Limits on P Systems with Proteins and Without Division

David Orellana-Martín, Luis Valencia-Cabrera, Agustín Riscos-Núñez,
Mario J. Pérez-Jiménez

Research Group on Natural Computing,
Department of Computer Science and Artificial Intelligence,
Universidad de Sevilla, Avda. Reina Mercedes s/n, 41012, Sevilla, Spain
E-mail: {dorellana, ariscosn, lvalencia, marper}@us.es

Summary. In the field of Membrane Computing, computational complexity theory has been widely studied trying to find frontiers of efficiency by means of syntactic or semantical ingredients. The objective of this is to find two kinds of systems, one non-efficient and another one, at least, *presumably* efficient, that is, that can solve **NP**-complete problems in polynomial time, and adapt a solution of such a problem in the former. If it is possible, then $\mathbf{P} = \mathbf{NP}$. Several borderlines have been defined, and new characterizations of different types of membrane systems have been published.

In this work, a certain type of P system, where proteins act as a supporting element for a rule to be fired, is studied. In particular, while division rules, the abstraction of cellular *mitosis* is forbidden, only problems from class **P** can be solved, in contrast to the result obtained allowing them.

Key Words: Membrane Computing, active membranes, proteins, computational complexity theory

1 Introduction

In the beginning, *Membrane Computing* was developed mainly to study certain fields of theoretical computer science, such as formal language theory and computability theory, from a different perspective [12]. In this framework, several models of *P systems*, the main computational device within this framework, have been demonstrated to be universal. For this purpose, the most used technique is to simulate another computationally complete machine, like Turing machines [8] or register machines [2].

Even if it started in this field, the *Membrane Computing* grew rapidly into another fields, such as computational complexity theory [15], biology [16], ecology [5], electrical networks [13] and a wide range of real-life applications [7, 10, 21]. For each field, different variants of *membrane systems* have been developed.

Computational complexity theory is devoted to classify classes of problems depending on their intrinsic complexity, in contrast with algorithms theory, where the complexity measured is that of the algorithm itself. This classification is based on the *resources needed* to solve *efficiently* a problem. In an informal way, we say that a problem is efficiently solved by a machine if the time that this machine takes to solve an instance of length n of such problem is upper bounded by a polynomial $p(n)$.

One of the seven Millenium Prize Problems is the so-called **P** vs. **NP** problem, whose response, whether positive or negative, would have a major impact in several fields such as cryptography, economics, proof theory, even in biology [3]. That is why it seems interesting to study it from a bio-inspired perspective. On the one hand, to demonstrate that a class of P systems is *presumably* efficient it is enough to give an efficient solution to a **NP**-complete problem. On the other hand, various techniques have been developed to show that a class of such devices can only efficiently solve problems from class **P**, such as the *dependency graph technique*, where a directed graph based on the behavior of the system is created from its definition and its resolution is characterized by the `REACHABILITY` problem [1]; the *algorithmic technique*, where an algorithm $\mathcal{A}$ working in polynomial time has as input a *recognizer* P system Π and a multiset m and reproduces a computation of $\Pi + m$ [2]; and the *simulation technique*, where a *recognizer* membrane system is simulated by means of another kind. By this, if Π is a recognizer P system that can solve efficiently problems from the complexity class $\mathcal{C}$, and Π' is another kind of recognizer P system, usually with less "ingredients" than the former, if Π' can simulate Π, then Π' can solve problems from class $\mathcal{C}$. In this work, we use the algorithmic technique to prove that a certain type of P systems cannot solve **NP**-complete problems (unless, of course, $\mathbf{P} = \mathbf{NP}$).

The paper is organized as follows: first of all, in order to make this work self-contained, we introduce some preliminary concepts. Section 3 is devoted to introduce P systems with proteins on membranes. In Section 4, a solution to the open problem from [18] is given, using the algorithmic technique for this purpose. Finally, some conclusions and future research lines are given.

2 Preliminaries

Here, we introduce some concepts that are going to be used through the work.

2.1 Alphabets and multisets

An *alphabet* Γ is a non-empty set and its elements are called *symbols*. A *string* u over Γ is an ordered finite sequence of symbols, that is, a mapping from a natural

[1] It is a problem from class **P**, so its solution can be found in polynomial time.

[2] Let us recall that a *recognizer* P system with a given input is confluent, that is, all its computations return the same answer, so it is enough to reproduce one of them. For a formal definition of *recognizer* membrane systems we refer to [15, 14]

number $n \in \mathbb{N}$ onto Γ. The number n is called the *length* of the string u and it is denoted by $|u|$. The empty string (with length 0) is denoted by λ. The set of all strings over an alphabet Γ is denoted by Γ^*. A *language* over Γ is a subset of Γ^*.

A *multiset* over an alphabet Γ is an ordered pair (Γ, f) where f is a mapping from Γ onto the set of natural numbers $\mathbb{N}$. The *support* of a multiset $m = (\Gamma, f)$ is defined as $supp(m) = \{x \in \Gamma \mid f(x) > 0\}$. A multiset is finite (respectively, empty) if its support is a finite (respectively, empty) set. We denote by $\emptyset$ the empty multiset. Let $m_1 = (\Gamma, f_1)$, $m_2 = (\Gamma, f_2)$ be multisets over Γ, then the union of m_1 and m_2, denoted by $m_1 + m_2$, is the multiset (Γ, g), where $g(x) = f_1(x) + f_2(x)$ for each $x \in \Gamma$. We denote by $M_f(\Gamma)$ the set of all multisets over Γ.

2.2 Graphs and trees

Let us recall some notions related with graph theory (see [6] for details). An *undirected graph* is an ordered pair (V, E) where V is a set whose elements are called nodes or vertices and $E = \{\{x, y\} \mid x \in V, y \in V, x \neq y\}$ whose elements are called *edges*. A *path* of length $k \geq 1$ from a node u to a node v in a graph (V, E) is a finite sequence $(x_0, x_1, \ldots, x_k)$ of nodes such that $x_0 = u$, $x_k = v$ and $\{x_i, x_{i+1}\} \in E$. If $k \geq 2$ and $x_0 = x_k$ then we say that the path is a *cycle* of the graph. A graph with no cycle is said to be *acyclic*. An undirected graph is *connected* if there exist paths between every pair of nodes.

