Efficient solutions to hard computational problems by P systems with symport/antiport rules and membrane division

Bosheng Song [a,b], Mario J. Pérez-Jiménez [b], Linqiang Pan [a]

[a] Key Laboratory of Image Information Processing and Intelligent Control, School of Automation, Huazhong University of Science and Technology, Wuhan 430074, Hubei, China
[b] Research Group on Natural Computing, Department of Computer Science and Artificial Intelligenc, University of Sevilla, Avda. Reina Mercedes s/n, 41012 Sevilla, Spain

Keywords:
Cell-like P system
Symport/Antiport rule
Membrane division
Subset Sum problem
QSAT problem

A B S T R A C T

P systems are computing models inspired by some basic features of biological membranes. In this work, membrane division, which provides a way to obtain an exponential workspace in linear time, is introduced into (cell-like) P systems with communication (symport/antiport) rules, where objects are never modified but they just change their places. The computational efficiency of this kind of P systems is studied. Specifically, we present a (uniform) linear time solution to the **NP**-complete problem, `Subset Sum` by using division rules for elementary membranes and communication rules of length at most 3. We further prove that such P system allowing division rules for non-elementary membranes can efficiently solve the **PSPACE**-complete problem, `QSAT` in a uniform way.

1. Introduction

Membrane computing is a flexible and versatile branch of natural computing, which arises as an abstraction of the compartmentalized structure of living cells, and the way biochemical substances are processed in (or moved between) membrane bounded regions Păun (2000). Membrane computing is a paradigm providing computing devices called *P systems*, which are parallel, non-deterministic and distributed computational models. Inspired by that structure, two main classes of P systems have been investigated: (a) a hierarchical arrangement of membranes as in a cell Păun (2000), processing information by multisets of symbols; and (b) a net of processor units placed in the nodes of a directed graph, inspired by the cell intercommunication in tissues Martín-Vide et al. (2003) or inspired by the way neurons communicate with each other by means of short electrical impulses, identical in shape (voltage), but emitted at precise moments of time Ionescu et al. (2006). These models have two interesting properties: the capability to solve as many problems as a Turing machine (computational completeness) and the ability to solve computationally hard problems in a feasible time, by trading space for time (computational efficiency). General information on membrane computing can be found in Păun et al. (2010), Frisco (2009), and for the most up-to-date references, one can refer to the P systems website http://ppage.psystems.eu.

A basic P system is based on a cell-like arrangement of membranes, that are placed in a nested hierarchical structure; each membrane delimits a *compartment* (also called *region*) where multisets of *objects* and *rules* for evolving these objects are placed. A membrane with no compartments inside is called *elementary*, otherwise it is called *non-elementary*. The outmost membrane is called a *skin membrane*, the space outside the skin membrane is called the *environment*. There are two main types of evolution rules of cell-like P systems associated with membranes: *multiset rewriting rules* and *communication (symport/antiport) rules*. The present work focuses on a class of P systems with symport/antiport rules, which were proposed in Păun and Păun (2002). Symport rules are of form (u, in) or (u, out), and move objects of multiset u through a membrane; antiport rules are of form $(u, out; v, in)$, and move the objects of multiset u outside a membrane while moving the objects of multiset v inside.

Cell-like P systems have been studied widely. Up to now, researchers have proposed lots of classes of cell-like P systems, and many of them have been proved to be computationally complete (see, e.g., Alhazov and Freund (2005), Alhazov et al. (2006),

Bernardini and Gheorghe (2003), Ciobanu et al. (2007), Păun and Păun (2002), Păun et al. (2005)). The computational efficiency of cell-like P systems has also been investigated Păun et al. (2010). A particularly interesting class of cell-like P systems is that of P systems with active membranes Păun (2000), which contain object evolution rules, *in* communication rules, *out* communication rules, dissolving rules and membrane division rules. Membrane division rules can generate an exponential workspace in polynomial time (even in linear time). Therefore, P systems with active membranes can be used to solve computationally hard problems by a time-space trade-off Păun (2001), Pérez-Jiménez and Riscos-Núñez (2004), Pérez-Jiménez and Riscos-Núñez (2005).

In this work, membrane division is introduced into cell-like P systems, and they evolve by means of symport/antiport or division rules, that is, we present a class of *P systems with symport/antiport rules and membrane division*, where objects are never modified but they just change their places, moreover, objects can be communicated with the environment only through the skin membrane. The computational efficiency of such kind of P systems is studied. Specifically, we present polynomial time solutions to the `Subset Sum` problem by using division rules for elementary membranes and for the `QSAT` problem by using division rules for non-elementary membranes. In both cases, the solutions are uniform and they are obtained by using communication rules of length at most 3.

2. Preliminaries

In this Section, we only introduce some basic notions and notations from formal languages theory. Readers can refer to Rozenberg and Salomaa (1997) for details.

An *alphabet* Σ is a non-empty set and their elements are called *symbols*. An ordered finite sequence of symbols over Σ forms a *string*. The *length* of a string u, denoted by $|u|$, is the number of occurrences of symbols it contains. We denote by Σ^* the set of all strings over Σ. The empty string (with length 0) is denoted by λ, and by $\Sigma^+ = \Sigma^* \setminus \{\lambda\}$ we denote the set of non-empty strings.

For an alphabet Σ, a *multiset* over Σ is a pair (Σ, f) where $f : \Sigma \to \mathbb{N}$ is a mapping, $\mathbb{N}$ is the set of natural numbers. If $m = (\Sigma, f)$ is a multiset, then its *support* is defined as $supp(m) = \{x \in \Sigma | f(x) > 0\}$. If $supp(m) = \{a_1, \ldots, a_k\}$ then we denote $m = \{a_1^{f(a_1)}, \ldots, a_k^{f(a_k)}\}$. A multiset is finite if its support is a finite set. We denote by $\emptyset$ the empty multiset and by $M_f(\Sigma)$ the set of all finite multisets over Σ. If $m_1 = (\Sigma, f_1)$, $m_2 = (\Sigma, f_2)$ are multisets over Σ, then the union of m_1 and m_2, denoted by $m_1 + m_2$, is the multiset (Σ, g), where $g(x) = f_1(x) + f_2(x)$ for each $x \in \Sigma$; the relative complement of m_2 in m_1, denoted by $m_1 \setminus m_2$, is the multiset (Σ, g), where $g(x) = f_1(x) - f_2(x)$ if $f_1(x) \geq f_2(x)$, and $g(x) = 0$ otherwise.

3. P systems with symport/antiport rules and membrane division

In this Section, we study P systems with symport/antiport rules, a class of computing devices aiming to abstract the active transport of molecules across the membranes. In this kind of P systems, objects are processed in a pure communicative way, that is, objects do not evolve and only change their places during the computation process. Membrane division rules are allowed in such P systems.

Definition 1. A P system with symport/antiport rules and membrane division of degree $q \geq 1$ is a tuple $\Pi = (\Gamma, \mathcal{E}, \mu, \mathcal{M}_1, \ldots, \mathcal{M}_q, \mathcal{R}_1, \ldots, \mathcal{R}_q, i_{out})$, where:

- Γ is an alphabet (the working alphabet) and $\mathcal{E} \subseteq \Gamma$;
- μ is a rooted tree with q nodes labeled by $1, \ldots, q$;
- $\mathcal{M}_i$, $1 \leq i \leq q$, are finite multisets over Γ;
- $\mathcal{R}_i$, $1 \leq i \leq q$, are finite sets of rules of the following forms:
  - (a_1) Symport rules: (u, out) or (u, in), for $u \in M_f(\Gamma)$, $|u| > 0$;
  - (a_2) Antiport rules: $(u, out; v, in)$, for $u, v \in M_f(\Gamma)$, $|u| > 0$, $|v| > 0$;
  - (b) Division rules: $[a]_i \to [b]_i[c]_i$, for $i \in \{2, \ldots, q\}$, $i \neq i_{out}$, $a, b, c \in \Gamma$;
- $i_{out} \in \{0, 1, \ldots, q\}$.

