View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Repository of the Academy's Library

A new method to simulate restricted variants of polarizationless P systems with active membranes

Zsolt Gazdag[1]* and Gábor Kolonits[2]**

[1] Department of Foundations of Computer Science
University of Szeged
gazdag@inf.u-szeged.hu

[2] Department of Algorithms and their Applications
Eötvös Loránd University
kolomax@inf.elte.hu

Abstract. According to the *P conjecture* by Gh. Păun, polarizationless P systems with active membranes cannot solve **NP**-complete problems in polynomial time. The conjecture is proved only in special cases yet. In this paper we consider the case where only elementary membrane division and dissolution rules are used and the initial membrane structure consists of one elementary membrane besides the skin membrane. We give a new approach based on the concept of object division polynomials introduced in this paper to simulate certain computations of these P systems. Moreover, we show how to compute efficiently the result of these computations using these polynomials.

Keywords: Membrane Computing, active membranes, computational complexity

1 Introduction

P systems with active membranes, introduced in [13], are among the most investigated variants of P systems. Using the polarizations of the membranes and the possibility of dividing elementary (or even non-elementary) membranes these systems can solve computationally hard problems efficiently. More precisely, with elementary membrane division they can solve **NP**-complete problems [8,13,17,20], while with non-elementary membrane division they can solve even **PSPACE**-complete problems efficiently [1,18]. Solving computationally hard problems with P systems with active membranes has a huge literature in Membrane Computing, see e.g. [2,5,6,10,12,16], and the references therein.

* Research of this author was supported by the Ministry of Human Capacities, Hungary, grant no. 20391-3/2018/FEKUSTRAT.

** Research of this author was supported by NKFIH – National Research, Development, and Innovation Office, Hungary, grant no. K 120558.

It is a frequently investigated question whether these P systems are still powerful enough to solve hard problems when the polarizations of the membranes are not used (see e.g. [3,7,9,11,19]). In the case when non-elementary membrane division is allowed the answer to this question is positive since in [3] the **PSPACE**-complete QSAT problem was solved efficiently without polarizations. On the other hand, no efficient solutions of hard problems exist when non-elementary membrane division is not allowed. In fact, Gh. Păun conjectured already in 2005 that without polarization and non-elementary membrane division P systems with active membranes cannot solve **NP**-complete problems in polynomial time [14]. Păun's conjecture, often called the *P conjecture*, has not been proven yet. A direct attempt to calculate efficiently all the elementary membranes of a computation of such a P system fails as in general the number of these membranes is exponential and, moreover, these membranes can contain pairwise different multisets. However, it was discovered in [7] that if dissolution rules are not allowed to use, then there is no need to simulate all the elementary membranes to determine the result of a computation. Instead, it is enough to consider a certain graph, called the *dependency graph* [4] of the P system. Roughly, this graph describes how the rules of the P system can evolve and move objects through the membranes. To determine the result of a computation in this case it is enough to check whether a distinguished object is reachable from certain objects in the dependency graph.

If dissolution rules are also allowed, then things became much more complicated. Consider a P system Π with active membranes and assume, for example, that Π contains a membrane sub-structure $[[a\ b]_2]_1$. Assume moreover that a can dissolve membrane 2 but b cannot. Then Π dissolves membrane 2 using a, and b immediately gets to membrane 1 without directly being involved in any application of a rule. Notice that when b gets to membrane 2 then it "knows" that Π contained an occurrence of a in the same membrane. This way the objects can send information to each other and this kind of behaviour cannot be captured by dependency graphs.

Using generalizations of dependency graphs the P conjecture was already proved in some special cases where the P systems were allowed to use dissolution rules as well. In [19], for example, the P conjecture was proved using *object division graphs* in the case where the initial membrane structure of the P system is a linearly nested sequence of membranes, and the system can employ only dissolution and elementary membrane division rules. In [9] the P conjecture was proved in another case using a generalization of dependency graphs. Here the P systems are deterministic, can use all types of rules except send-in communication rules, and the membrane structure is such that the skin contains only elementary membranes. In these papers the authors used these generalizations of dependency graphs in order to simulate a reasonable small part of the configurations in a computation of the investigated P systems.

In this paper we propose a new method for simulating polarizationless P systems using both division and dissolution rules. Using this method it is possible to calculate efficiently the number of objects appearing in the skin membrane during a computation of a P system. Our approach can be roughly described as follows. Consider a P system Π, its input multiset ω, and a computation $\mathcal{C}$ of Π. First we define *object division polynomials* based on the concept of object division graphs. The object division polynomial of an object a describes which and how many objects can be created by Π using only division rules starting the division using a. Then we consider a polynomial P_ω which is, roughly, the multiplication of the object division polynomials of objects in ω. We show that there is a strong relationship between the monomials of P_ω and the number of certain membranes in $\mathcal{C}$. Using this we can calculate efficiently which and how many objects get to the skin membrane in each step of $\mathcal{C}$.

In order to make the presentations as transparent as possible, we give our method only for a rather restricted variant of P systems. In this variant the P systems, for example, have only one elementary membrane in the skin at the beginning of the computation and can employ only membrane division and membrane dissolution rules. Moreover, we will simulate only such computations of these P systems, where division rules have priority over dissolution rules. However, we believe that our method can be extended to more general variants of P systems as it is discussed in the Conclusions section.

2 Preliminaries

Here we recall the necessary notions used later. Nevertheless, we assume that the reader is familiar with the basic concepts of membrane computing techniques (for a comprehensive guide see e.g. [15]).

$\mathbb{N}$ denotes the set of natural numbers and, for every $i, j \in \mathbb{N}$, $i \leq j$, $[i, j]$ denotes the set $\{i, i+1, \ldots, j\}$. If $i = 1$, then $[i, j]$ is denoted by $[j]$. We will use polynomials with coefficients in $\mathbb{N}$. A polynomial of the form $p = cx_1^{j_1} \ldots x_n^{j_n}$ where $c, n \in \mathbb{N}, x_1, \ldots, x_n$ are variables, and $j_1, \ldots, j_n \geq 1$ is called a *monomial* and c is called the *coefficient* of p. An $n \times m$ *matrix* $\mathbf{M}$ has n rows and m columns. We will consider matrices with entries in $\mathbb{N}$. $(\mathbf{M})_{ij}$ denotes the jth element of the ith row of $\mathbf{M}$. By a *vector* $\mathbf{v}$ we mean an $n \times 1$ matrix, for some $n \geq 1$. $\mathbf{v}^T$ denotes the transpose of $\mathbf{v}$, and instead of $(\mathbf{v})_{j1}$ and $(\mathbf{v}^T)_{1j}$ ($j \in [n]$) we will write simply $(\mathbf{v})_j$ and $(\mathbf{v}^T)_j$, respectively. If a vector $\mathbf{v}$ has n entries, for some $n \geq 1$, then $\mathbf{v}$ is called an *n-dimensional vector* or just an *n-vector*.

