Tissue P Systems With Channel States Working in the Flat Maximally Parallel Way

Bosheng Song, Mario J. Pérez-Jiménez, Gheorghe Păun, and Linqiang Pan*, *Member, IEEE*

***Abstract*— Tissue P systems with channel states are a class of bio-inspired parallel computational models, where rules are used in a sequential manner (on each channel, at most one rule can be used at each step). In this work, tissue P systems with channel states working in a flat maximally parallel way are considered, where at each step, on each channel, a maximal set of applicable rules that pass from a given state to a unique next state, is chosen and each rule in the set is applied once. The computational power of such P systems is investigated. Specifically, it is proved that tissue P systems with channel states and antiport rules of length two are able to compute Parikh sets of finite languages, and such P systems with one cell and noncooperative symport rules can compute at least all Parikh sets of matrix languages. Some Turing universality results are also provided. Moreover, the NP-complete problem SAT is solved by tissue P systems with channel states, cell division and noncooperative symport rules working in the flat maximally parallel way; nevertheless, if channel states are not used, then such P systems working in the flat maximally parallel way can solve only tractable problems. These results show that channel states provide a frontier of tractability between efficiency and non-efficiency in the framework of tissue P systems with cell division (assuming P $\neq$ NP).**

***Index Terms*— Bio-inspired computing, channel state, flat maximal parallelism, membrane computing, tissue P system, tractability border.**

I. Introduction

BIO-INSPIRED COMPUTING is an active research field that investigates a way of developing computational models or algorithmic methodologies from biological systems, whose effectiveness and broad range applicability can be validated in many aspects, e.g., artificial neural networks, molecular computing and particle swarm optimization, etc. Membrane computing seeks to discover new computational paradigms from biological cells, which is inspired by the structure and the functioning of living cells, abstracting computational ideas (e.g., computational models, data structures, data operations) from the way in which chemicals interact and cross cellular membranes. The computation models investigated in membrane computing are called *P systems* (also called *membrane systems*), which are distributed and parallel computing devices. Since the seminal definition of P systems [1], a large number of theoretical models and results have been obtained [2]–[9], and they have been used to solve various computational problems [10]–[17] from a wide number of areas such as mathematics, biology, economics and theoretical computer science. According to the membrane structure, there are two main families of P systems: *cell-like P systems*, which have a hierarchical arrangement of membranes as in a cell [1]; and *tissue-like P systems* [6] or *neural-like P systems* [4] which have a net of membranes (placed in the nodes of a graph) as in a tissue or a neural net. A comprehensive introduction of membrane systems can be found in [18], and for the most up-to-date news and results the reader can refer to the P systems web page `http://ppage.psystems.eu`. The present work focuses on a class of tissue-like P systems.

A tissue-like P system can be described by a directed graph implicitly given by means of communication (symport/antiport) rules [7]. The nodes of that graph are called *cells* and there is a distinguished node called *environment*. Each arc can be considered as a communication channel between two *regions* (two cells or a cell and the environment), that is, two regions of the system can communicate by means of communication (symport/antiport) rules [7]. Symport rules move objects across a membrane together in one direction, whereas antiport rules move objects across a membrane in opposite directions. The length of a communication rule is the total number of objects involved in that rule. A communication rule of length 1 is said to be *noncooperative*, otherwise it is called *cooperative*. A tissue P system is said to be *noncooperative* if the system has only symport rule of length 1, otherwise, the tissue P system is *cooperative*.

Tissue P systems with symport/antiport rules were first investigated in [19]. This initial model was then modified by incorporating various additional features motivated by some biological facts. We refer to [20]–[27] for a survey of these investigations. An interesting variant of tissue P systems with symport/antiport rules, called *tissue P systems with channel states*, was proposed in [28], where between two cells or between a cell and the environment at most one channel is established, and a state is associated with each channel to

The work of B. Song and L. Pan was supported by National Natural Science Foundation of China (61320106005, 61033003, 61602192 and 91130034), Ph.D. Programs Foundation of Ministry of Educa-tion of China (20120142130008), the Innovation Scientists and Technicians Troop Construction Projects of Henan Province (154200510012). The work of M. J. Pérez-Jiménez was supported by "Ministerio de Economía y Competitividad" of Spanish government (TIN2012-37434), cofunded by FEDER funds. *Asterisk indicates corresponding author.*

B. Song is with the Key Laboratory of Image Information Processing and Intelligent Control, School of Automation, Huazhong University of Science and Technology, Wuhan, Hubei 430074, China.

M. J. Pérez-Jiménez is with the Research Group on Natural Computing, Department of Computer Science and Artificial Intelligence, University of Sevilla, 41012 Sevilla, Spain.

G. Păun is with the Institute of Mathematics of the Romanian Academy, 014700 Bucureşti, Romania.

*L. Pan is with the Key Laboratory of Image Information Processing and Intelligent Control, School of Automation, Huazhong University of Science and Technology, Wuhan, Hubei 430074, China (e-mail: lqpan@mail.hust.edu.cn).

control the communication at each step. The rules of tissue P systems with channel states designed in [28] are used in a sequential manner, that is, on each channel between two cells or between a cell and the environment, at most one rule can be used at one step.

In this work, an attractive strategy of using rules, called *flat maximal parallelism*, is introduced into tissue P systems with channel states, where rules are applied in a flat maximally parallel manner at the level of each channel in the sense that at each step, on each channel, a maximal set of applicable rules which pass from a given state to a unique next state is chosen and each rule in the set is applied once [29], [30], and in a maximally parallel manner at the level of the system in the sense that all channels which can use rules in a flat maximally parallel manner must do it.

The computational power of tissue P systems with channel states working in the flat maximally parallel way is investigated. Specifically, it is proved that such P systems with antiport rules of length two are able to compute Parikh sets of finite languages, and such P systems with one cell and noncooperative symport rules can compute at least all Parikh sets of matrix languages. We further prove that this kind of P systems with one cell, one state and only using symport rules of length three, or two cells, any number of states and only using noncooperative symport rules, or arbitrarily many cells, four states and only using noncooperative symport rules are universal.

The computational efficiency of tissue P systems with channel states and cell division working in the flat maximally parallel way is studied. It is proved that the **NP**-complete problem SAT can be solved by noncooperative tissue P systems with channel states and cell division working in the flat maximally parallel way; if we consider such P systems without channel states, then only tractable problems can be solved. These results show that channel states provide a frontier of tractability between efficiency and non-efficiency in the framework of tissue P systems with cell division (assuming $\mathbf{P} \neq \mathbf{NP}$).

II. Preliminaries

It is useful for the reader to have some familiarity with (basic elements of) language theory, e.g., from [31]. Here we only recall some notions used in this work.

An *alphabet* Γ is a finite and non-empty set and whose elements are called *symbols*. For an alphabet Γ we denote by Γ^* the set of all strings of symbols from Γ. A *multiset m* over an alphabet Γ is a pair (Γ, f), where f is a mapping from Γ onto the set of natural numbers $\mathbb{N}$. If $m = (\Gamma, f)$ is a multiset, then its *support* is defined as $supp(m) = \{x \in \Gamma \mid f(x) > 0\}$. A multiset is finite (resp., empty) if its support is a finite (resp., empty) set. We denote by $M_f(\Gamma)$ the set of all finite non-empty multisets over Γ. If $m = (\Gamma, f)$ is a finite multiset over Γ, and $supp(m) = \{a_1, \ldots, a_k\}$, then it will be denoted as $m = \{a_1^{f(a_1)}, \ldots, a_k^{f(a_k)}\}$ and its *cardinality* is defined as follows: $|m| = f(a_1) + \ldots + f(a_k)$. All permutations of a multiset precisely identify the same multiset. Let $m_1 = (\Gamma, f_1)$, $m_2 = (\Gamma, f_2)$ be multisets over Γ, then the union of m_1 and m_2, denoted by $m_1 + m_2$, is the multiset (Γ, g), where $g(x) = f_1(x) + f_2(x)$ for each $x \in \Gamma$. We say that m_1 is contained in m_2, and we denote it by $m_1 \subseteq m_2$, if $f_1(x) \leq f_2(x)$ for each $x \in \Gamma$. The relative complement of m_2 in m_1, denoted by $m_1 \setminus m_2$, is the multiset (Γ, g), where $g(x) = f_1(x) - f_2(x)$ if $f_1(x) \geq f_2(x)$, and $g(x) = 0$ otherwise.

The *Parikh vector* associated with a string $x \in \Gamma^*$ with respect to the alphabet $\Gamma = \{a_1, \ldots, a_n\}$ is $\Psi_\Gamma(x) = (|x|_{a_1}, |x|_{a_2}, \ldots, |x|_{a_n})$. For a language $L \subseteq \Gamma^*$ we define $\Psi_\Gamma(L) = \{\Psi_\Gamma(x) \mid x \in L\}$, this is called the *Parikh image* of L.

A set of languages is usually called a *family* of languages. For a family of languages FL, the family of Parikh images of languages in FL is denoted by $PsFL$. By $PsFIN$ we denote the family of Parikh images of finite languages. By $PsRE$ we denote the family of recursively enumerable sets of vectors of natural numbers; this is equal to the family of Parikh sets of recursively enumerable languages.

A *matrix grammar without appearance checking* is a construct $G = (N, T, S, M)$, where N, T are disjoint sets called *nonterminal alphabet* and *terminal alphabet*, respectively, $S \in N$ is the *axiom*, M is a finite set of sequences of the form $(A_1 \to x_1, \ldots, A_n \to x_n)$, $n \geq 1$, of context-free rules over $N \cup T$ (with $A_i \in N, x_i \in (N \cup T)^*$, in all cases), where the elements of M are called *matrices*.

For $w, z \in (N \cup T)^*$, we write $w \Rightarrow z$ if there is a matrix $(A_1 \to x_1, \ldots, A_n \to x_n)$ in M and the strings $w_i \in (N \cup T)^*$, $1 \leq i \leq n+1$, such that $w = w_1, z = w_{n+1}$, and for all $1 \leq i \leq n$, it holds $w_i = w_i' A_i w_i''$, $w_{i+1} = w_i' x_i w_i''$, for some $w_i', w_i'' \in (N \cup T)^*$. The language generated by G is defined by $L(G) = \{w \in T^* \mid S \Rightarrow^* w\}$.

The family of languages generated by matrix grammars without appearance checking is denoted by MAT. It is known that $PsMAT \subset PsRE$. The power of matrix grammars (without appearance checking) is not decreased if we only work with matrix grammars in the *f-binary normal form* [28], [32].

A matrix grammar $G = (N, T, S, M)$ is in the *f-binary normal form* if $N = N_1 \cup N_2 \cup \{S, f\}$, where these three sets are mutually disjoint, and each matrix in M has one of the following forms:

1. $(S \to XA)$, with $X \in N_1, A \in N_2$,
2. $(X \to Y, A \to x)$, with $X, Y \in N_1, A \in N_2, x \in (N_2 \cup T)^*$, $|x| \leq 2$,
3. $(X \to f, A \to x)$, with $X \in N_1, A \in N_2$, and $x \in T^*$, $|x| \leq 2$,
4. $(f \to \lambda)$.

