

Computational complexity of tissue-like P systems

Linqiang Pan^{a,*}, Mario J. Pérez-Jiménez^b

^a Key Laboratory of Image Processing and Intelligent Control, Department of Control Science and Engineering, 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, 41012 Sevilla, Spain

A B S T R A C T

Membrane systems, also called P systems, are biologically inspired theoretical models of distributed and parallel computing. This paper presents a new class of tissue-like P systems with cell separation, a feature which allows the generation of new workspace. We study the efficiency of the class of P systems and draw a conclusion that only tractable problems can be efficiently solved by using cell separation and communication rules with the length of at most 1. We further present an efficient (uniform) solution to SAT by using cell separation and communication rules with length at most 6. We conclude that a borderline between efficiency and non-efficiency exists in terms of the length of communication rules (assuming $P \neq NP$). We discuss future research topics and open problems.

Keywords:

Membrane computing
Tissue P systems
Cell separation
SAT

1. Introduction

Membrane computing is a young branch of natural computing initiated by Păun at the end of 1998 [17]. It has received great attention from the scientific community since then. Computer scientists, biologists, formal linguists, and complexity theoreticians have contributed greatly to this new field of research, enriching each others' with results, open problems and promising new research lines. Membrane computing was selected by the Institute for Scientific Information, USA, as a fast *Emerging Research Front* in computer science, and the paper of Păun and Păun [19] was ranked in [25] as a highly cited paper in October 2003.

* Corresponding author.

E-mail addresses: lqpan@mail.hust.edu.cn (L. Pan), marper@us.es (M.J. Pérez-Jiménez).

Membrane computing is inspired by the structure and function of living cells, as well as the organization of cells in tissues, organs, and other higher order structures. The devices investigated in this field, called *P systems*, are distributed parallel and non-deterministic computing models.

In general, the main components of *P systems* are a *membrane structure*, where *multisets of symbol-objects* are placed in its *compartments*. Multisets of symbol-objects evolve in a synchronous maximally parallel manner according to given *evolution rules*. Evolution rules are associated with the membranes. Please refer to [18] or [4] for an introduction and [26] for further references.

In recent years, many different models of *P systems* have been proposed. The most studied variants are characterized by a *cell-like* membrane structure, where communication takes place between a membrane and the surrounding membrane. These models have a set of nested membranes, chosen in such a way that the graph of neighborhood relations is a tree.

One of the topics in the field is the study of the computational power and efficiency of *P systems*. In particular, different models of these cell-like *P systems* have been successfully used for designing solutions to **NP**-complete problems in polynomial time (see [7] and the references therein). These solutions are obtained by generating an exponential workspace in polynomial time and simultaneously checking all the candidate solutions in parallel. With inspiration from the living cell, several ways to obtain exponential workspace in polynomial time have been proposed: membrane division (*mitosis*) [16], membrane creation (*autopoiesis*) [9], membrane separation (*membrane fission*) [15]. These three ways have given rise to the corresponding *P system* models: *P systems with active membranes*, *P systems with membrane creation*, and *P systems with membranes separation*. These three models are universal from a computational point of view, but technically, they are very different. To the best of our knowledge, there exists no theoretical result which proves that these models can simulate each other in polynomial time.

Under the hypothesis $\mathbf{P} \neq \mathbf{NP}$, it was shown that *P systems* without membrane division cannot solve **NP**-complete problems in polynomial time [24]. This result was generalized in [21]: an **NP**-complete problem cannot be solved in polynomial time by means of language accepting *P systems* (without using rules that allow an increase in the size of the structure of membranes).

Here, we focus on *P systems* of another type, called *tissue P systems* due to the structure of their membrane. Instead of considering a hierarchical structure, membranes are placed at the nodes of a graph. This variant takes inspiration from two biological phenomena (see [14]): intercellular communication and communication between neurons. The common mathematical model of these two mechanisms is a net of processors dealing with symbols and communicating these symbols along channels specified in advance. Communication among cells is based on symport/antiport rules, which were introduced to *P systems* in [19]. Symport rules move objects across a membrane together in one direction, whereas antiport rules move objects across a membrane in opposite directions.

From the seminal definitions of tissue *P systems* [13,14], several research lines have been developed and other variants have arisen (see, for example, [1-3,10,11,23]). One of the most interesting variants of tissue *P systems* was presented in [20]. In that paper, the definitions of tissue *P systems* and *P systems* with active membranes are combined. This yields *tissue P systems with cell division*, and a polynomial-time uniform solution to the **NP**-complete problem SAT is shown. In these kinds of tissue *P systems* [20], replication is used, that is, the two new cells generated by a division rule have exactly the same objects except for at most a pair of different objects. However, in the biological phenomenon of separation, the contents of the two new cells evolved from the original one can be significantly different, and membrane separation inspired by this biological phenomenon in the framework of cell-like *P systems* was proved to be an efficient way for obtaining exponential workspace in polynomial time [15]. In this paper, a new class of *P systems*, called *tissue P systems with cell separation*, is presented. We study the efficiency of the class of *P systems* and draw a conclusion that only tractable problems can be efficiently solved by using cell separation and communication rules with the length of at most 1. We further present an efficient (uniform) solution to SAT by using cell separation and communication rules with length at most 6. We conclude that a borderline between efficiency and non-efficiency exists in terms of the length of communication rules (assuming $\mathbf{P} \neq \mathbf{NP}$). We discuss future research topics and open problems.

2. Preliminaries

An *alphabet* Σ is a non-empty set, whose elements are called *symbols*. An ordered sequence of symbols is a *string*. The number of symbols in a string u is the *length* of the string, and it is denoted by $|u|$. As usual, the empty string (with length 0) will be denoted by λ . The set of strings of length n built with symbols from the alphabet Σ is denoted by Σ^n and $\Sigma^* = \cup_{n \geq 0} \Sigma^n$. A *language* over Σ is a subset of Σ^* .

A *multiset* m over a set A is a pair (A, f) where $f : A \rightarrow \mathbb{N}$ is a mapping and \mathbb{N} is the set of natural numbers. If $m = (A, f)$ is a multiset then its *support* is defined as $\text{supp}(m) = \{x \in A \mid f(x) > 0\}$ and its *size* is defined as $\sum_{x \in A} f(x)$. A multiset is empty (resp. finite) if its support is the empty set (resp. finite).

If $m = (A, f)$ is a multiset over A , and $\text{supp}(m) = \{a_1, \dots, a_k\}$, then it will be denoted as $m = \{\{a_1^{f(a_1)}, \dots, a_k^{f(a_k)}\}\}$. Superscripts indicate the multiplicity of each element. If $f(x) = 0$ for any $x \in A$, then this element is omitted. If $m_1 = (A, f)$ and $m_2 = (A, g)$ are multisets over A , then the union of m_1 and m_2 is defined as the function $m_1 m_2 = (A, h)$, where $h = f + g$. In the following, we assume that the reader is already familiar with the basic notions and the terminology of P systems. See [18] for details.

3. Tissue P systems with cell separation

In the first definition of the model of tissue P systems [13,14], the membrane structure did not change during the computation. We will give a new model of *tissue P systems with cell separation* based on the cell-like model of P systems with membrane separation [15]. The biological inspiration is the following: live tissues are not *static* networks of cells, since new cells are generated by membrane fission in a natural way.

The main features of this model, from a computational point of view, are that cells are not polarized (the opposite holds in the cell-like model of P systems with active membranes; see [18]); the cells obtained by separation have the same labels as the original cell. If a cell is separated, its interaction with other cells or with the environment is blocked during the separation process. This means that a cell closes its communication channels while it is separating.

Formally, a *tissue P system with cell separation* of degree $q \geq 1$ is a tuple

$$\Pi = (\Gamma, O_1, O_2, w_1, \dots, w_q, \mathcal{E}, \mathcal{R}, i_0),$$

where:

- (1) Γ is an *alphabet* whose elements are called *objects*, $\Gamma = O_1 \cup O_2$, $O_1, O_2 \neq \emptyset$, $O_1 \cap O_2 = \emptyset$;
- (2) w_1, \dots, w_q are strings over Γ , representing the multisets of objects placed in the q cells of the system at the beginning of the computation;
- (3) $\mathcal{E} \subseteq \Gamma$ is an alphabet representing the set of objects in the environment in arbitrary copies of each;
- (4) \mathcal{R} is a finite set of rules of the following forms:
 - (a) $(i, u/v, j)$, for $i, j \in \{0, 1, 2, \dots, q\}$, $i \neq j$, $u, v \in \Gamma^*$;
communication rules: 1, 2, \dots , q identify the cells of the system, 0 is the environment; when applying a rule $(i, u/v, j)$, the objects of the multiset represented by u are sent from region i to region j and simultaneously the objects of the multiset v are sent from region j to region i ($|u| + |v|$ is the *length* of the communication rule $(i, u/v, j)$);
 - (b) $[a]_i \rightarrow [O_1]_i [O_2]_i$, where $i \in \{1, 2, \dots, q\}$ and $a \in \Gamma$, and $i \neq i_0$;
separation rules: with the presence of an object a , the cell is separated into two cells with the same label; at the same time, object a is consumed; the objects from O_1 are placed in the first cell, those from O_2 are placed in the second cell; the output cell i_0 cannot be divided;
- (5) The output region i_0 is the environment.

The rules of a system like the ones above are used in the non-deterministic maximally parallel manner, as is customary in membrane computing. During each step, all cells which can evolve must evolve in a maximally parallel way (in each step the system applies a multiset of rules which is maximal: no further rule can be added). This way of applying rules has only one restriction: when a cell is separated, the separation rule is the only one which is applied to that cell during that step; the objects inside that cell do not evolve by means of communication rules. The new cells can participate in the interaction with other cells or the environment by means of communication rules in the next step—providing that they are not separated once again. Their labels precisely identify the rules which can be applied to them.

A configuration of a tissue P system with cell separation is described by the multisets of objects over Γ associated with all the cells present in the system and the multiset of objects over $\Gamma - \mathcal{E}$ associated with environment. The tuple $(w_1, w_2, \dots, w_q; \emptyset)$ is the initial configuration. A halting configuration is a configuration such that there is no rule which can be applied. The computation starts from the initial configuration and proceeds as defined before; only halting computations (reaching a configuration where no rule can be applied) give a result, and the result is encoded by the multiset of objects over $\Gamma - \mathcal{E}$ appearing in the environment i_0 in the halting configuration.

4. Recognizer tissue P systems with cell separation

NP-completeness has usually been studied in the framework of *decision problems*, that is, pairs (I_X, θ_X) where I_X is a language over an alphabet (whose elements are called *instances*) and θ_X is a total Boolean function over I_X .

In order to study the computing efficiency, the notions from classical *computational complexity theory* are adapted for membrane computing. A class of cell-like P systems is introduced in [22]: *recognizer P systems*. For tissue P systems, with the same idea as for recognizer cell-like P systems, *recognizer tissue P systems* are introduced in [20].

A recognizer tissue P system with cell separation of degree $q \geq 1$ is a tuple

$$\Pi = (\Gamma, O_1, O_2, \Sigma, w_1, \dots, w_q, \mathcal{E}, \mathcal{R}, i_{\text{in}}, i_0)$$

where:

- $(\Gamma, O_1, O_2, w_1, \dots, w_q, \mathcal{E}, \mathcal{R}, i_0)$ is a tissue P system with cell separation of degree $q \geq 1$ (as defined in the previous section).
- The working alphabet Γ has two distinguished objects *yes* and *no*, with at least one copy of them present in some initial multisets w_1, \dots, w_q , but none of them present in \mathcal{E} .
- Σ is an (input) alphabet strictly contained in Γ .
- $i_{\text{in}} \in \{1, \dots, q\}$ is the input cell.
- The output region i_0 is the environment.
- 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.

The computations of the system Π with input $w \in \Sigma^*$ start from a configuration of the form $(w_1, w_2, \dots, w_{i_{\text{in}}} w, \dots, w_q; \emptyset)$. That is, the computations start after adding the multiset w to the content of the input cell i_{in} . We say that \mathcal{C} is an accepting computation (respectively, rejecting computation) if object *yes* (respectively, *no*) appears in the environment associated with the corresponding halting configuration of \mathcal{C} .

We denote by **TSC**(k) the class of recognizer tissue P systems with cell separation and with communication rules of length at most k .

Definition 1. A decision problem $X = (I_X, \theta_X)$ is solvable in polynomial time by a family $\Pi = \{\Pi(n) \mid n \in \mathbb{N}\}$ of recognizer tissue P systems with cell separation if the following conditions hold:

- The family Π is *polynomially uniform* via Turing machines. That is, there exists a deterministic Turing machine working in polynomial time which constructs the system $\Pi(n)$ 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 $\text{cod}(u)$ is an input multiset of the system $\Pi(s(u))$;
 - the family Π is *polynomially bounded* with regard to (X, cod, s) ; that is, there exists a polynomial function p such that for each $u \in I_X$ every computation of $\Pi(s(u))$ with input $\text{cod}(u)$ is halting and it performs at most $p(|u|)$ steps;
 - the family Π is *sound* with regard to (X, cod, s) ; that is, for each $u \in I_X$, if there exists an accepting computation of $\Pi(s(u))$ with input $\text{cod}(u)$, then $\theta_X(u) = 1$;
 - the family Π is *complete* with regard to (X, cod, s) ; that is, for each $u \in I_X$, if $\theta_X(u) = 1$, then every computation of $\Pi(s(u))$ with input $\text{cod}(u)$ is an accepting one.

