An Approach to the Degree of Parallelism in P 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, 41012 Sevilla, Spain
{magutier,marper,ariscosn}@us.es

Summary. In the literature, several designs of P systems were 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. In this paper, we introduce a new criterion for such a comparison: the degree of parallelism of a P system. To this aim, we define the *labeled 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 along the computation of a P system.

1 Introduction

In the last 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 has "computed" a numerical function over 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 systems 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 especially hard.

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 [1]. In this paper, a new tool for the descriptive complexity of P systems called *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 [6] and a multidimensional generalization of Sevilla carpets was presented in [8].

Sevilla carpets and their associated parameters provide very useful information to describe the evolution of a P system, but *only* for one computation. 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. But 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 basis 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 along the computation is the basis of the design of families of P systems which solve **NP**-complete problems in polynomial time (see, e.g., [5, 7, 12, 13, 16, 18, 19, 17], and also [20] and references therein)

In this paper we focus our attention on the parallelism in order to have 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, it works as a sequential machine: in each step only *one* object evolve in *one* membrane whereas the remaining objects do not evolve. On the other hand, a *good* design consists on 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 the general case the comparison is not so easy and, in most cases, it would be useful to have a numerical function which captures the idea of 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

[1] Recently two new parameters have been introduced in [2] 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.

different even in confluent P systems. In this paper we propose a parameter based on the (potentially) maximum number of 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, $\beta(\Pi, \mathcal{C})$, 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 $\mathcal{C}$. Finally, some conclusions and lines for future work are given.

2 Preliminaries

In this section we recall some concepts which will be used along the paper. First of all, we remind 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 will work with in this paper and adopt some conventions for notation.

2.1 Multisets

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

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 $\mathcal{A} = \langle A, f\rangle$ is a multiset, its *support*, $supp(\mathcal{A})$ is defined as $supp(\mathcal{A}) = \{x \in A \,|\, f(x) > 0\}$ and its cardinality, denoted by $\#\mathcal{A}$, is defined as

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

Suppose that $\mathcal{A} = \langle A, f\rangle$ and $\mathcal{B} = \langle A, g\rangle$ are two multisets over the set A.

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

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 will say that g is* injective with respect to the multiset $\langle D_2, f\rangle$ *if* $\forall y \in D_2 \ \big(\#\{x \in D_1 \,|\, g(x) = y\} \leq f(y)\big)$.

[2] A detailed presentation of multisets can be found, for example, in [21].

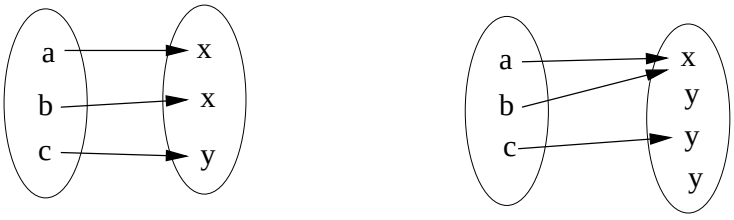

Fig. 1. Mappings on multisets

This definition can be illustrated with the following example.

Example 1. Let us consider the sets $D_1 = \{a, b, c\}$ and $D_2 = \{x, y\}$ and the multisets[3] 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 Figure 1.) Note that this definition expands the usual definition of injective mappings over sets.

2.2 Graphs

In this paper we will use directed graphs as a structure for organizing information. 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 set and $E \subseteq V \times V$. The elements of V are called *nodes*, and the elements of E are called *arcs*. Given a directed graph $\mathcal{G}$, a *subgraph* $\mathcal{G}'$ from $\mathcal{G}$ is a pair $\mathcal{G}' = (V', E')$ such that $V' \subseteq V$, $E' \subseteq E \cap (V' \times V')$. 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 will consider the *paths* in a directed graph as *subgraphs* following the next definition.

