On the degree of parallelism in membrane systems

Miguel A. Gutiérrez-Naranjo, Mario J. Pérez-Jiménez*, Agustín Riscos-Núñez

Research Group on Natural Computing, Department of Computer Science and Artificial Intelligence, University of Sevilla, Avda. Reina Mercedes s/n, 31012, Sevilla, Spain

Abstract

In the literature, several designs of P systems might be found for performing the same task. The use of different techniques or even different P system models makes it very difficult to compare these designs. In this paper, we introduce a new criterion for such a comparison: the degree of parallelism of a P system. With this aim, we define the *labelled dependency graph* associated with a P system, and we use this new concept for proving some results concerning the maximum number of applications of rules in a single step through the computation of a P system.

Keywords: Membrane Computing; P systems; Dependency graph; Degree of parallelism

1. Introduction

In recent years, an extensive literature on Membrane Computing has been produced, studying multiple approaches. We can consider the following rough classification:

- *Generative task*: from a given initial configuration several distinct computations may be developed (in a non-deterministic manner) and they may produce different outputs. We can consider that the system *generates* the set of all the *outputs* of all the computations (and this set can be interpreted as the *language generated* by the system).
- *Computing task*: if we can encode any natural number, n, in the initial configuration of a given P system and we consider the cardinality of the output multiset as the result of the computation, then we can interpret that the system "computes" a function over $\mathbb{N}$.
- *Decidability task*: another option is to consider that the output alphabet consists of two special objects, yes and no, in such a way that these are the only objects that determine the answer, irrespectively of the occurrence of other possible objects from the working alphabet in the output membrane.

Several designs of P systems might be used for performing the same task. The use of different techniques or even different P system models makes it very difficult to compare these designs. Furthermore, the intrinsic non-determinism of P systems usually yields computational trees of a very large size, which makes the comparison task specially hard.

* Corresponding author. Tel.: +34 954557952; fax: +34 954557952.
E-mail address: marper@us.es (M.J. Pérez-Jiménez).

A first attempt[1] to give an appropriate description of the complexity of the evolution of a P system was given by G. Ciobanu, Gh. Păun and Gh. Ştefănescu in [2]. In this paper, a new tool for the descriptive complexity of P systems, called a *Sevilla carpet*, was presented. Roughly speaking, a Sevilla carpet is a table with time and an explicit enumeration of the rules of the P system on its axes. For each pair step-rule, a piece of information is given. Additional parameters for studying Sevilla carpets were introduced in [7] and a multidimensional generalization of Sevilla carpets was presented in [9].

Sevilla carpets and their associated parameters provide very useful information to describe the evolution of a P system, but *only* for one computation. Therefore, if we consider two deterministic P systems which perform the same task, then the use of Sevilla carpets, together with the associated parameters, gives enough information to establish a comparison. It is clear though that this is not the general case.

In general we have non-deterministic P systems which may have infinite computations. We wonder how to "compare" two different P system designs which perform the same task, possibly implemented in different models.

One of the bases of the power of P systems as computational devices is the maximal parallelism in the use of rules. This maximal parallelism is twofold: all membranes process data in parallel and inside each membrane as many objects as possible evolve. Complementing this feature with the ability of producing new membranes through the computation is the basis of the design of families of P systems which solve **NP**-complete problems in polynomial time (see, e.g., [8,14], and also [15] and references therein).

In this paper we focus on the parallelism, viewed as a tool to compare the design of P systems which perform the same task. Intuitively, 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 evolves 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 number 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 than the first one.

In the general case the comparison is not so easy and, in most cases, it would be useful to have a numerical function which rates or estimates how good is the use of the parallelism in a P system. The quest for such a function is hard, since P systems are intrinsically non-deterministic and two computations can be quite different even in confluent P systems. In this paper we propose as a parameter the maximum number of simultaneous applications of rules in a step of *any* computation. This number depends on the initial configuration, the set of rules and the semantics of the model.

The paper is organized as follows: first some preliminaries are given, recalling some concepts related to multisets, introducing the new concept of *injective mapping with respect to a multiset* and fixing some ideas about graphs and P systems. In Section 3 we give an estimation of the use of parallelism in a P system from a given configuration *in one step*. In the following section we extend our study to the general case and provide a new parameter, $\beta(\Pi, C)$, which is an upper bound on the maximum number of simultaneous applications of rules in one step in any computation of Π with the initial configuration C. Finally, some conclusions and lines for future work are given.

2. Preliminaries

In this section we recall some concepts which will be used through the paper. First of all, we recall some basic ideas on *multisets* and introduce the new concept of *injective mapping with respect to a multiset*. Next we present the P system model we shall work with in this paper and adopt some conventions for notation.

2.1. Multisets

Multisets are the basic data structure in P systems. Their use is inspired by the chemical compounds inside the organelles of living cells. First of all, we recall the definition.[2]

[1] Recently two new parameters have been introduced in [4] in order to describe the complexity of P systems. They are related to the graph of reachable configurations of a given P system, namely the outdegree as a measure of the degree of non-determinism, and the indegree as a measure of the degree of confluence.

[2] A detailed presentation of multisets can be found, for example, in [16].

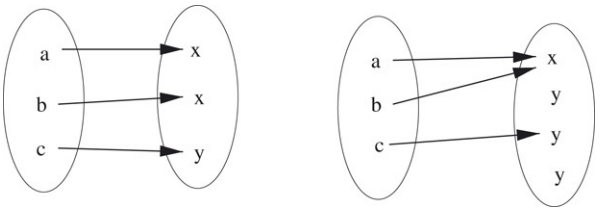

Fig. 1. Mappings on multisets.

Let D be a set. A *multiset* over D is a pair $\langle D, f\rangle$ where $f : D \to \mathbb{N}$ is a mapping. If $\mathcal{D} = \langle D, f\rangle$ is a multiset, its *support*, *supp*($\mathcal{D}$), is defined as $supp(\mathcal{D}) = \{x \in D \mid f(x) > 0\}$ and its cardinality, denoted by $\#\mathcal{D}$, is defined as

$$\#\mathcal{D} = \sum_{a \in D} f(a).$$

We shall use the standard notation $\{x^{f(x)} \mid x \in D\}$ for the multiset $\langle D, f\rangle$.
Suppose that $\mathcal{D}_1 = \langle D, f\rangle$ and $\mathcal{D}_2 = \langle D, g\rangle$ are two multisets over the set D.