Each recognizer P system with cell separation in [Definition 1](#) is *confluent* in the sense that all possible computations must give the same answer for a given instance.

We denote by $\text{PMC}_{\text{TSC}(k)}$ the set of all decision problems which can be solved by means of recognizer tissue P systems from $\text{TSC}(k)$.

5. The limitation on the efficiency of TSC(1)

In this section, we consider the efficiency of tissue P systems with cell separation and communication rules of length 1. Specifically, we will prove that such systems can efficiently solve only tractable problems in the classical sense.

Let Π be a tissue P system with cell separation and let all communication rules be of length 1. In this case, each rule of the system can be activated by a single object. Hence, there exists, in some sense, a *dependency* between the object triggering the rule and the object or objects produced by its application. This dependency allows us to adapt the ideas developed in [6] for cell-like P systems with active membranes to tissue P systems with cell separation and communication rules of length 1.

Definition 2. Let $\Pi = (\Gamma, O_1, O_2, \Sigma, \mathcal{M}_1, \dots, \mathcal{M}_q, \mathcal{E}, \mathcal{R}, i_{\text{in}}, i_0)$ be a tissue P system of degree $q \geq 1$ with cell separation and communication rules of length 1. Let $H = \{0, 1, \dots, q\}$. The *dependency graph associated with Π* is the directed graph $G_\Pi = (V_\Pi, E_\Pi)$ defined as follows:

$$V_\Pi = \{(a, i) \in \Gamma \times H \mid \text{there is } j \in H \text{ such that } (i, a/\lambda, j) \in \mathcal{R} \text{ or } (j, a/\lambda, i) \in \mathcal{R}\},$$

$$E_\Pi = \{((a, i), (b, j)) \mid a = b \text{ and } (i, a/\lambda, j) \in \mathcal{R}\}.$$

Note that when a separation rule is applied, objects do not evolve and the labels of membranes do not change. So separation rules do not add any nodes and edges to the associated dependency graph.

Lemma 3. Let $\Pi = (\Gamma, O_1, O_2, \Sigma, \mathcal{M}_1, \dots, \mathcal{M}_q, \mathcal{E}, \mathcal{R}, i_{\text{in}}, i_0)$ be a tissue P system with cell separation, in which the length of all communication rules is 1. Let $H = \{0, 1, \dots, q\}$. Then there exists a deterministic Turing machine that constructs the dependency graph G_Π associated with Π in polynomial time (that is, the computation time is bounded by a polynomial function depending on the total number of communication rules).

Proof. A deterministic algorithm that, given a P system Π with the set R_c of communication rules, constructs the corresponding dependency graph, is the following:

```

Input:  $\Pi$  (with  $R_c$  as its set of communication rules)
 $V_\Pi \leftarrow \emptyset; E_\Pi \leftarrow \emptyset$ 
for each rule  $r \in R_c$  of  $\Pi$  do
  if  $r = (i, a/\lambda, j)$  then
     $V_\Pi \leftarrow V_\Pi \cup \{(a, i), (a, j)\}; E_\Pi \leftarrow E_\Pi \cup \{((a, i), (a, j))\}$ 

```

The running time of this algorithm is bounded by $O(|R_c|)$. \square

Lemma 4. Let $\Pi = (\Gamma, O_1, O_2, \Sigma, \mathcal{M}_1, \dots, \mathcal{M}_q, \mathcal{E}, \mathcal{R}, i_{\text{in}}, i_0)$ be a tissue P system with cell separation, in which the length of all communication rules is 1. Let $H = \{0, 1, \dots, q\}$. Let Δ_Π be defined as follows:

$$\Delta_\Pi = \left\{ (a, i) \in V_\Pi \mid \text{there exists a path (within the dependency graph of } \Pi) \text{ from } (a, i) \text{ to } (\text{yes}, 0) \right\}.$$

Then, there exists a Turing machine that constructs the set Δ_Π in polynomial time (that is, the computation time is bounded by a polynomial function depending on the total number of communication rules).

Proof. We can construct the set Δ_Π from Π as follows:

- We construct the dependency graph G_Π associated with Π .
- Then we consider the following algorithm:

```

Input:  $G_\Pi = (V_\Pi, E_\Pi)$ 
 $\Delta_\Pi \leftarrow \emptyset$ 
for each  $(a, i) \in V_\Pi$  do
  if reachability ( $G_\Pi, (a, i), (\text{yes}, 0)$ ) = yes then
     $\Delta_\Pi \leftarrow \Delta_\Pi \cup \{(a, i)\}$ 

```

The *Reachability Problem* is the following: given a (directed or undirected) graph G and two nodes a, b , determine whether or not the node b is reachable from a , that is, whether or not there exists a path in the graph from a to b . It is easy to design an algorithm running in polynomial time solving this problem. For example, given a (directed or undirected) graph G and two nodes a, b , we consider a depth-first search with source a , and we check whether b belongs to the tree having root a in the computation forest. The total running time of this algorithm is $O(|V| + |E|)$, that is, in the worst case, quadratic in the number of nodes.

Therefore, the running time of the above algorithm for constructing the set Δ_Π is of order $O(|V_\Pi| \cdot |V_\Pi|^2)$, hence it is of order $O(|\Gamma|^3 \cdot |H|^3)$. \square

Let $\Pi = (\Gamma, O_1, O_2, \Sigma, \mathcal{M}_1, \dots, \mathcal{M}_q, \mathcal{E}, \mathcal{R}, i_{\text{in}}, i_0)$ be a tissue P system with cell separation. Let m be a multiset over Σ . Then we define $\mathcal{M}_j^* = \{(a, j) \mid a \in \mathcal{M}_j\}$, for $1 \leq j \leq q$, and $m^* = \{(a, i_{\text{in}}) \mid a \in m\}$.

Below we characterize accepting computations of a recognizer tissue P system with cell separation and communication rules of length 1 by distinguished paths in the associated dependency graph.

Lemma 5. Let $\Pi = (\Gamma, O_1, O_2, \Sigma, \mathcal{M}_1, \dots, \mathcal{M}_q, \mathcal{E}, \mathcal{R}, i_{\text{in}}, i_0)$ be a recognizer tissue P system with cell separation in which the length of all communication rules is 1. The following assertions are equivalent:

- (1) There exists an accepting computation of Π .
- (2) There exists $(a_0, i_0) \in \bigcup_{j=1}^q \mathcal{M}_j^*$ and a path in the dependency graph associated with Π , from (a_0, i_0) to $(\text{yes}, 0)$.

Proof. (1) \Rightarrow (2). First, we show that for each accepting computation \mathcal{C} of every P system Π there exists $(a_0, i_0) \in \bigcup_{j=1}^q \mathcal{M}_j^*$ and a path $\gamma_{\mathcal{C}}$ in the dependency graph associated with Π from (a_0, i_0) to $(\text{yes}, 0)$, by induction on the length n of \mathcal{C} .

If $n = 1$, a single step is performed in \mathcal{C} from C_0 to C_1 . A rule of the form $(j, \text{yes}/\lambda, 0)$, with $\text{yes} \in \Gamma, j \neq 0$, has been applied in that step. Then, $(\text{yes}, j) \in \mathcal{M}_j^*$, for some $j \in \{1, \dots, q\}$. Hence, $((\text{yes}, j), (\text{yes}, 0))$ is a path in the dependency graph associated with Π .

Let us suppose that the result holds for n . Let $\mathcal{C} = (C_0, C_1, \dots, C_n, C_{n+1})$ be an accepting computation of Π . Then $\mathcal{C}' = (C_1, \dots, C_n, C_{n+1})$ is an accepting computation of the system $\Pi' = (\Gamma, O_1, O_2, \Sigma, \mathcal{M}'_1, \dots, \mathcal{M}'_q, \mathcal{E}, \mathcal{R}, i_{\text{in}}, i_0)$, where \mathcal{M}'_j is the content of cell j in configuration C_1 , for $1 \leq j \leq q$. By the induction hypothesis there exists an object a_0 in a cell j_0 from C_1 , and a path in the dependency graph associated with Π' from (a_0, j_0) to $(\text{yes}, 0)$ (actually, by the [Definition 2](#), we have $a_0 = \text{yes}$). If (a_0, j_0) is an element of configuration C_0 (that means that in the first step a separation rule has been applied to cell j_0), then the result holds. Otherwise, there is an element (a_0, k_0) with

$k_0 \neq j_0$ in C_0 producing (a_0, j_0) by a communication rule. So, there exists a path γ_e in the dependency graph associated with Π from (a_0, k_0) to $(\text{yes}, 0)$.

(2) \Rightarrow (1). Let us prove that for each $(a_0, j_0) \in \bigcup_{j=1}^q \mathcal{M}_j^*$ and for each path in the dependency graph associated with every P system Π from (a_0, j_0) to $(\text{yes}, 0)$, there exists an accepting computation of Π . By induction on the length n of the path.

If $n = 1$, there is a path $((a_0, j_0), (\text{yes}, 0))$. Then, $a_0 = \text{yes}$ and the computation $\mathcal{C} = (C_0, C_1)$, where the rule $(j_0, \text{yes}/\lambda, 0)$ belongs to a multiset of rules m_0 that produces configuration C_1 from C_0 , is an accepting computation of Π .

Let us suppose that the result holds for n . Let

$$((a_0, j_0), (a_1, j_1), \dots, (a_n, j_n), (\text{yes}, 0))$$

be a path in the dependency graph of length $n + 1$. Let C_1 be the configuration of Π reached from C_0 by the application of a multiset of rules containing the rule that produces (a_1, j_1) from (a_0, j_0) . Then $((a_1, j_1), \dots, (a_n, j_n), (\text{yes}, 0))$ is a path of length n in the dependency graph associated with the system

$$\Pi' = (\Gamma, O_1, O_2, \Sigma, \mathcal{M}'_1, \dots, \mathcal{M}'_q, \mathcal{E}, \mathcal{R}, i_{\text{in}}, i_0)$$

where \mathcal{M}'_j is the content of cell j in configuration C_1 , for $1 \leq j \leq q$. By the induction hypothesis, there exists an accepting computation $\mathcal{C}' = (C_1, \dots, C_t)$ of Π' . Hence, $\mathcal{C} = (C_0, C_1, \dots, C_t)$ is an accepting computation of Π . \square

Next, given a family $\Pi = (\Pi(n))_{n \in \mathbb{N}}$ of recognizer tissue P systems with cell separation in which the length of all communication rules is 1, solving a decision problem, we will characterize the acceptance of an instance of the problem, w , using the set $\Delta_{\Pi(s(w))}$ associated with the system $\Pi(s(w))$, that processes the given instance w . More precisely, the instance is accepted by the system if and only if there is an object in the initial configuration of the system $\Pi(s(w))$ with input $\text{cod}(w)$ such that there exists a path in the associated dependency graph starting from that object and reaching the object yes in the environment.

Lemma 6. *Let $X = (I_X, \theta_X)$ be a decision problem. Let $\Pi = (\Pi(n))_{n \in \mathbb{N}}$ be a family of recognizer tissue P systems with cell separation, solving X according to [Definition 1](#), in which the length of all communication rules is 1. Let (cod, s) be the polynomial encoding associated with that solution. Then, for each instance w of the problem X the following assertions are equivalent:*

- (a) $\theta_X(w) = 1$ (that is, the answer to the problem is yes for w).
- (b) $\Delta_{\Pi(s(w))} \cap ((\text{cod}(w))^* \cup \bigcup_{j=1}^p \mathcal{M}_j^*) \neq \emptyset$, where $\mathcal{M}_1, \dots, \mathcal{M}_p$ are the initial multisets of the system $\Pi(s(w))$.

Proof. Let $w \in I_X$. Then $\theta_X(w) = 1$ if and only if there exists an accepting computation of the system $\Pi(s(w))$ with input multiset $\text{cod}(w)$. By [Lemma 5](#), this condition is equivalent to the following: in the initial configuration of $\Pi(s(w))$ with input multiset $\text{cod}(w)$ there exists at least one object $a \in \Gamma$ in a cell labelled with i such that in the dependency graph the node $(\text{yes}, 0)$ is reachable from (a, i) .

Hence, $\theta_X(w) = 1$ if and only if $\Delta_{\Pi(s(w))} \cap \mathcal{M}_j^* \neq \emptyset$ for some $j \in \{1, \dots, p\}$, or $\Delta_{\Pi(s(w))} \cap (\text{cod}(w))^* \neq \emptyset$. \square

Theorem 7. $\mathbf{P} = \mathbf{PMC}_{\text{TSC}(1)}$.

