Hindawi
Complexity
Volume 2018, Article ID 3745210, 21 pages
https://doi.org/10.1155/2018/3745210

Research Article

The Computational Complexity of Tissue P Systems with Evolutional Symport/Antiport Rules

Linqiang Pan,[1,2] **Bosheng Song**,[1] **Luis Valencia-Cabrera**,[3] **and Mario J. Pérez-Jiménez**[3]

[1]*Key Laboratory of Image Information Processing and Intelligent Control of Education Ministry of China, School of Automation, Huazhong University of Science and Technology, Wuhan, Hubei 430074, China*
[2]*School of Electric and Information Engineering, Zhengzhou University of Light Industry, Zhengzhou 450002, China*
[3]*Research Group on Natural Computing, Department of Computer Science and Artificial Intelligence, University of Sevilla, Avda. Reina Mercedes s/n, 41012 Sevilla, Spain*

Correspondence should be addressed to Bosheng Song; boshengsong@hust.edu.cn

Received 29 May 2017; Accepted 18 December 2017; Published 23 April 2018

Academic Editor: Sigurdur F. Hafstein

Tissue P systems with evolutional communication (symport/antiport) rules are computational models inspired by biochemical systems consisting of multiple individuals living and cooperating in a certain environment, where objects can be modified when moving from one region to another region. In this work, cell separation, inspired from membrane fission process, is introduced in the framework of tissue P systems with evolutional communication rules. The computational complexity of this kind of P systems is investigated. It is proved that only problems in class **P** can be efficiently solved by tissue P systems with cell separation with evolutional communication rules of length at most $(n, 1)$, for each natural number $n \geq 1$. In the case where that length is upper bounded by $(3, 2)$, a polynomial time solution to the SAT problem is provided, hence, assuming that $\mathbf{P} \neq \mathbf{NP}$ a new boundary between tractability and **NP**-hardness on the basis of the length of evolutional communication rules is provided. Finally, a new simulator for tissue P systems with evolutional communication rules is designed and is used to check the correctness of the solution to the SAT problem.

1. Introduction

A cell is the basic unit of biological organization that constitutes all living organisms. There are many different types of biological cells, which have different specialized functions that maintain an organism working properly. Inspired by the structure and functioning of living cells, Păun proposed a computing paradigm in 2000 [1], called *membrane computing*, which has become an active research area (for other research domains of natural computing, one can refer to [2, 3]). A large number of theoretical models were proposed [4–6] and they have been used to solve various real problems [7–10]. These distributed and parallel computation models investigated in membrane computing are called *P systems*. In general, there exist two main families of P systems: *cell-like P systems* [1], which have a hierarchical arrangement of membranes described by a rooted tree (corresponding to cell-like membrane structure), and *tissue-like P systems* [11] or *neural-like P systems* [12], which have a net of cells or neurons described by a directed graph. An overview of membrane computing can be found in [13]. The present work deals with tissue-like P systems.

Inspired by the biological phenomenon of trans-membrane transport of couples of chemicals, communication P systems with symport/antiport rules were proposed in [14], where symport rules move objects (without evolution) between two regions in one direction, and antiport rules move objects (without evolution) between two regions in opposite directions. Later, symport/antiport rules were considered in tissue-like P systems [11], where cells are placed in nodes of a directed graph, and an arc between two nodes corresponds to a communication channel between cells placed in these nodes.

Since the notion of tissue P systems was proposed, numerous research topics have been arisen [15–18], and various ingredients (energy, catalyst, mitosis, etc.) from other

computational models were considered in the context of tissue P systems. In [19], tissue P systems with channel states controlling the communication between two regions were proposed, and several Turing universality results were achieved, where the systems work in a maximally parallel way with sequential behavior on channels. In [20], a general model of tissue P systems with channel states that allow us to model hybrid cooperating grammar systems was considered, where the results were established for strings and arrays.

Tissue P systems have been used to find polynomial time solutions to **NP**-complete problems. In [21], membrane division rules used in P systems with active membranes have been introduced into tissue P systems yielding tissue P systems with cell division, and a polynomial time uniform solution to the SAT problem was shown. Since then, tissue P systems with cell division were also considered to solve other **NP**-complete problems: bin packing [22], subset sum [23], vertex cover [24], and so on. Cell division rules have a replication functioning; that is, two new created cells by a division rule have exactly the same objects except for at most a pair of different objects. Inspired from membrane fission process, cell separation rules are another way to obtain an exponential workspace in polynomial time, but they do not have the duplication functioning; that is, when a cell is separated, the objects in the cell are distributed in each of the newly created cells. Tissue P systems with cell separation have also been used to solve **NP**-complete problems in polynomial time; one can refer to [25, 26] for these investigations.

Computational complexity theory in the framework of tissue P systems was introduced in [21] and it has been studied in [27–30]. It was shown that in the framework of tissue P systems with cell division, only tractable problems can be efficiently solved by using communication rules with length at most one [31] (the length of such a rule is the total number of objects involved in it), but a uniform polynomial time solution to the HAM-CYCLE problem by a family of such P systems using communication rules with length at most two has been given [30]. On the other hand, in the framework of tissue P systems with cell separation, by using communication rules with length at most two, only tractable problems can be efficiently solved, but the SAT problem can be solved by this kind of P systems using communication rules with length at most three [26]. Moreover, frontiers between efficiency and nonefficiency in terms of the length of symport/antiport rules in the framework of cell-like P systems have been investigated in [32].

Tissue-like P systems with evolutional symport/antiport rules (TESA P systems, for short) were proposed in [33], where objects are moved from one region to another region and may be evolved during this process. In [33], the computational efficiency of TESA P systems with cell division (TESAD P systems, for short) was investigated. It is shown that a limit on the efficiency of TESAD P systems is provided with evolutional communication rules of length at most 2. However, when using evolutional communication rules of length at most 4, the SAT problem can be solved by TESAD P systems. However, it is still an open problem as formulated in [33] related to the role of evolutional communication rules in tissue P systems with cell separation from a computational complexity point of view.

During those computational complexity studies for new variants of P systems, the solutions designed for NP-complete problems are frequently difficult to follow, and requerying makes sure that the evolution of the systems is exactly as expected. In this context, the aid of computer tools to assist in both the design and verification tasks may be crucial, producing much more reliable solutions. In this sense, the development of P-Lingua [34–36] implied a significant progress. This open source framework includes a standard language aiming to specify the elements of different types of P systems using a notation very close to the researchers in membrane computing community. Besides, it contains simulation engines for a number of P system types. On top of that, MeCoSim [37, 38] provides an additional layer of abstraction with a visual application where researchers can explore at a higher level the evolution of their solutions based on P systems.

In this work, we investigate tissue P systems with evolutional symport/antiport rules and cell separation (TESAS P systems, for short) from a computational complexity point of view.

Contributions of the present work are summarized as follows:

(a) A variant of tissue P systems with evolutional symport/antiport rules, called *tissue P systems with evolutional symport/antiport rules and cell separation* (TESAS P systems, for short), and the corresponding recognizer version are proposed. In TESAS P systems, the length of an evolutional symport/antiport rule r is defined as an ordered pair whose first component is the total number of objects involved in the left hand side (LHS) of the rule and the second component is the total number of objects involved in the right hand side (RHS) of the rule; that is, $\text{length}(r) = (\text{length}(\text{LHS}(r)), \text{length}(\text{RHS}(r)))$. The set of all recognizer TESAS P systems with evolutional communication rules of length at most (k_1, k_2) is denoted by $\mathbf{TSEC}(k_1, k_2)$.

(b) The computation efficiency of recognizer TESAS P systems is investigated in terms of the length of evolutional communication rules. It is shown that only tractable problems can be efficiently solved by families of systems from $\mathbf{TSEC}(n, 1)$ or from $\mathbf{TSEC}(1, n)$, for each natural number $n \geq 1$. We further show that the SAT problem can be solved in polynomial time by a family of systems from $\mathbf{TSEC}(3, 2)$, hence, assuming that $\mathbf{P} \neq \mathbf{NP}$, a new boundary between tractability and **NP**-hardness on the basis of the length of evolutional communication rules, is provided.

(c) A new simulator MeCoSim is designed in order to check the correctness of the solution to the SAT problem. By using the software MeCoSim, we can analyse the designed P systems, then run the simulation, and obtain the computation results.

2. Tissue P Systems with Evolutional Symport/Antiport Rules and Cell Separation

Let us start this section by recalling some notions from formal language theory used in this work; the reader can find details in [39].

An alphabet Γ is a nonempty set. Any sequence u of elements from Γ is called a *string* over Γ. The *length* of u, denoted by $|u|$, is the number of occurrences in u of symbols from Γ.

A *multiset* over an alphabet Γ is a function m from Γ to the set of natural numbers $\mathbb{N}$. The multiplicity of a symbol $a \in \Gamma$ in the multiset m is $m(a)$. The *support* of m is the set of symbols a such that $m(a) > 0$. A multiset is finite and its support is a finite set. The set of all finite multisets over Γ is denoted by $M_f(\Gamma)$, and by $M_f^+(\Gamma)$ we denote the set of all nonempty finite multisets over an alphabet Γ, and the empty multiset is denoted by $\emptyset$. The cardinal of a finite multiset m, denoted by $|m|$, is the sum of all multiplicities of elements in the support of m. If m_1 and m_2 are multisets over Γ, then we define the union of m_1 and m_2, denoted by m_1+m_2, as follows: $(m_1 + m_2)(a) = m_1(a) + m_2(a)$, for each $a \in \Gamma$.

Definition 1. A tissue P system with evolutional symport/antiport rules and cell separation of degree $q \geq 1$ is a tuple

$$\Pi = \left(\Gamma, \Gamma_0, \Gamma_1, \mathscr{E}, \mathscr{M}_1, \ldots, \mathscr{M}_q, \mathscr{R}, i_{\text{out}}\right), \tag{1}$$

where

(i) Γ and $\mathscr{E}$ are finite alphabets such that $\mathscr{E} \subseteq \Gamma$;

(ii) Γ_0, Γ_1 are nonempty sets such that $\Gamma_0 \cup \Gamma_1 = \Gamma$ and $\Gamma_0 \cap \Gamma_1 = \emptyset$;

(iii) $\mathscr{M}_1, \ldots, \mathscr{M}_q$ are multisets over Γ;

(iv) $\mathscr{R}$ is a finite set of rules of the following forms:

(1) evolutional communication rules:

(a) $[u]_i[\ ]_j \rightarrow [\ ]_i[u']_j$, where $0 \leq i, j \leq q$, $i \neq j$, $u \in M_f^+(\Gamma)$, $u' \in M_f(\Gamma)$ (evolutional symport rules);

(b) $[u]_i[v]_j \rightarrow [v']_i[u']_j$, where $0 \leq i, j \leq q, i \neq j, u, v \in M_f^+(\Gamma), u', v' \in M_f(\Gamma)$ (evolutional antiport rules);

(2) separation rules:

(a) $[a]_i \rightarrow [\Gamma_0]_i[\Gamma_1]_i$, where $i \in \{1, \ldots, q\}$, $i \neq i_{\text{out}}$, $a \in \Gamma$;

(v) $i_{\text{out}} \in \{0, 1, \ldots, q\}$.

A tissue P system with evolutional symport/antiport rules and cell separation of degree $q \geq 1$, $\Pi = (\Gamma, \Gamma_0, \Gamma_1, \mathscr{E}, \mathscr{M}_1, \ldots, \mathscr{M}_q, \mathscr{R}, i_{\text{out}})$, can be viewed as a set of q cells, labelled by $1, \ldots, q$ such that (a) $\mathscr{M}_1, \ldots, \mathscr{M}_q$ represent the multisets of objects initially placed in the q cells of the system; (b) $\mathscr{E}$ is the set of objects initially located in the environment of the system, all of them available in an arbitrary number of copies; and (c) i_{out} represents a distinguished *region* which will encode the output of the system. We use the term *region* i $(0 \leq i \leq q)$ to refer to cell i in the case $1 \leq i \leq q$ and to refer to the environment in the case $i = 0$.

A *configuration* at any instant of a TESAS P system is described by multisets of objects in each cell and the multiset of objects over $\Gamma \setminus \mathscr{E}$ in the environment at that moment. The initial configuration of $\Pi = (\Gamma, \Gamma_0, \Gamma_1, \mathscr{E}, \mathscr{M}_1, \ldots, \mathscr{M}_q, \mathscr{R}, i_{\text{out}})$ is $(\mathscr{M}_1, \ldots, \mathscr{M}_q, \emptyset)$.

An evolutional symport rule $[u]_i[\ ]_j \rightarrow [\ ]_i[u']_j$ is enabled at a configuration $\mathscr{C}_t$ at an instant t if there is a region i from $\mathscr{C}_t$ which contains multiset u. By applying an evolutional symport rule, the multiset of objects u in region i from $\mathscr{C}_t$ is consumed and the multiset of objects u' is generated in region j from $\mathscr{C}_{t+1}$. An evolutional antiport rule $[u]_i[v]_j \rightarrow [v']_i[u']_j$ is enabled at a configuration $\mathscr{C}_t$ at an instant t if there is a region i from $\mathscr{C}_t$ which contains multiset of objects u and a region j from $\mathscr{C}_t$ which contains multiset of objects v. By applying an evolutional antiport rule, (a) the multiset of objects u in region i from $\mathscr{C}_t$ and the multiset of objects v in region j from $\mathscr{C}_t$ are consumed; (b) the multiset of objects u' is generated in region j from configuration $\mathscr{C}_{t+1}$; and (c) the multiset of objects v' is generated in region i from configuration $\mathscr{C}_{t+1}$. The length of an evolutional symport/antiport rule r is an ordered pair of natural numbers: $\text{length}(r) = (\text{length}(\text{LHS}(r)),\ \text{length}(\text{RHS}(r)))$.

A separation rule $[a]_i \rightarrow [\Gamma_0]_i[\Gamma_1]_i$ is enabled at a configuration $\mathscr{C}_t$ at an instant t if there is a cell i from $\mathscr{C}_t$ which contains object a and $i \neq i_{\text{out}}$. By applying a separation rule to a such a cell i, (a) object a is consumed from such cell; (b) two new cells with label i are generated at configuration $\mathscr{C}_{t+1}$; (c) in the original cell i, the objects from Γ_0 are placed in one of the new cells, while the other objects from Γ_1 are placed in another one.

The rules of a TESAS P system are applied in a maximally parallel manner, and we have the restriction that when a cell i is separated at one transition step, no other rules can be applied for that cell i at that step.

A transition from a configuration $\mathscr{C}_t$ to another configuration $\mathscr{C}_{t+1}$ is obtained by applying rules in a maximally parallel manner following the previous remarks. A *computation* of the system is a (finite or infinite) sequence of transitions starting from the initial configuration, where any term of the sequence other than the first is obtained from the previous configuration in one transition step. If the sequence is finite (called *halting computation*) then the last term of the sequence is a *halting configuration*, that is, a configuration where no rule is applicable to it. A computation gives a result only when an halting configuration is reached, and that result is encoded by the multiset of objects present in the output region i_{out}.

A natural framework to solve decision problems is to use recognizer P systems; one can refer to [21, 29] for further details.

Definition 2. A recognizer tissue P system with evolutional symport/antiport rules and cell separation of degree $q \geq 1$ is a tuple

$$\Pi = \left(\Gamma, \Gamma_0, \Gamma_1, \mathscr{E}, \Sigma, \mathscr{M}_1, \ldots, \mathscr{M}_q, \mathscr{R}, i_{\text{in}}, i_{\text{out}}\right), \tag{2}$$

where

(i) the tuple $(\Gamma, \Gamma_0, \Gamma_1, \mathscr{E}, \mathscr{M}_1, \ldots, \mathscr{M}_q, \mathscr{R}, i_{\text{out}})$ is a TESAS P system of degree $q \geq 1$, where Γ strictly contains an (input) alphabet Σ and two distinguished objects yes, no, and $\mathscr{M}_i$ $(1 \leq i \leq q)$ are multisets over $\Gamma \setminus \Sigma$;

(ii) $i_{\text{in}} \in \{1, \ldots, q\}$ is the input cell and i_{out} is the label of the environment;

(iii) for each multiset m over the input alphabet Σ, any computation of the system Π with input m starts from the configuration of the form $(\mathscr{M}_1, \ldots, \mathscr{M}_{i_{\text{in}}} + m, \ldots, \mathscr{M}_q; \emptyset)$ and *always halts* and either object yes or object no (but not both) must appear in the environment at the last step.