- **(Sub-multisets)** If for all $a \in D$ we have $f(a) \leq g(a)$, then we say that $\langle D, f\rangle$ is a sub-multiset of $\langle D, g\rangle$.
- **(Union of multisets)** The union of $\mathcal{D}_1$ and $\mathcal{D}_2$, denoted by $\mathcal{D}_1 \cup \mathcal{D}_2$, is the multiset $\langle D, h\rangle$, where $h(a) = f(a) + g(a)$ for all $a \in D$.

Next we introduce a new definition that will be useful in the following sections. It is a natural generalization of the definition of injective mapping between sets.

Definition 1. Let D_1 and D_2 be sets, $\langle D_2, f\rangle$ a multiset over D_2 and $g : D_1 \to D_2$ a mapping. We say that g is *injective with respect to the multiset* $\langle D_2, f\rangle$ if $\forall y \in D_2$ $(\#\{x \in D_1 \mid g(x) = y\} \leq f(y))$.

This definition can be illustrated with the following example.

Example 2. Let us consider the sets $D_1 = \{a, b, c\}$ and $D_2 = \{x, y\}$ and the multisets over D_2: $\langle D_2, f_1\rangle = \{x^2, y\}$ and $\langle D_2, f_2\rangle = \{x, y^3\}$. Then the mapping $g : D_1 \to D_2$ with $g(a) = x$, $g(b) = x$ and $g(c) = y$ is *injective* w.r.t. $\langle D_2, f_1\rangle$ and is not *injective* w.r.t. $\langle D_2, f_2\rangle$ (see Fig. 1).

Note that this definition extends the usual definition of injective mappings over sets (a set can be seen as a multiset where all elements have multiplicity one).

2.2. Graphs

We briefly fix some concepts which will be used later.

A *directed graph* $\mathcal{G}$ is a pair $\mathcal{G} = (V, E)$ where V is a (finite) set and $E \subseteq V \times V$. The elements of V are called *nodes*, and the elements of E are called *arcs*. In this paper we will consider that given a directed graph $\mathcal{G}$, every pair $\mathcal{G}' = (V', E')$ such that $V' \subseteq V$, $E' \subseteq E \cap (V' \times V')$ is a *subgraph* of $\mathcal{G}$. Given a set of graphs $\{\mathcal{G}_i\}_{i \in I}$ with $\mathcal{G}_i = (V_i, E_i)$, its *union* is the graph

$$\bigcup_{i \in I} \mathcal{G}_i = \left(\bigcup_{i \in I} V_i, \bigcup_{i \in I} E_i \right).$$

We shall consider the *paths* in a directed graph as *subgraphs*, following the next definition.

Definition 3. A *path* of length n from a vertex x to a vertex y in a directed graph $\mathcal{G} = (V, E)$ is a subgraph $\mathcal{G}' = (V', E')$ such that $V' = \{v_0, v_1, \ldots, v_n\}$ with $v_0 = x$, $v_n = y$ and $E' = \{(v_i, v_{i+1}) \mid i = 0, \ldots, n-1\}$. If $x = y$, then we shall say that the path is a *cycle*. The subgraph with a single vertex and no arcs is also considered a path. A path is *simple* if all the vertices in it are distinct.

Finally, we define the *subgraph* generated by a *source* A and a *sink* B.

Definition 4. Given a directed graph $\mathcal{G} = (V, E)$ and two sets of vertices $A, B \subseteq V$ we define the *subgraph* generated by the *source* A and the *sink* B as the subgraph of $\mathcal{G}$ obtained as the union of all the paths in $\mathcal{G}$ from x to y, with $x \in A$ and $y \in B$.

2.3. P systems

We assume that the reader is familiar with the basics of Membrane Computing (see [13]). In this section we briefly define a simple model that we shall be using through this paper.

Recall that, basically, a P system consists of a cell-like membrane structure together with associated multisets of objects and sets of rules expressing how these objects can evolve. The *membrane structure* of a P system is used to enclose *computing cells* in order to make them independent computing units. The objects can pass through membranes and, depending on the model we are dealing with, membranes can be dissolved, divided, or created. Nevertheless, in this paper we shall work in a simplified model without division, dissolution, and creation of membranes. We do not use cooperation nor priority among rules either.

A *configuration* is the instantaneous description of the current membrane structure and the multisets of objects associated with those membranes. In each time unit (a *step*), a transformation of a configuration of the system takes place by applying the rules of each region in a non-deterministic maximally parallel manner. In this way, transitions between two configurations of the system are obtained. A (finite or infinite) sequence of transitions is called a *computation*.

For the sake of simplicity, in this paper we consider a special case of P systems. Specifically, a P system with degree q is a tuple $\Pi = (\Gamma, H, \mu, w_1, \ldots, w_q, R)$ where:

- Γ is a finite alphabet (the working alphabet) whose elements are called objects.
- H is a finite set of labels for membranes.
- μ is a tree-like membrane structure of degree q. Membranes are labelled bijectively by elements from H.
- $w_1, \ldots, w_q$ are multisets over Γ describing the multisets of objects initially placed in membranes from μ.
- R is a finite set of developmental rules. These rules can be of one of the types described below:

  (1) $[a \to v]_l$, where $a \in \Gamma$, $v \in \Gamma^*$, $l \in H$ *(evolution rules)*.
  An object a evolves to a multiset v inside a membrane labelled by l.
  (2) $[a]_l \to b[\,]_l$, where $a, b \in \Gamma$, $l \in H$ *(send-out communication rules)*.
  An object a gets out of a membrane labelled by l, possibly transformed in a new object b.
  (3) $a[\,]_l \to [b]_l$, where $a, b \in \Gamma$, $l \in H$ *(send-in communication rules)*.
  An object a gets into a membrane labelled by l, possibly transformed in a new object b.

We would like to stress that neither dissolution rules nor membrane division nor membrane creation are permitted in the model we are considering. Hence, in this paper we assume that the membrane structure of a P system does not change through the computations. Neither cooperation nor priorities are used.

Let us observe that the rules of the system are associated with labels (e.g., the rule $[a \to v]_l$ is associated with the label $l \in H$). Rules are applied according to the following principles:

- The evolution rules are as customary usual in the framework of Membrane Computing, that is, in a maximal parallel way. In one step, each object in a membrane can only be used for one rule (non-deterministically chosen in case there are several possibilities), but any object which can trigger an evolution rule must evolve (with the restrictions indicated below for the communication case).
- All elements which are not specified in any of the operations to apply remain unchanged to the next step.

In the literature we can find two different semantics concerning rules which move objects across membranes.