A P system with symport/antiport rules and membrane division of degree $q \geq 1$ can be viewed as a set of q membranes labeled by $1, \ldots, q$, arranged in a hierarchical structure of a rooted tree (the root labeled by 1 is called the skin membrane), such that: (a) $\mathcal{M}_1, \ldots, \mathcal{M}_q$ represent the finite multisets of objects initially placed in the q membranes of the system; (b) $\mathcal{E}$ is the set of objects initially located in the environment of the system, all of them available in an arbitrary number of copies; (c) $\mathcal{R}_1, \ldots, \mathcal{R}_q$ are finite sets of rules ($\mathcal{R}_i$ corresponds to the membrane i of μ); (d) i_{out} is a distinguished region which will encode the output of the system. We use the term region i ($0 \leq i \leq q$) to refer to membrane i in case $1 \leq i \leq q$ and to refer to the environment in case $i = 0$. The length of a symport rule (u, out) or (u, in) (an antiport rule $(u, out; v, in)$, respectively) is defined as $|u|$ ($|u| + |v|$, respectively).

For each membrane $i \in \{2, \ldots, q\}$, we denote by $p(i)$ the parent of membrane i in the rooted tree μ, the "parent" of the skin membrane is the environment, denoted by $p(1) = 0$.

A *configuration* of such a P system at any moment is described by the current membrane structure (the rooted tree), together with all multisets of objects over Γ associated with the regions of this membrane structure and the multiset of objects over $\Gamma \setminus \mathcal{E}$ associated with the environment at that moment. The *initial configuration* of $(\Gamma, \mathcal{E}, \mu, \mathcal{M}_1, \ldots, \mathcal{M}_q, \mathcal{R}_1, \ldots, \mathcal{R}_q, i_{out})$ is $(\mu, \mathcal{M}_1, \ldots, \mathcal{M}_q; \emptyset)$.

A symport rule $(u, out) \in \mathcal{R}_i$ is applicable to a configuration at a moment if membrane i contains multiset u at that moment. When such a rule is applied, multiset u is sent to the parent of membrane i. A symport rule $(u, in) \in \mathcal{R}_i$ is applicable to a configuration at a moment if the parent of membrane i contains multiset u at that moment. When such a rule is applied, multiset u enters membrane i from the parent of i.

An antiport rule $(u, out; v, in) \in \mathcal{R}_i$ is applicable to a configuration at a moment if membrane i contains multiset u and its parent contains multiset v at that moment. When such a rule is applied, multiset u from membrane i is sent out of membrane i, and simultaneously, multiset v enters region i from the parent of i.

A division rule $[a]_i \to [b]_i[c]_i$ is applicable to a configuration at a moment if the following conditions hold at that moment: (1) membrane i contains object a; (2) membrane i is neither the skin membrane nor the output membrane. When applying such a rule, membrane i is divided into two membranes with the same label: in the first copy, object a is replaced by object b, in the second one object a is replaced by object c, and all the objects in the original membrane, different from the object triggering the rule, are replicated in the two new membranes. Besides, if membrane i is a non-elementary membrane, then all membranes inside membrane i as well as objects contained in them, will be replicated in each of the new membranes.

The rules of a P system with symport/antiport rules and membrane division are used in a maximally parallel way. At each step, a maximal multiset of rules is applied (no further rule can be added being applicable) with the following restriction: when a membrane is divided, the division rule is the only one which is applied for that membrane at that step, the objects inside that membrane do not evolve by means of communication rules. The objects in the new membranes resulting from division could participate in the interaction with the objects in the (upper or lower) neighbor of membranes by means of communication rules at the next step if these new membranes are not divided once again.

Starting from the initial configuration and applying rules as described above, one obtains a sequence of consecutive

configurations. Each passage from a configuration $\mathcal{C}$ to a next configuration $\mathcal{C}'$ is called a *transition* and denoted by $\mathcal{C} \Rightarrow \mathcal{C}'$. A configuration is a *halting configuration* if no rule of the system is applicable to it. A *computation* of such a P system is a (finite or infinite) sequence of transitions between configurations such that: (1) the first term of the sequence is the initial configuration; (2) each non-first term of the sequence is obtained from the previous configuration by applying rules in a maximally parallel way with the above mentioned restriction; (3) if the sequence is finite, then the last term of the sequence is a halting configuration, and such a computation is called a *halting computation*. Only halting computations give a result, encoded by the multiset of objects present in the output region i_{out}.

3.1. Recognizer P systems with symport/antiport rules and membrane division

Recognizer P systems were introduced in Pérez-Jiménez et al. (2006), as a natural framework to solve decision problems.

Definition 2. A recognizer P system with symport/antiport rules and membrane division of degree $q \geq 1$ is a tuple

$$\Pi = (\Gamma, \mathcal{E}, \Sigma, \mu, \mathcal{M}_1, \ldots, \mathcal{M}_q, \mathcal{R}_1, \ldots, \mathcal{R}_q, i_{in}, i_{out}),$$

such that:

- $(\Gamma, \mathcal{E}, \mu, \mathcal{M}_1, \ldots, \mathcal{M}_q, \mathcal{R}_1, \ldots, \mathcal{R}_q, i_{out})$ is a P system with symport/antiport rules and membrane division of degree $q \geq 1$;
- Γ has two distinguished objects `yes` and `no`, with at least one copy of them present in some multisets $\mathcal{M}_1, \ldots, \mathcal{M}_q$, but none of them present in $\mathcal{E}$;
- Σ is an (input) alphabet strictly contained in Γ, and such that $\mathcal{E} \subseteq \Gamma \setminus \Sigma$;
- $\mathcal{M}_1, \ldots, \mathcal{M}_q$ are finite multisets over $\Gamma \setminus \Sigma$;
- $i_{in} \in \{1, \ldots, q\}$ is the input membrane, and $i_{out} = 0$;
- all computations halt;
- if $\mathcal{C}$ is a computation of Π, then either object `yes` or object `no` (but not both) must have been released into the environment, and only at the last step of the computation.

For each finite multiset $w \in M_f(\Sigma)$, the *computation* of a P system with symport/antiport rules and membrane division with input w starts from a configuration of the form $(\mu, \mathcal{M}_1, \ldots, \mathcal{M}_{i_{in}} + w, \ldots, \mathcal{M}_q, \emptyset)$, where the input multiset w is added to the contents of the input membrane i_{in}.

We denote by $\mathbf{CD}_e\mathbf{C}(k)$ ($\mathbf{CD}_{ne}\mathbf{C}(k)$, respectively) the class of recognizer P systems with division rules for elementary membranes (non-elementary membranes, respectively) whose symport/antiport rules have length at most k.

3.2. Polynomial complexity classes of recognizer P systems

Next, we define the concept of polynomial time solvability (in a uniform way) by means of family of P systems (see Pérez-Jiménez (2005) for details).

Definition 3. A decision problem $X = (I_X, \theta_X)$ is solvable in polynomial time by a family $\mathbf{\Pi} = \{\Pi(n) \mid n \in \mathbb{N}\}$ of recognizer P systems from $\mathbf{CD}_e\mathbf{C}(k)$ or $\mathbf{CD}_{ne}\mathbf{C}(k)$, in a uniform way, if the following conditions hold:

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

We denote by $\mathbf{PMC}_{\mathbf{CD}_e\mathbf{C}(k)}$ ($\mathbf{PMC}_{\mathbf{CD}_{ne}\mathbf{C}(k)}$, respectively) the set of all decision problems which can be solved in a uniform way and polynomial time by means of recognizer P systems from $\mathbf{CD}_e\mathbf{C}(k)$ ($\mathbf{CD}_{ne}\mathbf{C}(k)$, respectively).

4. Solving the `Subset Sum` problem by using $\mathbf{CD}_e\mathbf{C}(3)$

`Subset Sum` problem is a numerical **NP**-complete problem, which has been investigated widely in membrane computing. Specifically, different (uniform) polynomial time solutions have been provided by using families of P systems with active membranes Pérez-Jiménez and Riscos-Núñez (2005), P systems with membrane creation Gutiérrez-Naranjo et al. (2005), and tissue P systems with cell division Díaz-Pernil et al. (2007). Next, we present a solution to the `Subset Sum` problem by combining the symport/antiport rules used in tissue-like P systems and the membrane division of P systems with active membranes.