Proof. We have $\mathbf{P} \subseteq \mathbf{PMC}_{\text{TSC}(1)}$ because the class $\mathbf{PMC}_{\text{TSC}(1)}$ is closed under polynomial-time reduction and non-empty. In what follows, we show that $\mathbf{PMC}_{\text{TSC}(1)} \subseteq \mathbf{P}$. Let $X \in \mathbf{PMC}_{\text{TSC}(1)}$ and let $\Pi = (\Pi(n))_{n \in \mathbb{N}}$ be a family of recognizer tissue P systems with cell separation solving X , according to [Definition 1](#). Let (cod, s) be the polynomial encoding associated with that solution.

We consider the following deterministic algorithm:

Input: An instance w of X

```

- Construct the system  $\Pi(s(w))$  with input multiset  $\text{cod}(w)$ .
- Construct the dependency graph  $G_{\Pi(s(w))}$  associated with  $\Pi(s(w))$ .
- Construct the set  $\Delta_{\Pi(s(w))}$  as indicated in Lemma 4
  answer  $\leftarrow$  no;  $j \leftarrow 1$ 
  while  $j \leq p \wedge \text{answer} = \text{no}$  do
    if  $\Delta_{\Pi(s(w))} \cap \mathcal{M}_j^* \neq \emptyset$  then
      answer  $\leftarrow$  yes
       $j \leftarrow j + 1$ 
    endwhile
  if  $\Delta_{\Pi(s(w))} \cap (\text{cod}(w))^* \neq \emptyset$  then
    answer  $\leftarrow$  yes

```

On one hand, the answer of this algorithm is yes if and only if there exists a pair (a, i) belonging to $\Delta_{\Pi(s(w))}$ such that the symbol a appears in the cell labelled with i in the initial configuration (with input the multiset $\text{cod}(w)$).

On the other hand, a pair (a, i) belongs to $\Delta_{\Pi(s(w))}$ if and only if there exists a path from (a, i) to $(\text{yes}, 0)$, that is, if and only if we can obtain an accepting computation of $\Pi(s(w))$ with input $\text{cod}(w)$. Hence, the algorithm above described solves the problem X .

The cost for determining whether or not $\Delta_{\Pi(s(w))} \cap \mathcal{M}_j^* \neq \emptyset$ (or $\Delta_{\Pi(s(w))} \cap (\text{cod}(w))^* \neq \emptyset$) is of order $O(|\Gamma|^3 \cdot |H|^3)$.

Hence, the running time of this algorithm can be bounded by $f(|w|) + O(q^3 \cdot |\Gamma|^3)$, where f is the (total) cost of a polynomial encoding from X to Π , and q is the number of (initial) cells of $\Pi(s(w))$. Furthermore, by Definition 1 we have that all parameters involved are polynomial in $|w|$. That is, the algorithm is polynomial in the size $|w|$ of the input. \square

Corollary 8. *The following two assertions are equivalent:*

- (a) $\mathbf{P} = \mathbf{NP}$.
- (b) *There is an NP-complete problem that can be solved by a family of systems from TSC(1) in polynomial time, according to Definition 1.*

6. Solving computationally hard problems by using TSC(6)

In this section, we consider the efficiency of tissue P systems with cell separation and communication rules of length of at most 6. Such systems can efficiently solve computationally hard problems. In the following, we show how to efficiently solve SAT via such systems.

The SAT is defined as the following: given a Boolean formula in conjunctive normal form (CNF), determine whether or not there exists an assignment to its variables such that the formula is evaluated to be true. This is a well known NP-complete problem [5].

The solution proposed follows a brute force approach in the framework of recognizer P systems with cell separation. The solution consists of the following stages:

- *Generation stage:* All truth-assignments for the n variables are produced by using cell separation.
- *Checking stage:* We determine whether there is a truth-assignment that makes the Boolean formula evaluate to true.
- *Output stage:* The system sends to the environment the right answer according to the results of the previous stage.

Let us consider the polynomial-time computable function $\langle n, m \rangle = ((m+n)(m+n+1)/2) + m$ (the pair function), which is a primitive recursive and bijective function from \mathbb{N}^2 to \mathbb{N} . We shall construct a family $\Pi = \{\Pi(t) \mid t \in \mathbb{N}\}$ such that each system $\Pi(t)$ will solve all instances of SAT with n variables and m clauses, where $t = \langle n, m \rangle$, provided that the appropriate input multisets are given.

For each $n, m \in \mathbb{N}$,

$$\Pi(\langle n, m \rangle) = (\Gamma(\langle n, m \rangle), O_1, O_2, \Sigma(\langle n, m \rangle), w_1, w_2, \mathcal{R}(\langle n, m \rangle), \mathcal{E}(\langle n, m \rangle), i_{\text{in}}, i_0),$$

with the following components:

- $\Gamma(\langle n, m \rangle) = O_1 \cup O_2,$

$$\begin{aligned} O_1 = & \{x_{i,j}, \bar{x}_{i,j}, c_{i,j}, z_{i,j} \mid 1 \leq i \leq n, 1 \leq j \leq m\} \\ & \cup \{A_i \mid 1 \leq i \leq n\} \cup \{a_{1,i}, b_{1,i}, g_i, h_i \mid 1 \leq i \leq n-1\} \\ & \cup \{d_{1,i}, e_i, l_i \mid 1 \leq i \leq n-2\} \cup \{a_{2,i}, b_{2,i}, d_{2,i} \mid 2 \leq i \leq n-1\} \\ & \cup \{a_{i,j,k}, b_{i,j,k}, d_{i,j,k} \mid 1 \leq i \leq n, 1 \leq j \leq m, 1 \leq k \leq n-1\} \\ & \cup \{B_i \mid 1 \leq i \leq 4n\} \cup \{C_i \mid 1 \leq i \leq 3n\} \\ & \cup \{D_i \mid 1 \leq i \leq 4n+2m\} \cup \{E_i \mid 1 \leq i \leq 4n+2m+3\} \\ & \cup \{r_j \mid 1 \leq j \leq m\} \cup \{T_i, F_i, t_i, f_i \mid 1 \leq i \leq n\} \cup \{c, p, s, y, z, \text{yes}, \text{no}\}, \end{aligned}$$

$$\begin{aligned} O_2 = & \{x'_{i,j}, \bar{x}'_{i,j}, z'_{i,j} \mid 1 \leq i \leq n, 1 \leq j \leq m\} \\ & \cup \{T'_i, F'_i \mid 1 \leq i \leq n\} \cup \{y', z'\}. \end{aligned}$$

- $\Sigma(\langle n, m \rangle) = \{c_{i,j}, x_{i,j}, \bar{x}_{i,j} \mid 1 \leq i \leq n, 1 \leq j \leq m\}.$
- $w_1 = \{\{a_{1,1}, a_{2,2}, g_1, B_1, C_1, D_1, E_1, p, \text{yes}, \text{no}\}\} \cup \{\{a_{i,j,1} \mid 1 \leq i \leq n, 1 \leq j \leq m\}\}.$
- $w_2 = A_1.$
- $\mathcal{E}(\langle n, m \rangle) = \Gamma(\langle n, m \rangle) - \{\text{yes}, \text{no}\}.$
- $i_{\text{in}} = 2$ is the input cell.
- $i_0 = 0$ is the output region.
- $\mathcal{R}(\langle n, m \rangle)$ is the following set of rules:

(1) *Separation rule:*

$$r_1 \equiv [s]_2 \rightarrow [O_1]_2 [O_2]_2.$$

(With the presence of object s , cells with label 2 are separated. As we will see below, at step $3k-1$ ($1 \leq k \leq n$), cells with label 2 contain object s , and are separated. In this way, in $3n-1$ steps, we get 2^n cells with label 2.)

(2) *Communication rules:*

$$r_{2,i,j,k} \equiv (1, a_{i,j,k}/b_{i,j,k}, 0) \text{ for } 1 \leq i \leq n, 1 \leq j \leq m, 1 \leq k \leq n-1;$$

$$r_{3,i,j,k} \equiv (1, b_{i,j,k}/c_{i,j,k}^2 d_{i,j,k}^2, 0) \text{ for } 1 \leq i \leq n, 1 \leq j \leq m, 1 \leq k \leq n-2;$$

$$r_{4,i,j} \equiv (1, b_{i,j,n-1}/c_{i,j}^2, 0) \text{ for } 1 \leq i \leq n, 1 \leq j \leq m;$$

$$r_{5,i,j,k} \equiv (1, d_{i,j,k}/a_{i,j,k+1}, 0) \text{ for } 1 \leq i \leq n, 1 \leq j \leq m, 1 \leq k \leq n-2$$

(at each step $3k+2$ ($0 \leq k \leq n-2$) the cell with label 1 gets 2^{k+1} copies of object $c_{i,j}$, which is used to duplicate $x_{i,j}, \bar{x}_{i,j}, x'_{i,j}$ and $\bar{x}'_{i,j}$ (see the following rules $r_{19,i,j}-r_{22,i,j}$));

$$r_{6,i} \equiv (1, a_{1,i}/b_{1,i}, 0) \text{ for } 1 \leq i \leq n-1;$$

$$r_{7,i} \equiv (1, b_{1,i}/c^2 d_{1,i}^2 e_i^2, 0) \text{ for } 1 \leq i \leq n-2;$$

$$r_8 \equiv (1, b_{1,n-1}/c^2, 0);$$

$$r_{9,i} \equiv (1, d_{1,i}/a_{1,i+1}, 0) \text{ for } 1 \leq i \leq n-2;$$

$$r_{10,i} \equiv (1, e_i/a_{2,i+1}, 0) \text{ for } 1 \leq i \leq n-2;$$

$$r_{11,i} \equiv (1, a_{2,i}/b_{2,i}, 0) \text{ for } 2 \leq i \leq n-1;$$

$$r_{12,i} \equiv (1, b_{2,i}/c^2 d_{2,i}^2, 0) \text{ for } 2 \leq i \leq n-2;$$

$$r_{13} \equiv (1, b_{2,n-1}/c^2, 0);$$

$$r_{14,i} \equiv (1, d_{2,i}/a_{2,i+1}, 0) \text{ for } 2 \leq i \leq n-2$$

(at each step $3k+2$ ($0 \leq k \leq n-2$), the cell with label 1 gets $2^{k+1}(k+1)$ copies of object c , which is used to duplicate the objects T_i, T'_i, F_i, F'_i (see the rules $r_{25,i}-r_{28,i}$));

$$r_{15,i} \equiv (1, g_i/h_i, 0) \text{ for } 1 \leq i \leq n-1;$$

$$r_{16,i} \equiv (1, h_i/l_i^2 A_{i+1}^2, 0) \text{ for } 1 \leq i \leq n-2;$$

$$r_{17} \equiv (1, h_{n-1}/A_n^2, 0);$$

$$r_{18,i} \equiv (1, l_i/g_{i+1}, 0) \text{ for } 1 \leq i \leq n-2$$

(at each step $3k+2$ ($0 \leq k \leq n-2$), the cell with label 1 gets 2^{k+1} copies of object A_{k+2} , which encodes the i -th variable in the propositional formula);

$$r_{19,i,j} \equiv (2, c_{i,j} x_{i,j} / z_{i,j} z'_{i,j} x_{i,j} x'_{i,j}, 0) \text{ for } 1 \leq i \leq n, 1 \leq j \leq m;$$

$$r_{20,i,j} \equiv (2, c_{i,j}\bar{x}_{i,j}/z_{i,j}z'_{i,j}\bar{x}_{i,j}\bar{x}'_{i,j}, 0) \text{ for } 1 \leq i \leq n, 1 \leq j \leq m;$$

$$r_{21,i,j} \equiv (2, c_{i,j}x'_{i,j}/z_{i,j}z'_{i,j}x_{i,j}x'_{i,j}, 0) \text{ for } 1 \leq i \leq n, 1 \leq j \leq m;$$

$$r_{22,i,j} \equiv (2, c_{i,j}\bar{x}'_{i,j}/z_{i,j}z'_{i,j}\bar{x}_{i,j}\bar{x}'_{i,j}, 0) \text{ for } 1 \leq i \leq n, 1 \leq j \leq m$$