It is worth pointing out that, in any recognizer TESAS P systems, all computations halt. Then, any symport rule of the type $[u]_0[\ ]_j \to [\ ]_0[u']_j$ must verify the following condition: multiset u contains some object from $\Gamma \setminus \mathscr{E}$.

For each ordered pair of natural numbers (k_1, k_2) greater than or equal to 1, the class of recognizer TESAS P systems with cell separation and with evolutional communication rules of length at most (k_1, k_2) is denoted by $\mathbf{TSEC}(k_1, k_2)$. This means that the LHS (resp., RHS) of any evolutional communication rule in a system from $\mathbf{TSEC}(k_1, k_2)$ involves at most k_1 objects (resp., k_2 objects).

Next, we define the concept of solving a problem in a uniform way and in polynomial time by a family of recognizer TESAS P systems (see [40] for details).

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

(i) the family $\mathbf{\Pi}$ is polynomially uniform by Turing machines;

(ii) there exists a polynomial encoding (cod, s) of I_X in $\mathbf{\Pi}$ such that (a) for each instance $u \in I_X$, $s(u)$ is a natural number and $\text{cod}(u)$ is an input multiset of the system $\Pi(s(u))$; (b) for each $n \in \mathbb{N}$, $s^{-1}(n)$ is a finite set; and (c) the family $\mathbf{\Pi}$ is polynomially bounded, sound, and complete with regard to (X, cod, s).

The set of all decision problems that can be solved by recognizer TESAS P systems with evolutional communication rules of length at most (k_1, k_2) in a uniform way and polynomial time is denoted by $\mathbf{PMC}_{\mathbf{TSEC}(k_1,k_2)}$.

3. The Computational Complexity of Tissue P Systems with Evolutional Symport/Antiport Rules

3.1. The Limitation on the Efficiency of $\mathbf{TSEC}(2, 1)$. In this subsection, we use tissue P systems with cell separation and evolutional communication rules of length at most $(2, 1)$ to provide a new characterization of the classical complexity class **P**.

The proof uses a similar technique as in [32]. Firstly, some representations of TESAS P systems $\Pi = (\Gamma, \Gamma_0, \Gamma_1, \mathscr{E}, \Sigma, \mathscr{M}_1, \ldots, \mathscr{M}_q, \mathscr{R}, i_{\text{in}}, i_{\text{out}})$ from $\mathbf{TSEC}(2, 1)$ are given. By $\mathscr{R}_C$ (resp., $\mathscr{R}_S$) we denote the set of communication rules (resp., separation rules) of Π. We will fix a total order in $\mathscr{R}_C$ and a total order in $\mathscr{R}_S$. Because several cells with the same label are generated by using separation rules, in order to identify the different cells with the same label, the following recursive definition is used to modify the labels of the new generated cells:

(i) We denote the label of a cell as a pair (i, σ), where $1 \leq i \leq q$ and $\sigma \in \{0, 1\}$ is a binary string.

(ii) If a separation rule is applied to a cell with label (i, σ), then the new created two cells will be labelled by $(i, \sigma 0)$ and $(i, \sigma 1)$, respectively. We mention that, for the system during any computation, we consider a lexicographical order over the set of labels of cells.

Note that if communication rules occur in two cells, then the labels of these two cells do not change.

A configuration at an instant t of a tissue P system from $\mathbf{TSEC}(2, 1)$ is described by the multisets of objects over Γ contained in each cell and the multiset of objects over $\Gamma \setminus \mathscr{E}$ in the environment. Hence, a configuration of Π can be described as follows:

$$\begin{aligned} &\{(a, i, \sigma) \mid a \in \Gamma \cup \{\lambda\},\ 1 \leq i \leq q,\ \sigma \in \{0, 1\}^*\} \\ &\quad \cup \{(a, 0) \mid a \in \Gamma \setminus \mathscr{E}\}. \end{aligned} \tag{3}$$

Let $r = [ab]_i[\ ]_j \to [\ ]_i[c]_j$, $1 \leq i, j \leq q$, be an evolutional symport rule of Π and $n \in \mathbb{N}$. We denote by $n \cdot \text{LHS}(r, (i, \sigma_i), (j, \sigma_j))$ the multiset of objects $(a, i, \sigma_i)^n (b, i, \sigma_i)^n$ and the corresponding $n \cdot \text{RHS}(r, (i, \sigma_i), (j, \sigma_j))$ the multiset of objects $(c, j, \sigma_j)^n$. In a similar way, $n \cdot \text{LHS}(r, (i, \sigma_i), (j, \sigma_j))$ and $n \cdot \text{RHS}(r, (i, \sigma_i), (j, \sigma_j))$ are defined when r is of the form $[a]_i[\ ]_j \to [\ ]_i[c]_j$.

Let $r = [ab]_i[\ ]_j \to [\ ]_i[\lambda]_j$, $1 \leq i, j \leq q$, be an evolutional symport rule of Π and $n \in \mathbb{N}$. We denote by $n \cdot \text{LHS}(r, (i, \sigma_i), (j, \sigma_j))$ the multiset of objects $(a, i, \sigma_i)^n (b, i, \sigma_i)^n$, and the corresponding $n \cdot \text{RHS}(r, (i, \sigma_i), (j, \sigma_j))$ is $\emptyset$. In a similar way, $n \cdot \text{LHS}(r, (i, \sigma_i), (j, \sigma_j))$ and $n \cdot \text{RHS}(r, (i, \sigma_i), (j, \sigma_j))$ are defined when r is of the form $[a]_i[\ ]_j \to [\ ]_i[\lambda]_j$.

Let $r = [ab]_0[\ ]_j \to [\ ]_0[c]_j$, $1 \leq j \leq q$, be an evolutional symport rule of Π and $n \in \mathbb{N}$. We denote by $n \cdot \text{LHS}(r, (j, \sigma_j), 0)$ the multiset of objects

$$\begin{aligned} &(a, 0)^n (b, 0)^n, \quad \text{if } a, b \in \Gamma \setminus \mathscr{E}; \\ &(a, 0)^n, \quad \text{if } a \in \Gamma \setminus \mathscr{E},\ b \in \mathscr{E}; \\ &(b, 0)^n, \quad \text{if } b \in \Gamma \setminus \mathscr{E},\ a \in \mathscr{E}; \\ &\emptyset, \quad \text{otherwise}; \end{aligned} \tag{4}$$

and we denote by the corresponding $n \cdot \text{RHS}(r, (j, \sigma_j), 0)$ the multiset of objects $(c, j, \sigma_j)^n$. In a similar way, $n \cdot \text{LHS}(r, (j, \sigma_j), 0)$ and $n \cdot \text{RHS}(r, (j, \sigma_j), 0)$ are defined when r is of the form $[a]_0[\ ]_j \to [\ ]_0[\lambda]_j$.

Let $r = [ab]_0[\ ]_j \to [\ ]_0[\lambda]_j$, $1 \le j \le q$, be an evolutional symport rule of Π and $n \in \mathbb{N}$. We denote by $n{\cdot}\mathrm{LHS}(r, (j, \sigma_j), 0)$ the multiset of objects

$$\begin{cases} (a,0)^n (b,0)^n, & \text{if } a, b \in \Gamma \setminus \mathscr{E}; \\ (a,0)^n, & \text{if } a \in \Gamma \setminus \mathscr{E},\ b \in \mathscr{E}; \\ (b,0)^n, & \text{if } b \in \Gamma \setminus \mathscr{E},\ a \in \mathscr{E}; \\ \emptyset, & \text{otherwise.} \end{cases} \tag{5}$$

The corresponding $n \cdot \mathrm{RHS}(r, (j, \sigma_j), 0)$ is $\emptyset$. In a similar way, $n \cdot \mathrm{LHS}(r, (j, \sigma_j), 0)$ and $n \cdot \mathrm{RHS}(r, (j, \sigma_j), 0)$ are defined when r is of the form $[a]_0[\ ]_j \to [\ ]_0[\lambda]_j$.

Let $r = [a]_i[b]_j \to [c]_i[\lambda]_j$, $1 \le i, j \le q$, be an evolutional antiport rule of Π and $n \in \mathbb{N}$. We denote by $n \cdot \mathrm{LHS}(r, (i, \sigma_i), (j, \sigma_j))$ the multiset of objects $(a, i, \sigma_i)^n (b, j, \sigma_j)^n$ and the corresponding $n \cdot \mathrm{RHS}(r, (i, \sigma_i), (j, \sigma_j))$ the multiset of objects $(c, i, \sigma_i)^n$. In a similar way, $n \cdot \mathrm{LHS}(r, (i, \sigma_i), (j, \sigma_j))$ and $n \cdot \mathrm{RHS}(r, (i, \sigma_i), (j, \sigma_j))$ are defined when r is of the form $[a]_i[b]_j \to [\lambda]_i[c]_j$.

Let $r = [a]_i[b]_j \to [\lambda]_i[\lambda]_j$, $1 \le i, j \le q$, be an evolutional antiport rule of Π and $n \in \mathbb{N}$. We denote by $n \cdot \mathrm{LHS}(r, (i, \sigma_i), (j, \sigma_j))$ the multiset of objects $(a, i, \sigma_i)^n (b, j, \sigma_j)^n$, and the corresponding $n \cdot \mathrm{RHS}(r, (i, \sigma_i), (j, \sigma_j))$ is $\emptyset$.

Let $r = [a]_0[b]_j \to [c]_0[\lambda]_j$, $1 \le j \le q$, be an evolutional antiport rule of Π and $n \in \mathbb{N}$. We denote by $n{\cdot}\mathrm{LHS}(r, (j, \sigma_j), 0)$ the multiset of objects

$$\begin{cases} (a,0)^n (b,j)^n, & \text{if } a \in \Gamma \setminus \mathscr{E}; \\ (b,j)^n, & \text{otherwise;} \end{cases} \tag{6}$$

and we denote by the corresponding $n \cdot \mathrm{RHS}(r, (j, \sigma_j), 0)$ the multiset of objects

$$\begin{cases} (c,0)^n, & \text{if } c \in \Gamma \setminus \mathscr{E}; \\ \emptyset, & \text{otherwise.} \end{cases} \tag{7}$$

Let $r = [a]_0[b]_j \to [\lambda]_0[c]_j$, $1 \le j \le q$, be an evolutional antiport rule of Π and $n \in \mathbb{N}$. We denote by $n{\cdot}\mathrm{LHS}(r, (j, \sigma_j), 0)$ the multiset of objects

$$\begin{cases} (a,0)^n (b,j)^n, & \text{if } a \in \Gamma \setminus \mathscr{E}; \\ (b,j)^n, & \text{otherwise;} \end{cases} \tag{8}$$

and we denote by the corresponding $n \cdot \mathrm{RHS}(r, (j, \sigma_j), 0)$ the multiset of objects $(c, j, \sigma_j)^n$.

Let $r = [a]_0[b]_j \to [\lambda]_0[\lambda]_j$, $1 \le j \le q$, be an evolutional antiport rule of Π and $n \in \mathbb{N}$. We denote by $n{\cdot}\mathrm{LHS}(r, (j, \sigma_j), 0)$ the multiset of objects

$$\begin{cases} (a,0)^n (b,j)^n, & \text{if } a \in \Gamma \setminus \mathscr{E}; \\ (b,j)^n, & \text{otherwise.} \end{cases} \tag{9}$$

The corresponding $n \cdot \mathrm{RHS}(r, (j, \sigma_j), 0)$ is $\emptyset$.

If $\mathscr{C}_t$ is a configuration of Π, then the multiset obtained by replacing in $\mathscr{C}_t$ every occurrence of (x, i, σ) by (x, i, σ') is denoted by $\mathscr{C}_t + \{(x, i, \sigma)/\sigma'\}$. Moreover, we denote by $\mathscr{C}_t + m$ (resp., $\mathscr{C}_t \setminus m$) a multiset m of labelled objects added to (resp., removed from) the configuration $\mathscr{C}_t$.

Next, we show that TESAS P systems from **TSEC**(2, 1) can only solve tractable problems.

If $\mathscr{C} = \{\mathscr{C}_t\}_{t<r+1}$ of Π ($r \in \mathbb{N}$) is a halting computation, then we denote by $|\mathscr{C}| = r$ the length of $\mathscr{C}$. For each i ($1 \le i \le q$), the multiset of objects over Γ contained in all cells labelled by i at configuration $\mathscr{C}_t$ is denoted by $\mathscr{C}_t(i)$. We denote by $\mathscr{C}_t(0)$ the multiset of objects over $\Gamma \setminus \mathscr{E}$ contained in the environment at configuration $\mathscr{C}_t$. Finally, the finite multiset $\mathscr{C}_t(0) + \mathscr{C}_t(1) + \cdots + \mathscr{C}_t(q)$ is denoted by $\mathscr{C}_t^*$.

Lemma 4. *Let $\Pi = (\Gamma, \Gamma_0, \Gamma_1, \mathscr{E}, \Sigma, \mathscr{M}_1, \ldots, \mathscr{M}_q, \mathscr{R}, i_{in}, i_{out})$ be a recognizer tissue P system of degree $q \ge 1$ from* **TSEC**(2, 1). *Let $M = |\mathscr{M}_1 + \cdots + \mathscr{M}_q|$ and let $\mathscr{C} = \{\mathscr{C}_0, \ldots, \mathscr{C}_r\}$ be a computation of Π. Then, one has*

(1) *$|\mathscr{C}_0^*| = M$, and for each t, $0 \le t < r$, $|\mathscr{C}_{t+1}^*| \le |\mathscr{C}_t^*|$;*

(2) *for each t, $0 \le t \le r$, $|\mathscr{C}_t^*| \le M$;*

(3) *the number of created cells along the computation $\mathscr{C}$ by the application of cell separation rules is bounded by $2M$.*

Proof. (1) Let us notice that

$$\begin{aligned} |\mathscr{C}_0^*| &= |\mathscr{C}_0(0) + \mathscr{C}_0(1) + \cdots + \mathscr{C}_0(q)| \\ &= |\mathscr{M}_1 + \cdots + \mathscr{M}_q| = M. \end{aligned} \tag{10}$$

Let Π be a recognizer tissue P system from **TSEC**(2, 1) and let $\mathscr{R}$ be the set of rules associated with Π, which contains the following types of evolutional communication rules:

(a) $[ab]_i[\ ]_j \to [\ ]_i[c]_j$, $(i = 0 \wedge \{a, b\} \not\subseteq \mathscr{E})$ or $(i \ne 0 \wedge \{a, b\} \subseteq \Gamma)$;

(b) $[ab]_i[\ ]_j \to [\ ]_i[\lambda]_j$, $(i = 0 \wedge \{a, b\} \not\subseteq \mathscr{E})$ or $(i \ne 0 \wedge \{a, b\} \subseteq \Gamma)$;

(c) $[a]_i[b]_j \to [c]_i[\lambda]_j$, $\{a, b\} \subseteq \Gamma$;

(d) $[a]_i[b]_j \to [\lambda]_i[c]_j$, $\{a, b\} \subseteq \Gamma$;

(e) $[a]_i[b]_j \to [\lambda]_i[\lambda]_j$, $\{a, b\} \subseteq \Gamma$;

(f) $[a]_i[\ ]_j \to [\ ]_i[b]_j$, $(i = 0 \wedge a \in \Gamma \setminus \mathscr{E})$ or $(i \ne 0 \wedge a \in \Gamma)$;

(g) $[a]_i[\ ]_j \to [\ ]_i[\lambda]_j$, $(i = 0 \wedge a \in \Gamma \setminus \mathscr{E})$ or $(i \ne 0 \wedge a \in \Gamma)$.

For each t, $0 \le t < r$, in the transition from configuration $\mathscr{C}_t$ to configuration $\mathscr{C}_{t+1}$, by using any rules of types from (a) to (g), at least one object from $\mathscr{C}_t$ is consumed and at most one object is produced in $\mathscr{C}_{t+1}$. Hence, in any transition step the number of objects in the system is not increased.

