Fundamenta Informaticae ? (2017) 1001–1014
DOI 10.3233/FI-2012-0000
IOS Press

Tissue P Systems with Small Cell Volume

Alberto Leporati, Luca Manzoni, Giancarlo Mauri,

Antonio E. Porreca, Claudio Zandron

Dipartimento di Informatica, Sistemistica e Comunicazione

Università degli Studi di Milano-Bicocca

Viale Sarca 336/14, 20126 Milano, Italy

{*leporati*, *luca.manzoni*, *mauri*, *porreca*, *zandron*}*@disco.unimib.it*

Abstract. Traditionally, P systems allow their membranes or cells to grow exponentially (or even more) in *volume* with respect to the size of the multiset of objects they contain in the initial configuration. This behaviour is, in general, biologically unrealistic, since large cells tend to divide in order to maintain a suitably large surface-area-to-volume ratio. On the other hand, it is usually the *number* of cells that needs to grow exponentially with time by binary division in order to solve **NP**-complete problems in polynomial time. In this paper we investigate families of tissue P systems with cell division where each cell has a small volume (i.e., sub-polynomial with respect to the input size), assuming that each bit of information contained in the cell, including both those needed to represent the multiset of objects and the cell label, occupies a unit of volume. We show that even a constant volume bound allows us to reach computational universality for families of tissue P systems with cell division, if we employ an exponential-time uniformity condition on the families. Furthermore, we also show that a sub-polynomial volume does not suffice to solve **NP**-complete problems in polynomial time, unless the satisfiability problem for Boolean formulae can be solved in sub-exponential time, and that solving an **NP**-complete problem in polynomial time with logarithmic cell volume implies $\mathbf{P} = \mathbf{NP}$.

Keywords: Membrane computing, tissue P systems, computability, space complexity

1. Introduction

The usual approaches when analysing the computational resources employed by membrane systems are the same as for Turing machines, namely measuring the number of steps required to solve a problem as a function of the input size (time complexity) and the total space required by the systems (space

complexity). For membrane systems, the latter is measured either in terms of the number of objects and membranes in the largest configurations across all computations [1] or, when dealing with small space bounds (where a classification based on that definition turns out to be too coarse), in terms of the number of bits required to store the largest configurations [7].

A defining characteristic of P systems is the ability to solve classically intractable problems in polynomial time by *trading space for time*: in P systems with active membranes, an exponential number of membranes is generated by membrane division (or separation), and then all membranes compute in parallel, for instance exploring the solution space of an $\mathbf{NP}$-complete problem [13]. It is the exponential number of cells that allows this exponential speed-up with respect to all known traditional algorithms, and *not* the number of objects that they contain, which can be polynomial if membrane division is allowed [13]; furthermore, it is useless to allow an exponential number of objects per membrane if division is not allowed, since in this case, by the "Milano Theorem" [15], no problem outside $\mathbf{P}$ can be solved in polynomial time. In tissue P systems, cell division or cell separation rules allow the implementation of similar algorithms, with exponentially many cells computing in parallel [14, 12].

Let us define the volume of a cell as the number of bits of information it encodes, that is, the logarithm of the number of possible cell labels, plus the logarithm of the size of the alphabet multiplied by the number of objects contained in the cell; this is the definition of configuration size for P systems previously proposed by the authors [7, Definition 4.1], but limited to a single cell. This definition assumes that each bit of information requires some corresponding amount of physical space. In this paper we investigate what happens when we do not restrict the total amount of cells (obtained by division or separation) but we do restrict the cell volume: what can the resulting tissue P systems compute, either in polynomial or in unbounded time?

In Section 3 we show that semi-uniform families of tissue P systems, where each P system is constructed from an input string, are computationally universal even with constant cell volume, if we allow an exponential-time uniformity condition. In Section 4 we prove (under standard complexity theory assumptions) that sub-polynomial cell volume does not allow the solution of $\mathbf{NP}$-hard problems in polynomial time by tissue P systems: any sub-polynomial cell volume solution would imply that 3-SAT is solvable in sub-exponential time (thus invalidating the exponential time hypothesis [6]), and a logarithmic cell volume solution would even imply $\mathbf{P} = \mathbf{NP}$.

2. Basic Notions

We begin by recalling the definition of tissue P systems, in particular the variants with division and separation rules from [9, 14, 12, 8].

