Journal of Membrane Computing (2019) 1:251–261
https://doi.org/10.1007/s41965-019-00024-z

REGULAR PAPER

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

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

Received: 25 July 2019 / Accepted: 29 October 2019 / Published online: 2 December 2019

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 P systems · Active membranes · Simulation · Polynomials · Matrix multiplication

1 Introduction

P systems with active membranes, introduced in [23], 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 [12, 23, 28, 33], while with non-elementary membrane division they can solve even **PSPACE**-complete problems in polynomial time [1, 30]. Solving computationally hard problems with P systems with active membranes has a huge literature in Membrane Computing, see, e.g. [2, 5, 7, 21, 22, 27, 29, 31], and the references therein.

It is also widely investigated how certain restrictions on P systems with active membrane affect the computation power of these systems (see for example [6, 8, 9, 11, 13, 14, 16, 17, 19, 20, 25]). Probably, the most investigated question in this research line is whether these P systems are still powerful enough to solve hard problems in polynomial time when the polarizations of the membranes are not used. 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 in polynomial time without polarizations. On the other hand, no polynomial-time solutions for hard problems exist when neither polarization nor non-elementary membrane division is 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 [24]. Păun's conjecture, often called the *P conjecture*, has not been proven yet although there are some partial solutions for it (see, e.g. [10, 15, 18, 32]). A direct attempt to calculate efficiently all the elementary membranes of a computation of a P system with active membranes fails as in general, the number of these membranes can grow exponentially and, moreover, these membranes can contain pairwise different multisets. However, it was discovered in [10] 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,

The results of this paper were presented at the 20th International Conference on Membrane Computing, CMC20, Curtea de Argeș, Romania, August 5–8, 2019.

✉ Zsolt Gazdag
gazdag@inf.u-szeged.hu

Gábor Kolonits
kolomax@inf.elte.hu

[1] Department of Foundations of Computer Science, University of Szeged, Szeged, Hungary

[2] Department of Algorithms and Their Applications, Eötvös Loránd University, Budapest, Hungary

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. Using dependency graphs it was shown in [10] that polarizationless P systems with active membranes using no dissolution rules and working in polynomial time can solve only problems in **P**.

If dissolution rules are also allowed, then things become 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 releases to membrane 1 without directly being involved in any application of a rule. Notice that in this case b immediately "knows" that Π contained an occurrence of a in the membrane with label 2. 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 [32], 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 [15] 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 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 to address Păun's conjecture. With this method we are able to simulate efficiently an entire computation of a P system. More precisely, we can compute in polynomial time the multiset content of the skin membrane at the end of certain computations of a P system. To make our ideas as clear as possible, we give our method only for a rather restricted variant of P systems, called *halting simple divide-dissolve* P systems. The computations of these P systems always terminate, initially they have only one elementary membrane in the skin, and they can employ only membrane division and membrane dissolution rules (see Definition 1 for further properties of these P systems). Moreover, we simulate only particular computations of these P systems, where, for example, division rules have priority over dissolution rules. We will call these computations *division-driven* computations (see Definition 2 for further details).

Our approach can be roughly described as follows. Consider a halting simple divide-dissolve P system Π, its input multiset ω_1, and a division-driven computation $\mathcal{C}$ of Π. First, based on the concept of object division graphs, we define *object division polynomials* (Definition 3). The object division polynomial of an object a describes which and how many objects can be created from a using only division rules of Π. Then, we consider a polynomial P_{ω_1} which is, roughly, the multiplication of the object division polynomials of objects in ω_1 (Definition 5). After that, we show that there is a strong relationship between the monomials of P_{ω_1} and the numbers of certain membranes and their contents in $\mathcal{C}$ (Lemma 3). Then, using P_{ω_1}, we show how to calculate in polynomial time which and how many objects are released to the skin membrane in each step of $\mathcal{C}$ (Theorem 1). Using this we conclude that the multiset content of the skin membrane in the last configuration of $\mathcal{C}$ can be computed in polynomial time (Corollary 1).

We believe that our method can be extended to more general variants of P systems. In particular, a generalization to P systems having also send-out communication rules and a more general initial membrane structure seems to be achievable 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. [26]).