In this paper we consider polarizationless P systems. A polarizationless *P system* with active membranes [13] is a construct of the form $\Pi = (O, H, \mu, w_1, \ldots, w_m, R)$, where m is the initial *degree* of the system, O is the alphabet of *objects*, H is the set of *labels* of the membranes, μ is a *membrane structure* consisting of m membranes labelled with the elements of H in a one-to-one manner, $w_1, \ldots, w_m \in O^*$ are the *initial multisets of objects* placed in the m regions of μ; and R is a finite set of *rules* defined as follows:

(a) $[a \to v]_h$, where $h \in H, a \in O, v \in O^*$
(object *evolution* rules);

(b) $a[\ ]_h \to [b]_h$, where $h \in H$, $a, b \in O$
(*send-in communication* rules);

(c) $[a]_h \to [\ ]_h b$, where $h \in H$, $a, b \in O$
(*send-out communication* rules);
(d) $[a]_h \to b$, where, $h \in H$, $a, b \in O$
(*membrane dissolution* rules);
(e) $[a]_h \to [b]_h[c]_h$, where $h \in H$, $a, b, c \in O$
(*division* rules for elementary membranes).

For any rule r of the form $u \to v$, u (resp. v) is called the *left-hand side* (resp. the *right-hand side*) of r.

As it is usual in membrane computing, P systems with active membranes work in a *maximally parallel* manner: at each step the system first nondeterministically assigns appropriate rules to the objects of the system such that the assigned multiset S of rules satisfies the following properties: (i) at most one rule from S is assigned to any object of the system, (ii) a membrane can be the subject of at most one rule in S, and (iii) S is maximal among the multisets of rules satisfying (i) and (ii).

Let $\Pi = (O, H, \mu, w_1, \ldots, w_m, R)$ be a polarizationless P system with active membranes. Π is called a *simple divide-dissolve P system* (*sdd P system*, for short) if

– $H = \{1, s\}$ and $\mu = [[\ ]_1]_s$,
– Π employs only membrane division and membrane dissolution rules,
– different rules of Π have different left-hand sides, and
– the dissolution rules have the form $[a]_1 \to a\,(a \in O)$.

We call a membrane with label 1 a *working membrane* (notice that as the skin cannot be divided or dissolved, the objects in the skin are not changing as they cannot evolve in any way). An object $a \in O$ is called a *divider* if a can divide working membranes, that is, R contains a division rule with $[a]_1$ on the left-hand side. Likewise, an object $a \in O$ is called a *dissolver* if a can dissolve working membranes, that is, R contains the dissolution rule $[a]_1 \to a$. Furthermore, Π is called *halting*, if each of its computations halts. In the rest of the paper we consider only polarizationless halting sdd P systems.

3 Results

In the rest of the paper let $\Pi = (O, \{s, 1\}, [[\ ]_1]_s, \omega_1, \omega_s, R)$ be a halting sdd P system, where $O = \{a_1, \ldots, a_n\}$ $(n \in \mathbb{N})$ and $\omega_1 = a_{i_1} \ldots a_{i_m}$, for some $m \geq 1$ and $i_1, \ldots, i_m \in [n]$. To simplify the arguments we assume also that $\omega_s = \varepsilon$.

In this section we show that the multiset content of the skin membrane of Π at the end of the so-called *division driven computations* can be computed in polynomial time in nm. In division driven computations division rules have priority over dissolution rules and there is a certain order between the division rules too. To specify these computations precisely we need some preparation.

Consider a halting computation $\mathcal{C} : C_0 \Rightarrow C_1 \Rightarrow \ldots \Rightarrow C_t$ of Π. We first assign to each occurrence of an object occurring in a working membrane a label defined inductively as follows. The label of an object a_{i_ℓ} ($\ell \in [m]$) in ω_1 in C_0 is ℓ. Now, let M be a working membrane in C_i, for some $i \in [0, t-1]$, and consider an occurrence of an object a in M with label ℓ ($\ell \in [m]$). Then we have exactly one of the following three cases: (i) this occurrence of a is not involved in the application of any rule, or (ii) it is involved in the application of a division rule $r : [a]_1 \to [b]_1[c]_1$, or (iii) it is involved in the application of a dissolution rule during $C_i \Rightarrow C_{i+1}$. In Case (i) the same occurrence of a occurs in C_{i+1} too. Then let the label of this occurrence of a in C_{i+1} be ℓ. In Case (ii) r divides M into two new membranes in C_{i+1}. Then let the label of the occurrences of b and c introduced by r in these two new membranes be ℓ. In Case (iii) no objects are introduced in the working membranes by the considered occurrence of a, thus no labelling is necessary in this case. If a is an object with label ℓ, then we will often denote this by $a^{(\ell)}$.

Notice that the multiset content of a working membrane in $\mathcal{C}$ always has the form $a_{j_1}^{(1)} \dots a_{j_m}^{(m)}$, for some $j_1, \dots, j_m \in [n]$. Using the labels of the objects we can define now division driven computations as follows.

Definition 1. *Consider a halting computation $\mathcal{C}$. $\mathcal{C}$ is called* division driven *if when a division rule is applied in a membrane M triggered by an object $a_i^{(\ell)}$ ($i \in [n], \ell \in [m]$), then M contains no object $a_j^{(\ell')}$ with $\ell' < \ell$ such that a_j can divide M.*

Intuitively, in a division driven computation $\mathcal{C}$ of Π the computation goes as follows. Assume that the labels of those objects in ω_1 that can divide working membranes are $\ell_1 < \ldots < \ell_k$, for some $k \in [m]$. Then first objects with label ℓ_1 are used to divide working membranes, then those objects which have label ℓ_2, and so on until at the end those objects are used which have label ℓ_k. Then those objects are used which can dissolve working membranes, and if no more working membranes can be dissolved, the computation terminates. Notice that if a non-dividing object a with label ℓ occurs in a working membrane, then this object remains unchanged until the computation halts.

Example 1. Let $\Pi_{ex} = (\{a_1, a_2, a_3, a_4\}, \{s, 1\}, [[\ ]_1]_s, a_1^{(1)} a_1^{(2)}, \varepsilon, R)$, where

$$R = \{[a_1]_1 \to [a_2]_1[a_3]_1, [a_2]_1 \to [a_4]_1[a_4]_1, [a_4]_1 \to a_4\}.$$

Figure 1 shows a division driven computation of Π_{ex}. Recall that the numbers in parentheses are the labels of the corresponding objects. Notice that each working membrane contains two objects with labels 1 and 2, respectively. $\square$

Now we define a concept similar to that of *object division graphs* (see e.g. in [19]). The *object division tree* of a_i ($i \in [n]$), denoted by odt_{a_i} is the smallest binary tree satisfying the following conditions:

- the root of odt_{a_i} is labelled by a_i, and
- if a node N of odt_{a_i} is labelled by a_j ($j \in [n]$) and $[a_j]_1 \to [a_k]_1[a_l]_1 \in R$ ($k, l \in [n]$) then N has exactly two children with labels a_k and a_l, respectively.

