On Descriptive Complexity of P Systems

Miguel A. Gutiérrez-Naranjo, Mario J. Pérez-Jiménez, and Agustin Riscos-Núñez

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

Abstract. In this paper we address the problem of describing the complexity of the evolution of a P system. This issue is is specially hard in the case of P systems with active membranes, where the number of steps of a computation is not sufficient to evaluate the complexity. Sevilla carpets were introduced in [1], and they describe the space-time complexity of P systems. Based on them, we define some new parameters which can be used to compare evolutions of P systems. To illustrate this, we also include two different cellular solutions to the Subset Sum problem and compare them via these new parameters.

1 Introduction

The evolution of a P system is a complex process where (possibly) a large number of symbol-objects, membranes and rules are involved. In the case of P systems with active membranes, the problem of describing the complexity of the computational process becomes specially hard. In this case, elementary membranes can divide into two new membranes and, due to the parallelism intrinsic to P systems, an exponential number of membranes can be obtained in polynomial time. This feature makes P systems with active membranes a powerful tool to attack NP-complete problems and, indeed, several efficient solutions to this type of problems have been presented (see, e.g., [4, 9, 10, 11] or [12]). These solutions are proposed in the framework of *recognizer P systems with external output*, and they present significant similarities among them. The basic idea in these designs is the creation of an exponential number of membranes (*workspace*) in polynomial time and the use of each membrane as an independent computational device. All membranes evolve *in parallel* and the computation has a polynomial cost in time. The process ends with a final stage (with polynomial cost) that checks the answers of these devices and sends an output to the environment.

The complexity in *time* (the number of 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. Ciobanu, Păun and Ştefănescu presented in [1] a new way to describe the complexity of a computation in a P system. The so-called *Sevilla carpet* is an extension of the notion of Szilard language from grammars to the case when several rules are used at the same time.

In this paper we make use of Sevilla carpets to describe the computations of P systems that solve the Subset Sum problem. Two families of recognizer P systems have been designed that need a polynomial time to send an output to the environment. We present their corresponding Sevilla carpets in order to compare them, and then some ideas to improve the design of P systems for solving other new problems are proposed.

The paper is organized as follows. In Section 2 we first give some preliminary notions about *recognizer P systems* and a polynomial complexity class on P systems is defined. Section 3 presents the Sevilla carpets and some new parameters related with them are introduced in Section 4. Finally, we use these parameters to compare two solutions of the Subset Sum problem.

2 Preliminaries

Roughly speaking, a P system consists of a cell-like membrane structure, in the compartments of which one places multisets of objects which evolve according to given rules in a synchronous non-deterministic maximally parallel manner.

Definition 1. *A* P system with input *is a tuple (Π, Σ, i_Π), where: Π is a P system, with working alphabet Γ, with p membranes labelled by $1, \dots, p$, and initial multisets $\mathcal{M}_1, \dots, \mathcal{M}_p$ associated with them; Σ is an (input) alphabet strictly contained in Γ; the initial multisets are over $\Gamma - \Sigma$; finally, i_Π is the label of a distinguished (input) membrane.*

The computations of a P system with input a multiset m over Σ, are defined in a natural way. The only novelty is that the initial configuration must be the initial configuration of the system to which the input multiset m is added to the multiset from region i_Π.

Definition 2. *Let (Π, Σ, i_Π) be a P system with input. Let Γ be the working alphabet of Π, μ the membrane structure and $\mathcal{M}_1, \dots, \mathcal{M}_p$ the initial multisets of Π. Let m be a multiset over Σ. The* initial configuration of (Π, Σ, i_Π) with input m *is $(\mu, \mathcal{M}_1, \dots, \mathcal{M}_{i_\Pi} \cup m, \dots, \mathcal{M}_p)$.*

In the case of P systems with input and with external output, the concept of computation is as standard in membrane computing, with a minor difference which will be explained below. We consider that it is not possible to observe the internal processes inside the P system and we can only know if the computation has halted via some distinguished objects sent out of the skin. We can formalize these ideas in the following way.

