Hindawi
Complexity
Volume 2021, Article ID 7120840, 14 pages
https://doi.org/10.1155/2021/7120840

Research Article

Tuning Frontiers of Efficiency in Tissue P Systems with Evolutional Communication Rules

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

[1]*Research Group on Natural Computing, Department of Computer Science and Artificial Intelligence, Universidad de Sevilla, Avda. Reina Mercedes s/n, 41012, Sevilla, Spain*
[2]*College of Information Science and Engineering, Hunan University, Changsha 410082, China*
[3]*Key Laboratory of Image Information Processing and Intelligent Control of Education Ministry of China, Institute of Artificial Intelligence, School of Artificial Intelligence and Automation, Huazhong University of Science and Technology, Wuhan 430074, Hubei, China*

Correspondence should be addressed to Linqiang Pan; lqpan@mail.hust.edu.cn

Received 2 November 2019; Accepted 31 March 2021; Published 28 April 2021

Academic Editor: Hocine Cherifi

Over the last few years, a new methodology to address the **P** versus **NP** problem has been developed, based on searching for borderlines between the nonefficiency of computing models (only problems in class **P** can be solved in polynomial time) and the presumed efficiency (ability to solve **NP**-complete problems in polynomial time). These borderlines can be seen as frontiers of efficiency, which are crucial in this methodology. "Translating," in some sense, an efficient solution in a presumably efficient model to an efficient solution in a nonefficient model would give an affirmative answer to problem **P** versus **NP**. In the framework of Membrane Computing, the key of this approach is to detect the syntactic or semantic ingredients that are needed to pass from a nonefficient class of membrane systems to a presumably efficient one. This paper deals with tissue P systems with communication rules of type symport/antiport allowing the evolution of the objects triggering the rules. In previous works, frontiers of efficiency were found in these kinds of membrane systems both with division rules and with separation rules. However, since they were not optimal, it is interesting to refine these frontiers. In this work, optimal frontiers of the efficiency are obtained in terms of the total number of objects involved in the communication rules used for that kind of membrane systems. These optimizations could be easier to translate, if possible, to efficient solutions in a nonefficient model.

1. Introduction

In biology, some basic requirements are proposed in order to define when something can be considered alive: replication, energy production, proteins synthesis, and metabolic processes. The minimal unit that meets these requirements is the cell. Usually, we find multiple cells forming larger structures, interchanging information between them in order to collaborate, improving in this way their lifespan. Membrane computing is a branch of natural computing inspired by the structure, organization, and functioning of living cells. In the seminal paper [1], computing devices are introduced having an internal structure of cell-like membranes (a rooted tree), delimiting compartments in which multisets of objects are placed. The objects evolve and pass through membranes in a synchronous parallel manner according to given evolution rules, also associated with the membranes. In [2], a new type of computing devices inspired by intercellular communication and cooperation between neurons was considered. They consist of a network of unit processors, cells, dealing with symbols and communicating these symbols along channels specified in advance, based on the biological phenomenon of trans-membrane transport of couples of chemicals. The synchronized movement of such chemical

substances present in a cell was modelled in [3]: groups of objects can pass together through a region either in the same (described by symport rules) or in the opposite direction (described by antiport rules). In these models, no change (evolution) occurs in the objects involved during their movement, that is, they remain unchanged after the application of symport/antiport rules. We refer the reader to [4, 5] for a further introduction to the field of membrane computing. Apart from applications in synthetic biology, ecology, engineering, and physics, among others (see [6–8] for a wide view of the state-of-the-art), membrane computing has been broadly used as a computational framework to study more theoretical areas such as computability theory and computational complexity theory.

The tractability of abstract problems is defined through polynomial-time solvability by deterministic Turing machines [9]. **P** is the class of tractable problems. A computing model is called efficient if it is capable of providing polynomial-time solutions to intractable problems. It is widely believed that $\mathbf{P} \neq \mathbf{NP}$, thus, **NP**-complete problems (the hard problems in **NP**) are called presumably intractable problems. A computing model is called presumably efficient if it is able to provide polynomial-time solutions to **NP**-complete problems.

Let us consider two models M_1 and M_1 in a computing paradigm (for instance, in membrane computing) such that M_2 is obtained from M_1 by adding some syntactic or semantic ingredients. If M_1 is a nonefficient model and M_2 is a presumably efficient one, then the ingredients needed to obtain M_2 from M_1 provide a frontier between the tractability of abstract problems and the presumed intractability. In some sense, passing from M_1 to M_2 amounts to passing from the nonefficiency to the presumed efficiency. Each such borderline provides a tool to tackle the **P** versus **NP** problem.

Decision problems are associated with languages in such a manner that solving a decision problem is defined by recognizing the language associated with it. Hence, recognizer membrane systems were defined in [10] (called decision P systems) and complexity classes associated with these systems were introduced in [11]. Over the last few years, the previous methodology for addressing the **P** versus **NP** problem has been applied in the framework of membrane computing. Specifically, in the framework of cell-like P systems with active membranes, a frontier between the nonefficiency and the presumed efficiency was obtained by forbidding division rules [12] or permitting them [13]. In the framework of tissue-like P systems with symport/antiport rules, (optimal) frontiers between the nonefficiency and the presumed efficiency, in terms of both the length of the rules and the kinds of rules able to construct an exponential workspace in terms of cells, have been obtained [14–17].

In [18], a new kind of tissue P systems based on the communication of cells within a living tissue is studied, where objects can evolve when the rules are applied. Specifically, tissue P systems with division rules using evolutional symport/antiport rules have been considered. In that paper, the length of a communication rule is defined as the total number of objects involved in it. For each natural number $n \geq 1$, $\mathcal{TDEC}(n)$ denotes the class of recognizer tissue P systems with division rules using evolutional communication rules of length at most n. The nonefficiency of the computing model $\mathcal{TDEC}(2)$ has been established by using the dependency graph technique, and a polynomial-time uniform solution to the SAT problem by a family of membrane systems from $\mathcal{TDEC}(4)$ has been given in [18]. Therefore, passing from evolutional communication rules of length at most 2 to evolutional communication rules of length at most 4 amounts to passing from the nonefficiency to the presumed efficiency. The following question then arises: What is the efficiency of the complexity class $\mathbf{PMC}_{\mathcal{TDEC}(3)}$?

In [19], tissue P systems with evolutional symport/antiport rules are studied, where separation rules instead of division rules are used as a mechanism to create an exponential workspace in polynomial time. In that paper, for each pair of natural numbers $k_1 \geq 1, k_2 \geq 1$, $\mathcal{TDEC}(k_1, k_2)$ denotes the class of recognizer tissue P systems with separation rules using evolutional communication rules such that the total number of objects involved in the left-hand side (LHS) is at most k_1 and the total number of objects involved in the right-hand side (RHS) is at most k_2. The nonefficiency of $\mathcal{TDEC}(1, n)$ and $\mathcal{TDEC}(n, 1)$, for each $n \geq 1$, has been established using the algorithmic technique. On the other hand, the presumed efficiency of $\mathcal{TDEC}(3, 2)$ has been shown.

The reader is supposed to have knowledge about languages, multisets, and other questions usually used in this kind of works. In any case, for a gentle introduction to them, we refer the reader to [20].

In this work, we deal with tissue P systems with evolutional symport/antiport rules where division rules or separation rules are used as mechanisms to create an exponential workspace in polynomial time. Following [19], for each pair of natural numbers $k_1 \geq 1, k_2 \geq 1$, the class $\mathcal{TDEC}(k_1, k_2)$ is defined similarly to $\mathcal{TDEC}(k_1, k_2)$, replacing separation rules by division rules. Following [18], for each natural number $n \geq 1$, $\mathcal{TDEC}(n)$ is defined similarly to $\mathcal{TDEC}(n)$, replacing division by separation rules. For each pair of natural numbers $k_1 \geq 1, k_2 \geq 1$, we have $\mathcal{TDEC}(k_1, k_2) \subseteq \mathcal{TDEC}(k_1 + k_2)$ and $\mathcal{TDEC}(k_1, k_2) \subseteq \mathcal{TDEC}(k_1 + k_2)$.

This paper focuses on the nonefficiency of $\mathcal{TDEC}(1, n)$, for each $n \geq 1$, and the presumed efficiency of $\mathcal{TDEC}(2, 1)$ and $\mathcal{TDEC}(2, 2)$. Specifically, we prove the following results: (1) For each natural number $n \geq 1$, we have $\mathbf{P} = \mathbf{PMC}_{\mathcal{TDEC}(1,n)}$; (2) the computing model $\mathcal{TDEC}(2, 1)$ is presumably efficient, that is, $\mathbf{NP} \cup \mathbf{co\text{-}NP} \subseteq \mathbf{PMC}_{\mathcal{TDEC}(2,1)}$; (3) the computing model $\mathcal{TDEC}(2, 2)$ is presumably efficient, that is, $\mathbf{NP} \cup \mathbf{co\text{-}NP} \subseteq \mathbf{PMC}_{\mathcal{TDEC}(2,2)}$. Bearing in mind that $\mathcal{TDEC}(2, 1) \subseteq \mathcal{TDEC}(3) \subseteq \mathcal{TDEC}(4)$ and $\mathcal{TDEC}(2, 2) \subseteq \mathcal{TDEC}(3, 2)$, the presumed efficiency of $\mathcal{TDEC}(3)$, $\mathcal{TDEC}(4)$, and $\mathcal{TDEC}(3, 2)$ follows, closing the question raised about $\mathcal{TDEC}(3)$, among others. In this manner, the results obtained in [18, 19] are complemented and improved.

This paper is organized as follows: Tissue P systems with division or separation rules and evolutional symport/antiport rules are defined in Section 2. The nonefficiency of

$\mathcal{TDEC}(1,n)$, for each $n \geq 1$, is established in Section 3, and two polynomial-time uniform solutions to the SAT problem by families of systems from $\mathcal{TDEC}(2,1)$ and $\mathcal{TSEC}(2,2)$ are provided in Sections 4 and 5, respectively. Finally, some conclusions and interesting open research lines are presented.

2. Tissue P Systems with Evolutional Communication Rules

Let us briefly recall the definition of recognizer tissue P systems with evolutional symport/antiport rules using division or separation rules as a mechanism to construct an exponential workspace in polynomial time.