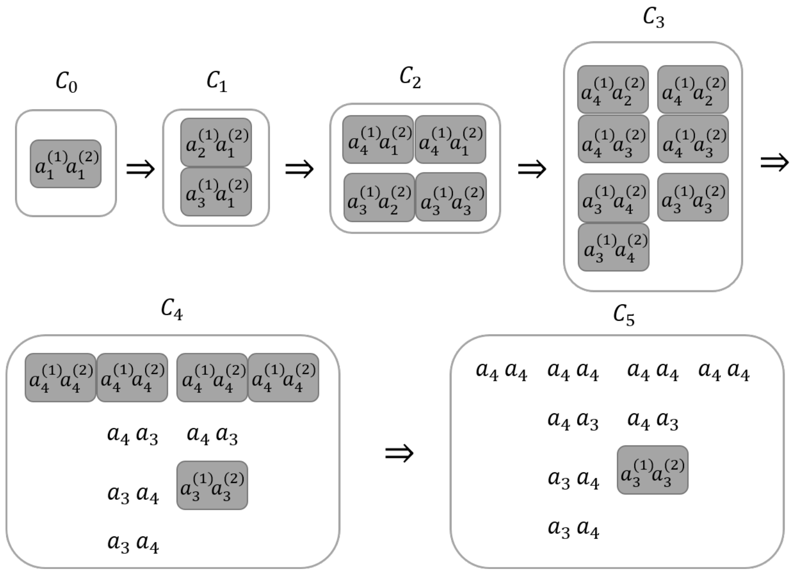
C_0
$a_1^{(1)}a_1^{(2)}$
C_1
$a_2^{(1)}a_1^{(2)}$
$a_3^{(1)}a_1^{(2)}$
C_2
$a_4^{(1)}a_1^{(2)}$ $a_4^{(1)}a_1^{(2)}$
$a_3^{(1)}a_2^{(2)}$ $a_3^{(1)}a_3^{(2)}$
C_3
$a_4^{(1)}a_2^{(2)}$ $a_4^{(1)}a_2^{(2)}$
$a_4^{(1)}a_3^{(2)}$ $a_4^{(1)}a_3^{(2)}$
$a_3^{(1)}a_4^{(2)}$ $a_3^{(1)}a_3^{(2)}$
$a_3^{(1)}a_4^{(2)}$
C_4
$a_4^{(1)}a_4^{(2)}$ $a_4^{(1)}a_4^{(2)}$ $a_4^{(1)}a_4^{(2)}$ $a_4^{(1)}a_4^{(2)}$
$a_4\, a_3$ $a_4\, a_3$
$a_3\, a_4$ $a_3^{(1)}a_3^{(2)}$
$a_3\, a_4$
C_5
$a_4\, a_4$ $a_4\, a_4$ $a_4\, a_4$ $a_4\, a_4$
$a_4\, a_3$ $a_4\, a_3$
$a_3\, a_4$ $a_3^{(1)}a_3^{(2)}$
$a_3\, a_4$

Fig. 1. A division driven computation of Π_{ex} from Example 1. Grey areas are regions of working membranes.

Since Π is an sdd P system, it does not have different division rules with the same left-hand side. Thus odt_{a_i} is well defined. Notice that in odt_{a_i} a subtree with a root labelled by an object a_j ($j \in [n]$) is equal to odt_{a_j}. The *height of* odt_{a_i}, denoted by $h(\mathrm{odt}_{a_i})$, is defined inductively as follows. If odt_{a_i} is a single node labelled by a_i, then $h(\mathrm{odt}_{a_i}) = 0$. Otherwise let h_{max} be the maximum of the heights of subtrees of the root in odt_{a_i}. Then $h(\mathrm{odt}_{a_i}) = h_{max} + 1$.

Example 2. Consider again Π_{ex} from Example 1. The tree odt_{a_1} can be seen in Figure 2. Notice that odt_{a_2} and odt_{a_3} are equal to the first and second subtrees of odt_{a_1}, respectively, and odt_{a_4} is equal, for example, to the first subtree of odt_{a_2}. □

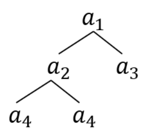

Fig. 2. The tree odt_{a_1} from Example 2.

Next we show a useful property of object division trees.

Lemma 1. *Consider Π and $i \in [n]$ such that a_i occurs in ω_1. Then $h(\mathrm{odt}_{a_i}) < n$.*

Proof. *We give an indirect proof. Assume that $h(\mathrm{odt}_{a_i}) \geq n$. Then there exists a path P in odt_{a_i} with length at least n. Due to the pigeonhole principle, there exists $j \in [n]$ such that a_j occurs at least twice in P. Let N_1 and N_2 be the first two nodes of P (counted from the root) labelled by a_j. Let t_1 and t_2 be the subtrees of odt_{a_i} with roots N_1 and N_2, respectively. Clearly, t_2 is a proper subtree of t_1. Moreover, by our above note $t_1 = t_2 = \mathrm{odt}_{a_j}$. This implies that odt_{a_i} is infinite, which further implies that a division driven computation will never halt. However, this contradicts to the fact that Π is a halting sdd P system, proving our statement.* □

Notice that, for every division driven halting computation $\mathcal{C} : C_0 \Rightarrow C_1 \Rightarrow \ldots \Rightarrow C_t$ of Π, $t \leq \sum\limits_{a_i \in \omega_1} h(odt_{a_i}) + 1$.

Every object division tree defines an *object division polynomial* as follows.

Definition 2. *Consider Π and let $V = \{x_i \mid i \in [n]\} \cup \{x\}$ be a set of variables. Let moreover $i \in [n]$ and $l = h(\mathrm{odt}_i)$. The* object division polynomial *of a_i (odp_{a_i} for short) is a polynomial with variables in V defined as follows:*

$$\mathrm{odp}_{a_i} = \sum_{j \in [0,l], k \in [n]} m_{jk} \cdot x_k \cdot x^j,$$

where m_{jk} is the number of leaves in odt_{a_i} at depth j labelled by a_k.