- *Parallel communication case*: Communication rules follow the same principle of maximality as evolution rules, i.e., several objects (as many as possible, via the application of their respective rules) can cross simultaneously the same membrane.
- *Sequential communication case*: At one step, a membrane can only be the subject of *only one* communication rule. This is the case for P systems with active membranes, which has been extensively used for designing solutions to **NP** problems (see [15] and references therein).

2.4. Notation

We shall adopt the notation for rules and configurations given in [6], that we briefly recall in what follows.

Roughly speaking, the application of a rule in a P system can be interpreted as follows: *The occurrence of an element a_0 in a membrane m_1 triggers the rule and sends a multiset $a_1a_2 \dots a_n$ into a membrane m_2*. In this way, the rules presented above fit into the following schema (with some constraints)

$$(a_0, m_1) \to (a_1, m_2)(a_2, m_2) \dots (a_n, m_2)$$

Obviously, if $m_1 \neq m_2$ then we have a communication rule. In this case, n must be equal to 1 and both membranes must be adjacent (one membrane is directly inside the other one). If m_1 is contained inside m_2, then we have a send-out communication rule, and if the opposite holds, then we have a send-in communication rule. On the other hand, if $m_1 = m_2$, then we have an *evolution* rule.

As usual, the pair (a_0, m_1) is called the *left hand side (LHS)* of the rule and the multiset of pairs $(a_1, m_2)(a_2, m_2) \dots (a_n, m_2)$ is the *right hand side (RHS)* of the rule.

In the next sections, we shall consider that a configuration is represented as a multiset of pairs (z, m) such that, for every object z of the alphabet and for every membrane m, the multiplicity of z in m is the multiplicity of the pair (z, m) in the multiset. This notion is defined as the *L-configuration* of a P system in [6].

3. Applications of rules in a single step

A first step in order to have an estimation of the use of parallelism of a P system is to consider a single configuration. In the general case, there are several configurations $\mathcal{C}_1, \dots, \mathcal{C}_n$ reachable *in one transition step* from a given configuration $\mathcal{C}$. The number of applications of rules performed to reach each configuration $\mathcal{C}_i$ can vary, as the next example shows.

Example 5. Consider the P system with alphabet $\Gamma = \{a, b\}$, membrane structure $[\,[\,]_e\,]_s$, initial multisets $w_e = \{a\}$, $w_s = \{a\}$, set of rules:

$$\mathbf{R}_1 : [\,a\,]_e \to b\,[\,]_e \qquad \mathbf{R}_2 : a\,[\,]_e \to [\,b\,]_e \qquad \mathbf{R}_3 : [\,a \to b\,]_s$$

and the sequential communication semantics. By using rules $\mathbf{R}_1$ and $\mathbf{R}_3$ we reach the configuration $\mathcal{C}_1 \equiv [[\,]_e\ b^2\,]_s$ from the initial one $[[\,a\,]_e\ a\,]_s$. On the other hand, by using $\mathbf{R}_2$ we can also reach the configuration $\mathcal{C}_2 \equiv [[\,ab\,]_e\,]_s$. Notice that $\mathcal{C}_1$ has been obtained by two applications of rules and $\mathcal{C}_2$ by only one application.

Next let us try to determine the maximum number of applications of rules from a given configuration in one step, that will be denoted by $\beta_1(\Pi, \mathcal{C})$. In our study the key is to consider the multiset of elements which can trigger a rule. Due to the massive parallelism, if an element can trigger an evolution rule, then we are certain that this element will be consumed by the application of one of the rules triggered by it. If an element only triggers communication rules, then in order to deduce whether the object will be consumed we need to know if the system works with sequential or parallel communication semantics, how many rules can be triggered by that object, and whether there exist more objects that can cross the same membranes.

We shall start by defining two distinguished sub-multisets, $\mathcal{C}_E$ and $\mathcal{C}_C$, of a given configuration $\mathcal{C}$. Let Π be a P system and $\mathcal{C}$ a configuration of Π. We define:

$$\begin{aligned}
\mathcal{C}_E = \{(a, m)^k \mid (a, m) \quad & \text{is the LHS of an } \textit{evolution} \text{ rule and} \\
& \text{it occurs } k \text{ times in } \mathcal{C}\}, \\
\mathcal{C}_C = \{(a, m)^k \mid (a, m) \quad & \text{is the LHS of a } \textit{communication} \text{ rule and} \\
& \text{it is } \textit{not} \text{ the LHS of any } \textit{evolution} \text{ rule and} \\
& \text{it occurs } k \text{ times in } \mathcal{C}\}.
\end{aligned}$$

3.1. Sequential communication case

Let us first consider the sequential semantics. In this context we define a *crossing mapping*. The intuition behind this definition is the following: we intend to map membrane labels onto objects that can cross them in the next step by the application of a communication rule. In general, it may not be possible to find such a mapping defined over all labels, so we have to consider a subset $S \subseteq H$. Besides, in order to avoid that two objects cross the same membrane, we demand that the mapping is injective in the sense of Definition 1.

Definition 6. Let Π be a P system, H its set of labels, and $\mathcal{C}$ a configuration of Π. A *crossing mapping on* $\mathcal{C}$ is a mapping $f : S \to supp(\mathcal{C}_C)$ with $S \subseteq H$, injective w.r.t. $\mathcal{C}_C$, and such that for every $h \in S$ there exists a communication rule associated with label h and having $f(h)$ as its LHS.

Notice that for a given configuration there may exist several crossing mappings, that correspond to different non-deterministic choices of communication rules to be applied over the membranes with labels in S.

Finally, we give an expression of the maximum number of applications of rules in a transition step from the current configuration.

Theorem 7. *Let Π be a P system and $\mathcal{C}$ a configuration. Let us consider the sequential communication semantics. The maximum number $\beta_1(\Pi, \mathcal{C})$ of applications of rules in one transition step starting from $\mathcal{C}$ is*

$$\beta_1(\Pi, \mathcal{C}) = \#\mathcal{C}_E + \max\{\#S \mid \exists \text{ a crossing mapping } f : S \to supp(\mathcal{C}_C)\}.$$

Proof. Let Π be a P system, $\mathcal{C}$ a configuration and let $f^* : S^* \to supp(\mathcal{C}_C)$ be a crossing mapping such that $\#S^* = \max\{\#S \mid \exists \text{ a crossing mapping } f : S \to supp(\mathcal{C}_C)\}$. Then, for every label $h \in S^*$ there exists a communication rule that crosses membrane h with $f^*(h)$ as its LHS. By the definition of crossing mapping, there exists one transition step starting from $\mathcal{C}$ in which all objects $f^*(h)$ trigger their corresponding communication rules crossing the membrane labelled by h, and this causes $\#S^*$ application of rules.