Definition 2. *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 will say that the path is a* cycle. *The subgraph with a single vertex and no arcs is also considered a path.*

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

Definition 3. *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 $y \in B$.*

[3] With the usual notation $\{x^{f(x)} \mid x \in D\}$ for the multiset $\langle D, f\rangle$.

2.3 P systems

We assume that the reader is familiar with the standard P system models. In this section we briefly define a simple model that we will be using along this paper.

Recall that, basically, a P system consists of a cell-like membrane structure together with associated multisets of objects and a set of rules expressing how these objects can evolve (see [15]). 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 variant of the model we are dealing with, the membranes can be dissolved, divided, or created. Nevertheless, in this paper we shall work in a simplified model without division, dissolution nor 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 the 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 sequence of such transitions (finite or infinite) is called a *computation*.

More formally, a P system 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 labeled bijectively by elements from H.
- $w_1, \ldots, w_q$ are multisets over Γ describing the multisets of objects initially placed in the membranes of μ .
- R is a finite set of developmental rules. These rules can be of two types: *evolution rules* where the object that triggers the rule do not cross any membrane and *communication rules* where the object which triggers the rule *do* cross a membrane. These communication rules can be of type *send-in* or *send-out* as 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 labeled 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 labeled 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 labeled by l, possibly transformed in a new object, b.

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 applied as usual in the framework of Membrane Computing, that is, in a maximally 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 evolve by a rule of any form 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 cross 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 profusely used for designing solutions to **NP** problems (see [20] and references therein).

2.4 Notation

We will adopt the notation for rules and configurations given in [4], that we briefly recall in what follows.

Roughly speaking, transitions in P systems are performed by rules in which the occurrence of an element a_0 in a membrane m_0 sends the element a_1 into a membrane m_1. In a certain sense, one can consider a dependency between the pair (a_0, m_0) and the pair (a_1, m_1). The rules in the P system model presented above fit into the following schema (with some constraints):

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

which can be interpreted as follows: *The occurrence of the element a_0 in the membrane m_1 triggers the rule and sends the multiset $a_1 a_2 \dots a_n$ into the membrane 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 contained 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 called an *L-configuration* of a P system in [4].

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 $\mathcal{C}_1,\ldots,\mathcal{C}_n$ reachable *in one step* of a computation from a given configuration $\mathcal{C}$. The number of applications of rules to reach each configuration $\mathcal{C}_i$ can vary as the next example shows.

Example 2. Consider the P system with the alphabet $\Gamma = \{a, b\}$, the membrane structure $[\,[\,]_e\,]_s$, the initial multisets $w_e = \{a\}$, $w_s = \{a\}$, and the set of rules:

$$\mathbf{R1} : [\,a\,]_e \to b\,[\,]_e, \qquad \mathbf{R2} : a\,[\,]_e \to [\,b\,]_e, \qquad \mathbf{R3} : [\,a \to b\,]_s,$$

and the sequential communication semantics. By using rules **R1** and **R3** we reach the configuration $\mathcal{C}_1 = [[\,]_e\ b^2\,]_s$ from the initial one $[[\,a\,]_e\ a\,]_s$. On the other hand, by using **R2** 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 know if the object will be consumed we need to know if only one or several objects can cross a membrane, how many rules can be triggered by that object, and whether there exist or not more objects that can cross the same membrane.

We will 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:

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

3.1 The 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 4. *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. $\mathcal{C}_C$, such that for every $h \in S$ there exists a communication rule associated with membrane h and having $f(h)$ on 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 1. *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 for reaching a configuration 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 $\#S^* = \max\{\#S \mid \exists$ 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 $f^*(h)$ on 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 labeled 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 $\#\mathcal{C}_E + \max\{\#S \mid \exists$ a crossing mapping $f : S \to supp(\mathcal{C}_C)\}$.