Example 3. Consider Π_{ex} from Example 1 and the object division trees considered in Example 2. The corresponding object division polynomials are as follows:

– $\mathrm{odp}_{a_1} = 2x_4x^2 + x_3x$,
– $\mathrm{odp}_{a_2} = 2x_4x$,
– $\mathrm{odp}_{a_3} = x_3$,
– $\mathrm{odp}_{a_4} = x_4$.

□

Next we show that object division polynomials can be calculated efficiently.

Lemma 2. *Consider Π and let $i \in [n]$. Then odp_{a_i} can be computed in polynomial time in n.*

Proof. *Let $l = h(\mathrm{odt}_{a_i})$ and, for every $j \in [0,l]$, let $\mathbf{v}_j$ be an n-vector such that $(\mathbf{v}_j)_k$ ($k \in [n]$) is the number of nodes labelled by a_k on the* j*th level of odt_{a_i}. Let $ndiv = \{j \in [n] \mid a_j$ is a non-divider$\}$. As the set of labels of leaves in odt_{a_i} is included in the set $\{a_j \mid j \in ndiv\}$, we get that*

$$\mathrm{odp}_{a_i} = \sum_{j \in [0,l], k \in [n]} \mathbf{v}_j \mathbf{e}_k x_k x^j,$$

where $\mathbf{e}_k$ $(k \in [n])$ *is an* n*-vector defined as follows:*

$$(e_k)_s = \begin{cases} 1 & \textit{if } s = k \textit{ and } k \in ndiv \\ 0 & \textit{otherwise.} \end{cases}$$

To compute $\mathbf{v}_j$ $(j \in [0,l])$ *let us define, for every* $k \in [n]$*, the* n*-vector* $\mathbf{m}_k$ *as follows: for every* $s \in [n]$*, if there is a rule* r *with* a_s *on the left- and* a_k *on the right-hand side, then let* $(\mathbf{m}_k)_s$ *be the number of occurrences of* a_k *on the right-hand side of* r*. If there is no such rule in* R*, then let* $(\mathbf{m}_k)_s$ *be* 0*. It can be clearly seen that if we multiply* $\mathbf{v}_j^T$ $(j \in [0, l-1])$ *with* $\mathbf{m}_k$ $(k \in [n])$*, we get the number of occurrences of* a_k *on the* $(\mathrm{j}+1)$*th level of* odt_{a_i}*. Thus, for every* $j \in [0, l-1]$*,* $\mathbf{v}_{j+1}^T = \mathbf{v}_j^T\mathbf{M}$*, where* $\mathbf{M}$ *is the* $n \times n$ *matrix whose* k*th column* $(k \in [n])$ *is* $\mathbf{m}_k$*. Since matrix multiplication is associative, we get that* $\mathbf{v}_j^T = \mathbf{v}_0^T\mathbf{M}^j$ $(j \in [l])$*. This implies that*

$$\mathrm{odp}_{a_i} = \sum_{j \in [0,l], k \in [n]} \mathbf{v}_0^T \mathbf{M}^j \mathbf{e}_k x_k x^j.$$

Notice that since the 0*th level of* odt_{a_i} *contains only the root of* odt_{a_i}*,* $(\mathbf{v}_0)_k = 1$ *if* $k = i$*, and* $(\mathbf{v}_0)_k = 0$ *otherwise. Therefore, the coefficient of a factor* $x_k x^j$ *in* odp_{a_i} *is* $(\mathbf{M}^j)_{ik}$*. Thus, we only have to compute* $\mathbf{M}^j$ *for every* $j \in [0,l]$*. Since every row in* $\mathbf{M}$ *contains at most two non-zero elements and the sum of these elements is two, it is easy to see that the largest value in* $\mathbf{M}^j$ *is at most* 2^j*. So these values can be stored using* n *bits and thus computing one entry of* $\mathbf{M}^{j+1}$ *can be done in* $\mathcal{O}(n)$ *steps. Since* $\mathbf{M}$ *is an* $n \times n$ *matrix, computing every necessary value can be done in polynomial time in* n*.* □

Example 4. Consider odp_{a_1} and odp_{a_2} given in Example 3. According to the proof of Lemma 2, we can compute these polynomials as follows. We will use $\mathbf{e}_i$ $(i \in [4])$ and $\mathbf{M}^j$ $(j \in [0,2])$ in the computation of each polynomial. These have the following values: $\mathbf{e}_1^T = \mathbf{e}_2^T = \begin{bmatrix} 0\,0\,0\,0 \end{bmatrix}, \mathbf{e}_3^T = \begin{bmatrix} 0\,0\,1\,0 \end{bmatrix}, \mathbf{e}_4^T = \begin{bmatrix} 0\,0\,0\,1 \end{bmatrix}$, and

$$\mathbf{M}^0 = \begin{bmatrix} 1\,0\,0\,0 \\ 0\,1\,0\,0 \\ 0\,0\,1\,0 \\ 0\,0\,0\,1 \end{bmatrix}, \mathbf{M}^1 = \begin{bmatrix} 0\,1\,1\,0 \\ 0\,0\,0\,2 \\ 0\,0\,0\,0 \\ 0\,0\,0\,0 \end{bmatrix}, \mathbf{M}^2 = \begin{bmatrix} 0\,0\,0\,2 \\ 0\,0\,0\,0 \\ 0\,0\,0\,0 \\ 0\,0\,0\,0 \end{bmatrix}.$$

Moreover, in the case of odp_{a_1} $\mathbf{v}_0^T = [1\ 0\ 0\ 0]$ and $l = 2$. Clearly,

$$\sum_{j \in [0,2], k \in [4]} \mathbf{v}_0^T \mathbf{M}^j \mathbf{e}_k x_k x^j =$$

$$\sum_{k \in [4]} \mathbf{v}_0^T \mathbf{M}^0 \mathbf{e}_k x_k + \sum_{k \in [4]} \mathbf{v}_0^T \mathbf{M}^1 \mathbf{e}_k x_k x + \sum_{k \in [4]} \mathbf{v}_0^T \mathbf{M}^2 \mathbf{e}_k x_k x^2.$$

Thus, in the case of odp_{a_1} we get that

$$\sum_{j\in[0,2],k\in[4]} \mathbf{v}_0^T\mathbf{M}^j\mathbf{e}_k x_k x^j =$$
$$\sum_{k\in[4]}[1\ 0\ 0\ 0]\mathbf{e}_k x_k + \sum_{k\in[4]}[0\ 1\ 1\ 0]\mathbf{e}_k x_k x +$$
$$\sum_{k\in[4]}[0\ 0\ 0\ 2]\mathbf{e}_k x_k x^2 = (0x_1 + 0x_2 + 0x_3 + 0x_4) +$$
$$(0x_1x + 0x_2x + 1x_3x + 0x_4x) + (0x_1x^2 + 0x_2x^2 + 0x_3x^2 + 2x_4x^2) =$$
$$2x_4x^2 + x_3x = \mathrm{odp}_{a_1}.$$