A *rooted tree* is a a connected, acyclic, undirected graph in which one of the vertices (called *the root of the tree*) is distinguished from the others. Given a node x (different from the root), if the last edge on the (unique) path from the root of the tree to the node x is $\{x, y\}$ (in this case, $x \neq y$), then y is **the** *parent* of node x and x is **a** *child* of node y. The root is the only node in the tree with no parent. A node with no children is called a *leaf*.

3 P systems with Proteins on Membranes

The inspiration comes from the biochemistry of living cells, where proteins take part regulating which reactions occur depending on whether certain proteins are present or not [1]. P systems with proteins on membranes, first introduced in [11], have been demonstrated to be universal devices [4], as well as able to solve computationally hard problems. In fact, in [17], a uniform solution to `QSAT` is given by means of a family of P systems with proteins on membranes and membrane division. Here, we define the syntax and semantics of these systems by adding the necessary elements to introduce cell separation as a method to create an exponential workspace in terms of cells, as it has been used in other variants of P systems [9, 20].

3.1 Syntax

Definition 1. *A P system with proteins on membranes and membrane division of degree $q \geq 1$ is a tuple*

$$\Pi = (\Gamma, \Gamma_0, \Gamma_1, P, P_0, P_1, \mathcal{E}, \mu, \mathcal{M}_1/\mathcal{Z}_1, \ldots, \mathcal{M}_q/\mathcal{Z}_q, \mathcal{R}_1, \ldots, \mathcal{R}_q, i_{out})$$

where:

- Γ *and* P *are finite multisets with* $\Gamma \cap P = \emptyset$, *and* $\mathcal{E} \subseteq \Gamma \setminus P$;
- $\{\Gamma_0, \Gamma_1\}$ *is a partition of the set* Γ, *where* $\Gamma_0 \cup \Gamma_1 = \Gamma$, $\Gamma_0 \cap \Gamma_1 = \emptyset$ *and* Γ_0, Γ_1 *are non-empty sets if separation rules are used,* $\Gamma_0 = \Gamma_1 = \emptyset$ *otherwise;*
- $\{P_0, P_1\}$ *is a partition of the set* P, *where* $P_0 \cup P_1 = P$, $P_0 \cap P_1 = \emptyset$ *and* P_0, P_1 *are non-empty sets if separation rules are used,* $P_0 = P_1 = \emptyset$ *otherwise;*
- μ *is a rooted tree;*
- $\mathcal{M}_1, \ldots, \mathcal{M}_q$ *are multisets over* Γ;
- $\mathcal{Z}_1, \ldots, \mathcal{Z}_q$ *are multisets over* P;
- $\mathcal{R}_1, \ldots, \mathcal{R}_q$ *are finite sets of rules associated with the nodes of the graph of the following forms:*
  (1) $[\,p\,|\,a\,]_i \to [\,p'\,|\,b\,]_i, p, p' \in P, a, b \in \Gamma, 1 \leq i \leq q$ *(in-membrane object evolution rules);*
  (2) $a\,[\,p\,|\,]_i \to b\,[\,p'\,|\,]_i, p, p' \in P, a, b \in \Gamma, 1 \leq i \leq q$ *(out-membrane object evolution rules);*
  (3) $[\,p\,|\,a\,]_i \to b\,[\,p'\,|\,]_i, p, p' \in P, a, b \in \Gamma, 1 \leq i \leq q$ *(send-out communication rules);*
  (4) $a\,[\,p\,|\,]_i \to [\,p'\,|\,b\,]_i, p, p' \in P, a, b \in \Gamma, 1 \leq i \leq q$ *(send-in communication rules);*
  (5) $a\,[\,p\,|\,b\,]_i \to c\,[\,p'\,|\,d\,]_i, p, p' \in P, a, b, c, d \in \Gamma, 1 \leq i \leq q$ *(antiport communication rules);*
  (6_p) $[\,p\,|\,]_i \to [\,p'\,|\,]_i[\,p''\,|\,]_i, p, p', p'' \in P, 1 \leq i \leq q, i \neq i_{out}$ *(protein-based division rules)*
  (6_o) $[\,|\,a\,]_i \to [\,|\,b\,]_i[\,|\,c\,]_i, a, b, c \in \Gamma, 1 \leq i \leq q, i \neq i_{out}$ *(object-based division rules)*
  ($6'_p$) $[\,p\,|\,]_i \to [\,p'\,|\,]_i[\,p''\,|\,]_i, p, p', p'' \in P, 1 \leq i \leq q, i \neq i_{out}$ *(protein-based division rules)*
  ($6'_o$) $[\,|\,a\,]_i \to [\,|\,b\,]_i[\,|\,c\,]_i, a, b, c \in \Gamma, 1 \leq i \leq q, i \neq i_{out}$ *(object-based division rules)*
- $i_{out} = 0$ *is the output membrane.*

A P system with proteins on membranes and membrane division (respectively, membrane separation) of degree $q \geq 1$,

$$\Pi = (\Gamma, \Gamma_0, \Gamma_1, P, P_0, P_1, \mathcal{E}, \mu, \mathcal{M}_1/\mathcal{Z}_1, \ldots, \mathcal{M}_q/\mathcal{Z}_q, \mathcal{R}_1, \ldots, \mathcal{R}_q, i_{out})\ ^{3},$$

[3] Let us note that Γ_0, Γ_1, P_0 and P_1 are usually ommited when separation rules are not used

can be viewed as a set of q membranes, labelled by $1, \ldots, q$ arranged in a hierarchical structure μ given by a rooted tree whose root is called the *skin membrane*, such that: (a) $\mathcal{M}_1, \ldots, \mathcal{M}_q$ represent the multisets of objects initially placed in the q membranes of the system; (b) $\mathcal{Z}_1, \ldots, \mathcal{Z}_q$ represent the multisets of proteins initially placed in the q membranes of the system; (c) $\mathcal{E}$ is the set of objects initially located in the environment of the system, all of them available in an arbitrary number of copies; and (d) i_{out} represent a distinguished *region* which will encode the output of the system. We use the term *region* i ($0 \leq i \leq q$) to refer to membrane i in the case $1 \leq i \leq q$ and to refer to the environment in the case $i = 0$.

