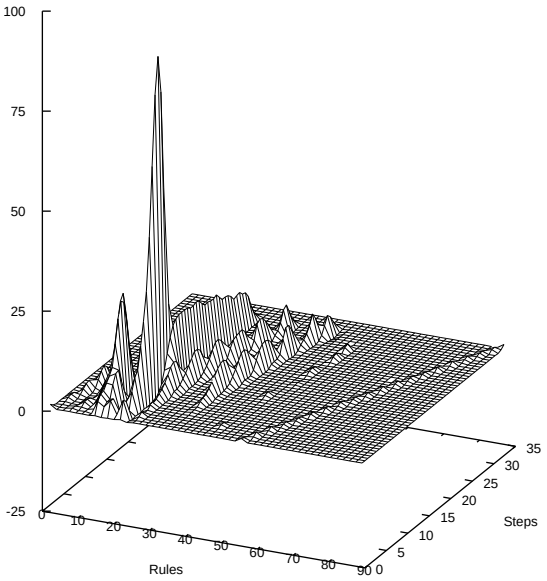

Fig. 1. Sevilla Carpet for solution 1

The surface of the Sevilla Carpet is 2112 and its weight is 2405. The height is 128 and it is reached at steps 10, 12, 13, 14 and 15. The average weight of the Sevilla Carpet is 1.13873.

The following table shows the parameters of both solutions:

	Sol. 1	Sol.2	Sol. 3
Rules	88	65	88
Steps	33	38	24
Surface	2904	2470	2112
Weight	2179	3368	2405
Height	82	108	128
Average Weight	0.749656	1.36275	1.13873

If we consider the number of steps as a complexity measure to compare the designs, then we conclude that the third solution is *better* than the other ones, since it needs less steps.

Moreover, concerning the weight of the Sevilla Carpet, solution 1 is *better* than the other ones, because it uses less resources during the computation.

Nonetheless, the key point of a design of a solution in Membrane Computing is the use of the massive parallelism. As pointed out in [5],

> *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*

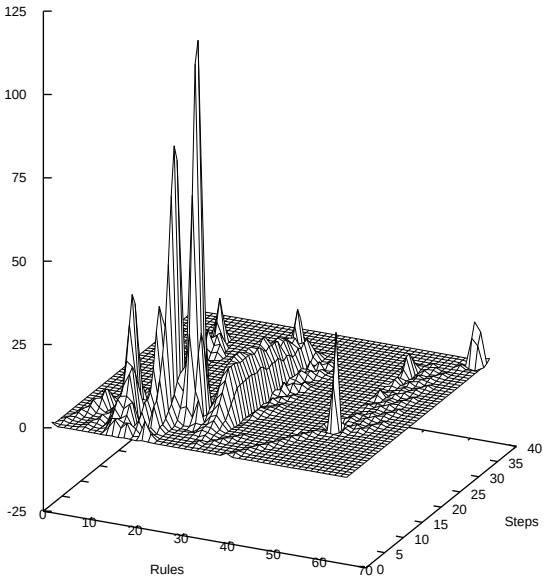

Fig. 2. Sevilla Carpet for solution 2

> *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.*

In this line, the fact that the average weight of solution 2 is larger than the average weight of the other solutions can be interpreted saying that the second design makes a better use of the parallelism in P systems.

6 Conclusions and Future Work

It is important to remark that these are not asymptotical comparisons, as we focus only on the data corresponding to one instance. Indeed, due to the exponential number of membranes created during the generation stage, we believe that considering another instance with a greater size will stress the differences between the design based only on n and the other one, based on n and k. The bound on the size of the instances that can be studied is imposed by the necessity to use a P systems simulator to obtain the detailed description of the computation: number of rules, number of cellular steps, and number of times that the rules are applied in each step.