(2) By induction on t, let us start analyzing the basic case $t = 0$. The result is trivial because of $|\mathscr{C}_0^*| = M$. By induction hypothesis, let us suppose the result holds for t, $0 \le t < r$. Then,

$$|\mathscr{C}_{t+1}^*| \overset{(1)}{\le} |\mathscr{C}_t^*| \overset{\text{h.i.}}{\le} M. \tag{11}$$

Hence, the result is also true for $t + 1$.

Input: A P system Π from **TSEC**$(2, 1)$ and an input multiset m
  Initialization phase: $\mathscr{C}_0$ is the initial configuration of $\Pi + m$
  $t \leftarrow 0$
  while $\mathscr{C}_t$ is a non-halting configuration **do**
    Selection phase: Input $\mathscr{C}_t$, Output $(\mathscr{C}'_t, A)$
    Execution phase: Input $(\mathscr{C}'_t, A)$, Output $\mathscr{C}_{t+1}$
    $t \leftarrow t + 1$
  end while
Output: *Yes* if $\Pi + m$ has an accepting computation, *No* otherwise

Pseudocode 1

(3) According to the fact that the application of a cell separation rule consumes an object and produces two new cells, result (3) can be obtained from (2) easily. □

Next, a deterministic algorithm $\mathscr{A}$ working in polynomial time is presented, which receives as input a P system Π from **TSEC**$(2, 1)$ and an input multiset m of Π, in such manner that algorithm $\mathscr{A}$ reproduces the behaviour of a computation of $\Pi + m$. If the system Π is confluent, then the algorithm $\mathscr{A}$ will provide the same answer of Π. We give Pseudocode 1 of the algorithm $\mathscr{A}$ to describe the simulation process.

The algorithm $\mathscr{A}$ receives a recognizer tissue P system

$$\Pi = \left(\Gamma, \Gamma_0, \Gamma_1, \mathscr{E}, \Sigma, \mathscr{M}_1, \ldots, \mathscr{M}_q, \mathscr{R}, i_{\text{in}}, i_{\text{out}}\right) \quad (12)$$

from **TSEC**$(2, 1)$ and an input multiset m. Let $M = |\mathscr{M}_1 + \cdots + \mathscr{M}_q|$. Let any computation of Π perform at most p (p is a natural number) transition steps. Hence, from Lemma 4, the number of cells in the system along any computation is bounded by $2M + q$.

A transition of a recognizer tissue P system $\Pi + m$ is performed in two phases: selection phase and execution phase (see Algorithms 1 and 2). For the detailed information of a transition of such P system, one can refer to [32].

It is easy to check that Algorithm 1 is deterministic and the running time of this algorithm is polynomial in the size of Π because the number of cycles of the first main loop **for** is of order $O(|\mathscr{R}| \cdot M^2 \cdot q^2)$; the number of cycles of the second main loop **for** is of order $O(|\mathscr{R}| \cdot M \cdot q)$; and the number of cycles of the third main loop **for** is of order $O(|\mathscr{R}| \cdot M \cdot q \cdot |\Gamma|)$.

Algorithm 2 is deterministic and the running time of this algorithm is polynomial in the size of Π because the number of cycles of the first main loop **for** is of order $O(|\mathscr{R}| \cdot M^2 \cdot q^2)$; the number of cycles of the second main loop **for** is of order $O(|\mathscr{R}| \cdot M \cdot q)$; and the number of cycles of the third main loop **for** is of order $O(|\mathscr{R}| \cdot M \cdot q \cdot |\Gamma|)$.

Theorem 5. *One has* $\mathbf{P} = \mathbf{PMC}_{\mathbf{TSEC}(2,1)}$.

Proof. Because $\mathbf{PMC}_{\mathbf{TSEC}(2,1)}$ is closed under polynomial time reduction and nonempty, hence $\mathbf{P} \subseteq \mathbf{PMC}_{\mathbf{TSEC}(2,1)}$. In what follows, we show that $\mathbf{C}_{\mathbf{TSEC}(2,1)} \subseteq \mathbf{P}$. Let $X \in \mathbf{PMC}_{\mathbf{TSEC}(2,1)}$ and let $\boldsymbol{\Pi} = \{\Pi(n) \mid n \in \mathbb{N}\}$ be a family of recognizer tissue P systems from **TSEC**$(2, 1)$ solving X according to Definition 3. Let (cod, s) be a polynomial encoding associated with that solution. If $u \in I_X$ is an instance of the problem X, then u will be processed by the system $\Pi(s(u)) + \text{cod}(u)$. We consider the deterministic algorithm $\mathscr{A}'$ as shown in Algorithm 3.

The algorithm $\mathscr{A}'$ receives an instance u of the decision problem $X = (I_X, \theta_X)$, working in a polynomial time. The following three assertions are equivalent:

(i) $\theta_X(u) = 1$; that is, the answer of problem X to instance u is affirmative.

(ii) Every computation of $\Pi(s(u)) + \text{cod}(u)$ is an accepting computation.

(iii) The output of the algorithm $\mathscr{A}'$ with input u is Yes.

Hence, $X \in \mathbf{P}$. □

Corollary 6. *For each* $n \in \mathbb{N}$, $n \geq 1$, *one has* $\mathbf{PMC}_{\mathbf{TSEC}(n,1)} = \mathbf{P}$.

Proof. Indeed, it suffices to notice that, at any transition step, the application of each rule consumes at least one object from $\Gamma \setminus \mathscr{E}$ and produces at most one object from Γ. Thus, along any computation of the system, the total number in it is not increased. □

Theorem 7. *For each* $n \in \mathbb{N}$, $n \geq 1$, *one has* $\mathbf{PMC}_{\mathbf{TSEC}(1,n)} = \mathbf{P}$.

Proof. In [31], it was shown that the class of decision problems solvable in polynomial time by means of families of tissue P systems with cell division and symport rules with length 1 is equal to class **P**. Bearing in mind that systems from **TSEC**$(1, n)$ are noncooperative ones, the dependency graph technique used in the cited paper can be used to obtain the result, in a similar way. □

3.2. An Efficient Solution to the SAT Problem by P Systems in **TSEC**$(3, 2)$. The SAT problem is a well-known **NP**-complete problem; here we give an efficient solution to the SAT problem by a family of tissue P systems with evolutional communication rules of length at most $(3, 2)$.

Theorem 8. *One has* $\text{SAT} \in \mathbf{PMC}_{\mathbf{TSEC}(3,2)}$.

Proof. We provide a polynomial time uniform solution to the SAT problem [41] by a family of recognizer tissue P systems $\boldsymbol{\Pi} = \{\Pi(t) \mid t \in \mathbb{N}\}$ from **TSEC**$(3, 2)$. Each system $\Pi(t)$ ($t = \langle n, m\rangle = ((n + m)(n + m + 1)/2) + n$) can process all Boolean formula φ in conjunctive normal form with n variables and m clauses.

Input: A configuration $\mathscr{C}_t$ of $\Pi + m$ at instant t
  $\mathscr{C}'_t \leftarrow \mathscr{C}_t$; $A \leftarrow \emptyset$; $B \leftarrow \emptyset$
  for $r = [u]_i[v]_j \rightarrow [v']_i[u']_j \in \mathscr{R}_C$, according to the order chosen **do**
    for each pair of cells (i, σ_i), (j, σ_j) of $\mathscr{C}'_t$ according to the lexicographical order **do**
      $n_r \leftarrow$ maximum number of times that r is applicable to (i, σ_i), (j, σ_j)
      if $n_r > 0$ **then**
        $\mathscr{C}'_t \leftarrow \mathscr{C}'_t - n_r \cdot \mathrm{LHS}(r, (i, \sigma_i), (j, \sigma_j))$
        $A \leftarrow A \cup \{(r, n_r, (i, \sigma_i), (j, \sigma_j))\}$
        $B \leftarrow B \cup \{(i, \sigma_i), (j, \sigma_j)\}$
      end if
    end for
  end for
  for $r = [u]_i[v]_0 \rightarrow [v']_i[u']_0 \in \mathscr{R}_C$ according to the order chosen **do**
    for each cell (i, σ_i) of $\mathscr{C}'_t$ according to the lexicographical order **do**
      $n_r \leftarrow$ maximum number of times that r is applicable to (i, σ_i)
      if $n_r > 0$ **then**
        $\mathscr{C}'_t \leftarrow \mathscr{C}'_t - n_r \cdot \mathrm{LHS}(r, (i, \sigma_i), 0)$
        $A \leftarrow A \cup \{(r, n_r, (i, \sigma_i), 0)\}$
        $B \leftarrow B \cup \{(i, \sigma_i)\}$
      end if
    end for
  end for
  for $r = [a]_i \rightarrow [\Gamma_0]_i[\Gamma_1]_i \in \mathscr{R}_S$ according to the order chosen **do**
    for each $(a, i, \sigma_i) \in \mathscr{C}'_t$ according to the lexicographical order, and such that $(i, \sigma_i) \notin B$ **do**
      $\mathscr{C}'_t \leftarrow \mathscr{C}'_t \setminus \{(a, i, \sigma_i)\}$
      $A \leftarrow A \cup \{(r, 1, (i, \sigma_i))\}$
    end for
  end for

Algorithm 1: Selection phase.

For each $n, m \in \mathbb{N}$, we consider the recognizer tissue P system

$$\Pi(\langle n, m\rangle) = \left(\Gamma, \Gamma_0, \Gamma_1, \mathscr{E}, \Sigma, \mathscr{M}_1, \mathscr{M}_2, \mathscr{M}_3, \mathscr{R}, i_{\text{in}}, i_{\text{out}}\right), \tag{13}$$

where

$$\begin{aligned}
\Gamma = \Sigma \cup \mathscr{E} \cup \Big\{ & A_i, A'_i, \overline{A}_i, \overline{A}'_i, \widehat{A}_i, \widehat{A}'_i, B_i, B'_i, \overline{B}_i, \overline{B}'_i, \widehat{B}_i, \widehat{B}'_i, \overline{C}_i, \\
& T_i, \overline{T}_i, T'_i, F_i, \overline{F}_i, F'_i, a_i, a'_i, b_i, b'_i, c_i, t_i, f_i, h_i, y_i, s_i, v_i, w_i, z_i \mid 1 \\
& \le i \le n\Big\} \cup \left\{\overline{D}_{i,j}, q_{i,j}, r_{i,j}, u_{i,j} \mid 1 \le i, j \le n\right\} \\
& \cup \left\{\overline{D}_{i,n+1} \mid 1 \le i \le n+1\right\} \cup \left\{g_{i,j,k}, \overline{g}_{i,j,k} \mid 1 \le i, k \right. \\
& \left. \le n,\ 1 \le j \le m\right\} \cup \left\{E_{i,j}, \overline{E}_{i,j}, e_{i,j}, \overline{e}_{i,j}, l_{j,i} \mid 1 \le i \le n, \right. \\
& \left. 1 \le j \le m\right\} \cup \left\{l_j, \bar{l}_j, l_{j,0}, E_j \mid 1 \le j \le m\right\} \cup \left\{A_{n+1}, \right. \\
& \left. \overline{A}_{n+1}, A'_{n+1}, B_{n+1}, \overline{B}_{n+1}, B'_{n+1}, \overline{C}_{n+1}, E_{m+1}, S, p, \mathtt{yes}, \mathtt{no}\right\},
\end{aligned}$$

$$\Gamma_1 = \left\{T'_i, F'_i \mid 1 \le i \le n\right\} \cup \left\{A'_i, B'_i \mid 2 \le i \le n+1\right\},$$

$$\Gamma_0 = \Gamma \setminus \Gamma_1,$$

$$\Sigma = \left\{x_{i,j}, \overline{x}_{i,j} \mid 1 \le i \le n,\ 1 \le j \le m\right\},$$

$$\begin{aligned}
\mathscr{E} = \Big\{ & \widetilde{A}_i, \widetilde{B}_i, \widetilde{a}_i, \widetilde{a}_i, \widetilde{b}_i, \widetilde{b}_i, \widetilde{c}_i, \widetilde{c}_i, \widetilde{d}_i, \widetilde{f}_i, \widetilde{s}_i, \widetilde{t}_i, \widetilde{v}_i, \widetilde{v}_{i,n+1}, \widetilde{w}_i, \widetilde{y}_i, \widetilde{y}_i, \\
& \widetilde{z}_i \mid 1 \le i \le n\Big\} \cup \left\{\widetilde{r}_{i,j}, \widetilde{r}_{i,j}, \widetilde{u}_{i,j}, \widetilde{v}_{i,j} \mid 1 \le i, j \le n\right\} \\
& \cup \left\{g'_{i,j,k}, \overline{g}'_{i,j,k} \mid 1 \le i \le n,\ 1 \le j \le m,\ 0 \le k \le n\right\} \\
& \cup \left\{\bar{l}_{j,k} \mid 1 \le j \le m,\ 0 \le k \le n\right\} \cup \left\{\alpha_i \mid 0 \le i \le 4n \right. \\
& \left. + 3m + 4\right\} \cup \left\{E_0\right\},
\end{aligned}$$

$$\mathscr{M}_1 = \left\{A_1, B_1\right\},$$

$$\mathscr{M}_2 = \left\{\overline{A}_1, \overline{B}_1, \overline{C}_1, \overline{D}_{1,1}\right\},$$

$$\mathscr{M}_3 = \left\{l_{1,0}, \ldots, l_{m,0}, \alpha_0, p\right\},$$

$i_{\text{in}} = 3$ is the input cell,

$i_{\text{out}} = 0$ is the output region, (14)

Input: The output $(\mathscr{C}'_t, A)$ of the selection phase
  for each $(r, n_r, (i, \sigma_i), (j, \sigma_j)) \in A$ **do**
    $\mathscr{C}'_t \leftarrow \mathscr{C}'_t + n_r \cdot \mathrm{RHS}(r, (i, \sigma_i), (j, \sigma_j))$
  end for
  for each $(r, n_r, (i, \sigma_i), 0) \in A$ **do**
    $\mathscr{C}'_t \leftarrow \mathscr{C}'_t + n_r \cdot \mathrm{RHS}(r, (i, \sigma_i), 0)$
  end for
  for each $(r, 1, (i, \sigma_i)) \in A$ **do**
    $\mathscr{C}'_t \leftarrow \mathscr{C}'_t + \{(\lambda, i, \sigma_i)/\sigma_i 0\}$
    $\mathscr{C}'_t \leftarrow \mathscr{C}'_t + \{(\lambda, i, \sigma_i 1)\}$
    for each $(x, i, \sigma_i) \in \mathscr{C}'_t$ according to the lexicographical order **do**
      if $x \in \Gamma_0$ **then**
        $\mathscr{C}'_t \leftarrow \mathscr{C}'_t + \{(x, i, \sigma_i)/\sigma_i 0\}$
      else
        $\mathscr{C}'_t \leftarrow \mathscr{C}'_t + \{(x, i, \sigma_i)/\sigma_i 1\}$
      end if
    end for
  end for
  $\mathscr{C}_{t+1} \leftarrow \mathscr{C}'_t$

Algorithm 2: Execution phase.

