An efficient time-free solution to SAT problem by P systems with proteins on membranes

Bosheng Song [a], 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 Intelligence, University of Sevilla, Avda. Reina Mercedes s/n, 41012 Sevilla, Spain

Keywords:

P system
Membrane protein
Time-free uniform solution
SAT

A B S T R A C T

P systems with proteins on membranes are a class of bio-inspired computing models, where the execution of each rule completes in exactly one time unit. However, in living cells, the execution time of biochemical reactions is difficult to know precisely because of various uncontrollable factors. In this work, we present a time-free uniform solution to SAT problem by P systems with proteins on membranes in the sense that the correctness of the solution is irrelevant to the times associated with the involved rules, and the P systems are constructed from the sizes of instances.

1. Introduction

Membrane computing is an active branch of natural computing, which is inspired by the structure and the functioning of living cells, abstracting computational ideas (e.g., computational models, data structure, data operation) from the way in which chemicals interact and cross cellular membranes. This computational model was initiated by Gh. Păun at the end of 1998 [15] and it has received great attention from computer scientists, biologists, formal linguists and complexity theoreticians. The computational models considered in the framework of membrane computing are called *P systems*, which are parallel and distributed computational models.

In general, three main types of P systems have been considered until now: cell-like P systems, which have a hierarchical arrangement of membranes delimiting compartments where *multisets* of chemicals (called *objects*) evolve according to given *evolution rules* [1,15,24]; tissue-like P systems, which have arbitrary graphs as underlying structures, with cells placed in the nodes while edges correspond to communication channels [3,8,9]; neural-like P systems, which are motivated by the neurophysiological behavior of neurons sending electrical impulses (spikes) along axons to other neurons [6,10,21]. A recent coverage of membrane computing can be found in [17], and for the most up-to-date information in this area, please refer to the P systems webpage: http://ppage.psystems.eu.

The present work deals with cell-like P systems with proteins on membranes (in what follows, when we say P systems with proteins on membranes, it means cell-like P systems with proteins on membranes), which can be viewed as a model combining membrane systems and brane calculi as introduced in [12]. P systems with proteins on membranes can have objects both **in** compartments/regions, as usual in P systems, and also **on** membranes (called *proteins*). In such P systems,

* Corresponding author.
E-mail address: lqpan@mail.hust.edu.cn (L. Pan).

the proteins (which are always placed on the membranes) are used mainly to control the evolution of other objects. The evolution rules used in P systems with proteins on membranes are inspired by the symport/antiport rules [11], but the objects can evolve under the control of proteins placed on membranes (the protein can also change). Moreover, P systems with proteins on membranes deal with only one protein, one object inside the region and/or one object outside of it.

P systems with proteins on membranes were proved to be universal by combining several various types of rules [7,12,14]. If we allow membrane division rules in this kind of P systems, then it can solve the computationally hard problems efficiently. In [13], the SAT problem was solved by a uniform family of P systems with proteins on membranes and membrane division in a polynomial time. Recently, P systems with proteins on membranes were used to characterize **PSPACE**-complete problem, that is, the polynomial complexity class (semi-uniform version) associated with recognizer P systems with proteins on membranes coincides with the class of **PSPACE**-complete problems [27]. In the semantics of P systems with proteins on membranes, a global clock is assumed, marking time units for the whole system, all the rules are applied synchronously and the execution of each rule takes exactly one time unit (one step). Actually, in cell biology, different biochemical reactions may take different times to be completed because they are influenced by various uncontrollable factors, such as biological signals, catalyst, and medium temperature. Thus, the assumption that the execution of each rule takes exactly one time unit is not in conformity with the biological reality.

With this biological motivation, time-free P systems were formulated in [2], where such P systems generate (or accept) the same family of vectors of natural numbers, independent from the value assigned to the execution time of each rule, and by simulating non-synchronized P systems, time-free P systems were proved to be universal. Time-free solutions to computational hard problems as open problems were formulated in [5]. In [19], a family of P systems with active membranes was designed for a time-free solution to SAT problem in the sense that the correctness of the solution is irrelevant to the execution time used by the involved rules. Since then, several variants of P systems were also considered to time-freely solve **NP**-complete problems [20,23,25,26].

In this work, we focus on timed P systems with proteins on membranes, and a time-free uniform solution to the SAT problem by using such systems is present, where the correctness of the solution is irrelevant to the execution time used by the involved rules, and the P systems are constructed from the sizes of instances.

2. P systems with proteins on membranes

2.1. Preliminaries

In this subsection, we only introduce a few basic notions and notations from formal languages theory that will be used in what follows, the reader can refer to [18] for more details.

An *alphabet* Σ is a non-empty set and its elements are called *symbols*. A *string* u over Σ is a finite sequence of symbols from Σ. The number of occurrences in u of symbols from Σ is called the *length* of the string u, denoted by $|u|$. By $|u|_a$ we denote the number of occurrences of symbol a in u. As usual, the empty string (with length 0) will be denoted by λ. The set of all strings over an alphabet Σ is denoted by Σ^* and by $\Sigma^+ = \Sigma^* \setminus \{\lambda\}$ we denote the set of non-empty strings. A *language* over Σ is a set of strings over Σ.

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 $supp(m) = \{x \in \Sigma \mid f(x) > 0\}$. 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. We also represent a multiset m over V as a string $w \in V^*$ such that $|w|_{a_i} = m(a_i)$, $1 \le i \le n$. All permutations of this string identify the same multiset m precisely. In what follows, we will not distinguish between the representation of multiset in mapping form or string form. 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 $f_1(x) \ge f_2(x)$, and $g(x) = 0$ otherwise.

2.2. P systems with proteins on membranes and membrane division

In this subsection, we first introduce the notion of P systems with proteins on membranes and membrane division from [12], then the definition of recognizer P systems with proteins on membranes is given.

In this work, we consider P systems with two types of symbols: *proteins* and *objects*. Proteins are placed on the membranes, which never leave their places during the evolution process (but they can evolve) and are used mainly to control the evolution of other objects, and the usual objects are placed in the regions delimited by membranes, which can leave the regions for the neighboring regions during the evolution. If a protein p is placed on a membrane (with label) i and an object a is placed in the region delimited by that membrane, it is denoted by $[\ p \mid a\ ]_i$. The regions of a membrane structure and the membranes themselves can contain multisets of objects and of proteins, respectively.

There are several types of rules for handling the proteins and the objects, in this work we consider the types of rules introduced in [12].

P systems can be used for solving decision problems by using the trade-off space for time. A P system solving such a problem demands that an exponential workspace can be generated in a polynomial (or linear) time, and membrane division rules can generate that workspace. This kind of rules can be introduced in membranes controlled by proteins on them in such manner that proteins will trigger these rules.