Moreover, there is only one matrix of type 1 and only one matrix of type 4, which is only used in the last step of a derivation.

A useful characterization of the family $PsRE$ of recursively enumerable sets of vectors of natural numbers is obtained by means of *register machines*. A register machine is a tuple $M = (m, H, l_0, l_h, I)$, where

- m is the number of registers;
- H is a set of labels;
- $l_0, l_h \in H$ are distinguished labels, where l_0 is the label of the initial instruction, and l_h is the label of the halting instruction;

- I is a set of labeled program instructions of the following forms (each label from H labels only one instruction from I, thus identifying it precisely):
  - l_i : (ADD(r), l_j, l_k)(add 1 to register r and then go to one of the instructions with labels l_j, l_k, non-deterministically chosen);
  - l_i : (SUB(r), l_j, l_k)(if register r is non-zero, then subtract 1 from it, and go to the instruction with label l_j; otherwise, go to the instruction with label l_k);
  - l_h : HALT(the halt instruction).

A register machine M generates a vector $(s_1, \ldots, s_k)$ of natural numbers in the following way: the machine starts with all registers being empty (i.e., storing the number zero); the machine applies the instruction with label l_0 and continues to apply instructions as indicated by the labels (and made possible by the contents of registers); if it reaches the halt instruction, then the register machine with the first k registers containing the numbers $s_1, \ldots, s_k$ is said to be generated by M. If the computation does not halt, then no number is generated. It is known that register machines generate all sets of vectors of natural numbers which are Turing computable, hence they characterize $PsRE$ [33].

III. Tissue P Systems With Channel States and Cell Division

A. Tissue P Systems With Channel States and Cell Division

Tissue P systems with channel states were defined in [28]. Here we introduce cell division into tissue P systems with channel states.

Definition 3.1: A tissue P system with channel states and cell division, of degree $q \geq 1$, is a tuple $\Pi = (\Gamma, T, K, \mathcal{E}, \mathcal{M}_1, \ldots, \mathcal{M}_q, ch, (s_{(i,j)})_{(i,j)\in ch}, (\mathcal{R}_{(i,j)})_{(i,j)\in ch}, i_{out})$, where

- Γ is an alphabet of objects;
- $T \subseteq \Gamma$ is an alphabet of terminal objects;
- K is an alphabet of states (not necessarily disjoint from Γ);
- $\mathcal{E} \subseteq \Gamma$ is a set of objects initially located in the environment of the system, all available in an arbitrary number of copies;
- $\mathcal{M}_i$, $1 \leq i \leq q$, are multisets of objects (symbols of alphabet Γ) initially placed in the q cells of the system;
- $ch \subseteq \{(i, j) \mid i, j \in \{0, 1, \ldots, q\}, i \neq j\}$ is a set of channels between cells or between a cell and the environment such that for $i, j \in \{0, 1, \ldots, q\}$ at most one of $(i, j), (j, i)$ appears in ch (0 is the label of the environment);
- $s_{i,j}$ is an initial state of channel $(i, j) \in ch$;
- $\mathcal{R}_{(i,j)}$ is a finite set of rules of the following forms (associated with the channel $(i, j) \in ch$):
  - Communication rules:
    - Symport rules: $(s, u/\lambda, s')$ or $(s, \lambda/u, s')$, where $s, s' \in K$, $u \in M_f(\Gamma)$, $|u| > 0$;
    - Antiport rules: $(s, u/v, s')$, where $s, s' \in K$, $u, v \in M_f(\Gamma)$, $|u| > 0$, $|v| > 0$;
  - Division rules:
    - $[a]_i \rightarrow [b]_i[c]_i$, where $i \in \{1, \ldots, q\}$, $i \neq i_{out}$, $a, b, c \in \Gamma$;
- $i_{out} \in \{0, 1, ..., q\}$ is the output region.

If a system has no cell division rules, then it is simply called a tissue P system with channel states.

We also note the important restriction that there is at most one channel between two given cells, and the channel is given as an ordered pair (i, j), with which a state from K is associated. This does not restrict the communication between two cells or between a cell and the environment, because the movement of objects in the two directions of a channel is allowed. The length of a rule $(s, u/\lambda, s')$ or $(s, \lambda/u, s')$ (resp., $(s, u/v, s')$) is defined as $|u|$ (resp., $|u| + |v|$).

A *configuration* of a tissue P system with channel states and cell division at any instant is described by all multisets of objects over Γ associated with all cells in the system, all states associated with each channel and the multiset of objects over $\Gamma \setminus \mathcal{E}$ associated with the environment at that moment. The *initial configuration* is $(\mathcal{M}_1, \ldots, \mathcal{M}_q, (s_{(i,j)})_{(i,j)\in ch}, \emptyset)$.

A division rule $[a]_i \rightarrow [b]_i[c]_i$ is applicable to a configuration at a moment if the following conditions hold: (1) cell i contains object a; (2) cell i is not the output cell. When applying such a rule, cell i is divided into two cells with the same label: in the first copy, object a is replaced by object b, in the second one object a is replaced by object c, and all the objects in the original cell, different from the object triggering the rule, are replicated in the two new cells.

A symport rule $(s, u/\lambda, s') \in \mathcal{R}_{ij}$ (resp., $(s, \lambda/u, s') \in \mathcal{R}_{ij}$) is applicable to a configuration at a moment if the channel between region i and region j has the state s and region i contains multiset u (resp., region j contains multiset u) at that moment. When such a rule is applied, multiset u is sent to region j (resp., region i) and the channel state between region i and region j is changed from s to s'.

An antiport rule $(s, u/v, s') \in \mathcal{R}_{ij}$ is applicable to a configuration at a moment if the channel between region i and region j has the state s, and region i contains multiset u as well as region j contains multiset v at that moment. When such a rule is applied, multiset u from region i is sent to region j, at the same time multiset v enters region i from region j, and the channel state between region i and region j is changed from s to s'.

The rules of a tissue P system with channel states and cell division considered in this work are applied in a flat maximally parallel manner at the level of each channel (at each step, on each channel, a maximal set of applicable rules which pass from a given state to a unique next state, is chosen and each rule in the set is applied once) and in a parallel manner at the level of the system (all channels which can use rules in a flat maximally parallel manner must do it) with the following important restriction: when a cell is divided, the division rule is the only one which is applied for that cell at that step, the objects inside that cell do not evolve by means of communication rules, in other words, before division, a cell interrupts all its communication channels with the other cells and with the environment. The new cells resulting from division will recover the communication channels (note that

the states of the corresponding channels are not changed) and interact with other cells or with the environment only at the next step, providing that they do not divide once again.

Starting from the initial configuration and applying rules as described above, one obtains a sequence of consecutive configurations. Each passage from a configuration to a successor configuration is called a *transition*. A configuration is a *halting* one if no rule of the system is applicable to it. A sequence of transitions starting from the initial configuration is a *computation*. Only a computation reaching a halting configuration gives a result, encoded by the vector which describes the multiplicity of objects from T present in the output region i_{out}.

The set of all vectors computed in the way mentioned above by a system Π is denoted by $Ps(\Pi)$. The family of all sets of vectors computed by systems with at most m cells initially present in the system, k states, and using symport rules of length at most t_1, antiport rules of length at most t_2 is denoted by $PsOtP_m(state_k, sym_{t_1}, anti_{t_2}, flat)$, where $flat$ stands for a flat maximally parallel use of rules on channels. If one of the parameters m, k, t_1, t_2 is not bounded, then it is replaced with $*$.

B. Recognizer Tissue P Systems With Channel States and Cell Division

Recognizer tissue P systems were introduced in [24], and they provide a natural framework to solve decision problems by means of computational devices in membrane computing.

Definition 3.2: A recognizer tissue P system with channel states and cell division of degree $q \geq 1$ is a tuple $\Pi = (\Gamma, T, K, \Sigma, \mathcal{E}, \mathcal{M}_1, \ldots, \mathcal{M}_q, ch, (s_{(i,j)})_{(i,j)\in ch}, (\mathcal{R}_{(i,j)})_{(i,j)\in ch}, i_{in}, i_{out})$, where

- $(\Gamma, T, K, \mathcal{E}, \mathcal{M}_1, \ldots, \mathcal{M}_q, ch, (s_{(i,j)})_{(i,j)\in ch}, (\mathcal{R}_{(i,j)})_{(i,j)\in ch}, i_{out})$ is a tissue P system with channel states and cell division of degree $q \geq 1$;
- Γ has two distinguished objects `yes` and `no`;
- Σ is an (input) alphabet strictly contained in Γ;
- $\mathcal{M}_1, \ldots, \mathcal{M}_q$ are finite multisets over $\Gamma \setminus \Sigma$;
- $i_{in} \in \{1, \ldots, q\}$ is the input cell, and $i_{out} = 0$;
- all computations halt;
- if $\mathcal{C}$ is a computation of Π, then either object `yes` or object `no` (but not both) must have been released into the environment, and only at the last step of the computation.

For each finite multiset $w \in \Sigma$, the *computation* of a tissue P system with channel states and cell division with input w starts from a configuration of the form $(\mathcal{M}_1, \ldots, \mathcal{M}_{i_{in}} + w, \ldots, \mathcal{M}_q, \emptyset)$, that is, the input multiset w has been added to the contents of the input cell i_{in}, and we denote it by $\Pi + w$. Therefore, we have an initial configuration associated with each input multiset w (over the input alphabet Σ) in this kind of P systems.

For a recognizer tissue P system with channel states and cell division, a computation $\mathcal{C}$ is said to be an accepting computation (resp., rejecting computation) if object `yes` (resp., object `no`) appears in the environment associated with the corresponding halting configuration of $\mathcal{C}$, and neither object `yes` nor `no` appears in the environment associated with any non-halting configuration of $\mathcal{C}$.

In what follows, we define what means solving a decision problem in the framework of tissue P systems efficiently and in a uniform way. Bearing in mind that they provide devices with a finite description, a countable family of tissue P systems will be necessary in order to solve a decision problem.