Let us consider now the set $\mathcal{C}_E$. By definition, and due to the maximal parallelism, all the objects in $\mathcal{C}_E$ trigger an evolution rule in any transition step starting from $\mathcal{C}$. In particular, there exists a transition step where the number of application of rules is exactly $\#\mathcal{C}_E + \#S^*$. In other words, the maximum number of applications of rules in a transition step starting from $\mathcal{C}$, $\beta_1(\Pi, \mathcal{C})$, is at least $\#\mathcal{C}_E + \max\{\#S \mid \exists \text{ a crossing mapping } f : S \to supp(\mathcal{C}_C)\}$.

Next, we shall prove that the maximum number of applications of rules $\beta_1(\Pi, \mathcal{C})$ is not greater than $\#\mathcal{C}_E + \max\{\#S \mid \exists \text{ a crossing mapping } f : S \to supp(\mathcal{C}_C)\}$.

Let $\mathcal{C}_1$ be a configuration obtained from $\mathcal{C}$ in one transition step and let T_E (T_C, resp.) be the multiset of elements of the configuration $\mathcal{C}$ which have triggered evolution (communication, resp.) rules in order to reach $\mathcal{C}_1$. Obviously, the number of applications of rules performed in the transition step $\mathcal{C} \Rightarrow \mathcal{C}_1$ is $\#T_E + \#T_C$. Now, let us split the multiset T_C into two multisets: T_C^e, the submultiset containing the elements from T_C which are the LHS of some evolution rule, and T_C^{ne}, the complementary submultiset containing the elements from T_C which are *not* the LHS of any evolution rule. We have that $\#T_C = \#T_C^e + \#T_C^{ne}$ and

- $\#T_E + \#T_C^e = \#\mathcal{C}_E$, since $T_E \cup T_C^e = \mathcal{C}_E$.
- $\#T_C^{ne} \leq \max\{\#S \mid \exists \text{ a crossing mapping } f : S \to supp(\mathcal{C}_C)\}$, since we can consider the natural crossing mapping $f^* : S_T \to supp(\mathcal{C}_C)$ where S_T is the set of labels of membranes crossed by the objects in T_C^{ne}.

Therefore, the number of applications of rules in the transition step from $\mathcal{C}$ to $\mathcal{C}_1$ is less than or equal to $\#\mathcal{C}_E + \max\{\#S \mid \exists \text{ a crossing mapping } f : S \to supp(\mathcal{C}_C)\}$, and this concludes the proof. □

Due to the high computational cost of computing $\beta_1(\Pi, \mathcal{C})$, it is worth looking for an upper bound that is easier to compute. The next corollary gives such an upper bound for $\beta_1(\Pi, \mathcal{C})$. The proof is immediate.

Corollary 8. *Let Π be a P system working with a sequential communication semantics. Let $\mathcal{C}$ be a configuration, and H the set of labels of Π. Then*

$$\beta_1(\Pi, \mathcal{C}) \leq \#\mathcal{C}_E + \#H.$$

We illustrate the theorem with the following example.

Example 9. Let us consider a P system with a set of labels $H = \{0, 1, 2, 3\}$ and the following set of rules:

$$\begin{array}{llll}
\mathbf{R}_1: & [\,a\,]_1 \to b\,[\,]_1 & \mathbf{R}_5: & z\,[\,]_1 \to [x]_1 \qquad \mathbf{R}_8: \quad [\,a \to b\,]_1 \\
\mathbf{R}_2: & [\,a\,]_2 \to b\,[\,]_2 & \mathbf{R}_6: & z\,[\,]_2 \to [x]_2 \qquad \mathbf{R}_9: \quad [\,a \to b\,]_2 \\
\mathbf{R}_3: & [\,a\,]_3 \to b\,[\,]_3 & \mathbf{R}_7: & z\,[\,]_3 \to [x]_3 \qquad \mathbf{R}_{10}: \quad [\,a \to b\,]_3 \\
\mathbf{R}_4: & [\,x\,]_1 \to b\,[\,]_1 & &
\end{array}$$

Consider now a configuration $\mathcal{C} = [\,[\,ax^3\,]_1\,[\,a^2z\,]_2\,[\,az\,]_3\,z^3\,]_0$. Then,

$$\mathcal{C}_E = \{(a, 1), (a, 2), (a, 2), (a, 3)\},$$
$$\mathcal{C}_C = \{(x, 1), (x, 1), (x, 1), (z, 0), (z, 0), (z, 0)\}.$$

Therefore, we have $\#\mathcal{C}_E = 4$ and $\#H = 4$. From the previous corollary we deduce that $\beta_1(\Pi, \mathcal{C}) \leq 8$.

Actually, a more detailed study of the example shows that

$$\max\{\#S \mid \exists \text{ a crossing mapping } f : S \to \mathcal{C}_C\} = 3$$

(e.g. by taking $S = \{1, 2, 3\}$ and $f(1) = (x, 1)$, $f(2) = (a, 2)$, $f(3) = (z, 0)$). Thus, from Theorem 7 we obtain the maximum number of applications of rules in a transition step from the current configuration: $\beta_1(\Pi, \mathcal{C}) = 7$.

3.2. Parallel communication case

In order to compute the maximum number of applications of rules performed in one step in the parallel case we do not need to study separately evolution and communication rules.

Theorem 10. *Let Π be a P system using a parallel communication semantics, and let $\mathcal{C}$ be a configuration of Π. Then the maximum number $\beta_1(\Pi, \mathcal{C})$ of applications of rules performed in one transition step starting from $\mathcal{C}$ is $\beta_1(\Pi, \mathcal{C}) = \#\mathcal{C}_E + \#\mathcal{C}_C$.*

Proof. In this case the number of applications of rules performed in one transition step starting from $\mathcal{C}$ is constant, regardless of the non-determinism of the P system. Indeed, because the maximal parallel condition applies both over evolution and communication rules, no "usable" object will remain unused. Therefore, the number of applications of rules for any possible transition step starting from $\mathcal{C}$ coincides with the number of objects in the configuration that can trigger at least one rule. □

The following example illustrates this result.