A *configuration* at any instant of such kind of P system is described by the membrane structure of the system, the multisets of objects in each membrane, the multisets of proteins in each membrane and the multiset of objects over $\Gamma \setminus \mathcal{E}$ in the environment at the moment. The initial configuration of $\Pi = (\Gamma, \Gamma_0, \Gamma_1, P, P_0, P_1, \mathcal{E}, \mu, \mathcal{M}_1/\mathcal{Z}_1, \ldots, \mathcal{M}_q/\mathcal{Z}_q, \mathcal{R}_1, \ldots, \mathcal{R}_q, i_{out})$ is $(\mu, \mathcal{M}_1/\mathcal{Z}_1, \ldots, \mathcal{M}_q/\mathcal{Z}_q; \emptyset)$.

3.2 Semantics

An in-membrane object evolution rule $[\,p\,|\,a\,]_i \rightarrow [\,p'\,|\,b\,]_i \in \mathcal{R}_i$ is *applicable* at a configuration $\mathcal{C}_t$ at an instant t if there is a region i from $\mathcal{C}_t$ which contains the object a and the protein p. By applying such rule, object a and protein p in region i from $\mathcal{C}_t$ are consumed and object b and protein p' are generated in region i from $\mathcal{C}_{t+1}$.

An out-membrane object evolution rule $a\,[\,p\,|\,]_i \rightarrow b\,[\,p'\,|\,]_i \in \mathcal{R}_i$ is *applicable* at a configuration $\mathcal{C}_t$ at an instant t if there is a region $p(i)$ from $\mathcal{C}_t$ which contains the object a and a region i from $\mathcal{C}_t$ which contains the protein p. By applying such rule, object a in region $p(i)$ and protein p in region i from $\mathcal{C}_t$ are consumed and object b is generated in region $p(i)$ and protein p' is generated in region i from $\mathcal{C}_{t+1}$.

A send-out communication rule $[\,p\,|\,a\,]_i \rightarrow b\,[\,p'\,|\,]_i \in \mathcal{R}_i$ is *applicable* at a configuration $\mathcal{C}_t$ at an instant t if there is a region i from $\mathcal{C}_t$ which contains the object a and the protein p. By applying such rule, object a and protein p in region i from $\mathcal{C}_t$ are consumed and object b is generated in region $p(i)$ and protein p' is generated in region i from $\mathcal{C}_{t+1}$.

A send-in communication rule $a\,[\,p\,|\,]_i \rightarrow [\,p'\,|\,b\,]_i \in \mathcal{R}_i$ is *applicable* at a configuration $\mathcal{C}_t$ at an instant t if there is a region $p(i)$ from $\mathcal{C}_t$ which contains the object a and a region i from $\mathcal{C}_t$ which contains the protein p. By applying such rule, object a in region $p(i)$ and protein p in region i from $\mathcal{C}_t$ are consumed and object b and protein p' are generated in region i from $\mathcal{C}_{t+1}$.

An antiport communication rule $a\,[\,p\,|\,b\,]_i \rightarrow c\,[\,p'\,|\,d\,]_i \in \mathcal{R}_i$ is *applicable* at a configuration $\mathcal{C}_t$ at an instant t if there is a region $p(i)$ from $\mathcal{C}_t$ which contains the object a and a region i from $\mathcal{C}_t$ which contains the object b and the protein p. By applying such rule, object a in region $p(i)$ and object b and protein p in region i

from $\mathcal{C}_t$ are consumed and object c is generated in region $p(i)$ and object d and protein p' are generated in region i from $\mathcal{C}_{t+1}$.

A protein-based division rule $[\,p\,|\,]_i \to [\,p'\,|\,]_i[\,p''\,|\,]_i \in \mathcal{R}_i$ is *applicable* at a configuration $\mathcal{C}_t$ at an instant t if there is a region i from $\mathcal{C}_t$ which contains the protein p. By applying such rule, protein p in region i from $\mathcal{C}_t$ is consumed, two new membranes with label i are generated at configuration $\mathcal{C}_{t+1}$ and objects and proteins from the original membrane are duplicated in both new membranes, except protein p that evolves in a protein p' that goes to one of the new membranes, and a protein p'' that goes to the other one.

An object-based division rule $[\,|\,a\,]_i \to [\,|\,b\,]_i[\,|\,c\,]_i \in \mathcal{R}_i$ is *applicable* at a configuration $\mathcal{C}_t$ at an instant t if there is a region i from $\mathcal{C}_t$ which contains the object a. By applying such rule, object a in region i from $\mathcal{C}_t$ is consumed, two new membranes with label i are generated at configuration $\mathcal{C}_{t+1}$ and objects and proteins from the original membrane are duplicated in both new membranes, except object a that evolves in an object b that goes to one of the new membranes, and an object c that goes to the other one.

A protein-based separation rule $[\,p\,|\,]_i \to [\,P_0\,|\,\Gamma_0\,]_i[\,P_1\,|\,\Gamma_1\,]_i \in \mathcal{R}_i$ is *applicable* at a configuration $\mathcal{C}_t$ at an instant t if there is a region i from $\mathcal{C}_t$ which contains the protein p. By applying such rule, protein p in region i from $\mathcal{C}_t$ is consumed, two new membranes with label i are generated at configuration $\mathcal{C}_{t+1}$ and objects and proteins from the original membrane are distributed in both new membranes, proteins in P_0 and objects in Γ_0 go to one of the new membranes and proteins in P_1 and objects in Γ_1 go to the other one.

An object-based separation rule $[\,|\,a\,]_i \to [\,P_0\,|\,\Gamma_0\,]_i[\,P_1\,|\,\Gamma_1\,]_i \in \mathcal{R}_i$ is *applicable* at a configuration $\mathcal{C}_t$ at an instant t if there is a region i from $\mathcal{C}_t$ which contains the object a. By applying such rule, object a in region i from $\mathcal{C}_t$ is consumed, two new membranes with label i are generated at configuration $\mathcal{C}_{t+1}$ and objects and proteins from the original membrane are distributed in both new membranes, proteins in P_0 and objects in Γ_0 go to one of the new membranes and proteins in P_1 and objects in Γ_1 go to the other one.