Definition 3.3: A decision problem $X = (I_X, \theta_X)$ is solvable in polynomial time by a family $\mathbf{\Pi} = \{\Pi(n) \mid n \in \mathbb{N}\}$ of recognizer tissue P systems with channel states and cell division in a uniform way, if the following conditions hold:

- the family $\mathbf{\Pi}$ is polynomially uniform by Turing machines, that is, there exists a deterministic Turing machine working in polynomial time which constructs the system $\Pi(n)$ from $n \in \mathbb{N}$;
- there exists a pair (cod, s) of polynomial-time computable functions over I_X such that:
  - for each instance $u \in I_X$, $s(u)$ is a natural number and $cod(u)$ is an input multiset of the system $\Pi(s(u))$;
  - for each $n \in \mathbb{N}$, $s^{-1}(n)$ is a finite set;
  - the family $\mathbf{\Pi}$ is polynomially bounded with regard to (X, cod, s), that is, there exists a polynomial function p, such that for each $u \in I_X$ every computation of $\Pi(s(u)) + cod(u)$ is halting and it performs at most $p(|u|)$ steps;
  - the family $\mathbf{\Pi}$ is sound with regard to (X, cod, s), that is, for each $u \in I_X$, if there exists an accepting computation of $\Pi(s(u)) + cod(u)$, then $\theta_X(u) = 1$;
  - the family $\mathbf{\Pi}$ is complete with regard to (X, cod, s), that is, for each $u \in I_X$, if $\theta_X(u) = 1$, then every computation of $\Pi(s(u)) + cod(u)$ is an accepting one.

We denote by $\mathbf{PMC}_{\mathbf{T}_s^f\mathbf{DC}(k)}$ (resp., $\mathbf{PMC}_{\mathbf{T}^f\mathbf{DC}(k)}$) the set of all decision problems which can be solved in a uniform way and polynomial time by means of recognizer tissue P systems with channel states, communication rules of length at most k and cell division (resp., recognizer tissue P systems with cell division and communication rules of length at most k) working in a flat maximally parallel manner.

IV. Computational Power of Tissue P Systems With Channel States Working in the Flat Maximally Parallel Manner

In this section, we investigate the computational power of tissue P systems with channel states working in the flat maximally parallel manner.

Theorem 4.1: $PsOtP_*(state_*, anti_2, flat) \subseteq PsFIN$.

Proof: The proof follows from the fact that the number of objects in a cell cannot be changed by using only antiport rules of length 2, therefore the number of objects in the tissue P system with channel states and antiport rules of length 2 working in a flat maximally parallel manner will not change during any sequence of transitions starting from the initial configuration and ending with a halting configuration (the number of channel states is finite during this process). Hence, only finite sets of vectors of natural numbers can be generated.

Theorem 4.2: $PsMAT \subseteq PsOtP_1(state_*, sym_1, flat)$.

Proof: Let us consider a matrix grammar $G = (N_1 \cup N_2 \cup \{S, f\}, T, S, M)$ in the f-binary normal form with $n+1$ matrices labelled as $m_0, \ldots, m_n$, where $m_0 = (S \to X_0 A_0)$ is the initial matrix of M.

Let us assume that we have k_1 matrices of the form m_i: $(X \to Y, A \to x)$, $X \in N_1$, $Y \in N_1 \cup \{f\}$, $A \in N_2$, $x \in N_2 \cup T \cup \{\lambda\}$, $1 \leq i \leq k_1$; k_2 matrices are of the form m_i: $(X \to Y, A \to x_1 x_2)$, $X \in N_1$, $Y \in N_1 \cup \{f\}$, $A \in N_2$, $x_1, x_2 \in N_2 \cup T$, $k_1 + 1 \leq i \leq k_1 + k_2$; $m_n = (f \to \lambda)$, such that $k_1 + k_2 = n - 1$. We also assume that $t = |N_2|$ with all the objects in N_2 labelled as $1, 2, \ldots, t$.

We construct the tissue P system with channel states Π to simulate the matrix grammar G.

$$\Pi = (\Gamma, T, K, \Gamma, \{A_0\}, \{(0,1)\}, X_0, \mathcal{R}_{(0,1)}, 1),$$

where

$$\begin{aligned}
\Gamma &= N_2 \cup T;\\
K &= N_1 \cup \{f\} \cup \{Y_i \mid 1 \leq i \leq k_1 + k_2\}\\
&\quad \cup \{f_i \mid 1 \leq i \leq t\};\\
\mathcal{R}_{(0,1)} &= \{(X, \lambda/A, Y_i), (Y_i, x/\lambda, Y) \mid m_i :\\
&\quad (X \to Y, A \to x), X \in N_1, Y \in N_1 \cup \{f\},\\
&\quad A \in N_2, x \in N_2 \cup T \cup \{\lambda\}, 1 \leq i \leq k_1\}\\
&\quad \cup \{(X, \lambda/A, Y_i), (Y_i, x_1/\lambda, Y), (Y_i, x_2/\lambda, Y) \mid\\
&\quad m_i : (X \to Y, A \to x_1 x_2), X \in N_1, Y \in N_1 \cup\\
&\quad \{f\}, A \in N_2, x_1, x_2 \in N_2 \cup T, k_1 + 1 \leq i \leq k_1 + k_2\}\\
&\quad \cup \{(f, \lambda/A, f_i), (f_i, A/\lambda, f) \mid A \in N_2, 1 \leq i \leq t\}.
\end{aligned}$$

A matrix m_i $(1 \leq i \leq k_1)$ can be simulated in two steps. At step 1, under the influence of state X on channel $(0,1)$, object A is sent to the environment from cell 1, and the channel state is changed to Y_i. At the next step, object x is sent into cell 1, and the channel state is changed from Y_i to Y.

A matrix m_i $(k_1 + 1 \leq i \leq k_1 + k_2)$ is simulated in the following way. At step 1, by using rule $(X, \lambda/A, Y_i)$, object A is sent to the environment, the channel state is changed from X to Y_i. At step 2, under the control of channel state Y_i, rules $(Y_i, x_1/\lambda, Y)$ and $(Y_i, x_2/\lambda, Y)$ are applied simultaneously, one copy of object x_1 and one copy of object x_2 are sent into cell 1 because of the flat maximally parallel use of rules on the channel, and the state of channel $(0,1)$ is changed from Y_i to Y.

When the state f is introduced, the system checks whether the derivation in G is terminal and only in the affirmative case it halts. If the state is f and there exists at least one non-terminal symbol, then by using rules $(f, \lambda/A, f_i)$ and $(f_i, A/\lambda, f)$ (in this case, object A in N_2 is labelled by i), the computation never halts. Consequently, $\Psi_T(L(G)) = Ps(\Pi)$.

It is known that P systems with only symport rules of length 3 in one membrane are computationally complete (see Theorem 2.2 in Chapter 5 from [18]). This result also holds for the case of tissue P systems with one cell, one channel state and symport rules of length at most 3 working in the flat maximally parallel way. Hence we obtain the following theorem.

Theorem 4.3: $PsOtP_1(state_1, sym_3, flat) = PsRE$.

In what follows, we prove that tissue P systems with two cells, any number of states and only using noncooperative symport rules, or arbitrarily many cells, four states and only using noncooperative symport rules are Turing universality.

Theorem 4.4: $PsOtP_2(state_*, sym_1, flat) = PsRE$.

Proof: We only prove the inclusion $PsOtP_2(state_*, sym_1, flat) \supseteq PsRE$. The reverse inclusion $PsOtP_2(state_*, sym_1, flat) \subseteq PsRE$ follows from the Church-Turing thesis.

Let us consider a register machine $M = (m, H, l_0, l_h, I)$ generating the set of vectors $N(M) \subseteq \mathbf{N}^k$, for some $k \geq 1$. We construct the tissue P system with channel states Π to simulate the register machine M. $\Pi = (\Gamma, T, K, \mathcal{E}, \mathcal{M}_1, \mathcal{M}_2, \{(0,1),(1,2)\}, l_0, s, \mathcal{R}_{(0,1)}, \mathcal{R}_{(1,2)}, 1)$, where

- $\Gamma = \{a_i \mid 1 \leq i \leq m\} \cup \{b, b', b'' \mid b \in H\} \cup \{c\}$,
- $T = \{a_i \mid 1 \leq i \leq k\}$,
- $K = \{s, s', s'', s'''\} \cup \{l, l', l'', l''', l^{iv}, l^{v}, l^{vi}, l^{viii} \mid l \in H\}$,
- $\mathcal{E} = \Gamma - (\{b, b' \mid b \in H\} \cup \{c\})$,
- $\mathcal{M}_1 = \{b, b' \mid b \in H\}$, $\mathcal{M}_2 = \{c\}$,

and the sets $\mathcal{R}_{(0,1)}$, $\mathcal{R}_{(1,2)}$ of rules are as follows:

- For each ADD instruction $l_i : (\texttt{ADD}(r), l_j, l_k)$ of M, we introduce the following rules in $\mathcal{R}_{(0,1)}$:

$$\begin{aligned}
r_1 &\equiv (l_i, \lambda/b_i, l_i'),\\
r_2 &\equiv (l_i', b_i/\lambda, l_j),\\
r_3 &\equiv (l_i', a_r/\lambda, l_j),\\
r_4 &\equiv (l_i', b_i/\lambda, l_k),\\
r_5 &\equiv (l_i', a_r/\lambda, l_k).
\end{aligned}$$

An ADD instruction l_i is simulated in two steps. At step 1, under the control of state l_i on channel $(0,1)$, object b_i is sent to the environment by using rule r_1, and the channel state is changed from l_i to l_i'. At the next step, one of sets of rules $\{r_2, r_3\}$ and $\{r_4, r_5\}$ is non-deterministically chosen and used. By applying rules r_2 and r_3 (resp., r_4 and r_5) in a flat maximally parallel way, one copy of object b_i and one copy of object a_r are sent into cell 1 (simulating that the number stored in register r is increased by one), and the state of channel $(0,1)$ is changed to l_j (resp., l_k). Hence, the system starts to simulate the instruction with label l_j or l_k. Clearly, instruction l_i of M is correctly simulated by Π.