Definition 1. A recognizer tissue P system with division/separation rules and evolutional communication rules of degree $q \geq 1$ (where q is the number of cells present in the system) is a tuple

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

where

(i) $\Gamma, \mathscr{E}, \Sigma$ are finite alphabets such that $\mathscr{E} \subseteq \Gamma$, $\Sigma \subseteq \Gamma \setminus \mathscr{E}$, and the alphabet Γ has two distinguished objects yes and no

(ii) $\{\Gamma_0, \Gamma_1\}$ is a partition of Γ; that is, $\Gamma_0, \Gamma_1 \neq \emptyset, \Gamma_0 \cap \Gamma_1 = \emptyset, \Gamma_0 \cup \Gamma_1 = \Gamma$

(iii) $\mathscr{M}_1, \ldots, \mathscr{M}_q$ are multisets over $\Gamma \setminus \Sigma$

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

(1) Evolutional communication rules:
(a) $[u]_{ij} \longrightarrow_i [u']_j$, where $i, j \in \{0,\ldots,q\}$, $i \neq j$, $u \in M^+(\Gamma)$, $u' \in M(\Gamma)$, and if $i=0$, then alph $(u) \cap (\Gamma \setminus \mathscr{E}) \neq \emptyset$ (evolutionalsymportrules)
(b) $[u]_i[v]_j \longrightarrow [v']_i[u']_j$, where $i, j \in \{0,\ldots,q\}$, $i \neq j, u, v \in M^+(\Gamma), u', v' \in M(\Gamma)$ (evolutionalantiportrules)

(2) Division rules: $[a]_i \longrightarrow [b]_i[c]_i$, where $i \in \{1,\ldots,q\}, i \neq i_{\text{out}}, a, b, c \in \Gamma$

(3) Separation rules: $[a]_i \longrightarrow [\Gamma_0]_i[\Gamma_1]_i$, where $i \in \{1,\ldots,q\}, i \neq i_{\text{out}}, a \in \Gamma$

(v) $i_{\text{in}} \in \{1,\ldots,q\}$ and $i_{\text{out}} = 0$ (that is, the label of the environment)

(vi) If $\mathscr{C}$ is a computation of Π then $\mathscr{C}$ is a halting computation and either object yes or object no (but not both) must have been released into the output region, and only at the last step of the computation

The semantic concepts (configuration, transition step, and computation) in this kind of recognizer membrane systems can be found in [18, 19]. Only division rules or separation rules (but not both) are considered, and in the case of recognizer tissue P systems with division rules, the partition $\{\Gamma_0, \Gamma_1\}$ can be removed from the expression.

In [18], the length of an evolutional communication rule r is defined as the total number of objects involved in the rule. For each natural number $n \geq 1$, $\mathcal{TDEC}(n)$ (resp., $\mathcal{TSEC}(n)$) denotes the class of recognizer tissue P systems with division rules (resp., separation rules) using evolutional communication rules of length at most n.

Following [19], for each pair of natural numbers $k_1 \geq 1, k_2 \geq 1$, $\mathcal{TDEC}(k_1, k_2)$ (resp., $\mathcal{TSEC}(k_1, k_2)$) denotes the class of recognizer tissue P systems with division rules (resp., separation rules) using evolutional communication rules such that the total number of objects involved in the left-hand side (LHS) (that is, the objects before the arrow in evolutional communication rules) is at most k_1 and the total number of objects involved in the right-hand side (RHS) (that is, the objects after the arrow in evolutional communication rules) is at most k_2. Obviously, for each pair of natural numbers $k_1 \geq 1, k_2 \geq 1$, we have $\mathcal{TDEC}(k_1, k_2) \subseteq \mathcal{TDEC}(k_1 + k_2)$ and $\mathcal{TSEC}(k_1, k_2) \subseteq \mathcal{TSEC}(k_1 + k_2)$. It is important to remark that, in this type of study, we do not take into account the number of objects implicitly involved neither in division nor in separation rules.

3. The Nonefficiency of $\mathcal{TDEC}(1,n)$

The nonefficiency of $\mathcal{TSEC}(1,n)$, for each natural number $n \geq 1$, has been shown by using the dependency graph technique in [19]. Let us recall that this technique consists in the following: if $\{\Pi(t) | t \in \mathbb{N}\}$ is a family of systems from $\mathcal{TSEC}(1,n)$ solving a decision problem X in polynomial time and uniform way, then a directed graph $G_{\Pi(s(u))+\text{cod}(u)}$ (with a source node) is associated with each instance $u \in I_X$, in such a manner that every computation $\mathscr{C}$ of $\Pi(s(u)) + \text{cod}(u)$ is an accepting computation if and only if there exists a path in $G_{\Pi(s(u))+\text{cod}(u)}$ from the source node to (yes, env). When working with membrane systems $\mathcal{TSEC}(1,n)$, each separation rule of the form $[a]_i \longrightarrow [\Gamma_0]_i[\Gamma_1]_i$ provides a single node (a,i) to the corresponding dependency graph, but no new arc is added since no new objects are created by the application of such a rule. In the case of membrane systems from $\mathcal{TDEC}(1,n)$, we can adapt the proof given in [19], taking into account the fact that a division rule $[a]_i \longrightarrow [a]_i[b]_i$ provides three new nodes, (a,i), (b,i), and (c,i) to the corresponding dependency graph, and two new arcs: $((a,i),(b,i))$ and $((a,i),(c,i))$. Of course, this process keeps having a polynomially bounded number of nodes and arcs, so the search for a path in the dependency graph from the source node to (yes, env) is still polynomial-time bounded. Therefore, for each natural number $n \geq 1$, the nonefficiency of $\mathcal{TDEC}(1,n)$ follows.

Theorem 1. *For each natural number $n \geq 1$, we have* $\mathbf{P} = \mathbf{PMC}_{\mathcal{TDEC}(1,n)}$.

4. A Solution to SAT in $\mathcal{TDEC}(2,1)$

We provide here a solution to SAT by means of a family of recognizer tissue P systems $\Pi = \{\Pi(t) | t \in \mathbb{N}\}$ from $\mathcal{TDEC}(2,1)$. Given a Boolean formula φ in CNF and simplified with n variables and p clauses, the system $\Pi(s(\varphi)) + \text{cod}(\varphi)$ processes it, being $s(\varphi) = \langle n, p \rangle$

and $\operatorname{cod}(\varphi) = \{x_{i,j,0} | x_i \in C_j\} \cup \{\overline{x}_{i,j,0} | x_i \in C_j\} \cup \{x^*_{i,j,0} | x_i \notin C_j, x_i \notin C_j\}$ and returns the answer to its satisfiability.

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

$$\Pi(\langle n, p\rangle) = (\Gamma, \mathscr{E}, \Sigma, \mathscr{M}_1, \ldots, \mathscr{M}_{np+3}, \mathscr{R}, i_{\text{in}}, i_{\text{out}}), \quad (2)$$

where we have the following:

(i) Working alphabet Γ:

$$\{\texttt{yes}, \texttt{no}, y_1, y_2, n_1, n_2, n_3, \#\} \cup \{a_{i,j}, T_{i,j}, F_{i,j} | 1 \le i \le n, 1 \le j \le p\} \cup \{x_{i,j,k}, \overline{x}_{i,j,k}, x^*_{i,j,k} | 1 \le i \le n, 1 \le j \le p, 0 \le k \le np+1\}$$
$$\cup \{\alpha_j \mid 1 \le j \le p+1\} \cup \{\delta_k \mid 0 \le k \le np+4\} \cup \{\delta'_k \mid 0 \le k \le np+2\} \quad (3)$$

(ii) Input alphabet $\Sigma = \{x_{i,j,0}, \overline{x}_{i,j,0}, x^*_{i,j,0} | 1 \le i \le n, 1 \le j \le p\}$.

(iii) Environment alphabet $\mathscr{E} = \{\gamma\}$.

(iv) $\mathscr{M}_k = \varnothing, 1 \le k \le np$; $\mathscr{M}_{np+1} = \{\delta_0\}$;
$\mathscr{M}_{np+2} = \{a_{i,j} | 1 \le i \le n, 1 \le j \le p\} \cup \{\alpha_j | 1 \le j \le p+1\}$;
$\mathscr{M}_{np+3} = \{\delta'_0\}$

(v) The rule set $\mathscr{R}$ consists of the following rules:

(1) Rules to generate p copies of the 2^n true possible truth assignments. For this, 2^{np} partial truth assignments will be generated.

$$\left.\begin{aligned} &[a_{i,j}]_{np+2} \longrightarrow [T_{i,j}]_{np+2}[F_{i,j}]_{np+2} \\ &[T_{i,j}F_{i,j'}]_{np+2}[\,]_0 \longrightarrow [\,]_{np+2}[\#]_0 \end{aligned}\right\} \quad \text{for } 1 \le i \le n, 1 \le j, j' \le p. \quad (4)$$

(2) Rules to generate 2^{np} copies of $\operatorname{cod}(\varphi)$.

$$\left.\begin{aligned} &[x_{i,j,0}]_{np+1}[\,]_{i+n\cdot(j-1)} \longrightarrow [\,]_{np+1}[x_{i,j,1}]_{i+n\cdot(j-1)} \\ &[\overline{x}_{i,j,0}]_{np+1}[\,]_{i+n\cdot(j-1)} \longrightarrow [\,]_{np+1}[\overline{x}_{i,j,1}]_{i+n\cdot(j-1)} \\ &[x^*_{i,j,0}]_{np+1}[\,]_{i+n\cdot(j-1)} \longrightarrow [\,]_{np+1}[x^*_{i,j,1}]_{i+n\cdot(j-1)} \end{aligned}\right\} \quad \text{for } \begin{matrix} 1 \le i \le n, \\ 1 \le j \le p, \end{matrix}$$

$$\left.\begin{aligned} &[x_{i,j,k}]_{i+n\cdot(j-1)} \longrightarrow [x_{i,j,k+1}]_{i+n\cdot(j-1)}[x_{i,j,k+1}]_{i+n\cdot(j-1)} \\ &[\overline{x}_{i,j,k}]_{i+n\cdot(j-1)} \longrightarrow [\overline{x}_{i,j,k+1}]_{i+n\cdot(j-1)}[\overline{x}_{i,j,k+1}]_{i+n\cdot(j-1)} \\ &[x^*_{i,j,k}]_{i+n\cdot(j-1)} \longrightarrow [x^*_{i,j,k+1}]_{i+n\cdot(j-1)}[x^*_{i,j,k+1}]_{i+n\cdot(j-1)} \end{aligned}\right\} \quad \begin{matrix} 1 \le i \le n, \\ \text{for } 1 \le j \le p, \\ 1 \le k \le np, \end{matrix} \quad (5)$$

$$[\delta'_0]_{np+3}[\gamma]_0 \longrightarrow [\delta'_1]_{np+3}[\,]_0,$$
$$[\delta'_k]_{np+3} \longrightarrow [\delta'_{k+1}]_{np+3}[\delta'_{k+1}]_{np+3} \quad \text{for } 1 \le k \le np,$$
$$[\delta'_{np+1}]_{np+3}[\gamma]_0 \longrightarrow [\delta'_{np+2}]_{np+3}[\,]_0.$$