Definition 1. A P system with proteins on membranes and membrane division of degree $m \geq 1$ is a tuple $\Pi = (O, P, \mu, w_1/z_1, \dots, w_m/z_m, \mathcal{E}, \mathcal{R}_1, \dots, \mathcal{R}_m, i_{out})$, where:

- O and P are finite non-empty alphabets such that $O \cap P = \emptyset$;
- μ is a rooted tree with m nodes labeled by $1, \dots, m$;
- w_i, $1 \leq i \leq m$, are multisets over O;
- z_i, $1 \leq i \leq m$, are multisets over P;
- $\mathcal{E} \subseteq O$ is a finite alphabet;
- $\mathcal{R}_i$, $1 \leq i \leq m$, are finite sets of rules of the following types:
  - *Evolution rules* of the types:
    (*a*) $[\ p \mid a\ ]_i \to [\ p' \mid b\ ]_i$,
    (*b*) $a[\ p \mid\ ]_i \to b[\ p' \mid\ ]_i$,
    (*c*) $[\ p \mid a\ ]_i \to b[\ p' \mid\ ]_i$,
    (*d*) $a[\ p \mid\ ]_i \to [\ p' \mid b\ ]_i$,
    (*e*) $a[\ p \mid b\ ]_i \to c[\ p' \mid d\ ]_i$,
    where $p, p' \in P$, $a, b, c, d \in O$ and $1 \leq i \leq m$.
  - *Division rules* of the type $[\ p \mid\ ]_i \to [\ p' \mid\ ]_i[\ p'' \mid\ ]_i$, where $p, p', p'' \in P$, $1 \leq i \leq m$, $i \neq i_{out}$, and i cannot be the root of the tree μ;
- $i_{out} \in \{0, 1, \dots, m\}$.

A P system with proteins on membranes and membrane division of degree $m \geq 1$ can be viewed as a set of m membranes, labeled by $1, \dots, m$, each of them delimiting regions (the space between a membrane and the immediately inner membranes) such that: (a) $w_1, \dots, w_m$ represent the finite multisets of objects (symbols of the alphabet O) initially placed in the m membranes of the system; (b) $z_1, \dots, z_m$ represent the finite multisets of proteins (symbols of the alphabet P) initially placed on the m membranes of the system; (c) $\mathcal{E}$ is the set of objects initially located in the environment of the system, all of them available in an arbitrary number of copies; and (d) i_{out} represents a distinguished zone which will encode the output of the system. We use the term zone i ($0 \leq i \leq m$) to refer to membrane i in the case $1 \leq i \leq m$ and to refer to the environment in the case $i = 0$. The length of an evolution rule is the total number of proteins and objects involved in that rule.

A *configuration* of a P system with proteins on membranes and membrane division is described by the current membrane structure, together with all multisets of objects located in its regions, and multisets of proteins present on membranes. The initial configuration is given by $(w_1/z_1, \dots, w_m/z_m, \mu)$, that is, by the multisets of objects $w_1, \dots, w_m$ placed in the membranes, the multisets of proteins $z_1, \dots, z_m$ placed on the membranes and the initial membrane structure.

An evolution rule $[\ p \mid a\ ]_i \to [\ p' \mid b\ ]_i$ is applicable to a configuration if membrane i belongs to that configuration, the object a is placed in membrane i and the protein p is placed on that membrane. When applying such a rule, in reaction with an object a and a protein p, in the membrane i, the object a is replaced by object b and the protein p is replaced by protein p'. An evolution rule $a[\ p \mid\ ]_i \to b[\ p' \mid\ ]_i$ applicable to a configuration if membrane i belongs to that configuration, the object a is placed in the father of membrane i (the environment in the case of membrane i is the root) and the protein p is placed on that membrane. When applying such a rule, in reaction with an object a and a protein p, in the father of the membrane i, the object a is replaced by object b and on the membrane i, the protein p is replaced by protein p'. An evolution rule $[\ p \mid a\ ]_i \to b[\ p' \mid\ ]_i$ applicable to a configuration if membrane i belongs to that configuration, the object a is placed in the membrane i and the protein p is placed on that membrane. When applying such a rule, in reaction with an object a and a protein p, in the father of the membrane i, the object a is replaced by object b and on the membrane i, the protein p is replaced by protein p'. In a similar way is defined the semantics of the remaining rules.

A division rule $[\ p \mid\ ]_i \to [\ p' \mid\ ]_i[\ p'' \mid\ ]_i$ is applicable to a configuration if membrane i belongs to that configuration, if the protein p is placed on membrane i. When applying such a rule, in reaction with the protein p, membrane i is divided into two membranes with the same label i; in the first copy, the protein p is replaced by protein p', in the second one, the protein p is replaced by protein p''; all the other objects in membrane i and all the other proteins on membrane i are replicated, and the copies of them are placed in and on the two new membranes, respectively. If $i_{out} \in \{1, \dots, m\}$, then membrane i_{out} cannot be divided. The root of the membrane structure cannot be divided.

The rules in a system are used in a non-deterministic maximally parallel manner as customary in membrane computing. At each step, all membranes which can evolve must evolve in a maximally parallel way (at each step we apply a multiset of rules which is maximal, no further rule can be added being applicable). This way of applying rules has only one restriction: when a membrane is divided, the division rule is the only one which is applied for that membrane at that step; thus, the proteins and objects in that membrane do not evolve by means of evolution rules. The new membranes resulting from division could participate in the interaction with other membranes or the environment by means of evolution rules at the next step, providing that they are not divided once again. The label of a membrane identifies the rules which can be applied to it precisely.

The system passes from one configuration to another one by a maximally parallel application of rules as described above. A *computation* is a (finite or infinite) sequence of configurations such that: the first term of the sequence is the initial configuration of the system; each non-initial configuration of the sequence is obtained from the previous configuration by applying rules of the system in a maximally parallel manner with the restrictions previously mentioned; if the sequence is finite (called *halting computation*), then the last term of the sequence is a halting configuration.

All the computations start from an initial configuration and proceed as stated above; only a halting computation gives a result, which is encoded by the objects present in the output zone i_{out} in the halting configuration.