The `Subset Sum` problem is described as follows: *Given a finite set A, a weight function, $w : A \to \mathbb{N}$, and a constant $k \in \mathbb{N}$, determine whether or not there exists a subset $B \subseteq A$, such that $w(B) = k$.*

Let $g : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ be the function defined by $g(n, k) = ((n+k)(n+k+1)/2) + n$. It is a primitive recursive and bijective function. We denote $g(n, k) = \langle n, k \rangle$.

For each $(n, k) \in \mathbb{N} \times \mathbb{N}$, we consider the recognizer P system from $\mathbf{CD}_e\mathbf{C}(3)$

$$\Pi(\langle n, k \rangle) = (\Gamma, \mathcal{E}, \Sigma, \mu, \mathcal{M}_1, \mathcal{M}_2, \mathcal{R}_1, \mathcal{R}_2, i_{in}, i_{out}),$$

defined as follows:

- $\Sigma = \{q\} \cup \{v_i \mid 1 \leq i \leq n\}$;
- $\mathcal{E} = \{\alpha_i \mid 1 \leq i \leq n + \lceil \log n \rceil + \lceil \log(k+1) \rceil + 7\} \cup \{a_i \mid 1 \leq i \leq n + \lceil \log n \rceil + \lceil \log(k+1) \rceil + 1\} \cup \{b_i \mid 1 \leq i \leq n + \lceil \log n \rceil + \lceil \log(k+1) \rceil + 2\} \cup \{c_i \mid 1 \leq i \leq n + \lceil \log n \rceil + \lceil \log(k+1) \rceil + 4\} \cup \{d_{i,j,l} \mid 1 \leq i \leq n,\ 1 \leq j \leq \lceil \log(k+1) \rceil,\ 1 \leq l \leq n + j + 1\} \cup \{e, p\}$;
- $\Gamma = \Sigma \cup \mathcal{E} \cup \{A_i, B_i \mid 1 \leq i \leq n\} \cup \{g, h, m, n, z, yes, no\}$;
- $\mu = [[\,]_2]_1$;
- $\mathcal{M}_1 = \{d_{1,1,1}, \ldots, d_{1,\lceil \log(k+1) \rceil, 1}, d_{2,1,1}, \ldots, d_{2,\lceil \log(k+1) \rceil, 1}, \ldots, d_{n,1,1}, \ldots, d_{n,\lceil \log(k+1) \rceil, 1}, a_1, b_1, c_1, n, \alpha_1, yes, no\}$; and $\mathcal{M}_2 = \{g, h, m, A_1, \ldots, A_n\}$;
- $i_{in} = 2$ is the input membrane; and $i_{out} = 0$ is the output region;
- The set $\mathcal{R}_1$ consists of the following rules:
  - $r_{1,i} \equiv (\alpha_i, out; \alpha_{i+1}, in)$, $1 \leq i \leq n + \lceil \log n \rceil + \lceil \log(k+1) \rceil + 6$.
  - $r_{2,i} \equiv (a_i, out; a_{i+1}^2, in)$, $1 \leq i \leq n + \lceil \log n \rceil + \lceil \log(k+1) \rceil$.
  - $r_{3,i} \equiv (b_i, out; b_{i+1}^2, in)$, $1 \leq i \leq n$.
  - $r_{4,i} \equiv (b_i, out; b_{i+1}, in)$, $n+1 \leq i \leq n + \lceil \log n \rceil + \lceil \log(k+1) \rceil + 1$.
  - $r_{5,i} \equiv (c_i, out; c_{i+1}^2, in)$, $1 \leq i \leq n$.
  - $r_{6,i} \equiv (c_i, out; c_{i+1}, in)$, $n+1 \leq i \leq n + \lceil \log n \rceil + \lceil \log(k+1) \rceil + 3$.
  - $r_{7,i,j,l} \equiv (d_{i,j,l}, out; d_{i,j,l+1}^2, in)$ $\begin{cases} 1 \leq i \leq n, & 1 \leq j \leq \lceil \log(k+1) \rceil \\ 1 \leq l \leq n + j \end{cases}$
  - $r_8 \equiv (a_{n+\lceil \log n \rceil + \lceil \log(k+1) \rceil + 1}, out; p, in)$.
  - $r_9 \equiv (b_{n+\lceil \log n \rceil + \lceil \log(k+1) \rceil + 2}, out; e, in)$.
  - $r_{10} \equiv (m\, n\, yes, out)$.
  - $r_{11} \equiv (n\, \alpha_{n+\lceil \log n \rceil + \lceil \log(k+1) \rceil + 7}\, no, out)$.
- The set $\mathcal{R}_2$ consists of the following rules:
  - $r_{12,i} \equiv [A_i]_2 \to [B_i]_2[z]_2$, $1 \leq i \leq n$.
  - $r_{13,i} \equiv (B_i, out; d_{i,1,n+2}^2, in)$, $1 \leq i \leq n$.
  - $r_{14,i,j,l} \equiv (d_{i,j,l}, out; d_{i,j+1,l+1}^2, in)$ $\begin{cases} 1 \leq i \leq n, & 1 \leq j \leq \lceil \log(k+1) \rceil - 1 \\ n+2 \leq l \leq n + \lceil \log(k+1) \rceil \end{cases}$
  - $r_{15,i} \equiv (d_{i,\lceil \log(k+1) \rceil, n+\lceil \log(k+1) \rceil + 1} v_i, out; p, in)$, $1 \leq i \leq n$.
  - $r_{16} \equiv (p\, q, out)$.
  - $r_{17} \equiv (g, out; e, in)$.

$r_{18} \equiv (e\,p, out)$.
$r_{19} \equiv (e\,q, out)$.
$r_{20} \equiv (h, out\,; c_{n+\lceil \log n\rceil+\lceil \log(k+1)\rceil+4}, in)$.
$r_{21} \equiv (e\,m\,c_{n+\lceil \log n\rceil+\lceil \log(k+1)\rceil+4}, out)$.

The computation process can be divided in the following phases.

- *Generation phase*: at the first n steps, all possible subsets of A are generated by applying membrane division rules to membranes with label 2. Simultaneously, 2^n copies of objects $d_{i,j,n+1}, a_{n+1}, b_{n+1}, c_{n+1}$ are produced in membrane 1, and the counter α_i evolves in membrane 1.
- *Pre-checking phase*: after the generation phase, there are 2^n copies of membrane 2, each of them containing a subset of A. Then as many copies of object p as the weight of the corresponding subset are introduced in membrane 2. To this aim, we should introduce enough copies of a_i in membrane 1 first, then as many copies of p as object $a_{n+\lceil \log n\rceil+\lceil \log(k+1)\rceil+1}$ will be introduced in membrane 1.
- *Checking phase*: in each membrane with label 2, the number of copies of objects p and q are compared by sending all possible pairs (p, q) to membrane 1. After doing that, a membrane with label 2 will encode a subset of A whose weight is equal to k if and only if no object p or q remains in that membrane.
- *Output phase*: after the checking phase, the system sends an *affirmative answer* to the environment if there exists at least one membrane with label 2 without objects p or q; or a *negative answer* if each membrane with label 2 contains some object(s) p or q.

4.1. An overview of the computation

First, we consider a pair (cod, s) of polynomial time functions over the set of instances of the `Subset Sum` problem. For that, let $u = (A, w, k)$ be an instance of the problem and $A = \{z_1, z_2, \ldots, z_n\}$. Then,

- $cod(u) = \{v_i^j \mid w(z_i) = j \wedge 1 \le i \le n\} \cup \{q^k\}_{,}$ where v_i^j (j copies of object v_i) represents that j is the weight of element z_i.
- $s(u) = \langle n, k\rangle$.

In this situation, instance $u = (A, w, k)$ will be processed by system $\Pi(s(u))$ with input multiset $cod(u)$. In what follows, we describe informally how the recognizing P system $\Pi(s(u))$ from $\mathbf{CD}_e\mathbf{C}(3)$ with input $cod(u)$ works.