Next, we will prove that the maximum number of applications of rules $\beta_1(\Pi, \mathcal{C})$ is not greater than $\#\mathcal{C}_E + \max\{\#S \mid \exists$ 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 in the transition step $\mathcal{C} \Rightarrow \mathcal{C}_1$ is $\#T_C + \#T_E$. Now, let us split the multiset T_C into two multisets: T_C^e, the submultiset from T_C containing the elements which are LHS of some evolution rule, and T_C^{ne}, the submultiset from T_C containing the elements which are *not* 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 $T_E \cup T_C^e = \mathcal{C}_E$.
- $\#T_C^{ne} \leq \max\{\#S \mid \exists$ 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 $\#\mathcal{C}_E + \max\{\#S \mid \exists$ a crossing mapping $f : S \to supp(\mathcal{C}_C)\}$ and this concludes the proof. □

Due to the high computational cost of computing $\beta_1(\Pi, \mathcal{C})$, sometimes we would rather look for an upper bound easier to compute. The next corollary gives such an upper bound for $\beta_1(\Pi, \mathcal{C})$. The proof is immediate.

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

Rule 1: $[\,a\,]_1 \to b\,[\,]_1$, **Rule 5:** $z\,[\,]_1 \to [x]_1$, **Rule 8:** $[\,a \to b\,]_1$,
Rule 2: $[\,a\,]_2 \to b\,[\,]_2$, **Rule 6:** $z\,[\,]_2 \to [x]_2$, **Rule 9:** $[\,a \to b\,]_2$,
Rule 3: $[\,a\,]_3 \to b\,[\,]_3$, **Rule 7:** $z\,[\,]_3 \to [x]_3$, **Rule 10:** $[\,a \to b\,]_3$,
Rule 4: $[\,x\,]_1 \to b\,[\,]_1$.

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 1 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 The parallel communication case

In order to compute the number of rule applications, in this case we do not distinguish between evolution and communication rules.

Theorem 2. *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 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 is constant regardless of the non-determinism of the P system. Indeed, because of the maximal parallel condition applied 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 4. Let us consider again the P system from Example 3 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. It is worth to remark that this estimation is done *without* performing all the computations.

In order to obtain such an estimation, we start by building an extension of the notion of *dependency graph* of a P system, that will be called *labeled dependency graph*. Dependency graphs of P systems were presented in [3] as a tool for finding short paths in computation trees. Later, this notion was used as an efficient tool for studying the borderline of the tractability through P systems (see, e.g., [9, 10, 11]). Before going on, we shortly 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, $((z_1, m_1), (z_2, m_2))$, such that (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 Subsection 2.4).

Given a rule $r = (a_0, m_1) \rightarrow (a_1, m_2)(a_2, m_2) \ldots (a_n, m_2)$, for each (e, m_2) in the RHS of r, we will consider a directed labeled arc

$$(z_1, m_1) \xrightarrow{r/k} (e, m_2),$$

where k is the multiplicity of (e, m_2) in the RHS of rule r. Note that, given two nodes of the graph, there might exist several arcs connecting them with different labels [4].

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

We shall extract information from this labeled 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 upper bound of the number of simultaneous applications of rules 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 along the computation. The greater $\beta(\Pi, \mathcal{C})$ is, the *better* is the design, from the point of view of using the parallelism.

[4] Following [14], the definition of labeled dependency graph corresponds to an *edge-labeled, vertex-labeled, directed loop multigraph.*

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

4.1 The sequential communication case

Let us first consider a semantics where only *one* object can cross a membrane in a step of computation.