Definition 2. A recognizer P system with proteins on membranes of degree $m \geq 1$ is a tuple $\Pi = (O, P, \Sigma, \mu, w_1/z_1, \ldots, w_m/z_m, \mathcal{E}, \mathcal{R}_1, \ldots, \mathcal{R}_m, i_{out}, i_{in})$ such that:

- The tuple $(O, P, \mu, w_1/z_1, \ldots, w_m/z_m, \mathcal{E}, \mathcal{R}_1, \ldots, \mathcal{R}_m, i_{out})$, is a P system with proteins on membranes of degree m such that the output zone is the environment ($i_{out} = 0$) and in this case, i_{out} is usually omitted from the tuple;
- Σ is an (input) alphabet strictly contained in O such that $\mathcal{E} \cap \Sigma = \emptyset$;
- The initial multisets $w_1, \ldots, w_m$ are over $O \setminus \Sigma$;
- $i_{in} \in \{1, \ldots, m\}$ is the label of a distinguished (input) membrane;
- The working alphabet contains two distinguished elements yes and no;
- All the computations halt;
- If $\mathcal{C}$ is a computation of the system, then either object yes or object no (but not both) must appear in the environment when the system halts.

For recognizer P systems with proteins on membranes, we say that a computation is an *accepting computation* (resp., *rejecting computation*) if the object yes (resp., no) appears in the environment associated with the corresponding halting configuration.

For each multiset w over the input alphabet Σ, the *computation of the system* Π *with input* w starts from the configuration of the form

$$(w_1/z_1, \ldots, (w_{i_{in}} + w)/z_{i_{in}}, \ldots, w_m/z_m, \mu),$$

that is, the input multiset w has been added to the contents of the input membrane i_{in}. Therefore, we have an initial configuration associated with each input multiset w (over the input alphabet Σ) in this kind of systems.

We denote by *CPE*(k) the class of recognizer P systems with proteins on membranes with the length of evolution rules at most k.

2.3. Recognizer timed P systems with proteins on membranes

In this subsection, we first give the definition of timed P systems with proteins on membranes, and then introduce the notion of recognizer timed P systems with proteins on membranes.

Definition 3. A timed P system with proteins on membranes of degree $m \geq 1$ is a pair (Π, e), where Π is a P system with proteins on membranes of degree m, and e is a computable time-mapping of Π, that is, e is a mapping from the finite set of rules $\mathcal{R}_1 \cup \cdots \cup \mathcal{R}_m$ into the set of natural numbers $\mathbb{N}$.

For each rule r, the natural number $e(r)$ represents the execution time of rule r. We denote by $\Pi(e)$ the timed P system with proteins on membranes (Π, e).

A timed P system with proteins on membranes $\Pi(e)$ works in the following way: an external clock is assumed, which marks time-units of equal length, starting from instant 0. According to this clock, the step t of a computation is defined by the period of time between instant $t-1$ and instant t. If a membrane i contains a rule r (evolution rule or division rule) selected to be executed, then the execution of such rule takes $e(r)$ time units to complete. Therefore, if the execution of a rule r is started at instant j, then the execution is completed at instant $j + e(r)$ and the resulting objects, protein and membranes generated by the division rule become available only at the beginning of step $j + e(r) + 1$. If an evolution rule is started, then the occurrences of objects and protein subject to this rule cannot be subject to other rules until the implementation of the rule completes; if the division rule is started, then the occurrences of protein and membrane subject to this rule cannot be subject to other rules until the implementation of the rule completes.

Definition 4. A recognizer timed P system with proteins on membranes of degree $m \geq 1$ is a tuple (Π, e), where Π is a recognizer P system with proteins on membranes of degree m and e is a computable time-mapping of Π.

2.4. Time-free uniform solutions to decision problems by recognizer P systems with proteins on membranes

In a timed P systems with proteins on membranes, we use the *rule starting step* (RS-step, for short) to define the computation step, which was proposed in [19].

Definition 5. In timed P systems with proteins on membranes, a computation step is called an RS-step if at this step at least one rule starts its execution, that is, steps in which some objects and proteins "start" to evolve or membranes "start" to divide.

Let us recall that a *decision problem*, X, is a pair (I_X, θ_X) such that I_X is a language over a finite alphabet (whose elements are called *instances*) and θ_X is a total Boolean function (that is, predicate) over I_X.

Definition 6. We say that a decision problem $X = (I_X, \theta_X)$ is solvable in polynomial RS-steps by a family $\mathbf{\Pi} = \{\Pi_n \mid n \in \mathbb{N}\}$ of recognizer P systems with proteins on membranes in a time-free manner, if the following holds:

- the family $\mathbf{\Pi}$ is polynomially uniform by Turing machines, that is, there exists a deterministic Turing machine working in polynomial time which constructs the system Π_n from $n \in \mathbb{N}$.
- there exists a pair (cod, s) of polynomial-time computable functions over I_X such that:
  - for each instance $u \in I_X$, $s(u)$ is a natural number and $cod(u)$ is an input multiset of the system $\Pi_{s(u)}$.
  - the family $\mathbf{\Pi}$ is time-free sound with respect to (X, cod, s); that is, for any time-mapping e, the following property holds: for each instance of the problem $u \in I_X$ such that there exists an accepting computation of $\Pi_{s(u)}(e)$ with input $cod(u)$, we have $\Theta_X(u) = 1$.
  - the family $\mathbf{\Pi}$ is time-free complete with respect to (X, cod, s); that is, for any time-mapping e, the following property holds: for each instance of the problem $u \in I_X$ such that $\Theta_X(u) = 1$, every computation of $\Pi_{s(u)}(e)$ with input $cod(u)$ is an accepting computation.
  - the family $\mathbf{\Pi}$ is time-free polynomially bounded with respect to (X, cod, s); that is, there exists a polynomial function $p(n)$ such that for any time-mapping e and for each $u \in I_X$, all the computations in $\Pi_{s(u)}(e)$ with input $cod(u)$ halt in, at most, $p(|u|)$ RS-steps.

We also say that the family $\mathbf{\Pi}$ provides an efficient *time-free solution* to the decision problem X. We denote by $\mathbf{PMC}^{tf}_{CPE(k)}$ the family of problems that can be solved by recognizer P systems with proteins on membranes and the length of evolution rules at most k in a time-free manner.

3. A time-free uniform solution to SAT problem by using recognizer P systems with proteins on membranes

The SAT problem is the best known **NP**-complete problem [4], which is defined as follows: *Given a Boolean formula in conjunctive normal form (CNF), determine whether or not it is satisfiable, that is, whether there exists an assignment to its variables on which it evaluates to be true.*

The following theorem provides a polynomial RS-steps solution to the SAT problem by a family of P systems with proteins on membranes in a time-free uniform manner.

Theorem 3.1. *The* SAT *problem can be solved in polynomial RS-steps by a family of P systems with proteins on membranes in a time-free uniform manner, where the length of evolution rules is at most 6.*

Proof. Consider a propositional formula $\varphi = C_1 \wedge C_2 \wedge \cdots \wedge C_m$, with $C_i = y_{i,1} \vee \cdots \vee y_{i,p_i}$, for some $m \geq 1$, $p_i \geq 1$, and $y_{i,j} \in \{x_k, \neg x_k \mid 1 \leq k \leq n\}$, for each $1 \leq i \leq m$, $1 \leq j \leq p_i$, where $\neg x_k$ is the negation of a propositional variable x_k, the two connections $\vee$, $\wedge$ are *or*, *and*, respectively.