Input: an instance u of the problem X
  Construct the system $\Pi(s(u)) + \mathrm{cod}(u)$
  Run algorithm $\mathscr{A}$ with input $\Pi(s(u)) + \mathrm{cod}(u)$
Output: Yes if $\Pi(s(u)) + \mathrm{cod}(u)$ has an accepting computation
  No otherwise

Algorithm 3

and the set $\mathscr{R}$ of rules consists of the following rules:

$$r_{1,i} \equiv [A_i]_1 \left[\overline{A}_i\right]_2 \longrightarrow \left[\overline{A}_i\right]_1 \left[\widehat{A}_i\right]_2, \quad 1 \le i \le n,$$

$$r_{2,i} \equiv \left[A'_i\right]_1 \left[\overline{A}_i\right]_2 \longrightarrow \left[\overline{A}'_i\right]_1 \left[\widehat{A}'_i\right]_2, \quad 1 \le i \le n,$$

$$r_{3,i} \equiv [B_i]_1 \left[\overline{B}_i\right]_2 \longrightarrow \left[\overline{B}_i\right]_1 \left[\widehat{B}_i\right]_2, \quad 1 \le i \le n,$$

$$r_{4,i} \equiv \left[B'_i\right]_1 \left[\overline{B}_i\right]_2 \longrightarrow \left[\overline{B}'_i\right]_1 \left[\widehat{B}'_i\right]_2, \quad 1 \le i \le n,$$

$$r_{5,i} \equiv \left[\overline{C}_i\right]_2 [\overline{v}_i]_0 \longrightarrow [v_i]_2 [\ ]_0, \quad 1 \le i \le n,$$

$$r_{6,i,j} \equiv \left[\overline{D}_{i,j}\right]_2 \left[\overline{v}_{i,j}\right]_0 \longrightarrow \left[q_{i,j}\right]_2 [\ ]_0, \quad 1 \le i, j \le n,$$

$$r_{7,i} \equiv [T_i]_1 [t_i]_2 \longrightarrow [t_i]_1 [\ ]_2, \quad 1 \le i \le n,$$

$$r_{8,i} \equiv \left[T'_i\right]_1 [t_i]_2 \longrightarrow [t_i]_1 [\ ]_2, \quad 1 \le i \le n,$$

$$r_{9,i} \equiv [F_i]_1 [f_i]_2 \longrightarrow [f_i]_1 [\ ]_2, \quad 1 \le i \le n,$$

$$r_{10,i} \equiv \left[F'_i\right]_1 [f_i]_2 \longrightarrow [f_i]_1 [\ ]_2, \quad 1 \le i \le n,$$

$$r_{11,i} \equiv \left[\overline{A}_i\right]_1 \left[\widetilde{A}_i\right]_0 \longrightarrow \left[a_i a'_i\right]_1 [\ ]_0, \quad 1 \le i \le n,$$

$$r_{12,i} \equiv \left[\overline{A}'_i\right]_1 \left[\widetilde{A}_i\right]_0 \longrightarrow \left[a_i a'_i\right]_1 [\ ]_0, \quad 2 \le i \le n,$$

$$r_{13,i} \equiv \left[\overline{B}_i\right]_1 \left[\widetilde{B}_i\right]_0 \longrightarrow \left[b_i b'_i\right]_1 [\ ]_0, \quad 1 \le i \le n,$$

$$r_{14,i} \equiv \left[\overline{B}'_i\right]_1 \left[\widetilde{B}_i\right]_0 \longrightarrow \left[b_i b'_i\right]_1 [\ ]_0, \quad 2 \le i \le n,$$

$$r_{15,i} \equiv \left[\widehat{A}_i\right]_2 \left[\widetilde{A}_i\right]_0 \longrightarrow [A_i]_2 [\ ]_0, \quad 1 \le i \le n,$$

$$r_{16,i} \equiv \left[\widehat{A}'_i\right]_2 \left[\widetilde{A}_i\right]_0 \longrightarrow \left[A'_i\right]_2 [\ ]_0, \quad 2 \le i \le n,$$

$$r_{17,i} \equiv \left[\widehat{B}_i\right]_2 \left[\widetilde{B}_i\right]_0 \longrightarrow [B_i]_2 [\ ]_0, \quad 1 \le i \le n,$$

$$r_{18,i} \equiv \left[\widehat{B}'_i\right]_2 \left[\widetilde{B}_i\right]_0 \longrightarrow \left[B'_i\right]_2 [\ ]_0, \quad 2 \le i \le n,$$

$$r_{19,i} \equiv [v_i]_2 [\overline{y}_i]_0 \longrightarrow \left[y_i^2\right]_2 [\ ]_0, \quad 1 \le i \le n,$$

$$r_{20} \equiv [q_{1,1}]_2 [\overline{r}_{1,1}]_0 \longrightarrow [r_{1,1}]_2 [\ ]_0,$$

$$r_{21,i,j} \equiv \left[q_{i,j}\right]_2 \left[\overline{r}_{i,j}\right]_0 \longrightarrow \left[r_{i,j}^2\right]_2 [\ ]_0, \quad 1 \le i \le n,\ 2 \le j \le n,$$

$$r_{22,i} \equiv [t_i]_1 [\overline{t}_i]_0 \longrightarrow \left[T_i T'_i\right]_1 [\ ]_0, \quad 1 \le i \le n,$$

$$r_{23,i} \equiv [f_i]_1 \left[\overline{f}_i\right]_0 \longrightarrow \left[F_i F'_i\right]_1 [\ ]_0, \quad 1 \le i \le n,$$

$$r_{24,i} \equiv [A_i]_2 [\overline{c}_i]_0 \longrightarrow [c_i]_2 [\ ]_0, \quad 1 \le i \le n,$$

$$r_{25,i} \equiv \left[A'_i\right]_2 [\overline{c}_i]_0 \longrightarrow [c_i]_2 [\ ]_0, \quad 2 \le i \le n,$$

$$
\begin{aligned}
r_{26,i} &\equiv [B_i]_2\,[\overline{c}_i]_0 \longrightarrow [c_i]_2\,[\ ]_0, \quad 1 \le i \le n,\\
r_{27,i} &\equiv \left[B'_i\right]_2\,[\widetilde{c}_i]_0 \longrightarrow [c_i]_2\,[\ ]_0, \quad 2 \le i \le n,\\
r_{28,i} &\equiv [a_i]_1\,[\overline{a}_i]_0 \longrightarrow [T_i A_{i+1}]_1\,[\ ]_0, \quad 1 \le i \le n,\\
r_{29,i} &\equiv \left[a'_i\right]_1\,[\widetilde{a}_i]_0 \longrightarrow \left[F'_i A'_{i+1}\right]_1\,[\ ]_0, \quad 1 \le i \le n,\\
r_{30,i} &\equiv [b_i]_1\,[\overline{b}_i]_0 \longrightarrow [B_{i+1}S]_1\,[\ ]_0, \quad 1 \le i \le n,\\
r_{31,i} &\equiv \left[b'_i\right]_1\,[\widetilde{b}_i]_0 \longrightarrow \left[B'_{i+1}\right]_1\,[\ ]_0, \quad 1 \le i \le n,\\
r_{32,i} &\equiv [y_i]_2\,[\widetilde{y}_i]_0 \longrightarrow [z_i w_i]_2\,[\ ]_0, \quad 1 \le i \le n,\\
r_{33,i,j} &\equiv \left[r_{i,j}\right]_2\,[\widetilde{r}_{i,j}]_0 \longrightarrow \left[s_i u_{i,j}\right]_2\,[\ ]_0, \quad 1 \le i, j \le n,\\
r_{34,i} &\equiv [w_i]_2\,[\widetilde{w}_i]_0 \longrightarrow \left[\overline{A}_{i+1}\right]_2\,[\ ]_0, \quad 1 \le i \le n,\\
r_{35,i} &\equiv [c_i]_2\,[\widetilde{d}_i]_0 \longrightarrow \left[\overline{B}_{i+1}\right]_2\,[\ ]_0, \quad 1 \le i \le n,\\
r_{36,i} &\equiv [z_i]_2\,[\widetilde{z}_i]_0 \longrightarrow \left[\overline{C}_{i+1}\right]_2\,[\ ]_0, \quad 1 \le i \le n,\\
r_{37,i} &\equiv [s_i]_2\,[\widetilde{s}_i]_0 \longrightarrow [t_i f_i]_2\,[\ ]_0, \quad 1 \le i \le n,\\
r_{38,j} &\equiv \left[u_{1,j}\right]_2\,[\widetilde{u}_{1,j}]_0 \longrightarrow \left[\overline{D}_{1,j+1}\overline{D}_{2,j+1}\right]_2\,[\ ]_0, \quad 1 \le j \le n,\\
r_{39,i,j} &\equiv \left[u_{i,j}\right]_2\,[\widetilde{u}_{i,j}]_0 \longrightarrow \left[\overline{D}_{i+1,j+1}\right]_2\,[\ ]_0, \quad 2 \le i, j \le n,\\
r_{40} &\equiv [S]_1 \longrightarrow [\Gamma_0]_1\,[\Gamma_1]_1,\\
r_{41,i} &\equiv \left[\overline{D}_{i,n+1}\right]_2\,[\widetilde{v}_{i,n+1}]_0 \longrightarrow \left[h_i^2\right]_2\,[\ ]_0, \quad 1 \le i \le n,\\
r_{42} &\equiv [A_{n+1}]_1\,\left[\overline{A}_{n+1}\right]_2 \longrightarrow [E_1]_1\,[\ ]_2,\\
r_{43} &\equiv \left[A'_{n+1}\right]_1\,\left[\overline{B}_{n+1}\right]_2 \longrightarrow [E_1]_1\,[\ ]_2,\\
r_{44,i} &\equiv \left[T_i T'_i\right]_1\,[h_i]_2 \longrightarrow \left[\overline{T}_i\right]_1\,[\ ]_2, \quad 1 \le i \le n,\\
r_{45,i} &\equiv \left[F_i F'_i\right]_1\,[h_i]_2 \longrightarrow \left[\overline{F}_i\right]_1\,[\ ]_2, \quad 1 \le i \le n,\\
r_{46,i,j} &\equiv \left[x_{i,j}\right]_3\,\left[g'_{i,j,0}\right]_0 \longrightarrow \left[g^2_{i,j,1}\right]_3\,[\ ]_0, \quad 1 \le i \le n,\ 1 \le j \le m,\\
r_{47,i,j} &\equiv \left[\overline{x}_{i,j}\right]_3\,\left[\overline{g}'_{i,j,0}\right]_0 \longrightarrow \left[\overline{g}^2_{i,j,1}\right]_3\,[\ ]_0, \quad 1 \le i \le n,\ 1 \le j \le m,\\
r_{48,i,j,k} &\equiv \left[g_{i,j,k}\right]_3\,\left[g'_{i,j,k}\right]_0 \longrightarrow \left[g^2_{i,j,k+1}\right]_3\,[\ ]_0, \quad 1 \le i \le n,\ 1 \le j \le m,\ 1 \le k \le n-1,\\
r_{49,i,j,k} &\equiv \left[\overline{g}_{i,j,k}\right]_3\,\left[\overline{g}'_{i,j,k}\right]_0 \longrightarrow \left[\overline{g}^2_{i,j,k+1}\right]_3\,[\ ]_0, \quad 1 \le i \le n,\ 1 \le j \le m,\ 1 \le k \le n-1,\\
r_{50,i,j} &\equiv \left[g_{i,j,n}\right]_3\,\left[g'_{i,j,n}\right]_0 \longrightarrow \left[e_{i,j}\right]_3\,[\ ]_0, \quad 1 \le i \le n,\ 1 \le j \le m,\\
r_{51,i,j} &\equiv \left[\overline{g}_{i,j,n}\right]_3\,\left[\overline{g}'_{i,j,n}\right]_0 \longrightarrow \left[\overline{e}_{i,j}\right]_3\,[\ ]_0, \quad 1 \le i \le n,\ 1 \le j \le m,\\
r_{52,j,k} &\equiv \left[l_{j,k}\right]_3\,\left[\overline{l}_{j,k}\right]_0 \longrightarrow \left[l^2_{j,k+1}\right]_3\,[\ ]_0, \quad 1 \le j \le m,\ 0 \le k \le n-1,\\
r_{53,j} &\equiv \left[l_{j,n}\right]_3\,\left[\overline{l}_{j,n}\right]_0 \longrightarrow \left[l_j\right]_3\,[\ ]_0, \quad 1 \le j \le m,\\
r_{54,i,j} &\equiv \left[\overline{T}_i E_j\right]_1\,\left[e_{i,j}\right]_3 \longrightarrow \left[E_{i,j}\right]_1\,[\ ]_3, \quad 1 \le i \le n,\ 1 \le j \le m,\\
r_{55,i,j} &\equiv \left[\overline{F}_i E_j\right]_1\,\left[\overline{e}_{i,j}\right]_3 \longrightarrow \left[\overline{E}_{i,j}\right]_1\,[\ ]_3, \quad 1 \le i \le n,\ 1 \le j \le m,\\
r_{56,i,j} &\equiv \left[E_{i,j}\right]_1\,\left[l_j\right]_3 \longrightarrow \left[l_{i,j}\right]_1\,\left[E_{i,j}\right]_3, \quad 1 \le i \le n,\ 1 \le j \le m,\\
r_{57,i,j} &\equiv \left[\overline{E}_{i,j}\right]_1\,\left[l_j\right]_3 \longrightarrow \left[\overline{l}_{i,j}\right]_1\,\left[\overline{E}_{i,j}\right]_3, \quad 1 \le i \le n,\ 1 \le j \le m,\\
r_{58,i,j} &\equiv \left[l_{i,j}\right]_1\,\left[E_{i,j}\right]_3 \longrightarrow \left[\overline{T}_i E_{j+1}\right]_1\,[\ ]_3, \quad 1 \le i \le n,\ 1 \le j \le m,\\
r_{59,i,j} &\equiv \left[\overline{l}_{i,j}\right]_1\,\left[\overline{E}_{i,j}\right]_3 \longrightarrow \left[\overline{F}_i E_{j+1}\right]_1\,[\ ]_3, \quad 1 \le i \le n,\ 1 \le j \le m,\\
r_{60,i} &\equiv [\alpha_i]_3\,[\overline{\alpha}_i]_0 \longrightarrow [\alpha_{i+1}]_3\,[\ ]_0, \quad 0 \le i \le 4n+3m+3,\\
r_{61} &\equiv [E_{m+1}]_1\,[p]_3 \longrightarrow [\ ]_1\,[\mathtt{yes}]_3,\\
r_{62} &\equiv [\mathtt{yes}]_3\,[\ ]_0 \longrightarrow [\ ]_3\,[\mathtt{yes}]_0,\\
r_{63} &\equiv [\alpha_{4n+3m+4}p]_3\,[\ ]_0 \longrightarrow [\ ]_3\,[\mathtt{no}]_0.
\end{aligned}
\tag{15}
$$

We consider a propositional formula $\varphi = C_1 \wedge \cdots \wedge C_m$, which contains m ($m \ge 1$) clauses, $C_i = y_{i,1} \vee \cdots \vee y_{i,p_i}$, for $p_i \ge 1$, and $y_{i,j} \in \{x_k, \neg x_k \mid 1 \le k \le n\}$, for $1 \le i \le m$, $1 \le j \le p_i$.

Let (cod, s) be a polynomial encoding of instances from SAT in Π, where $s(\varphi) = \langle n, m\rangle$ and $\mathrm{cod}(\varphi) = \{x_{i,j} \mid x_i \in C_j\} \cup \{\overline{x}_{i,j} \mid \neg x_i \in C_j\}$. So the propositional formula φ will be processed by the system $\Pi(s(\varphi)) + \mathrm{cod}(\varphi)$.

An instance of the SAT problem is solved by the system $\Pi(s(\varphi)) + \mathrm{cod}(\varphi)$, which can be separated into three phases: generation phase, checking phase, and output phase.

Generation Phase. In this phase, by using separation rules in cell with label 1, all truth assignments for the variables associated with the Boolean formula $\varphi(x_1, \ldots, x_n)$ will be generated. When this phase completes, there are 2^n copies of cell with label 1 such that each of them encodes a different truth assignment of variables $\{x_1, \ldots, x_n\}$.

The generation phase takes $4n + 3$ steps, which has two parallel processes. On the one hand, n loops are executed, and each loop takes four steps. When the loops are completed, three additional steps are executed. On the other hand, there is a counter object α in cell 3 that evolves from α_0 to $\alpha_{4n+3m+3}$, and objects $(\mathrm{cod}(\varphi))_e^{2^n}$, $(l_j)^{2^n}$ $(1 \le j \le m)$ are produced in cell 3 after $n + 1$ steps at this phase.

In the initial configuration, we have objects A_1, B_1 in cell 1, objects $\overline{A}_1$, $\overline{B}_1$, $\overline{C}_1$, $\overline{D}_{1,1}$ in cell 2, and objects $l_{1,0}, \ldots, l_{m,0}$, p, α_0, $\mathrm{cod}(\varphi)$ in cell 3.

In what follows, we first analyze the computation process that takes place in cells 1 and 2; then we explain the computation process that takes place in cell 3.