Definition 2.1. A *tissue P system* is a structure $\Pi = (\Gamma, E, w_1, \ldots, w_d, R, i_0)$, where:

- Γ is an alphabet, i.e., a finite non-empty set of symbols, usually called *objects*;
- $E \subseteq \Gamma$ is the alphabet of objects initially located in the external environment, in *infinitely* many copies;
- $d \geq 1$ is the *degree* of the system, i.e., the initial number of cells; each cell is identified by a unique *label* in the range $1, \ldots, d$, and 0 is the label of the external environment;

- $w_1, \ldots, w_d$ are *finite* multisets over Γ, describing the initial contents of the d cells;
- R is a finite set of rules;
- $i_0 \in \{0, \ldots, d\}$ is the output region.

A *configuration* $\mathcal{C}$ of a tissue P system consists of a multiset over $\Gamma - E$ describing the objects appearing with finite multiplicity in the environment, and a multiset of pairs (h, w), where h is a cell label and w a finite multiset over Γ, describing the cells. A configuration may contain, as a result of cell division or separation, multiple pairs $(h, w_1), (h, w_2)$ with the same membrane label h and possibly distinct multisets.

The rules of R are of the following types:

(a) *Communication rules*, denoted in this paper by $[u]_h \leftrightarrow [v]_k$ and in the literature by $(h, u/v, k)$, where h and k are distinct labels (including the environment), and u and v are multisets over Γ (at least one of them nonempty): these rules are applicable in a configuration $\mathcal{C}$ if there exists a region of $\mathcal{C}$ with label h containing u as a submultiset and a region k of $\mathcal{C}$ containing v as a submultiset; the effect of the rule is to exchange u and v between the two regions. If $h = 0$ (resp., $k = 0$) then the rule is denoted by $u \leftrightarrow [v]_k$ (resp., $[u]_h \leftrightarrow v$), and in that case we require multiset u (resp., v) to contain at least an object from $\Gamma - E$, i.e., an object with finite multiplicity, if v (resp., u) is empty[1]. In this paper we consider two rules $[u]_h \leftrightarrow [v]_k$ and $[v]_k \leftrightarrow [u]_h$ to be the same, since they would exhibit the same behaviour in all circumstances.

(b) *Division rules*, of the form $[a]_h \rightarrow [b]_h\ [c]_h$, where $h \neq 0, h \neq i_0$ is a cell label and $a, b, c \in \Gamma$: these rules can be applied in a configuration $\mathcal{C}$ to a cell with label h containing at least one copy of a; the effect of the rule is to divide the cell into two cells, both with label h; the object a is replaced in the two cells by b and c, respectively, while the rest of the original multiset contained in h is replicated in both cells.

(c) *Separation rules*, of the form $[a]_h \rightarrow [\Gamma_1]_h\ [\Gamma_2]_h$, where $h \neq 0, h \neq i_0$ is a cell label, $a \in \Gamma$, and $\{\Gamma_1, \Gamma_2\}$ is a partition of Γ: these rules can be applied in a configuration $\mathcal{C}$ to a cell with label h containing at least one copy of a; the effect of the rule is to separate the cell into two cells, both with label h; the object a is consumed, while the objects from Γ_1 in the original multiset contained in h are placed inside one of the cells, and those from Γ_2 in the other. All separation rules in R must share the same partition $\{\Gamma_1, \Gamma_2\}$ of Γ.

A *tissue P system with cell division* only uses communication and division rules, while a *tissue P system with cell separation* only uses communication and separation rules.

A *computation step* changes the current configuration according to the following set of principles:

- Each object can be subject to at most one rule, and each cell can be subject to *any number* of communication rules (consistent with the remaining principles) or, *alternatively*, a *single* division or separation rule.
- The application of rules is *maximally parallel*: each region is subject to a maximal multiset of rules (i.e., no further rule can be applied).

[1] Since communication rules are applied in a maximally parallel way, this restriction avoids the situation where infinitely many objects from the environment simultaneously enter a cell.

- When several conflicting rules can be applied at the same time, a nondeterministic choice is performed; this implies that, in general, multiple possible configurations can be reached after a computation step.