Let us consider the polynomial time computable function $\langle m, n\rangle = ((m+n)(m+n+1)/2) + m$. It is primitive recursive and bijective from $\mathbb{N}^2$ onto $\mathbb{N}$. In what follows, we construct a family $\mathbf{\Pi} = \{\Pi_t \mid t \in \mathbb{N}\}$, such that giving the appropriate input multisets, each system Π_t will solve all instances of SAT with n variables and m clauses, where $t = \langle m, n\rangle$.

We use the following notations to encode a propositional formula φ:

$$cod(\varphi) = \alpha_{1,1} \dots \alpha_{1,n}\, \alpha_{2,1} \dots \alpha_{2,n} \dots\dots \alpha_{m,1} \dots \alpha_{m,n},$$

where for $1 \leq i \leq m$, $1 \leq j \leq n$ we have:

$$\alpha_{i,j} = \begin{cases} b_{i,j} & \text{if } x_j \text{ appears in } C_i; \\ b'_{i,j} & \text{if } \neg x_j \text{ appears in } C_i; \\ b''_{i,j} & \text{if } x_j \text{ and } \neg x_j \text{ do not appear in } C_i. \end{cases}$$

For each $m, n \in \mathbb{N}$, we construct the recognizer P system with proteins on membranes

$$\Pi_{\langle m,n\rangle} = (O, P, \Sigma, \mu, w_1/z_1, \dots, w_4/z_4, \emptyset, \mathcal{R}_1, \dots, \mathcal{R}_4, i_{in}),$$

with the following components:

- $O = \Sigma \cup \{a_j, a_j^{(1)}, a_j^{(2)}, a_j^{(3)}, t_j, f_j \mid 1 \leq i \leq n\} \cup \{d_i \mid 1 \leq i \leq m\} \cup \{a_{n+1}, c, \mathtt{yes}, \mathtt{no}\}$ is the set of objects,

- $P = \{p_j^+, p_j^-, q_j, q_j', r_{m+1,j} \mid 1 \le j \le n\} \cup \{p_k \mid 1 \le k \le n+m+1\} \cup \{r_{i,j} \mid 1 \le i \le m, 1 \le j \le n\} \cup \{r_{1,n+1}, q_{n+1}, s, s', s''\}$ is the set of proteins,
- $\Sigma = \{b_{i,j}, b'_{i,j}, b''_{i,j} \mid 1 \le i \le m, 1 \le j \le n\}$ is the input alphabet,
- $\mu = [\ [\ [\ \ ]_3\ ]_2 [\ \ ]_4\ ]_1$ is the initial membrane structure,
- $w_1 = \mathrm{no}$, $w_2 = a_1$, $w_3 = \lambda$ and $w_4 = \lambda$ are the initial multisets contained in membranes,
- $z_1 = s$, $z_2 = p_1$, $z_3 = r_{1,1}$, $z_4 = q_1$ are the initial proteins placed on membranes 1, 2, 3, 4, respectively,
- $i_{in} = 3$ is the input membrane,

and $\mathcal{R} = \bigcup_{1 \le i \le 4} \mathcal{R}_i$ is a finite set of rules of the following:

$G_{1,j} : [\ p_j \mid\ \ ]_2 \to [\ p_j^+ \mid\ \ ]_2 [\ p_j^- \mid\ \ ]_2,\ 1 \le j \le n.$
$G_{2,j} : [\ p_j^+ \mid a_j\ ]_2 \to [\ p_j^+ \mid t_j\ ]_2,\ 1 \le j \le n.$
$G_{3,j} : [\ p_j^- \mid a_j\ ]_2 \to [\ p_j^- \mid f_j\ ]_2,\ 1 \le j \le n.$
$G_{4,i,j} : \{t_j [\ r_{i,j} \mid b_{i,j}\ ]_3 \to d_i [\ r_{i+1,j} \mid t_j\ ]_3,$
$\quad t_j [\ r_{i,j} \mid b'_{i,j}\ ]_3 \to c [\ r_{i+1,j} \mid t_j\ ]_3,$
$\quad t_j [\ r_{i,j} \mid b''_{i,j}\ ]_3 \to c [\ r_{i+1,j} \mid t_j\ ]_3 \mid 1 \le i \le m, 1 \le j \le n\}.$
$G_{5,i,j} : \{f_j [\ r_{i,j} \mid b_{i,j}\ ]_3 \to c [\ r_{i+1,j} \mid f_j\ ]_3,$
$\quad f_j [\ r_{i,j} \mid b'_{i,j}\ ]_3 \to d_i [\ r_{i+1,j} \mid f_j\ ]_3,$
$\quad f_j [\ r_{i,j} \mid b''_{i,j}\ ]_3 \to c [\ r_{i+1,j} \mid f_j\ ]_3 \mid 1 \le i \le m, 1 \le j \le n\}.$
$G_{6,i,j} : [\ r_{i,j} \mid t_j\ ]_3 \to t_j [\ r_{i,j} \mid\ \ ]_3,\ 2 \le i \le m,\ 1 \le j \le n.$
$G_{7,i,j} : [\ r_{i,j} \mid f_j\ ]_3 \to f_j [\ r_{i,j} \mid\ \ ]_3,\ 2 \le i \le m, 1 \le j \le n.$
$G_{8,j} : [\ r_{m+1,j} \mid t_j\ ]_3 \to a_j^{(1)} [\ r_{1,j+1} \mid\ \ ]_3,\ 1 \le j \le n.$
$G_{9,j} : [\ r_{m+1,j} \mid f_j\ ]_3 \to a_j^{(2)} [\ r_{1,j+1} \mid\ \ ]_3,\ 1 \le j \le n.$
$G_{10,j} : [\ p_j^+ \mid a_j^{(1)}\ ]_2 \to a_j^{(1)} [\ p_j^- \mid\ \ ]_2,\ 1 \le j \le n.$
$G_{11,j} : [\ p_j^- \mid a_j^{(2)}\ ]_2 \to a_j^{(2)} [\ p_j^+ \mid\ \ ]_2,\ 1 \le j \le n.$
$G_{12,j} : a_j^{(2)} [\ p_j^- \mid\ \ ]_2 \to [\ p_j^- \mid a_j^{(3)}\ ]_2,\ 1 \le j \le n.$
$G_{13,j} : [\ p_j^- \mid a_j^{(3)}\ ]_2 \to a_j^{(3)} [\ p_j^+ \mid\ \ ]_2,\ 1 \le j \le n.$
$G_{14,j} : a_j^{(3)} [\ q_j \mid\ \ ]_4 \to [\ q_j' \mid a_{j+1}\ ]_4,\ 1 \le j \le n.$
$G_{15,j} : [\ q_j' \mid\ \ ]_4 \to [\ q_{j+1} \mid\ \ ]_4 [\ q_{j+1} \mid\ \ ]_4,\ 1 \le j \le n.$
$G_{16,j} : [\ q_j \mid a_j\ ]_4 \to a_j [\ q_j \mid\ \ ]_4,\ 2 \le j \le n+1.$
$G_{17,j} : a_{j+1} [\ p_j^+ \mid\ \ ]_2 \to [\ p_{j+1} \mid a_{j+1}\ ]_2,\ 1 \le j \le n.$
$C_{1,i} : [\ p_{n+i} \mid d_i\ ]_2 \to d_i [\ p_{n+i+1} \mid\ \ ]_2,\ 1 \le i \le m.$
$O_1 : [\ p_{n+m+1} \mid a_{n+1}\ ]_2 \to \mathrm{yes} [\ p_{n+m+1} \mid\ \ ]_2.$
$O_2 : [\ s' \mid \mathrm{yes}\ ]_1 \to \mathrm{yes} [\ s'' \mid\ \ ]_1.$
$O_3 : \mathrm{no} [\ s'' \mid\ \ ]_1 \to [\ s'' \mid \mathrm{no}\ ]_1.$
$O_4 : [\ s \mid \mathrm{no}\ ]_1 \to \mathrm{no} [\ s' \mid\ \ ]_1.$