On the other hand, in the case of odp_{a_2} $\mathbf{v}_0^T = [0\ 1\ 0\ 0]$ and $l = 1$. Thus we get the following calculation.

$$\sum_{j\in[0,1],k\in[4]} \mathbf{v}_0^T\mathbf{M}^j\mathbf{e}_k x_k x^j =$$
$$\sum_{k\in[4]}[0\ 1\ 0\ 0]\mathbf{e}_k x_k + \sum_{k\in[4]}[0\ 0\ 0\ 2]\mathbf{e}_k x_k x =$$
$$(0x_1 + 0x_2 + 0x_3 + 0x_4) + (0x_1x + 0x_2x + 0x_3x + 2x_4x) =$$
$$2x_4x = \mathrm{odp}_{a_2}.$$

□

Object division polynomials can be used to calculate the multiset contents of certain working membranes occurring in a division driven computation of Π. To see this we need some preparation. First we extend the definition of object division polynomials to objects having labels.

Definition 3. *Consider Π and let* $\mathrm{odp}_{a_i} = \sum_{j\in[0,l],k\in[n]} m_{jk}x_kx^j$. *Let moreover $\ell \in [m]$. The* labelled object division polynomial *of $a_i^{(\ell)}$ (* $\mathrm{lodp}_{a_i^{(\ell)}}$ *for short) is the polynomial* $\sum_{j\in[0,l],k\in[n]} m_{jk}x_{\ell k}x^j$ *(that is, we add ℓ to the indices of variables of* odp_{a_i}*, referring this way to the label of the corresponding object).*

Consider a division driven computation $\mathcal{C} : C_0 \Rightarrow C_1 \Rightarrow \ldots \Rightarrow C_t$ of Π. Let M be a working membrane in $\mathcal{C}$ and $\ell \in [m]$. If M contains no dividers with label $\ell' \leq \ell$, then M is called ℓ-divider-stable. *Moreover, m-divider-stable working membranes are called* non-dividing. *Consider an ℓ-divider-stable membrane M in C_i ($\ell \in [m], i \in [t]$). M is called* primary *if either $i = 0$ or the following holds. Let N be that membrane in C_{i-1} from which Π derives M. Then N is not ℓ-divider-stable.*

Example 5. Let Π_{ex} be the P system given in Example 1 and consider the working membrane M containing $a_3^{(1)}a_2^{(2)}$ in C_2. Then M is 1-divider-stable, as the only object in M having label 1 or less is a_3 which is a non-divider. However, this M is not 2-divider-stable, since it contains a_2 having label 2 and a_2 is a divider. M is neither primary, as M is derived from the working membrane N in C_1 containing $a_3^{(1)}a_1^{(2)}$ but N is 1-divider-stable, too. However, N is primary, since it is derived from the working membrane in C_0 containing $a_1^{(1)}a_2^{(2)}$, which is not 1-divider-stable.

The only working membrane in C_5 is non-dividing, as it is 2-divider-stable, and 2 is the greatest label in this example. Notice that non-dividing working membranes are those which do not contain dividers. □

Next we define a product of labelled object division polynomials of Π which we will use frequently in what follows.

Definition 4. *Consider again Π and its initial membrane content ω_1. The ω_1-product of Π is* $P_{\omega_1} = \prod_{\ell \in [m]} \mathrm{lodp}_{a_{i_\ell}^{(\ell)}}$.

It is easy to see that all monomials in P_{ω_1} have the form $\alpha x_{1j_1} \dots x_{mj_m} x^j$, for some $\alpha, j \in \mathbb{N}$ and $j_1, \dots, j_m \in [n]$. In the next lemma we show that there is a strong relationship between the monomials in P_{ω_1} and the primary non-dividing working membranes of a division driven halting computation of Π.

Lemma 3. *Consider Π, its ω_1-product P_{ω_1}, and a division driven halting computation $\mathcal{C} : C_0 \Rightarrow C_1 \Rightarrow \dots \Rightarrow C_t$ of Π. Let moreover $j_1, \dots, j_m \in [n]$ and $j \in [0, t]$. Then the coefficient of $x_{1j_1} \dots x_{mj_m} x^j$ in P_{ω_1} equals to the number of those primary m-divider-stable working membranes in C_j which contain $a_{j_1}^{(1)} \dots a_{j_m}^{(m)}$.*

Proof. We show the statement by induction on m. If $m = 0$, then P_{ω_1} is the empty product, that is, $P_{\omega_1} = 1$. In this case P_{ω_1} can be considered as the monomial x^0, where the coefficient is one. On the other hand, $\mathcal{C}$ consists of C_0 only. C_0 has exactly one working membrane, which is empty and primary m-divider-stable. This proves the statement in this case.

Now assume that the statement holds if $m = m'$, for some $m' \geq 0$. We show it for $m = m' + 1$. Let α be the coefficient of a monomial $x_{1j_1} \dots x_{mj_m} x^j$ in P_{ω_1}. Let moreover $\hat{\alpha}$ be the number of those primary m-divider-stable working membranes in C_j which contain $a_{j_1}^{(1)} \dots a_{j_m}^{(m)}$. We show that $\alpha = \hat{\alpha}$.

Let $\omega_1' = a_{i_1} \dots a_{i_{m'}}$ and $P_{\omega_1'} = \prod_{\ell \in [m']} \mathrm{lodp}_{a_{i_\ell}^{(\ell)}}$. Clearly, $P_{\omega_1} = P_{\omega_1'} \mathrm{lodp}_{a_{i_m}^{(m)}}$. Let us denote by $\alpha_{j'}$ and $\beta_{j''}$ ($j', j'' \in [t]$) the coefficients of $x_{1j_1} \dots x_{m'j_{m'}} x^{j'}$ in $P_{\omega_1'}$ and $x_{mj_m} x^{j''}$ in $\mathrm{lodp}_{a_{i_m}^{(m)}}$, respectively. One can see that α can be calculated by summing up the products $\alpha_{j'}\beta_{j''}$, for every $j', j'' \in [t]$ with $j' + j'' = j$.

On the other hand, let $j', j'' \in [t]$ and denote $\hat{\alpha}_{j'}$ the number of those primary m'-divider-stable working membranes in $C_{j'}$ which contain $a_{j_1}^{(1)} \dots a_{j_{m'}}^{(m')}$. Denote, moreover, $\hat{\beta}_{j''}$ the number of leaves labelled by a_{j_m} in $\mathrm{odt}_{a_{i_m}}$ at depth j''. Consider now a membrane M in C_j containing $a_{j_1}^{(1)} \dots a_{j_m}^{(m)}$. One can see that the only way for Π to create M is the following. First Π creates a membrane N containing $a^{(1)}_{j_1} \dots a^{(m')}_{j_{m'}} a^{(m)}_{i_m}$ in j' steps ($j' \in [t]$) using only dividers having labels m' or less. Then, using dividers with label m, Π creates M in $j'' = j - j'$ steps. Thus $\hat{\alpha}$ can be calculated by summing up the products $\hat{\alpha}_{j'} \hat{\beta}_{j''}$, for every $j', j'' \in [t]$ with $j' + j'' = j$.