At the initial configuration, we have objects $a_1, b_1, c_1, n, \alpha_1, d_{1,1,1}, \ldots, d_{1,\lceil \log(k+1)\rceil,1}, d_{2,1,1}, \ldots, d_{2,\lceil \log(k+1)\rceil,1}, \ldots, d_{n,1,1}, \ldots, d_{n,\lceil \log(k+1)\rceil,1}$, `yes,no` in membrane 1, objects $A_1, \ldots, A_n, g, h, m, cod(u)$ in membrane 2.

The generation phase spends n steps and the division rules $r_{12,i}$ are applied in membrane 2 producing 2^n copies of membrane 2, each of them contains a subset of A; simultaneously, in membrane 1, by using rules $r_{2,i}, r_{3,i}, r_{5,i}, r_{7,i,j,l}$, objects $a_i, b_i, c_i, d_{i,j,l}$ will be duplicated until getting 2^n copies in exactly n steps; besides, by applying rule $r_{1,i}$, the counter object α_i increases its subscript by 1 at each step. Rule $r_{1,i}$ will be applied in the whole computation process except for the last step in the case that the answer is negative. Note that object z is an idle object in cells with label 2.

The pre-checking phase starts at step $n+1$. In this phase, we should introduce enough copies of object p in each membrane 2, but those objects are first introduced in membrane 1. Specifically, at step $n+1$, 2^{n+1} copies of objects $d_{i,j,n+2}$ are produced in membrane 1 by applying rules $r_{7,i,j,n+1}$ ($1 \le i \le n$, $1 \le j \le \lceil \log(k+1)\rceil$). At next step, by using rules $r_{13,i}$, one copy of object B_i in each membrane 2 is exchanged with two copies of object $d_{i,1,n+2}$ in membrane 1 (at that step, in all membranes 2 there are at most 2^n copies of object B_i, and in membrane 1, there are 2^{n+1} copies of object $d_{i,1,n+2}$). Besides, at step $n+2$, objects $d_{i,j,n+2}$ ($1 \le i \le n$, $2 \le j \le \lceil \log(k+1)\rceil$) are duplicated by using rules $r_{7,i,j,n+2}$, and 2^{n+2} copies of objects $d_{i,j,n+3}$ are produced in membrane 1. At step $n+3$, objects $d_{i,j,n+3}$ ($1 \le i \le n$, $3 \le j \le \lceil \log(k+1)\rceil$) are duplicated by using rules $r_{7,i,j,n+3}$, and 2^{n+3} copies of objects $d_{i,j,n+4}$ are produced in membrane 1. Simultaneously, each copy of object $d_{i,1,n+2}$ in all membranes 2 is exchanged with two copies of object $d_{i,2,n+3}$ in membrane 1 (at that step, in all membranes 2 there are at most 2^{n+1} copies of object $d_{i,1,n+2}$, and in membrane 1, there are 2^{n+2} copies of object $d_{i,2,n+3}$). By applying rules $r_{7,i,j,l}$ and $r_{14,i,j,l}$ as many times as possible, at step $n+\lceil \log(k+1)\rceil+1$, in each membrane 2 that contains object B_i, we obtain $2^{\lceil \log(k+1)\rceil}$ copies of object $d_{i,\lceil \log(k+1)\rceil,n+\lceil \log(k+1)\rceil+1}$, so at least $k+1$ copies will be available.

From step $n+1$ to step $n+\lceil \log n\rceil+\lceil \log(k+1)\rceil$, object a_i will be duplicated by using rule $r_{2,i}$, that is, at step $n+\lceil \log n\rceil+\lceil \log(k+1)\rceil$, there are $2^{n+\lceil \log n\rceil+\lceil \log(k+1)\rceil}$ copies of object $a_{n+\lceil \log n\rceil+\lceil \log(k+1)\rceil+1}$ in membrane 1. In this process, counters b_i, c_i increase their subscripts by applying $r_{4,i}, r_{6,i}$.

At step $n+\lceil \log n\rceil+\lceil \log(k+1)\rceil+1$, object $a_{n+\lceil \log n\rceil+\lceil \log(k+1)\rceil+1}$ in membrane 1 is exchanged with object p from the environment by using rule r_8. All copies of objects b_i, c_i increase their subscripts at this step.

At step $n+\lceil \log n\rceil+\lceil \log(k+1)\rceil+2$, object $b_{n+\lceil \log n\rceil+\lceil \log(k+1)\rceil+2}$ has 2^n copies in membrane 1 and each such copy is exchanged with object e from the environment (rule r_9). By using rule $r_{6,i}$, 2^n copies of object $c_{n+\lceil \log n\rceil+\lceil \log(k+1)\rceil+3}$ will present in membrane 1. Simultaneously, by applying rules $r_{15,i}$ in all membranes 2, for each element B_i in the subset associated with the membrane we get $\min\{2^{\lceil \log(k+1)\rceil}, w(s_i)\}$ copies of object p.

The checking phase starts at step $n+\lceil \log n\rceil+\lceil \log(k+1)\rceil+3$. In this step, by using rule r_{16}, all pairs of objects p and q present in any membrane with label 2 are sent to membrane 1. In this way, if the weight of the subset associated with a membrane 2 is equal to k, then no object p or q remains in this membrane at the next step. Otherwise, at least one copy of object p or q will remain in the membrane. Simultaneously, in every membrane 2 object g is exchanged with object e in membrane 1 (rule r_{17}), that is, each membrane 2 will contain one copy of object e; object c_i increases its subscript, and 2^n copies of object $c_{n+\lceil \log n\rceil+\lceil \log(k+1)\rceil+4}$ appear in membrane 1.

When the checking phase finishes, the output phase starts at step $n+\lceil \log n\rceil+\lceil \log(k+1)\rceil+4$. There are two cases.

- There exists at least one membrane 2 with a subset whose weight is equal to k. In this case, at step $n+\lceil \log n\rceil+\lceil \log(k+1)\rceil+4$, if a membrane 2 contains at least one copy of object p or q, then object e will be sent to membrane 1 by using rule r_{18} or r_{19}; otherwise, object e will remain in that membrane. Simultaneously, object h in all membranes 2 is exchanged with object $c_{n+\lceil \log n\rceil+\lceil \log(k+1)\rceil+4}$ by using rule r_{20} (each membrane 2 will contain one copy of object $c_{n+\lceil \log n\rceil+\lceil \log(k+1)\rceil+4}$). At the next step, by applying rule r_{21}, objects e, m, $c_{n+\lceil \log n\rceil+\lceil \log(k+1)\rceil+4}$ are sent to membrane 1 in case the weight of the subset in that membrane 2 is equal to k. At step $n+\lceil \log n\rceil+\lceil \log(k+1)\rceil+6$, by using rule r_{10}, objects m, n, *yes* are sent to the environment and the computation halts. Thus, the answer of the system is *affirmative*.
- There is no membrane 2 with a subset whose weight is equal to k. In this case, at step $n+\lceil \log n\rceil+\lceil \log(k+1)\rceil+4$, object e is sent to membrane 1 from each membranes 2 by using rule r_{18} or r_{19}. Simultaneously, by applying rule r_{20}, each membrane 2 will contain one copy of object $c_{n+\lceil \log n\rceil+\lceil \log(k+1)\rceil+4}$. At the next two steps, only the counter object α_i evolves by applying rule $r_{1,i}$. At step $n+\lceil \log n\rceil+\lceil \log(k+1)\rceil+7$, by using rule r_{11}, objects n, $\alpha_{n+\lceil \log n\rceil+\lceil \log(k+1)\rceil+7}$, `no` are sent to the environment and the computation halts. Thus, the answer of the system is *negative*.

4.2. Some formal details

Family $\mathbf{\Pi} = \{\Pi(\langle n, k\rangle) \mid n, k \in \mathbb{N}\}$ is polynomially uniform by Turing machines, because the rules of a system $\Pi(\langle n, k\rangle)$ of the family are defined recursively from the values n and k. Besides, the necessary resources for defining each such system are of polynomial order.