In what follows, we give the overview of a computation to show how the propositional formula φ with n variables and m clauses is solved by the system $\Pi_{\langle m,n \rangle}$ with input $cod(\varphi)$.

The solution is proposed via a brute force algorithm, which consists in the following phases.

- Generation phase: the membrane with label 2 is divided for n time, all truth-assignments for the n variables are produced, and the clauses which are satisfiable by the corresponding truth-assignment are checked.
- Checking phase: checking whether there is a truth-assignment that makes the boolean formula evaluate to be true.
- Output phase: the system sends the right answer to the environment according to the results of the previous phase.

Let e be any time-mapping from $\mathcal{R}$ to $\mathbb{N}$, which represents the execution times for the rules from $\mathcal{R}$. In what follows, we will check how the above constructed system works in each phase.

Generation phase.

In the initial configuration of the system, we have object no in membrane 1 and protein s on membrane 1, object a_1 in membrane 2 and protein p_1 on membrane 2, input multiset $cod(\varphi)$ in membrane 3 and protein $r_{1,1}$ on membrane 3, and protein q_1 is placed on membrane 4, where there is no object initially.

At step 1, under the influence of the protein p_1 on membrane 2, division rule $G_{1,1}$ is applied, membrane 2 is divided into two copies of membrane 2, with the protein p_1 replaced by p_1^+ and p_1^-, respectively. For any given time-mapping e, the execution of rule $G_{1,1}$ completes in $e(G_{1,1})$ steps. At step 1, except for the application of rule $G_{1,1}$, under the influence of the protein s on membrane 1, the application of rule O_4 also starts; and from step 2 to step $e(G_{1,1})$, there is no rule starting. So, during the execution of rule $G_{1,1}$ (i.e., from step 1 to step $e(G_{1,1})$), there is one RS-step. Note that the number of RS-steps during the execution of rule $G_{1,1}$ is independent on the time-mapping e.

With the proteins p_1^+ and p_1^- present on two separate copies of membrane 2, rules $G_{2,1}$ and $G_{3,1}$ are enabled and applied at the same step, but the executions of rules $G_{2,1}$ and $G_{3,1}$ may complete at different steps due to the fact that the execution time associated with rules $G_{2,1}$ and $G_{3,1}$ can be different. By applying the rule $G_{2,1}$, the object a_1 evolves to t_1 (representing the truth value *true*); by using the rule $G_{3,1}$, the object a_1 evolves to f_1 (representing the truth value *false*).

When the object t_1 (resp. f_1) appears in membrane 2, and the protein $r_{1,1}$ placed on membrane 3, one of rules in $G_{4,1,1}$ (resp. $G_{5,1,1}$) is enabled, which corresponds to check whether the first clause is satisfied by the truth-assignment *true* (resp. *false*) of variable x_1. By using one of rules in $G_{4,1,1}$ (resp. $G_{5,1,1}$), the object d_1 appears in membrane 2 only when the object $b_{1,1}$ (resp. $b'_{1,1}$) is present in membrane 3; meanwhile, the protein $r_{1,1}$ placed on membrane 3 is changed to $r_{2,1}$. If membrane 3 contains the object $b'_{1,1}$ or $b''_{1,1}$ (resp. $b_{1,1}$ or $b''_{1,1}$), one copy of the object c will appear in membrane 2, and the protein placed on membrane 3 changed from $r_{1,1}$ to $r_{2,1}$. Note that the rule in $G_{4,1,1}$ and the rule in $G_{5,1,1}$ may start at different steps since the executions of rules $G_{2,1}$ and $G_{3,1}$ may finish at different steps.

If the object t_1 (resp. f_1) appears in membrane 3, under the influence of the protein $r_{2,1}$ on membrane 3, the application of rule $G_{6,2,1}$ (resp. $G_{7,2,1}$) starts, the object t_1 (resp. f_1) exits the membrane 3. With the object t_1 (resp. f_1) in membrane 2 and the protein $r_{2,1}$ on membrane 3, the execution of one of rules in $G_{4,2,1}$ (resp. $G_{5,2,1}$) starts, which corresponds to check whether the second clause is satisfied by the truth-assignment *true* (resp. *false*) of variable x_1. The object t_1 (resp. f_1) enters the membrane 3, and the object d_2 appears in membrane 2 if there exists object $b_{2,1}$ (resp. $b'_{2,1}$) in membrane 3, otherwise, the object c appears in membrane 2. If the object t_2 (resp. f_2) presents in membrane 3, under the influence of the protein $r_{3,1}$ on membrane 3, rule $G_{6,3,1}$ (resp. $G_{7,3,1}$) is enabled and applied, the object t_2 (resp. f_2) exits the membrane 3. In this way, rules in $G_{4,i,1}$ $(1 \le i \le m)$ and rules $G_{6,i,1}$ $(2 \le i \le m)$ (resp. rules in $G_{5,i,1}$ and rule $G_{7,i,1}$) can be applied one by one. All the clauses are checked that whether they are satisfied by the truth-assignment *true* (resp. *false*) of variable x_1. Rules $G_{4,i,1}$ and $G_{5,i,1}$ $(2 \le i \le m)$ may start at different steps since the executions of rules $G_{6,i,1}$ and $G_{7,i,1}$ may complete at different steps, rules $G_{6,i,1}$ and $G_{7,i,1}$ $(2 \le i \le m)$ may start at different steps since the executions of rules $G_{4,i-1,1}$ and $G_{5,i-1,1}$ may complete at different steps.