At the first step of the ith $(3 \le i \le n)$ loop involving cells 1 and 2, by using rules $r_{1,i}$–$r_{4,i}$, objects A_i, A_i', B_i, B_i' in cell 1 are sent to cell 2 and evolved to $\widehat{A}_i$, $\widehat{A}_i'$, $\widehat{B}_i$, $\widehat{B}_i'$, respectively; simultaneously, objects $\overline{A}_i$, $\overline{A}_i$, $\overline{B}_i$, $\overline{B}_i$ in cell 2 are sent to cell 1 and evolved to $\overline{A}_i$, $\overline{A}_i'$, $\overline{B}_i$, $\overline{B}_i'$, respectively. By applying rule $r_{5,i}$, object $\overline{C}_i$ in cell 2 is consumed, and object $\overline{\nu}_i$ in the environment is sent into cell 2 and evolved to ν_i. By using rule $r_{6,i,j}$, object $\widetilde{\nu}_{i,j}$ in the environment is sent into cell 2 and evolved to $q_{i,j}$, and object $\overline{D}_{i,j}$ in cell 2 is consumed. By using rules $r_{7,i}$–$r_{10,i}$, objects t_i, t_i, f_i, f_i in cell 2 are sent to the corresponding cell 1, respectively; simultaneously, objects T_i, T_i', F_i, F_i' in cell 1 are consumed.

Note that at the first step of the 1st loop involving cells 1 and 2, only rules $r_{1,1}$, $r_{3,1}$, $r_{5,1}$, $r_{6,1}$ are applied; at the first step of the 2nd loop involving cells 1 and 2, only rules $r_{1,2}$, $r_{2,2}$, $r_{3,2}$, $r_{4,2}$, $r_{5,2}$, $r_{6,2}$, $r_{7,2}$, $r_{10,2}$ are applied.

At the second step of the ith $(2 \le i \le n)$ loop involving cells 1 and 2, by using rules $r_{11,i}$–$r_{14,i}$, objects $\overline{A}_i$, $\overline{A}_i'$, $\overline{B}_i$, $\overline{B}_i'$ in cell 1 are exchanged with objects $\widetilde{A}_i$, $\widetilde{A}_i$, $\widetilde{B}_i$, $\widetilde{B}_i$ in the environment, and objects a_ia_i', a_ia_i', b_ib_i', b_ib_i' are produced in cell 1, respectively. By applying rules $r_{15,i}$–$r_{18,i}$, objects A_i, A_i', B_i, B_i' are produced in corresponding cell 2, respectively. Object ν_i in cell 2 is exchanged with object $\overline{y}_i$ in the environment by using rule $r_{19,i}$, and two copies of object y_i are produced in cell 2. By applying rule $r_{21,i,j}$, two copies of object $r_{i,j}$ are produced in cell 2. By applying rules $r_{22,i}$, $r_{23,i}$, objects $\overline{t}_i$, $\overline{f}_i$ in the environment are sent into corresponding cell 1 and evolved to T_iT_i', F_iF_i', respectively.

Note that at the second step of the 1st loop involving cells 1 and 2, only rules $r_{11,1}$, $r_{13,1}$, $r_{15,1}$, $r_{17,1}$, $r_{19,1}$, r_{20} are applied.

At the third step of the ith $(2 \le i \le n)$ loop involving cells 1 and 2, by using rules $r_{24,i}$–$r_{27,i}$, objects A_i, A_i', B_i, B_i' in cell 2 are exchanged with objects $\widetilde{c}_i$, $\widetilde{c}_i$, $\widetilde{c}_i$, $\widetilde{c}_i$ in the environment, respectively; objects A_i, A_i', B_i, B_i' are sent to the environment and consumed, and objects $\widetilde{c}_i$, $\widetilde{c}_i$, $\widetilde{c}_i$, $\widetilde{c}_i$ are sent into cell 2 and evolved to c_i. By applying rules $r_{28,i}$–$r_{31,i}$, objects T_iA_{i+1}, $F_i'A_{i+1}'$, $B_{i+1}S$, B_{i+1}' are produced in corresponding cell 1, respectively. By using rule $r_{32,i}$, object y_i in cell 2 is sent to the environment and consumed; simultaneously, object $\widetilde{y}_i$ in the environment is sent into cell 2 and evolved to z_iw_i. By using rule $r_{33,i,j}$, object $r_{i,j}$ in cell 2 is sent to the environment and consumed; simultaneously, object $\widetilde{r}_{i,j}$ in the environment is sent into cell 2 and evolved to $s_iu_{i,j}$.

Note that at the third step of the 1st loop involving cells 1 and 2, only rules $r_{24,1}$, $r_{26,1}$, $r_{28,1}$, $r_{29,1}$, $r_{30,1}$, $r_{31,1}$, $r_{32,1}$, $r_{33,1,1}$ are applied.

At the fourth step of the ith $(2 \le i \le n)$ loop involving cells 1 and 2, by using rule r_{40}, object S is consumed, and the objects from Γ_0 already existing in cell 1 are placed in the first cell, while those from Γ_1 are placed in the second cell. By using rules $r_{34,i}$–$r_{39,i,j}$, objects w_i, c_i, z_i, s_i, $u_{1,j}$, $u_{i,j}$ in cell 2 are sent to the environment and consumed; simultaneously, objects $\widetilde{w}_i$, $\widetilde{d}_i$, $\widetilde{z}_i$, $\widetilde{s}_i$, $\widetilde{u}_{1,j}$, $\widetilde{u}_{i,j}$ in the environment are sent into cell 2 and evolved to $\overline{A}_{i+1}$, $\overline{B}_{i+1}$, $\overline{C}_{i+1}$, t_if_i, $\overline{D}_{1,j+1}\overline{D}_{2,j+1}$, $\overline{D}_{i+1,j+1}$, respectively.

Note that at the fourth step of the 1st loop involving cells 1 and 2, only rules $r_{34,1}$, $r_{35,1}$, $r_{36,1}$, $r_{37,1}$, $r_{38,1}$, r_{40} are applied.

At step $4n + 1$, rules $r_{7,i}$–$r_{10,i}$, $r_{41,i}$–r_{43} are enabled. By applying rules $r_{7,i}$–$r_{10,i}$, objects T_i, T_i', F_i, F_i' in each cell 1 are changed to objects t_i, t_i, f_i, f_i, respectively. By using rule $r_{41,i}$, object $\overline{D}_{i,n+1}$ in cell 2 is sent to the environment; simultaneously, object $\widetilde{\nu}_{i,n+1}$ in the environment is sent into cell 2 and evolved to two copies of object h_i. By applying rules r_{42}, r_{43}, each cell with label 1 will produce an object E_1.

At step $4n + 2$, by using rules $r_{22,i}$, $r_{23,i}$, objects t_i, f_i in each cell 1 are changed to objects T_iT_i', F_iF_i', respectively.

At step $4n + 3$, rules $r_{44,i}$, $r_{45,i}$ are enabled and applied, and objects T_iT_i', F_iF_i' in each cell 1 are changed to $\overline{T}_i$, $\overline{F}_i$, respectively.

At the first $n + 1$ steps of the generation phase, by using rules $r_{46,i,j}$–$r_{53,j}$, 2^n copies of objects $(\mathrm{cod}(\varphi))_e$, l_j $(1 \le j \le m)$ are produced in cell 3.

Checking Phase. This phase takes $3m$ steps and consists of a loop with m iterations, where each iteration takes three steps.

At the first step of the jth loop $(1 \le j \le m)$, objects $\overline{T}_i$, E_j (resp., $\overline{F}_i$, E_j) in cell 1 are exchanged with object $e_{i,j}$ (resp., $\overline{e}_{i,j}$) in cell 3 by using rule $r_{54,i,j}$ (resp., $r_{55,i,j}$), in case cell 1 encodes a truth assignment making clauses $C_1, \ldots, C_j$ true. Note that if a cell with label 1 contains a truth assignment that does not make the clause C_j true, the computation will stop in that cell when rule $r_{54,i,j}$ or rule $r_{55,i,j}$ is applied. When using rule $r_{54,i,j}$ (resp., $r_{55,i,j}$), object $e_{i,j}$ (resp., $\overline{e}_{i,j}$) is sent into cell 1 and evolved to $E_{i,j}$ (resp., $\overline{E}_{i,j}$); simultaneously, objects $\overline{T}_i$, E_j (resp., $\overline{F}_i$, E_j) in cell 1 are sent to cell 3 and consumed. In addition, counter object α in cell 3 is evolved.

At the second step of the jth loop $(1 \le j \le m)$, if a cell with label 1 contains object $E_{i,j}$ (resp., $\overline{E}_{i,j}$), then rule $r_{56,i,j}$ (resp., $r_{57,i,j}$) is enabled and applied, and object $E_{i,j}$ (resp., $\overline{E}_{i,j}$) in cell 1 is sent to cell 3; simultaneously, object l_j in cell 3 is sent to the corresponding cell 1 and evolved to $l_{i,j}$ (resp., $\overline{l}_{i,j}$). Moreover, counter object α in cell 3 is evolved.

At the third step of the jth loop ($1 \leq j \leq m$), rule $r_{58,i,j}$ (resp., $r_{59,i,j}$) is enabled in cell 1 in case object $l_{i,j}$ (resp., $\overline{l}_{i,j}$) appears in that cell. By using rule $r_{58,i,j}$ (resp., $r_{59,i,j}$), object $l_{i,j}$ (resp., $\overline{l}_{i,j}$) in cell 1 is sent to cell 3 and consumed; simultaneously, object $E_{i,j}$ (resp., $\overline{E}_{i,j}$) in cell 3 is sent into corresponding cell 1 and evolved to $\overline{T}_i$, E_{j+1} (resp., $\overline{F}_i$, E_{j+1}). In addition, counter object α increases its subscript.

Output Phase. In this phase, the system sends the right answer into the environment according to the result of the previous phase.

If the input formula φ is satisfiable, then there exists at least one cell with label 1 that contains object E_{m+1} after $4n + 3m + 3$ steps. In this case, at step $4n + 3m + 4$, by using rule $r_{60,i}$, object $\alpha_{4n+3m+4}$ will present in cell 3; simultaneously, rule r_{61} is enabled and applied; object p in cell 3 is sent into the corresponding cell 1 and consumed; object E_{m+1} in cell 1 is sent into cell 3 and evolved to `yes`, which will be sent to the environment at the next step by using rule r_{62}, and the system halts, and the answer of the system is *affirmative*.

If the input formula φ is not satisfiable, then there is no cell with label 1 that contains object E_{m+1} after $4n + 3m + 3$ steps. In this case, at step $4n+3m+4$, only rule $r_{60,i}$ is enabled and applied, and object $\alpha_{4n+3m+4}$ will appear in cell 3. At step $4n+3m+5$, if object p appears in cell 3, then rule r_{63} is applied, objects $\alpha_{4n+3m+4}$, p are sent to the environment and evolved to `no`, the system halts, and the answer of the system is *negative*.

From the computation process, we can check that rules of a system $\Pi(\langle n, m\rangle)$ of the family are defined recursively from values n and m, and the necessary resources to build each such system are as follows:

(i) size of the alphabet: $4n^2m+8n^2+10nm+55n+8m+19 \in O(n^2m)$;

(ii) initial number of cells: $3 \in O(1)$;

(iii) initial number of objects: $m + 8 \in O(m)$;

(iv) number of rules: $2n^2m + 4n^2 + 9nm + 39n + 4m + 5 \in O(n^2m)$;

(v) maximum length of a rule (the total number of objects involved in a rule): $4 \in O(1)$.

Hence, there exists a deterministic Turing machine that builds the system $\Pi(\langle m, n\rangle)$ in a polynomial time with respect to m and n. So the family $\mathbf{\Pi} = \{\Pi(\langle m, n\rangle) \mid m, n \in \mathbb{N}\}$ is polynomially uniform by Turing machines.

The tissue P system $\Pi(\langle n, m\rangle)$ with input multiset $\mathrm{cod}(\varphi)$ always halts and sends object `yes` or `no` to the environment at the last step, that is, at step $4n + 3m + 5$. Hence the family $\mathbf{\Pi} = \{\Pi(\langle m, n\rangle) \mid m, n \in \mathbb{N}\}$ is polynomially bounded.

According to Definition 3, the family $\mathbf{\Pi}$ of recognizer tissue P systems from $\mathbf{TSEC}(3, 2)$ solve `SAT` problem in linear time. Hence, the theorem holds. □

Corollary 9. *One has* $\mathbf{NP} \cup \mathbf{co} - \mathbf{NP} \subseteq \mathbf{PMC}_{\mathbf{TSEC}(3,2)}$.

Proof. The theorem holds because the `SAT` problem is **NP**-complete, `SAT` $\in \mathbf{PMC}_{\mathbf{TSEC}(3,2)}$, and the class $\mathbf{PMC}_{\mathbf{TSEC}(3,2)}$ is closed under polynomial time reduction and under complement. □

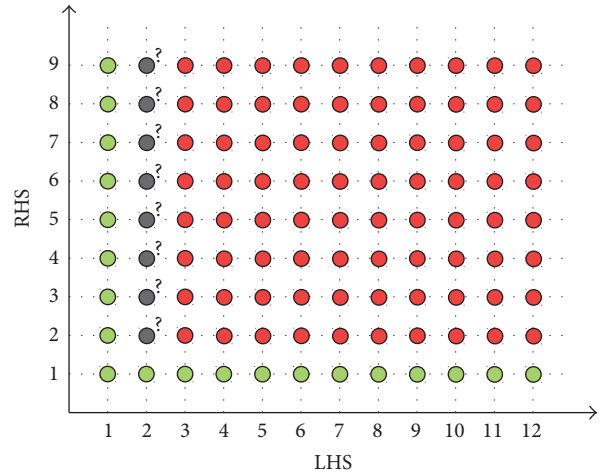

Figure 1: Summary of the results and open problems.

Corollary 10. *For each natural number* $n \geq 3$, *one has* $\mathbf{NP} \cup \mathbf{co} - \mathbf{NP} \subseteq \mathbf{PMC}_{\mathbf{TSEC}(n,2)}$.

Proof. It suffices to notice that for each natural number $n \geq 3$ the following holds: $\mathbf{PMC}_{\mathbf{TSEC}(3,2)} \subseteq \mathbf{PMC}_{\mathbf{TSEC}(n,2)}$. □

Figure 1 illustrates the results obtained in this paper as well as some open problems. The point (p, q) represents the complexity class $\mathbf{PMC}_{\mathbf{TSEC}(p,q)}$. The green circles correspond to class **P**, the red circles correspond to a class containing $\mathbf{NP} \cup \mathbf{co} - \mathbf{NP}$, and the black circles correspond to open problems.

4. A Software Tool to Aid in Formal Validation

The solution to the `SAT` problem by TESAS P systems is very complicated (see Theorem 8), not obviously a trivial solution, easy to follow and check. In order to evaluate the correctness of the solution to the `SAT` problem by TESAS P systems, a visual application was provided (see Algorithm 4 for extensive details). With this software, P systems designers could analyse carefully the design proposed; in addition, end users were able to introduce their propositional formulas visually, then run the simulations, and finally get the results.

As described in Algorithm 4 mentioned, the new tool extends the framework provided by `P-Lingua` and `MeCoSim` in terms of language accepted, simulation engines, and custom apps provided. We present all the aspects of the simulation in detail in Algorithm 4, but let us highlight the main features included at these different levels.