(with the presence of $c_{i,j}$ in cells with label 2, objects $x_{i,j}$, $\bar{x}_{i,j}$, $x'_{i,j}$, $\bar{x}'_{i,j}$ are duplicated; at the same time, auxiliary objects $z_{i,j}$, $z'_{i,j}$ are brought into cells with label 2 from the environment);

$$r_{23,i} \equiv (2, A_i/T_iF'_izz'yy's, 0) \text{ for } 1 \leq i \leq n - 1;$$

$$r_{24} \equiv (2, A_n/T_nF'_nyy's, 0)$$

(object A_i introduces objects T_i , F'_i , z , z' , y , y' , s (in particular, object A_n introduces T_n , F'_n , y , y' , s));

$$r_{25,i} \equiv (2, cT_i/zz'T_iT'_i, 0) \text{ for } 1 \leq i \leq n - 1;$$

$$r_{26,i} \equiv (2, cT'_i/zz'T_iT'_i, 0) \text{ for } 1 \leq i \leq n - 1;$$

$$r_{27,i} \equiv (2, cF_i/zz'F_iF'_i, 0) \text{ for } 1 \leq i \leq n - 1;$$

$$r_{28,i} \equiv (2, cF'_i/zz'F_iF'_i, 0) \text{ for } 1 \leq i \leq n - 1$$

(with the presence of object c , objects T_i , T'_i , F_i , F'_i are duplicated and objects z , z' are brought into cells with label 2);

$$r_{29,i,j} \equiv (1, c_{i,j}/z_{i,j}, 2) \text{ for } 1 \leq i \leq n, 1 \leq j \leq m;$$

$$r_{30,i,j} \equiv (1, c_{i,j}/z'_{i,j}, 2) \text{ for } 1 \leq i \leq n, 1 \leq j \leq m;$$

$$r_{31} \equiv (1, c/z, 2);$$

$$r_{32} \equiv (1, c/z', 2);$$

$$r_{33,i} \equiv (1, A_i/y, 2) \text{ for } 2 \leq i \leq n;$$

$$r_{34,i} \equiv (1, A_i/y', 2) \text{ for } 2 \leq i \leq n$$

(the object $c_{i,j}$ (resp. c , A_i) in cell 1 is exchanged with objects $z_{i,j}$ and $z'_{i,j}$ (resp. z and z' , y and y') in cells with label 2);

$$r_{35,i} \equiv (1, B_i/B_{i+1}, 0) \text{ for } 1 \leq i \leq 2n - 1;$$

$$r_{36,i} \equiv (1, B_i/B_{i+1}^2, 0) \text{ for } 2n \leq i \leq 3n - 1;$$

$$r_{37,i} \equiv (1, C_i/C_{i+1}, 0) \text{ for } 1 \leq i \leq 2n - 1;$$

$$r_{38,i} \equiv (1, C_i/C_{i+1}^2, 0) \text{ for } 2n \leq i \leq 3n - 1;$$

$$r_{39,i} \equiv (1, D_i/D_{i+1}, 0) \text{ for } 1 \leq i \leq 2n - 1;$$

$$r_{40,i} \equiv (1, D_i/D_{i+1}^2, 0) \text{ for } 2n \leq i \leq 3n - 1$$

(the counters B_i , C_i , D_i in the cell with label 1 grow their subscripts at each step; from step $2n$ to step $3n - 1$, the number of copies of objects is doubled, and hence after $3n - 1$ steps, the cell with label 1 contains 2^n copies of B_{3n} , C_{3n} , and D_{3n} ; object B_i will check which clauses are satisfied by a given truth-assignment; object C_i is used to introduce objects t_i and f_i ; object D_i is used to check whether there is at least one truth-assignment which satisfies all clauses);

$$r_{41,i} \equiv (1, E_i/E_{i+1}, 0) \text{ for } 1 \leq i \leq 4n + 2m + 2$$

(the counter E_i in the cell with label 1 grows its subscript, and it will be used to bring the object no to the environment in the end of the computation, if the formula is not satisfiable);

$$r_{42,i,j} \equiv (1, z_{i,j}/\lambda, 0);$$

$$r_{43,i,j} \equiv (1, z'_{i,j}/\lambda, 0);$$

$$r_{44} \equiv (1, y/\lambda, 0);$$

$$r_{45} \equiv (1, y'/\lambda, 0);$$

$$r_{46} \equiv (1, z/\lambda, 0);$$

$$r_{47} \equiv (1, z'/\lambda, 0)$$

(the objects $z_{i,j}$, $z'_{i,j}$, y , y' , z , and z' in the cell with label 1 are removed);

$$r_{48} \equiv (1, B_{3n}C_{3n}D_{3n}/y, 2);$$

$$r_{49} \equiv (1, B_{3n}C_{3n}D_{3n}/y', 2)$$

(objects B_{3n} , C_{3n} , D_{3n} are moved to cells with label 2 exchanging with objects y and y');

$$r_{50,i} \equiv (2, C_{3n}T_i/C_{3n}t_i, 0) \text{ for } 1 \leq i \leq n;$$

$$r_{51,i} \equiv (2, C_{3n}T'_i/C_{3n}t_i, 0) \text{ for } 1 \leq i \leq n;$$

$$r_{52,i} \equiv (2, C_{3n}F_i/C_{3n}f_i, 0) \text{ for } 1 \leq i \leq n;$$

$$r_{53,i} \equiv (2, C_{3n}F'_i/C_{3n}f_i, 0) \text{ for } 1 \leq i \leq n$$

(with the presence of object C_{3n} in cells with label 2, objects T_i and T'_i introduce t_i , and objects F_i and F'_i introduce f_i ; objects t_i and f_i correspond to the value *true* and *false* of variable x_i);

$r_{54,i} \equiv (2, B_i/B_{i+1}^2, 0)$ for $3n \leq i \leq 4n - 1$
 (from step $3n$ to step $4n - 1$, the number of object B_i is doubled and its subscript is increased by 1 at each step);
 $r_{55,i} \equiv (2, D_i/D_{i+1}, 0)$ for $3n \leq i \leq 4n - 1$
 (from step $3n$ to step $4n - 1$, the subscript of object D_i is increased by 1 at each step);
 $r_{56,i,j} \equiv (2, B_{4n}t_i x_{i,j}/B_{4n}t_j r_j, 0)$ for $1 \leq i \leq n, 1 \leq j \leq m$;
 $r_{57,i,j} \equiv (2, B_{4n}t_i \bar{x}_{i,j}/B_{4n}t_j, 0)$ for $1 \leq i \leq n, 1 \leq j \leq m$;
 $r_{58,i,j} \equiv (2, B_{4n}t_i x'_{i,j}/B_{4n}t_j r_j, 0)$ for $1 \leq i \leq n, 1 \leq j \leq m$;
 $r_{59,i,j} \equiv (2, B_{4n}t_i \bar{x}'_{i,j}/B_{4n}t_j, 0)$ for $1 \leq i \leq n, 1 \leq j \leq m$;
 $r_{60,i,j} \equiv (2, B_{4n}f_i \bar{x}_{i,j}/B_{4n}f_j r_j, 0)$ for $1 \leq i \leq n, 1 \leq j \leq m$;
 $r_{61,i,j} \equiv (2, B_{4n}f_i x_{i,j}/B_{4n}f_j, 0)$ for $1 \leq i \leq n, 1 \leq j \leq m$;
 $r_{62,i,j} \equiv (2, B_{4n}f_i \bar{x}_{i,j}/B_{4n}f_j r_j, 0)$ for $1 \leq i \leq n, 1 \leq j \leq m$;
 $r_{63,i,j} \equiv (2, B_{4n}f_i x'_{i,j}/B_{4n}f_j, 0)$ for $1 \leq i \leq n, 1 \leq j \leq m$
 (these rules are used to check which clauses are satisfied by a given truth-assignment; if the object r_j is introduced into a cell with label 2, then it means that the corresponding truth-assignment satisfies clause M_j);
 $r_{64,i} \equiv (2, D_i/D_{i+1}, 0)$ for $4n \leq i \leq 4n + m - 1$
 (from step $4n$ to step $4n + m - 1$, the subscript of object D_i is increased by 1 at each step);
 $r_{65,i} \equiv (2, D_{4n+m+i} r_{i+1}/D_{4n+m+i+1}, 0)$ for $0 \leq i \leq m - 1$
 (in each cell with label 2, we check whether or not all clauses are satisfied by the corresponding truth-assignment; For each clause which is satisfied, the subscript of object D_i is increased by 1; hence, the subscript reaches the value $4n + 2m$ if and only if all clauses are satisfied);
 $r_{66} \equiv (2, D_{4n+2m}/\lambda, 1)$
 (if one of the truth-assignments from a cell with label 2 satisfies all clauses, then object D_{4n+2m} appears in this cell, which is moved to the cell with label 1);
 $r_{67} \equiv (1, D_{4n+2m} p \text{ yes}/\lambda, 0)$
 (the object *yes* leaves the system, signaling the fact that the formula is satisfiable; in the cell with label 1, the counter E_i will increase its subscript for one more step, but after that it will remain unchanged—it can leave the cell with label 1 only in the presence of p , but this object was already moved to the environment);
 $r_{68} \equiv (1, E_{4n+2m+3} p \text{ no}/\lambda, 0)$
 (if the counter E_i reaches the subscript $4n + 2m + 3$ and the object p is still in the cell with label 1, then the object *no* can exist the system, signaling that the formula is not satisfiable).

6.1. An overview of the computation

A family of recognizer tissue P systems with cell separation is constructed above. For an instance of SAT $\varphi = M_1 \wedge \dots \wedge M_m$, consisting of m clauses $M_i = y_{i,1} \vee \dots \vee y_{i,l_i}$, $1 \leq i \leq m$, where $\text{Var}(\varphi) = \{x_1, \dots, x_n\}$, $y_{i,k} \in \{x_j, \neg x_j \mid 1 \leq j \leq n\}$, $1 \leq i \leq m$, $1 \leq k \leq l_i$, the size mapping on the set of instances is defined as $s(\varphi) = \langle n, m \rangle = ((m + n)(m + n + 1)/2) + m$, and the encoding of the instance is the multiset $\text{cod}(\varphi) = \{\{c_{i,j} x_{i,j} \mid x_i \in \{y_{j,k} \mid 1 \leq k \leq l_i\}, 1 \leq i \leq n, 1 \leq j \leq m\}\} \cup \{\{c_{i,j} \bar{x}_{i,j} \mid \neg x_i \in \{y_{j,k} \mid 1 \leq k \leq l_i\}, 1 \leq i \leq n, 1 \leq j \leq m\}\}$.

Now, we informally describe how system $\Pi(s(\varphi))$ with input $\text{cod}(\varphi)$ works.

Let us start with the *generation stage*. This stage has several parallel processes, which we describe in several items.

- In the initial configuration of the system, the cell with label 2 contains an object A_1 (A_i encodes the i -th variable in the propositional formula). The objects T_1, F'_1, z, z', y, y' and s are brought in the cell with label 2, in exchange for A_1 , by the rule $r_{23,i}$. The objects T_1 and F'_1 correspond to the values *true* and *false* which the variable x_1 may assume (in general, T_i (or T'_i) and F_i (or F'_i) are for the variable x_i), and in the next step they are separated into the new daughter cells with label 2 by the separation rule, because $T_1 \in O_1$ and $F'_1 \in O_2$. The object s is used to activate the separation rule r_1 , and is consumed during the application of the separation rule. The objects y and y' are used to introduce A_2 from the cell with label 1, and the process of truth-assignment for variable x_2 can continue. In this way, in $3n - 1$ steps, 2^n cells with label 2 are generated, and each one contains one of the 2^n possible truth-assignments for the n variables.