(3) Rules to check which clauses are satisfied by the truth assignments.

$$\left.\begin{array}{l}
\left[T_{i,j}\right]_{np+2}\left[x_{i,j,np+1}\right]_{i+n\cdot(j-1)} \longrightarrow \left[c_j\right]_{np+2}[\,]_{i+n\cdot(j-1)}\\
\left[T_{i,j}\right]_{np+2}\left[\overline{x}_{i,j,np+1}\right]_{i+n\cdot(j-1)} \longrightarrow [\#]_{np+2}[\,]_{i+n\cdot(j-1)}\\
\left[T_{i,j}\right]_{np+2}\left[x^*_{i,j,np+1}\right]_{i+n\cdot(j-1)} \longrightarrow [\#]]_{np+2}[\,]_{i+n\cdot(j-1)}\\
\left[F_{i,j}\right]_{np+2}\left[x_{i,j,np+1}\right]_{i+n\cdot(j-1)} \longrightarrow [\#]_{np+2}[\,]_{i+n\cdot(j-1)}\\
\left[F_{i,j}\right]_{np+2}\left[\overline{x}_{i,j,np+1}\right]_{i+n\cdot(j-1)} \longrightarrow \left[c_j\right]_{np+2}[\,]_{i+n\cdot(j-1)}\\
\left[F_{i,j}\right]_{np+2}\left[x^*_{i,j,np+1}\right]_{i+n\cdot(j-1)} \longrightarrow [\#]_{np+2}[\,]_{i+n\cdot(j-1)}
\end{array}\right\} \quad \text{for } \begin{array}{l}1\le i\le n,\\ 1\le j\le p.\end{array} \tag{6}$$

(4) Rules to check if all the clauses are satisfied by a truth assignment.

$$\begin{aligned}
&\left[\alpha_{p+1}\right]_{np+2}\left[\delta'_{np+2}\right]_{np+3} \longrightarrow \left[\alpha'_{p+1}\right]_{np+2}[\,]_{np+3},\\
&\left[c_j\alpha_j\right]_{np+2}[\,]_{np+3} \longrightarrow [\,]_{np+2}[\#]_{np+3} \quad \text{for } 1\le j\le p.
\end{aligned} \tag{7}$$

(5) General counter.

$$\begin{aligned}
&[\delta_k]_{np+1}[\gamma]_0 \longrightarrow [\delta_{k+1}]_{np+1}[\,]_0\\
&0\le k\le np+3
\end{aligned}. \tag{8}$$

(6) Rules to return a negative answer.

$$\begin{aligned}
&\left[\alpha_j\alpha'_{p+1}\right]_{np+2}[\,]_0 \longrightarrow [\,]_{np+2}[n_1]_0 \quad \text{for } 1\le j\le p,\\
&[n_1]_0[\,]_{np+2} \longrightarrow [\,]_0[n_2]_{np+2},\\
&[n_2]_{np+2}\left[\delta_{np+4}\right]_{np+1} \longrightarrow [n_3]_{np+2}[\,]_{np+1},\\
&[n_3]_{np+2}[\,]_0 \longrightarrow [\,]_{np+2}[\mathtt{no}]_0.
\end{aligned} \tag{9}$$

(7) Rules to return a positive answer.

$$\begin{aligned}
&\left[\alpha'_{p+1}\right]_{np+2}\left[\delta_{np+4}\right]_{np+1} \longrightarrow [y_1]_{np+2}[\,]_{np+1},\\
&[y_1]_{np+2}[\gamma]_0 \longrightarrow [y_2]_{np+2}[\,]_0,\\
&[y_2]_{np+2}[\,]_0 \longrightarrow [\,]_{np+2}[\mathtt{yes}]_0.
\end{aligned} \tag{10}$$

(vi) The input cell is the cell labelled by $np+1$ ($i_{\text{in}} = np+1$) and the output zone is the environment ($i_{\text{out}} = \text{env}$).

Let φ be a Boolean formula in simplified CNF. The P system $\Pi(s(\varphi)) + \text{cod}(\varphi)$ will follow a brute-force algorithm in the framework of tissue P systems with evolutional symport/antiport rules with length, at most, $(2,1)$ and division rules, and it consists of the following stages:

(1) Generation stage: using rules from (1), p copies of the 2^n possible truth assignments associated with variables $\{x_1,\ldots,x_n\}$ will be generated in cells labelled by $np+2$. For this, $2^{np}-2^n$ "partial" truth assignments will be generated too; that is, some cells will contain different assignments for a single variable, but these objects will be "cancelled" by the application of the rule $[T_{i,j}F_{i,j'}]_{np+20} \longrightarrow {}_{np+2}[\#]_0$. In parallel, 2^{np} copies of each object $x_{i,j,np+1}$, $\overline{x}_{i,j,np+1}$, $x^*_{i,j,np+1}$ that appears in $\text{cod}(\varphi)$ will be generated by applying rules from (2). In order to synchronize the process, the third subscript, k, will "evolve" until reaching $np+1$. This stage takes exactly $np+1$ computation steps.

(2) First checking stage: in this stage, by means of the application of rules from (3), the p copies of each one of the 2^n possible truth assignments associated with variables $\{x_1,\ldots,x_n\}$ cooperate with objects $x_{i,j,np+1}$, $\overline{x}_{i,j,np+1}$, $x^*_{i,j,np+1}$ by using rules from (3) in order to know which clauses are satisfied by each truth assignment. It is worth pointing out that through this checking stage, the $2^{np}-2^n$ partial truth assignments cannot satisfy φ if it is unsatisfiable. This stage takes 1 computation step.

(3) Second checking stage: by using rules from (4), object α_j is removed from a cell labelled as $np+2$ if and only if the truth assignment associated with this cell makes the clause C_j true. Therefore, a cell labelled as $np+2$ will encode a truth assignment that makes φ true if and only if it does not contain any object α_j at the end of this stage, taking exactly 1 computation step.

(4) Output stage: finally, rules from (5)–(7) will send either an object `yes` or an object `no` to the environment, depending on whether the formula is satisfactory or not. This stage takes exactly 4 computation steps, regardless of the answer being affirmative or negative.

4.1. Formal Verification. In this section, an exhaustive verification of the system is given.

4.1.1. Generation Stage. The purpose of this stage is twofold: on the one hand, 2^{np} objects of each $l_{i,j,np+1}$, $l \in \{x,\overline{x},x^*\}$, $1\le i\le n$, $1\le j\le p$ will be generated in cells $i+n\cdot(j-1)$. On the other hand, 2^n valid truth assignments will be generated in cells labelled by $np+2$. For that, 2^{np} of such cells will be generated, each of them containing a truth assignment. Of course, there will be repeated truth assignments, but in this framework, due to the restrictions of the length of the rules, it seems necessary to create all the possible assignments (including the ones where

two different truth values are assigned to a single variable), and then "remove" the incompatibilities.

Proposition 1. *Let $\mathscr{C} = (\mathscr{C}_0, \mathscr{C}_1, \ldots, \mathscr{C}_q)$ be a computation of the system $\Pi(s(\varphi))$ with input multiset $cod(\varphi)$.*

(i) For each k $(0 \le k \le np)$ at configuration $\mathscr{C}_k$, we have the following:

(1) $\mathscr{C}_0(i + n \cdot (j-1)) = \varnothing$ and there are 2^{k-1} cells labelled by $i + n \cdot (j-1)$ in the configuration $\mathscr{C}_k$ such that each of them contains an object $l_{i,j,k}$, being $l = x$ if x_i is a literal of the clause C_j, $\overline{x}$ if x_i is a literal of the clause C_j, and x^* if neither x_i nor x_i are literals of the clause C_j, for $k \ge 1$

(2) $\mathscr{C}_0(np+1) = \{\delta_k\} \cup cod(\varphi)$ and $\mathscr{C}_k(np+1) = \{\delta_k\}$ for $k > 0$

(3) There are 2^k cells labelled by $np+2$ in the configuration $\mathscr{C}_k$ such that each of them contains the set $\{\alpha_1, \ldots, \alpha_{p+1}\}$, $np-k$ objects from the set $\{a_{1,1}, \ldots, a_{1,p}, \ldots, a_{n,1}, \ldots, a_{n,p}\}$ and less than k or k objects from the set $\{R_{1,1}, \ldots, R_{1,p}, \ldots, R_{n,1}, \ldots, R_{n,p}\}$, $R \in \{T, F\}$

(4) $\mathscr{C}_0(np+3) = \{\delta_0'\}$ and there are 2^{k-1} cells labelled by $np+3$ in the configuration $\mathscr{C}_k$ such that each of them contains an object δ_k', for $k \ge 1$

(ii) In the configuration $\mathscr{C}_{np+1}$, we have that there are 2^{np} cells labelled by $i + n \cdot (j-1)$ for each $1 \le i \le n, 1 \le j \le p$ such that each of them contains an object $l_{i,j,np+1}$ being $l = x$ if x_i is a literal of the clause C_j, $\overline{x}$ if x_i is a literal of the clause C_j, and x^* if neither x_i nor x_i are literals of the clause C_j, $\mathscr{C}_{np+1}(np+1) = \{\delta_{np+1}\}$; there are 2^{np} cells labelled by $np+2$ such that each of them contains the set $\{\alpha_1, \ldots, \alpha_{p+1}\}$, less than np or np objects from the set $\{R_{1,1}, \ldots, R_{1,p}, \ldots, R_{n,1}, \ldots, R_{n,p}\}$, $R \in \{T, F\}$ and there are 2^{np} cells labelled by $np+3$ such that each of them contains an object δ_{np+1}'

Proof. (i) is going to be proved by induction on k

(i) The base case $k = 0$ is trivial because at the initial configuration, we have the following:

(1) $\mathscr{C}_0(i + n \cdot (j-1)) = \varnothing$
(2) $\mathscr{C}_0(np+1) = \{\delta_0\} \cup \mathrm{cod}(\varphi)$
(3) There is one cell labelled by $np+2$ such that contains the set $\{\alpha_1, \ldots, \alpha_{p+1}\}$ and the set $\{a_{1,1}, \ldots, a_{1,p}, \ldots, a_{n,1}, \ldots, a_{n,p}\}$
(4) There is one cell labelled by $np+3$ such that it contains an object δ_0'