Example 11. Let us consider again the P system from Example 9 and the same configuration, but now considering the parallel communication semantics. In this case, the sets $\mathcal{C}_E$ and $\mathcal{C}_C$ do not change, because they are independent of the semantics of the system. Therefore, $\beta_1(\Pi, \mathcal{C}) = \#\mathcal{C}_E + \#\mathcal{C}_C = 10$.

4. An estimation of maximal parallelism

In the previous section we have obtained the maximum number of applications of rules for any transition step starting from a given configuration. We intend to extend our study to all possible transition steps in the computation tree. Given a configuration, there might exist several possible computations to follow, and many different reachable configurations. We shall give an upper bound to the number of applications of rules performed in a transition step from any configuration of the computation tree.

Such a bound can be used for estimating, given a P system design, which is the degree of parallelism that it is using, as it gives a maximum number of simultaneous applications of rules for any step of any computation of the system. It is worth remarking that this estimation is done *without* performing all the computations.

In order to obtain such an estimation, we start by building an extension to the notion of *dependency graphs* of a P system, that will be called *labelled dependency graphs*. Dependency graphs of P systems were presented in [5] as a tool for finding short paths in computation trees. Later they were used for studying the borderline of the tractability through P systems (see, e.g., [10]). Before going on, we briefly recall the main features of dependency graphs.

The *dependency graph* of a P system Π is a directed graph $G_\Pi = (V_\Pi, E_\Pi)$ such that V_Π is the set of all the pairs (z, m) where z is a symbol of the alphabet of Π and m is the label of a membrane, and E_Π is the set of all ordered

pairs (arcs) of elements of V_Π, $((z_1, m_1), (z_2, m_2))$, where (z_1, m_1) is the LHS of a rule and (z_2, m_2) *belongs* to the RHS of that rule (recall the notation introduced in Section 2.4).

The previous definition can be extended to *labelled dependency graphs* by modifying the arcs in the following way. Given a rule

$$r \equiv (a_0, m_1) \to (a_1, m_2)(a_2, m_2) \dots (a_n, m_2),$$

for each (e, m_2) in the RHS of r we shall consider a directed labelled arc

$$(a_0, m_1) \xrightarrow{r/k} (e, m_2),$$

where k is the multiplicity of (e, m_2) in the RHS of rule r. It is clear that, given two nodes of the graph, there might exist several arcs connecting them with different labels,[3] in case both nodes are involved together in more than one rule.

Note that the *labelled dependency graph*, in the same way as the dependency graph, does not depend on the initial configuration. It only depends on the alphabet, the membrane structure and the set of rules of the P system.

We shall extract information from this labelled dependency graph that allows us to compute an integer $\beta(\Pi, \mathcal{C})$, associated with a P system Π and a configuration $\mathcal{C}$, denoting an "accurate" (in a sense that will be explained later) upper bound of the number of simultaneous applications of rules performed in any transition step starting from any configuration reachable from $\mathcal{C}$. In a certain sense, $\beta(\Pi, \mathcal{C})$ represents the capability of the P system to exploit the parallelism through the computation. The greater $\beta(\Pi, \mathcal{C})$ is, the *better* is the design, from the point of view of using the parallelism.

Next, we give an explicit expression for $\beta(\Pi, \mathcal{C})$ (if it exists) based on the labelled dependency graph. As we did in the previous section, we distinguish between the different semantics of the P systems.

4.1. Sequential communication case

Let us first consider the semantics where only *one* object can cross each membrane in a transition step.

Theorem 12. *Let Π be a P system, $\mathcal{C}$ a configuration of Π and let $\mathbb{T}$ be the set of LHS of the rules of Π. Let us consider the sequential communication semantics. Let $\mathcal{G}$ be the subgraph of the labelled dependency graph of Π generated by the source $\mathcal{C}$ and sink $\mathbb{T}$. The following two statements are equivalent:*

(1) *The number of simultaneous applications of rules in one step starting from any configuration of Π reachable from $\mathcal{C}$ is bounded, and therefore there is a* maximum *number of such applications.*
(2) *There is no cycle in $\mathcal{G}$ such that the labels r/k of its arcs verify that* (*a*) *all the rules r involved in the cycle are* evolution *rules, and* (*b*) *at least one of the multiplicities k appearing in the cycle is greater than* 1.

Proof. Let us prove $1 \Rightarrow 2$ by *reductio ad absurdum*. That is, let us consider the case when there exists in $\mathcal{G}$ a cycle such that all the rules involved in it are evolution rules and such that at least one of the multiplicities k appearing in the cycle is greater than 1. We have to deduce that there is no upper bound on the number of simultaneous applications of rules in one step starting from any configuration of Π reachable from $\mathcal{C}$.

Let $\psi = (V, E)$ be such a cycle, with $V = \{x_0, \dots, x_n\}$ and $E = \{(x_i, x_{i+1}) \mid i \in \{0, \dots, n-1\}\} \cup \{(x_n, x_0)\}$ and let K be the product of multiplicities associated with the arcs in ψ, $K \geq 2$. Consider now an element z_0 in the configuration $\mathcal{C}$ such that there exists a path, $\mathcal{P}$, from z_0 to x_0 in the labelled dependency graph. The existence of such z_0 is granted since, by definition, $\mathcal{G}$ is the subgraph of the labelled dependency graph of Π generated by the source $\mathcal{C}$ and sink $\mathbb{T}$, and therefore every vertex in $\mathcal{G}$ (in particular, x_0) is reachable from a vertex from $\mathcal{C}$. Let us denote the product of multiplicities of $\mathcal{P}$ by S, and let m be the length of $\mathcal{P}$.

There exists a computation such that in the configuration $\mathcal{C}'$ obtained after m steps there are at least S occurrences of x_0 (these copies of x_0 are obtained by application of the rules of the path $\mathcal{P}$). Then, after n further steps (n is the length of the cycle ψ) we can reach another configuration where each element x_0 has produced K copies of itself, following the cycle ψ (i.e., the total number of copies of x_0 is now at least $S \cdot K$). This cycle of n steps can be iterated,

[3] Following [12], the definition of the labelled dependency graph corresponds to an *edge-labelled, vertex-labelled, directed loop multigraph*.

and after r loops there will be at least $S \cdot K^r$ elements x_0. All these elements may continue evolving in the next step of the computation, granting the existence of an infinite computation with unbounded number of objects evolving. Thus, we conclude that there is no upper bound for the number of applications in a single transition step.

Let us now prove $2 \Rightarrow 1$. We start from the assumption that if there are cycles in the labelled dependency graph, then they fail to satisfy either the condition (a) or (b).