It makes no sense in this kind of systems to define the concept *length*, because all the rules have a fixed amount of objects involved in them.

The rules of these systems are applied in a maximally parallel manner, and we have the restriction that when a membrane i is divided or separated at one transition step, then no other rules can be applied for that membrane i at that step.

A transition from a configuration $\mathcal{C}_t$ to another configuration $\mathcal{C}_{t+1}$ is obtained by applying rules in a maximally parallel manner following the previous remarks. A *computation* of the system is a (finite or infinite) sequence of transitions starting from the initial configuration, where any term of the sequence other than the first, is obtained from the previous configuration in one transition step, and it is denoted by $\mathcal{C}_t \Rightarrow_\Pi \mathcal{C}_{t+1}$. If the sequence is finite (called *halting computation*) then the last term of the sequence is a *halting configuration*, that is, a configuration where no rule is applicable to it. A computation gives a result only when an halting

configuration is reached, and that result is encoded by the multiset of objects present in the output region i_{out}. A natural framework to solve decision problems is to use recognizer P systems.

Definition 2. *A recognizer P system with proteins on membranes and membrane division/separation of degree $q \geq 1$ is a tuple*

$$\Pi = (\Gamma, \Gamma_0, \Gamma_1, P, P_0, P_1, \Sigma, \mathcal{E}, \mu, \mathcal{M}_1/\mathcal{Z}_1, \ldots, \mathcal{M}_q/\mathcal{Z}_q, \mathcal{R}_1, \ldots, \mathcal{R}_q, i_{in}, i_{out})$$

where:

- *The tuple $\Pi = (\Gamma, \Gamma_0, \Gamma_1, P, P_0, P_1, \mathcal{E}, \mu, \mathcal{M}_1/\mathcal{Z}_1, \ldots, \mathcal{M}_q/\mathcal{Z}_q, \mathcal{R}_1, \ldots, \mathcal{R}_q, i_{out})$ is a P system with proteins on membranes and membrane division/separation of degree $q \geq 1$, where Γ strictly contains an (input) alphabet Σ and two distinguished objects* `yes` *and* `no`*, and $\mathcal{M}_i$ ($1 \leq i \leq q$) are multisets over $\Gamma \setminus \Sigma$;*
- *$i_{in} \in \{1, \ldots, q\}$ is the input membrane and i_{out} is the label of the environment;*
- *for each multiset m over the input alphabet Σ, any computation of the system Π with input m starts from the configuration $(\mathcal{M}_1, \ldots, \mathcal{M}_{i_{in}} + m, \ldots, \mathcal{M}_q; \emptyset)$, always halts and either object* `yes` *ir object* `no` *(but not both) must appear in the environment at the last step.*

Next, we define the concept of solving a problem in a uniform way and in polynomial time by a family of recognizer P systems with proteins on membranes and membrane division/separation.

Definition 3. *A decision problem $X = (I_X, \theta_X)$ is solvable in a* uniform way *and in polynomial time by a family $\mathbf{\Pi} = \{\Pi(n) | n \in \mathbb{N}\}$ of recognizer P systems with proteins on membranes and membrane division/separation. if the following conditions hold:*

- *the family $\mathbf{\Pi}$ is polynomially uniform by Turing machines;*
- *there exists a polynomial encoding (cod, s) of I_X such that: (a) for each instance $u \in I_X$, $s(u)$ is a natural number and $cod(u)$ is an* input multiset *of the system $\Pi(s(u))$; (b) for each $n \in \mathbb{N}$, $s^{-1}(n)$ is a finite set; and (c) the family $\mathbf{\Pi}$ is polynomially bounded, sound and complete with regard to (X, cod, s).*

4 Limits of P systems with proteins on membranes when division rules are not allowed

In [18], an open problem in this framework is given:

Open Problem 5: What is the computational power of families of these P systems without membrane division? Do they characterize the class **P***, and what happens under various restrictions on the form of the rules?*

Here, we obtain a stronger result dealing with P systems with proteins on membranes and membrane separation.

4.1 Representation of P systems with proteins on membranes and membrane separation

Let $\Pi = (\Gamma, \Gamma_0, \Gamma_1, P, P_0, P_1, \Sigma, \mathcal{E}, \mu, \mathcal{M}_1/\mathcal{Z}_1, \ldots, \mathcal{M}_q/\mathcal{Z}_q, \mathcal{R}_1, \ldots, \mathcal{R}_q, i_{in}, i_{out})$ be a recognizer P system with proteins on membranes and membrane separation. We denote by $\mathcal{R}_E$ (resp., $\mathcal{R}_S$) the set of evolution and communication rules (resp., separation rules) of Π. We will fix a total order in $\mathcal{R}_E$ and a total order in $\mathcal{R}_S$. Because several membranes with the same label are generated by using separation rules, in order to identify the different membranes with the same label, the following recursive definition is used to modify the labels of the new generated membranes:

- We denote the label of a membrane as a pair (i, σ), where $1 \leq i \leq q$ and $\sigma \in \{0,1\}^*$ is a binary string.
- If a separation rule is applied to a membrane with label (i, σ), then the new created membranes will be labelled by $(i, \sigma 0)$ and $(i, \sigma 1)$, respectively. We mention that for the system during any computation, we consider a lexicographical order over the set of labels of membranes.

Note that if evolution or communication rules occur between membranes, the labels of these do not change.