If the protein $r_{m+1,1}$ is present on membrane 3, which contains the object t_1 (resp. f_1), the application of rule $G_{8,1}$ (resp. $G_{9,1}$) starts, the object t_1 (resp. f_1) evolves to $a_1^{(1)}$ (resp. $a_1^{(2)}$), and the object $a_1^{(1)}$ (resp. $a_1^{(2)}$) exits the membrane 3, changing the protein from $r_{m+1,1}$ to $r_{1,2}$. With the object $a_1^{(1)}$ (resp. $a_1^{(2)}$) in membrane 2 and the protein p_1^+ (resp. p_1^-) on membrane 2, rule $G_{10,1}$ (resp. $G_{11,1}$) is enabled and used, the object $a_1^{(1)}$ (resp. $a_1^{(2)}$) exits membrane 2 and the protein p_1^+ (resp. p_1^-) placed on membrane 2 is changed to p_1^- (resp. p_1^+). Note that rules $G_{8,1}$ and $G_{9,1}$ may start at different steps since the executions of rules $G_{4,m,1}$ and $G_{5,m,1}$ may complete at different steps, rules $G_{10,1}$ and $G_{11,1}$ may start at different steps due to the fact that the executions of rules $G_{8,1}$ and $G_{9,1}$ may complete at different steps.

Rule $G_{12,1}$ is enabled only when the object $a_1^{(2)}$ appears in membrane 1 and there is a membrane 2 having the protein p_1^- placed on it, that is, rule $G_{12,1}$ can be used only when the executions of both rules $G_{10,1}$ and $G_{11,1}$ have completed. Thus, rule $G_{12,1}$ plays a synchronization function because $e(G_{2,1}) + \Sigma_{1\le i\le m} e(G_{4,i,1}) + \Sigma_{2\le i\le m} e(G_{6,i,1}) + e(G_{8,1}) + e(G_{10,1})$ may not equal to $e(G_{3,1}) + \Sigma_{1\le i\le m} e(G_{5,i,1}) + \Sigma_{2\le i\le m} e(G_{7,i,1}) + e(G_{9,1}) + e(G_{11,1})$. Anyway, when the execution of rule $G_{12,1}$ completes, the computation takes at most $4m+5$ RS-steps, which is independent on any time-mapping e.

By applying rule $G_{12,1}$, the object $a_1^{(2)}$ evolves to $a_1^{(3)}$ and the object $a_1^{(3)}$ enters membrane 2. When the execution of rule $G_{12,1}$ finishes, under the influence of the protein p_1^- placed on membrane 2, the application of rule $G_{13,1}$ starts, the object $a_1^{(3)}$ exits membrane 2 changing the protein from p_1^- to p_1^+. With the object $a_1^{(3)}$ in membrane 1 and the protein q_1 on membrane 4, rule $G_{14,1}$ is enabled and applied, the object $a_1^{(3)}$ evolves to a_2, and the object a_2 enters membrane 4, changing the protein from q_1 to q'_1. When the protein q'_1 is present on membrane 4, rule $G_{15,1}$ is enabled and applied, membrane 4 is divided into two copies of membranes with the same label, and the protein q'_1 is replaced by q_2 on each membrane. The division rule $G_{15,1}$ has a copy function, which makes the number of the object a_2 doubled. When the protein q_2 appears on membrane 4, the application of rule $G_{16,2}$ in all membranes 4 starts and completes at the same step (the execution of rule $G_{16,2}$ in all membranes 4 takes the same time $e(G_{15,1})$ for a given time-mapping e). By using rule $G_{16,2}$, the object a_2 exits membrane 4. With the object a_2 in membrane 1 and the protein p_1^+ on membrane 2, rule $G_{17,1}$ is enabled and used, each object a_2 enters a membrane 2, changing the protein from p_1^+ to p_2 (there are two copies of object a_2 in membrane 1 and two membranes 2 with protein p_1^+ placed on, and the system works in a maximally parallel manner).

In general, when the object a_2 appears in membrane 2, the computation takes at most $4m+10$ RS-steps.

The appearance of object a_2 in membrane 2 (or the protein p_2 appears on membrane 2) means that the system completes the following processes: assigning truth-assignment of variable x_1, and looking for the clauses satisfied by the truth-assignment of variable x_1; and the system starts the following processes: assigning truth-assignment of variable x_2, and looking for the clauses satisfied by the truth-assignment of variable x_2.

When the protein p_2 appears on membrane 2, rule $G_{1,2}$ is enabled and applied. Note that the application of rule $G_{1,2}$ in all membranes 2 starts at the same step and completes at the same step, since the execution of rule $G_{1,2}$ in all membranes 2 takes the same time $e(G_{1,2})$ for a given time-mapping e. The application of rules $G_{2,2}$ and $G_{3,2}$ starts at the same step, but may complete at different steps. Furthermore, the application of rules $G_{4,i,2}$ $(1 \le i \le m)$, $G_{5,i,2}$ $(1 \le i \le m)$, $G_{6,i,2}$ $(2 \le i \le m)$,

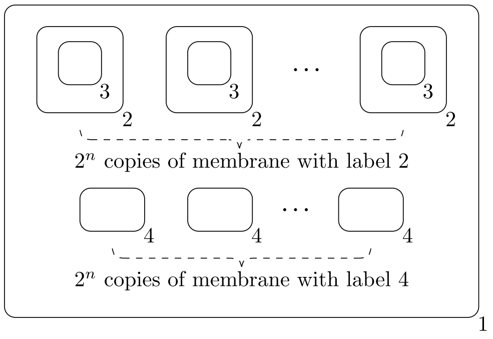
3
2
2^n copies of membrane with label 2
4
2^n copies of membrane with label 4
1

Fig. 1. The membrane structure at the moment when generation phase completes.

$G_{7,i,2}$ $(2 \le i \le m)$, $G_{8,2}$, $G_{9,2}$, $G_{10,2}$ and $G_{11,2}$ may start and complete at different steps. However, rule $G_{12,2}$ is enabled only when all rules that have started and completed their executions. Similar to the case of variable x_1, the processes of assigning truth-assignment of variable x_2 as well as looking for the clauses satisfied by the truth-assignment of variable x_2 take at most $4m + 10$ RS-steps.