- size of the alphabet: $((2n^2+3n+8)\cdot\lceil\log(k+1)\rceil+n\cdot\lceil\log(k+1)\rceil^2+12n+8\lceil\log n\rceil+46)/2 \in O(n^2\cdot\lceil\log k\rceil+n\cdot\lceil\log k\rceil^2)$;
- initial number of membranes: $2 \in O(1)$;
- initial number of objects: $n\cdot\lceil\log(k+1)\rceil+n+10 \in O(n\cdot\lceil\log k\rceil)$;
- number of rules: $((2n^2-3n+8)\cdot\lceil\log(k+1)\rceil+3n\cdot\lceil\log(k+1)\rceil^2+8\lceil\log n\rceil+20n+40)/2 \in O(n^2\cdot\lceil\log k\rceil+n\cdot\lceil\log k\rceil^2)$;
- maximum length of a rule: $3 \in O(1)$.

Hence, there exists a deterministic Turing machine that builds the system $\Pi(\langle n, k\rangle)$ in a polynomial time with respect to n and k.

According to the above mentioned computation process, it is clear that P system $\Pi(\langle n, k\rangle)$ with input multiset $cod(u)$ always halts and sends to the environment object `yes` (at step $n+\lceil\log n\rceil+\lceil\log(k+1)\rceil+6$) or object `no` (at step $n+\lceil\log n\rceil+\lceil\log(k+1)\rceil+7$). Therefore, there exists a polynomial bound for the number of steps of the computation.

Hence, the family $\mathbf{\Pi}$ of recognizer P systems with symport/antiport rules and membrane division solves the `Subset Sum` problem in polynomial time according to Definition 3. So, the following result is obtained.

Theorem 1. *SubsetSum* $\in \mathbf{PMC}_{\mathbf{CD}_e\mathbf{C}(3)}$.

Corrolary 1. $\mathbf{NP} \cup \text{co} - \text{NP} \subseteq \mathbf{PMC}_{\mathbf{CD}_e\mathbf{C}(3)}$.

Proof 1. It suffices to make the following observations: the `Subset Sum` problem is **NP**-complete, *SubsetSum* $\in \mathbf{PMC}_{\mathbf{CD}_e\mathbf{C}(3)}$ and the class $\mathbf{PMC}_{\mathbf{CD}_e\mathbf{C}(3)}$ is closed under polynomial time reduction, and is also closed under complement. □

5. Solving the `QSAT` problem by using $\mathbf{CD}_{ne}\mathbf{C}(3)$

In this Section, we provide a (*uniform*) polynomial time solution to the `QSAT` problem (quantified satisfiability problem), a well-known **PSPACE**-complete problem Papadimitriou (1994), by a family of P systems with division rules for non-elementary membranes and symport/antiport of length at most 3.

Given a Boolean formula $\varphi(x_1, \ldots, x_n)$ in conjunctive normal form, with Boolean variables $x_1, \ldots, x_n$, the sentence $\varphi^* = \exists x_1 \forall x_2 \ldots Q_n x_n \varphi(x_1, \ldots, x_n)$ (where Q_n is $\exists$ if n is odd, and Q_n is $\forall$ otherwise) is said to be the (existential) *fully quantified* formula associated with $\varphi(x_1, \ldots, x_n)$. We say that φ^* is satisfiable if there exists a truth assignment, σ, over $\{i : 1 \le i \le n \wedge i \text{ odd}\}$ such that each extension, σ^*, of σ over $\{1, \ldots, n\}$ verifies $\sigma^*(\varphi(x_1, \ldots, x_n)) = 1$.

The `QSAT` problem is the following one: *Given the (existential) fully quantified formula φ^* associated with a Boolean formula $\varphi(x_1, \ldots, x_n)$ in conjunctive normal form, determine whether or not φ^* is satisfiable.*

The solution proposed follows a brute force algorithm in the framework of recognizer P systems with symport/antiport rules and membrane division, and it consists of the following phases:

- *Generation phase*: using membrane division for non-elementary membranes, all truth assignments for the variables associated with the Boolean formula are produced.
- *Checking phase*: checking whether or not the formula $\varphi(x_1, \ldots, x_n)$ is satisfied.
- *Quantifier phase*: checking whether the whole formula φ^* with quantifiers is satisfied.

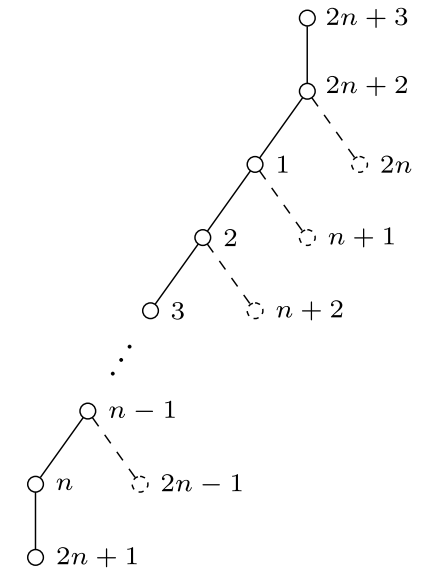

Fig. 1. The initial membrane structure of the P system.

- *Output phase*: the system sends to the environment the right answer according to the results of the previous phase.

We define a family $\mathbf{\Pi} = \{\Pi(t) \mid t \in \mathbb{N}\}$ of recognizer P systems from $\mathbf{CD}_{ne}\mathbf{C}(3)$ such that each system $\Pi(t)$ will process all instances of the `QSAT` problem with n variables and m clauses, where $t = \langle m, n\rangle$, provided that the appropriate input multiset is supplied to the system.

For each $(m, n) \in \mathbb{N} \times \mathbb{N}$, we consider the recognizer P system from $\mathbf{CD}_{ne}\mathbf{C}(3)$,

$$\Pi(\langle m, n\rangle) = (\Gamma, \mathcal{E}, \Sigma, \mu, \mathcal{M}_1, \ldots, \mathcal{M}_{2n+3}, \mathcal{R}_1, \ldots, \mathcal{R}_{2n+3}, i_{in}, i_{out}),$$

defined as follows:

- $\Sigma = \{x_{i,j}, \bar{x}_{i,j} \mid 1 \le i \le n, 1 \le j \le m\}$.
- $\mathcal{E} = \{d_i, g_i \mid 0 \le i \le n^2+3n+2m+3-k\} \cup \{g_{n^2+3n+2m+4-k}\}$ (where $k = \lfloor \frac{n}{2} \rfloor$ is the number of universal quantifiers in φ^*).
- $\Gamma = \Sigma \cup \mathcal{E} \cup \{a_i, b_i, t_i, f_i \mid 1 \le i \le n\} \cup \{E_i \mid 0 \le i \le m+1\} \cup \{t, yes, no\}$.
- $\mu = [[[[\ldots[[[]_{2n+1}]_n[]_{2n-1}]_{n-1}\cdots]_2[]_{n+1}]_1[]_{2n}]_{2n+2}]_{2n+3}$ (the root is the membrane with label $2n+3$ whose child is labeled by $2n+2$; this membrane has two children labeled by $2n$ and 1, respectively; each membrane with label i, $1 \le i \le n-1$, contains a non-elementary membrane with label $i+1$ and an elementary membrane with label $n+i$, see Fig. 1).
- $\mathcal{M}_1 = \{a_1\}, \mathcal{M}_{n+i} = \{b_i\}$, $1 \le i \le n$, $\mathcal{M}_{2n+2} = \mathcal{M}_i = \emptyset$, $2 \le i \le n$.
- $\mathcal{M}_{2n+1} = \{t, a_2, \ldots, a_n, E_0, E_1, \ldots, E_{m+1}\}$, $\mathcal{M}_{2n+3} = \{d_0, g_0, yes, no\}$.
- $i_{in} = 2n+1$ is the input membrane, and $i_{out} = 0$ is the output region.
- The sets of rules are defined below:

- Rules in $\mathcal{R}_i$ $(1 \le i \le n)$: $r_{1,i} \equiv [a_i]_i \to [t_i]_i[f_i]_i$.
- Rules in $\mathcal{R}_i$ $(2 \le i \le n)$: $r_{2,i,j} \equiv (t_j, in)$ and $r_{3,i,j} \equiv (f_j, in)$, $1 \le j \le i-1$.
- Rules in $\mathcal{R}_i$ $(3 \le i \le n)$: $r_{4,i,j} \equiv (a_j, out)$, $2 \le j \le i-1$.
- Rules in $\mathcal{R}_{2n+1}$:
  $r_{5,i} \equiv (a_{i+1}, out\,; t_i, in)$, $1 \le i \le n-1$.
  $r_{6,i} \equiv (a_{i+1}, out\,; f_i, in)$, $1 \le i \le n-1$.
  $r_7 \equiv (E_0\, E_1, out\,; t_n, in)$.
  $r_8 \equiv (E_0\, E_1, out\,; f_n, in)$.
  $r_{9,i,j} \equiv (t_i\, x_{i,j}, out\,; E_j, in)$, $1 \le i \le n$, $1 \le j \le m$.
  $r_{10,i,j} \equiv (f_i\, \bar{x}_{i,j}, out; E_j, in)$, $1 \le i \le n$, $1 \le j \le m$.
  $r_{11,i,j} \equiv (E_{j+1}, out\,; t_i\, x_{i,j}, in)$, $1 \le i \le n$, $1 \le j \le m$.

$r_{12,i,j} \equiv (E_{j+1}, out; f_i\, \bar{x}_{i,j}, in)$, $1 \le i \le n$, $1 \le j \le m$.
$r_{13} \equiv (t, out\,; E_{m+1}, in)$.

- Rules in $\mathcal{R}_n$: $r_{14} \equiv (t, out)$.
- If $Q_{i+1} = \forall$ $(1 \le i \le n-1)$:
  ($\star$) Rules in $\mathcal{R}_{n+i}$: $r_{15,i} \equiv (b_i, out\,; t, in)$.
  ($\star$) Rules in $\mathcal{R}_i$: $r_{16,i} \equiv (b_i\, t, out)$.
- If $Q_{i+1} = \exists$ $(1 \le i \le n-1)$:
  ($\star$) Rules in $\mathcal{R}_{n+i}$: $r_{17,i} \equiv (t, in)$ and $r_{18,i} \equiv (b_i\, t, out)$.
  ($\star$) Rules in $\mathcal{R}_i$: $r_{19,i} \equiv (b_i\, t, out)$.
- Rules in $\mathcal{R}_{2n}$: $r_{20} \equiv (t, in)$ and $r_{21} \equiv (b_n\, t, out)$.
- Rules in $\mathcal{R}_{2n+2}$: $r_{22} \equiv (b_n\, t, out)$.
- Rules in $\mathcal{R}_{2n+3}$:
  $r_{23} \equiv (d_{n^2+3n+2m+3-k}\, t\, yes, out)$.
  $r_{24,i} \equiv (d_i, out\,; d_{i+1}, in)$, $0 \le i \le n^2+3n+2m+2-k$.
  $r_{25,i} \equiv (g_i, out\,; g_{i+1}, in)$, $0 \le i \le n^2+3n+2m+3-k$.
  $r_{26} \equiv (d_{n^2+3n+2m+3-k}\, g_{n^2+3n+2m+4-k}\, no, out)$.

5.1. An overview of the computation

In what follows, we informally describe how P system $\Pi(s(\varphi^*))$ with input multiset $cod(\varphi^*)$ works. Let us recall that k denotes the number of universal quantifiers in φ^*, that is, $k = \lfloor \frac{n}{2} \rfloor$.

At the initial configuration, we have object a_1 in membrane 1, object b_i $(1 \le i \le n)$ in membrane $n+i$, objects $a_2, \ldots, a_n, E_1, \ldots, E_{m+1}$, t, $cod(\varphi^*)$ in membrane $2n+1$, objects g_0, `yes, no` in membrane $2n+3$.

Let us start with the generation phase. This phase has two parallel processes. On the one hand, the system assigns truth-assignments to all variables x_i $(1 \le i \le n)$. When generation phase completes, we produce 2^n copies of membrane $2n+1$, each of them contains a different truth assignments of the variable set $\{x_1, \ldots, x_n\}$ associated with the Boolean formula. On the other hand, the counter objects d_i, g_i in membrane $2n+3$ grow their subscripts by using rules $r_{26,i}$, $r_{27,i}$. We describe the process of generation phase as follows.

With the appearance of object a_i in each membrane i at the same step, the system starts to assign truth-assignment of variable x_i $(1 \le i \le n-1)$. Specifically, by using rule $r_{1,i}$ in all membranes i, each non-elementary membrane i is divided into two copies with the same label, which contain objects t_i (representing the truth value *true*) and f_i (representing the truth value *false*), respectively. Note that all the membranes and objects placed inside the membrane i are replicated in the new copies. At next steps, all object t_i (f_i, respectively) in membranes with label i will be sent to membranes n by using rules $r_{2,j,i}$ ($r_{3,j,i}$, respectively) $(i+1 \le j \le n)$ one by one. When object t_i (f_i, respectively) appears in each membrane n at the same step, rule $r_{5,i}$ ($r_{6,i}$, respectively) is enabled and applied, object a_{i+1} in membrane $2n+1$ is exchanged with object t_i (f_i, respectively) in membrane n. When object a_{i+1} appears in membranes n, by using rules $r_{4,j,i+1}$ $(i+2 \le j \le n)$ one by one, object a_{i+1} will be presented in each membrane $i+1$ at the same step. Similar to the case of variable x_i, the system continues to assign a truth-assignment to variable x_{i+1}. It is easy to see that the process of assigning truth-assignment of variable x_i $(1 \le i \le n-1)$ takes $2n-2i+1$ steps. With the appearance of object a_n in membranes n at the same step, the system starts to assign truth-assignment of variable x_n. By using rule $r_{1,n}$, each non-elementary membrane n is divided into two copies at the same step, which contain objects t_n (representing the truth value *true*) and f_n (representing the truth value *false*), respectively. At the next step, rule r_7 (r_8, respectively) is enabled and applied, objects E_0, E_1 in membranes $2n+1$ are exchanged with object t_n (f_n, respectively) in membranes n. The process of assigning a truth-assignment to variable x_n takes two steps. Hence, the generation phase takes n^2+1 steps. The membrane structure of the system at that moment when the generation phase completes is shown in Fig. 2.

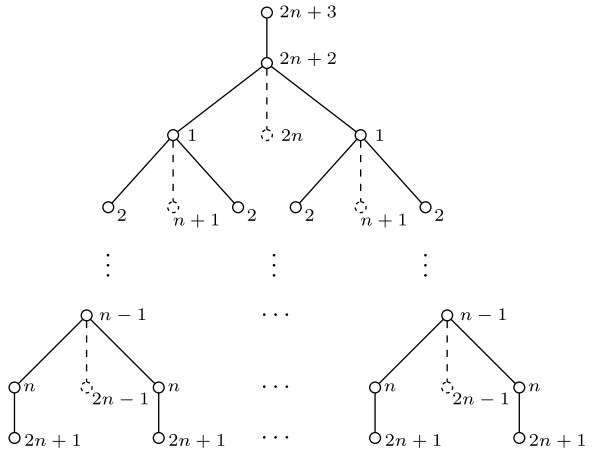

Fig. 2. The membrane structure of the system when the generation phase completes. Numbers at nodes indicate labels of membranes.

The checking phase takes $2m$ steps and consists of m loops (each loop takes 2 steps). In parallel with checking whether there is a truth assignment that makes the formula φ evaluate to be true, the counter objects d_i, g_i also grow their subscripts by one for each step in membrane $2n+3$.