- For each SUB instruction $l_i : (\texttt{SUB}(r), l_j, l_k)$ of M,
  - we introduce the following rules in $\mathcal{R}_{(0,1)}$:

$$\begin{aligned}
r_6 &\equiv (l_i, \lambda/b_i', l_i'),\\
r_7 &\equiv (l_i', b_i'/\lambda, l_i''),\\
r_8 &\equiv (l_i', b_i''/\lambda, l_i''),\\
r_9 &\equiv (l_i'', \lambda/a_r, l_i'''),\\
r_{10} &\equiv (l_i'', \lambda/c, l_i^{iv}),\\
r_{11} &\equiv (l_i''', \lambda/c, l_i^{v}),\\
r_{12} &\equiv (l_i^{iv}, c/\lambda, l_i^{vi}),\\
r_{13} &\equiv (l_i^{v}, c/\lambda, l_i^{viii}),\\
r_{14} &\equiv (l_i^{vi}, \lambda/b_i'', l_k),\\
r_{15} &\equiv (l_i^{viii}, \lambda/b_i'', l_j);
\end{aligned}$$

  - and we introduce the following rules in $\mathcal{R}_{(1,2)}$:

$$\begin{aligned}
r_{16} &\equiv (s, b_i''/\lambda, s'),\\
r_{17} &\equiv (s', \lambda/c, s''),\\
r_{18} &\equiv (s'', \lambda/b_i'', s'''),\\
r_{19} &\equiv (s''', c/\lambda, s).
\end{aligned}$$

TABLE I

THE APPLICATION OF RULES IN $\mathcal{R}_{(0,1)}$ AND $\mathcal{R}_{(1,2)}$, THE EVOLUTION OF CHANNEL STATES $s_{(0,1)}$ AND $s_{(1,2)}$, AND THE REWRITING OF MULTISETS $\mathcal{M}_1$ AND $\mathcal{M}_2$ IN CELLS 1 AND 2, RESPECTIVELY, DURING THE SIMULATION OF A SUB INSTRUCTION l_i : (SUB$(r), l_j, l_k$) WITH REGISTER r NOT EMPTY, WHERE z, z' ARE MULTISETS OF OBJECTS FROM THE SET $\{a_1, \ldots, a_m\}$, AND $\mathcal{R}_{(1,2)}0$ ARE MULTISETS WHICH CONTAIN EACH $\mathcal{R}_{(1,2)}1$ EXACTLY ONCE, RESPECTIVELY

Step	$\mathcal{R}_{(0,1)}$	$\mathcal{R}_{(1,2)}$	$s_{(0,1)}$	$s_{(1,2)}$	$\mathcal{M}_1$	$\mathcal{M}_2$
0	$-$	$-$	l_i	s	$z + u(H) + v(H)$	$\{c\}$
1	r_6	$-$	l_i'	s	$z + u(H) + (v(H) \setminus \{b_i'\})$	$\{c\}$
2	r_7, r_8	$-$	l_i''	s	$z + u(H) + v(H) + \{b_i''\}$	$\{c\}$
3	r_9	r_{16}	l_i'''	s'	$z' + u(H) + v(H)$	$\{c, b_i''\}$
4	$-$	r_{17}	l_i'''	s''	$z' + u(H) + v(H) + \{c\}$	$\{b_i''\}$
5	r_{11}	r_{18}	l_i^v	s'''	$z' + u(H) + v(H) + \{b_i''\}$	$-$
6	r_{13}	$-$	l_i^{vii}	s'''	$z' + u(H) + v(H) + \{c, b_i''\}$	$-$
7	r_{15}	r_{19}	l_j	s	$z' + u(H) + v(H)$	$\{c\}$

TABLE II

THE APPLICATION OF RULES IN $\mathcal{R}_{(0,1)}$ AND $\mathcal{R}_{(1,2)}$, THE EVOLUTION OF CHANNEL STATES $s_{(0,1)}$ AND $s_{(1,2)}$, AND THE REWRITING OF MULTISETS $\mathcal{M}_1$ AND $\mathcal{M}_2$ IN CELLS 1 AND 2, RESPECTIVELY, DURING THE SIMULATION OF A SUB INSTRUCTION l_i : (SUB$(r), l_j, l_k$) WITH REGISTER r EMPTY, WHERE z IS A MULTISET OF OBJECTS FROM THE SET $\{a_1, \ldots, a_m\}$, AND $\mathcal{R}_{(1,2)}0$ ARE MULTISETS WHICH CONTAIN EACH $\mathcal{R}_{(1,2)}1$ EXACTLY ONCE, RESPECTIVELY

Step	$\mathcal{R}_{(0,1)}$	$\mathcal{R}_{(1,2)}$	$s_{(0,1)}$	$s_{(1,2)}$	$\mathcal{M}_1$	$\mathcal{M}_2$
0	$-$	$-$	l_i	s	$z + u(H) + v(H)$	$\{c\}$
1	r_6	$-$	l_i'	s	$z + u(H) + (v(H) \setminus \{b_i'\})$	$\{c\}$
2	r_7, r_8	$-$	l_i''	s	$z + u(H) + v(H) + \{b_i''\}$	$\{c\}$
3	$-$	r_{16}	l_i''	s'	$z + u(H) + v(H)$	$\{c, b_i''\}$
4	$-$	r_{17}	l_i''	s''	$z + u(H) + v(H) + \{c\}$	$\{b_i''\}$
5	r_{10}	r_{18}	l_i^{iv}	s'''	$z + u(H) + v(H) + \{b_i''\}$	$-$
6	r_{12}	$-$	l_i^{vi}	s'''	$z + u(H) + v(H) + \{c, b_i''\}$	$-$
7	r_{14}	r_{19}	l_k	s	$z + u(H) + v(H)$	$\{c\}$

A SUB instruction l_i is simulated in the following way. At step 1, under the influence of state l_i on channel $(0, 1)$, object b_i' is sent to the environment by using rule r_6, and the channel state is changed to l_i'. At the next step, due to the flat maximally parallel use of rules on channel $(0, 1)$, only one copy of object b_i' and one copy of object b_i'' are sent into cell 1 by applying rules r_7, r_8, which change state l_i' to the same state l_i''. In what follows, we have two cases.

- There is at least one copy of object a_r in cell 1 (corresponding to the fact that the number stored in register r is greater than 0). In this case, at step 3, rules r_9, r_{16} are enabled, by using rule r_9, one copy of object a_r is sent to the environment, and the state of channel $(0, 1)$ is changed from l_i'' to l_i'''; by applying rule r_{16}, object b_i'' is sent into cell 2 from cell 1, the state of channel $(1, 2)$ is changed to s'. At step 4, rule r_{17} can be used, object c is sent to cell 1, which will be sent to the environment at the next step by using rule r_{11} (the state of channel $(0, 1)$ is changed to l_i^v); when the state of channel $(1, 2)$ is changed to s'', object b_i'' is sent to cell 1 from cell 2 by using rule r_{18}, changing the state of its channel from s'' to s'''. At step 6, object c is sent into cell 1 from the environment by using rule r_{13}, the state of channel $(0, 1)$ is changed to l_i^{vii}, which will be changed to state l_j at the next step by applying rule r_{15}; object c is sent back to cell 2 by using rule r_{19} at step 7, and the state of channel $(1, 2)$ is changed to s. In this case, one copy of object a_r is consumed in cell 1 (simulating that the number stored in register r is decreased by one), and the system starts to simulate the instruction l_j (see Table I).
- There is no object a_r in cell 1 (corresponding to the fact that the number stored in register r is 0). In this case, at step 3, only rule r_{16} can be used, object b_i'' is sent into cell 2 and object c will be sent to cell 1 at the next step by using rule r_{17}. At the next two steps, by applying rules r_{10} and r_{12} one by one, the state of channel $(0, 1)$ will be changed from l_i^{iv} to l_i^{vi}; rule r_{18} is enabled and applied at step 5, object b_i'' is sent to cell 1 from cell 2, changing the state of its channel to s'''. At step 7, the state of channel $(0, 1)$ is changed to l_k by using rule r_{14}; simultaneously, by applying rule r_{19}, object c is sent back to cell 2, and the state of channel $(1, 2)$ is changed to s again. Hence, the system starts to simulate the instruction l_k (see Table II).

Hence, the SUB instruction of M is correctly simulated by system Π.

When the state of channel $(0, 1)$ is l_h, no rule can be used in the system, and the computation halts. The numbers of copies of objects a_i $(1 \leq i \leq k)$ in cell 1 correspond to the result of the computation, hence $N(M) = Ps(\Pi)$.

Theorem 4.5: $PsOtP_*(state_4, sym_1, flat) = PsRE$.

Proof: We only prove the inclusion $PsOtP_*(state_4, sym_1, flat) \supseteq PsRE$. The reverse inclusion follows from the Church-Turing thesis.

Let us consider a register machine $M = (m, H, l_0, l_h, I)$, where the number of ADD instructions is p (labelled by $add_1, \ldots, add_p$), the number of SUB instructions is q (labelled by $sub_1, \ldots, sub_q$), and such that the register machine generates the set of vectors $N(M) \subseteq \mathbf{N}^k$, for some $k \geq 1$. We construct the tissue P system with channel states of degree $p + q + 1$, Π, to simulate the register machine M,

$$\Pi = (\Gamma, T, K, \mathcal{E}, \mathcal{M}_1, \ldots, syn, s, \ldots, s, \mathcal{R}_{(0,1)}, \ldots, 1),$$

TABLE III

THE APPLICATION OF RULES IN $\mathcal{R}_{(1,add_i)}$ AND $\mathcal{R}_{(0,add_i)}$, THE EVOLUTION OF CHANNEL STATES $s_{(1,add_i)}$ AND $s_{(0,add_i)}$, AND THE REWRITING OF MULTISETS $\mathcal{M}_1$ AND $\mathcal{M}_{add_i}$ IN CELLS 1 AND add_i, RESPECTIVELY, DURING THE SIMULATION OF AN ADD INSTRUCTION l_i : (ADD(r), l_j, l_k), WHERE z IS A MULTISET OF OBJECTS FROM THE SET $\{a_1, \ldots, a_m, e\}$

Step	$\mathcal{R}_{(1,add_i)}$	$\mathcal{R}_{(0,add_i)}$	$s_{(1,add_i)}$	$s_{(0,add_i)}$	$\mathcal{M}_1$	$\mathcal{M}_{add_i}$
0	$-$	$-$	s	s	$z + \{l_i\}$	$-$
1	r_1	$-$	s'	s	z	$\{l_i\}$
2	$-$	r_8	s'	s'	z	$-$
3	$-$	r_9, r_{10}, r_{11}	s'	s''	z	$\{a_r, l_j, l_k\}$
4	(r_2, r_3) or (r_4, r_5)	r_{12}	s'' or s'''	s'''	$z + \{a_r, l_j\}$ or $z + \{a_r, l_k\}$	$\{e, l_k\}$ or $\{e, l_j\}$
5	r_6 or r_7	r_{14} or r_{13}	s	s	$z + \{a_r, l_j, e\}$ or $z + \{a_r, l_k, e\}$	$-$

where

- $\Gamma = \{a_i \mid 1 \leq i \leq m\} \cup \{l \mid l \in H\} \cup \{e\}$,
- $T = \{a_i \mid 1 \leq i \leq k\}$,
- $K = \{s, s', s'', s'''\}$,
- $\mathcal{E} = \Gamma$,
- $\mathcal{M}_1 = \{l_0\}$, $\mathcal{M}_{add_i} = \emptyset$, $1 \leq i \leq p$,
- $\mathcal{M}_{sub_i} = \emptyset$, $1 \leq i \leq q$,
- $syn = \{(0,1)\} \cup \{(1, add_i), (0, add_i) \mid 1 \leq i \leq p\} \cup \{(1, sub_i), (0, sub_i) \mid 1 \leq i \leq q\}$,

and the sets of rules associated with channels are as follows (the value of register r is represented as the number of copies of object a_r in cell 1):

- For each ADD instruction l_i : (ADD(r), l_j, l_k) of M,
  - we introduce the following rules in $\mathcal{R}_{(1,add_i)}$:

$$
\begin{aligned}
r_1 &\equiv (s, l_i/\lambda, s'),\\
r_2 &\equiv (s', \lambda/a_r, s''),\\
r_3 &\equiv (s', \lambda/l_j, s''),\\
r_4 &\equiv (s', \lambda/a_r, s'''),\\
r_5 &\equiv (s', \lambda/l_k, s'''),\\
r_6 &\equiv (s'', \lambda/e, s),\\
r_7 &\equiv (s''', \lambda/e, s);
\end{aligned}
$$