- In cells with label 2, by rules $r_{19,i,j}-r_{22,i,j}$, objects $c_{i,j}x_{i,j}$, $c_{i,j}\bar{x}_{i,j}$, $c_{i,j}x'_{i,j}$, $c_{i,j}\bar{x}'_{i,j}$ introduce objects $z_{i,j}z'_{i,j}x_{i,j}x'_{i,j}$, $z_{i,j}z'_{i,j}\bar{x}_{i,j}\bar{x}'_{i,j}$, $z_{i,j}z'_{i,j}x_{i,j}x'_{i,j}$, $z_{i,j}z'_{i,j}\bar{x}_{i,j}\bar{x}'_{i,j}$, respectively. In the next step, the objects with primes and the objects without primes are separated into the new daughter cells with label 2. The idea is that $c_{i,j}$ is used to duplicate $x_{i,j}$ and $\bar{x}_{i,j}$ (in the sense ignoring the prime), so that one copy of each of them will appear in each cell with label 2. The objects $z_{i,j}$ and $z'_{i,j}$ in cells with label 2 are exchanged with the objects $c_{i,j}$ in the cell with label 1 by the rules $r_{29,i,j}$ and $r_{30,i,j}$. In this way, the cycle of duplication and separation can be iterated.
- In parallel with the above duplication-separation process, the objects c are used to duplicate the objects T_i , T'_i , F_i , and F'_i by the rules $r_{25,i}-r_{28,i}$; the rules r_{31} and r_{32} take care of introducing the object c from the cell with label 1 to cells with label 2.
- In parallel with the operations in the cells with label 2, the objects $a_{i,j,k+1}$ from the cell with label 1 are exchanged with objects $b_{i,j,k+1}$ from the environment at the step $3k + 1$ ($0 \leq k \leq n - 3$) by the rule $r_{2,i,j,k}$. In the next step, each object $b_{i,j,k+1}$ is exchanged with two copies of objects $c_{i,j}$ and $d_{i,j,k+1}$ by the rule $r_{3,i,j,k}$. At step $3k + 3$ ($0 \leq k \leq n - 3$), the object $d_{i,j,k}$ is exchanged with object $a_{i,j,k+2}$ by the rule $r_{4,i,j,k}$. In particular, at step $3n - 5$, $a_{i,j,n-1}$ is exchanged with $b_{i,j,n-1}$ by $r_{2,i,j,k}$; at step $3n - 4$, each copy of object $b_{i,j,n-1}$ is exchanged with two copies of $c_{i,j}$ by $r_{4,i,j}$. After step $3n - 4$, there is no object $a_{i,j,k}$ that appears in the cell with label 1, and the group of rules $r_{2,i,j,k}-r_{5,i,j,k}$ will not be used again. Note that the subscript k of the object $a_{i,j,k}$ grows by 1 in every three steps until the value $n - 1$ is reached, and the number of copies of $a_{i,j,k}$ is doubled in every three steps. At step $3k + 3$ ($0 \leq k \leq n - 2$), the cell with label 1 has 2^{k+1} copies of object $c_{i,j}$. At the same time, there are 2^{k+1} cells with label 2, and each cell with label 2 contains one copy of object $z_{i,j}$ or one copy of object $z'_{i,j}$. Due to the maximality of the parallelism of using the rules, each cell with label 2 gets exactly one copy of $c_{i,j}$ from the cell with label 1 by the rules $r_{29,i,j}$ and $r_{30,i,j}$. The object $c_{i,j}$ in cell with label 2 is used for duplication as described above.
- The objects $a_{1,i}$ and $a_{2,i}$ in the cell with label 1 have a similar role to object $a_{i,j,k}$ in cell 1, which introduces appropriate copies of object c for the duplication of objects T_i , T'_i , F_i , and F'_i by the rules $r_{6,i}-r_{14,i}$. Note that at step $3k + 3$ ($0 \leq k \leq n - 2$), there are $2^{k+1}(k + 1)$ copies of object c , which ensure that each cell with label 2 gets $k + 1$ copies of object c by the maximality of the parallelism of using the rules.
- The object g_{i+1} in the cell with label 1 is exchanged with h_{i+1} from the environment at step $3i + 1$ ($0 \leq i \leq n - 3$) by the rule $r_{15,i}$. In the next step, the object h_{i+1} is exchanged with two copies of objects l_{i+1} and A_{i+2} by the rule $r_{13,i}$. At the step $3i + 3$ ($0 \leq i \leq n - 3$), the object l_{i+1} is exchanged with two copies of g_{i+2} , so the process can be iterated, until the subscript i of g_i reaches $n - 1$. In particular, at step $3n - 5$, object g_{n-1} is exchanged with h_{n-1} by the rule $r_{15,i}$, and at step $3n - 4$, each object h_{n-1} is exchanged with two copies of A_n . After step $3n - 4$, there is no object g_i that appears in the cell with label 1, and the group of rules $r_{15,i}-r_{18,i}$ will not be used again. At the step $3i + 3$ ($0 \leq i \leq n - 2$), the cell with label 1 contains 2^{i+1} copies of A_{i+2} , and there are 2^{i+1} cells with label 2, each of them containing one copy of object y or one copy of object y' . Due to the maximality of the parallelism of using the rules, each cell with label 2 gets exactly one copy of A_{i+2} from the cell 1 by the rules $r_{33,i}$ and $r_{34,i}$. In this way, the truth-assignment for the variable x_{i+1} can continue.
- The counters B_i , C_i , D_i , and E_i in the cell with label 1 grow their subscripts by the rules $r_{35,i}-r_{41,i}$. From step $2n$ to step $3n - 1$, the number of copies of objects of the first three types is doubled; hence after $3n - 1$ steps, the cell with label 1 contains 2^n copies of B_{3n} , C_{3n} , and D_{3n} . Objects B_i will check which clauses are satisfied by a given truth-assignment, objects C_i are used to multiply the number of copies of t_i , f_i as we will see immediately, objects D_i are used to check whether there is at least one truth-assignment which satisfies all clauses, and E_i will be used to bring the object no to the environment, if this is the case, at the end of the computation.
- The objects $z_{i,j}$, $z'_{i,j}$, y , y' , z , and z' in the cell with label 1 are removed by the rules $r_{42,i,j}-r_{47}$. (Actually, if the objects $z_{i,j}$, $z'_{i,j}$, y , y' , z , and z' stay in the cell with label 1, they do not influence the work of the system. The rules $r_{38}-r_{43}$ are designed just in order to simplify the formal verification.)

In this way, after the $(3n - 1)$ -th step the generation stage finishes and the *checking stage* starts. At this moment, the cell with label 1 contains 2^n copies of objects B_{3n} , C_{3n} , and D_{3n} , and there are 2^n

cells with label 2, each of them containing a copy of y and $n - 1$ copies of z , or a copy of y' and $n - 1$ copies of z' . The objects z and z' in cells with label 2 will not evolve any further, because the cell with label 1 contains no object c from now on, and the rules r_{31} and r_{32} cannot be applied.

At the step $3n$, objects y or y' are exchanged with objects B_{3n} , C_{3n} , and D_{3n} by rules r_{48} and r_{49} . (Note that the rules $r_{33,i}$ and $r_{34,i}$ cannot be used, because there is no object A_i in the cell with label 1 at this moment and henceforth. And the cells with label 2 cannot separate any further.) Due to the maximality of the parallelism of using the rules, each cell with label 2 gets exactly one copy of each of B_{3n} , C_{3n} , and D_{3n} .

In the presence of C_{3n} , the objects T_i and T'_i , F_i and F'_i introduce the objects t_i and f_i , respectively. Because there is only one copy of C_{3n} available, for each one of t_i and f_i we need one step. So this phase needs n steps; that is, this phase ends at step $4n$.

In parallel with the previous operations, the counters B_i and D_i increase their subscripts, until the value $4n$ is reached, by the rules $r_{54,i}$ and $r_{55,i}$. Each cell with label 2 contains one copy of D_{4n} and 2^n copies of B_{4n} . Simultaneously, E_i increases its subscript in the cell with label 1.

At step $4n + 1$, with the presence of B_{4n} , we start to check the values assumed by clauses for the truth-assignments from each cell with label 2 by the rules $r_{56,i,j}$ – $r_{63,i,j}$. Each membrane with label 2 contains nm objects $x_{i,j}$ and $\bar{x}_{i,j}$ or nm objects $x'_{i,j}$ and $\bar{x}'_{i,j}$, because each clause contains at most n literals, and there are m clauses. Note that each membrane with label 2 contains 2^n copies of B_{4n} and n objects t_i and f_i . At each step, n objects $x_{i,j}$ and $\bar{x}_{i,j}$, or n objects $x'_{i,j}$ and $\bar{x}'_{i,j}$ are checked. So it takes m steps. In parallel, D_i increases the subscript, until the value $4n + m$ is reached (at step $4n + m$), by the rule $r_{64,i}$.

By the rule $r_{65,i}$, in each cell with label 2, we check whether or not all clauses are satisfied by the corresponding truth-assignment. For each clause which is satisfied, the subscript of D_i is increased by 1; hence the subscript reaches the value $4n + 2m$ if and only if all clauses are satisfied.

The output stage starts at the $(4n + 2m + 1)$ -th step.

- *Affirmative answer*: If one of the truth-assignments from a cell with label 2 has satisfied all clauses, then in that cell there is an object D_{4n+2m} as described above, which is sent to the cell with label 1 by the rule r_{66} . In the next step, the object *yes* leaves the system by the rule r_{67} , signaling the fact that the formula is satisfiable. In cell 1, the counter E_i increases its subscript by the rule $r_{41,i}$, until the value $4n + 2m + 3$ is reached, but after that it will remain unchanged—it can leave the cell with label 1 only in the presence of p , but this object p was already moved to the environment at step $4n + 2m + 2$. The computation halts at step $4n + 2m + 2$.
- *Negative answer*: If the counter E_i reaches the subscript $4n + 2m + 3$ and the object p is still in the cell with label 1, then the object *no* can be moved to the environment by the rule r_{68} , signaling that the formula is not satisfiable. The computation finishes at step $4n + 2m + 3$.

6.2. Formal verification

In this subsection, we prove that the family built previously solves SAT in polynomial time, according to Definition 1. First of all, the Definition 1 requires that the defined family is *consistent*, in the sense that all systems of the family must be recognizer tissue P systems with cell separation. By the construction (rule type and working alphabet) it is clear that it is a family of tissue P systems with cell separation. In order to show that all members in Π are recognizer systems it suffices to check that all the computations halt (this will be deduced from the polynomial bound), and that either an object *yes* or an object *no* is sent out exactly in the last step of the computation (this will be deduced from the soundness and completeness).

6.2.1. Polynomial uniformity of the family

We will show that the family $\Pi = \{\Pi(\langle n, m \rangle) \mid n, m \in \mathbb{N}\}$ defined above is polynomially uniform via Turing machines. To this aim, it will be proved that $\Pi(\langle n, m \rangle)$ is built in polynomial time with respect to the size parameter n and m of instances of SAT.

It is easy to check that the rules of a system $\Pi(\langle n, m \rangle)$ of the family are defined recursively from the values n and m . And the necessary resources for building an element of the family are of a polynomial

order, as shown below:

- Size of the alphabet: $3n^2m + 4nm + 30n + 5m - 5 \in O(n^2m)$.
- Initial number of cells: $2 \in O(1)$.
- Initial number of objects: $nm + 10 \in O(nm)$.
- Number of rules: $3n^2m + 15nm + 36n + 3m - 12 \in O(n^2m)$.
- Maximum length of a rule: $6 \in O(1)$.

Therefore, a deterministic Turing machine can build $\Pi((n, m))$ in polynomial time with respect to n and m .

6.2.2. The polynomial bound of the family

For an instance of SAT $\varphi = M_1 \wedge \dots \wedge M_m$, consisting of m clauses $M_i = y_{i,1} \vee \dots \vee y_{i,l_i}$, $1 \leq i \leq m$, where $\text{Var}(\varphi) = \{x_1, \dots, x_n\}$, $y_{i,k} \in \{x_j, \neg x_j \mid 1 \leq j \leq n\}$, $1 \leq i \leq m$, $1 \leq k \leq l_i$, we recall the size mapping function $s(\varphi)$ and the encoding function $\text{cod}(\varphi)$: $s(\varphi) = \langle n, m \rangle$, and $\text{cod}(\varphi) = \{\{c_{i,j}x_{i,j} \mid x_i \in \{y_{j,k} \mid 1 \leq k \leq l_j\}, 1 \leq i \leq n, 1 \leq j \leq m\}\} \cup \{\{c_{i,j}\bar{x}_{i,j} \mid \neg x_i \in \{y_{j,k} \mid 1 \leq k \leq l_j\}, 1 \leq i \leq n, 1 \leq j \leq m\}\}$. The pair (cod, s) can be computed in polynomial time; $\text{cod}(\varphi)$ is an input multiset of the system $\Pi(s(\varphi))$.

In order to prove that the system $\Pi(s(\varphi))$ with input $\text{cod}(\varphi)$ is polynomially bounded, it suffices to find the moment in which the computation halts, or at least, an upper bound for it.

Proposition 9. *The family $\Pi = \{\Pi((n, m)) \mid n, m \in \mathbb{N}\}$ is polynomially bounded with respect to (SAT, cod, s).*

Proof. We will informally go through the stages of the computation in order to estimate a bound for the number of steps. The computation will be checked in more detail when addressing the soundness and completeness proof.

Let $\varphi = M_1 \wedge \dots \wedge M_m$ be an instance of SAT. We shall study what happens during the computation of the system $\Pi(s(\varphi))$ with input $\text{cod}(\varphi)$ in order to find the halting step, or at least, an upper bound for it.

First, the generation stage has exactly $3n - 1$ steps, where at steps $3k + 2$ ($0 \leq k \leq n - 1$) the cells with label 2 are separated. In this way, we get 2^n cells with label 2; each of them contains one of the 2^n possible truth-assignments for the n variables.

After one more step, the objects B_{3n} , C_{3n} , and D_{3n} arrive at cells with label 2, and the checking stage starts. The object C_{3n} works for n steps introducing objects t_i or f_i into cells with label 2, until all objects T_i , T'_i , F_i and F'_i are consumed, at the step $4n$. From step $4n + 1$, the objects B_{4n} start to work, checking which clauses are satisfied by the truth-assignment from each cell with label 2. This checking takes m steps. When the subscript of D_i grows to $4n + m$ at step $4n + m$, the system starts to check whether or not all clauses are satisfied by the corresponding truth-assignment. It takes m steps, and the checking stage ends at step $4n + 2m$.