A configuration at an instant t of such kind of P system is described by the multisets of objects over Γ contained in each membrane and the multiset of objects over $\Gamma \setminus \mathcal{E}$ in the environment. Hence, a configuration of Π can be described as follows:

$$\{(a, i, \sigma) \mid a \in \Gamma \cup \{\lambda\}, 1 \leq i \leq q, \sigma \in \{0,1\}^*\} \cup \{(a, 0) \mid a \in \Gamma \setminus \mathcal{E}\}$$

We use LHS and RHS to refer to the left-hand side and the right-hand side of a rule. They are defined in a natural way according to the definition of a rule:

- $[\, p \,|\, a \,]_i \rightarrow [\, p' \,|\, b \,]_i \in \mathcal{R}_i$, with $p, p' \in P$ and $a, b \in \Gamma$. Then, we denote by $n \cdot LHS(r, (i, \sigma_i)) = (p, i, \sigma_i)^n (a, i, \sigma_i)^n$, and by $n \cdot RHS(r, (i, \sigma_i)) = (p', i, \sigma_i)^n (b, i, \sigma_i)^n$.
- $r \equiv a\,[\, p \,|\,]_i \rightarrow b\,[\, p' \,|\,]_i \in \mathcal{R}_i$, with $p, p' \in P$ and $a, b \in \Gamma$. Then, we denote by $n \cdot LHS(r, (i, \sigma_i)) = (p, i, \sigma_i)^n (a, j, \sigma_j)^n$, and by $n \cdot RHS(r, (i, \sigma_i)) = (p', i, \sigma_i)^n (b, j, \sigma_j)^n$, where (j, σ_j) is the parent membrane of the membrane (i, σ_i).
- $r \equiv [\, p \,|\, a \,]_i \rightarrow b\,[\, p' \,|\,]_i \in \mathcal{R}_i$, with $p, p' \in P$ and $a, b \in \Gamma$. Then, we denote by $n \cdot LHS(r, (i, \sigma_i)) = (p, i, \sigma_i)^n (a, i, \sigma_i)^n$, and by $n \cdot RHS(r, (i, \sigma_i)) = (p', i, \sigma_i)^n (b, j, \sigma_j)^n$, where (j, σ_j) is the parent membrane of the membrane (i, σ_i).
- $r \equiv a\,[\, p \,|\,]_i \rightarrow [\, p' \,|\, b \,]_i \in \mathcal{R}_i$, with $p, p' \in P$ and $a, b \in \Gamma$. Then, we denote by $n \cdot LHS(r, (i, \sigma_i)) = (p, i, \sigma_i)^n (a, j, \sigma_j)^n$, and by $n \cdot RHS(r, (i, \sigma_i)) = (p', i, \sigma_i)^n (b, i, \sigma_i)^n$, where (j, σ_j) is the parent membrane of the membrane (i, σ_i).

- $r \equiv a\,[\,p\,|\,b\,]_i \rightarrow c\,[\,p'\,|\,d\,]_i \in \mathcal{R}_i$, with $p, p' \in P$ and $a, b, c, d \in \Gamma$. Then, we denote by $n \cdot LHS(r,(i,\sigma_i)) = (p,i,\sigma_i)^n(a,j,\sigma_j)^n(b,i,\sigma_i)^n$, and by $n \cdot RHS(r,(i,\sigma_i)) = (p',i,\sigma_i)^n(c,j,\sigma_j)^n(d,i,\sigma_i)^n$, where (j,σ_j) is the parent membrane of the membrane (i,σ_i).
- $r \equiv [\,p\,|\,]_i \rightarrow [\,P_0\,|\,\Gamma_0\,]_i[\,P_1\,|\,\Gamma_1\,]_i \in \mathcal{R}_i$, with $p \in P$. Then, we denote by $LHS(r,(i,\sigma_i)) = (p,i,\sigma_i)$.
- $r \equiv [\,|\,a\,]_i \rightarrow [\,P_0\,|\,\Gamma_0\,]_i[\,P_1\,|\,\Gamma_1\,]_i \in \mathcal{R}_i$, with $a \in \Gamma$. Then, we denote by $LHS(r,(i,\sigma_i)) = (a,i,\sigma_i)$.

If $\mathcal{C}_t$ is a configuration of Π, then the multiset obtained by replacing in $\mathcal{C}_t$ every occurrence of (x,i,σ) by (x,i,σ') is denoted by $\mathcal{C}_t + \{(x,i,\sigma)/\sigma'\}$. Moreover, we denote by $\mathcal{C}_t + m$ (resp., $\mathcal{C}_t \setminus m$) a multiset m of labelled objects addition to (resp., removal from) the configuration $\mathcal{C}_t$.

Next, we show that P systems with proteins on membranes and membrane separation can only solve tractable problems.

If $\mathcal{C} = (\mathcal{C}_0, \mathcal{C}_1, \ldots, \mathcal{C}_n)$ is a halting computation, then we denote by $|\mathcal{C}| = n+1$ the length of $\mathcal{C}$. For each i $(1 \leq i \leq q)$, the multiset of objects over Γ contained in all membranes labelled by i at configuration $\mathcal{C}_t$ is denoted by $\mathcal{C}_{t,o}(i)$, and the multiset of proteins over P contained in all membranes labelled by i at configuration $\mathcal{C}_t$ by $\mathcal{C}_{t,p}(i)$. We denote by $\mathcal{C}_t(0)$ the multiset of objects over $\Gamma \setminus \mathcal{E}$ contained in the environment at configuration $\mathcal{C}_t$. We define in a natural way $\mathcal{C}^*_{t,o} = \mathcal{C}_t(0) + \mathcal{C}_{t,o}(1) + \cdots + \mathcal{C}_{t,o}(q)$ and $\mathcal{C}^*_{t,p} = \mathcal{C}_{t,p}(1) + \cdots + \mathcal{C}_{t,p}(q)$. Finally, the finite multiset $\mathcal{C}_t(0) + \mathcal{C}_{t,o}(1) + \mathcal{C}_{t,p}(1) + \cdots + \mathcal{C}_{t,o}(q) + \mathcal{C}_{t,p}(q) = \mathcal{C}^*_{t,o} + \mathcal{C}^*_{t,p}$ is denoted by $\mathcal{C}^*_t$.