New Elements in P-Lingua Framework. Taking the existing P-Lingua syntax for P systems introduced in [25] as a starting point, some extensions in the syntax of the language had to be included for tissue P systems with evolutional symport/antiport rules. We can mention some of them:

```
@model<tsec>

def main()
{
        call init_membrane_structure();
        call init_first_alphabet(m,n);
        call init_second_alphabet(m,n);
        call init_environment(m,n);
        call init_multisets(m);
        call init_rules(m,n);
        call define_input();
}

def init_membrane_structure()
{
        @mu = [ [ ]'1 [ ]'2 [ ]'3 ]'0;
}

/* @ms1 is %*$\Gamma_0 = \Gamma\setminus\Gamma_1$*), including %*$\Sigma\cup\mathcal{E}$*) */
/* Some special symbols as a bar, tilde, hat or prime are replaced in
P-Lingua plain text file with suffixes b (bar), t (tilde), h (hat) or
p (prime), respectively. Thus, %*$\hat{A}'$*) is written as Ahp ("A hat prime").*/
def init_first_alphabet(m,n)
{
        @ms1 += A{i},Ab{i},Abp{i},Ah{i},Ahp{i},B{i},Bb{i},Bbp{i},
                              Bh{i},Bhp{i},Cb{i},T{i},Tb{i},F{i},Fb{i},a{i},
                              ap{i},b{i},bp{i},c{i},t{i},f{i},h{i},y{i},s{i},
                              v{i},w{i},z{i} : 1<=i<=n;
        @ms1 += Db{i,j},q{i,j},r{i,j},u{i,j} : 1<=j<=n,1<=i<=n;
        @ms1 += Db{i,n+1} : 1<=i<=n+1;
        @ms1 += g{i,j,k},gb{i,j,k} : 1<=k<=n,1<=j<=m,1<=i<=n;
        @ms1 += E{i,j},Eb{i,j},e{i,j},eb{i,j},l{j,i} : 1<=j<=m,1<=i<=n;
        @ms1 += l{j},lb{j},l{j,0},E{j} : 1<= j <= m;
        @ms1 += A{n+1},Ab{n+1},B{n+1},Bb{n+1},Cb{n+1},E{m+1},
                       S,p,yes,no;

        /* Input alphabet %*$\Sigma$*) */
        @ms1 += x{i,j},nx{i,j} : 1<=j<=m,1<=i<=n;

        /* Environment alphabet %*$\mathcal{E}$*) */
        @ms1 += At{i},Bt{i},ab{i},at{i},bb{i},bt{i},cb{i},ct{i},dt{i},
                tb{i},fb{i},st{i},vb{i},vt{i,n+1},wt{i},yb{i},yt{i},
                zt{i} : 1<=i<=n;
        @ms1 += rb{i,j},rt{i,j},ut{i,j},vt{i,j} : 1<=j<=n,1<=i<=n;
        @ms1 += gp{i,j,k},gbp{i,j,k} : 0<=k<=n,1<=j<=m,1<=i<=n;
        @ms1 += lb{j,k} : 0<=k<=n,1<=j<=m;
        @ms1 += al{i},alb{i} : 0<=i<=4*n+3*m+4;
        @ms1 += E{0};
}

def init_second_alphabet(m,n)
{
        @ms2 += Tp{i},Fp{i} : 1<=i<=n;
        @ms2 += Ap{i},Bp{i} : 2<=i<=n+1;
}

def init_environment(m,n)
{
        @ms(0) += At{i},Bt{i},ab{i},at{i},bb{i},bt{i},cb{i},ct{i},
                  dt{i},tb{i},fb{i},st{i},vb{i},vt{i,n+1},wt{i},
                  yb{i},yt{i},zt{i} : 1<=i<=n;
        @ms(0) += rb{i,j},rt{i,j},ut{i,j},vt{i,j} : 1<=j<=n,1<=i<=n;
        @ms(0) += gp{i,j,k},gbp{i,j,k} : 0<=k<=n,1<=j<=m,1<=i<=n;
```

Algorithm 4: Continued.

```
        @ms(0) += lb{j,k} : 0<=k<=n,1<=j<=m;
        @ms(0) += al{i},alb{i} : 0<=i<=4*n+3*m+4;
        @ms(0) += E{0};
}

def init_multisets(m)
{
        @ms(1) = A{1},B{1};
        @ms(2) = Ab{1},Bb{1},Cb{1},Db{1,1};
        @ms(3) = p,al{0};
        @ms(3) += l{i,0} : 1<=i<=m;
}

def init_rules(m,n)
{
        {
                /* r_{1,i} */    [A{i}]'1 [Ab{i}]'2 --> [Ab{i}]'1 [Ah{i}]'2;
                /* r_{2,i} */    [Ap{i}]'1 [Ab{i}]'2 --> [Abp{i}]'1 [Ahp{i}]'2;
                /* r_{3,i} */    [B{i}]'1 [Bb{i}]'2 --> [Bb{i}]'1 [Bh{i}]'2;
                /* r_{4,i} */    [Bp{i}]'1 [Bb{i}]'2 --> [Bbp{i}]'1 [Bhp{i}]'2;
                /* r_{5,i} */    [Cb{i}]'2 [vb{i}]'0 --> [v{i}]'2 []'0;
                /* r_{6,i,j} */  [Db{i,j}]'2 [vt{i,j}]'0 --> [q{i,j}]'2 []'0  :1<=j<=n;
                /* r_{7,i} */    [T{i}]'1 [t{i}]'2 --> [t{i}]'1 []'2;
                /* r_{8,i} */    [Tp{i}]'1 [t{i}]'2 --> [t{i}]'1 []'2;
                /* r_{9,i} */    [F{i}]'1 [f{i}]'2 --> [f{i}]'1 []'2;
                /* r_{10,i} */   [Fp{i}]'1 [f{i}]'2 --> [f{i}]'1 []'2;
        } : 1<=i<=n;
        /* r_{11,i} */    [Ab{i}]'1 [At{i}]'0 --> [a{i},ap{i}]'1 []'0 : 1<=i<=n;
        /* r_{12,i} */    [Abp{i}]'1 [At{i}]'0 --> [a{i},ap{i}]'1 []'0 :2<=i<=n;
        /* r_{13,i} */    [Bb{i}]'1 [Bt{i}]'0 --> [b{i},bp{i}]'1 []'0 : 1<=i<=n;
        /* r_{14,i} */    [Bbp{i}]'1 [Bt{i}]'0 --> [b{i},bp{i}]'1 []'0 :2<=i<=n;

        /* r_{15,i} */    [Ah{i}]'2 [At{i}]'0 --> [A{i}]'2 []'0 : 1<=i<=n;
        /* r_{16,i} */    [Ahp{i}]'2 [At{i}]'0 --> [Ap{i}]'2 []'0 : 2<=i<=n;
        /* r_{17,i} */    [Bh{i}]'2 [Bt{i}]'0 --> [B{i}]'2 []'0 : 1<=i<=n;
        /* r_{18,i} */    [Bhp{i}]'2 [Bt{i}]'0 --> [Bp{i}]'2 []'0 : 2<=i<=n;
        /* r_{19,i} */    [v{i}]'2 [yb{i}]'0 --> [y{i}*2]'2 []'0 : 1<=i<=n;
        /* r_{20} */      [q{1,1}]'2 [rb{1,1}]'0 --> [r{1,1}]'2 []'0;
        /* r_{21,i,j} */  [q{i,j}]'2 [rb{i,j}]'0 --> [r{i,j}*2]'2 []'0
                                : 2<=j<=n, 1<=i<=n;

        {
               /* r_{22,i} */ [t{i}]'1 [tb{i}]'0 --> [T{i},Tp{i}]'1 []'0;
               /* r_{23,i} */ [f{i}]'1 [fb{i}]'0 --> [F{i},Fp{i}]'1 []'0;
        } : 1<=i<=n;

        /* r_{24,i} */   [A{i}]'2 [cb{i}]'0 --> [c{i}]'2 []'0 : 1<=i<=n;
        /* r_{25,i} */   [Ap{i}]'2 [ct{i}]'0 --> [c{i}]'2 []'0 : 2<=i<=n;
        /* r_{26,i} */   [B{i}]'2 [cb{i}]'0 --> [c{i}]'2 []'0 : 1<=i<=n;
        /* r_{27,i} */   [Bp{i}]'2 [ct{i}]'0 --> [c{i}]'2 []'0 : 2<=i<=n;

        /* r_{28,i} */   [a{i}]'1 [ab{i}]'0 --> [T{i},A{i+1}]'1 []'0 : 1<=i<=n;
        /* r_{29,i} */   [ap{i}]'1 [at{i}]'0 --> [Fp{i},Ap{i+1}]'1 []'0
                        : 1<=i<=n;

        {
               /* r_{30,i} */    [b{i}]'1 [bb{i}]'0 --> [B{i+1},S]'1 []'0;
               /* r_{31,i} */    [bp{i}]'1 [bt{i}]'0 --> [Bp{i+1}]'1 []'0;
               /* r_{32,i} */    [y{i}]'2 [yt{i}]'0 --> [z{i},w{i}]'2 []'0;
               /* r_{33,i,j} */  [r{i,j}]'2 [rt{i,j}]'0 --> [s{i},u{i,j}]'2 []'0
                                 : 1<=j<=n;
               /* r_{34,i} */    [w{i}]'2 [wt{i}]'0 --> [Ab{i+1}]'2 []'0;
               /* r_{35,i} */    [c{i}]'2 [dt{i}]'0 --> [Bb{i+1}]'2 []'0;
```

Algorithm 4: Continued.

```
                /* r_{36,i} */    [z{i}]'2 [zt{i}]'0 --> [Cb{i+1}]'2 []'0;
                /* r_{37,i} */    [s{i}]'2 [st{i}]'0 --> [t{i},f{i}]'2 []'0;
        } : 1<=i<=n;
        /* r_{38,j} */ [u{1,j}]'2 [ut{1,j}]'0 --> [Db{1,j+1},Db{2,j+1}]'2 []'0
                       : 1<=j<=n;
        /* r_{39,i,j} */ [u{i,j}]'2 [ut{i,j}]'0 --> [Db{i+1,j+1}]'2 []'0
                         : 2<=j<=n, 2<=i<=n;

        /* r_{40} */ [S]'1 --> []'1 []'1;

        /* r_{41,i} */ [Db{i,n+1}]'2 [vt{i,n+1}]'0 --> [h{i}*2]'2 []'0
                       : 1<=i<=n;

        /* r_{42} */ [A{n+1}]'1 [Ab{n+1}]'2 --> [E{1}]'1 []'2;
        /* r_{43} */ [Ap{n+1}]'1 [Bb{n+1}]'2 --> [E{1}]'1 []'2;

        {
                /* r_{44,i} */  [T{i},Tp{i}]'1 [h{i}]'2 --> [Tb{i}]'1 []'2;
                /* r_{45,i} */  [F{i},Fp{i}]'1 [h{i}]'2 --> [Fb{i}]'1 []'2;
        } : 1<=i<=n;

        {
                /* r_{46,i,j} */ [x{i,j}]'3 [gp{i,j,0}]'0 --> [g{i,j,1}*2]'3 []'0;
                /* r_{47,i,j} */ [xb{i,j}]'3 [gbp{i,j,0}]'0 --> [gb{i,j,1}*2]'3 []'0;
                {
                /*r_{48,i,j,k}*/ [g{i,j,k}]'3 [gp{i,j,k}]'0 --> [g{i,j,k+1}*2]'3 []'0;
                /*r_{49,i,j,k}*/ [gb{i,j,k}]'3 [gbp{i,j,k}]'0-->[gb{i,j,k+1}*2]'3 []'0;
                } : 1<=k<=n-1;
                /* r_{50,i,j} */ [g{i,j,n}]'3 [gp{i,j,n}]'0 --> [e{i,j}]'3 []'0;
                /* r_{51,i,j} */ [gb{i,j,n}]'3 [gbp{i,j,n}]'0 --> [eb{i,j}]'3 []'0;
        } : 1<=j<=m, 1<=i<=n;
        /* r_{52,j,k} */ [l{j,k}]'3 [lb{j,k}]'0 --> [l{j,k+1}*2]'3 []'0
                         : 0<=k<=n-1, 1<=j<=m;
        /* r_{53,j} */ [l{j,n}]'3 [lb{j,n}]'0 --> [l{j}]'3 []'0 : 1<=j<=m;

        {
                /* r_{54,i,j} */ [Tb{i},E{j}]'1 [e{i,j}]'3 --> [E{i,j}]'1 []'3;
                /* r_{55,i,j} */ [Fb{i},E{j}]'1 [eb{i,j}]'3 --> [Eb{i,j}]'1 []'3;
                /* r_{56,i,j} */ [E{i,j}]'1 [l{j}]'3 --> [l{i,j}]'1 [E{i,j}]'3;
                /* r_{57,i,j} */ [Eb{i,j}]'1 [l{j}]'3 --> [lb{i,j}]'1 [Eb{i,j}]'3;
                /* r_{58,i,j} */ [l{i,j}]'1 [E{i,j}]'3 --> [Tb{i},E{j+1}]'1 []'3;
                /* r_{59,i,j} */ [lb{i,j}]'1 [Eb{i,j}]'3 --> [Fb{i},E{j+1}]'1 []'3;
        } : 1<=j<=m, 1<=i<=n;

        /*r_{60,i}*/ [al{i}]'3 [alb{i}]'0-->[al{i+1}]'3 []'0 : 0<=i<=4*n+3*m+3;

        /* r_{61} */ [E{m+1}]'1 [p]'3 --> []'1 [yes]'3;

        /* r_{62} */ [yes]'3 []'0 --> []'3 [yes]'0;
        /* r_{63} */ [al{4*n+3*m+4},p]'3 []'0 --> []'3 [no]'0;
}

/* Input cell: 3 */
def define_input()
{
        @ms(3) += xb{variable{i},clause{i}}*valn{i},
                  x{variable{i},clause{i}}*val{i} : 1<=i<=nvals;
}
```

Algorithm 4: P-Lingua translation for SAT solution.

(i) New model type, `tsec`:

```
@model<tsec>
```

(ii) Communication rules with two cells in the right hand of the rule, as:

```
[Eb{i,j}]'1 [l{j}]'3 --> [lb{i,j}]'1
[Eb{i,j}]'3 : 1<=j<=m, 1<=i<=n;
[al{4*n+3*m+4},p]'3 [ ]'0 --> [ ]'3 [no]'0;
```

Custom App Based on `MeCoSim`. A new visual app has been customised using `MeCoSim` for the solution of `SAT` with TESAS P systems, including a series of relevant elements:

(i) Input tables to introduce the propositional formulas. The data introduced in these tables are then converted into parameters for the model that can be used to generate the corresponding cod(φ).

(ii) Use of existing SAT plugin to provide end users with a more natural way of introducing the formulas that are then converted to the format of the previous tables.

(iii) Definition of the output showing in different formats if the formula is satisfiable or not, generated from the objects of the computation.

New Simulators for Tissue P Systems with Evolutional Rules. In the proof of Theorem 5, a deterministic algorithm was given to reproduce the behaviour of a computation of $\Pi + m$, with m an input multiset coding the input of the system. To mimic the algorithm described, several simulators were developed to compute P systems in **TSEC**. These simulators present subtle differences to select the rules to be applied, all of them compliant with the semantics defined for these systems (more details in Algorithm 4). The main differences among them are the following:

(i) *Nondeterministic simulator*: this simulator distinguishes two states (first communication rules are selected and later separation rules if no communication rules can be selected over a cell); this simulator visits cell by cell, according to its order of appearance in P-Lingua file; for each cell, the set of rules is shuffled and then applied as many times as possible, to guarantee maximality.

(ii) *Deterministic by cell*: the selection phase keeps the same two states of the previous simulator, but the cells are visited according to the lexicographical order of their labels. The set of rules inside each cell is not shuffled, but visited according to P-Lingua file order.

(iii) *Deterministic by rule*: this simulator is more similar to the deterministic algorithm described in the proof of Theorem 5. Thus, its selection phase, instead of going cell by cell (as the previous simulators), starts visiting at the most external level each rule. As described in Section 3, the rules are visited in the following order: direct communication between cells, then communication with environment, and finally separation, if possible; for each type of rules of the same type, it selects the rules following the order in the P-Lingua file.

(iv) *Deterministic by rule, lexicographically*: it performs the computations as described in the proof of Theorem 5. In addition to the most external visit of the different set of rules, in the same order of the previous simulator, inside a specific type of rules, the cells are visited according to the lexicographical order of their labels, thus meeting the structure defined in the referred section.

Computer-Aided Design and Validation of Tissue P Systems with Evolutional Symport/Antiport Rules. The tools described played a significant role in the design, simulation, and validation of the solution for the `SAT` problem described in Theorem 8. Handling these complex solutions is hard and error-prone, and here the availability of the software tools helped significantly in the checking of the model and its properties, as detailed in the Appendix. Additionally, the development of the tools implied revisiting the general algorithms proposed for the model, thus paying attention to subtle details, given the need of reproducing every detail specified in the formal definition of the model, so these development tasks emerged as enriching complementary works to enhance the robustness of the computing models and their corresponding theoretical definitions.