The last one is the answer stage. The longest case is obtained when the answer is negative. In this case there are two steps where only the counter E_i is working. At the step $4n + 2m + 3$ the object $E_{4n+2m+3}$ works together with object p bringing no from the cell with label 1 into the environment.

Therefore, there exists a linear bound (with respect to n and m) on the number of steps of the computation. \square

6.2.3. Soundness and completeness of the family

In order to prove the soundness and completeness of the family Π with respect to (SAT, cod, s), we shall prove that for a given instance φ of SAT, the system $\Pi(s(\varphi))$ with input $\text{cod}(\varphi)$ sends out an object yes if and only if the answer to the problem for the instance considered, φ , is affirmative and the object no is sent out otherwise. In both cases the answer will be sent to the environment in the last step of the computation.

For the sake of simplicity of the notation, we consider the following two functions: $\psi(\sigma_j(x_i))$ and $\gamma(\sigma_j(x_i))$. Let \mathcal{F} be the set of all assignments of the variables x_1, x_2, \dots, x_n . We order the set \mathcal{F} in

lexicographical order, that is, $\mathcal{F} = \{\sigma_1, \sigma_2, \dots, \sigma_{2^n}\}$, where $\sigma_j(x_i) \in \{0, 1\}$ ($1 \leq j \leq 2^n, 1 \leq i \leq n$) is an assignment of variables. For $1 \leq j \leq 2^n, 1 \leq i \leq n$, we define ψ as follows: if j is odd, then

$$\psi(\sigma_j(x_i)) = \begin{cases} T_i, & \text{if } \sigma_j(x_i) = 1, \\ F_i, & \text{if } \sigma_j(x_i) = 0; \end{cases}$$

if j is even, then

$$\psi(\sigma_j(x_i)) = \begin{cases} T'_i, & \text{if } \sigma_j(x_i) = 1, \\ F'_i, & \text{if } \sigma_j(x_i) = 0. \end{cases}$$

For each assignment $\sigma_j(x_i) \in \{0, 1\}$, and for $i = 1, \dots, n$, we define γ as follows:

$$\gamma(\sigma_j(x_i)) = \begin{cases} t_i, & \text{if } \sigma_j(x_i) = 1; \\ f_i, & \text{if } \sigma_j(x_i) = 0. \end{cases}$$

In this way, each assignment of variables σ_j is associated with a multiset $\{\{\psi(\sigma_j(x_1)), \psi(\sigma_j(x_2)), \dots, \psi(\sigma_j(x_n))\}\}$ and a multiset $\{\{\gamma(\sigma_j(x_1)), \gamma(\sigma_j(x_2)), \dots, \gamma(\sigma_j(x_n))\}\}$.

Given a computation \mathcal{C} we denote the configuration at the i -th step as \mathcal{C}_i . Moreover, $\mathcal{C}_i(1)$ will denote the multiset associated with cell 1 in such a configuration.

Lemma 10. *Let \mathcal{C} be an arbitrary computation of the system, then at step $3k + 2$ for all k such that $0 \leq k \leq n - 2$, the cell with label 1 gets 2^{k+1} copies of object $c_{i,j}$, $2^{k+1}(k + 1)$ copies of object c , and 2^{k+1} copies of object A_{k+2} from the environment. And after step $3n - 3$, the cell with label 1 cannot get objects $c_{i,j}$, c , A_i any longer.*

Proof. It is not difficult to find that in the set of all rules there are six rule types related to object $c_{i,j}$, that is, rules $r_{2,i,j,k} - r_{5,i,j,k}$, $r_{29,i,j}$ and $r_{30,i,j}$. The rules $r_{29,i,j}$ and $r_{30,i,j}$ are used to move $c_{i,j}$ from the cell with label 1 to cells with label 2 in exchange of $z_{i,j}$ or $z'_{i,j}$, which happens at steps $3k + 3$ for all k such that $0 \leq k \leq n - 2$. Anyway, these two rules do not bring object $c_{i,j}$ into the cell with label 1. So we need only to check how these four rule types $r_{2,i,j,k} - r_{5,i,j,k}$ work.

First, by induction on k , we prove that at step $3k + 2$ ($0 \leq k \leq n - 3$), the cell with label 1 gets 2^{k+1} copies of object $c_{i,j}$ and the cell with label 1 has exactly 2^{k+1} copies of $d_{i,j,k+1}$.

In the multiset $\mathcal{C}_0(1)$, there is one copy of each object $a_{i,j,1}$ ($1 \leq i \leq n, 1 \leq j \leq m$). By application of rules $r_{2,i,j,1}$ and $r_{3,i,j,1}$, the cell with label 1 gets two copies of $c_{i,j}$ and two copies of $d_{i,j,1}$ at step 2.

Now suppose the result is true for $k < n - 4$. We have, by the inductive hypothesis, that at step $3k + 2$, the cell with label 1 gets 2^{k+1} copies of object $c_{i,j}$ and the cell with label 1 has exactly 2^{k+1} copies of $d_{i,j,k+1}$. At step $3k + 3$, among these four rule types $r_{2,i,j,k+1} - r_{5,i,j,k}$, only rule $r_{5,i,j,k+1}$ can be applied; 2^{k+1} copies of $d_{i,j,k+1}$ are traded for 2^{k+1} copies of $a_{i,j,k+2}$. At step $3(k + 1) + 1$, 2^{k+1} copies of $a_{i,j,k+2}$ are traded for 2^{k+1} copies of $b_{i,j,k+2}$ by $r_{2,i,j,k+2}$. At step $3(k + 1) + 2$, by the rule $r_{3,i,j,k+2}$, the cell with label 1 gets 2^{k+2} copies of object $c_{i,j}$ and the cell with label 1 has exactly 2^{k+2} copies of $d_{i,j,k+2}$.

On the basis of the above result, specifically, we have, at step $3(n - 3) + 2$, that the cell with label 1 has exactly 2^{n-2} copies of $d_{i,j,n-2}$. In the next three steps, the rules $r_{5,i,j,n-2}$, $r_{2,i,j,n-1}$, and $r_{4,i,j,n-1}$ are applied in order. At step $3(n - 2) + 2$, the cell with label 1 gets 2^{n-1} copies of $c_{i,j}$. Note that no object $d_{i,j,k}$ is brought into the cell with label 1 at step $3(n - 2) + 2$, and the group of rules $r_{5,i,j,k}$, $r_{2,i,j,k}$, and $r_{4,i,j,k}$ cannot be used any more. Therefore the cell with label 1 will not get object $c_{i,j}$ any longer, and the result holds.

For the cases of objects c and A_i , the results can be proved similarly. We omit them here. \square

Lemma 11. *Let \mathcal{C} be an arbitrary computation of the system; then:*

- (1) *For each assignment $\sigma_j(x_i), i = 1, 2, \dots, n$, there exists only one cell with label 2 in \mathcal{C}_{3n-1} that contains the multiset $\{\{\psi(\sigma_j(x_i)) \mid i = 1, 2, \dots, n\}\}$.*
- (2) *There exist exactly 2^n cells with label 2 in configuration \mathcal{C}_k ($k \geq 3n - 1$). In particular, in configuration \mathcal{C}_{3n-1} , each cell with label 2 contains the multiset $\{\{y\}\} \cup \{\{z_{i,j}, x_{i,j} \mid x_{i,j} \in \text{cod}(\varphi)\}\} \cup \{\{z_{i,j}, \bar{x}_{i,j} \mid \bar{x}_{i,j} \in \text{cod}(\varphi)\}\}$ or the multiset $\{\{y'\}\} \cup \{\{z'_{i,j}, x'_{i,j} \mid x_{i,j} \in \text{cod}(\varphi)\}\} \cup \{\{z'_{i,j}, \bar{x}'_{i,j} \mid \bar{x}_{i,j} \in \text{cod}(\varphi)\}\}$.*
- (3) $\mathcal{C}_{3n-1}(1) = \{\{C_{3n}^2, C_{3n}^2, D_{3n}^2, E_{3n}, p, \text{yes}, \text{no}\}\}$.

Proof. We prove the items (1) and (2) of Lemma 11 by induction.

In the configuration \mathcal{C}_0 , there is only one cell with label 2, which has multiset $\{c_{ij}, x_{i,j}, \bar{x}_{i,j} \mid 1 \leq i \leq n, 1 \leq j \leq m\} \cup \{A_1\}$. The rules $r_{19,i,j}-r_{22,i,j}$ and $r_{23,1}$ can be applied. At step 1, the cell with label 2 has multiset $\{z_{i,j}, x_{i,j} \mid x_{i,j} \in \text{cod}(\varphi)\} \cup \{z_{i,j}, \bar{x}_{i,j} \mid \bar{x}_{i,j} \in \text{cod}(\varphi)\} \cup \{z'_{i,j}, x'_{i,j} \mid x_{i,j} \in \text{cod}(\varphi)\} \cup \{z'_{i,j}, \bar{x}'_{i,j} \mid \bar{x}_{i,j} \in \text{cod}(\varphi)\} \cup \{z, z', T_1, F'_1, y, y', s\}$. At step 2, with the appearance of object s , the separation rule r_1 is used to separate the cell with label 2, object s is consumed, and the multiset $\{z_{i,j}, x_{i,j} \mid x_{i,j} \in \text{cod}(\varphi)\} \cup \{z_{i,j}, \bar{x}_{i,j} \mid \bar{x}_{i,j} \in \text{cod}(\varphi)\} \cup \{z, T_1, y\}$ is placed in one new cell with label 2, and the multiset $\{z'_{i,j}, x'_{i,j} \mid x_{i,j} \in \text{cod}(\varphi)\} \cup \{z'_{i,j}, \bar{x}'_{i,j} \mid \bar{x}_{i,j} \in \text{cod}(\varphi)\} \cup \{z', F'_1, y'\}$ is placed in another new cell with label 2. We take the cell with label 2 where $\psi(\sigma_j(x_1))$ appears.

By Lemma 10, at step 2, the cell with label 1 has two copies of $c_{i,j}$, two copies of c , and two copies of A_2 . So, at step 3, the rules $r_{29,i,j}, r_{30,i,j}, r_{33,2}$ and $r_{34,2}$ can be applied. Due to the maximality of the parallelism of using rules, each cell with label 2 gets exactly one copy of $c_{i,j}$, one copy of c , and one copy of A_2 from the cell with label 1. Object $c_{i,j}$ and c are used for duplication, and A_2 is used to assign truth-values to the valuable x_2 . In this way, the next cycle of duplication and separation can continue.

In general, after step $3k + 2$ ($0 \leq k \leq n - 1$) (that is, the second step in the $(k + 1)$ -th cycle of duplication and separation), we take the cell with label 2 where $\{\psi(\sigma_j(x_1)), \dots, \psi(\sigma_j(x_{k+1}))\}$ appears. In this way, at step $3n - 1$, there exists exactly one cell with label 2 whose multiset is $\{\psi(\sigma_j(x_1)), \dots, \psi(\sigma_j(x_n))\}$. (Note the difference of the rule r_{24} and the rules $r_{23,i}$. The rule r_{24} does not bring objects z and z' into cells with label 2 from the environment, and this rule is used at step $3n - 2$. So the object z does not appear in the multiset of cell 2 that corresponds to the assignment σ_j .)

From the above proof, it is easy to see that the multiset $\{\psi(\sigma_j(x_1)), \dots, \psi(\sigma_j(x_n))\}$ appears only in the corresponding cell with label 2.

In every cycle of duplication and separation, the number of cells with label 2 is doubled. In the $3n - 1$ steps, there are n cycles. So there exist exactly 2^n cells with label 2 in configuration \mathcal{C}_{3n-1} ; and from now on, cells with label 2 will not separate any further.

In the last cycle of duplication and separation, at step $3n - 2$, each of the 2^{n-1} cells with label 2 contains one copy of y and one copy of y' by the rule r_{24} ; at step $3n - 1$, the 2^{n-1} cells with label 2 are separated by the rule r_1 , and each of the 2^n new cells gets one copy of object y or one copy of object y' .

In order to prove $\mathcal{C}_{3n-1}(1) = \{\{B_{3n}^{2^n}, C_{3n}^{2^n}, D_{3n}^{2^n}, E_{3n}, p, \text{yes}, \text{no}\}\}$, we will check how all the rules related to the cell 1 work in the first $3n - 1$ steps.