2.1 Recognizer P Systems

Recall that a decision problem X is a pair (I_X, θ_X) such that 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 solve decision problems we need P systems with input such that all halting computations starting from an initial configuration with a given input multiset (encoding an instance of the problem) produce the same output. The systems of this type will be called *recognizer* P systems.

Definition 3. *A* recognizer P system *is a P system with input, (Π, Σ, i_Π), and with external output such that:*

1. *The working alphabet contains two distinguished elements YES, NO.*
2. *All computations halt.*
3. *If $\mathcal{C}$ is a computation of Π, then either object YES or object NO (but not both) must have been released into the environment, and only in the last step of the computation. We say that $\mathcal{C}$ is an accepting computation (respectively, rejecting computation) if the object YES (respectively, NO) appears in the environment associated with the corresponding halting configuration of $\mathcal{C}$.*

The above definitions are stated in a general way, but in this paper P systems with active membranes will be used. We refer to [8] (see chapter 7) for a detailed definition of evolution rules, transition steps, configurations and computations in this model.

We denote by $\mathcal{AM}$ the class of all recognizer P systems with active membranes.

2.2 The Computational Complexity Class $\mathbf{PMC}_\mathcal{F}$

The first results about "solvability" of **NP**–complete problems in polynomial time (even linear) by cellular computing systems with membranes were obtained using variants of P systems that lack an input membrane (see e.g. [7] or [14]). Thus, the constructive proofs of such results need to design one system for each instance of the problem.

If we wanted to perform such a solution of some decision problem in a laboratory, we will find a drawback on this approach: a system constructed to solve a concrete instance is useless when trying to solve another instance. This shortcoming can be easily overtaken if we consider a P system with input. Then, a system could solve different instances of the problem, provided that the corresponding input multisets are introduced in the input membrane.

Instead of looking for a single system that solves a problem, we prefer designing a *family* of P systems such that each element decides all the instances of "equivalent size", in certain sense.

Definition 4. *Let $\mathcal{F}$ be a class of recognizer P systems. We say that a decision problem $X = (I_X, \theta_X)$ is* solvable *in polynomial time by a family $\mathbf{\Pi} = (\Pi(n))_{n \in \mathbf{N}^+}$ of type $\mathcal{F}$, and we denote this by $X \in \mathbf{PMC}_\mathcal{F}$, if the following is true:*

– *The family $\mathbf{\Pi}$ is polynomially uniform by Turing machines; that is, there exists a deterministic Turing machine constructing $\Pi(n)$ from $n \in \mathbf{N}^+$ in polynomial time.*

– *There exists a pair (g, h) of polynomial-time computable functions $g : L \rightarrow \bigcup_{n \in \mathbf{N}^+} I_{\Pi(n)}$ and $h : L \rightarrow \mathbf{N}^+$ such that for every $u \in L$ we have $g(u) \in I_{\Pi(h(u))}$, and*
  - *The family $\mathbf{\Pi}$ is polynomially bounded with regard to (g, h); that is, there exists a polynomial function p, such that for each $u \in I_X$ every computation of $\Pi(h(u))$ with input $g(u)$ is halting and, moreover, it performs at most $p(|u|)$ steps.*
  - *The family $\mathbf{\Pi}$ is sound, with regard to (X, g, h); that is, for each $u \in I_X$ it is verified that if there exists an accepting computation of $\Pi(h(u))$ with input $g(u)$, then $\theta_X(u) = 1$.*
  - *The family $\mathbf{\Pi}$ is complete, with regard to (X, g, h); that is, for each $u \in I_X$ it is verified that if $\theta_X(u) = 1$, then every computation of $\Pi(h(u))$ with input $g(u)$ is an accepting one.*

In the above definition we have imposed every P system $\Pi(n)$ to be *confluent*, in the following sense: every computation with the *same* input produces the *same* output. From the dfinition, one can easily prove that the class $\mathbf{PMC}_{\mathcal{F}}$ is closed under polynomial–time reduction and complement.