Lemma 1. *Let Π be a recognizer P system with proteins on membranes and membrane separation. Let $M = |\mathcal{M}_1 + \cdots + \mathcal{M}_q|$, $Z = |\mathcal{Z}_1 + \cdots + \mathcal{Z}_q|$ and let $\mathcal{C} = (\mathcal{C}_0, \ldots, \mathcal{C}_n)$ be a computation of Π. Then, we have*

1. *$|\mathcal{C}^*_0| = M + Z = S$, and for each t, $0 \leq t \leq n-1$, $|\mathcal{C}^*_{t+1}| \leq |\mathcal{C}^*_t| + Z$;*
2. *for each t, $0 \leq t \leq n$, $|\mathcal{C}^*_t| \leq S + t \cdot Z$; and*
3. *the number of created membranes along the computation $\mathcal{C}$ by the application of membrane separation rules is bounded by $2M + (2+n)Z$.*

Proof. (1) Let us notice that $|\mathcal{C}^*_0| = |\mathcal{C}_0(0) + \mathcal{C}_0(1) + \cdots + \mathcal{C}_0(q)| = |\mathcal{M}_1 + \cdots + \mathcal{M}_q| + |\mathcal{Z}_1 + \cdots + \mathcal{Z}_q| = M + Z = S$. Let Π be a recognizer P system with proteins on membranes and membrane separation, $\mathcal{R}_1, \ldots, \mathcal{R}_q$ be the sets of rules associated with Π, which contains the following types of communication rules:

- $[\,p\,|\,a\,]_i \rightarrow [\,p'\,|\,b\,]_i \in \mathcal{R}_i, p, p' \in P, a, b \in \Gamma, 1 \leq i \leq q$.
- $a\,[\,p\,|\,]_i \rightarrow b\,[\,p'\,|\,]_i \in \mathcal{R}_i, p, p' \in P, a, b \in \Gamma, 1 \leq i \leq q$.
- $[\,p\,|\,a\,]_i \rightarrow b\,[\,p'\,|\,]_i \in \mathcal{R}_i, p, p' \in P, a, b \in \Gamma, 1 \leq i \leq q$.
- $a\,[\,p\,|\,]_i \rightarrow [\,p'\,|\,b\,]_i \in \mathcal{R}_i, p, p' \in P, a, b \in \Gamma, 1 \leq i \leq q$.
- $a\,[\,p\,|\,b\,]_i \rightarrow c\,[\,p'\,|\,d\,]_i \in \mathcal{R}_i, p, p' \in P, a, b, c, d \in \Gamma, 1 \leq i \leq q$.

For each t, $0 \leq t \leq n-1$, in the transition from configuration $\mathcal{C}_t$ to configuration $\mathcal{C}_{t+1}$, by using any rule, at least one object and one protein from $\mathcal{C}_t$ is consumed

and at most one object and one protein is produced in $\mathcal{C}_{t+1}$. Let us note that by the use of out-membrane object evolution rules, send-in communication rules and antiport communication rules, $p \neq p'$ if $i = i_{skin}$. If $a \in \mathcal{E}$, then a new object is created in the system. So the number of objects created in a single computation step is bounded by Z, that is, the number of proteins present in the system in configuration $\mathcal{C}_t$. By means of separation rules, neither new objects nor proteins are going to appear. Hence, in any transition step the number of objects in the system is increased at most by Z new objects.

(2) By induction on t. Let us start analyzing the basic case $t = 0$. The result is trivial because of $|\mathcal{C}_0^*| = S$. By induction hypothesis, let us suppose the result holds for t, $0 \leq t \leq n-1$. Then $|\mathcal{C}_{t+1}^*| \geq |\mathcal{C}_t^*| + Z$, that is true because of (1), and by induction hypothesis we know that $|\mathcal{C}_t^*| \leq S + t \cdot Z$, so $|\mathcal{C}_{t+1}^*| \leq |\mathcal{C}_t^*| + Z \leq S + t \cdot Z + Z = S + (t+1) \cdot Z$. Hence, the result is also true for $t+1$.

(3) According to the fact that the application of a separation rule consumes an object and produces two new cells, result (3) can be obtained from (2) easily, since the maximum number of separation rules that can be performed in this kind of systems comes defined by the initial multisets of elements. If new objects are created as explained in (2), then at most $n \cdot Z$ new objects can be created in n computation steps, therefore at most $n \cdot Z$ separation rules can be applied by means of these objects. □

Next, a deterministic algorithm $\mathcal{A}$ working in polynomial time is presented, which receives as input a P system with proteins on membranes and membrane separation Π and an input multiset m of Π, in such manner that algorithm $\mathcal{A}$ reproduces the behavior of a computation of $\Pi + m$. If the system Π is confluent, then the algorithm $\mathcal{A}$ will provide the same answer of Π. We give the following pseudocode of the algorithm $\mathcal{A}$ to describe the simulation process:

Input: A P system with proteins on membranes and membrane separation Π and an input multiset m

Initialization phase: $\mathcal{C}_0$ is the initial configuration of $\Pi + m$

$t \leftarrow 0$

while $\mathcal{C}_t$ is a non-halting configuration **do**

Selection phase: Input $\mathcal{C}_t$, Output $(\mathcal{C}'_t, A)$

Execution phase: Input $(\mathcal{C}'_t, A)$, Output $\mathcal{C}_{t+1}$

$t \leftarrow t + 1$

end while

Output: yes if $\Pi + m$ has an accepting computation, no otherwise

The algorithm $\mathcal{A}$ receives a recognizer P system with proteins on membranes and membrane separation

$$\Pi = (\Gamma, \Gamma_0, \Gamma_1, P, P_0, P_1, \mu, \mathcal{M}_1/\mathcal{Z}_1, \ldots, \mathcal{M}_q/\mathcal{Z}_q, \mathcal{E}, \mathcal{R}_1, \ldots, \mathcal{R}_q, i_{out}),$$

where m is an input multiset for this system. Let $M = |\mathcal{M}_1 + \cdots + \mathcal{M}_q|$, $Z = |\mathcal{Z}_1 + \ldots \mathcal{Z}_q|$ and $S = M + Z$. Let any computation of Π perform at most p transition steps, $p \in \mathbb{N}^+$. Hence, from Lemma 1, the number of membranes in the system along any computation is bounded by $2M + (2 + n)Z$.