The system continues to assign the truth-assignment of variables $x_3, x_4, \dots, x_n$ and look for the clauses satisfied by the truth-assignment of variables. In general, after $4mn + 10n$ RS-steps, 2^n copies of membrane with label 2 and 2^n copies of membrane with label 4 are generated, all of which are placed in membrane 1, and in each copy of membrane 2, there exists a membrane 3 (see Fig. 1 for the membrane structure at the moment when generation phase completes).

Checking phase.

When the generation phase completes, each membrane with label 2 contains some of objects from the set $\{d_1, d_2, \dots, d_m\}$, whose elements denote the corresponding clauses satisfied by the truth assignments of the variables. If there is at least one membrane with label 2 that contains all the objects $d_1, d_2, \dots, d_m$, which means the corresponding truth assignment in that membrane satisfies all clauses, hence formula φ is satisfiable; if there is no membrane with label 2 that contains all the objects $d_1, d_2, \dots, d_m$, the formula φ is not satisfiable.

At the last of the generation phase, under the influence of the protein p_n^+ on membrane 2, all the objects a_{n+1} enter membranes 2, changing the protein from p_n^+ to p_{n+1} at the same step (there are 2^n copies of object a_{n+1} and 2^n copies of membrane 2, each object a_{n+1} enters a membrane 2 by applying the rule $G_{17,n}$ in a maximally parallel manner). Thus, the protein p_{n+1} placed on each membrane with label 2 appears at the same step.

When the execution of rule $G_{17,n}$ completes, the system starts to check whether the object d_1 presents in each membrane 2 (i.e., checking whether the corresponding true assignment satisfies the clause C_1). If the object d_1 appears in a membrane 2, with the protein p_{n+1} on membrane 2, rule $C_{1,1}$ is enabled and applied, the object d_1 exits membrane 2, changing the protein from p_{n+1} to p_{n+2} on membrane 2. When the execution of rule $C_{1,1}$ finishes, the system starts to check whether the object d_2 appears in a membrane 2 (i.e., checking whether the corresponding true assignment satisfies the clause C_2). The proteins p_{n+3} appear only when the corresponding membranes 2 contain the objects d_2.

The system continues to check whether the objects $d_3, d_4, \dots, d_m$ appear in membranes 2. For any membrane 2 that does not contain the object d_i, $i = 1, 2, \dots, m$, then the computation in this membrane stops at the time when $C_{1,i}$ is supposed to be applied. In general, the checking phase takes at most m RS-steps (if there is a membrane 2 contains all objects $d_1, d_2, \dots, d_m$, the process takes m RS-steps).

Output phase.

In the initial configuration of the system, membrane 1 contains the object $\texttt{no}$. At step 1, with the protein s on membrane 1, rule O_4 is applied, where the object $\texttt{no}$ exits membrane 1, changing the protein from s to s'. When checking phase finishes, we have the following two cases.

- If no protein p_{n+m+1} is present on any membrane with label 2, then rules O_1, O_2, O_3 cannot be applied. In this case, when the system halts, the object $\texttt{no}$ remains in the environment, which means that the formula is not satisfiable.
- If there exits at least one membrane 2 with the protein p_{n+m+1} on it, then rule O_1 will be used, the object a_{n+1} evolves to $\texttt{yes}$, and the object $\texttt{yes}$ exits membrane 2. When the execution of rule O_1 finishes, at this moment, if the execution of rule O_4 is not yet finished, then no rule can be started in the system. Only when the execution of rule O_4 completes, the protein on membrane 1 is changed to s', rule O_2 will be enabled. By applying rule O_2, the object $\texttt{yes}$ exits to the environment, changing the protein from s' to s'' (it ensures that only one copy of object $\texttt{yes}$ exits to the environment if there exists more than one copy of object $\texttt{yes}$ in membrane 1). When the protein s'' presents on membrane 1, rule O_3 will be applied, the object $\texttt{no}$ enters membrane 1. In this case, when the system halts, one copy of object $\texttt{yes}$ remains in the environment, which means that the formula is satisfiable. This process takes three RS-steps.

According to the analysis above, it is obvious that for any time-mapping $e : R \to \mathbb{N}$, the object `yes` appears in the environment when the computation halts if and only if the formula φ is satisfiable; and the object `no` presents in the environment when the computation halts if and only if the formula φ is not satisfiable. Thus, the system $\Pi_{\langle m,n \rangle}$ is time-free sound and time-free complete.

For any time-mapping $e : R \to \mathbb{N}$, if the formula φ is satisfiable, the computation takes at most $4mn + 10n + m + 3$ RS-steps: it takes at most $4mn + 10n$ RS-steps to generate 2^n copies of membranes with label 2 and 2^n copies of membranes with label 4; it takes m RS-steps to check whether all clauses are satisfied by an assignment; the output phase takes three RS-steps, and the system halts. If the formula φ is not satisfiable, the computation takes at most $4mn + 10n + m$ RS-steps (it takes no RS-step at the output phase), and the system halts. Therefore, the family of P systems with proteins on membranes is time-free polynomially bounded.

The family $\mathbf{\Pi} = \{\Pi_{\langle m,n \rangle} \mid m, n \in \mathbb{N}\}$ defined above is polynomially uniform by Turing machines because the construction of P system is built in a polynomial time with respect to the size parameters n and m, and necessary resources are as follows:

– size of the set O: $3mn + 6n + m + 4$;
– size of the set P: $mn + 6n + m + 6$;
– initial number of membranes: 4;
– total size of initial multisets of objects: $mn + 2$;
– initial number of proteins on membranes: 4;
– number of rules: $8mn + 11n + m + 4$;
– the maximal length of a rule including proteins: 6.

We omit the detailed construction due to the fact that it is straightforward but cumbersome as explained in the proof of Theorem 7.2.3 in [16].

Therefore, the `SAT` problem can be solved in polynomial RS-steps by a family of recognizer P systems with proteins on membranes in a time-free uniform manner, and the length of evolution rules is at most 6. That is, we have shown that $\mathtt{SAT} \in \mathbf{PMC}^{tf}_{CPE(6)}$. $\square$

4. Conclusions and further works

In this work, we investigate the computational efficiency of timed P systems with proteins on membranes. Specifically, we construct a family of P systems with proteins on membranes, which solves `SAT` problem in a time-free uniform manner, where the correctness of the solution is irrelevant to the time used by the involved rules, and the P systems are constructed from the sizes of instances.

The solution to `SAT` problem given in Section 3 uses the division rules for elementary and non-elementary membranes. It remains open how we construct P systems with proteins on membranes and only elementary membrane division rules to time-freely solve the **NP**-complete problem.