3 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., [5, 6] or [13]). 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 labelling 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 now 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 this paper we popose two new parameters (*height* and *average weight*) that will be described in the next section.

4 Parameters for the Descriptive Complexity

Many times we are not interested only in the number of cellular steps of the computation, but also in other types of resources required to perform the computation. Especially 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:** This 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 in 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:** This 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, and gives an indication on how the P system exploits its massive parallelism.

5 Comparing Two Solutions to the Subset Sum Problem

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

We will use a tuple $(n, (w_1, \ldots, w_n), k)$ 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.

We propose here two solutions to this problem based on a brute force algorithm implemented in the framework of P systems with active membranes. The idea of the design is better understood if we divide the solution to the problem 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.

First Design

Next we present a family of recognizer P systems solving Subset Sum, according to Definition 4. This family can be found in [9].

First, we consider a polynomial–time computable and bijective function from $\mathbf{N}^2$ onto $\mathbf{N}$ (for example, $\langle x, y\rangle = ((x+y)(x+y+1)/2)+y$). For each $(n,k) \in \mathbf{N}^2$ we consider the P system $(\Pi_1(\langle n,k\rangle), \Sigma(n,k), i(n,k))$, where the input alphabet is $\Sigma(n,k) = \{x_1,\dots,x_n\}$, the input membrane is $i(n,k) = e$ and $\Pi_1(\langle n,k\rangle) = (\Gamma(n,k), \{e,s\}, \mu, \mathcal{M}_e, \mathcal{M}_s, R)$ is defined as follows:

• Alphabet: $\Gamma(n,k) = \Sigma(n,k) \ \cup \ \{\bar{a}_0, \bar{a}, a_0, a, d_+, e_0, \dots, e_n, q, q_0, \dots, q_{2k+1}, z_0, \dots, z_{2n+2k+2}, Yes, \overline{no}, No, \#\}$.
• membrane structure: $\mu = [\ [\ ]_e\ ]_s$.
• Initial multisets: $\mathcal{M}_s = z_0$; $\mathcal{M}_e = e_0\bar{a}^k$.
• The set R of evolution rules consists of the following rules:

(a) $[e_i]_e^0 \rightarrow [q]_e^-[e_i]_e^+$, for $i = 0,\dots,n$.
$[e_i]_e^+ \rightarrow [e_{i+1}]_e^0[e_{i+1}]_e^+$, for $i = 0,\dots,n-1$.
(b) $[x_0 \rightarrow \bar{a}_0]_e^0$; $\quad [x_0 \rightarrow \lambda]_e^+$; $\quad [x_i \rightarrow x_{i-1}]_e^+$, for $i = 1,\dots,n$.
(c) $[q \rightarrow q_0]_e^-$; $\quad [\bar{a}_0 \rightarrow a_0]_e^-$; $\quad [\bar{a} \rightarrow a]_e^-$.
(d) $[a_0]_e^- \rightarrow [\ ]_e^0\#$; $\quad [a]_e^0 \rightarrow [\ ]_e^-\#$.
(e) $[q_{2j} \rightarrow q_{2j+1}]_e^-$, for $j = 0,\dots,k$.
$[q_{2j+1} \rightarrow q_{2j+2}]_e^0$, for $j = 0,\dots,k-1$.
(f) $[q_{2k+1}]_e^- \rightarrow [\ ]_e^0 Yes$; $\quad [q_{2k+1}]_e^0 \rightarrow [\ ]_e^0\#$.
$[q_{2j+1}]_e^- \rightarrow [\ ]_e^-\#$, for $j = 0,\dots,k-1$.
(g) $[z_i \rightarrow z_{i+1}]_s^0$, for $i = 0,\dots,2n+2k+1$; $\quad [z_{2n+2k+2} \rightarrow d_+\overline{no}]_s^0$.
(h) $[d_+]_s^0 \rightarrow [\ ]_s^+ d_+$; $[\overline{no} \rightarrow No]_s^+$; $[Yes]_s^+ \rightarrow [\ ]_s^0 Yes$; $[No]_s^+ \rightarrow [\ ]_s^0 No$.