- Checking the rules $r_{2,i,j,k}-r_{18,i}$.
From the proofs of Lemmas 10 and 11, we can find that after step $3n - 3$, there are no objects $a_{i,j,k}, b_{i,j,k}, c_{i,j}, d_{i,j,k}, a_{1,i}, b_{1,i}, c, d_{1,i}, e_i, a_{2,i}, b_{2,i}, d_{2,i}, g_i, h_i, l_i, A_i$ in the cell with label 1, and the rules $r_{2,i,j,k}-r_{18,i}$ will not bring any more objects into the cell with label 1.
- Checking the rules $r_{35,i}-r_{40,i}$.
In the first $2n - 1$ steps of the computation, by the rules $r_{35,i}, r_{37,i}$, and $r_{39,i}$, the subscripts of B_1, C_1 , and D_1 grow to $2n$. In the next n steps, by the rules $r_{36,i}, r_{38,i}$, and $r_{40,i}$, the subscripts of B_{2n}, C_{2n} , and D_{2n} grow to $3n$, and at every step, the numbers of objects of each type B_i, C_i , and D_i are doubled. So the cell with label 1 has 2^n copies of B_{3n} , 2^n copies of C_{3n} , and 2^n copies of D_{3n} at the step $3n - 1$.
- Checking the rule $r_{41,i}$.
By the rule $r_{41,i}$, the subscript of E_1 grows to $3n$ in the first $3n - 1$ steps of the computation. So the cell 1 has the object E_{3n} at step $3n - 1$.
- Checking the group of rules $r_{29,i,j}-r_{34,i,j}$ and the group of rules $r_{42,i,j}-r_{47}$.
In the first $3n - 3$ steps, the cell with label 1 has communication with cells with label 2 getting objects $z_{i,j}, z'_{i,j}, z, z', y, y'$ from cells with label 2, by the rules $r_{29,i,j}-r_{34,i,j}$. In the next step after the objects $z_{i,j}, z'_{i,j}, z, z', y, y'$ reach the cell 1, they are sent to the environment by the rules $r_{42,i,j}-r_{47}$.
- Checking the group of rules $r_{48}-r_{49}$.
At the step $3n - 1$, the subscripts of objects B_i, C_i and D_i grow to $3n$. The rules $r_{48}-r_{49}$ can be applied at step $3n$. But, in the first $3n - 1$ steps of the computation, they cannot be applied.
- Checking the rules $r_{66}-r_{68}$.
In the first steps $3n - 1$, there are no objects D_{4n+2m} appearing in cells with label 2, and no object $E_{4n+2m+3}$ appearing in the cell with label 1. The rules $r_{66}-r_{68}$ cannot be applied in the first $3n - 1$ steps of the computation, so the cell with label 1 has objects yes, no and p at the step $3n - 1$.

Therefore, $\mathcal{C}_{3n-1}(1) = \{\{B_{3n}^{2^n}, C_{3n}^{2^n}, D_{3n}^{2^n}, E_{3n}, p, \text{yes}, \text{no}\}\}$. \square

Lemma 12. Let \mathcal{C} be an arbitrary computation of the system; then:

- (1) $\mathcal{C}_{3n}(1) = \{(y^{2^{n-1}}, (y')^{2^{n-1}}, E_{3n+1}, p, \text{yes}, \text{no})\}$.
- (2) For each assignment σ_j there exists only one cell with label 2 in \mathcal{C}_{3n} that contains multiset $\{\{\psi(\sigma_j(x_i)) \mid i = 1, 2, \dots, n\}\}$. In configuration \mathcal{C}_{3n} , each cell with label 2 contains the multiset $\{\{B_{3n}, C_{3n}, D_{3n}\} \cup \{z_{i,j}, x_{i,j} \mid x_{i,j} \in \text{cod}(\varphi)\}\} \cup \{z_{i,j}, \bar{x}_{i,j} \mid \bar{x}_{i,j} \in \text{cod}(\varphi)\}\}$ or multiset $\{\{B_{3n}, C_{3n}, D_{3n}\} \cup \{z'_{i,j}, x'_{i,j} \mid x_{i,j} \in \text{cod}(\varphi)\}\} \cup \{z'_{i,j}, \bar{x}'_{i,j} \mid \bar{x}_{i,j} \in \text{cod}(\varphi)\}\}$.

The configuration \mathcal{C}_{3n} is obtained from \mathcal{C}_{3n-1} using the rules $r_{41,i}$, r_{48} and r_{49} . So, Lemma 12 can be proved by checking the application of the rules $r_{41,i}$, r_{48} and r_{49} . We here omit the details of the proof of Lemma 12.

Lemma 13. Let \mathcal{C} be an arbitrary computation of the system; then:

- (1) $\mathcal{C}_{4n}(1) = \{E_{4n+1}, p, \text{yes}, \text{no}\}$.
- (2) For each assignment σ_j , there exists only one cell with label 2 in \mathcal{C}_{4n} that contains multiset $\{\{\gamma(\sigma_j(x_i)) \mid i = 1, 2, \dots, n\}\}$. In configuration \mathcal{C}_{4n} , each cell with label 2 contains the multiset $\{\{B_{4n}^{2^n}, C_{3n}, D_{4n}\} \cup \{z_{i,j}, x_{i,j} \mid x_{i,j} \in \text{cod}(\varphi)\}\} \cup \{z_{i,j}, \bar{x}_{i,j} \mid \bar{x}_{i,j} \in \text{cod}(\varphi)\}\}$ or the multiset $\{\{B_{4n}^{2^n}, C_{3n}, D_{4n}\} \cup \{z'_{i,j}, x'_{i,j} \mid x_{i,j} \in \text{cod}(\varphi)\}\} \cup \{z'_{i,j}, \bar{x}'_{i,j} \mid \bar{x}_{i,j} \in \text{cod}(\varphi)\}\}$.

The configuration \mathcal{C}_{4n} is obtained from \mathcal{C}_{3n} using the rules $r_{41,i}$, r_{44} , r_{45} and $r_{50,i}-r_{55,i}$. Lemma 13 can be proved by checking the application of these rules, so the proof is omitted.

Lemma 14. Let \mathcal{C} be an arbitrary computation of the system; then:

- (1) $\mathcal{C}_{4n+m}(1) = \{E_{4n+m+1}, p, \text{yes}, \text{no}\}$.
- (2) For each assignment σ_j , there exists only one cell with label 2 in \mathcal{C}_{4n+m} that contains multiset $(\cup_{i=1}^n \{r_{j_1} r_{j_2} \cdots r_{j_k} \mid \gamma(\sigma_j(x_i)) = t_i, \text{ and } x_{i,j_l} \in \text{cod}(\varphi), l = 1, 2, \dots, k, 1 \leq j_1 < j_2 < \cdots < j_k \leq n\}) \cup (\cup_{i=1}^n \{r_{j_1} r_{j_2} \cdots r_{j_k} \mid \gamma(\sigma_j(x_i)) = f_i, \text{ and } \bar{x}_{i,j_l} \in \text{cod}(\varphi), l = 1, 2, \dots, k, 1 \leq j_1 < j_2 < \cdots < j_k \leq n\})$. In configuration \mathcal{C}_{4n+m} , each cell with label 2 contains the multiset $\{\{B_{4n}^{2^n}, C_{3n}, D_{4n+m}\}\}$.

Proof. By the rule $r_{41,i}$, the subscript of object E_{4n+1} in $\mathcal{C}_{4n}(1)$ grows to the value $4n + m + 1$ at step $4n + m$. The objects p, yes, no stay unchanged. Therefore, the item (1) of Lemma 14 holds.

By Lemma 13, for each assignment σ_j , there exists only one cell with label 2 in \mathcal{C}_{4n} that contains multiset $\{\{\gamma(\sigma_j(x_i)) \mid i = 1, 2, \dots, n\}\}$; and each cell with label 2 contains a multiset $\{\{B_{4n}, C_{3n}, D_{4n}\} \cup \{z_{i,j}, x_{i,j} \mid x_{i,j} \in \text{cod}(\varphi)\}\} \cup \{z_{i,j}, \bar{x}_{i,j} \mid \bar{x}_{i,j} \in \text{cod}(\varphi)\}\}$ or a multiset $\{\{B_{4n}, C_{3n}, D_{4n}\} \cup \{z'_{i,j}, x'_{i,j} \mid x_{i,j} \in \text{cod}(\varphi)\}\} \cup \{z'_{i,j}, \bar{x}'_{i,j} \mid \bar{x}_{i,j} \in \text{cod}(\varphi)\}\}$. In the following, we consider this unique cell with label 2 that contains multiset $\{\{\gamma(\sigma_j(x_i)) \mid i = 1, 2, \dots, n\}\}$.

The objects C_{3n} remain unchanged, and the subscript of D_{4n} reaches $4n + m$ at step $4n + m$ by the rule $r_{64,i}$.

With the presence of B_{4n} in \mathcal{C}_{4n} (not appearing in \mathcal{C}_i ($i < 4n$)), the rules $r_{56,i,j}-r_{63,i,j}$ can be applied. We start to check which clauses are satisfied. If $\sigma_j((x_i)) = t_i$ and $x_{i,j} \in \text{cod}(\varphi)$, then rule $r_{56,i,j}$ or $r_{58,i,j}$ is applied, and an object r_j is introduced into the corresponding cell with label 2. If $\sigma_j((x_i)) = t_i$ and $\bar{x}_{i,j} \in \text{cod}(\varphi)$, then rule $r_{57,i,j}$ or $r_{59,i,j}$ is applied, and the object $\bar{x}_{i,j}$ or $\bar{x}'_{i,j}$ is removed from the corresponding cell with label 2. Similarly, if $\sigma_j((x_i)) = f_i$ and $\bar{x}_{i,j} \in \text{cod}(\varphi)$, then rule $r_{60,i,j}$ or $r_{62,i,j}$ is applied, and an object r_j is introduced into the corresponding cell with label 2. If $\sigma_j((x_i)) = f_i$ and $x_{i,j} \in \text{cod}(\varphi)$, then rule $r_{61,i,j}$ or $r_{63,i,j}$ is applied, and the object $x_{i,j}$ or $x'_{i,j}$ is removed from the corresponding cell with label 2. The sizes of both $\text{cod}(\varphi)$ and $\{x'_{i,j} \mid x_{i,j} \in \text{cod}(\varphi)\} \cup \{\bar{x}'_{i,j} \mid \bar{x}_{i,j} \in \text{cod}(\varphi)\}$ are nm , and each cell with label 2 contains multiset $\text{cod}(\varphi)$ or $\{x'_{i,j} \mid x_{i,j} \in \text{cod}(\varphi)\} \cup \{\bar{x}'_{i,j} \mid \bar{x}_{i,j} \in \text{cod}(\varphi)\}$. We have 2^n copies of B_{4n} , n objects t_i and f_i from the multiset $\{\{\gamma(\sigma_j(x_i)) \mid i = 1, 2, \dots, n\}\}$, so it takes m steps to check which clauses are satisfied. In total, all the objects introduced, r_i , form the multiset $(\cup_{i=1}^n \{r_{j_1} r_{j_2} \cdots r_{j_k} \mid \gamma(\sigma_j(x_i)) = t_i, \text{ and } x_{i,j_l} \in \text{cod}(\varphi), l = 1, 2, \dots, k, 1 \leq j_1 < j_2 < \cdots < j_k \leq n\}) \cup (\cup_{i=1}^n \{r_{j_1} r_{j_2} \cdots r_{j_k} \mid \gamma(\sigma_j(x_i)) = f_i, \text{ and } \bar{x}_{i,j_l} \in \text{cod}(\varphi), l = 1, 2, \dots, k, 1 \leq j_1 < j_2 < \cdots < j_k \leq n\})$. \square

Lemma 15. Let \mathcal{C} be an arbitrary computation of the system; then:

- (1) $\mathcal{C}_{4n+2m}(1) = \{\{E_{4n+2m+1}, p, \text{yes}, \text{no}\}\}$.
- (2) If σ_j is an assignment that does not satisfy the formula φ , then there exists only one cell with label 2 in \mathcal{C}_{4n+2m} associated with σ_j , and whose associated multiset contains an object $D_{4n+m+\alpha}$, where $0 \leq \alpha < m$, such that the clauses M_1, \dots, M_α are satisfied by the assignment f , but $M_{\alpha+1}$ is not satisfied by the assignment σ_j .
- (3) If σ_j is an assignment that satisfies the formula φ , then there exists only one cell 2 in \mathcal{C}_{4n+2m} associated with σ_j , and whose associated multiset contains one copy of object D_{4n+2m} .

Proof. By the rule $r_{41,i}$, the subscript of object E_{4n+m+1} in $\mathcal{C}_{4n+m}(1)$ grows to the value $4n + 2m + 1$ at step $4n + 2m$. The objects p, yes, no remain unchanged. Therefore, the item (1) of Lemma 15 holds.