Then, configuration $\mathscr{C}_0$ yields configuration $\mathscr{C}_1$ by applying the rules:

$$\left.\begin{array}{l} \left[a_{i,j}\right]_{np+2} \longrightarrow \left[T_{i,j}\right]_{np+2}\left[F_{i,j}\right]_{np+2} \\ \left[x_{i,j,0}\right]_{np+1}[\,]_{i+n\cdot(j-1)} \longrightarrow [\,]_{np+1}\left[x_{i,j,1}\right]_{i+n\cdot(j-1)} \\ \left[\overline{x}_{i,j,0}\right]_{np+1}[\,]_{i+n\cdot(j-1)} \longrightarrow [\,]_{np+1}\left[x_{i,j,1}\right]_{i+n\cdot(j-1)} \\ \left[x^*_{i,j,0}\right]_{np+1}[\,]_{i+n\cdot(j-1)} \longrightarrow [\,]_{np+1}\left[x^*_{i,j,1}\right]_{i+n\cdot(j-1)} \\ \left[\delta_0'\right]_{np+3}[\gamma]_0 \longrightarrow \left[\delta_1'\right]_{np+3}[\,]_0 \\ \left[\delta_0\right]_{np+1}[\gamma]_0 \longrightarrow \left[\delta_1\right]_{np+1}[\,]_0 \end{array}\right\} \quad \text{for } \begin{array}{l} 1 \le i \le n, \\ 1 \le j \le p. \end{array} \tag{11}$$

Thus, we have the following:

(1) $\mathscr{C}_1(i + n \cdot (j-1)) = \{l_{i,j,1}\}$, being $l = x$ if x_i is a literal of the clause C_j, $\overline{x}$ if x_i is a literal of the clause C_j and x^* if neither x_i nor x_i are literals of the clause C_j
(2) $\mathscr{C}_1(np+1) = \{\delta_1\}$
(3) There are two cells labelled by $np+2$ such that contains the set $\{\alpha_1, \ldots, \alpha_{p+1}\}$, $np-1$ objects of the set $\{a_{1,1}, \ldots, a_{1,p}, \ldots, a_{n,1}, \ldots, a_{n,p}\}$ and one object $T_{i,j}$ and one object $F_{i,j}$, respectively, for a $i, 1 \le i \le n$ and a $j, 1 \le j \le p$
(4) There is one cell labelled by $np+3$ such that it contains an object δ_1'

(ii) Assuming that, by induction, the result is true for k $(0 \le k \le np)$; that is, that we have the following:

(1) $\mathscr{C}_0(i + n \cdot (j-1)) = \varnothing$ and there are 2^{k-1} cells labelled by $i + n \cdot (j-1)$ in the configuration $\mathscr{C}_k$ such that each of them contains an object $l_{i,j,k}$, being $l = x$ if x_i is a literal of the clause C_j, $\overline{x}$ if x_i is a literal of the clause C_j and x^* if neither x_i nor x_i are literals of the clause C_j, for $k \ge 1$
(2) $\mathscr{C}_0(np+1) = \{\delta_k\} \cup \mathrm{cod}(\varphi)$ and $\mathscr{C}_k(np+1) = \{\delta_k\}$ for $k > 0$
(3) There are 2^k cells labelled by $np+2$ in the configuration $\mathscr{C}_k$ such that each of them contains the set $\{\alpha_1, \ldots, \alpha_{p+1}\}$, $np-k$ objects from the set $\{a_{1,1}, \ldots, a_{1,p}, \ldots, a_{n,1}, \ldots, a_{n,p}\}$ and less than k or k objects from the set $\{R_{1,1}, \ldots, R_{1,p}, \ldots, R_{n,1}, \ldots, R_{n,p}\}$, $R \in \{T, F\}$
(4) $\mathscr{C}_0(np+3) = \{\delta_0'\}$ and there are 2^{k-1} cells labelled by $np+3$ in the configuration $\mathscr{C}_k$ such that each of them contains an object δ_k' for $k \ge 1$

Then, configuration $\mathscr{C}_k$ yields configuration $\mathscr{C}_{k+1}$ by applying the rules:

$$\left.\begin{array}{l}\left[a_{i,j}\right]_{np+2} \longrightarrow \left[T_{i,j}\right]_{np+2}\left[F_{i,j}\right]_{np+2}\\ \left[T_{i,j}F_{i,j'}\right]_{np+2}[\,]_0 \longrightarrow [\,]_{np+2}[\#]_0\end{array}\right\} \quad \text{for } 1 \le i \le n, 1 \le j, j' \le p,$$

$$\left.\begin{array}{l}\left[x_{i,j,k}\right]_{i+n\cdot(j-1)} \longrightarrow \left[x_{i,j,k+1}\right]_{i+n\cdot(j-1)}\left[x_{i,j,k+1}\right]_{i+n\cdot(j-1)}\\ \left[\overline{x}_{i,j,k}\right]_{i+n\cdot(j-1)} \longrightarrow \left[\overline{x}_{i,j,k+1}\right]_{i+n\cdot(j-1)}\left[\overline{x}_{i,j,k+1}\right]_{i+n\cdot(j-1)}\\ \left[x^*_{i,j,k}\right]_{i+n\cdot(j-1)} \longrightarrow \left[x^*_{i,j,k+1}\right]_{i+n\cdot(j-1)}\left[x^*_{i,j,k+1}\right]_{i+n\cdot(j-1)}\end{array}\right\} \quad \text{for } \begin{array}{l}1 \le i \le n,\\ 1 \le j \le p,\end{array} \tag{12}$$

$$\left[\delta'_k\right]_{np+3} \longrightarrow \left[\delta'_{k+1}\right]_{np+3}\left[\delta'_{k+1}\right]_{np+3},$$

$$\left[\delta_k\right]_{np+1}[\gamma]_0 \longrightarrow \left[\delta_{k+1}8i\right]_{np+1}[\,]_0.$$

Thus, we have the following:

(1) There are 2^k cells labelled by $i + n \cdot (j-1)$ in the configuration $\mathscr{C}_k$ such that each of them contains an object $l_{i,j,k+1}$, being $l = x$ if x_i is a literal of the clause C_j, $\overline{x}$ if x_i is a literal of the clause C_j and x^* if neither x_i nor x_i are literals of the clause C_j

(2) $\mathscr{C}_{k+1}(np+1) = \{\delta_{k+1}\}$

(3) There are 2^{k+1} cells labelled by $np+2$ in the configuration $\mathscr{C}_{k+1}$ such that each of them contains the set $\{\alpha_1, \ldots, \alpha_{p+1}\}$, $np-(k+1)$ objects from the set $\{a_{1,1}, \ldots, a_{1,p}, \ldots, a_{n,1}, \ldots, a_{n,p}\}$ and less than $k+1$ or $k+1$ objects from the set $\{R_{1,1}, \ldots, R_{1,p}, \ldots, R_{n,1}, \ldots, R_{n,p}\}$, $R \in \{T, F\}$

(4) There are 2^k cells labelled by $np+3$ in the configuration $\mathscr{C}_{k+1}$ such that each of them contains an object $\delta_{k+1}{}'$

(iii) In order to prove (ii) it is enough to take into account the fact that, from (i), we have the following:

(1) There are 2^{np-1} cells labelled by $i + n \cdot (j-1)$ in the configuration $\mathscr{C}_{np}$ such that each of them contains an object $l_{i,j,np}$, being $l = x$ if x_i is a literal of the clause C_j, $\overline{x}$ if x_i is a literal of the clause C_j and x^* if neither x_i nor x_i are literals of the clause C_j

(2) $\mathscr{C}_{np}(np+1) = \{\delta_{np}\}$

(3) There are 2^{np} cells labelled by $np+2$ in the configuration $\mathscr{C}_{np}$ such that each of them contains the set $\{\alpha_1, \ldots, \alpha_{p+1}\}$ and less than np or np objects from the set $\{R_{1,1}, \ldots, R_{1,p}, \ldots, R_{n,1}, \ldots, R_{n,p}\}$, $R \in \{T, F\}$, representing a truth assignment

(4) There are 2^{np-1} cells labelled by $np+3$ in the configuration $\mathscr{C}_{np}$ such that each of them contains an object $\delta_{np}{}'$

Then, configuration $\mathscr{C}_{np}$ yields configuration $\mathscr{C}_{np+1}$ by applying the rules:

$$\left[T_{i,j}F_{i,j'}\right]_{np+2}[\,]_0 \longrightarrow [\,]_{np+2}[\#]_0 \quad \text{for } 1 \le i \le n, 1 \le j, j' \le p,$$

$$\left.\begin{array}{l}\left[x_{i,j,np}\right]_{i+n\cdot(j-1)} \longrightarrow \left[x_{i,j,np+1}\right]_{i+n\cdot(j-1)}\left[x_{i,j,np+1}\right]_{i+n\cdot(j-1)}\\ \left[\overline{x}_{i,j,np}\right]_{i+n\cdot(j-1)} \longrightarrow \left[\overline{x}_{i,j,np+1}\right]_{i+n\cdot(j-1)}\left[\overline{x}_{i,j,np+1}\right]_{i+n\cdot(j-1)}\\ \left[x^*_{i,j,np}\right]_{i+n\cdot(j-1)} \longrightarrow \left[x^*_{i,j,np+1}\right]_{i+n\cdot(j-1)}\left[x^*_{i,j,np+1}\right]_{i+n\cdot(j-1)}\end{array}\right\} \quad \text{for } \begin{array}{l}1 \le i \le n,\\ 1 \le j \le p,\end{array} \tag{13}$$

$$\left[\delta_{np}{}'\right]_{np+3} \longrightarrow \left[\delta_{np+1}{}'\right]_{np+3}\left[\delta_{np+1}{}'\right]_{np+3},$$

$$\left[\delta_{np}\right]_{np+1}[\gamma]_0 \longrightarrow \left[\delta_{np+1}\right]_{np+1}]_0.$$

Then, we have the following:

(1) There are 2^{np} cells labelled by $i + n \cdot (j-1)$ in the configuration $\mathscr{C}_{np+1}$ such that each of them contains an object $l_{i,j,np+1}$, being $l = x$ if x_i is a literal of the clause C_j, $\overline{x}$ if x_i is a literal of the clause C_j and x^* if neither x_i nor x_i are literals of the clause C_j