Let us recall that the instance $u = (n, (w_1,\dots,w_n), k)$ is processed by the P system $\Pi_1(\langle n,k\rangle)$ with input the multiset $x_1^{w_1}x_2^{w_2}\dots x_n^{w_n}$.

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 apropriate 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).

According to Definition 4 and using the above family of P systems, we can prove that, *Subset Sum* $\in \mathbf{PMC}_{\mathcal{AM}}$ (see [9], for details).

Second Design

Next we present a new family of recognizer P systems solving Subset Sum, inspired in the previous one. Some modifications are made following the design presented in [3].

For each $n \in \mathbf{N}$ we consider the P system $(\Pi_2(n), \Sigma(n), i(n))$, where the input alphabet is $\Sigma(n) = \{x_1, \dots, x_n\}$, the input membrane is $i(n) = e$ and $\Pi_2(n) = (\Gamma(n), \{e, r, s\}, \mu, \mathcal{M}_e, \mathcal{M}_r, \mathcal{M}_s, R)$ is defined as follows:

- Alphabet: $\Gamma(n) = \Sigma(n) \cup \{\bar{a}_0, \bar{a}, a_0, a, c, d_0, d_1, d_2, e_0, \dots, e_n, g, \bar{g}, \hat{g}, h_0, h_1, q, q_0, q_1, q_2, q_3, Yes, No, \overline{no}, z_0, \dots, z_{2n+1}, \#\}$.
- Membrane structure: $\mu = [\ [\ ]_e\ ]_s$.
- Initial multisets: $\mathcal{M}_s = z_0$; $\mathcal{M}_e = e_0 g \bar{a}^k$; $\mathcal{M}_r = h_0 b$.
- The set R of evolution rules consists of the following rules:

(a) $[e_i]_e^0 \rightarrow [q]_e^- [e_i]_e^+$, for $i = 0, \dots, n$.
$[e_i]_e^+ \rightarrow [e_{i+1}]_e^0 [e_{i+1}]_e^+$, for $i = 0, \dots, n-1$.

(b) $[x_0 \rightarrow \bar{a}_0]_e^0$; $[x_0 \rightarrow \lambda]_e^+$; $[x_i \rightarrow x_{i-1}]_e^+$, for $i = 1, \dots, n$.

(c) $[q \rightarrow q_0]_e^-$; $[\bar{a}_0 \rightarrow a_0]_e^-$; $[\bar{a} \rightarrow a]_e^-$.
$[g]_e^- \rightarrow [\ ]_e^- \bar{g}$.

$[e_n]_e^+ \rightarrow \#$.
$[\bar{a}_0 \rightarrow \lambda]_s^0$; $[\bar{a} \rightarrow \lambda]_s^0$; $[g \rightarrow \lambda]_s^0$.
$[a \rightarrow \lambda]_e^+$; $[a_0 \rightarrow \lambda]_e^+$.

(d) $[a_0]_e^- \rightarrow [\ ]_e^0 \#$; $[a]_e^0 \rightarrow [\ ]_e^- \#$.

(e) $[q_0 \rightarrow q_1]_e^-$; $[q_1 \rightarrow q_0]_e^0$.
$[q_0]_e^0 \rightarrow [\ ]_e^+ \overline{no}$.
$[q_1 \rightarrow q_2 c]_e^-$; $[q_2 \rightarrow q_3]_e^0$; $[c]_e^- \rightarrow [\ ]_e^0 k$.

(f) $[q_3]_e^0 \rightarrow [\ ]_e^+ Yes$; $[q_3]_e^- \rightarrow [\ ]_e^+ \overline{no}$.

(g) $[z_i \rightarrow z_{i+1}]_s^0$, for $i = 0, \dots, 2n$; $[z_{2n+1} \rightarrow d_0 d_1]_s^0$.
$d_0 [\ ]_r^0 \rightarrow [d_0]_r^-$; $[d_1]_s^0 \rightarrow [\ ]_s^+ d_1$.