Theorem 3. *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 labeled dependency graph of Π generated by the source $\mathcal{C}$ and sink $\mathbb{T}$.*

1. *If there exists in $\mathcal{G}$ a cycle such that the labels r/k of its arcs verify the following:*
   - *all the rules r involved in the cycle are* evolution *rules,*
   - *at least one of the multiplicities k appearing in the cycle is greater than 1, then, there is no upper bound on the number of simultaneous applications of rules in one step starting from any configuration of Π reachable from $\mathcal{C}$.*
2. *Otherwise, $\beta(\Pi, \mathcal{C}) = (\#\mathcal{C}_E + \#\mathcal{C}_C) \cdot B_\Pi^{(\rho_{\Pi,\mathcal{C}})-1}$, where:*
   - *$\mathcal{B}_\Pi$ is the maximality cardinal of the RHS of the rules of Π.*
   - *$\rho_{\Pi,\mathcal{C}}$ is the maximum of the lengths of the paths in the labeled dependency graph of Π starting from an element of $\mathcal{C}$ and not containing duplicated nodes.*

Proof. Let us first consider the case where 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. Let $\psi = (V, E)$ be such a cycle, with $V = \{x_0, \ldots, x_n\}$ and $E = \{(x_i, x_{i+1}) \,|\, i \in \{0, \ldots, n-1\}\} \cup \{(x_n, x_0)\}$ and let K be the product of multiplicities in ψ, $K \geq 2$.

Consider now an element z_0 in the configuration $\mathcal{C}$ such that there exists a path, $\mathcal{P}$, in the labeled dependency graph starting in z_0 and ending on x_0. The existence of such z_0 is granted, since, by definition, $\mathcal{G}$ is the subgraph of the labeled 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, 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 an unbounded number of objects evolving. Thus, we conclude that there is no upper bound for the number of applications in a single transition step.

Notice that in the case when for every cycle we have that at least one of the rules involved in it is a communication rule, then such rule can be triggered *only once* for each transition step, regardless the number of objects that could trigger it. That is, although there may be an unbounded amount of objects appearing in the LHS of a communication rule, the number of simultaneous applications of any communication rule is always 1. In other words, the number of applications of rules in a single step inside the cycle is bounded, although the product of multiplicities in the arcs in the cycle was greater than 1.

Let us now consider the case where there is no cycle in $\mathcal{G}$ such that the product of the multiplicities of its arcs is greater than 1. The number of objects in the initial configuration that can trigger a rule is $\#\mathcal{C}_E+\#\mathcal{C}_C$. Therefore, the number of applications of rules in the first computation step is at most $\#\mathcal{C}_E+\#\mathcal{C}_C$. Let B_Π be the maximal cardinality of the RHS of the rules of Π; then, the number of objects in the second configuration that can trigger a rule 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 computation. 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 the $(n + 1)$-th step of *any* possible computation starting from $\mathcal{C}$.

Let us consider now the labeled dependency graph of the P system, and consider all the paths in this graph starting from objects in $\mathcal{C}$ and such that they do not contain duplicated nodes. Let us denote by $\rho_{\Pi,\mathcal{C}}$ the maximum of the lengths 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 the computation is $(\#C_E + \#C_C) \cdot B^{(\rho_{\Pi,\mathcal{C}})-1}$.

Note that the computation can be infinite in case that there exist cycles in $\mathcal{G}$, but as these cycles have all their arcs with multiplicity 1, such cycles do not cause an increase on the number of applications of rules. □

We illustrate this theorem 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 5. Let us consider the P system Π with alphabet $\Gamma = \{a, b, c\}$, membrane structure $\mu = [\,[\,]_1\,]_0$, and set of rules:

$$\mathbf{R1} : [\,a\,]_1 \to b[\,]_1, \qquad \mathbf{R2} : [\,b \to c^2\,]_0, \qquad \mathbf{R3} : c[\,]_1 \to [\,a\,]_1,$$

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

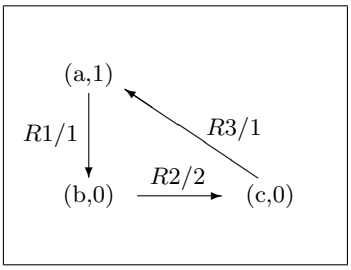

Fig. 2. The labeled dependency graph of the system in Example 5

Π is a non-deterministic P system. Let $[\,[\,a\,]_1\,]_0$ be a configuration of Π. A possible computation starting from $\mathcal{C}$ is:

From $[\,[\,a\,]_1\,]_0$ with *one* application of rule **R1** we reach $[\,[\,]_1\,b\,]_0$
From $[\,[\,]_1\,b\,]_0$ with *one* application of rule **R2** we reach $[\,[\,]_1\,c^2\,]_0$
From $[\,[\,]_1\,c^2\,]_0$ with *one* application of rule **R3** we reach $[\,[\,a\,]_1\,c\,]_0$
From $[\,[\,a\,]_1\,c\,]_0$ with *one* application of rule **R1** we reach $[\,[\,]_1\,bc\,]_0$
From $[\,[\,]_1\,bc\,]_0$ $\left\{\begin{array}{l}\text{with } one \text{ application of rule } \mathbf{R2}\\ \text{and } one \text{ application of rule } \mathbf{R3}\end{array}\right\}$ we reach $[\,[\,a\,]_1\,c^2\,]_0$
From $[\,[\,a\,]_1\,c^2\,]_0$ with *one* application of rule **R1** we reach $[\,[\,]_1\,bc^2\,]_0$
From $[\,[\,]_1\,bc^2\,]_0$ $\left\{\begin{array}{l}\text{with } one \text{ application of rule } \mathbf{R2}\\ \text{and } one \text{ application of rule } \mathbf{R3}\end{array}\right\}$ 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 to remark that we do not need to develop the computation tree in order to calculate this upper bound. Notice also that in this example, there exists a cycle in the labeled dependency graph such that one of the multiplicities appearing in it is greater than 1. However, we are in case 2 of Theorem 3, since there are communication rules involved in the cycle.

We have $\mathcal{C}_C = \{(a,1)\}$, $\mathcal{C}_E = \emptyset$, $B_\Pi = 2$, and $\rho_{\Pi,\mathcal{C}} = 2$, so from Theorem 3 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 $\beta(\Pi,\mathcal{C})$ can be reached in some cases.

Example 6. 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:

$\mathbf{R}_1$: $[\,x\,]_1 \to a_1\,[\,]_1$, $\mathbf{R}_3$: $[\,z \to a_1]_0$,
$\mathbf{R}_2$: $[\,y\,]_2 \to a_1\,[\,]_2$, $\mathbf{R}_{3+i}$: $[\,a_i \to a_{i+1}^2\,]_0$, with $i \in \{1,\ldots,k-1\}$,

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

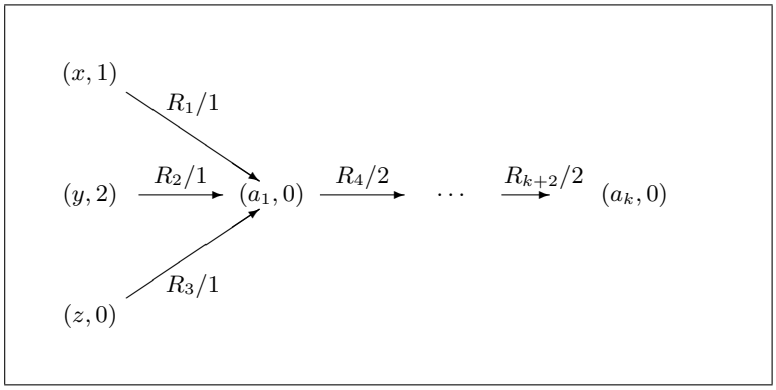

Fig. 3. The labeled dependency graph of the system in Example 6

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 objects a_1 in the membrane labeled 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 labeled by 0. Therefore, the maximum number of applications of rules in any step of computations 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 the theorem. Since $\mathcal{C}_E = \{(a_1, 0)\}$, $\mathcal{C}_C = \{(x, 1), (y, 2)\}$, $B_\Pi = 2$, and $\rho_{\Pi,\mathcal{C}} = k$, we have