On one hand, consider cycles having at least one communication rule involved in it. Notice that in this case, such a communication rule can be applied *only once* for each transition step, regardless of the number of objects that could trigger it. That is, the number of simultaneous applications of any communication rule is always 1, irrespectively of the number of available copies of the object appearing in its LHS. In other words, the number of applications of rules in a single step inside the cycle is bounded, even if the product of multiplicities in the arcs in the cycle was greater than 1.

On the other hand, it is clear that cycles in $\mathcal{G}$ such that all the multiplicities of its arcs are equal to 1 do not cause an increase in the number of objects, and therefore do not cause an increase in the number of simultaneous applications of rules in one transition step.

Now, we are going to find, in a constructive way, a bound on the number of simultaneous applications of rules in one transition step starting from any configuration reachable from $\mathcal{C}$. First of all, we know that the number of objects in $\mathcal{C}$ that can trigger a rule is $\#\mathcal{C}_E + \#\mathcal{C}_C$, and thus the number of applications of rules in any transition step starting from $\mathcal{C}$ is at most $\#\mathcal{C}_E + \#\mathcal{C}_C$.

Let B_Π be the maximum of the cardinality of the RHS of the rules of Π; then, the number of objects that can trigger a rule in any configuration reachable in one transition step from $\mathcal{C}$ is bounded by $(\#\mathcal{C}_E + \#\mathcal{C}_C) \cdot B_\Pi$. This number is obviously also an upper bound for the number of applications of rules in the second step of any computation starting from $\mathcal{C}$. Following a similar reasoning, we deduce that $(\#\mathcal{C}_E + \#\mathcal{C}_C) \cdot (B_\Pi)^n$ is an upper bound for the number of applications of rules performed in any of the $(n+1)$ first steps of *any* possible computation starting from $\mathcal{C}$.

Let us consider now the labelled dependency graph of the P system, and consider all the simple paths in this graph starting from objects in $\mathcal{C}$. Let us denote by $\rho_{\Pi,\mathcal{C}}$ the maximum length of these paths.

The number of consecutive transition steps increasing the number of applications of rules is bounded by $\rho_{\Pi,\mathcal{C}} - 1$, so the upper bound for the number of applications of rules in a single transition step during any computation starting from $\mathcal{C}$ is $(\#C_E + \#C_C) \cdot B^{(\rho_{\Pi,\mathcal{C}})-1}$.

Note that the computation can be infinite in the case that there exist cycles in $\mathcal{G}$, but as we checked above, they do not cause an increase in the number of applications of rules. □

From the constructive proof of the previous theorem we deduce the following corollary.

Corollary 13. *Let Π be a P system, and let $\mathcal{C}$ be a configuration of Π. If the maximum number of simultaneous applications of rules in any step starting from $\mathcal{C}$ exists (i.e., if it is finite), then it is less than or equal to the bound*

$$\beta(\Pi, \mathcal{C}) = (\#\mathcal{C}_E + \#\mathcal{C}_C) \cdot B_\Pi^{(\rho_{\Pi,\mathcal{C}})-1}$$

where:

- *B_Π is the maximum cardinal of the RHS of the rules of Π.*
- *$\rho_{\Pi,\mathcal{C}}$ is the maximum length of the simple paths in the labelled dependency graph of Π starting from an element of $\mathcal{C}$.*

We illustrate Theorem 12 with some examples. In the first one $\beta(\Pi, \mathcal{C})$ is finite, although we have an infinite computation and the number of objects in a configuration is not bounded.

Example 14. Let us consider the P system Π with alphabet $\Gamma = \{a, b, c\}$, membrane structure $\mu = [\,[\,]_1\,]_0$, and set of rules:

$$\mathbf{R}_1 : [\,a\,]_1 \to b\,[\,]_1 \qquad \mathbf{R}_2 : [\,b \to c^2\,]_0 \qquad \mathbf{R}_3 : c\,[\,]_1 \to [\,a\,]_1$$

and let us consider the sequential semantics. The labelled dependency graph of this P system is depicted in Fig. 2.

Π is a non-deterministic P system (as there are two communication rules associated with label 1, and we are considering the sequential semantics). Let $[\,[\,a\,]_1\,]_0$ be a configuration of Π. A possible computation starting from $\mathcal{C}$ is:

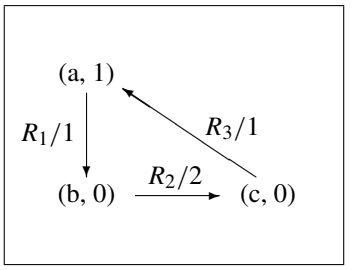

Fig. 2. The labelled dependency graph of Example 14.

From	$[\,[\,a\,]_1\,]_0$	with *one* application of rule $\mathbf{R}_1$ we reach	$[\,[\,]_1\,b\,]_0$
From	$[\,[\,]_1\,b\,]_0$	with *one* application of rule $\mathbf{R}_2$ we reach	$[\,[\,]_1\,c^2\,]_0$
From	$[\,[\,]_1\,c^2\,]_0$	with *one* application of rule $\mathbf{R}_3$ we reach	$[\,[\,a\,]_1\,c\,]_0$
From	$[\,[\,a\,]_1\,c\,]_0$	with *one* application of rule $\mathbf{R}_1$ we reach	$[\,[\,]_1\,bc\,]_0$
From	$[\,[\,]_1\,bc\,]_0$	with *one* application of rule $\mathbf{R}_2$ and *one* application of rule $\mathbf{R}_3$ we reach	$[\,[\,a\,]_1\,c^2\,]_0$
From	$[\,[\,a\,]_1\,c^2\,]_0$	with *one* application of rule $\mathbf{R}_1$ we reach	$[\,[\,]_1\,bc^2\,]_0$
From	$[\,[\,]_1\,bc^2\,]_0$	with *one* application of rule $\mathbf{R}_2$ and *one* application of rule $\mathbf{R}_3$ we reach	$[\,[\,a\,]_1\,c^3\,]_0$
		...	

Notice that in this computation, after $2k+1$ transition steps we reach a configuration $[\,[\,a\,]_1\,c^k\,]_0$. This computation is infinite and the number of objects c is not bounded. However, the number of applications of rules in any single transition step from any configuration is bounded by 2.

It is worth remarking that in order to calculate this upper bound we do not need to develop the computation tree. Notice also that in this example there exists a cycle in the labelled dependency graph such that one of the multiplicities appearing in it is greater than 1. However, since there are communication rules involved in the cycle, the statement (2) of Theorem 12 holds.