The P system constructed in the proof of Theorem 3.1 has the maximum length of evolution rules 6. It remains open whether the `SAT` problem can be efficiently solved by P systems with maximum length of evolution rules 4, that is, avoiding evolution rules of type (e).

Flip-flop membrane systems with proteins are a particular class of P systems with proteins on membranes [12]. In such P systems, each protein has at most two states: p and $\bar{p}$, and the rules are used either without changing the protein or changing the protein from p to $\bar{p}$ and back. It is of interest to investigate whether we can give a time-free uniform solution to `SAT` problem by using flip-flop membrane systems with proteins.

Proteins that used to control evolution rules are also investigated in tissue-like P systems [22], where there is one and only one copy of protein placed on each cell, and multisets of objects together with proteins between cells are exchanged if a symport/antiport rule is used between these cells in a computation step. It remains open whether we can construct tissue-like P systems with protein on cells and cell division to solve **NP**-complete problems in the context of time-freeness.

Recently, cell-like spiking neural P systems have been proposed in [28], where only one kind of objects (i.e., spikes) is considered. It is proved that cell-like spiking neural P systems as number generating devices are Turing universal [28]. It is interesting to construct cell-like spiking neural P systems with cell division to solve **NP**-complete problems.

It is interesting to consider a more general variant of P systems with proteins on membranes where each application of a rule has an execution time and study if there exists clock-free uniform solution to SAT.

Acknowledgments

The work of B. Song and L. Pan was supported by National Natural Science Foundation of China (61033003, 61320106005 and 61472154), Ph.D. Programs Foundation of Ministry of Education of China (2012014213008), the Innovation Scientists and Technicians Troop Construction Projects of Henan Province (154200510012). The work of M.J. Pérez-Jiménez was supported by "Ministerio de Economía y Competitividad" of Spain (TIN2012-37434), cofunded by FEDER funds.

References

[1] A. Alhazov, On determinism of evolution-communication P systems, J. Univers. Comput. Sci. 10 (2004) 502–508.

[2] M. Cavaliere, D. Sburlan, Time-independent P systems, in: 5th International Workshop, WMC 2004, in: Lect. Notes in Comput. Sci., vol. 3365, Springer-Verlag, 2005, pp. 239–258.

[3] R. Freund, Gh. Păun, M.J. Pérez-Jiménez, Tissue P systems with channel states, Theor. Comput. Sci. 330 (1) (2005) 101–116.

[4] M.R. Garey, D.J. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W.H. Freeman, San Francisco, 1979.

[5] M. Gheorghe, Gh. Păun, M.J. Pérez-Jiménez (Eds.), Research Frontiers of Membrane Computing: Open Problems and Research Topics: Section 11, M. Cavaliere, Time-free solutions to hard computational problems, Int. J. Found. Comput. Sci. 24 (5) (2013) 579–582.

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

[7] S. Krishna, On the computational power of flip-flop proteins on membranes, in: Third Conference on Computability in Europe, CiE 2007, in: Lect. Notes in Comput. Sci., vol. 4497, Springer, 2007, pp. 695–704.

[8] C. Martin-Vide, J. Pazos, Gh. Păun, A. Rodriguez-Patón, Tissue P systems, Theor. Comput. Sci. 296 (2) (2003) 295–326.

[9] L. Pan, M.J. Pérez-Jiménez, Computational complexity of tissue-like P systems, J. Complex. 26 (3) (2010) 296–315.

[10] L. Pan, X. Zeng, Small universal spiking neural P systems working in exhaustive mode, IEEE Trans. Nanobiosci. 10 (2) (2011) 99–105.

[11] A. Păun, Gh. Păun, The power of communication: P systems with symport/antiport, New Gener. Comput. 20 (3) (2002) 295–306.

[12] A. Păun, B. Popa, P systems with proteins on membranes, Fundam. Inform. 72 (2006) 467–483.

[13] A. Păun, B. Popa, P systems with proteins on membranes and membrane division, in: 10th International Conference, DLT 2006, in: Lect. Notes in Comput. Sci., vol. 4036, Springer-Verlag, 2006, pp. 292–303.

[14] A. Păun, A. Rodríguez-Patón, On flip-flop membrane systems with proteins, in: 8th International Workshop, WMC 2007, in: Lect. Notes in Comput. Sci., vol. 4860, Springer-Verlag, 2007, pp. 414–427.

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

[16] Gh. Păun, Membrane Computing: An Introduction, Springer-Verlag, Berlin, 2002.

[17] Gh. Păun, G. Rozenberg, A. Salomaa (Eds.), Handbook of Membrane Computing, Oxford University Press, Oxford, 2010.

[18] G. Rozenberg, A. Salomaa (Eds.), Handbook of Formal Languages, vol. 3, Springer-Verlag, Berlin, 1997.

[19] T. Song, L.F. Macías-Ramos, L. Pan, M.J. Pérez-Jiménez, Time-free solution to SAT problem using P systems with active membranes, Theor. Comput. Sci. 529 (2014) 61–68.

[20] B. Song, L. Pan, Computational efficiency and universality of timed P systems with active membranes, Theor. Comput. Sci. 567 (2015) 74–86.

[21] T. Song, L. Pan, Spiking neural P systems with rules on synapses working in maximum spikes consumption strategy, IEEE Trans. Nanobiosci. 14 (1) (2015) 37–43.

[22] B. Song, L. Pan, M.J. Pérez-Jiménez, Tissue P systems with protein on cells, Fundam. Inform. 144 (2016) 77–107.

[23] B. Song, M.J. Pérez-Jiménez, L. Pan, Computational efficiency and universality of timed P systems with membrane creation, Soft Comput. 19 (11) (2015) 3043–3053.

[24] B. Song, M.J. Pérez-Jiménez, L. Pan, Efficient solutions to hard computational problems by P systems with symport/antiport rules and membrane division, Biosystems 130 (2015) 51–58.

[25] B. Song, T. Song, L. Pan, Time-free solution to SAT problem by P systems with active membranes and standard cell division rules, Nat. Comput. 14 (4) (2015) 673–681.

[26] B. Song, T. Song, L. Pan, A time-free uniform solution to subset sum problem by tissue P systems with cell division, Math. Struct. Comput. Sci. (2016), http://dx.doi.org/10.1017/S0960129515000018.

[27] P. Sosík, A. Păun, A. Rodríguez-Patón, P systems with proteins on membranes characterize PSPACE, Theor. Comput. Sci. 488 (2013) 78–95.

[28] T. Wu, Z. Zhang, Gh. Păun, L. Pan, Cell-like spiking neural P systems, Theor. Comput. Sci. (2016), http://dx.doi.org/10.1016/j.tcs.2015.12.038.