$\mathbb{N}$ denotes the set of natural numbers including zero and, for every $i, j \in \mathbb{N}, i \leq j$, $[i, j]$ denotes the set $\{i, \ldots, j\}$. 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 $c, n, j_1, \ldots, j_n \in \mathbb{N}$ and $x_1, \ldots, x_n$ are variables 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*.

Next, we define a variant of polarizationless P systems with active membranes.

Definition 1 A *simple divide-dissolve P system* (*sdd P system*, for short) is a polarizationless P system with active membranes having the following properties. Π is of the form $\Pi = (O, H, \mu, \omega_1, R)$, where

- O is the alphabet of *objects*,

- $H = \{1, s\}$ is the set of *labels* of the membranes,
- $\mu = [[\ ]_1]_s$ is the initial *membrane structure* containing two membranes labelled with 1 and s, respectively, s being the *skin* membrane,
- $\omega_1 \in O^*$ is the *initial multiset* of objects placed in the membrane with label 1 (initially, the skin membrane is empty), and
- R is a finite set of *rules* defined as follows:

  - $[a]_1 \to a$, where $a \in O$
    (*membrane dissolution* rules; in reaction with an object, a membrane can be dissolved, the objects of a dissolved membrane remain in the region surrounding it) and
  - $[a]_1 \to [b]_1[c]_1$, where $a, b, c \in O$
    (*division* rules for elementary membranes; in reaction with an object, the membrane is divided into two membranes with the same label; the object specified in the rule is replaced in the two new membranes by possibly new objects; all other objects are duplicated in the two new copies of the membrane).

Moreover, for every $a \in O$, R has at most one division rule of the form $[a]_1 \to [b]_1[c]_1$. Π is called *halting*, if each of its computations halts.

As it is usual in membrane computing, sdd P systems 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 $\Pi = (O, \{1, s\}, [[\ ]_1]_s, \omega_1, R)$ be a halting sdd P system. 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. We call membranes with label 1 *working membranes* (notice that as the skin cannot be divided or dissolved, the objects in the skin remain unchanged during all computations of Π, that is, only objects in working membranes can be changed). An object $a \in O$ is called a *divider* if a can divide working membranes, that is, R contains a division rule with left-hand side $[a]_1$. Likewise, an object $a \in O$ is called a *dissolver* if R contains a dissolution rule with left-hand side $[a]_1$.

3 Results

In this paper we consider only halting sdd P systems. In the rest of this section Π is always a halting sdd P system of the form $\Pi = (O, \{1, s\}, [[\ ]_1]_s, \omega_1, R)$, where $O = \{a_1, \dots, a_n\}$ ($n \in \mathbb{N}$) and $\omega_1 = a_{i_1} \dots a_{i_m}$ ($m \geq 1$ and $i_1, \dots, i_m \in [n]$).

In this section we show that the multiset content of the skin membrane of Π at the end of so-called *division-driven computations* can be computed in polynomial time in nm. In a division-driven computation 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 $\mathcal{C} : C_0 \Rightarrow C_1 \Rightarrow \cdots \Rightarrow C_t$ 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 2 Let Π be a halting sdd P system with object alphabet $O = \{a_1, \dots, a_n\}$ and initial multiset $\omega_1 = a_{i_1} \dots a_{i_m}$. A halting computation $\mathcal{C}$ of Π is called *division-driven* if the following conditions hold.

1. If Π can apply both division and dissolution rules to a membrane in $\mathcal{C}$, then Π applies a division rule, and
2. when a division rule is applied to a membrane M in $\mathcal{C}$ with an object $a_i^{(\ell)}$ on the left-hand side ($i \in [n], \ell \in [m]$), then M contains no dividers with label $\ell' < \ell$.

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 $\ell_1 < \cdots < \ell_k$, 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-divider object with label ℓ occurs in a working membrane, then this object remains unchanged until the computation halts.