5. Conclusions and Further Works

The computational complexity of tissue P systems with cell separation was first investigated in [29], where using communication rules with the length of at most 1, only tractable problems can be efficiently solved; and using communication rules with length at most 8, the `SAT` problem can be solved. In [42], it is shown that only tractable problems can be efficiently solved by tissue P systems with cell separation using communication rules with length at most 2; and the `SAT` problem can be solved by this kind of P systems using communication rules with length at most 3 [26]. In [33], tissue P systems with evolutional symport/antiport rules and cell division (TESAD P systems, for short) were proposed, where objects are moved from one region to another region and may be evolved during this process. It is shown that a limit on the efficiency of TESAD P systems is provided with evolutional communication rules of length at most 2; and when using evolutional communication rules of length at most 4, the `SAT` problem can be solved by TESAD P systems.

In this work, membrane fission as a mechanism to generate an exponential workspace (expressed in terms of number of membranes and number of objects) has been considered instead of membrane division, in the framework of tissue P systems with evolutional symport/antiport rules. The computational efficiency of this kind of tissue P systems has been investigated. In this context, the main contributions

of the paper are the following: (a) only problems in class **P** can be efficiently solved by means of families of tissue P systems with cell separation and evolutional communication rules of length at most $(n, 1)$ (or rules of length at most $(1, n)$), for each natural number $n \geq 1$; (b) computationally hard problems can be solved in polynomial time by recognizer TESAS P systems when using evolutional communication rules of length at most $(n, 2)$, for each natural number $n \geq 3$; and (c) a new `MeCoSim` based simulator has been designed to check the correctness of the solution to the `SAT` problem.

As future works, we propose the following:

(1) In Section 3, a polynomial time solution to the `SAT` problem by means of a family of tissue P systems from **TSEC**$(3, 2)$ has been provided. It remains open whether **NP**-complete problems can be efficiently solved by tissue P systems from **TSEC**$(2, 2)$. What about the computational efficiency of **TSEC**$(2, n)$, for each natural number $n \geq 2$? Moreover, the solution to the `SAT` problem in Theorem 8 has both evolutional symport rules and evolutional antiport rules. It is of interest to investigate the computational power of tissue P systems with cell separation that use only either evolutional symport or evolutional antiport rules.

(2) In the framework of tissue P systems, the environment is a singular region since the objects initially placed in it have an arbitrary large number of copies. Tissue P systems with cell separation and without environment (the alphabet of the environment is empty) were considered in [43]. It would be interesting to analyze the computational efficiency of recognizer TESAS P systems when the alphabet of the environment is an empty set.

(3) Besides much investigated maximal parallelism, several ways of using rules were also considered in membrane computing, such as flat maximal parallelism [44, 45]. It remains open what the computation power of recognizer TESAS P systems working in a flat maximally parallel manner is.

(4) In [33], recognizer TESA P systems with cell division were introduced and the length of an evolutional communication rule was defined as the total number of objects involved in it. The computational efficiency of **TDEC**(4), the class of all recognizer TESA P systems with cell division using evolutional communication rules with length at most 4, was provided. We propose to analyze the efficiency of recognizer TESA P systems with cell division with respect to the length of evolutional communication rules given in this paper: length(r) = (length(LHS(r)), length(RHS(r))). In fact, the solution to `SAT` problem provided in the cited paper corresponds to a family from **TDEC**$(3, 2)$. What about the computational efficiency of **TDEC**$(2, n)$, for each natural number $n \geq 2$?

Appendix

A Software Aid for Validation: Details and Remarks

The solution to the `SAT` problem by TESAS P systems is very complicated (see Theorem 8); it is not obviously a trivial solution, easy to follow and check. In order to check whether the solution to the `SAT` problem by TESAS P systems is correct, a custom visual application based on MeCoSim (`http://www.p-lingua.org/mecosim/`) was provided (see [25] for further study). With this software application, P systems designers could analyse carefully their design; in addition, end users were able to introduce their propositional formulas visually, then run the simulations, and finally get the results.

In what follows, we will develop a new simulator MeCoSim to check the correctness of the solution to the `SAT` problems.

Taking the existing P-Lingua syntax for P systems introduced in [25] as a starting point, we will introduce the syntax for tissue P systems with evolutional symport/antiport rules (that we will abbreviate referring to the class as **TSEC**), along the following subsections. Then, we will provide a brief description about the simulators developed to run this kind of systems.

New Elements in P-Lingua Framework. P-Lingua framework includes up to now a significant number of P systems supported; however, it cannot obviously include elements not conceived so far. Therefore, if we want to simulate a new model, some elements might be necessary. The following types of rules were available for classical symport/antiport rules:

(i) $(1, T_1/t_1, 2)$;

(ii) $(3, E_3/\lambda, 0)$.

The P-Lingua syntax for these classical types of communication rules was the following:

```
[ T{1} ]'1 <--> [ t{1} ]'2;
/* antiport rule $(1, T_1/t_i, 2))$ */
[ E{3} ]'3 <--> [ ]'0.
/* symport rule $(3, E_3/\lambda, 0))$ */
```

Thus, the syntax tried to mimic the theoretical description of the computing model. This kind of rules might also be present in P systems in **TSEC**. However, in this new model, additional types of rules appear, and the previous ingredients are not enough, so P-Lingua language had to be extended to include the possibility of rules with two cells in the left hand side of the rule, as the ones described along the paper:

(i) $[\overline{E}_{i,j}]_1\,[l_j]_3 \rightarrow [\bar{l}_{i,j}]_1[\overline{E}_{i,j}]_3$, $1 \leq i \leq n$, $1 \leq j \leq m$;

(ii) $[\alpha_{4n+3m+4}p]_3[\ ]_0 \rightarrow [\ ]_3[\mathtt{no}]_0$.

Model Specification. Any P-Lingua file defining a P system in **TSEC** must set `tsec` as its model, thus beginning such file with the sentence:

```
@model<tsec>
```

The rest of the file will then define the main elements describing the P system, typically consisting of

$$\Pi = \left(\Gamma, \Gamma_0, \Gamma_1, \mathscr{E}, \Sigma, \mathscr{M}_1, \ldots, \mathscr{M}_q, \mathscr{R}, i_{\text{in}}, i_{\text{out}}\right). \quad \text{(A.1)}$$

These elements are described in detail in the following subsections. The sets of rules will include the new elements introduced by this work, and other subsections, without significant changes with respect to other existing models, are included here to make the work self-contained.

Cells Structure. Before starting with the translation of the elements explicitly written in Π, let us pay attention to the structure of the system. While the structure of cells is implicit in the definition above, which is deduced from $\mathscr{M}_1, \ldots, \mathscr{M}_q$, in P-Lingua, the structure of cells must be explicitly described. For instance, the structure of a tissue P system in **TSEC** with $q = 3$ should be defined as follows:

```
@mu = [ [ ]'1 [ ]'2 [ ]'3 ]'0;
```

As it can be seen, an external structure labelled by 0 is included, representing the environment *containing* the three cells.

Alphabets. The first element appearing in the definition of a P system is Γ, with partition divided into Γ_0 and Γ_1. These alphabets are described in P-Lingua in the following way:

```
@ms1 = a, b, c;
/* Describe $\Gamma_0$), containing the set $\{a, b, c\})$ */
@ms2 = d, e, f;
/* Describe $\Gamma_1$), containing the set $\{d, e, f\})$ */
```

The alphabet of the environment $\mathscr{E}$ is specified through the assignment of an initial set of elements to the environment, in the usual way to specify multisets in P-Lingua, but in this case with the target region 0 representing the environment. For instance, let us consider an alphabet of the environment as

$$\mathscr{E} = \{E_0\} \cup \{\alpha_i \mid 0 \leq i \leq 4n + 3m + 4\}. \quad \text{(A.2)}$$

It would be written in P-Lingua as

```
@ms(0) = E{0};
@ms(0) += al{i} : 0<=i<=4*n+3*m+4;
/* $\alpha_i$ is written as al{i} */
```

Note that while a simple set can be assigned by the symbol `=`, the incremental inclusion of new elements requires the use of symbol `+=`. Otherwise, the previous assignment is replaced by the new one, not only between the first instruction and the second, but also for each iteration of the loop involving variable `i`.

No explicit definition of Σ is given in P-Lingua. On the contrary, the corresponding symbols are introduced in Γ_0 or Γ_1, thus being part of Γ, and their use as elements of the input alphabet will be given by its inclusion in the input cell, which is also not explicitly given.

Definition of Initial Multisets. When defining P systems in **TSEC**, a relevant element to include is the multiset of objects initially present inside each cell, $\mathscr{M}_1, \ldots, \mathscr{M}_q$. This is performed through the assignment of the multiset to the label of the corresponding cell. Thus, given $\mathscr{M}_1 = \{A_1, B_1\}$, it can be specified in P-Lingua as follows:

```
@ms(1) = A{1}, B{1};
```

Definition of Rules. The main focus of this work was on the study of the computational complexity of tissue P systems with evolutional symport/antiport rules. More specifically, relevant results regarding frontiers of the efficiency of P systems in **TSEC** were provided, and in this kind of systems the role of the evolutional rules was crucial. It is not therefore surprising that the most relevant ingredients included in P-Lingua framework as part of this work were the evolutional communication rules needed as part of the definition of the sets R_i described below.

(i) $[\overline{E}_{i,j}]_1[l_j]_3 \to [\bar{l}_{i,j}]_1[\overline{E}_{i,j}]_3$, $1 \leq i \leq n$, $1 \leq j \leq m$,

(ii) $[\alpha_{4n+3m+4}p]_3[\ ]_0 \to [\ ]_3[\text{no}]_0$

The grammar designed in P-Lingua for `tsec` includes the required elements, so that these rules can be specified as shown:

```
[Eb{i,j}]'1 [l{j}]'3 --> [lb{i,j}]'1
[Eb{i,j}]'3 : 1<=j<=m, 1<=i<=n;
[al{4*n+3*m+4},p]'3 []'0 --> []'3 [no]'0;
```

Note that the suffix `b` in some objects stands for *bar*, given the impossibility to express the top bars over the letters in plain text files. Apart from that, the design of the language tried to preserve the syntax as close as possible of that for defining the P systems throughout the paper. Thus, the first rule is an evolutional antiport rule involving cells 1 and 3, and the second one is an evolutional symport rule.

Input and Output Region. The last syntactic elements in the definition of P systems in **TSEC** are the input and the output region. There is no explicit definition of these elements in P-Lingua. Instead, some elements can be defined in MeCoSim website in relation with that input/output. Regarding the input, MeCoSim provides most user-oriented layer of the framework, allowing the customization of input tables in the user interface. The data introduced in these tables are then converted into parameters for the model that can be used to generate the corresponding $\text{cod}(\varphi)$.

Let us illustrate this with the solution for the `SAT` problem given in the previous section and whose P-Lingua file is given in Algorithm 4. A MeCoSim custom app has been defined,

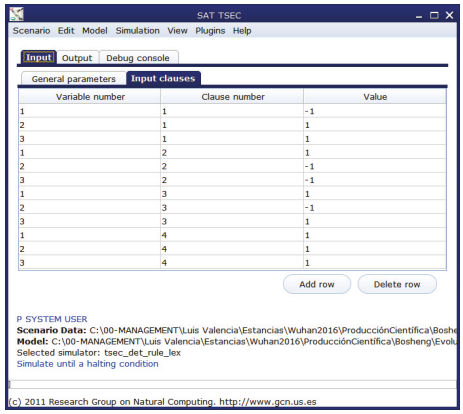

Figure 2: MeCoSim input table.

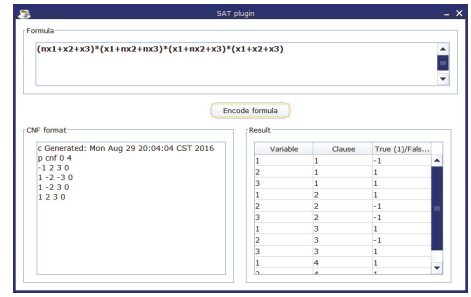

Figure 3: MeCoSim SAT plugin.

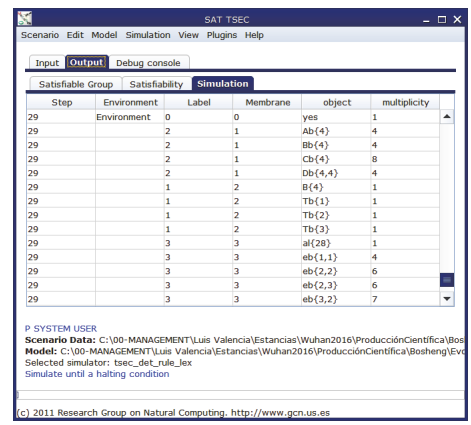

Figure 4: MeCoSim output for P systems designer.

including an input table whose rows represent tuples (clause, variable, and value), so that, for each variable appearing in a clause and input formula, the value can be 1 if the variable appears in the clause in positive form and −1 if it is negated (see Figure 2). For each row `i`, the configuration given in MeCoSim defines parameters for `variable{i}`, `clause{i}`, and `value{n}`; this last parameter is converted to other two parameters, val and valn, each one being 1 if the value is 1 (resp., −1), and 0 otherwise.

In the given solution, the input cell is 3, so the input multiset cod(φ) is finally generated, from an input given by the input and automatically converted into parameters, with the following sentence in P-Lingua:

```
def define_input ()
{
@ms(3) += xb{variable{i}, clause{i}}*valn{i},
x{variable{i}, clause{i}}*val{i} : 1<=i<=nvals;
}
```

As it can be seen, the assignment of objects to the input cell iterates over each pair ⟨variable, clause⟩ (note that `nvals` is another parameter corresponding with the number of rows in the input table, that is, the number of pairs ⟨variable, clause⟩). Thus, if row 1 of the input table states that variable 2 appears in clause 1 in negated form (resp., in affirmed form), then an object `xb{2,1}` (resp., `x{2,1}`) is added to the input multiset, as expected by the definition of the input given in the previous subsection, where objects $\overline{x}_{i,j}$ (resp., $x_{i,j}$) were described. It is worth noting at this point that the end user was provided with an additional aid to introduce their formulas, through a MeCoSim plugin for `SAT` that generates the information for the tables from a more user-friendly input format, as shown in Figure 3.

Regarding the output cell, again there is no explicit definition in P-Lingua, but the whole system will follow a computation; when a halting configuration is reached, one can interpret the output in many different ways. However, another mechanism in MeCoSim allows the customization of outputs to focus on the regions, objects, or results expected by the user. Thus, for SAT problem, a P system designer can pay attention to every object appearing inside each cell or in the environment, as shown in Figure 4; however, an end user interested in `SAT` problem itself will only focus on the final answer, given internally by the presence of object *yes* or *no* in the environment in the last step of the computation and abstracted to the user by tables or charts as shown in Figure 5.

New Simulators for Tissue P Systems with Evolutional Rules. The main elements of the language designed for the specification of P systems in **TSEC** within the framework of P-Lingua have been described. In addition, some features have been shown concerning the interaction layer provided by MeCoSim, the introduction of inputs, and the visualization of results. However, nothing has been said about what happens with the specification of the P system given in P-Lingua and the input given by the end user, before the result actually appears. Obviously, the answer is clear: we need something able to generate the initial configuration of the P system and perform its computation until a halting configuration is reached.

In the proof of Theorem 5, a deterministic algorithm was given to reproduce the behaviour of a computation of $\Pi + m$, with m an input multiset coding the input of the system. We will see that a simulator has been developed to mimic the algorithm described, but before that other alternative approaches have been followed to complement it. Thus, several simulators have been developed within the

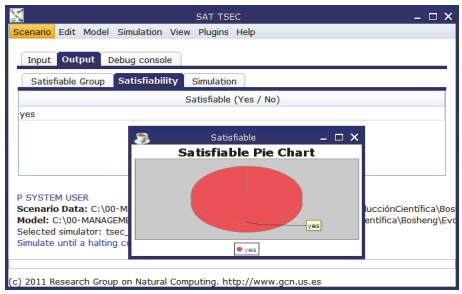

FIGURE 5: MeCoSim output for end user.

engine of P-Lingua, in order to perform the computations of P systems in **TSEC**. All the simulators follow a general schema:

(1) Initialization

(2) For each computation step, while some rules are applicable:

   (a) Selection of rules

   (b) Execution of rules

The *initialization phase* will set the initial structures needed by the algorithm, and the details are mainly technical, not considered very relevant for scientific purposes. Then, the main loop will run until a halting configuration is reached, that is, until no rule is applicable at a given computation step.