  - and we introduce the following rules in $\mathcal{R}_{(0,add_i)}$:

$$
\begin{aligned}
r_8 &\equiv (s, \lambda/l_i, s'),\\
r_9 &\equiv (s', a_r/\lambda, s''),\\
r_{10} &\equiv (s', l_j/\lambda, s''),\\
r_{11} &\equiv (s', l_k/\lambda, s''),\\
r_{12} &\equiv (s'', e/\lambda, s'''),\\
r_{13} &\equiv (s''', \lambda/l_j, s),\\
r_{14} &\equiv (s''', \lambda/l_k, s).
\end{aligned}
$$

An ADD instruction l_i is simulated in five steps. At step 1, rule r_1 is used, object l_i is sent to cell add_i (the state of channel $(1, add_i)$ is changed to s'), which will be sent to the environment at the next step by applying rule r_8 (the state of channel $(0, add_i)$ is changed to s') (we assume that $l_j \neq l_i$ and $l_k \neq l_i$). At step 3, by using rules r_9, r_{10}, r_{11} in a flat maximally parallel way, one copy of object a_r, one copy of object l_j and one copy of object l_k will be sent into cell add_i, changing the state of its channel to s''. At step 4, with the presence of state s'' on channel $(0, add_i)$, rule r_{12} is used, one copy of object e is sent to cell add_i due to the flat maximally parallel use of rules (the channel state is changed to s'''), and this object will be sent to cell 1 at the next step by using rule r_6 or r_7 (the state of channel $(1, add_i)$ is changed to s); one of sets of rules $\{r_2, r_3\}$ and $\{r_4, r_5\}$ is non-deterministically chosen and used at step 4, by applying rules r_2, r_3 (resp., r_4, r_5), objects a_r, l_j (resp., a_r, l_k) are sent to cell 1, the state of channel $(1, add_i)$ is changed from s' to s'' (resp., from s' to s'''). With the appearance of state s''' on channel $(0, add_i)$, rule r_{13} or r_{14} is enabled at step 5; by using one of these rules, the remaining object l_j or l_k is sent to the environment, the state of channel $(0, add_i)$ is changed to s again. Hence, one copy of object a_r is introduced in cell 1 (simulating that the number stored in register r is increased by one), and one copy of object e has been added to cell 1, which in general, at each time step, contains any number of copies of this object (these objects are always idle), the system starts to simulate an instruction with label l_j or l_k. So instruction l_i of M is correctly simulated by Π (see Table III).

- For each SUB instruction l_i : (SUB(r), l_j, l_k) of M,
  - we introduce the following rules in $\mathcal{R}_{(1,sub_i)}$:

$$
\begin{aligned}
r_{15} &\equiv (s, l_i/\lambda, s'),\\
r_{16} &\equiv (s', a_r/\lambda, s''),\\
r_{17} &\equiv (s'', \lambda/l_j, s'''),\\
r_{18} &\equiv (s', \lambda/l_k, s'''),\\
r_{19} &\equiv (s''', \lambda/e, s);
\end{aligned}
$$

  - and we introduce the following rules in $\mathcal{R}_{(0,sub_i)}$:

$$
\begin{aligned}
r_{20} &\equiv (s, \lambda/l_i, s'),\\
r_{21} &\equiv (s', l_j/\lambda, s''),\\
r_{22} &\equiv (s', l_k/\lambda, s''),\\
r_{23} &\equiv (s'', e/\lambda, s'''),\\
r_{24} &\equiv (s''', \lambda/l_j, s),\\
r_{25} &\equiv (s''', \lambda/l_k, s).
\end{aligned}
$$

A SUB instruction l_i is simulated in the following way. At step 1, object l_i is sent to cell sub_i by using rule r_{15}, the state of channel $(1, sub_i)$ is changed to s'. In what follows, we have two cases.

TABLE IV

THE APPLICATION OF RULES IN $\mathcal{R}_{(1,sub_i)}$ AND $\mathcal{R}_{(0,sub_i)}$, THE EVOLUTION OF CHANNEL STATES $s_{(1,sub_i)}$ AND $s_{(0,sub_i)}$, AND THE REWRITING OF MULTISETS $\mathcal{M}_1$ AND $\mathcal{M}_{sub_i}$ IN CELLS 1 AND sub_i, RESPECTIVELY, DURING THE SIMULATION OF A SUB INSTRUCTION l_i : (SUB$(r), l_j, l_k$) WITH REGISTER r NOT EMPTY, WHERE z, z' ARE MULTISETS OF OBJECTS FROM THE SET $\mathcal{R}_{(0,sub_i)}0, \mathcal{R}_{(0,sub_i)}1$

Step	$\mathcal{R}_{(1,sub_i)}$	$\mathcal{R}_{(0,sub_i)}$	$s_{(1,sub_i)}$	$s_{(0,sub_i)}$	$\mathcal{M}_1$	$\mathcal{M}_{sub_i}$
0	$-$	$-$	s	s	$z + \{l_i\}$	$-$
1	r_{15}	$-$	s'	s	z	$\{l_i\}$
2	r_{16}	r_{20}	s''	s'	z'	$\{a_r\}$
3	$-$	r_{21}, r_{22}	s''	s''	z'	$\{a_r, l_j, l_k\}$
4	r_{17}	r_{23}	s'''	s'''	$z' + \{l_j\}$	$\{a_r, e, l_k\}$
5	r_{19}	r_{25}	s	s	$z' + \{e, l_j\}$	$\{a_r\}$

TABLE V

THE APPLICATION OF RULES IN $\mathcal{R}_{(1,sub_i)}$ AND $\mathcal{R}_{(0,sub_i)}$, THE EVOLUTION OF CHANNEL STATES $s_{(1,sub_i)}$ AND $s_{(0,sub_i)}$, AND THE REWRITING OF MULTISETS $\mathcal{M}_1$ AND $\mathcal{M}_{sub_i}$ IN CELLS 1 AND sub_i, RESPECTIVELY, DURING THE SIMULATION OF A SUB INSTRUCTION l_i : (SUB$(r), l_j, l_k$) WITH REGISTER r EMPTY, WHERE z IS A MULTISET OF OBJECTS FROM THE SET $\mathcal{R}_{(0,sub_i)}0$

Step	$\mathcal{R}_{(1,sub_i)}$	$\mathcal{R}_{(0,sub_i)}$	$s_{(1,sub_i)}$	$s_{(0,sub_i)}$	$\mathcal{M}_1$	$\mathcal{M}_{sub_i}$
0	$-$	$-$	s	s	$z + \{l_i\}$	$-$
1	r_{15}	$-$	s'	s	z	$\{l_i\}$
2	$-$	r_{20}	s'	s'	z	$-$
3	$-$	r_{21}, r_{22}	s'	s''	z	$\{l_j, l_k\}$
4	r_{18}	r_{23}	s'''	s'''	$z + \{l_k\}$	$\{e, l_j\}$
5	r_{19}	r_{24}	s	s	$z + \{e, l_k\}$	$-$

- There is at least one copy of object a_r in cell 1 (corresponding to the fact that the number stored in register r is greater than 0). In this case, at step 2, rules r_{16} and r_{20} are enabled (we assume that $l_k \neq l_i$). By using rule r_{16} in a flat maximally parallel way, one copy of object a_r is sent to cell sub_i from cell 1, changing its channel state from s' to s''; by applying rule r_{20}, object l_i is sent to the environment, the state of channel $(0, sub_i)$ is changed from s to s'. At the next step, by using rules r_{21}, r_{22} in a flat maximally parallel way, one copy of object l_j and one copy of object l_k are sent into cell sub_i, which lead from a state s' to the same state s''. At step 4, rule r_{23} is enabled and used, one copy of object e is sent to cell sub_i (the state of channel $(0, sub_i)$ is changed to s'''), which will be sent to cell 1 at the next step by using rule r_{19} (the state of channel $(1, sub_i)$ is changed to s). Rule r_{17} is enabled at step 4; by using this rule, object l_j is sent to cell 1, so object l_k will be sent to the environment at the next step by applying rule r_{25}, changing the state of channel $(0, sub_i)$ to s. In this case, one copy of object a_r is consumed in cell 1 (simulating that the number stored in register r is decreased by one), and the system starts to simulate the instruction l_j (see Table IV).
- There is no object a_r in cell 1 (corresponding to the fact that the number stored in register r is 0). In this case, at step 2, only rule r_{20} is applied, the state of channel $(0, sub_i)$ is changed to s' (we assume that $l_k \neq l_i$). With the presence of state s' on this channel, rules r_{21}, r_{22} are enabled and used, one copy of object l_j and one copy of object l_k are sent into cell sub_i, changing the state of channel $(0, sub_i)$ from s' to s''. At step 4, rule r_{23} is used, one copy of object e is sent to cell sub_i (the state of channel $(0, sub_i)$ is changed to s'''), which will be sent to cell 1 at the next step by using rule r_{19} (the state of channel $(1, sub_i)$ is changed to s). Rule r_{18} is applied at step 4, object l_k is sent to cell 1, so object l_j will be sent to the environment at the next step by applying rule r_{24}, changing the state of channel $(0, sub_i)$ to s. Hence, the system starts to simulate the instruction l_k (see Table V).

When object l_h appears in cell 1, by using rule $(s, \lambda/l_h, s)$ in $\mathcal{R}_{(0,1)}$, object l_h is sent to the environment and the computation stops. Note that at the last step, one of rules r_6, r_7, r_{19} is also used except for rule $(s, \lambda/l_h, s)$, and one copy of object e is sent into cell 1. The numbers of copies of objects a_i $(1 \leq i \leq k)$ in cell 1 correspond to the result of the computation, hence $N(M) = Ps(\Pi)$.

V. COMPUTATIONAL EFFICIENCY

In this section, a family of recognizer tissue P systems with channel states and cell division, working in the flat maximally parallel way, is designed for giving a polynomial time and uniform solution to the SAT problem. Moreover, if we consider such P systems without channel states, then only tractable problems can be solved.

Theorem 5.1: SAT $\in \mathbf{PMC}_{\mathbf{T}_s^f \mathbf{DC}(1)}$.

Proof: The SAT problem is a well known **NP**-complete problem [34], which is defined as follows: *given a Boolean formula in conjunctive normal form (CNF), determine whether or not there exists an assignment to its variables such that the formula is evaluated to be true*.

In what follows, we give a polynomial time solution to the SAT problem by a family of recognizer tissue P systems with channel states, noncooperative symport rules and cell division, $\mathbf{\Pi} = \{\Pi(t) \mid t \in \mathbb{N}\}$, working in the flat maximally parallel way. Each system $\Pi(t)$ will process any Boolean formula φ in conjunctive normal form with n variables and m clauses, where $t = \langle n, m \rangle = ((n+m)(n+m+1)/2) + n$, provided that the appropriate input multiset $cod(\varphi)$ is supplied to the system.