Example 1 Let $\Pi_{ex} = (\{a_1, a_2, a_3, a_4\}, \{1, s\}, [[\ ]_1]_s, a_1^{(1)}a_1^{(2)}, R)$, 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 label 1 and 2, respectively. One can see that this computation is indeed a division-driven one. In the first step, $a_1^{(1)}$ is used to divide the working membrane. Then, in the second step the upper working membrane is divided by $a_2^{(1)}$ and the other working membrane is divided by $a_1^{(2)}$ (no dividers with label 1 are present in this membrane). In the third step, the upper two working membranes are divided by $a_1^{(2)}$, the working membrane containing $a_3^{(1)}a_2^{(2)}$ is divided by $a_2^{(2)}$, and the remaining working membrane is unchanged. Then, the computation continues according to the definition of division-driven computations: dividers have priority over dissolvers and dividers (resp. dissolvers) with label 1 have priority over dividers (resp. dissolvers) with label 2. □

As it is mentioned in the introduction, our work is based on the concept of object division polynomials. To define this concept precisely we first define *object division trees* similarly as *object division graphs* were defined, for example, in [32]. Let Π be a halting sdd P system with object alphabet $O = \{a_1, \ldots, a_n\}$. 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 $[a_j]_1 \to [a_k]_1[a_l]_1$ ($k, l \in [n]$) is a rule of Π, then N has exactly two children with labels a_k and a_l, respectively.

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 Fig. 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}. □

Next we show a useful property of object division trees.

Lemma 1 *Let Π be a halting sdd P system with object alphabet $O = \{a_1, \ldots, a_n\}$ and initial multiset ω_1. Let $i \in [n]$ such that a_i occurs in ω_1. Then $h(\mathrm{odt}_{a_i}) < n$.*

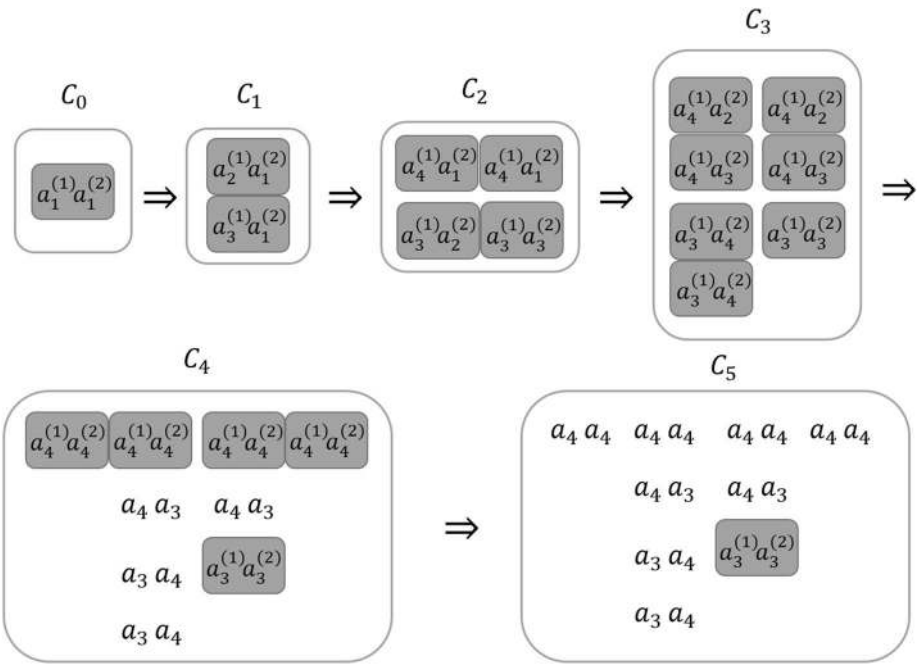

Fig. 1 A division-driven computation of Π_{ex} from Example 1. Grey areas are the working membranes. Working membranes appearing closer to each other are results of a membrane division

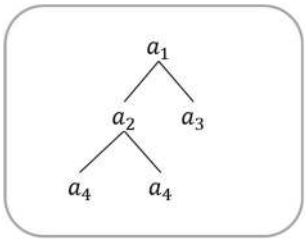

Fig. 2 The tree odt_{a_1} from Example 2

Proof We give an indirect proof. Assume that $h(\text{odt}_{a_i}) \geq n$. 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 $t_1 = t_2 = \text{odt}_{a_j}$. 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. □

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

Definition 3 Consider a halting sdd P system Π with object alphabet $O = \{a_1, \ldots, a_n\}$. Let $V = \{x_i \mid i \in [n]\} \cup \{x\}$ be a set of variables. Let moreover $i \in [n]$ and $l = h(\text{odt}_i)$. The *object division polynomial* of a_i (odp_{a_i} for short) is a polynomial with variables in V defined as follows:

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

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:

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

□

Next we show that object division polynomials can be calculated in polynomial time.