(det) $[h_0 \rightarrow h_1]_r^-$, $[h_1 \rightarrow h_0]_r^+$,
$[b]_r^- \rightarrow [\ ]_r^+ b$, $\hat{g}[\ ]_r^+ \rightarrow [\hat{g}]_r^-$,
$b[\ ]_r^- \rightarrow [b]_r^+$, $[\hat{g}]_r^+ \rightarrow [\ ]_r^- \hat{g}$,
$[h_0]_r^+ \rightarrow [\ ]_r^+ d_2$, $[d_2]_s^+ \rightarrow [\ ]_s^- d_2$.

(h) $[\overline{no} \rightarrow No]_s^-$; $[Yes]_s^- \rightarrow [\ ]_s^0 Yes$; $[No]_s^- \rightarrow [\ ]_s^0 No$.

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.

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 two 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 input multiset is then: $x_1^3 x_2^5 x_3^3 x_4^2 x_5^5$.

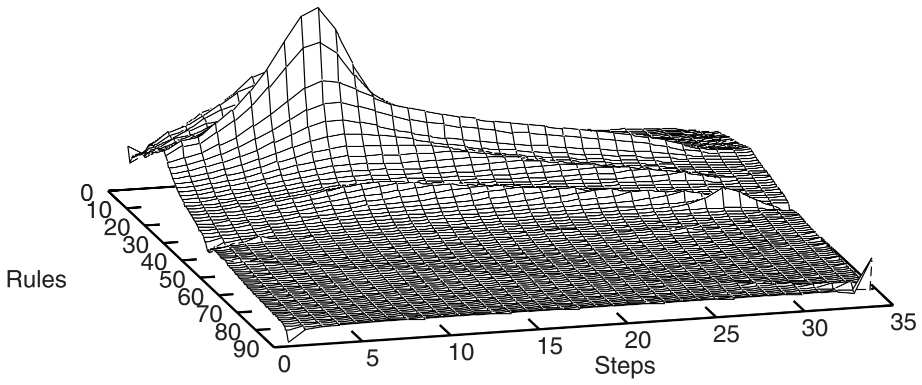

Fig. 1. Sevilla carpet for solution 1

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: $[q_{19}]_e^- \rightarrow [\ ]_e^0 Yes$, $[q_3]_e^- \rightarrow [\ ]_e^- \#$, $[q_9]_e^- \rightarrow [\ ]_e^- \#$ and $[Yes]_s^- \rightarrow [\ ]_s^0 Yes$. 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, and its height (the maximal number of times that a rule is applied in one evolution step) is 82 and it is reached at Step 9 by the rule $[\bar{a}_0 \rightarrow a_0]_e^-$. The surface of the Sevilla carpet is 2904, and its average weight is 0.749656

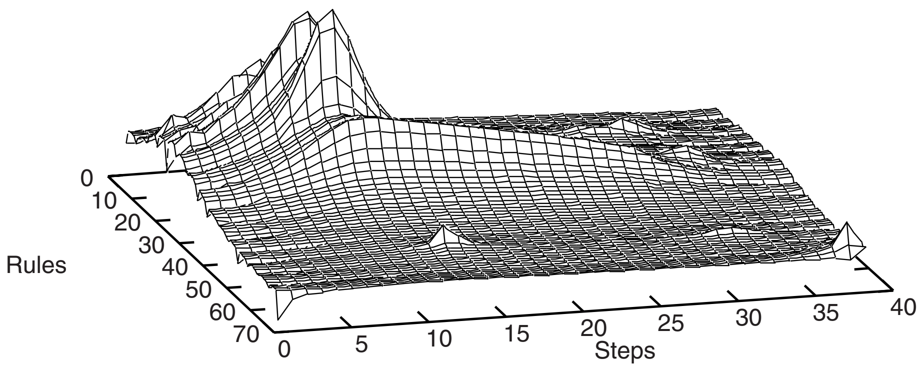

Fig. 2. Sevilla carpet for solution 2

The P system $\Pi_2(5)$ has 65 evolution rules, and all of them are applied with the exception of the rules: $[q_3]_e^0 \rightarrow [\ ]_e^+ Yes$ and $[Yes]_s^- \rightarrow [\ ]_s^0 Yes$. 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, and its height is 108, this height is reached at Step 10 by the rule $[\bar{a}_0 \rightarrow \lambda]_s^0$. The surface of the Sevilla carpet is 2470, and its average weight is 1.36275