We have $\mathcal{C}_C = \{(a,1)\}$, $\mathcal{C}_E = \emptyset$, $B_\Pi = 2$, and $\rho_{\Pi,\mathcal{C}} = 2$, so from Theorem 12 and Corollary 13 we deduce

$$\beta(\Pi,\mathcal{C}) = (\#\mathcal{C}_E + \#\mathcal{C}_C)\cdot B_\Pi^{(\rho_{\Pi,\mathcal{C}})-1} = 1\cdot 2^{2-1} = 2.$$

The following example shows that the upper bound $\beta(\Pi,\mathcal{C})$ coincides in some cases with the *maximum*.

Example 15. Let us consider the P system Π with alphabet $\Gamma = \{a_1,\ldots,a_k,x,\ y,z\}$, membrane structure $\mu = [\,[\,]_1\,[\,]_2\,]_0$, and set of rules:

$\mathbf{R}_1$: $[\,x\,]_1 \to a_1\,[\,]_1$ $\quad$ $\mathbf{R}_3$: $[\,z \to a_1]_0$

$\mathbf{R}_2$: $[\,y\,]_2 \to a_1\,[\,]_2$ $\quad$ $\mathbf{R}_{3+i}$: $[\,a_i \to a_{i+1}^2\,]_0$ with $i \in \{1,\ldots,k-1\}$

and let us consider the sequential semantics. The labelled dependency graph of this P system is depicted in Fig. 3.

Π is a deterministic P system. Let $[\,[\,x\,]_1\,[\,y\,]_2\,z\,]_0$ be a configuration of Π. In the first step we have three applications of rules ($\mathbf{R}_1$, $\mathbf{R}_2$, and $\mathbf{R}_3$ are applied once each) which produces the configuration $[\,[\ ]_1\,[\ ]_2\,a_1^3\,]_0$, with three objects a_1 in the membrane labelled by 0. It is easy to check that in the i-th step, with $i \in 2,\ldots,k$, we have $3\cdot 2^{i-2}$ applications of rule $\mathbf{R}_{i+2}$ which produce $3\cdot 2^{i-1}$ objects a_i in the membrane labelled by 0. Therefore, the maximum number of applications of rules in one transition step of any computation is reached in the last step, where we have $3\cdot 2^{k-2}$ applications of rules.

The upper bound $\beta(\Pi,\mathcal{C})$ can be obtained following the formula of Corollary 13. Since $\mathcal{C}_E = \{(a_1,0)\}$, $\mathcal{C}_C = \{(x,1),(y,2)\}$, $B_\Pi = 2$ and $\rho_{\Pi,\mathcal{C}} = k$, then we have

$$\beta(\Pi,\mathcal{C}) = (\#\mathcal{C}_E + \#\mathcal{C}_C)\cdot B_\Pi^{(\rho_{\Pi,\mathcal{C}})-1} = 3\cdot 2^{k-1}.$$

Therefore, the maximum number of applications of rules actually performed in the computation ($3\cdot 2^{k-2}$) is strictly less than the upper bound $\beta(\Pi,\mathcal{C})$.

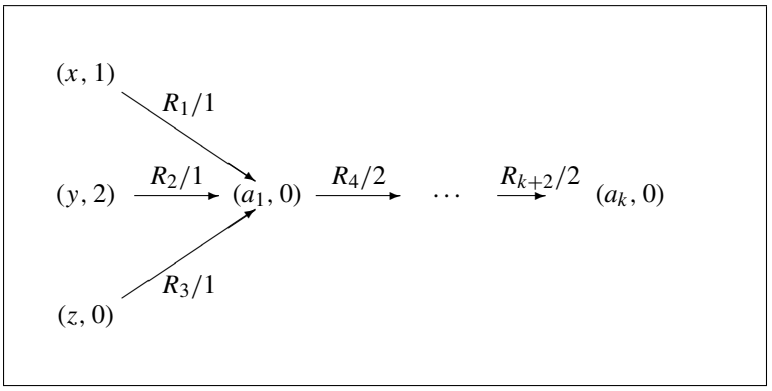

Fig. 3. The labelled dependency graph of Example 15.

Let us consider now a different initial configuration, namely $[\,[\,]_1\,[\,]_2\,a_1^3]_0$. Then, in the first step three applications of $\mathbf{R}_4$ are performed, producing the configuration $[\,[\,]_1\,[\,]_2\,a_2^6]_0$. In the second step, by $3\cdot 2$ applications of $\mathbf{R}_5$ we obtain $[\,[\,]_1\,[\,]_2\,a_3^{12}]_0$. Finally, in the $(k-1)$-th step, by $3\cdot 2^{k-2}$ applications of rule $\mathbf{R}_{k+2}$ we obtain the final configuration $[\,[\,]_1\,[\,]_2\,a_k^{3\cdot 2^{k-1}}]_0$.

In this case we have $\mathcal{C}_E = \{(a_1,0),(a_1,0),(a_1,0)\}$, $\mathcal{C}_C = \emptyset$, $B_\Pi = 2$ like before, but $\rho_{\Pi,\mathcal{C}} = k-1$. Therefore, the upper bound

$$\beta(\Pi,\mathcal{C}) = (\#\mathcal{C}_E + \#\mathcal{C}_C)\cdot B_\Pi^{(\rho_{\Pi,\mathcal{C}})-1} = 3\cdot 2^{(k-2)}$$

is actually reached.

4.2. Parallel communication case

Let us now consider the parallel communication semantics. We obtain a similar result as in the sequential case. The only difference is that in the previous case we have to check whether the cycle contains or not a communication rule, since the flow in a cycle with communication rules is *one* due to the semantics. In the parallel communication case, whenever there is a cycle in $\mathcal{G}$ such that at least one of the multiplicities of its arcs is greater than 1, there is no upper bound on the number of applications of rules in a transition step, regardless of the occurrence of communication rules in the cycle.

Theorem 16. *Let Π be a P system, $\mathcal{C}$ a configuration of Π and let $\mathbb{T}$ be the set of LHS of the rules of Π. Let us consider the parallel communication semantics. Let $\mathcal{G}$ be the subgraph of the labelled dependency graph of Π generated by the source $\mathcal{C}$ and sink $\mathbb{T}$. The following two statements are equivalent:*

(1) *The number of simultaneous applications of rules in one step starting from any configuration of Π reachable from $\mathcal{C}$ is bounded, and therefore there is a* maximum *number of such applications.*
(2) *There is no cycle in $\mathcal{G}$ such that the labels r/k of its arcs verify that at least one of the multiplicities k appearing in the cycle is greater than* 1.