Lemma 2 *Consider a halting sdd P system Π with object alphabet $O = \{a_1, \ldots, a_n\}$ and let $i \in [n]$. Then the object division polynomial of a_i can be computed in polynomial time in n.*

Proof Let $l = h(\text{odt}_{a_i})$ 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 jth level of odt_{a_i}. Let $ndiv = \{j \in [n] \mid a_j \text{ is a non-divider}\}$. 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

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

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

$$(e_k)_\xi = \begin{cases} 1 \text{ if } \xi = k \text{ and } k \in ndiv \\ 0 \text{ otherwise.} \end{cases}$$

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 $\xi \in [n]$, if there is a rule r of Π with a_ξ on the left- and a_k on the right-hand side, then let $(\mathbf{m}_k)_\xi$ be the number of occurrences of a_k on the right-hand side of r. If there is no such rule of Π, then let $(\mathbf{m}_k)_\xi$ 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 (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 kth 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

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

Notice that since the 0th 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

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

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

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

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.

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

□

Let Π be a halting sdd P system with object alphabet $O = \{a_1, \ldots, a_n\}$ and initial multiset $\omega_1 = a_{i_1} \ldots a_{i_m}$. Consider a division-driven computation $\mathcal{C} : C_0 \Rightarrow C_1 \Rightarrow \cdots \Rightarrow C_t$ of Π. First we specify certain working membranes of $\mathcal{C}$, then show how to use object division polynomials to calculate the multiset contents of these working membranes.

Let M be a working membrane in $\mathcal{C}$ and $\ell \in [m]$. If M contains no dividers with label $\ell' \le \ell$, 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. □

Let $\mathcal{C}$ be a division-driven computation of Π. To calculate the multiset contents of primary non-dividing working membranes of $\mathcal{C}$, we extend first the definition of object division polynomials to objects having labels.

Definition 4 Consider a halting sdd P system Π with object alphabet $O = \{a_1, \ldots, a_n\}$ and initial multiset $\omega_1 = a_{i_1} \ldots a_{i_m}$. Let $i \in [n]$, $\ell \in [m]$, and consider the object division polynomial $\mathrm{odp}_{a_i} = \sum_{j\in[0,l],k\in[n]} m_{jk} x_k x^j$ of a_i.

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 added ℓ to the indices of certain variables of odp_{a_i}, referring this way to the label of the corresponding object).

Next we define a product of labelled object division polynomials of objects in ω_1.

Definition 5 Let Π be a halting sdd P system with object alphabet $O = \{a_1, \ldots, a_n\}$ and initial multiset $\omega_1 = a_{i_1} \ldots a_{i_m}$. The *$\omega_1$-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 $\alpha x_{1j_1} \ldots x_{mj_m} x^j$, for some $\alpha, j \in \mathbb{N}$ and $j_1, \ldots, j_m \in [n]$. Using the next lemma we can determine the multiset contents of primary non-dividing working membranes of the computation $\mathcal{C}$ by calculating the monomials of P_{ω_1}.

Lemma 3 *Let Π be a halting sdd P system with object alphabet $O = \{a_1, \dots, a_n\}$ and initial multiset $\omega_1 = a_{i_1} \dots a_{i_m}$. Consider the ω_1-product P_{ω_1} and a division-driven computation $\mathcal{C} : C_0 \Rightarrow C_1 \Rightarrow \cdots \Rightarrow C_t$ 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 non-dividing 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 $m = 1$, then $P_{\omega_1} = \mathrm{lodp}_{a_{i_1}^{(1)}}$. Then, by Definitions 3 and 4, P_{ω_1} contains a monomial $m_{jk} x_{1k} x^j$ if and only if the number of leaves in odt_{a_1} at depth j labelled by a_k is m_{jk}. Thus, the number of those non-dividing working membranes in C_j which contain $a_k^{(1)}$ equals to the coefficient of $x_{1k} x^j$ in P_{ω_1}. Then the statement follows taking into consideration that every non-dividing working membrane in $\mathcal{C}$ is primary.