A transition of a recognizer P system $\Pi + m$ is performed in two phases: selection phase and execution phase.

Selection phase.

Input: A configuration $\mathcal{C}_t$ of $\Pi + m$ at an instant t

$\mathcal{C}'_t \leftarrow \mathcal{C}_t$; $A \leftarrow \emptyset$; $B \leftarrow \emptyset$

for $r \in \mathcal{R}_i \wedge r \in \mathcal{R}_C$, according to the chosen order **do**

for each membrane (i, σ_i) of $\mathcal{C}'_t$ according to the lexicographical order **do**

$n_r \leftarrow$ maximum number of times that r is applicable to (i, σ_i)

if $n_r > 0$ **then**

$\mathcal{C}'_t \leftarrow \mathcal{C}'_t - n_r \cdot LHS(r, (i, \sigma_i))$

$A \leftarrow A \cup \{(r, n_r, (i, \sigma_i))\}$

$B \leftarrow B \cup \{(i, \sigma_i)\}$

end if

end for

end for

for $r \in \mathcal{R}_i \wedge r \in \mathcal{R}_S$, according to the chosen order **do**

for each (a, i, σ_i) according to the lexicographical order, and such that $(i, \sigma_i) \notin B$ **do**

$\mathcal{C}'_t \leftarrow \mathcal{C}'_t \setminus \{(a, i, \sigma_i)\}$

$A \leftarrow A \cup \{(r, 1, (i, \sigma_i))\}$

$B \leftarrow B \cup \{(i, \sigma_i)\}$

end for

end for

It is easy to check that this algorithm is deterministic and its running time is polynomial in the size of Π because the number of cycles of the first main loop **for** is of the order $O(|\mathcal{R}| \cdot (M^2 + Z^2) \cdot q^2)$; and the number of cycles of the second main loop **for** is of the order $O(|\mathcal{R}| \cdot (M + Z) \cdot q \cdot (|\Gamma| + |P|))$.

Execution phase.

Input: The output $(\mathcal{C}'_t, A)$ of the selection phase

for each $(r, n_r, (i, \sigma_i)) \in A, r \in \mathcal{R}_C$ **do**

$\mathcal{C}'_t \leftarrow \mathcal{C}'_t + n_r \cdot RHS(r, (r, \sigma_i))$

end for

for each $(r, 1, (i, \sigma_i)) \in A, r \in \mathcal{R}_S$ **do**
$\mathcal{C}'_t \leftarrow \mathcal{C}'_t + \{(\lambda, i, \sigma_i)/\sigma_i 0\}$
$\mathcal{C}'_t \leftarrow \mathcal{C}'_t + \{(\lambda, i, \sigma_i 1)\}$
for each $(x, i, \sigma_i) \in \mathcal{C}'_t$ according to the lexicographical order **do**
if $x \in \Gamma_0$ **then**
$\mathcal{C}'_t \leftarrow \mathcal{C}'_t + \{(x, i, \sigma_i)/\sigma_i 0\}$
else
$\mathcal{C}'_t \leftarrow \mathcal{C}'_t + \{(x, i, \sigma_i)/\sigma_i 1\}$
end if
end for
end for
$\mathcal{C}_{t+1} \leftarrow \mathcal{C}'_t$

This algorithm is deterministic and its running time is polynomial in the size of Π because the number of cycles of the first main loop **for** is of the order $O(|\mathcal{R}| \cdot (M^2 + Z^2) \cdot q^2)$; and the number of cycles of the second main loop **for** is of the order $O(|\mathcal{R}| \cdot (M + Z) \cdot q \cdot (|\Gamma| + |P|))$.

Theorem 1. *Only problems from class* **P** *can be solved efficiently by P systems with proteins on membranes and membrane separation.*

Proof. Because the complexity class of recognizer P systems with proteins on membranes and membrane separation is closed under polynomial time reduction and non-empty, **P** is a subset of this class. In what follows, we show the reverse inclusion. Let X a decision problem that can be solved efficiently by recognizer P systems with proteins on membranes and membrane separation and let $\mathbf{\Pi} = \{\Pi(n) \mid n \in \mathbb{N}\}$ be a family of these kinds of P systems solving X according to Definition 3. Let (cod, s) be a polynomial encoding associated with that solution. If $u \in I_X$ is an instance of the problem X, then u will be processed by the system $\Pi(s(u)) + cod(u)$. We consider the following deterministic algorithm $\mathcal{A}'$:

Input: `An instance` u `of the problem` X
`Construct the system` $\Pi(s(u)) + cod(u)$
`Run algorithm` $\mathcal{A}$ `with input` $\Pi(s(u)) + cod(u)$
Output: `yes if` $\Pi(s(u)) + cod(u)$ `has an accepting computation, no otherwise`

The algorithm $\mathcal{A}'$ receives an instance u of the decision problem $X = (I_X, \theta_X)$, and working in a polynomial time. The following three assertions are equivalent:

- $\theta_X = 1$, that is, the answer of problem X to instance u is affirmative.
- Every computation of $\Pi(s(u)) + cod(u)$ is an accepting computation.
- The output of the algorithm $\mathcal{A}'$ with input u is `yes`.

Hence, $X \in \mathbf{P}$. □

5 Conclusions and future work

In this work, P systems with proteins on membranes and separation rules have been studied. While in [19] it has been shown that these systems can solve computationally hard problems using division rules, forbidding them takes from efficiency to non-efficiency. Moreover, even if we add the power of creating an exponential workspace in linear time by means of separation rules, it is shown by the *algorithmic technique* that only problems of the class **P** can be solved efficiently with this kind of systems. This result shows that an exponential workspace in terms of membranes is not enough, but some kind of creation of an exponential workspace of objects [4] is needed, thus a new borderline between efficiency and non-efficiency has been defined.

There are some open problems noted in [18] regarding P systems with proteins on membranes that have not been solved yet, so it seems an interesting research line to work on in order to obtain new frontiers of efficiency. In fact, in the Open Problem 4, the restriction of changing proteins while firing rules is allowed. If not, another frontier of efficiency could be found there.