By induction hypothesis, $\alpha_{j'} = \hat{\alpha}_{j'}$, for every $j' \in [t]$. Moreover, by the definition of object division polynomials, $\beta_{j''} = \hat{\beta}_{j''}$, for every $j'' \in [t]$. Thus we have that

$$\alpha = \sum_{\substack{j', j'' \in [t],\\ j' + j'' = j}} \alpha_{j'} \beta_{j''} = \sum_{\substack{j', j'' \in [t],\\ j' + j'' = j}} \hat{\alpha}_{j'} \hat{\beta}_{j''} = \hat{\alpha},$$

which finishes the proof of the lemma. □

We show now through an example how to use P_{ω_1} to calculate multiset contents of certain membranes.

Example 6. Consider Π_{ex} from Example 1 and the computation $\mathcal{C}$ given in Figure 1. From Example 3 we know that $\mathrm{odp}_{a_1} = 2x_4x^2 + x_3x$. Thus $\mathrm{lodp}_{a^{(1)}_1} = 2x_{14}x^2 + x_{13}x$ and $\mathrm{lodp}_{a^{(2)}_1} = 2x_{24}x^2 + x_{23}x$. As $P_{a^{(1)}_1 a^{(2)}_1} = \mathrm{lodp}_{a^{(1)}_1} \mathrm{lodp}_{a^{(2)}_1}$ we get that

$$\begin{aligned} P_{a^{(1)}_1 a^{(2)}_1} = (2x_{14}x^2 + x_{13}x)(2x_{24}x^2 + x_{23}x) = \\ 4x_{14}x_{24}x^4 + 2x_{14}x_{23}x^3 + 2x_{13}x_{24}x^3 + x_{13}x_{23}x^2. \end{aligned}$$

Figure 3 shows the correspondence between the monomials of $P_{a^{(1)}_1 a^{(2)}_1}$ and the primary non-dividing working membranes of $\mathcal{C}$. Notice that the variables x_{ij} ($i \in [2], j \in [4]$) correspond to objects $a^{(i)}_j$, the coefficient of a monomial corresponds to the number of the corresponding membranes, and the power of x shows the index of the corresponding configuration. □

Consider again Π and a division driven halting computation $\mathcal{C}$ of Π. As we have seen, the multiset content of the primary non-dividing working membranes of $\mathcal{C}$ can be calculated by determining the monomials of P_{ω_1}. Clearly, if we know these multisets, then we can tell which and how many objects are sent to the skin (by applying membrane dissolution rules) in each step of $\mathcal{C}$. However, the size of P_{ω_1} can be exponential in nm, which means that we cannot use P_{ω_1} directly to calculate the number of these objects efficiently. Instead, we will use another polynomial given by using the following definition.

Definition 5. *Consider Π and let P be a polynomial over the variables $V = (\{x_{\ell k} \mid \ell \in [m], k \in [n]\} \cup \{x\})$. Let moreover $i \in [n]$ and y be a new variable not occurring in V. The i-*reduction *of P is the polynomial $P^{\langle i \rangle}$ which we get from P by performing the following operations. First, for every $\ell \in [m], k \in [n]$ with*

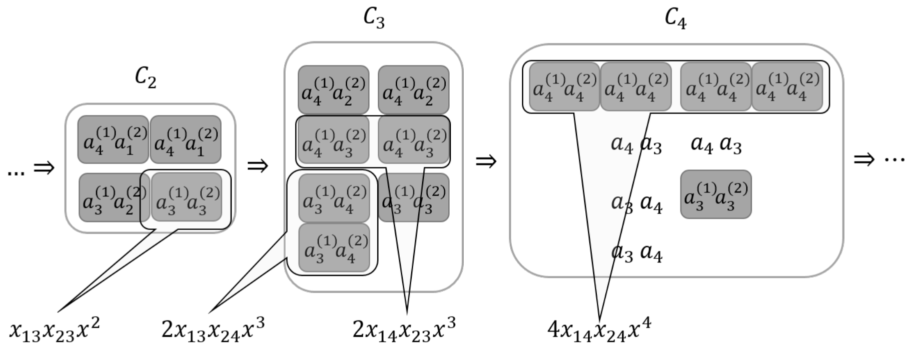

Fig. 3. The representation of non-dividing working membranes of Π_{ex} by monomials.

$k \neq i$, let us substitute $x_{\ell k}$ in P with z, where

$$z = \begin{cases} y, & \text{if } a_k^{(\ell)} \text{ can dissolve working membranes, and} \\ 1, & \text{otherwise.} \end{cases}$$

Let the given new polynomial be P'. Then let $P^{\langle i \rangle}$ be the polynomial created from P' by substituting $x_{\ell i}$ with x_i, for every $\ell \in [m]$.

Example 7. The i-reductions ($i \in [4]$) of $P_{a_1^{(1)}a_1^{(2)}}$ given in Example 6 are as follows:

$$P^{\langle 1 \rangle}_{a_1^{(1)}a_1^{(2)}} = 4yyx^4 + 2y1x^3 + 2 \cdot 1yx^3 + 1 \cdot 1x^2 = 4y^2x^4 + 4yx^3 + x^2$$
$$P^{\langle 2 \rangle}_{a_1^{(1)}a_1^{(2)}} = 4yyx^4 + 2y1x^3 + 2 \cdot 1yx^3 + 1 \cdot 1x^2 = 4y^2x^4 + 4yx^3 + x^2$$
$$P^{\langle 3 \rangle}_{a_1^{(1)}a_1^{(2)}} = 4yyx^4 + 2yx_3x^3 + 2x_3yx^3 + x_3x_3x^2 = 4y^2x^4 + 4yx_3x^3 + x_3^2x^2$$
$$P^{\langle 4 \rangle}_{a_1^{(1)}a_1^{(2)}} = 4x_4x_4x^4 + 2x_41x^3 + 2 \cdot 1x_4x^3 + 1 \cdot 1x^2 = 4x_4^2x^4 + 4x_4x^3 + x^2.$$

Lemma 4. *Consider Π, its initial multiset ω_1, and its ω_1-product P_{ω_1}. Let moreover $i \in [n]$. Then the i-reduction of P_{ω_1} can be calculated in polynomial time in nm.*