Now assume that the statement holds if $m = m'$, for some $m' \geq 1$. 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 $a_{j_1}^{(1)} \dots a_{j_{m'}}^{(m')} a_{i_m}^{(m)}$ 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 the ω_1 -product P_{ω_1} to calculate multiset contents of primary non-dividing working membranes.

Example 6 Consider Π_{ex} from Example 1 and the computation $\mathcal{C}$ given in Fig. 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 $\mathrm{lodp}_{a_1^{(2)}} = 2x_{24}x^2 + x_{23}x$. As $P_{a_1^{(1)}a_1^{(2)}} = \mathrm{lodp}_{a_1^{(1)}} \mathrm{lodp}_{a_1^{(2)}}$ we get that

$$\begin{aligned} P_{a_1^{(1)}a_1^{(2)}} &= (2x_{14}x^2 + x_{13}x)(2x_{24}x^2 + x_{23}x) \\ &= 4x_{14}x_{24}x^4 + 2x_{14}x_{23}x^3 + 2x_{13}x_{24}x^3 + x_{13}x_{23}x^2. \end{aligned}$$

Figure 3 shows the correspondence between the monomials of $P_{a_1^{(1)}a_1^{(2)}}$ 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 $a_j^{(i)}$, 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 a halting sdd Π, the ω_1-product P_{ω_1} of Π, and a division-driven 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 released 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

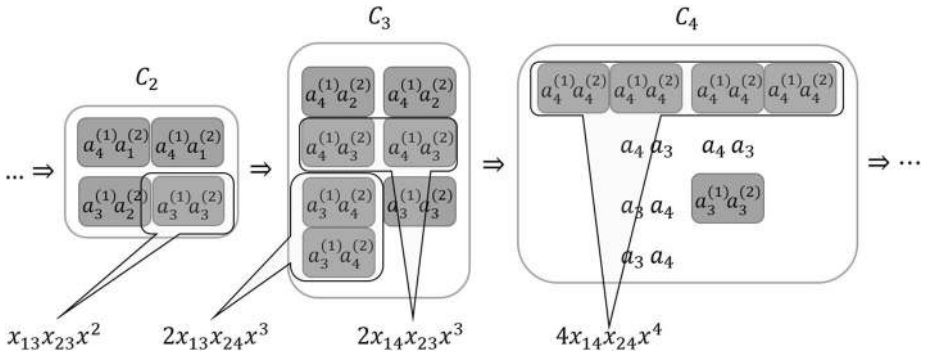

Fig. 3 Representing primary non-dividing working membranes of Π_{ex} by monomials

means that we cannot use P_{ω_1} directly to calculate efficiently the number of these objects. Instead, we will use another polynomial yielded by using the next definition.

Definition 6 Consider a halting sdd P system Π with object alphabet $O = \{a_1, \dots, a_n\}$ and initial multiset $\omega_1 = a_{i_1} \dots a_{i_m}$. 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 using the following operations. First, for every $\ell \in [m], k \in [n]$ with $k \neq i$, we 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' and 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 *Let Π be a halting sdd P system with object alphabet $O = \{a_1, \dots, a_n\}$ and initial multiset $\omega_1 = a_{i_1} \dots a_{i_m}$. Consider the ω_1-product P_{ω_1} of Π. 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. □

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

Theorem 1 *Let Π be a halting sdd P system with object alphabet $O = \{a_1, \dots, a_n\}$ and initial multiset $\omega_1 = a_{i_1} \dots a_{i_m}$. Consider a division-driven computation $C : C_0 \Rightarrow C_1 \Rightarrow \cdots \Rightarrow C_t$ of Π. Let moreover $i \in [n], j \in [0, t-1]$ and denote N_{ij} the number of copies of a_i released to the skin membrane 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_{ω_1} be the ω_1-product of Π and $P^{\langle i \rangle}_{\omega_1}$ be the i-reduction of P_{ω_1}. Clearly, $P^{\langle i \rangle}_{\omega_1}$ can be written in the form

$$P^{\langle i \rangle}_{\omega_1} = \sum_{\substack{\mu, \nu \in [0,m],\ \mu + \nu \leq m \\ j \in [0, mn]}} m_{\mu\nu j} x_i^\mu y^\nu x^j.$$