We encode the input multiset $cod(\varphi)$ as follows: $cod(\varphi) = \{\alpha_{1,1}, \ldots, \alpha_{1,n}, \ \alpha_{2,1}, \ldots, \alpha_{2,n}, \ldots, \alpha_{m,1}, \ldots, \alpha_{m,n}\}$, where for $1 \leq i \leq m, 1 \leq j \leq n$, we have:

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

For each $n, m \in \mathbb{N}$, we consider the recognizer tissue P system with channel states and cell division (of degree 3) defined as follows: $\Pi(\langle n, m\rangle) = (\Gamma, K, \mathcal{E}, \Sigma, \mathcal{M}_1, \mathcal{M}_2, \mathcal{M}_3, \{(0,1), (0,2), (2,3)\}, d_1, s, s_0, \mathcal{R}_{(0,1)}, \mathcal{R}_{(0,2)}, \mathcal{R}_{(2,3)}, i_{in}, i_{out})$, where

- $\Gamma = \Sigma \cup \{a_i, t_i, f_i \mid 1 \leq i \leq n\} \cup \{c_j \mid 1 \leq j \leq m\} \cup \{a_{n+1}, b, c, e, t, \texttt{yes}, \texttt{no}\}$,
- $\Sigma = \{g_{i,j}, g'_{i,j}, g''_{i,j} \mid 1 \leq i \leq n, 1 \leq j \leq m\}$,
- $K = \{d_i, d'_i, d''_i \mid 1 \leq i \leq n\} \cup \{s_j \mid 0 \leq j \leq 2nm + 4n + m + 2\} \cup \{s_{i,j}, s'_{i,j}, s''_{i,j}, \bar{s}_{i,j}, \bar{s}'_{i,j}, \bar{s}''_{i,j} \mid 1 \leq i \leq n, 1 \leq j \leq m\} \cup \{d_{n+1}, s, s', s'', s_t, s_f\}$,
- $\mathcal{E} = \{a_i \mid 1 \leq i \leq n+1\} \cup \{c_j \mid 1 \leq j \leq m\} \cup \{e\}$,
- $\mathcal{M}_1 = \{a_1, t\}$, $\mathcal{M}_2 = \{\texttt{yes}, \texttt{no}\}$, $\mathcal{M}_3 = \{b, c\}$,
- $i_{in} = 1$ is the input cell,
- $i_{out} = 0$ is the output region,
- Division rules: $r_{1,i} \equiv [a_i]_1 \to [t_i]_1[f_i]_1$, $1 \leq i \leq n$,
- The set $\mathcal{R}_{(0,1)}$ consists of the following rules:

$$\begin{aligned}
r_{2,i} &\equiv (d_i, \lambda/t_i, d'_i), 1 \leq i \leq n, \\
r_{3,i} &\equiv (d_i, \lambda/f_i, d''_i), 1 \leq i \leq n, \\
r_{4,i} &\equiv (d'_i, t_i/\lambda, d_{i+1}), 1 \leq i \leq n, \\
r_{5,i} &\equiv (d'_i, a_{i+1}/\lambda, d_{i+1}), 1 \leq i \leq n, \\
r_{6,i} &\equiv (d''_i, f_i/\lambda, d_{i+1}), 1 \leq i \leq n, \\
r_{7,i} &\equiv (d''_i, a_{i+1}/\lambda, d_{i+1}), 1 \leq i \leq n, \\
r_8 &\equiv (d_{n+1}, \lambda/a_{n+1}, s_1), \\
r_{9,i} &\equiv (s_i, \lambda/t_i, s_{i,1}), 1 \leq i \leq n, \\
r_{10,i} &\equiv (s_i, \lambda/f_i, \bar{s}_{i,1}), 1 \leq i \leq n, \\
r_{11,i,j} &\equiv (s_{i,j}, \lambda/g_{i,j}, s'_{i,j}), 1 \leq i \leq n, 1 \leq j \leq m, \\
r_{12,i,j} &\equiv (s_{i,j}, \lambda/g'_{i,j}, s''_{i,j}), 1 \leq i \leq n, 1 \leq j \leq m, \\
r_{13,i,j} &\equiv (s_{i,j}, \lambda/g''_{i,j}, s''_{i,j}), 1 \leq i \leq n, 1 \leq j \leq m, \\
r_{14,i,j} &\equiv (\bar{s}_{i,j}, \lambda/g'_{i,j}, \bar{s}'_{i,j}), 1 \leq i \leq n, 1 \leq j \leq m, \\
r_{15,i,j} &\equiv (\bar{s}_{i,j}, \lambda/g_{i,j}, \bar{s}''_{i,j}), 1 \leq i \leq n, 1 \leq j \leq m, \\
r_{16,i,j} &\equiv (\bar{s}_{i,j}, \lambda/g''_{i,j}, \bar{s}''_{i,j}), 1 \leq i \leq n, 1 \leq j \leq m, \\
r_{17,i,j} &\equiv (s'_{i,j}, c_j/\lambda, s_{i,j+1}), 1 \leq i \leq n, 1 \leq j \leq m-1, \\
r_{18,i,j} &\equiv (s''_{i,j}, e/\lambda, s_{i,j+1}), 1 \leq i \leq n, 1 \leq j \leq m-1, \\
r_{19,i,j} &\equiv (\bar{s}'_{i,j}, c_j/\lambda, \bar{s}_{i,j+1}), 1 \leq i \leq n, 1 \leq j \leq m-1, \\
r_{20,i,j} &\equiv (\bar{s}''_{i,j}, e/\lambda, \bar{s}_{i,j+1}), 1 \leq i \leq n, 1 \leq j \leq m-1, \\
r_{21,i} &\equiv (s'_{i,m}, c_m/\lambda, s_{i+1}), 1 \leq i \leq n, \\
r_{22,i} &\equiv (s''_{i,m}, e/\lambda, s_{i+1}), 1 \leq i \leq n, \\
r_{23,i} &\equiv (\bar{s}'_{i,m}, c_m/\lambda, s_{i+1}), 1 \leq i \leq n, \\
r_{24,i} &\equiv (\bar{s}''_{i,m}, e/\lambda, s_{i+1}), 1 \leq i \leq n, \\
r_{25,j} &\equiv (s_{n+j}, \lambda/c_j, s_{n+j+1}), 1 \leq j \leq m, \\
r_{26} &\equiv (s_{n+m+1}, \lambda/t, s_{n+m+2}).
\end{aligned}$$

- The set $\mathcal{R}_{(0,2)}$ consists of the following rules:

$$\begin{aligned}
r_{27} &\equiv (s, t/\lambda, s'), \\
r_{28} &\equiv (s', \lambda/\texttt{yes}, s_t), \\
r_{29} &\equiv (s, \lambda/b, s''), \\
r_{30} &\equiv (s'', \lambda/\texttt{no}, s_f).
\end{aligned}$$

- The set $\mathcal{R}_{(2,3)}$ consists of the following rules:

$$\begin{aligned}
r_{31,i} &\equiv (s_i, \lambda/c, s_{i+1}), 0 \leq i \leq 2nm + 4n + m + 1, \\
r_{32,i} &\equiv (s_i, c/\lambda, s_{i+1}), 0 \leq i \leq 2nm + 4n + m + 1, \\
r_{33} &\equiv (s_{2nm+4n+m+2}, \lambda/b, s).
\end{aligned}$$

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

- the size of set Γ: $3nm + 3n + m + 7 \in O(nm)$,
- the size of set K: $8nm + 7n + m + 9 \in O(nm)$,
- the number of rules: $14nm + 17n + m + 11 \in O(nm)$,

while the initial number of cells, the initial number of objects and the maximum length of a rule do not depend on n, m, hence they belong to $O(1)$. Thus, there exists a deterministic Turing machine that builds the system $\Pi(\langle n, m\rangle)$ in polynomial time with respect to n and m. Therefore, the family $\mathbf{\Pi} = \{\Pi(\langle n, m\rangle) \mid n, m \in \mathbb{N}\}$ defined above is polynomially uniform by Turing machines.

In what follows, we give the overview of a computation to show how an instance of the SAT problem is solved by the system defined above.

Let us consider a propositional formula $\varphi = C_1 \wedge \ldots \wedge C_m$, with $n \geq 1$ variables $\{x_1, \ldots, x_n\}$ and m clauses $C_1, \ldots, C_m$ such that $C_i = y_{i,1} \vee \ldots \vee y_{i,p_i}$, for some $p_i \geq 1$, and $y_{i,j} \in \{x_k, \neg x_k \mid 1 \leq k \leq n\}$, for each $1 \leq i \leq m, 1 \leq j \leq p_i$, where $\neg x_k$ is the negation of a propositional variable x_k.

We consider the polynomial encoding (cod, s) of instances from SAT in Π defined as follows: $s(\varphi) = \langle n, m\rangle$ and $cod(\varphi) = \{g_{i,j} \mid x_i \in C_j\} \cup \{g'_{i,j} \mid \neg x_i \in C_j\} \cup \{g''_{i,j} \mid x_i \notin C_j \text{ and } \neg x_i \notin C_j\}$ for each instance. Hence, the Boolean formula φ will be processed by the system $\Pi(s(\varphi)) + cod(\varphi)$.

Generation phase. In this phase, by dividing cell with label 1 for n times, all truth assignments for the variables associated with the Boolean formula $\varphi(x_1, \ldots, x_n)$ will be generated. In this way, after completing this phase, there exist 2^n cells with label 1 such that each of them encodes a different truth assignment of variables $\{x_1, \ldots, x_n\}$.

In the initial configuration of the system, we have objects $a_1, t, cod(\varphi)$ in cell 1, objects $\texttt{yes}, \texttt{no}$ in cell 2 and objects b, c in cell 3.

The generation phase consists of a loop with n iterations and one additional final step; each iteration of the loop takes three steps. Thus this phase takes $3n + 1$ steps. Specifically, this phase has two parallel processes. On the one hand, n loops are executed; on the other hand, there is an object c that moves between cell 2 and cell 3, which makes the state of channel $(2,3)$ evolve at each step by using rules $r_{31,i}$ and $r_{32,i}$.

At the first step of the i-th loop ($1 \leq i \leq n$), division rule $r_{1,i}$ is applied, one cell 1 is divided into two copies of

cell 1, producing objects t_i and f_i, which are placed in these two separate copies of cell 1, respectively. Simultaneously, by using rule $r_{31,i}$ or $r_{32,i}$, the subscript of the state object of channel $(2,3)$ increases by one.

At the second step of the i-th loop ($1 \leq i \leq n$), with the appearance of state d_i on channel $(0,1)$, object t_i (resp., f_i) is sent out of cell 1 by applying rule $r_{2,i}$ (resp., $r_{3,i}$), and the channel state is changed to d_i' (resp., d_i''). Simultaneously, the state object of channel $(2,3)$ evolves.