(2) $\mathscr{C}_{np+1}(np+1) = \{\delta_{np+1}\}$

(3) There are 2^{np} cells labelled by $np+2$ in the configuration $\mathscr{C}_{np+1}$ such that each of them

contains the set $\{\alpha_1, \ldots, \alpha_{p+1}\}$ and less than np or np objects from the set $\{R_{1,1}, \ldots, R_{1,p}, \ldots, R_{n,1}, \ldots, R_{n,p}\}$, $R \in \{T, F\}$, representing a truth assignment

(4) There are 2^{np} cells labelled by $np+3$ in the configuration $\mathscr{C}_{np+1}$ such that each of them contains an object $\delta_{np+1}{}'$ □

4.1.2. First Checking Stage. From the previous stage, we have that $\mathscr{C}_{np+1}(np+1) = \{\delta_{np+1}\}$, there are 2^{np} cells labelled by $np+3$ which contains an object $\delta_{np+1}{}'$, 2^{np} cells labelled by $np+2$ that contains objects $\alpha_1, \ldots, \alpha_p, \alpha_{p+1}$ and a set of objects $R_{1,1}, R_{1,2}, \ldots, R_{1,p}, R_{2,1}, \ldots, R_{2,p}, \ldots, R_{n,1} \ldots, R_{n,p}$ $R \in \{T, F\}$, representing the truth assignment associated with that cell. Not all objects of such a set have to be present in a single cell since they could have been consumed by the application of a rule $[T_{i,j}F_{i,j'}]_{np+20} \longrightarrow {}_{np+2}[\#]_0$. Moreover, there exist 2^{np} cells labelled by $i + n \cdot (j-1)$, $1 \le i \le n, 1 \le j \le p$ each of them containing an object $l_{i,j,np+1}$. Then, by the application of the rules from 2.1, objects c_j will be created in the cells labelled by $np+2$ such that the truth assignment associated with such cell satisfies clause C_j due to any literal of it. Besides, rules $[\delta_{np+1}]_{np+1}$ $[\gamma]_0 \longrightarrow [\delta_{np+2}]_{np+10}$, $[\delta_{np+1}{}']_{np+3}[\gamma]_0 \longrightarrow [\delta_{np+2}{}']_{np+30}$ are applied.

Therefore, we have $\mathscr{C}_{np+2}(np+1) = \{\delta_{np+2}\}$, there are 2^{np} cells labelled by $np+3$ which contains an object $\delta_{np+2}{}'$, 2^{np} cells labelled by $np+2$ that contains objects $\alpha_1, \ldots, \alpha_p, \alpha_{p+1}$ and a multiset of objects from $\{c_1, \ldots, c_p\}$, representing the clauses satisfied by the truth assignment associated with that cell. The remaining objects $l_{i,j,np+1}$ in cells labelled by $i + n \cdot (j-1)$ are not going to be taken into account from now since they cannot "react" with something from this point of the computation.

4.1.3. Second Checking Stage. The step $np+3$ is devoted to checking whether all the clauses C_j, $1 \le j \le p$ of the Boolean formula φ have been satisfied by the truth assignment associated with each cell labelled by $np+2$. We have that $\mathscr{C}_{np+2}(np+1) = \{\delta_{np+2}\}$, and there will be 2^{np} cells labelled by $np+3$ which contains an object $\delta_{np+2}{}'$ and 2^{np} cells labelled by $np+2$ such that contains objects $\alpha_1, \ldots, \alpha_p, \alpha_{p+1}$ and objects c_j, $1 \le j \le p$ corresponding to the clauses satisfied by the truth assignment associated with that cell. Then, by applying the rules $[\alpha_{p+1}]_{np+2}[\delta_{np+2}{}']_{np+3} \longrightarrow [\alpha_{p+1}{}']_{np+2np+3}$, $[c_j\alpha_j]_{np+2np+3} \longrightarrow {}_{np+2}[\#]_{np+3}$, $[\delta_{np+2}]_{np+1}[\gamma]_0$ $\longrightarrow [\delta_{np+3}]_{np+10}$, we obtain the following: $\mathscr{C}_{np+3}(np+1) = \{\delta_{np+3}\}$ and there will be 2^{np} cells labelled by $np+2$ that will contain an object α_j if and only if clause C_j has not been satisfied by the truth assignment associated with such cell.

4.1.4. Output Stage. The output phase starts at the $(np+4)$-th step and takes exactly four steps, regardless of whether the input formula φ is satisfied or not by some truth assignment.

(i) Affirmative answer: if the input formula φ of SAT problem is satisfiable then at least one of the truth assignments from a cell with label $np+2$ has satisfied all clauses. Therefore, there will be at least one cell labelled by $np+2$ such that it will not have any object α_j, since all of them have been "consumed" in some sense by an object c_j. Therefore, in the step $np+4$, object δ_{np+3} will evolve into δ_{np+4} by the application of the rule $[\delta_{np+3}]_{np+1}[\gamma]_0$ $\longrightarrow [\delta_{np+4}]_{np+10}$, besides the application of rules $[\alpha_j\alpha_{p+1}{}']_{np+20} \longrightarrow {}_{np+2}[n_1]_0$ in other cells. In the next step, since $\alpha_{p+1}{}'$ remains in a cell labelled by $np+2$, it can evolve with object δ_{np+4} by the rule $[\alpha_{p+1}{}']_{np+2}[\delta_{np+4}]_{np+1} \longrightarrow [y_1]_{np+2np+1}$. Since there is only one object δ_{np+4}, then we can be sure that the object y_1 (that will produce the object `yes` later) is unique. Besides, objects n_1 will be sent as n_2 to cells labelled by $np+2$ in a nondeterministic way. In the $(np+5)$-th step, it will evolve into the object y_2. Let us keep in mind that objects n_2 cannot react with object δ_{np+4} since it has been consumed in the previous step. At the last step of the computation, the only rule applicable will be $[y_2]_{np+20} \longrightarrow {}_{np+2}[\mathtt{yes}]_0$. Then, an object `yes` will be sent to the environment, and the computation halts.

(ii) Negative answer: if the input formula φ of SAT problem is not satisfiactory, then none of the truth assignments encoded by a cell labelled as $np+2$ makes the formula φ true. Thus, in each cell, there will be at least one object of the type α_j, $1 \le j \le p$. It means that the clause C_j is not satisfied by the truth assignment corresponding to that cell. Therefore, at step $np+3$ rules $[\alpha_j\alpha_{p+1}{}']_{np+20} \longrightarrow {}_{np+2}[n_1]_0$ will be applied in all the cells. Therefore, at configuration $\mathscr{C}_{np+3}$ there will not be any object $\alpha_{p+1}{}'$ in the system, so object δ_{np+4} remains in the cell labelled by $np+1$. In the next step, objects n_1 go to cells labelled by $np+2$, selected in a nondeterministic way. Since object δ_{np+4} is in the cell labelled as $np+1$ it can make object n_2 evolve into object n_3, and then in the last step, it is sent to the environment as an object `no`. Since there is no applicable rule, the system halts.

Theorem 2. $\mathtt{SAT} \in \mathbf{PMC}_{\mathcal{TDEC}(2,1)}$

Proof. The family of P systems previously constructed verifies the following:

(i) Every system of the family Π is a recognizer P system from $\mathbf{TDEC}(2,1)$.

(ii) The family Π is polynomially uniform by Turing machines because, for each $n, p \in \mathbb{N}$, the rules of $\Pi(\langle n, p\rangle)$ of the family are recursively defined from $n, p \in \mathbb{N}$, and the amount of resources needed to build an element of the family is of a polynomial order in n and p, as shown below:

(1)Size of the alphabet: $3n^2p^2 + 11np + p + 17 \in \Theta(n^2p^2)$

(2) Initial number of cells: $np + 3 \in \Theta(np)$

(3) Initial number of objects in cells: $np + p + 3 \in \Theta(np)$
(4) Number of rules: $3n^2p^2 + np^2 + 12np + 2p + 13 \in \Theta(n^2p^2)$
(5) Maximal number of objects involved in any rule: $3 \in \Theta(1)$

(iii) The pair (cod, s) of polynomial-time computable functions defined fulfills the following: for each input formula φ of SAT problem, $s(\varphi)$ is a natural number, $\mathrm{cod}(\varphi)$ is an input multiset for the system $\Pi(s(\varphi))$, and for each $n \in \mathbb{N}$, $s^{-1}(n)$ is a finite set.

(iv) The family Π is polynomially bounded: indeed, for each input formula φ of SAT problem, the deterministic P system $\Pi(s(\varphi)) + \mathrm{cod}(\varphi)$ takes exactly $np + 7$ steps, being n the number of variables in φ and p its number of clauses.

(v) The family Π is sound with regard to (X, cod, s): for each formula φ, if the computation of $\Pi(s(\varphi)) + \mathrm{cod}(\varphi)$ is an accepting computation, then φ is satisfiable.

(vi) The family Π is complete with regard to (X, cod, s): for each input formula φ such that it is satisfiable, the computation of $\Pi(s(\varphi)) + \mathrm{cod}(\varphi)$ is an accepting computation. □

Corollary 1. $\mathbf{NP} \cup \mathbf{co} - \mathbf{NP} \subseteq \mathbf{PMC}_{\mathcal{TDEC}(2,1)}$

Proof. It is simple to prove this since it is known that SAT is a **NP**-complete problem, $\mathtt{SAT} \in \mathbf{PMC}_{\mathcal{TDEC}(2,1)}$ and the complexity class $\mathbf{PMC}_{\mathcal{TDEC}(2,1)}$ is closed under polynomial-time reduction and under complementary. □ □

Corollary 2. $\mathbf{NP} \cup \mathbf{co} - \mathbf{NP} \subseteq \mathbf{PMC}_{\mathcal{TDEC}(3)}$

5. The Presumed Efficiency of $\mathcal{TSEC}(2,2)$

In this section, the presumed efficiency of $\mathcal{TSEC}(2,2)$ is established by providing a polynomial time uniform solution to the SAT problem by means of a family of membrane systems $\Pi = \{\Pi(t) | t \in \mathbb{N}\}$ from $\mathcal{TSEC}(2,2)$.