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 $P^{\langle i \rangle}_{\omega_1}$ 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.

$$N_{ij} = \sum_{\substack{\mu, \nu \in [0,m],\ \nu \geq 1 \\ \mu + \nu \leq m}} m_{\mu\nu j} \mu, \tag{1}$$

if a_i is a non-dissolver, and

$$N_{ij} = \sum_{\substack{\mu, \nu \in [0,m] \\ \mu + \nu \leq m}} m_{\mu\nu j} \mu \tag{2}$$

otherwise. As we have seen in Lemma 4, $P^{\langle i \rangle}_{\omega_1}$ 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 Fig. 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 Eq. (1) in this case. However, the monomials in $P^{\langle i \rangle}_{a_1^{(1)} a_1^{(2)}}$ 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 Eq. (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 Eq. (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 *Let Π be a halting sdd P system with object alphabet $O = \{a_1, \ldots, a_n\}$ and initial multiset $\omega_1 = a_{i_1} \ldots a_{i_m}$. Consider a division-driven 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 $\sum_{j \in [0,t-1]} N_{ij}$, where N_{ij} is the number defined in Theorem 1. By Theorem 1, N_{ij} can be calculated in polynomial time in nm. Thus, to see the statement it is enough to show that $t \leq nm$.

On the one hand, ω_1 contains m objects. On the other hand, by Lemma 1, for every $\ell \in [m]$, $h(\mathrm{odt}_{a_{i_\ell}}) < n$. This means that there are at most $n-1$ steps in $\mathcal{C}$ where objects with label ℓ are used to divide working membranes. Thus there are at most $m(n-1)$ steps where division rules are applied, and there is at most one step, where only dissolution rules are applied. Thus, $t \leq m(n-1) + 1 \leq mn$. □

4 Conclusions

In this paper we proposed a polynomial-time 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 [28] 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 structures. 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.

It seems that we can extend our method to send-out communication rules too. To this end, we need to extend the definition of division-driven computations, for example such that send-out communication rules have less priority than that of dissolution rules. Moreover, in the calculation of N_{ij} in the proof of Theorem 1 we should add a case where a_i is a non-dissolver but can trigger a send-out communication rule. Notice that a working membrane can contain more than one occurrence of such an object a which can trigger send-out communication rules. However, during one step only one a can be used by a rule. Therefore, in the computation of N_{ij} we might need to consider such monomials too which contain $x^{j'}$, for some $j' < j$.

Moreover, our method seems to be suitable for generalisation to such P systems which initially have more than one working membranes (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 our method again 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.

Acknowledgements Open access funding provided by University of Szeged (SZTE). Research of the first author was supported by Grant TUDFO/47138-1/2019-ITM of the Ministry for Innovation and Technology, Hungary. Research of the second author was supported by NKFIH—National Research, Development, and Innovation Office, Hungary, Grant no. K 120558.