Proof. One can see using basic properties of polynomials that

$$P^{\langle i \rangle}_{\omega_1} = \prod_{\ell \in [m]} \mathrm{odp}^{\langle i \rangle}_{a_{i_\ell}},$$

where $P^{\langle i \rangle}_{\omega_1}$ and $\mathrm{odp}^{\langle i \rangle}_{a_{i_\ell}}$ denote the i-reductions of P_{ω_1} and $\mathrm{lodp}_{a_{i_\ell}^{(\ell)}}$, respectively. By Lemma 2, we can compute $\mathrm{odp}_{a_{i_\ell}}$, and in turn $\mathrm{odp}^{\langle i \rangle}_{a_{i_\ell}}$ as well, in polynomial time in n. Moreover, $\mathrm{odp}^{\langle i \rangle}_{a_{i_\ell}}$ contains only at most three variables, x_i, x, and y, for every $\ell \in [m]$. Thus, multiplying these polynomials can be done in polynomial time in nm. $\square$

Using the i-reduction of P_{ω_1} we can compute which and how many objects are sent to the skin during a division driven computation of Π as follows.

Theorem 1. *Consider Π and a division driven halting computation $\mathcal{C} : C_0 \Rightarrow C_1 \Rightarrow \ldots \Rightarrow C_t$ of Π. Let $i \in [n], j \in [0, t-1]$ and denote N_{ij} the number of copies of a_i produced in the skin by dissolutions of elementary membranes during the step $C_j \Rightarrow C_{j+1}$. Then N_{ij} can be computed in polynomial time in nm.*

Proof. Let $P_{\omega_1}^{\langle i \rangle}$ be the i-reduction of P_{ω_1}. Clearly, $P_{\omega_1}^{\langle i \rangle}$ can be written in the form $P_{\omega_1}^{\langle i \rangle} = \sum_{\substack{\mu,\nu \in [0,m], \mu+\nu \le m \\ j \in [0,mn]}} m_{\mu\nu j} x_i^{\mu} y^{\nu} x^{j}$. Using Lemma 3 and the definition of i-reductions, we get the following. A monomial $m_{\mu\nu j} x_i^{\mu} y^{\nu} x^{j}$ in $P_{\omega_1}^{\langle i \rangle}$ represents that there are $m_{\mu\nu j}$ primary m-divider-stable membranes in C_j containing μ copies of a_i and ν copies of such objects different from a_i which can dissolve the membrane. Distinguishing between the cases whether a_i is a dissolver or not, we get the following equations.

$$N_{ij} = \sum_{\substack{\mu,\nu \in [0,m], \nu \ge 1 \\ \mu+\nu \le m}} m_{\mu\nu j} \mu, \qquad (1)$$

if a_i is a non-dissolver, and

$$N_{ij} = \sum_{\substack{\mu,\nu \in [0,m] \\ \mu+\nu \le m}} m_{\mu\nu j} \mu \qquad (2)$$

otherwise. As we have seen in Lemma 4, $P_{\omega_1}^{\langle i \rangle}$ can be computed in polynomial time in nm. Thus, the corresponding (polynomial number of) coefficients of the monomials in the sums (1) and (2) can be calculated in polynomial time in nm as well. □

Example 8. Consider Example 1 and the computation shown in Figure 1. Let, for every $i \in [4], j \in [0,4]$, N_{ij} be the value defined in Theorem 1. Then $N_{ij} = 0$, for $i \in [2], j \in [0,4]$ and $i \in [3,4], j \in [0,2]$. Moreover, $N_{33} = N_{43} = 4$, $N_{34} = 0$, and $N_{44} = 8$.

We show that these values can be calculated using the i-reductions given in Example 7 and the equations (1) and (2) given in the proof of Theorem 1. If $i \in [2]$, then a_i is a non-dissolver, thus we have to use Equation (1) in this case. However, the monomials in $P_{a_1^{(1)} a_1^{(2)}}^{\langle i \rangle}$ do not contain x_i, hence in this case μ is 0, for each monomial. Therefore the sum equals to 0, for every $j \in [0,4]$. Now let $i = 3$. Since a_3 is a non-dissolver, we should use again Equation (1) in this case. Now the only monomial which contains both x_3 and y is $4yx_3x^3$, which means that $N_{33} = 4 \cdot 1 = 4$ and $N_{3j} = 0$, for every $j \in \{0,1,2,4\}$. Lastly, let $i = 4$. Since a_4 is a dissolver, we should use Equation (2) in this case. Now the monomials that contain x_4 are $4x_4^2x^4$ and $4x_4x^3$. Therefore $N_{44} = 4 \cdot 2 = 8$, $N_{43} = 4 \cdot 1 = 4$, and $N_{4j} = 0$, for every $j \in [0,2]$. □

From Theorem 1 we immediately get the following result.

Corollary 1. *Consider Π and a division driven halting computation $\mathcal{C} : C_0 \Rightarrow C_1 \Rightarrow \ldots \Rightarrow C_t$ of Π. Then the multiset content of the skin in C_t can be computed in polynomial time in nm.*

Proof. Let $i \in [n]$. It can be clearly seen that the number N_i of occurrences of a_i in the skin membrane in C_t is $\sum\limits_{j\in[0,t-1]} N_{ij}$, where N_{ij} is the number defined in Theorem 1. Since t is at most nm, using Theorem 1 we get that N_i can be computed in polynomial time in nm. $\square$

4 Conclusions

In this paper we proposed an efficient method for calculating the number of each object occurring in the skin membrane at the end of a division driven computation of a halting sdd P system Π. To calculate these numbers we used multiplications of certain polynomials which were created from the object division polynomials of the objects initially contained in the working membrane of Π.

Although our method considers only division driven computations of halting sdd P systems, we can use it to simulate recognizer P systems too. Recognizer P systems [17] are common tools in membrane computing to solve decision problems with P systems. They have only halting computations and they are confluent, which means that all of their computations yield the same result. That is, a division driven computation gives the same result as that of the other computations.

By definition, sdd P systems have no different rules with the same left-hand side. In fact, we can safely assume that a recognizer P system having only dissolution and division rules possesses this property, too. To see this consider such a recognizer P system Π. If Π has two different rules r_1 and r_2 with the same left-hand side, then there is a computation of Π where in each situation when r_2 is applicable, Π applies r_1 instead (clearly, if r_2 is applicable, then r_1 should be applicable, too). That is, if we remove r_2 from Π, then the remaining part of Π will still compute the same result as before.

Concerning the future work, we would like to extend our method to P systems having other types of rules or different initial membrane structure. The method can easily be extended to the case when the dissolution rules can have arbitrary objects in their right-hand sides. Indeed, in this case we only need to change the calculation of the value N_{ij} in the proof of Theorem 1 accordingly.