For each pair of natural numbers $n, p \in \mathbb{N}$, we consider the recognizer tissue P system from $\mathcal{TSEC}(2,2)$, $\Pi(\langle n, p\rangle) = (\Gamma, \Gamma_0, \Gamma_1, \mathcal{E}, \Sigma, \mathcal{M}_1, \mathcal{M}_2, \mathcal{R}, i_{\mathrm{in}}, i_{\mathrm{out}})$, defined as follows:

(i) The working alphabet:

$$
\begin{aligned}
\Gamma = {} & \{\mathtt{yes}, \mathtt{no}, y_1, y_2, n_1, n_2, \#\} \cup \{a_{i,j} | 1 \le i \le n, 0 \le j \le i\} \cup \\
& \{a_{i,j}' | 2 \le i \le n, 0 \le j \le i-1\} \cup \\
& \{a_{i,j}^L, a_{i,j}^R | 2 \le i \le n, 0 \le j \le i-1\} \cup \\
& \{\alpha_j, \alpha_j', \alpha_j^L, \alpha_j^R | 1 \le j \le p+1\} \cup \\
& \{t_i, f_i, t_i', t_i'', f_i'', t_i^L, t_i^R, f_i^L, f_i^R | 1 \le i \le n\} \cup \\
& \{\beta_{l,k}, \beta_{l,k}', \beta_{l,k}^L, \beta_{l,k}^R 0 | \le k \le n, 1 \le l \le n\} \cup \\
& \{x_{i,j,k}, \overline{x}_{i,j,k}, x_{i,j,k}^* | 1 \le i \le n, 1 \le j \le p, 0 \le k \le n+j-1\} \cup \\
& \{x_{i,j,k}', \overline{x}_{i,j,k}', x_{i,j,k}^{*'}, x_{i,j,k}'', \overline{x}_{i,j,k}'', x_{i,j,k}^{*''}, x_{i,j,k}'', \overline{x}_{i,j,k}'', x_{i,j,k}^{*''} | 1 \le i \le n, 1 \le j \le p, 1 \le k \le n\} \cup \\
& \{c_{j,k} | 1 \le j \le p, j \le k \le p\} \cup \\
& \{\delta_i | 0 \le i \le 4n + p + 2\} \cup \{\delta_i' | 0 \le i \le 4n + p\}.
\end{aligned}
\tag{14}
$$

(ii) The partition $\{\{\Gamma_0, \Gamma_1\}$ defined as: $\Gamma_1 = \Gamma \setminus \Gamma_0$ and

$$
\begin{aligned}
\Gamma_0 = {} & \{a_{i,j}^L | 2 \le i \le n, 1 \le j \le i-1\} \cup \{\alpha_j^L | 1 \le j \le p+1\} \cup \\
& \{t_i^L, f_i^L | 1 \le i \le n\} \cup \{\beta_{l,k}^L | 0 \le k \le n, k+1 \le k \le n\}.
\end{aligned}
\tag{15}
$$

(iii) The input alphabet $\Sigma = \{x_{i,j,0}, \overline{x}_{i,j,0}, x_{i,j,0}^* | 1 \le i \le n, 1 \le j \le p\}$.

(iv) The environment alphabet $\mathcal{E} = \{\gamma\}$.

(v) $\mathcal{M}_1 = \{\delta_0, \delta_0'\} \cup \{\beta_{l,0}^{n+p+1} | 1 \le l \le n\}$;
$\mathcal{M}_2 = \{a_{i,0} | 1 \le i \le n\} \cup \{\alpha_j | 1 \le j \le p+1\}$.

(vi) The set $\mathcal{R}$ consists of the following rules:

(1) Rules for steps $4k + 1$.

$$
\begin{aligned}
&\left[a_{i,i-1}\right]_2[\gamma]_0 \longrightarrow \left[a_{i,i-1}{}'t_i'\right]_2[\,]_0 \quad \text{for } 1 \le i \le n,\\
&\left.\begin{aligned}
&[t_i]_2[\gamma]_0 \longrightarrow \left[t_i''\right]_2[\,]_0\\
&[f_i]_2[\gamma]_0 \longrightarrow \left[f_i''\right]_2[\,]_0
\end{aligned}\right\} \quad \text{for } 1 \le i \le n,\\
&\left[a_{i,j}\right]_2[\gamma]_0 \longrightarrow \left[a_{i,j}'\right]_2[\,]_0 \quad \text{for } 2 \le i \le n, 0 \le j \le i-2,\\
&\left[\alpha_j\right]_2[\gamma]_0 \longrightarrow \left[\alpha_j'\right]_2[\,]_0 \quad \text{for } 1 \le j \le p+1,\\
&\left[\beta_{l,k}\right]_1[\gamma]_0 \longrightarrow \left[\beta_{l,k}'\right]_1[\,]_0 \quad \text{for } 0 \le k \le n, k+1 \le j \le n,\\
&\left.\begin{aligned}
&\left[x_{i,j,k}\right]_1[\gamma]_0 \longrightarrow \left[x_{i,j,k}'\right]_1[\,]_0\\
&\left[\overline{x}_{i,j,k}\right]_1[\gamma]_0 \longrightarrow \left[\overline{x}_{i,j,k}'\right]_1[\,]_0\\
&\left[x_{i,j,k}^*\right]_1[\gamma]_0 \longrightarrow \left[x_{i,j,k}^{*\prime}\right]_1[\,]_0
\end{aligned}\right\} \quad \begin{aligned}&1 \le i \le n,\\ \text{for } &1 \le j \le p,\\ &0 \le k \le n-1.\end{aligned}
\end{aligned}
\tag{16}
$$

(2) Rules for steps $4k + 2$.

$$
\begin{aligned}
&\left.\begin{aligned}
&\left[a_{i,i-1}{}'\right]_2[\gamma]_0 \longrightarrow \left[a_{i,i-1}{}'f_i^R\right]_2[\,]_0\\
&[t_i']_2[\gamma]_0 \longrightarrow \left[t_i^L\right]_2[\,]_0\\
&[t_i'']_2[\gamma]_0 \longrightarrow \left[t_i^L t_i^R\right]_2[\,]_0\\
&[f_i'']_2[\gamma]_0 \longrightarrow \left[f_i^L f_i^R\right]_2[\,]_0
\end{aligned}\right\} \quad \text{for } 1 \le i \le n,\\
&\left[a_{i,j}{}'\right]_2[\gamma]_0 \longrightarrow \left[a_{i,j+1}^L a_{i,j+1}^R\right]_2[\,]_0 \quad \text{for } 2 \le i \le n, 0 \le j \le i-2,\\
&\left[\alpha_j'\right]_2[\gamma]_0 \longrightarrow \left[\alpha_j^L \alpha_j^R\right]_2[\,]_0 \quad \text{for } 1 \le j \le p+1,\\
&\left[\beta_{l,k}{}'\right]_1[\gamma]_0 \longrightarrow \left[\beta_{l,k}^L \beta_{l,k}^R\right]_1[\,]_0 \quad \text{for } 0 \le k \le n, k+1 \le j \le n,\\
&\left.\begin{aligned}
&\left[x_{i,j,k}{}'\right]_1[\gamma]_0 \longrightarrow \left[x_{i,j,k}^{\prime\prime 2}\right]_1[\,]_0\\
&\left[\overline{x}_{i,j,k}{}'\right]_1[\gamma]_0 \longrightarrow \left[\overline{x}_{i,j,k}^{\prime\prime 2}\right]_1[\,]_0\\
&\left[x_{i,j,k}^{*\prime}\right]_1[\gamma]_0 \longrightarrow \left[x_{i,j,k}^{*\prime\prime 2}\right]_1[\,]_0
\end{aligned}\right\} \quad \begin{aligned}&1 \le i \le n,\\ \text{for } &1 \le j \le p,\\ &0 \le k \le n-1.\end{aligned}
\end{aligned}
\tag{17}
$$

(3) Rules for steps $4k + 3$.

$$
\begin{aligned}
&\left[a_{i,i}\right]_2 \longrightarrow [\Gamma_0]_2[\Gamma_1]_2 \text{ for } 1 \le i \le n,\\
&\left.\begin{aligned}
&\left[\beta_{k,k}^O\right]_1[\,]_0 \longrightarrow [\,]_1\left[\beta_{k,k}^O\right]_0\\
&\left[\beta_{l,k}^O\right]_1[\,]_0 \longrightarrow [\,]_1\left[\beta_{l,k}^O\right]_0
\end{aligned}\right\} \quad \text{for } \begin{aligned}&O \in \{L, R\},\\ &1 \le k \le n, k+1 \le l \le n,\end{aligned}\\
&\left.\begin{aligned}
&\left[x_{i,j,k}{}''\right]_1[\gamma]_0 \longrightarrow \left[x_{i,j,k}''\right]_1[\,]_0\\
&\left[\overline{x}_{i,j,k}{}''\right]_1[\gamma]_0 \longrightarrow \left[\overline{x}_{i,j,k}''\right]_1[\,]_0\\
&\left[x_{*i,j,k}''\right]_1[\gamma]_0 \longrightarrow \left[x_{i,j,k}^{*\prime\prime}\right]_1[\,]_0
\end{aligned}\right\} \quad \begin{aligned}&1 \le i \le n,\\ \text{for } &1 \le j \le p,\\ &1 \le k \le n.\end{aligned}
\end{aligned}
\tag{18}
$$

(4) Rules for steps $4k$.

$$\left.\begin{array}{l}\left[a_{i,j}^{O}\right]_2\left[\beta_{k,k}^{O}\right]_0 \longrightarrow \left[a_{i,j}\right]_2[\,]_0\\ \left[r_i^{O}\right]_2\left[\beta_{k,k}^{O}\right]_0 \longrightarrow \left[r_i\right]_2[\,]_0\end{array}\right\} \quad \begin{array}{l} O \in \{L,R\}, r \in \{t,f\},\\ \text{for } 1 \le i \le n, 1 \le j \le n,\\ 1 \le k \le n,\end{array}$$

$$\left.\begin{array}{l}\left[x_{i,j,k}''\right]_1[\gamma]_0 \longrightarrow \left[x_{i,j,k}\right]_1[\,]_0\\ \left[\overline{x}_{i,j,k}''\right]_1[\gamma]_0 \longrightarrow \left[\overline{x}_{i,j,k}\right]_1[\,]_0\\ \left[x_{i,j,k}^{*\prime\prime}\right]_1[\gamma]_0 \longrightarrow \left[x_{i,j,k}^{*}\right]_1[\,]_0\end{array}\right\} \quad \begin{array}{l} 1 \le i \le n,\\ \text{for } 1 \le j \le p,\\ 0 \le k \le n,\end{array} \tag{19}$$

$$\left[\beta_{l,k}\right]_0[\,]_1 \longrightarrow [\,]_0\left[\beta_{l,k}\right]_1 \quad \text{for } 0 \le k \le n, k+1 \le l \le n.$$