References

1. Alhazov, A., Martín-Vide, C., & Pan, L. (2003). Solving a PSPACE-complete problem by P systems with restricted active membranes. *Fundamenta Informaticae*, *58*, 67–77.
2. Alhazov, A., Pan, L., & Păun, G. (2004). Trading polarizations for labels in P systems with active membranes. *Acta Informatica*, *41*(2–3), 111–144.
3. Alhazov, A., & Pérez-Jiménez, M.J. (2007). Uniform solution of QSAT using polarizationless active membranes. In J. Durand-Lose, & M. Margenstern (Eds.), *Machines, Computations, and Universality*, MCU 2007, LNCS vol. 4664, pp. 122–133.
4. Cordón-Franco, A., Gutiérrez-Naranjo, M.A., Pérez-Jiménez, M.J., & Riscos-Núñez, A. (2005). Exploring computation trees associated with P systems. In G. Mauri, G., Paun, M. J. Pérez-Jiménez, G. Rozenberg, & Salomaa, (Eds.), *Membrane Computing*, 5th International Workshop, WMC 2004, LNCS vol. 3365, pp. 278–286.
5. Gazdag, Z. (2014). Solving SAT by P systems with active membranes in linear time in the number of variables. In A. Alhazov, S. Cojocaru, M. Gheorghe, Y. Rogozhin, G. Rozenberg, & A. Salomaa (Eds.), Membrane Computing: 14th International Conference, LNCS, vol. 8340, pp. 189–205.
6. Gazdag, Z., & Gutiérrez-Naranjo, M.A. (2014). Solving the ST-connectivity problem with pure membrane computing techniques. In M. Gheorghe, G., Rozenberg, A., Salomaa, P., Sosík, & C., Zandron, (Eds.), Membrane Computing: 15th International Conference, LNCS vol. 8961, pp. 215–228.
7. Gazdag, Z., & Kolonits, G. (2013). A new approach for solving SAT by P systems with active membranes. In E. Csuhaj-Varjú, M. Gheorghe, G. Rozenberg, A. Salomaa, & G. Vaszil (Eds.), Membrane Computing: 13th International Conference, LNCS vol. 7762, pp. 195–207.
8. Gazdag, Z., & Kolonits, G. (2017). Remarks on the computational power of some restricted variants of P systems with active membranes. In A. Leporati, G. Rozenberg, A. Salomaa, & C. Zandron (Eds.), Membrane Computing, 17th International Conference, LNCS vol. 10105, pp. 209–232.
9. Gazdag, Z., Kolonits, G., & Gutiérrez-Naranjo, M.A. (2014). Simulating Turing machines with polarizationless P systems with active membranes. In M. Gheorghe, G. Rozenberg, A. Salomaa, P. Sosík, & C. Zandron, (Eds.), Membrane Computing: 15th International Conference, LNCS vol. 8961, pp. 229–240.
10. Gutierrez-Naranjo, M.A., Perez-Jimenez, M.J., Riscos-Núñez, A., & Romero-Campero, F.J. (2006). On the power of dissolution in P systems with active membranes. In R. Freund, G. Păun, G. Rozenberg, & A. Salomaa (Eds.), Membrane Computing: 6th International Workshop, LNCS vol. 3850, pp. 224–240.
11. Kolonits, G. (2015). A solution of horn-SAT with P systems using antimatter. In: Membrane Computing: 16th International Conference, LNCS vol. 9504, pp. 236–250.
12. Krishna, S. N., & Rama, R. (1999). A variant of P systems with active membranes: Solving NP-complete problems. *Romanian Journal of Information Science and Technology*, *2*(4), 357–367.
13. Leporati, A., Ferretti, C., Mauri, G., & Pérez-Jiménez, M. J. (2009). Complexity aspects of polarizationless membrane systems. *Natural Computing: An International Journal*, *8*(4), 703–717.
14. Leporati, A., Manzoni, L., Mauri, G., Porreca, A.E., & Zandron, C. (2014). Simulating elementary active membranes, with an application to the P conjecture. In M. Gheorghe, G. Rozenberg, P. Sosík, & C. Zandron, (Eds.), Membrane Computing, 15th International Conference, LNCS vol. 8961, pp. 284–299.
15. Leporati, A., Manzoni, L., Mauri, G., Porreca, A.E., & Zandron, C. (2017). Solving a special case of the P conjecture using dependency graphs with dissolution. In M. Gheorghe, G. Rozenberg, A. Salomaa, & C. Zandron, (Eds.), Membrane Computing: 18th International Conference, LNCS vol. 10725, pp. 196–213.
16. Leporati, A., Manzoni, L., Mauri, G., Porreca, A. E., & Zandron, C. (2018). Solving QSAT in Sublinear Depth. In: Membrane Computing: 19th International Conference, LNCS vol. 11399, pp. 188–201.
17. Leporati, A., Zandron, C., Ferretti, C., & Mauri, G. (2009). Solving PSPACE-complete problems by polarizationless recognizer P systems with strong division and dissolution. Emerging Paradigms in Informatics, Systems and Communication, pp. 93–98.
18. Murphy, N., & Woods, D. (2007). Active membrane systems without charges and using only symmetric elementary division characterise P. In G. Eleftherakis, P. Kefalas, G. Păun, G. Rozenberg, & A. Salomaa, (Eds.), Membrane Computing: 8th International Workshop, LNCS vol. 4860, pp. 367–384.
19. Murphy, N., & Woods, D. (2008). A Characterisation of NL using membrane systems without charges and dissolution. In C.S. Calude, J.F.G. da Costa, R. Freund, M. Oswald, & G. Rozenberg, (Eds.), Unconventional Computing: 7th International Conference, LNCS vol. 5204, pp. 164–176.
20. Murphy, N., & Woods, D. (2011). The computational power of membrane systems under tight uniformity conditions. *Natural Computing: An International Journal*, *10*(1), 613–632.
21. Pan, L., & Alhazov, A. (2006). Solving HPP and SAT by P systems with active membranes and separation rules. *Acta Informatica*, *43*(2), 131–145.
22. Pan, L., Alhazov, A., & Ishdorj, T.-O. (2004). Further remarks on P systems with active membranes, separation, merging, and release rules. *Soft Computing*, *9*(9), 686–690.
23. Păun, G. (2001). P systems with active membranes: attacking NP-complete problems. *Journal of Automata, Languages and Combinatorics*, *6*(1), 75–90.
24. Păun, Gh. (2005). Further twenty six open problems in membrane computing. In: Third Brainstorming Week on Membrane Computing, pp. 249–262. Fénix Editora, Sevilla.