To extend the method to send-out communication rules we need to modify first the definition of z in Definition 5 so that it involves also the case when $a_l^{(k)}$ triggers a send-out communication rule. After this we need to incorporate this new case into the calculation of N_{ij} in the proof of Theorem 1.

Moreover, our method seems to be suitable for generalisation to such P systems which initially have more than one working membrane (possibly with different labels). On the other hand, to extend it to such P systems where the initial membrane structure is deeper than one is not so trivial. Consider for example a P system Π having an initial membrane structure of the form $[\ldots[\,[\,]_1\,]_2\ldots]_n$, where $n \geq 3$ and n is the skin. Assume also that the other properties of Π correspond to those of the sdd P systems. Since membranes with label $i > 1$ cannot be divided until membranes with label 1 are present, we could use our method to calculate the number of objects in the regions of Π until the last membrane with label 1 is dissolved. Assume that at this point the elementary membrane has label i, for some $i \in [2, n]$. We can use again our method to calculate the number of objects in the regions of Π until the last membrane with label i is dissolved. Continuing this way the application of our method, we can calculate the number of objects occurring in the skin membrane when the computation of Π halts. However, we cannot assume that the above described computation is efficient because of the following reasons. Consider that point of the computation when the last membrane with label 1 is dissolved and the new elementary membrane is the one with label i. Then this membrane can contain exponentially many objects, which means that to apply our method we should multiply exponentially many polynomials. Nevertheless, it is more or less clear that if Π works in polynomial time, then only a polynomially large number of these objects are used by Π during the computation. This means that we can apply our method taking into consideration only a polynomially large number of objects.

References

1. Alhazov, A., Martín-Vide, C., Pan, L.: Solving a PSPACE-complete problem by P systems with restricted active membranes. Fundamenta Informaticae **58** (2003) 67–77
2. Alhazov, A., Pan, L., Păun, Gh.: Trading polarizations for labels in P systems with active membranes. Acta Informatica **41**(2-3) (2004) 111–144
3. Alhazov, A., Pérez-Jiménez, M.J.: Uniform solution of QSAT using polarizationless active membranes. International Conference on Machines, Computations and Universality (2007) 122-133
4. Cordón-Franco, A., Gutiérrez-Naranjo, M.A., Pérez-Jiménez, M.J., Riscos-Núñez, A.: Exploring computation trees associated with P systems. In: Mauri, G., Paun, Gh., Pérez-Jiménez, M.J., Rozenberg, G., Salomaa, A. (eds.) Membrane Computing, 5th International Workshop, WMC 2004, LNCS vol. 3365 (2005) 278-–286
5. Gazdag, Z.: Solving SAT by P systems with active membranes in linear time in the number of variables. In: Alhazov, A., Cojocaru, S., Gheorghe, M., Rogozhin, Y., Rozenberg, G., Salomaa, A. (eds.) Membrane Computing: 14th International Conference, LNCS vol. 8340 (2014) 189–205
6. Gazdag, Z., Kolonits, G.: A new approach for solving SAT by P systems with active membranes. In: Csuhaj-Varjú, E., Gheorghe, M., Rozenberg, G., Salomaa, A., Vaszil, G. (eds.) Membrane Computing: 13th International Conference, LNCS vol. 7762 (2013) 195–207
7. Gutierrez-Naranjo, M.A., Perez-Jimenez, M.J., Riscos-Núñez, A., Romero-Campero, F.J.: On the power of dissolution in P systems with active membranes. In: Freund, R., Păun, Gh., Rozenberg, G., Salomaa, A. (eds.) Membrane Computing: 6th International Workshop, LNCS vol. 3850 (2006) 224–240

8. Krishna, S.N., Rama, R.: A variant of P systems with active membranes: Solving NP-complete problems. Romanian Journal of Information Science and Technology, 2, 4 (1999) 357–367
9. Leporati, A., Manzoni, L., Mauri, G., Porreca, A.E., Zandron, C.: Solving a special case of the P conjecture using dependency graphs with dissolution. In: Gheorghe, M., Rozenberg, G., Salomaa, A., Zandron, C. (eds.) Membrane Computing: 18th International Conference, LNCS vol. 10725 (2017) 196-213
10. Leporati, A., Zandron, C., Ferretti, C., Mauri, G.: Solving PSPACE-complete problems by polarizationless recognizer P systems with strong division and dissolution. Emerging Paradigms in Informatics, Systems and Communication (2009) 93-98
11. Murphy, N., Woods, D.: Active membrane systems without charges and using only symmetric elementary division characterise P. In: Eleftherakis, G., Kefalas, P., Păun, Gh., Rozenberg, G., Salomaa, A. (eds.) Membrane Computing: 8th International Workshop, LNCS vol. 4860 (2007) 367–384
12. Pan, L., Alhazov, A., Ishdorj, T.-O.: Further remarks on P systems with active membranes, separation, merging, and release rules. Soft Computing **9**(9) (2004) 686–690
13. Păun, Gh.: P systems with active membranes: attacking NP-complete problems. Journal of Automata, Languages and Combinatorics **6**(1) (2001) 75–90
14. Păun, Gh.: Further twenty six open problems in membrane computing. In: Third Brainstorming Week on Membrane Computing. Fénix Editora, Sevilla (2005) 249–262
15. Păun, Gh., Rozenberg, G., Salomaa, A. (eds.): The Oxford Handbook of Membrane Computing. Oxford University Press, Oxford, England (2010)
16. Pérez-Jiménez, M.J., Romero-Campero, F.J.: Trading polarization for bi-stable catalysts in P systems with active membranes. In: Mauri, G., Păun, Gh., Pérez-Jiménez, M.J., Rozenberg, G., Salomaa, A. (eds.) Membrane Computing: 5th International Workshop, LNCS vol. 3365 (2005) 373–388
17. Pérez-Jiménez, M.J., Romero-Jiménez, Á., Sancho-Caparrini, F.: Complexity classes in models of cellular computing with membranes. Natural Computing **2**(3) (2003) 265–285
18. Sosík, P.: The computational power of cell division in P systems. Natural Computing **2**(3) (2003) 287–298
19. Woods, D., Murphy, N., Pérez-Jiménez, M.J., Riscos-Núñez, A.: Membrane dissolution and division in P. In: Calude, C.S., da Costa, J.F.G., Dershowitz, N., Freire, E., Rozenberg, G. (eds.) Unconventional Computation: 8th International Conference, LNCS vol. 5715 (2009) 262–276
20. Zandron, C., Ferretti, C., Mauri, G.: Solving NP-complete problems using P systems with active membranes. In: Unconventional Models of Computation, UMC'2K: Proceedings of the Second International Conference on Unconventional Models of Computation. Springer London, London (2001) 289–301