The *selection phase* will check which rules can be applied. The simulators developed present subtle differences in this phase, but all of them meet the semantics defined for this kind of newly defined systems. Thus, the maximality is guaranteed, and the applicability of the rules is the same in all the cases: the multisets present in the left hand side of the rules must be present in order to make a rule applicable. Besides, at most one selection rule can be performed over the same cell in a computation step, and the simultaneous execution of separation and communication rules in the same step affecting the same cell is not allowed. In addition, the simulators developed prioritize the selection of evolutional symport/antiport rules (abbreviated as communication rules), so that selection rules will only be applied to a cell if no evolutional symport/antiport rules can be selected affecting the cell. The main differences in the four simulators developed are the following:

(i) *Nondeterministic simulator*: this simulator distinguishes two states (first communication rules are selected and later separation rules if no communication rules can be selected over a cell); this simulator visits cell by cell, according to its order of appearance in P-Lingua file; for each cell, the set of rules is shuffled and then applied as many times as possible, to guarantee maximality.

(ii) *Deterministic by cell*: the selection phase keeps the same two states of the previous simulator, but the cells are visited according to the lexicographical order of their labels. The set of rules inside each cell is not shuffled but visited according to P-Lingua file order.

(iii) *Deterministic by rule*: this simulator is more similar to the deterministic algorithm described in the proof of Theorem 5. Thus, its selection phase, instead of going cell by cell (as the previous simulators), starts visiting at the most external level each rule. As described in that section, the rules are visited in the following order: direct communication between cells, then communication with environment, and finally separation, if possible; for each type of rules of the same type, it selects the rules following the order in the P-Lingua file.

(iv) *Deterministic by rule, lexicographically*: it performs the computations as described in the proof of Theorem 5. In addition to the most external visit of the different set of rules, in the same order of the previous simulator, inside a specific type of rules, the cells are visited according to the lexicographical order of their labels, thus meeting the structure defined in the referred section.

As a result of this selection phase, a set of rules will have been selected, verifying that every membrane with applicable rules has selected exactly one.

Then, the *execution phase* applies the change in the configuration, passing from $\mathscr{C}_t$ to $\mathscr{C}_{t+1}$, removing the objects consumed by the selected rules, and adding the objects produced by the rules to the corresponding target indicators.

Computer-Aided Design and Validation of Tissue P Systems with Evolutional Symport/Antiport Rules. The tools described in the previous subsections have played a significant role in the design, simulation, and validation of the solution for the `SAT` problem described in Theorem 8. As mentioned at the beginning of the present section, handling complex solutions as the one presented here is not an easy task, frequently tedious end error-prone. Therefore, the availability of the software tools developed has provided a significant help in the checking of the model and its properties.

First of all, these tools include parsers to detect possible errors in P-Lingua files that could be translation errors or due to inaccuracies from the solution, maybe not meeting some features required by the computing model. Throughout the debugging process, we are informed about the rules that are being generated for the system, so that we can check that the expected sets of rules are actually available for the computation of the system, as shown in Figure 6. If some errors are detected, we are informed in Errors tab. Other possible alerts are given in Warnings tab.

In addition, parsing tab will show at the end the initial configuration of the system, thus allowing the P systems designer to contrast if the expected multisets were produced.

Once the solution has been proved correct after the debugging process, we can be interested in checking that the system evolves according to our manual traces; to do so, we can follow the computation step by step, informing about the rules applied for each step and the objects contained inside each region of every configuration, as shown in Figure 7.

In addition, several visual aids are available to ease the checking of information concerning alphabets, structure, or

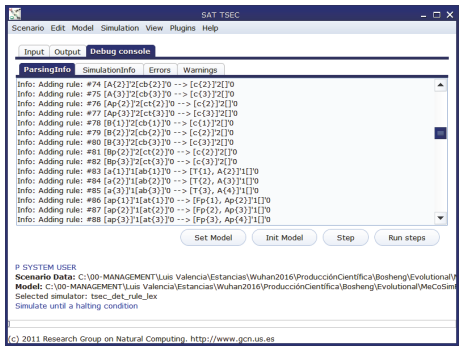

Figure 6: Debugging process.

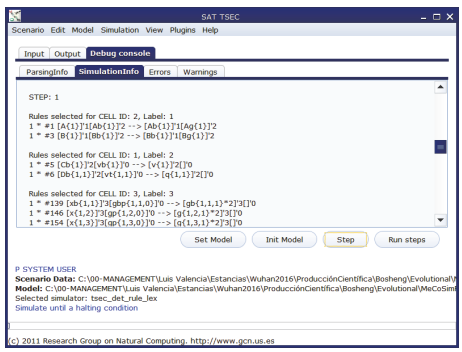

Figure 7: Step by step simulation process.

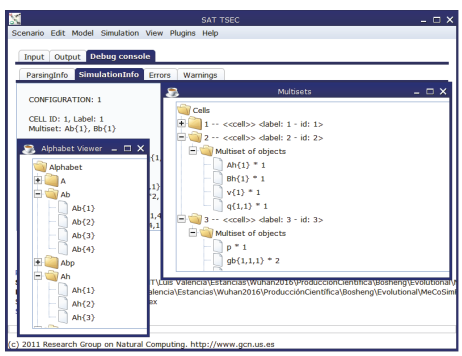

Figure 8: An example of simulation information in a computation.

multisets inside each region. These aids can be visualized in any moment of the computation. An example of this kind of viewers is given in Figure 8.

Additionally, along with the checking of the specific solution, the development of this type of tools shows itself as a good way of revisiting the general algorithms proposed for the model, paying attention to subtle details, given the need of reproducing every detail specified in the formal definition of the model, so these development tasks emerge as enriching complementary tasks for the robustness of the computing models and their corresponding theoretical definitions.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this article.

Acknowledgments

The work of Linqiang Pan and Bosheng Song was supported by National Natural Science Foundation of China (61602192, 61772214, 61320106005, and 61033003), China Postdoctoral Science Foundation (2016M600592, 2017T100554), and the Innovation Scientists and Technicians Troop Construction Projects of Henan Province (154200510012).

References

[1] Gh. Păun, "Computing with membranes," *Journal of Computer and System Sciences*, vol. 61, no. 1, pp. 108–143, 2000.

[2] M. Amos, A. Gibbons, and P. E. Dunne, "Toward feasible and efficient DNA computation," *Complexity*, vol. 4, no. 1, pp. 20–24, 1998.

[3] C. S. Calude and Gh. Păun, "Computing with cells and atoms in a nutshell," *Complexity*, vol. 6, no. 1, pp. 38–48 (2001), 2000.

[4] B. Aman, P. Battyányi, G. Ciobanu, and G. Vaszil, "Simulating P systems with membrane dissolution in a chemical calculus," *Natural Computing*, vol. 15, no. 4, pp. 521–532, 2016.

[5] M. Ionescu, Gh. Păun, and T. Yokomori, "Spiking neural P systems with an exhaustive use of rules," *International Journal of Unconventional Computing*, vol. 3, no. 2, pp. 135–156, 2007.

[6] S. N. Krishna and Gh. Păun, "P systems with mobile membranes," *Natural Computing*, vol. 4, no. 3, pp. 255–274, 2005.

[7] V. Manca and L. Marchetti, "Solving dynamical inverse problems by means of Metabolic P systems," *BioSystems*, vol. 109, no. 1, pp. 78–86, 2012.

[8] H. Peng, J. Wang, M. J. Pérez-Jiménez, and A. Riscos-Núñez, "An unsupervised learning algorithm for membrane computing," *Information Sciences*, vol. 304, pp. 80–91, 2015.

[9] H. Peng, J. Wang, P. Shi, M. J. Pérez-Jiménez, and A. Riscos-Núñez, "An extended membrane system with active membranes to solve automatic fuzzy clustering problems," *International Journal of Neural Systems*, vol. 26, no. 3, pp. 1–17, 2016.

[10] G. Zhang, H. Rong, F. Neri, and M. J. Pérez-Jiménez, "An optimization spiking neural P system for approximately solving combinatorial optimization problems," *International Journal of Neural Systems*, vol. 24, no. 5, 2014.

[11] C. Martín-Vide, J. Pazos, Gh. Păun, and A. Rodríguez-Patón, "Tissue P systems," *Theoretical Computer Science*, vol. 296, no. 2, pp. 295–326, 2003.

[12] M. Ionescu, Gh. Păun, and T. Yokomori, "Spiking neural P systems," *Fundamenta Informaticae*, vol. 71, no. 2-3, pp. 279–308, 2006.

[13] Gh. Păun, G. Rozenberg, and A. Salomaa, Eds., *The Oxford Handbook of Membrane Computing*, Oxford University Press, New York, NY, USA, 2010.

[14] A. Păun and Gh. Păun, "The power of communication: P systems with symport/antiport," *New Generation Computing*, vol. 20, no. 3, pp. 295–305, 2002.

[15] A. Alhazov, R. Freund, and M. Oswald, "Cell/symbol complexity of tissue P systems with symport/antiport rules," *International Journal of Foundations of Computer Science*, vol. 17, no. 1, pp. 3–25, 2006.

[16] S. N. Krishna, K. Lakshmanan, and R. Rama, "Tissue P systems with contextual and rewriting rules," *Lecture Notes in Computer Science*, vol. 2597, pp. 339–351, 2003.

[17] A. Păun, Gh. Păun, and G. Rozenberg, "Computing by communication in networks of membranes," *International Journal*

of Foundations of Computer Science, vol. 13, no. 6, pp. 779–798, 2002.

[18] B. Song, T. Song, and L. Pan, "A time-free uniform solution to subset sum problem by tissue P systems with cell division," *Mathematical Structures in Computer Science*, vol. 27, no. 1, pp. 17–32, 2017.

[19] R. Freund, Gh. Păun, and M. J. Pérez-Jiménez, "Tissue P systems with channel states," *Theoretical Computer Science*, vol. 330, no. 1, pp. 101–116, 2005.

[20] R. Freund and M. Oswald, "Modelling grammar systems by tissue P systems working in the sequential mode," *Fundamenta Informaticae*, vol. 76, no. 3, pp. 305–323, 2007.

[21] Gh. Păun, M. J. Pérez-Jiménez, and A. Riscos-Núñez, "Tissue P systems with cell division," *International Journal of Computers, Communications & Control*, vol. 3, no. 3, pp. 295–303, 2008.

[22] B. Aman and G. Ciobanu, "Efficiently solving the bin packing problem through bio-inspired mobility," *Acta Informatica*, vol. 54, no. 4, pp. 435–445, 2017.

[23] D. Díaz-Pernil, M. A. Gutiérrez-Naranjo, M. J. Pérez-Jiménez, and A. Riscos-Núñez, "Solving subset sum in linear time by using tissue P systems with cell division," *Lecture Notes in Computer Science*, vol. 4527, no. 1, pp. 170–179, 2007.

[24] D. Díaz-Pernil, M. J. Pérez-Jiménez, A. Riscos-Núñez, and Á. Romero-Jiménez, "Computational efficiency of cellular division in tissue-like membrane systems," *Romanian Journal of Information, Science and Technology*, vol. 11, no. 3, pp. 229–241, 2008.

[25] I. Pérez-Hurtado, L. Valencia-Cabrera, J. M. Chacón, A. Riscos-Núñez, and M. J. Pérez-Jiménez, "A P-Lingua based simulator for tissue P systems with cell separation," *Romanian Journal of Information Science and Technology*, vol. 17, no. 1, pp. 89–102, 2014.

[26] M. J. Pérez-Jiménez and P. Sosík, "An optimal frontier of the efficiency of tissue P systems with cell separation," *Fundamenta Informaticae*, vol. 138, no. 1-2, pp. 45–60, 2015.

[27] A. Alhazov, A. Leporati, G. Mauri, A. E. Porreca, and C. Zandron, "Space complexity equivalence of P systems with active membranes and Turing machines," *Theoretical Computer Science*, vol. 529, pp. 69–81, 2014.

[28] A. Leporati, L. Manzoni, G. Mauri, A. E. Porreca, and C. Zandron, "Membrane division, oracles, and the counting hierarchy," *Fundamenta Informaticae*, vol. 138, no. 1-2, pp. 97–111, 2015.

[29] L. Pan and M. J. Pérez-Jiménez, "Computational complexity of tissue-like P systems," *Journal of Complexity*, vol. 26, no. 3, pp. 296–315, 2010.

[30] A. E. Porreca, N. Murphy, and M. J. Pérez-Jiménez, "An optimal frontier of the efficiency of tissue P systems with cell division," in *Proceedings of the Tenth Brainstorming Week on Membrane Computing*, pp. 141–166, Seville, Spain, 2012.

[31] R. Gutiérrez-Escudero, M. J. Pérez-Jiménez, and M. Rius-Font, "Characterizing tractability by tissue-like P systems," *Lecture Notes in Computer Science*, vol. 5957, pp. 289–300, 2009.

[32] L. F. Macías-Ramos, B. Song, L. Valencia-Cabrera, L. Pan, and M. J. Pérez-Jiménez, "Membrane fission: a computational complexity perspective," *Complexity*, vol. 21, no. 6, pp. 321–334, 2016.

[33] B. Song, C. Zhang, and L. Pan, "Tissue-like P systems with evolutional symport/antiport rules," *Information Sciences*, vol. 378, pp. 177–193, 2017.

[34] D. Díaz-Pernil, I. Pérez-Hurtado, M. J. Pérez-Jiménez, and A. Riscos-Núñez, "A P-lingua programming environment for membrane computing," *Lecture Notes in Computer Science*, vol. 5391, pp. 187–203, 2009.

[35] M. Garca-Quismondo, R. Gutiérrez-Escudero, I. Pérez-Hurtado, M. J. Pérez-Jiménez, and A. Riscos-Núñez, "An overview of P-Lingua 2.0," *Lecture Notes in Computer Science*, vol. 5957, pp. 264–288, 2010.

[36] P-Lingua, http://www.p-lingua.org/.

[37] I. Pérez-Hurtado, L. Valencia-Cabrera, M. J. Pérez-Jiménez, M. A. Colomer, and A. Riscos-Núñez, "MeCoSim: A general purpose software tool for simulating biological phenomena by means of P systems," in *Proceedings of the 2010 IEEE 5th International Conference on Bio-Inspired Computing: Theories and Applications, BIC-TA 2010*, K. Li, Z. Tang, R. Li, A. K. Nagar, and R. Thamburaj, Eds., vol. 1, pp. 637–643, IEEE Press, Changsha, China, September 2010.

[38] MeCoSim, http://www.p-lingua.org/mecosim/.

[39] G. Rozenberg and A. Salomaa, Eds., *Handbook of Formal Languages*, vol. 3, Springer, Berlin, Germany, 1997.

[40] M. J. Pérez Jiménez, A. Romero, and F. Caparrini, "Complexity classes in models of cellular computing with membranes," *Natural Computing*, vol. 2, no. 3, pp. 265–285, 2003.

[41] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman, Ed., San Francisco, Calif, USA, 1979.

[42] L. Pan, M. J. Pérez-Jiménez, A. Riscos-Núñez, and M. Rius-Font, "New frontiers of the efficiency in tissue P systems," in *Proceedings of the Asian Conference on Membrane Computing*, pp. 61–73, Wuhan, China, 2012.

[43] L. F. Macías-Ramos, M. J. Pérez-Jiménez, A. Riscos-Núñez, M. Rius-Font, and L. Valencia-Cabrera, "The efficiency of tissue P systems with cell separation relies on the environment," *Lecture Notes in Computer Science*, vol. 7762, pp. 243–256, 2013.

[44] L. Pan, Gh. Păun, and B. Song, "Flat maximal parallelism in P systems with promoters," *Theoretical Computer Science*, vol. 623, pp. 83–91, 2016.

[45] B. Song, M. J. Pérez-Jimenez, Gh. Păun, and L. Pan, "Tissue P systems with channel states working in the flat maximally parallel way," *IEEE Transactions on NanoBioscience*, vol. 15, no. 7, pp. 645–656, 2016.