From the configuration \mathcal{C}_{4n+m} , we start to check whether or not all clauses are satisfied by the corresponding assignment. Such checking is simultaneous in all 2^n cells with label 2. Let us consider an assignment σ_j . By Lemma 14, with the presence of object D_{4n+m} , the rule $r_{65,i}$ can be applied. The clauses are checked in the order from M_1 to M_m . For each clause which is satisfied (that is, the corresponding object r_i appears), we increase by 1 the subscript of D_i ; hence the subscript of D_i reaches the value $4n + 2m$ if and only if all clauses are satisfied. If the clauses M_1, \dots, M_α ($0 \leq \alpha < m$) are satisfied, but $M_{\alpha+1}$ is not satisfied (that is, r_1, \dots, r_α appear, but $r_{\alpha+1}$ does not appear), then the subscript of D_i can only reach the value $4n + m + \alpha$. Therefore, the items (2) and (3) of Lemma 15 hold. \square

Lemma 16. Let \mathcal{C} be an arbitrary computation of the system; let us suppose that there exists an assignment that satisfies the formula φ . Then:

- (a) $\mathcal{C}_{4n+2m+1}(1) = \{\{E_{4n+2m+2}, D_{4n+2m}^\beta, p, \text{yes}, \text{no}\}\}$,
- (b) $\mathcal{C}_{4n+2m+2}(1) = \{\{E_{4n+2m+3}, D_{4n+2m}^{\beta-1}, \text{no}\}\}$,

where β is the number of assignments that satisfy the formula φ . Furthermore, the object yes appears in $\mathcal{C}_{4n+2m+2}(0)$.

Proof. The configuration of item (a) is obtained by the application of rules $r_{41,i}$ and r_{66} to the previous configuration \mathcal{C}_{4n+2m} . By the rule $r_{41,i}$, the object $E_{4n+2m+1}$ in $\mathcal{C}_{4n+2m}(1)$ grows by 1 its subscript at step $4n + 2m + 1$. By Lemma 15, for each assignment that satisfies the formula φ , there exists exactly one cell associated with label 2 in \mathcal{C}_{4n+2m} whose multiset contains an object D_{4n+2m} . The object D_{4n+2m} is moved to the cell with label 1 by the rule r_{66} . If there are β assignments that satisfy the formula φ , then the cell 1 gets β copies of object D_{4n+2m} .

The configuration of item (b) is obtained by the application of rules $r_{41,i}$ and r_{67} to the previous configuration $\mathcal{C}_{4n+2m+1}(1)$. By the rule $r_{41,i}$, the object $E_{4n+2m+2}$ in $\mathcal{C}_{4n+2m+1}(1)$ grows by 1 its subscript at step $4n + 2m + 2$. By the rule r_{67} , the object yes together with objects D_{4n+2m} and p leaves the system, going into the environment, signaling that the formula φ is satisfiable. The one copy of object p is consumed by the rule r_{67} , so the rule r_{68} cannot be applied. The object no cannot exit into the environment. \square

Lemma 17. Let \mathcal{C} be an arbitrary computation of the system, let us suppose that there does not exist any assignment that satisfies the formula φ . Then:

- (a) $\mathcal{C}_{4n+2m+1}(1) = \{\{E_{4n+2m+2}, p, \text{yes}, \text{no}\}\}$,
- (b) $\mathcal{C}_{4n+2m+2}(1) = \{\{E_{4n+2m+3}, p, \text{yes}, \text{no}\}\}$,
- (c) $\mathcal{C}_{4n+2m+3}(1) = \{\{\text{yes}\}\}$.

Furthermore, the object no appears in $\mathcal{C}_{4n+2m+3}(0)$.

Proof. If there does not exist any assignment that satisfies the formula φ , by Lemma 15, all cells with label 2 do not contain object D_{4n+2m} . Of course, the cell with label 1 cannot get object D_{4n+2m} .

The configurations of items (a) and (b) are obtained by the application of rules $r_{41,i}$ to the previous configuration \mathcal{C}_{4n+2m} .

The configuration of item (c) is obtained by the application of rules r_{68} to the previous configuration. \square

6.3. Efficiency results of TSC(6)

The system constructed for the solution of SAT in Section 6 has communication rules with length at most 6. From the discussion in the previous sections and according to the definition of solvability given in Section 4, we have the following result:

Theorem 18. $SAT \in \mathbf{PMC}_{TSC(6)}$.

Corollary 19. $\mathbf{NP} \cup \mathbf{co-NP} \subseteq \mathbf{PMC}_{TSC(6)}$.

Proof. It suffices to make the following observations: SAT is **NP**-complete, $SAT \in \mathbf{PMC}_{TSC(6)}$ and this complexity class is closed under polynomial-time reduction and under complement. \square

7. Conclusion and further works

This paper studies the efficiency of P systems in solving **NP**-complete problems. The space–time tradeoff method is used to efficiently solve **NP**-complete problems in the framework of cell-like P systems. Membrane division, membrane creation, and membrane separation are three efficient approaches for obtaining exponential workspace in polynomial time. In this paper, membrane separation is introduced into tissue P systems, and a polynomial-time solution for SAT using tissue P systems with cell separation and communication rules of lengths at most 6 is presented. We also prove that tissue P systems with cell separation and communication rules of length 1 can only solve tractable problems in the classical sense. Hence, in the framework of recognizer tissue P systems with cell separation, the lengths of the communication rules provide a borderline between efficiency and non-efficiency. Specifically, a frontier exists between length 1 and length 6. The role of the lengths of communication rules is worthy of further investigation. That is, what does happen if we consider tissue P systems with communication rules of length k , where $k \in \{2, 3, 4, 5\}$?

In the framework of tissue P systems, when cell division is used to generate exponential workspace in polynomial time, there is an advantage: all the other objects in the cell are duplicated except the objects that activate the cell division operation. Both cell creation and cell separation have no such duplication feature. In this sense, the solution for SAT presented in this paper gives some hints for answering the following open problem: how do we efficiently solve **NP**-complete problems using tissue P systems with cell creation?

Although SAT is **NP**-complete (other **NP** problems can be reduced to SAT in polynomial time), we wish to stress that up to now there has existed no methodology for computing the reduction process by using P systems. The solution to SAT using tissue P systems with cell separation can be used as a scheme for designing solutions to other **NP**-complete problems such as the *vertex-cover* problem, the *clique* problem, the *Hamiltonian path* problem, etc.

Recently, a new kind of P system model with a neural-like architecture, called the spiking neural P systems, was introduced [8]. It was proved that spiking neural P systems are Turing complete [8].

There is an interesting result for the efficiency of spiking neural P systems in solving computationally hard problems: an spiking neural P system of polynomial size cannot solve an **NP**-complete problem in a deterministic way and in polynomial time (unless $\mathbf{P} = \mathbf{NP}$) [12]. Hence, under the assumption that $\mathbf{P} \neq \mathbf{NP}$, efficient solutions to **NP**-complete problems cannot be obtained without introducing features which enhance the efficiency. One of the possible features is some way of increasing the workspace exponentially during the computation. Cell division, cell creation and cell separation are candidates for being introduced into spiking neural P systems for exponential workspace generation. Although the architectures of spiking neural P systems are similar to those of tissue P systems, it is not a trivial work to introduce cell division, cell creation and cell separation into spiking neural P systems and give efficient solutions to **NP**-complete problems.

In general, it is expected that the research into efficiency and complexity on P systems will provide insight into unconventional parallel computing models, and also help us clarify the relations between classic complexity classes.

Acknowledgments

Comments from three anonymous referees are greatly appreciated. The work of L. Pan was supported by the National Natural Science Foundation of China (Grant Nos. 60674106, 30870826, 60703047, and 60533010), the Program for New Century Excellent Talents in University (NCET-05-0612), the Ph.D. Programs Foundation of the Ministry of Education of China (20060487014), the Chenguang Program of Wuhan (200750731262), HUST-SRF (2007Z015A), and the Natural Science Foundation of Hubei Province (2008CDB113 and 2008CDB180). The work of M.J. Pérez-Jiménez was supported by Project TIN2006-13452 of the Ministerio de Educación y Ciencia of Spain and the Project of Excellence with *Investigador de Reconocida Valía*, from Junta de Andalucía, Grant P08—TIC 04200.

References

- [1] A. Alhazov, R. Freund, M. Oswald, Tissue P systems with antiport rules and small numbers of symbols and cells, *Lecture Notes in Computer Science* 3572 (2005) 100–111.
- [2] F. Bernardini, M. Gheorghe, Cell communication in tissue P systems and cell division in population P systems, *Soft Computing* 9 (9) (2005) 640–649.
- [3] R. Freund, Gh. Păun, M.J. Pérez-Jiménez, Tissue P systems with channel states, *Theoretical Computer Science* 330 (2005) 101–116.
- [4] P. Frisco, *Computing with Cells: Advances in Membrane Computing*, Oxford University Press, 2009.
- [5] M.R. Garey, D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman and Company, 1979.
- [6] M.A. Gutiérrez-Naranjo, M.J. Pérez-Jiménez, A. Riscos-Núñez, F.J. Romero-Campero, On the power of dissolution in P systems with active membranes, *Lecture Notes in Computer Science* 3850 (2006) 224–240.
- [7] M.A. Gutiérrez-Naranjo, M.J. Pérez-Jiménez, F.J. Romero-Campero, A linear solution for QSAT with membrane creation, *Lecture Notes in Computer Science* 3850 (2006) 241–252.
- [8] M. Ionescu, Gh. Păun, T. Yokomori, Spiking neural P systems, *Fundamenta Informaticae* 71 (2–3) (2006) 279–308.
- [9] M. Ito, C. Martín-Vide, Gh. Păun, A characterization of Parikh sets of ETOL languages in terms of P systems, in: M. Ito, Gh. Păun, S. Yu (Eds.), *Words, Semigroups and Transducers*, World Scientific, Singapore, 2001, pp. 239–254.
- [10] S.N. Krishna, K. Lakshmanan, R. Rama, Tissue P systems with contextual and rewriting rules, *Lecture Notes in Computer Science* 2597 (2003) 339–351.
- [11] K. Lakshmanan, R. Rama, On the power of tissue P systems with insertion and deletion rules, in: A. Alhazov, C. Martín-Vide, Gh. Păun (Eds.), *Preproceedings of the Workshop on Membrane Computing*, Tarragona, Report RGML 28/03, 2003, pp. 304–318.
- [12] A. Leporati, C. Zandron, C. Ferretti, G. Mauri, On the computational power of spiking neural P systems, in: Eleftherakis, G., et al. (Eds.), *Proceedings of the Eighth Workshop on Membrane Computing*, Thessaloniki, June 2007, pp. 405–423.
- [13] C. Martín-Vide, J. Pazos, Gh. Păun, A. Rodríguez Patón, A new class of symbolic abstract neural nets: tissue P systems, *Lecture Notes in Computer Science* 2387 (2002) 290–299.
- [14] C. Martín-Vide, J. Pazos, Gh. Păun, A. Rodríguez Patón, Tissue P systems, *Theoretical Computer Science* 296 (2003) 295–326.
- [15] L. Pan, T.-O. Ishdorj, P systems with active membranes and separation rules, *Journal of Universal Computer Science* 10 (5) (2004) 630–649.
- [16] Gh. Păun, *Attacking NP-complete problems*, in: I. Antoniou, C. Calude, M.J. Dinneen (Eds.), *Unconventional Models of Computation, UMC'2K*, Springer-Verlag, 2000, pp. 94–115.
- [17] Gh. Păun, *Computing with membranes*, *Journal of Computer and System Sciences* 61 (1) (2000) 108–143.
- [18] Gh. Păun, *Membrane Computing. An Introduction*, Springer-Verlag, Berlin, 2002.
- [19] A. Păun, Gh. Păun, The power of communication: P systems with symport/antiport, *New Generation Computing* 20 (3) (2002) 295–305.
- [20] Gh. Păun, M.J. Pérez-Jiménez, A. Riscos-Núñez, Tissue P system with cell division, *International Journal of Computers, Communications & Control* 3 (3) (2008) 295–303.
- [21] M.J. Pérez-Jiménez, A. Romero-Jiménez, F. Sancho-Caparrini, The P versus NP problem through cellular computing with membranes, *Lecture Notes in Computer Science* 2950 (2004) 338–352.
- [22] M.J. Pérez-Jiménez, A. Romero-Jiménez, F. Sancho-Caparrini, A polynomial complexity class in P systems using membrane division, *Journal of Automata, Languages and Combinatorics* 11 (4) (2006) 423–434. A preliminary version, in: E. Csuhaj-Varjú, C. Kintala, D. Wotschke, Gy. Vaszil (Eds.), *Proceedings of the Fifth International Workshop on Descriptive Complexity of Formal Systems, DCFs 2003*, Budapest, Hungary, July 12–14, 2003, pp. 284–294.
- [23] V.J. Prakash, On the power of tissue P systems working in the maximal-one mode, in: A. Alhazov, C. Martín-Vide, Gh. Păun (Eds.), *Preproceedings of the Workshop on Membrane Computing*, Tarragona, Report RGML 28/03, 2003, pp. 356–364.
- [24] C. Zandron, C. Ferretti, G. Mauri, Solving NP-complete problems using P systems with active membranes, in: I. Antoniou, C.S. Calude, M.J. Dinneen (Eds.), *Unconventional Models of Computation, UMC'2K*, Springer-Verlag, 2000, pp. 289–301.
- [25] ISI web page. <http://esi-topics.com/erf/october2003.html>.
- [26] P systems web page. <http://ppage.psysteams.eu/>.