(5) Rules to check the satisfied clauses.

$$\left.\begin{array}{l}[t_i]_2\left[x_{i,j,n+j-1}\right]_1 \longrightarrow \left[c_{j,j}t_i\right]_2[\,]_1\\ [t_i]_2\left[\overline{x}_{i,j,n+j-1}\right]_1 \longrightarrow [t_i]_2[\,]_1\\ [t_i]_2\left[x_{i,j,n+j-1}^{*}\right]_1 \longrightarrow [t_i]_2[\,]_1\\ [f_i]_2\left[x_{i,j,n+j-1}\right]_1 \longrightarrow [f_i]_2[\,]_1\\ [f_i]_2\left[x_{i,j,n+j-1}\right]_1 \longrightarrow \left[c_{j,j}f_i\right]_2[\,]_1\\ [f_i]_2\left[x_{i,j,n+j-1}^{*}\right]_1 \longrightarrow [f_i]_2[\,]_1\end{array}\right\} \quad \text{for } \begin{array}{l}1 \le i \le n,\\ 1 \le j \le p,\end{array}$$

$$\left.\begin{array}{l}\left[x_{i,j,n+k}\right]_1[\gamma]_0 \longrightarrow \left[x_{i,j,n+k+1}\right]_1[\,]_0\\ \left[\overline{x}_{i,j,n+k}\right]_1[\gamma]_0 \longrightarrow \left[\overline{x}_{i,j,n+k+1}\right]_1[\,]_0\\ \left[x_{i,j,n+k}^{*}\right]_1[\gamma]_0 \longrightarrow \left[x_{i,j,n+k+1}^{*}\right]_1[\,]_0\end{array}\right\} \quad \begin{array}{l} 1 \le i \le n,\\ \text{for } 1 \le j \le p,\\ 0 \le k \le j-2.\end{array} \tag{20}$$

(6) Rules to check if all the clauses are satisfied by a truth assignment.

$$\begin{array}{l}\left[\alpha_{p+1}\right]_2\left[\delta_{4n+p}'\right]_1 \longrightarrow \left[\alpha_{p+1}'\right]_2[\,]_1,\\ \left[\alpha_j c_{j,p}\right]_2[\,]_1 \longrightarrow [\,]_2[\#]_1 \quad \text{for } 1 \le j \le p.\end{array} \tag{21}$$

(7) General counters.

$$\begin{array}{l}[\delta_i]_1[\gamma]_0 \longrightarrow [\delta_{i+1}]_2[\,]_1 \quad \text{for } 0 \le i \le 4n+p+1,\\ [\delta_{4i+1}']_1[\gamma]_0 \longrightarrow \left[\delta_{4i+1}'^{2}\right]_2[\,]_1 \quad \text{for } 0 \le i \le n-1,\\ [\delta_{4i+k}']_1[\gamma]_0 \longrightarrow [\delta_{4i+k+1}']_2[\,]_1 \quad \text{for } 0 \le i \le n-1, k \in \{0,2,3\},\\ [\delta_{4n+i}']_1[\gamma]_0 \longrightarrow [\delta_{4n+i+1}']_2[\,]_1 \quad \text{for } 0 \le i \le p-1.\end{array} \tag{22}$$

(8) Rules to return a negative answer.

$$\begin{array}{l}\left[\alpha_j\alpha_{p+1}'\right]_2[\,]_0 \longrightarrow [\,]_1[\#]_0 \quad \text{for } 1 \le j \le p,\\ [n_1]_0[\,]_2 \longrightarrow [\,]_0[n_1]_2,\\ [n_1]_2\left[\delta_{4n+p+2}\right]_1 \longrightarrow [n_2]_2[\,]_1,\\ [n_2]_2[\,]_0 \longrightarrow [\,]_2[\mathtt{no}]_2.\end{array} \tag{23}$$

(9) Rules to return a positive answer.

$$\begin{array}{l}\left[\alpha_{p+1}'\right]_2\left[\delta_{4n+p+2}\right]_1 \longrightarrow [y_1]_2[\,]_1,\\ [y_1]_2[\gamma]_0 \longrightarrow [y_2]_2[\,]_0,\\ [y_2]_2[\,]_0 \longrightarrow [\,]_2[\mathtt{yes}]_2.\end{array} \tag{24}$$

(ii) The input cell is the cell labelled by 1 (that is, $i_{\text{in}} = 1$) and the output zone is the environment (that is, $i_{\text{out}} = \text{env}$).

5.1. An Overview of the Computations. A brief overview of the computation is explained below. For a more exhaustive verification, we refer the reader to [21].

5.1.1. Generation Stage. The objective of this stage is twofold: on the one hand, all truth assignments for the variables associated with the Boolean formula $\varphi(x_1, \ldots, x_n)$ are generated by applying rules from 2 in cells labelled by 2, in such a manner that in the $4i+2$-th step ($1 \le i \le n-1$) of this stage, separation rule associated with an object $a_{i,i}$ is triggered, two new cells distributing t_i and f_i between them. In the last step of this stage, each cell labelled by 2 will contain a truth assignment of the formula. On the other hand, 2^n copies of each object $a \in \text{cod}(\varphi)$ are going to be generated in the cell labelled by 1. These objects are being prepared for the next stage. It is interesting to keep in mind that separation rules are executed each 4 time steps, so objects keep evolving within their corresponding cells in order to maintain the

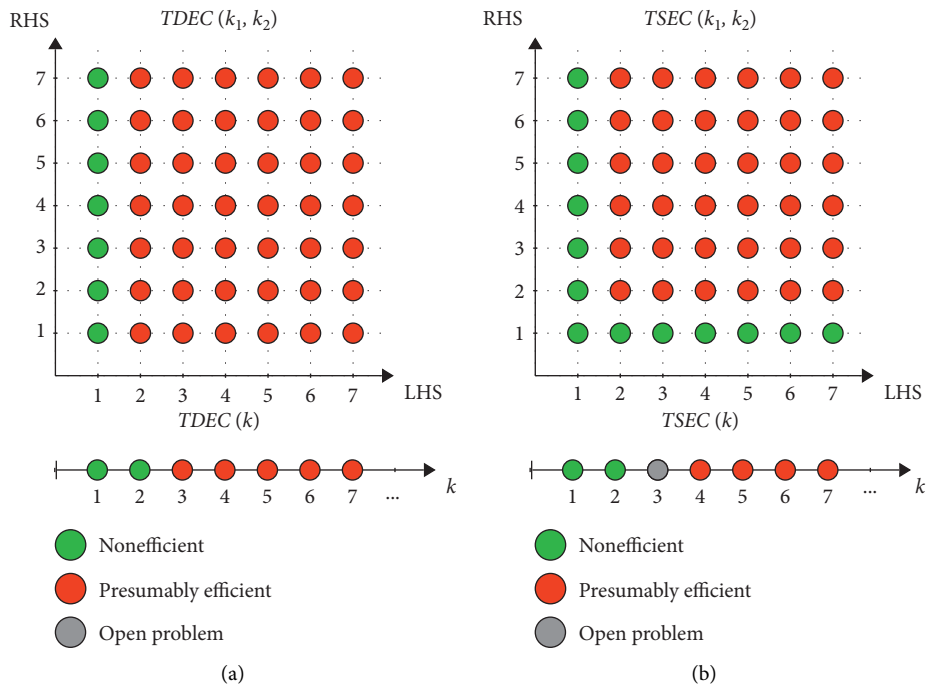

Figure 1: Known complexity results on tissue P systems with evolutional symport/antiport rules.

synchronization. It is done by means of a simulation of evolution rules, by using object γ as a "catalyzer" of the reaction:

$$[a]_1[\gamma]_0 \longrightarrow [a']_{10}. \tag{25}$$

5.1.2. First Checking Stage. In this stage, objects $c_{j,k}$ are generated in the cells labelled by 2 such that the truth assignment associated with them makes the clause C_j true. In each step, the system processes a single clause so that p steps are needed to process the whole formula. Objects $c_{j,k}$ are generated in any of the following cases:

(i) Literal x_i appears in clause C_j, and the value of variable x_i in a truth assignment is True. Then, we can say that such truth assignment satisfies this clause.

(ii)Literal x_i appears in clause C_j, and the value of variable x_i in a truth assignment is False. Then, we can say that such truth assignment satisfies this clause.

In any other case, variable x_i has nothing to do with clause C_j. At the final step of this stage, cells labelled by 2 will have objects $c_{j,p}$ where C_j are clauses satisfied by such truth assignment. We obtain an object α'_{p+1} to use it in the next stage.

5.1.3. Second Checking Stage. Here, rules from 6 are fired at the $(4n + p + 1)$-th step, and objects α_j within a cell labelled by 2 are removed if and only if the truth assignment associated with that cell makes a true clause C_j, that is, if there is at least one object c_j in such cell. In this step, objects α_j from cells labelled by 2 such that their corresponding element $c_{j,p}$ appears in it disappear. At the same time, an object α'_{p+1} appears in each cell labelled by 2 for the next stage, where they will interact with the remaining, if any, objects a_j.

5.1.4. Output Stage. This stage takes four steps, regardless of the formula being satisfactory or not. If there is at least one truth assignment that makes the input formula true, then at least one cell labelled by 2 will preserve an object α'_{p+1}, and it will interact with the object δ_{4n+p+2} to finally return an object `yes` to the environment. On the contrary, if none of the truth assignments makes the formula φ true, then object δ_{4n+p+2} will remain and it will interact with the object n_1, leading to the sending of an object `no` into the environment.

Theorem 3. $\mathtt{SAT} \in \mathbf{PMC}_{\mathcal{TSEC}(2,2)}$

Proof. The family of P systems previously constructed verifies the following:

(i) Every system of the family Π is a recognizer P system from $\mathbf{TSEC}(2, 2)$.

(ii) The family Π is polynomially uniform by Turing machines because, for each $n, p \in \mathbb{N}$, the rules of $\Pi(\langle n, p\rangle)$ of the family are recursively defined from $n, p \in \mathbb{N}$, and the amount of resources needed to build an element of the family is of a polynomial order in n and p, as follows:

(1) Size of the alphabet: $9n^2p + 6n^2 + 3np^2/2 - 3np + 22n + p^2/2 + 13p/2 + 14 \in \Theta(\max\{n^2p, np^2\})$

(2) Initial number of cells: $2 \in \Theta(1)$

(3) Initial number of objects in cells: $n^2 + n(p+1) + p + 3 \in \Theta(n^2)$
(4) Number of rules: $8n^3 + 27n^2 p/\quad 2 + 4n^1 + 19np/2 + 23n + p^2/2 + 17p/2 + 11 \in \Theta(n^3)$
(5) Maximal number of objects involved in any rule: $4 \in \Theta(1)$

(iii) The pair (cod, s) of polynomial-time computable functions defined fulfill the following: for each input formula φ of SAT problem, $s(\varphi)$ is a natural number, $\mathrm{cod}(\varphi)$ is an input multiset of the system $\Pi(s(\varphi))$, and for each $n \in \mathbb{N}$, $s^{-1}(n)$ is a finite set.