The proof is similar to the proof of Theorem 12.

Example 17. Let us consider again Example 14, with the same configuration $[\,[\,a\,]_1\,]_0$, but now under the parallel communication semantics. The labelled dependency graph is the same, but the system is now deterministic. Indeed, the possible choices in the sequential case occurred only when several communication rules were applicable to the same membrane.

The reader can easily check that for every $k \geq 0$, we have

$$\mathcal{C}_{3k} \equiv [\,[\,a^{2^k}\,]_1\,]_0 \qquad \mathcal{C}_{3k+1} \equiv [\,[\,]_1\,b^{2^k}\,]_0 \qquad \mathcal{C}_{3k+2} \equiv [\,[\,]_1\,c^{2^{k+1}}\,]_0$$

In every step all the objects trigger a rule, and hence the number of applications of rules is not bounded. We have deduced this by tracing the computation, but this is not necessary, as we can draw the same conclusion from Theorem 16.

Example 15 also shows that the upper bound $\beta(\Pi, \mathcal{C})$ can be reached in the parallel communication case, since the behaviour of the computation is identical regardless of the communication semantics.

5. Conclusions and further work

The comparison of two P systems which perform the same task is very hard. These computational devices are too complex to be evaluated only with usual *time* and *space* parameters. Sevilla carpets and their associated parameters are useful tools for the comparison of two computations, and therefore useful for deterministic P systems, but not in the general case.

In this paper we propose a first step in this direction. We introduce the concept of *injective mapping with respect to a multiset*, and we extend the definition of dependency graph to *labelled dependency graph* by adding more information. The usefulness of this new tool should be further investigated in the future.

We have proved several results concerning the estimation of the *actual* use of parallelism in P systems (working in the maximally parallel mode). We consider four main open directions for developing the ideas presented in this paper.

First of all, a very natural improvement is to search for an estimation of the *average* number of applications of rules in each step of the computation of a P system, instead of knowing only the maximum.

Secondly, the P system model studied in this paper is quite simple, only allowing pure evolution or communication rules. It is interesting to study the effect on the bounds of parallelism of other types of rules, e.g. symport/antiport, dissolution, etc.

Another open line is related to calculating estimations for the parallelism on families of P systems. This is motivated by the fact that in the literature (uniform or semi-uniform) solutions to decision problems are usually carried out via families of P systems, not by using only a single P system. It is thus very interesting to extend the results presented here to families of P systems, in order to be able to compare different solutions to the same decision problem.

Finally, other P system models can also be explored, for instance, those with minimal [3] or bounded [11] parallelism. We also refer to [1] for a study (from the computing point of view) of a series of variants of parallelism in P systems with string objects.

Acknowledgements

The authors wish to acknowledge the support of the Project TIN2005-09345-C04-01 of the Ministerio de Educación y Ciencia of Spain, cofinanced by FEDER funds, and the support of the Project of Excellence TIC 581 of the Junta de Andalucía.

References

[1] D. Besozzi, Computational and modelling power of P systems, Ph.D. Thesis, Università degli Studi di Milano, 2004.

[2] G. Ciobanu, Gh. Păun, Gh. Ştefănescu, Sevilla carpets associated with P systems, in: M. Cavaliere, C. Martín-Vide, Gh. Păun (Eds.), Proceedings of the Brainstorming Week on Membrane Computing, Tarragona, Spain, 2003, Report RGML 26/03, pp. 135–140.

[3] G. Ciobanu, Gh. Păun, M.J. Pérez-Jiménez, P systems with minimal paralellism, Theoretical Computer Science (2006) (in press).

[4] G. Ciobanu, Gh. Păun, M.J. Pérez-Jiménez, On the branching complexity of P systems, Fundamenta Informaticae 73 (1–2) (2006) 27–36.

[5] A. Cordón-Franco, M.A. Gutiérrez-Naranjo, M.J. Pérez-Jiménez, A. Riscos-Núñez, Weak metrics on configurations of a P system, in: Gh. Păun, A. Riscos-Núñez, A. Romero-Jiménez, F. Sancho-Caparrini (Eds.), Proceedings of the Second Brainstorming Week on Membrane Computing, Seville, Spain, 2004, Report RGNC 01/04, pp. 139–151.

[6] A. Cordón-Franco, M.A. Gutiérrez-Naranjo, M.J. Pérez-Jiménez, A. Riscos-Núñez, Exploring computation trees associated with P systems, Lecture Notes in Computer Science 3365 (2005) 278–286.

[7] M.A. Gutiérrez-Naranjo, M.J. Pérez-Jiménez, A. Riscos-Núñez, On descriptive complexity of P systems, Lecture Notes in Computer Science 3365 (2005) 320–330.

[8] M.A. Gutiérrez-Naranjo, M.J. Pérez-Jiménez, A. Riscos-Núñez, A fast P system for finding a balanced 2-partition, Soft Computing 9 (9) (2005) 673–678.

[9] M.A. Gutiérrez-Naranjo, M.J. Pérez-Jiménez, A. Riscos-Núñez, Multidimensional descriptional complexity of P systems, in: C. Mereghetti, B. Palano, G. Pighizzini, D. Wotschke (Eds.), Proceedings of the 7th International Workshop on Descriptional Complexity of Formal Systems, Como, Italy, 2005, pp. 134–145.

[10] M.A. Gutiérrez-Naranjo, M.J. Pérez-Jiménez, A. Riscos-Núñez, F.J. Romero-Campero, On the power of dissolution in P systems with active membranes, Lecture Notes in Computer Science 3850 (2005) 373–394.

[11] O.H. Ibarra, H.-C. Yen, Z. Dang, On various notions of parallelism in P Systems, International Journal of Foundations of Computer Science 16 (4) (2005) 683–705.

[12] B. Mehdi, G. Chartrand, Introduction to the Theory of Graphs, Allyn and Bacon, Inc., Boston, 1971.
[13] Gh. Păun, Membrane Computing, An Introduction, Springer-Verlag, Berlin, 2002.
[14] M.J. Pérez-Jiménez, A. Riscos-Núñez, Solving the Subset-Sum problem by active membranes, New Generation Computing 23 (4) (2005) 367–384.
[15] A. Riscos-Núñez, Cellular programming: efficient resolution of numerical NP-complete problems, Ph.D. Thesis, University of Seville, 2004.
[16] A. Syropoulos, Mathematics of multisets, Lecture Notes in Computer Science 2235 (2001) 347–358.