At the third step of the i-th loop ($1 \leq i \leq n$), with the presence of state d_i' (resp., d_i'') on channel $(0,1)$, rules $r_{4,i}$ and $r_{5,i}$ (resp., $r_{6,i}$ and $r_{7,i}$) are used in a flat maximally parallel manner. Thus, only one copy of object t_i and one copy of object a_{i+1} (resp., only one copy of object f_i and one copy of object a_{i+1}) are sent into one cell with label 1, and the channel state evolves to d_{i+1}. Note that at this step, if a cell with label 1 contained object t_i (resp., f_i) which was produced by the division rule at the first step of the i-th loop, then the set of rules $\{r_{4,i}, r_{5,i}\}$ (resp., $\{r_{6,i}, r_{7,i}\}$) is chosen, and object t_i (resp., f_i) is sent back to that cell with label 1 again because of the control of channel state d_i' (resp., d_i''). In addition, the state object of channel $(2,3)$ evolves.

After $3n$ steps, there are 2^n copies of cell with label 1, each of them containing an object a_{n+1}, as well as a different truth assignment of the variables $\{x_1, \ldots, x_n\}$, and the state of each channel $(0,1)$ is d_{n+1}. At step $3n+1$, rule r_8 is enabled and used, so that object a_{n+1} in each cell 1 is sent to the environment, and the state of each channel $(0,1)$ is changed to s_1; simultaneously, the subscript of the state object of channel $(2,3)$ increases by one.

Pre-checking phase. When the generation phase completes, the pre-checking phase starts. In each cell with label 1, the system looks for the clauses satisfied by the truth-assignment of variables $x_1, \ldots, x_n$. Specifically, state $s_{i,j}'$ on channel $(0,1)$ is obtained if both objects t_i and $g_{i,j}$ appear in that cell 1; this means that $x_i \in C_j$ and x_i is set to true, so clause C_j is satisfied. On the other hand, the occurrence of state $s_{i,j}''$ on channel $(0,1)$ means that x_i is set to true but $x_i \notin C_j$, hence we cannot infer that C_j is true. Similarly for $\overline{s}_{i,j}'$ (x_i is set to false and $\neg x_i \in C_j$, hence C_j is true) and $\overline{s}_{i,j}''$ (we cannot infer that C_j is true). When the system infers that clause C_j is true, then object c_j is brought in the corresponding cell 1, otherwise object e is brought in the corresponding cell 1, which is the idle object.

This phase begins at computation step $3n+2$ and consists of a loop with n iterations, where each iteration takes $2m+1$ steps. Hence the pre-checking phase takes $(2m+1)n$ steps.

At the first step of the i-th loop ($1 \leq i \leq n$), with the presence of state s_i on channel $(0,1)$, object t_i (resp., f_i) is sent to the environment by using rule $r_{9,i}$ (resp., $r_{10,i}$), the channel state is changed to $s_{i,1}$ (resp., $\bar{s}_{i,1}$). Simultaneously, the state object of channel $(2,3)$ evolves.

At the $2j$-th ($1 \leq j \leq m$) step of the i-th loop ($1 \leq i \leq n$), when the state $s_{i,j}$ appears on channel $(0,1)$, it will be changed to state $s_{i,j}'$ (resp., $s_{i,j}''$) if that cell 1 contains object $g_{i,j}$ (resp., $g_{i,j}'$ or $g_{i,j}''$) by using rule $r_{11,i,j}$ (resp., $r_{12,i,j}$ or $r_{13,i,j}$); similarly, the state of channel $(0,1)$ will be changed to $\bar{s}_{i,j}'$ (resp., $\bar{s}_{i,j}''$) from $\bar{s}_{i,j}$ if that cell 1 contains object $g_{i,j}'$ (resp., $g_{i,j}$ or $g_{i,j}''$) by using rule $r_{14,i,j}$ (resp., $r_{15,i,j}$ or $r_{16,i,j}$). In addition, the state object of channel $(2,3)$ evolves.

At the $2j+1$-th ($1 \leq j \leq m-1$) step of the i-th loop ($1 \leq i \leq n$), with the presence of state $s_{i,j}'$ (resp., $s_{i,j}''$) on channel $(0,1)$, rule $r_{17,i,j}$ (resp., $r_{18,i,j}$) is used in a flat maximally parallel way, one copy of object c_j (resp., e) is sent into that cell with label 1, and the channel state is changed to $s_{i,j+1}$; similarly, with the appearance of state $\bar{s}_{i,j}'$ (resp., $\bar{s}_{i,j}''$) on channel $(0,1)$, by using rule $r_{19,i,j}$ (resp., $r_{20,i,j}$), one copy of object c_j (resp., e) is sent into that cell with label 1, and the channel state is changed to $\bar{s}_{i,j+1}$. Simultaneously, the subscript of the state object of channel $(2,3)$ increases by one.

At the $2m+1$-th step of the i-th loop ($1 \leq i \leq n$), by applying rule $r_{21,i}$ (resp., $r_{22,i}$) in a flat maximally parallel way, one copy of object c_m (resp., e) will be sent into a cell 1 if the state of that channel $(0,1)$ is $s_{i,m}'$ (resp., $s_{i,m}''$), changing that channel state to s_{i+1}; similarly, by using rule $r_{23,i}$ (resp., $r_{24,i}$), one copy of object c_m (resp., e) will be sent into a cell 1 if the state of that channel $(0,1)$ is $\bar{s}_{i,m}'$ (resp., $\bar{s}_{i,m}''$), changing that channel state to s_{i+1}. Simultaneously, the state object of channel $(2,3)$ evolves.

Checking phase. In this phase, the system checks whether or not the formula is satisfied by some truth assignment.

When the pre-checking phase completes, each cell with label 1 contains some objects from the set $\{c_1, \ldots, c_m\}$, which correspond to the clauses satisfied by that assignment. If there is at least one cell with label 1 that contains all the objects $c_1, \ldots, c_m$, it means that the corresponding truth assignment in that cell satisfies all clauses, hence formula φ is satisfiable; if there is no cell with label 1 that contains all the objects $c_1, \ldots, c_m$, the formula φ is not satisfiable. The checking phase begins at step $2nm+4n+2$, and it takes m steps.

At step $2nm+4n+1+j$ ($1 \leq j \leq m$), the object c_j is checked in cell 1. With the presence of state s_{n+j} on channel $(0,1)$, one copy of object c_j is sent to the environment by using rules $r_{25,j}$ if that cell 1 encodes a truth assignment making clauses $C_1, \ldots, C_j$ true, the state of that channel $(0,1)$ is changed to s_{n+j+1}. Simultaneously, the state object of channel $(2,3)$ evolves.

Output phase. If the input formula φ is satisfiable, then there exists at least one cell with label 1 such that the state of channel $(0,1)$ is s_{n+m+1} after $2nm+4n+m+1$ steps. In this case, at step $2nm+4n+m+2$, by using rule r_{26}, object t is sent to the environment, and the state of channel $(0,1)$ is changed from s_{n+m+1} to s_{n+m+2}; by applying rule $r_{31,i}$ or $r_{32,i}$, the state of channel $(2,3)$ is changed to $s_{2nm+4n+m+2}$. At the next step, by using rule r_{27} in a flat maximally parallel way, only one copy of object t (if there exists more than one copy of object t in the environment) is sent into cell 2, the state of channel $(0,2)$ is changed from s to s'; under the influence of state $s_{2nm+4n+m+2}$ on channel $(2,3)$, object b in cell 3 is sent to cell 2 by using rule r_{33}. At step $2nm+4n+m+4$, object `yes` is sent to the environment by using rule r_{28} and the state of channel $(0,2)$ is changed to s_t. Thus, object `yes` is released into the environment at step $2nm+4n+m+4$

and the computation halts, thus the answer of the system is affirmative.

If the input formula φ is not satisfiable, then there is no cell with label 1 such that the state of channel $(0,1)$ is s_{n+m+1} after $2nm+4n+m+1$ steps. In this case, at step $2nm+4n+m+2$, the state of channel $(2,3)$ is changed to $s_{2nm+4n+m+2}$, and at the next step, object b is sent to cell 2 by using rule r_{33}. At step $2nm+4n+m+4$, under the influence of state s on channel $(0,2)$, object b is sent to the environment by using rule r_{29}, changing the state of channel $(0,2)$ to s''. With the presence of state s'' on channel $(0,2)$, rule r_{30} is enabled and applied, object no is released into the environment, and the computation halts at step $2nm+4n+m+5$, hence the answer of the system is *negative*.

In general, the P system $\Pi(\langle n,m\rangle)$ with input multiset $cod(\varphi)$ always halts and sends object yes or no to the environment at the last step, that is, at step $2nm+4n+m+4$, object yes is sent to the environment and the system halts, or object no is sent to the environment at step $2nm+4n+m+5$ and the system halts. Hence there exists a polynomial bound for the number of steps of the computation. Therefore, according to Definition 3, the SAT problem can be solved in polynomial time by the family $\mathbf{\Pi}$ of recognizer tissue P systems with channel states, using noncooperative symport rules and cell division, and working in a flat maximally parallel way. Hence the theorem holds.

Corollary 5.1: $\mathbf{NP} \cup \mathbf{co}-\mathbf{NP} \subseteq \mathbf{PMC}_{\mathbf{T}_s^f\mathbf{DC}(1)}$.

Proof: It suffices to make the following observations: the SAT problem is **NP**-complete, $\mathtt{SAT} \in \mathbf{PMC}_{\mathbf{T}_s^f\mathbf{DC}(1)}$ and the class $\mathbf{PMC}_{\mathbf{T}_s^f\mathbf{DC}(1)}$ is closed under polynomial time reductions, and is also closed under complement.

It is known that tissue P systems with cell division and communication rules of length at most 1 can only solve tractable problems [35], and the characterization of the standard computational class **P** of tractable problems by using the family of such P systems is based on the idea of dependency graph. Moreover, the concept of dependency graph associated with a P system can be extended easily to the class of recognizer tissue P systems with cell division working in a flat maximally parallel way. Hence we have the following theorem.

Theorem 5.2: $\mathbf{P} = \mathbf{PMC}_{\mathbf{T}^f\mathbf{DC}(1)}$.

Theorem 5.1 and Theorem 5.2 mean that there exists a frontier of tractability between efficiency and non-efficiency in terms of channel states (assuming $\mathbf{P} \neq \mathbf{NP}$).

VI. Conclusions and Further Works

In this work, the computational power of tissue P systems with channel states working in the flat maximally parallel way has been investigated. We have shown that tissue P systems with channel states and using antiport rules of length two working in the flat maximally parallel way are able to compute Parikh sets of finite languages, and such P systems with one cell and noncooperative symport rules can compute at least all Parikh sets of matrix languages, some Turing universality results have also been provided. We further solved the SAT problem by tissue P systems with channel states, cell division and noncooperative symport rules, working in the flat maximally parallel way; nevertheless, if we consider this kind of P systems without channel states, then a limit on the efficiency has been obtained. These results show that channel states are an essential parameter for the computational power.