A *halting computation* $\vec{\mathcal{C}} = (\mathcal{C}_0, \ldots, \mathcal{C}_k)$ of the tissue P system Π is a finite sequence of configurations, where $\mathcal{C}_0$ is the initial configuration, every $\mathcal{C}_{i+1}$ is reachable from $\mathcal{C}_i$ via a single computation step, and no rules are applicable in $\mathcal{C}_k$.

Tissue P systems can be used as language *recognisers* by employing two distinguished objects yes and no: we assume that all computations are halting, and that one of the objects yes or no (but not both) is released into the environment, and only in the last computation step, in order to signal acceptance or rejection, respectively. If all computations starting from the same initial configuration are accepting, or all are rejecting, the tissue P system is said to be *confluent*. A confluent P system is said to *accept* if its computations are accepting, and to *reject* otherwise. Confluent tissue P systems may be locally nondeterministic (i.e., a configuration can have multiple successors), as long as all computations agree on the final result. As a special case, tissue P systems whose initial configuration generates a single computation are called *deterministic*.

In order to solve decision problems (i.e., decide languages over an alphabet Σ), we use *families* of recogniser tissue P systems $\mathbf{\Pi} = \{\Pi_x : x \in \Sigma^\star\}$. Each input x is associated with a tissue P system Π_x that decides the membership of x in the language $L \subseteq \Sigma^\star$ by accepting or rejecting. The mapping $x \mapsto \Pi_x$ must be efficiently computable for inputs of any length, as discussed in detail in [11].

Definition 2.2. A family of tissue P systems $\mathbf{\Pi} = \{\Pi_x : x \in \Sigma^\star\}$ is said to be *(polynomial-time) uniform* if the mapping $x \mapsto \Pi_x$ can be computed by two polynomial-time deterministic Turing machines E and F as follows:

- $F(1^n) = \Pi_n$, where n is the length of the input x and Π_n is a common tissue P system for all inputs of length n, with a distinguished input cell.
- $E(x) = w_x$, where w_x is a multiset encoding the specific input x.
- Finally, Π_x is simply Π_n with w_x added to its input cell.

On the other hand, the family $\mathbf{\Pi}$ is said to be *(polynomial-time) semi-uniform* if there exists a single deterministic polynomial-time Turing machine H such that $H(x) = \Pi_x$ for each $x \in \Sigma^\star$.

Any explicit encoding of Π_x is allowed as output of the construction, as long as the number of cells and objects represented by it does not exceed the length of the whole description, and the rules are listed one by one. This is also called a *permissible encoding* [11].

Let $\mathcal{D}$ be a class of tissue P systems (e.g., tissue P systems with cell division, or with cell separation) The class of problems solved in polynomial time by uniform (resp., semi-uniform) families of confluent recogniser tissue P systems of that class is denoted by $\mathbf{PMC}_{\mathcal{D}}$ (resp., $\mathbf{PMC}^\star_{\mathcal{D}}$). The inclusion $\mathbf{PMC}_{\mathcal{D}} \subseteq \mathbf{PMC}^\star_{\mathcal{D}}$ holds by definition, since uniformity is a special case of semi-uniformity.

As auxiliary tools, in this paper we employ *register machines* [10] without tape, that is, with an input register containing a natural number whose membership in a set is to be verified. In particular, we employ the variant with deterministic increment instruction $i\colon \mathrm{inc}(r), j$ and branching decrement instruction $i\colon \mathrm{dec}(r), j_1, j_2$. Here i represents the label of the current instruction; the increment instruction

adds one to the value of register r and jumps to instruction j; the decrement instruction subtracts one to the value of register r and jumps to j_1 if $r > 0$, and jumps to j_2 without changing the register if $r = 0$. Register machines will be simulated by place/transition Petri nets, where places may contain an (a priori) unbounded number of tokens [3], using a construction described by Frisco [4].

3. Universality with Constant Cell Volume

We define the *volume* of a cell as the amount of bits of information it encodes, both in terms of the objects it contains and its label.

Definition 3.1. The *volume* of a cell of a tissue P system (with alphabet Γ and set of labels Λ) containing multiset $w \in \Gamma^\star$ is given by $\lceil \log_2 |\Lambda| \rceil + |w| \times \lceil \log_2 |\Gamma| \rceil$. We say that a tissue P system has *limited cell volume* v if none of its cells ever exceeds volume v in any computation starting from its initial configuration. Finally, we say that a family of tissue P systems $\mathbf{\Pi} = \{\Pi_x : x \in \Sigma^\star\}$ has cell volume $v(n)$ if all Π_x with $|x| = n$ have limited cell volume $v(n)$.