At the first step of the j-th loop $(1 \le j \le m)$ of checking phase, objects t_i, $x_{i,j}$ ($f_i, \bar{x}_{i,j}$, respectively) in membrane $2n+1$ are exchanged with object E_j in membrane n by using rule $r_{9,i,j}$ ($r_{10,i,j}$, respectively), in case membrane $2n+3$ encodes a truth assignment making clauses $C_1, \ldots, C_j$ true. Note that for any membrane $2n+1$, which contains a truth assignment that does not make the clause C_j true, the computation in that membrane stops at the time when $r_{9,i,j}$ or $r_{10,i,j}$ is applied. At the second step of the j-th loop $(1 \le j \le m)$ of checking phase, with the appearance of objects t_i, $x_{i,j}$ ($f_i, \bar{x}_{i,j}$, respectively) in membrane n, by using rule $r_{11,i,j}$ ($r_{12,i,j}$, respectively), object E_{j+1} in membrane $2n+1$ is exchanged with the objects $t_i, x_{i,j}$ ($f_i, \bar{x}_{i,j}$, respectively) in membrane n.

After n^2+2m+1 steps, we have checked whether or not formula φ is satisfied by the corresponding truth assignment. For each clause C_j which is satisfied, the subscript j of E is increased by one; hence object E_{m+1} will appear in membrane n if and only if its lower neighbor membrane $2n+1$ encodes the truth assignment that satisfies all clauses. At step n^2+2m+2, if membrane n contains E_{m+1} then by using rule r_{13}, object t in membrane $2n+1$ is exchanged with object E_{m+1} in membrane n. At the next step, if object t appears in membrane n, by applying rule r_{14}, then it will be sent to membrane $n-1$.

The quantifier phase starts at the (n^2+2m+4)-th step. In membrane $2n+3$ the counter objects d_i, g_i grow their subscripts by one for each step. A membrane with label i corresponds to the quantifier Q_j, where $1 \le i \le n-1$, $j = i+1$, and a membrane with label $2n+2$ corresponds to the quantifier Q_1. If $Q_j = \forall$, object t is passed to the upper level only if it comes from both lower level membranes, that is, the respective clauses are satisfied for both truth values of x_j. If $Q_j = \exists$, then a single object t coming from lower level is enough. In what follows, we describe how the system simulates quantifiers $\forall$ and $\exists$.

For quantifier $Q_{i+1} = \forall$ $(1 \le i \le n-1$ and i is odd), one copy of object t can be sent to the upper level membrane if and only if there are two copies of object t in a membrane i, where each lower level membrane provides one copy of object t. Specifically, by using rule $r_{15,i}$, object b_i in membranes $n+i$ is exchanged with object t in their upper level membranes i. Note that there is one copy of object b_i in a membrane $n+i$, thus, only one copy of object t is sent to membrane $n+i$ from a membrane i. At the next step, objects b_i, t in membranes i are sent to their upper level membranes by applying rule $r_{16,i}$.

In case $Q_{i+1} = \exists$ ($1 \leq i \leq n-1$ and i is even), by using the rule $r_{17,i}$ in all membranes $n+i$, all copies of object t will be sent to their lower level membranes $n+i$. At the next step, objects b_i, t are sent to membrane i by using rule $r_{18,i}$. If there are two copies of object t in membrane $n+i$, then only one copy of object t is sent to membrane i because there is only one copy of object b_i in a membrane $n+i$. At the next step, objects b_i, t in membranes i are sent to their upper level membranes by using rule $r_{19,i}$. As $Q_1 = \exists$, by using the rules r_{20}, r_{21}, r_{22} one by one, one copy of object t is sent to the membrane $2n+3$.

Hence, the simulation of all universal quantifiers takes $2k$ steps, and the simulation of all existential quantifiers takes $3(n-k)$ steps (the number of such quantifiers is $n-k$). Thus, the quantifier phase takes $3n-k$ steps.

The output phase starts at the $(n^2+3n+2m+4-k)$-th step, and it takes 1 step (affirmative answer) or 2 steps (negative answer). Let us recall that membrane $2n+3$ at configuration $\mathcal{C}_{n^2+3n+2m+3-k}$ contains objects $d_{n^2+3n+2m+3-k}$, $g_{n^2+3n+2m+3-k}$, `yes` and `no`. There are two cases.

- *Affirmative answer*: if object t appears in membrane $2n+3$ at configuration $\mathcal{C}_{n^2+3n+2m+3-k}$, then rules r_{23} and $r_{25,n^2+3n+2m+3-k}$ are applicable and objects $d_{n^2+3n+2m+3-k}$, t and `yes` are sent to the environment. Membrane $2n+3$ at configuration $\mathcal{C}_{n^2+3n+2m+4-k}$ only contains objects $g_{n^2+3n+2m+4-k}$ and `no`. Therefore, the computation halts.
- *Negative answer*: if object t does not appear in membrane $2n+3$ at configuration $\mathcal{C}_{n^2+3n+2m+3-k}$, then only rule $r_{25,n^2+3n+2m+3-k}$ is applicable. Thus, that membrane at configuration $\mathcal{C}_{n^2+3n+2m+4-k}$ contains objects `yes`, `no`, $g_{n^2+3n+2m+4-k}$, $d_{n^2+3n+2m+3-k}$. At the next step, by using rule r_{26}, objects $g_{n^2+3n+2m+4-k}$, $d_{n^2+3n+2m+3-k}$ and `no` are sent to the environment and membrane $2n+3$ at configuration $\mathcal{C}_{n^2+3n+2m+5-k}$ only contains object `yes`, so it is a halting configuration.

5.2. Some formal details

In order to show that the family $\mathbf{\Pi} = \{\Pi(\langle m, n\rangle) \mid m, n \in \mathbb{N}\}$ defined above is polynomially uniform by Turing machines we need to prove that $\Pi(\langle m, n\rangle)$ is built in polynomial time with respect to the size parameter m, n of instances of the `QSAT` problem (recalling that $k \leq n$).

It is easy to show that the rules of a system $\Pi(\langle m, n\rangle)$ of the family are defined recursively from the values m, n, and the necessary resources to construct $\Pi(\langle m, n\rangle)$ are as follows:

- size of the alphabet: $2n^2+2nm+10n+5m-2k+12 \in O(n^2+nm)$;
- initial number of membranes: $2n+3 \in O(n)$;
- initial number of objects: $2n+m+7 \in O(n+m)$;
- number of rules: $(7n^2+8nm+23n+8m-4k+20)/2 \in O(n^2+nm)$;
- maximum length of a rule: $3 \in O(1)$.

Thus, there exists a deterministic Turing machine that builds the system $\Pi(\langle m, n\rangle)$ in a polynomial time with respect to m and n.

By the above checking of the computation process, we can prove that the P system $\Pi(\langle m, n\rangle)$ with input multiset $cod(\varphi^*)$ always halts and sends to the environment object `yes` or `no` in the last step, that is, at step $n^2+3n+2m+4-k$, object `yes` is sent to the environment and the system halts; object `no` is sent to the environment at step $n^2+3n+2m+5-k$ and the system halts. Therefore, there exists a polynomial bound for the number of steps of the computation.

Hence, the family $\mathbf{\Pi}$ of recognizer P systems with symport/antiport rules and membrane division solves the `QSAT` problem in polynomial time according to Definition 3. So, the following result is obtained.

Theorem 2. *QSAT* $\in \mathbf{PMC}_{\mathbf{CD}_{ne}\mathbf{C}(3)}$.

Since the complexity class $\mathbf{PMC}_{\mathbf{CD}_{ne}\mathbf{C}(3)}$ is closed under polynomial time reductions, we have the following result.

Corrolary 2. $\mathbf{PSPACE} \subseteq \mathbf{PMC}_{\mathbf{CD}_{ne}\mathbf{C}(3)}$.

6. Conclusions and discussion

The computational efficiency of different computing devices in membrane computing has been established. With regard to cell-like computing devices, efficient solutions to hard problems have been given in the framework of P systems with active membranes that use evolution, send-in, send-out, dissolution and division rules. Specifically, polynomial time solutions to the SAT problem Pérez-Jiménez et al. (2003) (in a uniform way by using electrical charges) and the `QSAT` problem Alhazov and Pérez-Jiménez (2007) (in a semi-uniform way, without electrical charges and allowing division for non-elementary membranes) have been proposed. With regard to tissue-like computing devices, an efficient (uniform) solution to the HAM-CYCLE problem by a family of tissue P systems with cell division and symport/antiport rules of length at most 2, has been given Porreca et al. (2012).