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

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

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 The 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 4. *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 labeled dependency graph of Π generated by the source $\mathcal{C}$ and sink $\mathbb{T}$.*

1. *If there exists in $\mathcal{G}$ a cycle 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, then there is no upper bound on the number of applications from a configuration of Π reachable from $\mathcal{C}$.*
2. *Otherwise, $\beta(\Pi, \mathcal{C}) = (\#\mathcal{C}_E + \#\mathcal{C}_C) \cdot B_\Pi^{(\rho_{\Pi,\mathcal{C}})-1}$, where:*
   - *B_Π is the maximal cardinality of the RHS of the rules of Π.*
   - *$\rho_{\Pi,\mathcal{C}}$ is the maximum of the lengths of the paths in the labeled dependency graph of Π starting from an element of $\mathcal{C}$ and not containing duplicated nodes.*

Proof. The proof is similar to the proof of Theorem 3. □

Example 7. Let us consider again Example 5, with the same configuration $[\,[\,a\,]_1\,]_0$, but now under the parallel communication semantics. The labeled 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 4.

Example 6 also shows that the upper bound $\beta(\Pi, \mathcal{C})$ can be reached in the parallel communication case, since the behavior 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 extremely hard. These computational devices are too complex to be evaluated only with usual parameters *time* and *space*. Sevilla Carpets and its 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 make a first step in this direction. We introduce the concept of *injective mapping w.r.t. a multiset*, and we extend the definition of dependency graph to *labeled 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 use of parallelism in P systems.

We consider three 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.

Finally, 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 solutions to decision problems are usually carried out via uniform or semi-uniform 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 problem.

Acknowledgement

Work supported by project TIN2005-09345-C04-01 of Ministerio de Educación y Ciencia of Spain, cofinanced by FEDER funds.

References

1. G. Ciobanu, Gh. Păun, Gh. Ştefănescu: Sevilla carpets associated with P systems. In *Proceedings of the Brainstorming Week on Membrane Computing* (M. Cavaliere, C. Martín-Vide, Gh. Păun, eds.), Tarragona, Spain, 2003, Report RGML 26/03, 135–140.
2. G. Ciobanu, Gh. Păun, M.J. Pérez-Jiménez: On the branching complexity of P systems. *Fundamenta Informaticae*, 2006, in press.