P systems using rules in a minimally parallel way were investigated in [11], [36], where each membrane which can evolve in a given step should do it by using at least one rule. It is of interest to investigate the computational power of tissue P systems with channel states using rules in a minimally parallel way.

Time-free solutions to **NP**-complete problems by various timed P systems have been investigated, e.g., in [37]–[40]. It remains open whether we can construct tissue P systems with channel states and cell division to solve **NP**-complete problems in the context of time-freeness.

Small universal P systems have been studied widely [41]–[45]. It is interesting to see whether we can construct small universal tissue P systems with channel states working in a sequential way (on each channel, at most one rule can be used at each step) or in a flat maximally parallel way (at each step, on each channel, a maximal set of applicable rules which pass from a given state to a unique next state, is chosen and each rule in the set is applied once).

In [21], [46], it is shown that tissue P systems with only one object are computationally complete. It remains open whether the computational completeness result still holds for tissue P systems with channel states and one object working in the flat maximally parallel way. If the answer is positive, then what is the optimal length of communication rules used in such P systems?

Cell separation, which provides an efficient approach for obtaining an exponential workspace in polynomial time, has already been introduced into cell-like P systems with symport/antiport rules and tissue-like P systems [47]–[50]. In these P systems, cells do not have the duplication function; that is, when a cell is separated, the objects in the cell are divided and distributed in the newly generated cells instead of copying the objects and then placing one copy in each of the newly generated cells. It remains open whether **NP**-complete problems can be solved by tissue P systems with channel states and cell separation working in the flat maximally parallel way.

References

[1] G. Păun, "Computing with membranes," *J. Comput. Syst. Sci.*, vol. 61, no. 1, pp. 108–143, 2000.

[2] A. Alhazov and R. Freund, "Variants of small universal P systems with catalysts," *Fundam. Informat.*, vol. 138, nos. 1–2, pp. 227–250, 2015.

[3] F. Bernardini and M. Gheorghe, "Population P systems," *J. Univ. Comput. Sci.*, vol. 10, no. 5, pp. 509–539, 2004.

[4] M. Ionescu, G. Păun, and T. Yokomori, "Spiking neural P systems," *Fundam. Informat.*, vol. 71, nos. 2–3, pp. 279–308, 2006.

[5] A. Leporati, L. Manzoni, G. Mauri, A. E. Porreca, and C. Zandron, "Membrane division, oracles, and the counting hierarchy," *Fundam. Informat.*, vol. 138, nos. 1–2, pp. 97–111, 2015.

[6] C. Martín-Vide, J. Pazos, G. Păun, and A. Rodriguez-Paton, "Tissue P systems," *Theor. Comput. Sci.*, vol. 296, no. 2, pp. 295–326, 2003.

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

[8] B. Song, M. J. Pérez-Jiménez, and L. Pan, "Efficient solutions to hard computational problems by P systems with symport/antiport rules and membrane division," *BioSystems*, vol. 130, pp. 51–58, Apr. 2015.

[9] P. Sosík and M. Langer, "Small (purely) catalytic P systems simulating register machines," *Theor. Comput. Sci.*, vol. 623, pp. 65–74, Apr. 2016.

[10] D. Besozzi, P. Cazzaniga, D. Pescini, and G. Mauri, "Modelling metapopulations with stochastic membrane systems," *BioSystems*, vol. 91, no. 3, pp. 499–514, 2008.

[11] G. Ciobanu, L. Pan, G. Păun, and M. J. Pérez-Jiménez, "P systems with minimal parallelism," *Theor. Comput. Sci.*, vol. 378, no. 1, pp. 117–130, 2007.

[12] G. Păun and R. Păun, "Membrane computing and economics: Numerical P systems," *Fundam. Informat.*, vol. 73, nos. 1–2, pp. 213–227, 2006.

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

[14] G. Zhang, M. Gheorghe, L. Pan, and M. J. Pérez-Jiménez, "Evolutionary membrane computing: A comprehensive survey and new results," *Inf. Sci.*, vol. 279, pp. 528–551, Sep. 2014.

[15] G. Zhang, H. Rong, F. Neri, and M. J. Pérez-Jiménez, "An optimization spiking neural P system for approximately solving combinatorial optimization problems," *Int. J. Neural Syst.*, vol. 24, no. 5, pp. 1–16, 2014.

[16] G. Zhang, H. Rong, Z. Ou, M. J. Pérez-Jiménez, and M. Gheorghe, "Automatic design of deterministic and non-halting membrane systems by tuning syntactical ingredients," *IEEE Trans. NanoBiosci.*, vol. 13, no. 3, pp. 363–371, Sep. 2014.

[17] A. M. Colomer, A. Margalida, L. Valencia-Cabrera, and A. Palau, "Application of a computational model for complex fluvial ecosystems: The population dynamics of zebra mussel Dreissena polymorpha as a case study," *Ecol. Complex.*, vol. 20, pp. 116–126, Dec. 2014.

[18] G. Păun, G. Rozenberg, and A. Salomaa, Eds., *The Oxford Handbook of Membrane Computing*. New York: Oxford Univ. Press, 2010.

[19] A. Păun, G. Păun, and G. Rozenberg, "Computing by communication in networks of membranes," *Int. J. Found. Comput. Sci.*, vol. 13, no. 6, pp. 779–798, 2002.

[20] A. Alhazov, R. Freund, A. Leporati, M. Oswald, and C. Zandron, "(Tissue) P systems with unit rules and energy assigned to membranes," *Fundam. Informat.*, vol. 74, no. 4, pp. 391–408, 2006.

[21] R. Freund and M. Oswald, "Tissue P systems with symport/antiport rules of one symbol are computational complete," in *Proc. Eur. Sci. Found. PESC Exploratory Workshop Cellular Comput. (Complex. Aspects)*, 2005, pp. 178–187.

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

[23] R. Freund and S. Verlan, "A formal framework for static (tissue) P systems," in *Membrane Computing*, Berlin, Germany: Springer, 2007, pp. 271–284.

[24] G. Păun, M. J. Pérez-Jiménez, and A. Riscos-Núñez, "Tissue P systems with cell division," *Int. J. Comput. Commun.*, vol. 3, no. 3, pp. 295–303, 2008.

[25] D. Díaz-Pernil, M. A. Gutiérrez-Naranjo, M. J. Pérez-Jiménez, and A. Riscos-Núñez, "A uniform family of tissue P system with cell division solving 3-COL in a linear time," *Theor. Comput. Sci.*, vol. 404, no. 1, pp. 76–87, 2008.

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

[27] B. Song and L. Pan, "The computational power of tissue-like P systems with promoters," *Theor. Comput. Sci.*, vol. 641, pp. 43–52, Aug. 2016. [Online]. Available: http://dx.doi.org/10.1016/j.tcs.2016.05.022

[28] R. Freund, G. Păun, and M. J. Pérez-Jiménez, "Tissue P systems with channel states," *Theor. Comput. Sci.*, vol. 330, no. 1, pp. 101–116, 2005.

[29] R. Freund and S. Verlan, "(Tissue) P systems working in the k-restricted minimally or maximally parallel transition mode," *Natural Comput.*, vol. 10, no. 2, pp. 821–833, 2011.

[30] L. Pan, G. Păun, and B. Song, "Flat maximal parallelism in P systems with promoters," *Theor. Comput. Sci.*, vol. 623, pp. 83–91, Apr. 2016.

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

[32] J. Dassow and G. Păun, *Regulated Rewriting in Formal Language Theory*. Berlin, Germany: Springer, 1989.

[33] M. L. Minsky, *Computation: Finite and Infinite Machines*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1967.

[34] M. R. Garey and D. J. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. San Francisco, CA, USA: Freeman, 1979.

[35] R. Gutiérrez-Escudero, M. J. Pérez-Jiménez, and M. Rius-Font, "Characterizing tractability by tissue-like P systems," in *Membrane Computing*, Berlin, Germany: Springer, 2009, pp. 289–300.

[36] R. Freund, G. Păun, and M. J. Pérez-Jiménez, "Polarizationless P systems with active membranes working in the minimally parallel mode," in *Proc. 6th Brainstorming Week Membrane Comput.*, 2007, pp. 131–155.

[37] T. Song, L. F. Macías-Ramos, L. Pan, and M. J. Pérez-Jiménez, "Time-free solution to SAT problem using P systems with active membranes," *Theor. Comput. Sci.*, vol. 529, pp. 61–68, Apr. 2014.

[38] B. Song and L. Pan, "Computational efficiency and universality of timed P systems with active membranes," *Theor. Comput. Sci.*, vol. 567, pp. 74–86, Feb. 2015.

[39] B. Song, M. J. Pérez-Jiménez, and L. Pan, "Computational efficiency and universality of timed P systems with membrane creation," *Soft Comput.*, vol. 19, no. 11, pp. 3043–3053, 2015.

[40] B. Song, T. Song, and L. Pan, "Time-free solution to SAT problem by P systems with active membranes and standard cell division rules," *Natural Comput.*, vol. 14, no. 4, pp. 673–681, 2015.

[41] E. Csuhaj-Varjú, M. Margenstern, G. Vaszil, and S. Verlan, "On small universal antiport P systems," *Theor. Comput. Sci.*, vol. 372, no. 2, pp. 152–164, 2007.

[42] A. Păun and G. Păun, "Small universal spiking neural P systems," *BioSystems*, vol. 90, no. 1, pp. 48–60, 2007.

[43] T. Song, Y. Jiang, X. Shi, and X. Zeng, "Small universal spiking neural P systems with anti-spikes," *J. Comput. Theor. Nanosci.*, vol. 10, no. 4, pp. 999–1006, 2013.

[44] X. Zhang, X. Zeng, and L. Pan, "Smaller universal spiking neural P systems," *Fundam. Informat.*, vol. 87, no. 1, pp. 117–136, 2008.

[45] L. Pan and X. Zeng, "Small universal spiking neural P systems working in exhaustive mode," *IEEE Trans. NanoBiosci.*, vol. 10, no. 2, pp. 99–105, Jun. 2011.

[46] A. Alhazov, R. Freund, and M. Oswald, "Cell/symbol complexity of tissue P systems with symport/antiport rules," *Int. J. Found. Comput. Sci.*, vol. 17, no. 1, pp. 3–25, 2006.

[47] 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, Jul./Aug. 2016. [Online]. Available: http://dx.doi.org/10.1002/cplx.21691

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

[49] M. J. Pérez-Jiménez and P. Sosík, "An optimal frontier of the efficiency of tissue P systems with cell separation," *Fundam. Informat.*, vol. 138, nos. 1–2, pp. 45–60, 2015.

[50] X. Zhang, S. Wang, Y. Niu, and L. Pan, "Tissue P systems with cell separation: Attacking the partition problem," *Sci. China Inf. Sci.*, vol. 54, no. 2, pp. 293–304, 2011.