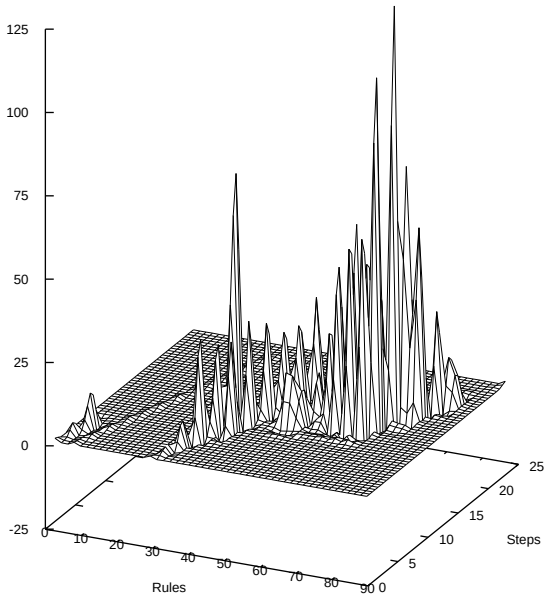

Fig. 3. Sevilla Carpet for solution 3

Acknowledgements

The authors acknowledge the support of the project TIN2006-13425 of the Ministerio de Educación y Ciencia of Spain, cofinanced by FEDER funds, and the support of the Project of Excellence with *Investigador de Reconocida Valía* of the Junta de Andalucía, grant P08-TIC-04200.

References

1. G. Ciobanu, Gh. Păun, Gh. Ştefănescu: 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.
2. D. Díaz-Pernil, M.A. Gutiérrez-Naranjo, M.J. Pérez-Jiménez, A. Riscos-Núñez: Solving Subset Sum in Linear Time by Using Tissue P Systems with Cell Division. In *Bio-inspired Modeling of cognitive tasks*, J. Mira and J.R. Alvarez, eds., LNCS 4527, Springer, 2007, 170–179.
3. M.A. Gutiérrez-Naranjo, M.J. Pérez-Jiménez, A. Riscos-Núñez: Towards a programming language in cellular computing. *Proceedings of the 11th Workshop on Logic, Language, Information and Computation* (WoLLIC'2004), July 19-22, 2004, 1-16 Campus de Univ. Paris 12, Paris, France. A preliminary version in Gh. Păun, A. Riscos, A. Romero, F. Sancho, eds., *Proceedings of the Second Brainstorming Week on Membrane Computing*, Report RGNC 01/04, 2004, 247–257.

4. 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, 2005, 320–330.
5. 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.
6. E. Mäkinen: A Bibliography on Szilard Languages, Dept. of Computer and Information Sciences, University of Tampere, `http://www.cs.uta.fi/reports/pdf/Szilard.pdf`.
7. A. Mateescu, A. Salomaa: Aspects of Classical Language Theory. In G. Rozenberg, A. Salomaa, eds., *Handbook of Formal Languages* (vol. 1), Springer, Berlin, 1997.
8. Gh. Păun, M.J. Pérez-Jiménez, A. Riscos-Núñez: Tissue P System with cell division. In Gh. Păun, A. Riscos-Núñez, A. Romero-Jiménez, F. Sancho-Caparrini, eds., *Second Brainstorming Week on Membrane Computing*, Sevilla, Report RGNC 01/2004, 2004, 380–386.
9. M.J. Pérez-Jiménez, A. Riscos-Núñez: Solving the Subset Sum Problem by Active Membranes. *New Generation Computing*, 23, 4 (2005), 367–384.
10. M.J. Pérez-Jiménez, A. Romero-Jiménez, F. Sancho-Caparrini: A polynomial complexity class in P systems using membrane division. In E. Csuhaj-Varjú, C. Kintala, D. Wotschke, Gy. Vaszyl, eds., *Proceedings of the 5th Workshop on Descriptional Complexity of Formal Systems, DCFS 2003*, 2003, 284–294.
11. A. Salomaa: *Formal Languages*. Academic Press, New York, 1973.