25. Porreca, A.E., Leporati, A., Mauri, G., & Zandron, C. (2012). Sublinear-space P systems with active membranes. In E. Csuhaj-Varjú, M. Gheorghe, G. Rozenberg, A. Salomaa, & G. Vaszil, (Eds.), Membrane Computing 13th International Conference, LNCS vol. 7762, pp. 342–357.
26. Păun, G., Rozenberg, G., & Salomaa, A. (Eds.). (2010). *The Oxford Handbook of Membrane Computing*. Oxford, England: Oxford University Press.
27. Pérez-Jiménez, M.J., & Romero-Campero, F.J. (2005). Trading polarization for bi-stable catalysts in P systems with active membranes. In G. Mauri, G. Păun, M.J. Pérez-Jiménez, G. Rozenberg, & A. Salomaa, (eds.) Membrane Computing: 5th International Workshop, LNCS vol. 3365, pp. 373–388.
28. Pérez-Jiménez, M. J., Romero-Jiménez, Á., & Sancho-Caparrini, F. (2003). Complexity classes in models of cellular computing with membranes. *Natural Computing*, *2*(3), 265–285.
29. Pérez-Jiménez, M. J., Romero-Jiménez, Á., & Sancho-Caparrini, F. (2006). A polynomial complexity class in P systems using membrane division. *Journal of Automata, Languages and Combinatorics*, *11*(4), 423–434.
30. Sosík, P. (2003). The computational power of cell division in P systems. *Natural Computing*, *2*(3), 287–298.
31. Sosík, P., & Rodríguez-Patón, A. (2007). Membrane computing and complexity theory: A characterization of PSPACE. *Journal of Computer and System Sciences*, *73*(1), 137–152.
32. Woods, D., Murphy, N., Pérez-Jiménez, M.J., & Riscos-Núñez, A. (2009). Membrane dissolution and division in P. In C.S. Calude, J.F.G. da Costa, N. Dershowitz, E. Freire, & G. Rozenberg (Eds.), Unconventional Computation: 8th International Conference, LNCS vol. 5715, pp. 262–276.
33. Zandron, C., Ferretti, C., & Mauri, G. (2001). Solving NP-complete problems using P systems with active membranes. *Unconventional Models of Computation* (pp. 289–301), UMC'2K: Proceedings of the Second International Conference on Unconventional Models of Computation, London: Springer.

Zsolt Gazdag is an associate professor at the Institute of Informatics, University of Szeged (Hungary). He received his Ph.D. degree in Mathematics and Computer Science from the University of Szeged in 2007. His research interests include the computational complexity of P systems with active membranes, the expressive power of grammars with controlled derivations, and the theory of weighted tree automata.

Gábor Kolonits is currently an assistant lecturer at the Faculty of Informatics, Eötvös Loránd University, Budapest, Hungary. He finished his B.Sc. and M.Sc. in Computer Engineering at Eötvös Loránd University in 2009 and 20012, respectively. His current main focus is on exploring the computational power of restricted variants of P systems with active membranes. His research interests include discrete mathematics and bio-inspired computations.