(iv) The family Π is polynomially bounded: indeed for each input formula φ of SAT problem, the deterministic P system $\Pi(s(\varphi)) + \mathrm{cod}(\varphi)$ takes exactly $4n + p + 5$ steps, being n the number of variables of φ and p the number of clauses.

(v) The family Π is sound with regard to (X, cod, s): for each formula φ, if the computation of $\Pi(s(\varphi)) + \mathrm{cod}(\varphi)$ is an accepting computation, then φ is satisfiable.

(vi) The family Π is complete with regard to (X, cod, s): for each input formula φ such that it is satisfiable, the computation of $\Pi(s(\varphi)) + \mathrm{cod}(\varphi)$ is an accepting computation. □

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

Proof. It is simple to prove this since it is known that SAT is a **NP**-complete problem, $\mathtt{SAT} \in \mathbf{PMC}_{\mathcal{TSEC}(2,2)}$ and the complexity class $\mathbf{PMC}_{\mathcal{TSEC}(2,2)}$ is closed under polynomial-time reduction and under complementary. □

Bearing in mind that $\mathcal{TSEC}(2,2) \subseteq \mathcal{TSEC}(3,2)$, the result of the previous corollary improves the one obtained in [19] concerning to the presumed efficiency of the computing model $\mathcal{TSEC}(3,2)$. □

6. Conclusions and Future Work

In this paper, following the works initiated in [18, 19], tissue P systems with evolutional symport/antiport rules have been studied from a computational complexity view, where either division rules or separation rules are used as mechanisms to create an exponential workspace in polynomial time. In this context, new results have been achieved:

(i) With regard to tissue P systems with division rules:

(1) The nonefficiency of membrane systems from $\mathcal{TDEC}(1,n)$, for each natural number $n \geq 1$, has been established, complementing the result previously obtained in [18] concerning the nonefficiency of membrane systems from $\mathcal{TDEC}(2)$.
(2) The presumed efficiency of $\mathcal{TDEC}(2,1)$ has been established, improving the result previously obtained in [18] concerning the presumed efficiency of $\mathcal{TDEC}(4)$, since $\mathcal{TDEC}(2,1) \subseteq \mathcal{TDEC}(3) \subseteq \mathcal{TDEC}(4)$.

(ii) With regard to tissue P systems with separation rules:

(1) The presumed efficiency of $\mathcal{TSEC}(2,2)$ has been established, improving the result previously obtained in [19] concerning the presumed efficiency of $\mathcal{TSEC}(3,2)$.

All complexity results obtained until now in the framework of tissue P systems with evolutional symport/antiport rules are described in Figure 1:

The following diagram describes the known frontiers between the nonefficiency and the presumed efficiency in the framework of tissue P systems with evolutional symport/antiport rules [22–25]:

$$\begin{aligned}
\mathcal{TDEC}(\mathbf{2}) &\rightsquigarrow \mathcal{TDEC}(\mathbf{3}),\\
\mathcal{TSEC}(\mathbf{2}) &\rightsquigarrow \mathcal{TSEC}(\mathbf{4}),\\
\mathcal{TDEC}(\mathbf{1}, n) &\rightsquigarrow \mathcal{TDEC}(\mathbf{2}, n),\\
\mathcal{TSEC}(\mathbf{1}, 2) &\rightsquigarrow \mathcal{TSEC}(\mathbf{2}, 2),\\
\mathcal{TSEC}(2, \mathbf{1}) &\rightsquigarrow \mathcal{TSEC}(2, \mathbf{2}),\\
\mathcal{TSEC}(2, 1) &\rightsquigarrow \mathcal{TDEC}(2, 1).
\end{aligned} \tag{26}$$

We propose the following open problems for future researches:

(i) What is the upper bound of complexity classes $\mathbf{PMC}_{\mathcal{R}}$, where $\mathcal{R}$ is a presumably efficient class of tissue P systems with division or separation rules and evolutional communication rules?

(ii) Is the class of membrane systems from $\mathcal{TSEC}(3)$ nonefficient or presumably efficient?

(iii)What is the role of the environment in the framework of tissue P systems with division or separation rules and evolutional communication rules?

(iv)What is the behaviour of recognizer cell-like P systems with division or separation rules and evolutional communication rules from a computational complexity perspective?

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (61772214, 61602192, and 61320106005), the Fundamental Research Funds for the Central Universities (531118010355), Hunan Provincial Natural Science Foundation of China (2020JJ4215), and Key Research and Development Program of Changsha (kq2004016). Authors from the University of Seville also acknowledge the support of the research project TIN2017-

89842-P, cofinanced by Ministerio de Economía, Industria y Competitividad (MINECO) of Spain, through the Agencia Estatal de Investigación (AEI), and by Fondo Europeo de Desarrollo Regional (FEDER) of the European Union.

References

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

[2] C. Martín-Vide, J. Pazos, G. Păun, and A. Rodríguez-Patón, "A new class of symbolic abstract neural nets: tissue P systems," *Lecture Notes in Computer Science*, vol. 2387, pp. 290–299, 2002.

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

[4] Gh. Păun, *Membrane Computing. An Introduction*, Springer-Verlag, Berlin, Germany, 2002.

[5] Gh. Păun, *The Oxford Handbook of Membrane Computing*, Oxford University Press, Oxford, UK, 2010.

[6] P. Frisco, M. Gheorghe, and M. J. Pérez-Jiménez, "Applications of Membrane Computing in Systems and Synthetic Biology," *Emergence, Complexity and Computation*, Springer, Cham, Switzerland, vol. 7, 2014.

[7] L. Pan, Gh. Păun, M. J. Pérez-Jiménez, and T. Song, "Bio-inspired computing: theories and applications," *Communications in Computer and Information Science (Series ISSN 1865-0929)*, Springer-Verlag Berlin Heidelberg, New York, NY, USA, 2014.

[8] G. Zhang, M. J. Pérez-Jiménez, and M. Gheorghe, "Real-life applications with membrane computing," *Emergence, Complexity and Computation*, Springer, Cham, Switzerland, vol. 25, 2017.

[9] Ch.H. Papadimitriou, *Computational Complexity*, Addison-Wesley Publishing Company, New York, NY, USA, 1994.

[10] M. J. Pérez-Jiménez, A. Romero, and F. Sancho, "Decision P systems and the $\mathbf{P} \neq \mathbf{NP}$ conjecture," *Lecture Notes In Computer Science*, vol. 2597, pp. 388–399, 2003.

[11] M. J. P. Jiménez, Á. R. Jiménez, and F. S. Caparrini, "Complexity classes in models of cellular computing with membranes," *Natural Computing*, vol. 2, no. 3, pp. 265–285, 2003.

[12] C. Zandron, C. Ferretti, and G. Mauri, "Solving NP-complete problems using P systems with active membranes," in *Unconventional Models of Computation*, I. Antoniou, C. S. Calude, and M. J. Dinneen, Eds., pp. 289–301, Springer, Berlin, Germany, 2000.

[13] M. J. Pérez-Jiménez, "An approach to computational complexity in Membrane Computing," *Lecture Notes in Computer Science*, vol. 3365, pp. 85–109, 2005.

[14] R. Gutiérrez-Escudero, M. J. Pérez-Jiménez, and M. Rius-Font, "Characterizing tractability by tissue-like P systems. Membrane computing, 10th international workshop, WMC 2009, curtea de Arges, Romania, august 24-27, 2009, revised selected and invited papers," *Lecture Notes in Computer Science*, vol. 5957, pp. 289–300, 2010.

[15] L. Pan, M. J. Pérez-Jiménez, A. Riscos-Núñez, and M. Rius-Font, "New frontiers of the efficiency in tissue P systems," in *Pre-proceedings of Asian Conference on Membrane Computing (ACMC 2012)*, L. Pan, Gh. Paun, and T. Song, Eds., pp. 61–73, Huazhong University of Science and Technology, Wuhan, China, 2012.

[16] M. J. Pérez-Jiménez and P. Sosík, "Improving the efficiency of tissue P systems with cell separation," in *Proceedings of the 10th Brainstorming Week on Membrane Computing, Sevilla*, pp. 105–140, Spain, 2012.

[17] A. E. Porreca, N. Murphy, and M. J. Pérez-Jiménez, "An optimal frontier of the efficiency of tissue P systems with cell division," *Fundamenta Informaticae*, vol. 138, no. 1-2, pp. 45–60, 2012.

[18] B. Song, C. Zhang, and L. Pan, "Tissue-like P systems with evolutional symport/antiport rules," *Information Sciences*, vol. 378, no. 1–2, pp. 45–60, 2017.

[19] L. Pan, B. Song, L. Valencia-Cabrera, and M. J. Pérez-Jiménez, "The computational complexity of tissue P systems with evolutional symport/antiport rules," *Complexity*, vol. 2018, 2018.

[20] D. Orellana-Martín, L. Valencia-Cabrera, A. Riscos-Núñez, and M. J. Pérez-Jiménez, "P systems with proteins: a new frontier of efficiency when membrane division dissapears," *Journal of Membrane Computing*, vol. 1, no. 1, pp. 29–39, 2019.

[21] D. Orellana-Martín, L. Valencia-Cabrera, B. Song, L. Pan, and M. J. Pérez-Jiménez, "Narrowing frontiers with evolutional communication rules and cell separation," in *Proceedings of the 16th Brainstorming Week on Membrane Computing*, Sevilla, Spain, 2018, pp. 123–162.

[22] R. Fueter and G. Pólya, "Rationale abzählung der Gitterpunkte," *Vierteljschr Naturforsch*, vol. 58, pp. 380–386, 1923.

[23] 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.

[24] M. R. Garey, D. S. J.. Computers, and Intractability, *A Guide to the Theory of **NP**-Completeness*, W. H. Freeman, New York, NY, USA, 1990.

[25] P. R. Halmos, *Naive Set Theory. The University Series in Undergraduate Mathematics*, D. Van Nostrand Company, Inc., New York, NY, USA, 1960.