Acknowledgements

This work was supported by Project TIN2017-89842-P of the Ministerio de Economía y Competitividad of Spain and by Grants No 61320106005 of the National Natural Science Foundation of China.

References

1. B. Alberts, A. Johnson, J. Lewis, M. Raff, K. Roberts, P. Walter. Molecular Biology of the Cell. (4th ed.), Garland Science, New York (2002).
2. A. Alhazov, R. Freund, and S. Ivanov. Extended spiking neural P systems with states. *Proceedings of the Fourteenth Brainstorming Week on Membrane Computing*, Seville, Spain, February 1 - 5, 2016. Report RGNC 01/2016, Fénix Editora, 2016, pp. 43–58.
3. J.M. Bahi, W. Bienia, N. Côté, C. Guyeux. Is protein folding problem really a NP-complete one ? First investigations, 2013, `arXiv:1306.1372`.
4. R. Brijder, M. Cavaliere, A. Riscos-Núñez, G. Rozenberg, D. Sburlan, Membrane systems with proteins embedded in membranes. *Theoretical Computer Science*, **404** (2008), Issues 12, 26–39

[4] When division rules are permitted, new objects are created along with the membranes.

5. M. Cardona, M.A. Colomer, M.J. Pérez-Jiménez, D. Sanuy, A. Margalida. Modeling ecosystems using P systems: The bearded vulture, a case study. Membrane Computing, 9th International Workshop, WMC 2008, Edinburgh, UK, July 28-31, 2008, Revised Selected and Invited Papers. *Lecture Notes in Computer Science*, 5391 (2009), 137-156.
6. T.H. Cormen, C.E. Leiserson, R.L. Rivest. *An Introduction to Algorithms*. The MIT Press, Cambridge, Massachusetts, 1994.
7. P. Frisco, M. Gheorghe, M. J. Pérez-Jiménez. Applications of Membrane Computing in Systems and Synthetic Biology. *Emergence, Complexity and Computation* (Series ISSN 2194-7287), Volume 7. Springer International Publishing, eBook ISBN 978-3-319-03191-0, Hardcover ISBN 978-3-319-03190-3, 2014, XVII + 266 pages (doi: 10.1007/978-3-319-03191-0).
8. M.Á. Gutiérrez-Naranjo, M.J. Pérez-Jiménez, A. Riscos-Núñez, F.J. Romero-Campero. Characterizing tractability by cell-like membrane systems. *Formal models, languages and applications*, World Scientific, Series in Machine Perception and Artificial Intelligence **66** (2006), chapter 9, pp. 137-154.
9. L. Pan, T.-O. Ishdorj. P systems with active membranes and separation rules. *Journal of Universal Computer Science*, **10**, 5, (2004), 630649.
10. L. Pan, Gh. Păun, M. J. Pérez-Jiménez, T. Song. Bio-inspired Computing: Theories and Applications. *Communications in Computer and Information Science* (Series ISSN 1865-0929), Volume 472, Springer-Verlag Berlin Heidelberg, Print ISBN 978-3-662-45048-2, Online ISBN 978-3-662-45049-9, 2014, XX + 672 pages (doi: 10.1007/978-3-662-45049-9).
11. A. Păun, B. Popa. P Systems with Proteins on Membranes and Membrane Division. *Developments in Language Theory*, DLT 2006. Lecture Notes in Computer Science, vol 4036. Springer, Berlin, Heidelberg, 292-303.
12. Gh. Păun. Computing with membranes. *Journal of Computer and System Sciences*, **61**, 1 (2000), 108–143, and *Turku Center for CS-TUCS Report* No. 208, 1998
13. H. Peng, J. Wang, J. Ming, P. Shi, M.J. Pérez-Jiménez, W. Yu, Ch. Tao. Fault diagnosis of power systems using intuitionistic fuzzy spiking neural P systems. *IEEE Transactions on Smart Grid*, in press (2017) (doi: 10.1109/TSG.2017.2670602).
14. M.J. Pérez-Jiménez, A. Riscos, A. Romero, D. Woods. Complexity: Membrane division, membrane creation. In Gh. Păun, G. Rozenberg, A. Salomaa (eds.) *The Oxford Handbook of Membrane Computing*, Oxford University Press, Oxford (U.K.), 2009, Chapter 12, pp. 302-336.
15. M.J. Pérez-Jiménez, Á. Romero-Jiménez, F. Sancho-Caparrini. Complexity classes in models of cellular computing with membranes. *Natural Computing*, **2**, 3 (2003), 265–285 (doi: 10.1023/A:1025449224520).
16. F.J. Romero-Campero, M.J. Pérez-Jiménez. A model of the Quorum Sensing System in Vibrio Fischeri using P systems. *Artificial Life*, 14, 1 (2008), 95-109 (doi: 10.1162/artl.2008.14.1.95).
17. B. Song, M.J. Pérez-Jiménez, L. Pan. An efficient time-free solution to QSAT problem using P systems with proteins on membranes. *Information and Computation*, **256** (2017), 287–299.
18. P. Sosík. Attacking Hard Problems beyond NP: A Survey. *Bulletin of the International Membrane Computing Society*, **4**, 89–106.
19. P. Sosík, A. Păun, A. Rodríguez-Patón. P systems with proteins on membranes characterize PSPACE. *Theoretical Computer Science*, **488** (2013), 78–95.

20. L. Valencia-Cabrera, B. Song, L.F. Macías-Ramos, L. Pan, A. Riscos-Núñez, M.J. Pérez-Jiménez. Computational Efficiency of P Systems with Symport/Antiport Rules and Membrane Separation. *Proceedings of the Thirteenth Brainstorming Week on Membrane Computing*, Seville, Spain, February 2 - 6, 2015. Report RGNC 01/2015, Fénix Editora, 2015, pp. 325–370.
21. G. Zhang, M. J. Pérez-Jiménez, M. Gheorghe. Real-life applications with Membrane Computing. *Emergence, Complexity and Computation* (Series ISSN 2194-7287), Volume 25. Springer International Publishing, Online ISBN 978-3-319-55989-6, Print ISBN 978-3-319-55987-2, 2017, X + 367 pages (`doi: 10.1007/978-3-319-55989-6`).