3. A. Cordón-Franco, M.A. Gutiérrez-Naranjo, M.J. Pérez-Jiménez, A. Riscos-Núñez: Weak metrics on configurations of a P system. In *Proceedings of the Second Brainstorming Week on Membrane Computing* (Gh. Păun, A. Riscos-Núñez, A. Romero-Jiménez, F. Sancho-Caparrini, eds.), Report RGNC 01/04, University of Seville, 2004, 139–151.
4. A. Cordón-Franco, M.A. Gutiérrez-Naranjo, M.J. Pérez-Jiménez, A. Riscos-Núñez: Exploring computation trees associated with P systems. In *Membrane Computing*, LNCS 3365, Springer, 2005, 278–286.
5. A. Cordón-Franco, M.A. Gutiérrez-Naranjo, M.J. Pérez-Jiménez, A. Riscos-Núñez, F. Sancho-Caparrini: Cellular solutions of some numerical NP-complete problems: A Prolog implementation. In *Molecular Computational Models: Unconventional Approaches* (M. Gheorghe, ed.), Idea Group, Inc., London, 2005.
6. M.A. Gutiérrez-Naranjo, M.J. Pérez-Jiménez, A. Riscos-Núñez: On descriptive complexity of P systems. In *Membrane Computing*, LNCS 3365, Springer, 2005, 320–330.
7. 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.
8. M.A. Gutiérrez-Naranjo, M.J. Pérez-Jiménez, A. Riscos-Núñez: Multidimensional descriptional complexity of P systems. In *Proceedings of the 7th International Workshop on Descriptional Complexity of Formal Systems* (C. Mereghetti, B. Palano, G. Pighizzini, D. Wotschke, eds.), Como, Italy, 2005, 134–145.
9. M.A. Gutiérrez-Naranjo, M.J. Pérez-Jiménez, A. Riscos-Núñez, F. Romero-Campero: On the power of dissolution in P systems with active membranes. In *Membrane Computing*, LNCS 3850, Springer, 2005, 373–394.
10. M.A. Gutiérrez-Naranjo, M.J. Pérez-Jiménez, A. Riscos-Núñez, F. Romero-Campero: P systems with active membranes, without polarizations and with dissolution: A characterization of P. In *Unconventional Computation*, LNCS 3699, Springer, 2005, 105–116.
11. M.A. Gutiérrez-Naranjo, M.J. Pérez-Jiménez, A. Riscos-Núñez, F. Romero-Campero: Computational efficiency of dissolution rules in membrane systems. *International Journal of Computer Mathematics*, 2006, in press.
12. M.A. Gutiérrez-Naranjo, M.J. Pérez-Jiménez, F.J. Romero-Campero: Solving SAT with membrane creation. In *Computability in Europe 2005, CiE 2005: New Computational Paradigms* (S. Barry Cooper, B. Lowe, L. Torenvliet, eds.), Report ILLC X-2005-01, University of Amsterdam, 82–91.
13. M.A. Gutiérrez-Naranjo, M.J. Pérez-Jiménez, F.J. Romero-Campero: A linear solution of Subset Sum problem by using membrane creation. In *Mechanisms, Symbols and Models Underlying Using Cognition, First International Work-Conference on the Interplay between Natural and Artificial Computation, IWINAC 2005* (J. Mira, J.R. Alvarez, eds.), LNCS 3561, Springer, 2005, 258–267.
14. B. Mehdi, G. Chartrand: *Introduction to the Theory of Graphs*. Allyn and Bacon, Inc. Boston, 1971.
15. Gh. Păun: *Membrane Computing. An Introduction*. Springer-Verlag, Berlin, 2002.
16. M.J. Pérez-Jiménez, A. Riscos-Núñez: Solving the Subset-Sum problem by active membranes. *New Generation Computing*, 23, 4 (2005), 367–384.
17. M.J. Pérez-Jiménez, F.J. Romero-Campero: Solving the Bin Packing problem by recognizer P systems with active membranes. In *Proceedings of the Second Brainstorming Week on Membrane Computing* (Gh. Păun, A. Riscos-Núñez, A. Romero-Jiménez, F. Sancho-Caparrini, eds.), Report RGNC 01/04, University of Seville, 2004, 414–430.

18. M.J. Pérez-Jiménez, A. Romero-Jiménez, F. Sancho-Caparrini: A polynomial complexity class in P systems using membrane division. In *Proceedings of the 5th Workshop on Descriptional Complexity of Formal Systems, DCFS 2003* (E. Csuhaj-Varjú, C. Kintala, D. Wotschke, Gy. Vaszyl, eds.), Budapest, 2003, 284–294.
19. M.J. Pérez-Jiménez, A. Romero-Jiménez, F. Sancho-Caparrini: Solving VALIDITY problem by active membranes with input. In *Proceedings of the Brainstorming Week on Membrane Computing* (M. Cavaliere, C. Martín-Vide, Gh. Păun, eds.), Tarragona, Spain, 2003, Report RGML 26/03, 279–290.
20. A. Riscos-Núñez: *Cellular Programming: Efficient Resolution of Numerical NP-complete Problems*. Ph.D. Thesis, University of Seville, 2004.
21. A. Syropoulos: Mathematics of multisets. In *Multisets Processing*, LNCS 2235, Springer, 2001, 347–358.