In this work, the computational efficiency of cell-like polarizationless P systems with symport/antiport rules and membrane division has been investigated. Specifically, a (uniform) linear time solution to the **NP**-complete problem `Subset Sum` by using division rules for elementary membranes and communication rules of length at most 3, has been given. We further proved that such P system can efficiently solve the **PSPACE**-complete problem `QSAT` problem, in a uniform way, when division rules for non-elementary membranes are allowed. It is worth noting that the solution to `QSAT` given in the paper can be adapted to a uniform solution bearing in mind that for (existential) *fully quantified* formula associated with $\varphi(x_1, \ldots, x_n)$, the number of universal quantifiers is $\lfloor \frac{n}{2} \rfloor$.

We propose some open problems related to the role of communication rules in cell-like P systems with membrane division from a computational complexity point of view.

(a) In the solution to `QSAT` proposed in this paper, division rules for non-elementary membranes have been considered. Is it possible to provide an efficient solution to `QSAT` by using only division rules for elementary membranes?
(b) The P systems constructed in Section 4 and in Section 5 have both symport rules and antiport rules. What about the computational efficiency of P systems with membrane division that use only either symport or antiport rules?
(c) It is known that $\mathbf{NP} \cup \mathbf{co}-\mathbf{NP} \subseteq \mathbf{PMC}_{\mathbf{TDC}(2)}$ Porreca et al. (2012). What about the computational efficiency of $\mathbf{CD}_e\mathbf{C}(2)$?
(d) The environment is not relevant for tissue P systems with symport/antiport rules and cell division from a complexity point of view Pérez-Jiménez et al. (2013). What about the efficiency of recognizer P systems from $\mathbf{CDC}(k)$ when the alphabet of the environment is an empty set?

P systems capture the inherent degree of freedom present in biological systems through the non-determinism. With the inclusion of this ingredient, P systems are able to solve hard problems in an "efficient" way at the theoretical level by trading time for space. New boundaries between tractability and **NP**-hardness, in terms of syntactical ingredients of P systems, provide new tools to tackle the **P** versus **NP** problem.

Different simulators for P systems running on conventional computers have been developed during the last decade. The addition of parallel computing techniques, like those based on GPU, is accelerating these simulators. With the aid of P systems simulators, researchers aim to solve larger instances of **NP** hard problems than

the best ones provided so far, running on electronic computers. However, we would like to stress that such simulators are not real implementation of P systems. Obviously, an efficient real implementation of P systems would provide a "constructive" proof of the result **P=NP**, that is, similarly to what would happen with a practical implementation of non-deterministic Turing machines.

Acknowledgements

The work of B. Song and L. Pan was supported by National Natural Science Foundation of China (61033003, 91130034, and 61320106005), Ph.D. Programs Foundation of Ministry of Education of China (2012014213008), and Natural Science Foundation of Hubei Province (2011CDA027). The work of M.J. Pérez-Jiménez was supported by "Ministerio de Economía y Competitividad" of Spain (TIN2012-37434), cofunded by FEDER funds.

References

Alhazov, A., Freund, R., 2005. P systems with one membrane and symport/antiport rules of five symbols are computationally complete. In: Proceedings of Third Brainstorming Week On Membrane Computing, Sevilla, Spain, pp. 19–28.

Alhazov, A., Rogozhin, Yu., 2006. Towards a characterization of P systems with minimal symport/antiport and two membranes. In: Vol. 4361 of Lecture Notes in Computer Science. Springer, Berlin/Heidelberg, pp. 135–153.

Alhazov, A., Pérez-Jiménez, M.J., 2007. Uniform solution of QSAT using polarizationless active membranes. In: Vol. 4664 of Lecture Notes in Computer Science. Springer, Berlin/Heidelberg, pp. 122–133.

Bernardini, F., Gheorghe, M., 2003. On the power of minimal symport/antiport. In: Proceedings of the 3rd Workshop on Membrane Computing, Tarragona, pp. 72–83.

Ciobanu, G., Pan, L., Păun, Gh., Pérez-Jiménez, M.J., 2007. P systems with minimal parallelism. Theor. Comput. Sci. 378, 117–130.

Díaz-Pernil, D., Gutiérrez-Naranjo, M.A., Pérez-Jiménez, M.J., Riscos-Núñez, A., 2007. A linear solution for Subset Sum problem with tissue P systems with cell division. In: Vol. 4527 of Lecture Notes in Computer Science. Springer, Berlin/Heidelberg, pp. 170–179.

Frisco, P., 2009. Computing with Cells: Advances in Membrane Computing. Oxford University Press, Oxford.

Gutiérrez-Naranjo, M.A., Pérez-Jiménez, M.J., Romero-Campero, F.J., 2005. A linear solution of `subset sum` problem by using membrane creation. In: Vol. 3561 of Lecture Notes in Computer Science. Springer, Berlin/Heidelberg, pp. 258–267.

Ionescu, M., Păun, Gh., Yokomori, T., 2006. Spiking neural P systems. Fund. Inform. 71 (2–3), 279–308.

Martín-Vide, C., Pazos, J., Păun, Gh., Rodriguez-Paton, A., 2003. Tissue P systems. Theor. Comput. Sci. 296 (2), 295–326.

Papadimitriou, C.H., 1994. Computational Complexity. Addison-Wesley, Reading, Mass.

Păun, A., 2000. On P systems with active membranes. In: Proceedings of the Second International Conference on Unconventional Models of Computation, UMC'2K, Brussels, Belgium, pp. 187–201.

Păun, A., Păun, Gh., 2002. The power of communication: P systems with symport/antiport. New Gener. Comput. 20 (3), 295–305.

Păun, Gh., 2000. Computing with membranes. J. Comput. Syst. Sci. 61 (1), 108–143.

Păun, Gh., 2001. P systems with active membranes: attacking NP-complete problems. J. Auto. Lang. Comb. 6, 75–90.

Păun, Gh., Pazos, J., Pérez-Jiménez, M.J., Rodríguez-Patón, A., 2005. Symport/antiport P systems with three objects are universal. Fund. Inform. 64, 1–4.

Păun, Gh., Rozenberg, G., Salomaa, A., 2010. The Oxford Handbook of Membrane Computing. Oxford University Press, Oxford.

Pérez-Jiménez, M.J., Romero-Jiménez, A., Sancho-Caparrini, F., 2003. Complexity classes in models of cellular computing with membranes. Nat. Comput. 2 (3), 265–285.

Pérez-Jiménez, M.J., Riscos-Núñez, A., 2004. A linear time solution to the Knapsack problem using P systems with active membranes. In: Vol. 2933 of Lecture Notes in Computer Science. Springer, Berlin/Heidelberg, pp. 250–268.

Pérez-Jiménez, M.J., Riscos-Núñez, A., 2005. Solving the Subset-Sum problem by active membranes. New Gener. Comput 23, 367–384.

Pérez-Jiménez, M.J., 2005. An approach to computational complexity in membrane computing. In: Vol. 3365 of Lecture Notes in Computer Science. Springer, Berlin/Heidelberg, pp. 85–109.

Pérez-Jiménez, M.J., Romero-Jiménez, A., Sancho-Caparrini, F., 2006. A polynomial complexity class in P systems using membrane division. J. Auto. Lang. Comb. 11 (4), 423–434.

Pérez-Jiménez, M.J., Riscos-Núñez, A., Rius-Font, M., Romero-Campero, F.J., 2013. A polynomial alternative to unbounded environment for tissue P systems with cell division. Int. J. Comput. Math. 90 (4), 760–775.

Porreca, A.E., Murphy, N., Pérez-Jiménez, M.J., 2012. An optimal frontier of the efficiency of tissue P systems with cell division. In: Proceedings of the Tenth Brainstorming Week on Membrane Computing, Volume II, Sevilla, Spain, pp. 141–166.

Rozenberg, G., Salomaa, A., 1997. Handbook of Formal Languages, vol. 3. Springer-Verlag, Berlin.