The following table shows the parameters of both solutions:

	Solution 1	Solution 2
Rules	88	65
Steps	33	38
Surface	2904	2470
Weight	2179	3368
Height	82	108
Average Weight	0.749656	1.36275

If we consider the number of steps as a complexity measure to compare both designs, then we conclude that the first solution is *better* than the second one (although not asymptotically), since it needs less steps.

Moreover, concerning the weight of the Sevilla carpet, solution 1 is again *better* than solution 2, because it uses less resources during the computation. However, the fact that the average weight of solution 2 is larger than the average weight of solution 1 can be interpreted by saying that the second design makes a better use of the parallelism in P systems (the computation is more *intense*).

We would like 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 both n and k. The bound on the size of the intances 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. The simulator we are using (presented in [2]) is written in Prolog, and it runs on a sequential conventional computer.

6 Final Comments and Future Work

This paper illustrates the necessity of a deeper study of parameters which describe the complexity of P systems as computational devices. In order to analyze this complexity we use the Sevilla carpets. We also define two new parameters which provide us with a more detailed description of the evolution of a P system.

A more detailed study of the differences between the computations of the two solutions discussed here is to be done, in order to extract some conclusions about the usefullness and/or the interest of these new complexity parameters that can be used to evaluate the design of cellular solutions to problems.

In the example illustrated in the previous section, the second design solves the same instance in 5 additional cellular steps, but the number of rules is much lower. Can we decrease more the number of rules and keep a linear bound on the number of steps? Is it worth it?

In the near future, we plan to carry out descriptive complexity studies of other variants of P systems, maybe giving rise to new significant parameters. We would like also to improve the graphical treatment of Sevilla carpets, designing a software able to go directly from the description of the computation provided by our P systems simulator to the picture of the carpet.

Acknowledgement

The support for this research through the project TIC2002-04220-C03-01 of the Ministerio de Ciencia y Tecnología of Spain, cofinanced by FEDER funds, is gratefully acknowledged.

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. C. Cordón-Franco, M.A. Gutiérrez-Naranjo, M.J. Pérez-Jiménez, F. Sancho-Caparrini, A Prolog simulator for deterministic P systems with active membranes. *New Generation Computing*, **22** (4), 2004, 349–363.
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, A fast P system for finding a balanced 2-partition. *Soft Computing*. Springer. To appear.
5. 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`
6. A. Mateescu, A. Salomaa, Aspects of classical language theory. In G. Rozenberg and A. Salomaa (eds.), *Handbook of Formal Languages* (vol. 1), Springer-Verlag, Berlin Heidelberg, 1997.
7. Gh. Păun, P systems with active membranes: Attacking NP complete problems. *Journal of Automata, Languages and Combinatorics*, **6**(1), 2001, 75–90.
8. Gh. Păun, *Membrane Computing. An Introduction*. Springer-Verlag, Berlin, 2002.
9. M.J. Pérez-Jiménez, A. Riscos-Núñez, Solving the Subset Sum problem by active membranes. *New Generation Computing*, to appear.
10. M.J. Pérez-Jiménez, A. Riscos-Núñez, 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.
11. 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, and Gy. Vaszyl (eds.), *Proceedings of the 5th Workshop on Descriptional Complexity of Formal Systems*, Budapest, Hungary, 2003, 284–294.
12. M.J. Pérez-Jiménez, A. Romero-Jiménez, F. Sancho-Caparrini, Solving VALIDITY problem by active membranes with input. 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, 279–290.
13. A. Salomaa, *Formal Languages*. Academic Press, New York, 1973.
14. C. Zandron, *A Model for Molecular Computing: Membrane Systems*. Ph.D. Thesis, Università degli Studi di Milano, 2001.