Notice that the volume bound v (or $v(n)$) in this definition does not represent a limit imposed on the volume of cells, i.e., blocking the entering of further objects in a cell, but a *measured upper bound* across all computations. As a consequence, there is no change in the semantics of the application of rules from the standard definition.

We focus our analysis on tissue P systems with small cell volume, that is, sub-polynomial cell volume (with a particular interest for constant and logarithmic cell volume). We begin by proving that even the restriction to constant volume does not affect the universality of tissue P systems with cell division, *if we allow multiple cells to share the same label in the initial configuration*. Furthermore, all communication rules required have *length* 2, meaning that only two objects per rule are involved. This proof exploits the fact that register machines can be simulated by place/transition Petri nets operating in the maximally parallel way, as shown by Frisco [4] based on the zero-test proposed by Burkhard [2].

Theorem 3.2. For each register machine R there exists an exponential-time semi-uniform family of tissue P systems with constant cell volume simulating R using only communication rules of length 2 and division rules, assuming that multiple cells with the same label are allowed in the initial configuration.

Proof:
Each register machine R can be simulated by a Petri net working under the maximally parallel execution semantics [2, 4]. Each register is implemented as a place of the net, and its value is encoded as the number of tokens contained therein. Furthermore, each value i of the program counter has an associated place p_i, which contains a (single) token if and only if the machine is executing that particular instruction.

While it is straightforward to simulate an increment instruction $i\colon \mathrm{inc}(r), j$, a decrement instruction $i\colon \mathrm{dec}(r), j_1, j_2$ requires a more sophisticated net, which exploits the maximal parallelism to synchronise two tokens, in order to perform the zero test (Figure 1). These two subnets are constructed from the three building blocks S (sequence), F (fork), and J (join), also depicted in Figure 1; two building blocks are connected by identifying their input and output places suitably. All places contain at most one token, except those labelled by ∞: these are the places representing the registers, and contain an unbounded number of tokens (encoding the value of the register in unary). The fact that the other places

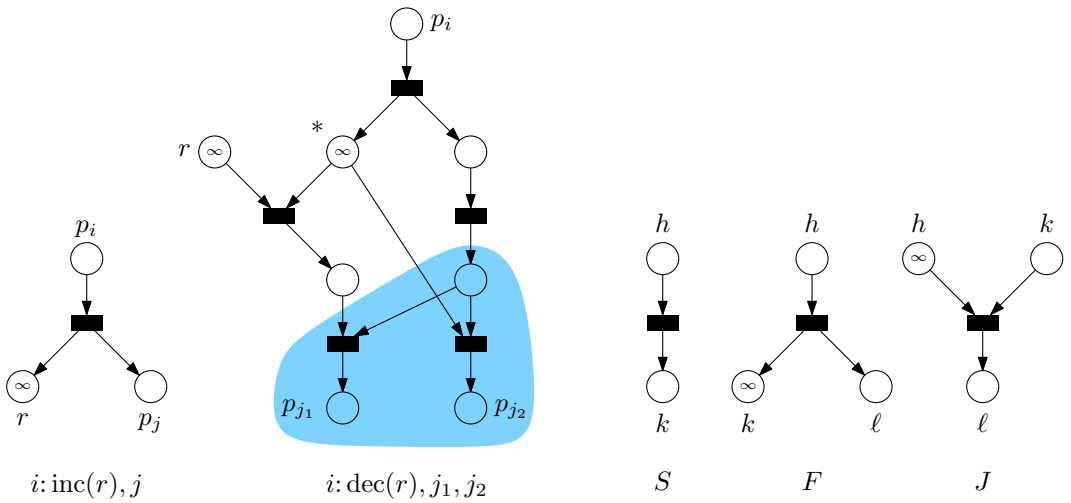

Figure 1. Petri nets implementing the register machine instructions $i\colon \mathrm{inc}(r), j$ and $i\colon \mathrm{dec}(r), j_1, j_2$, and the building blocks S (sequence), F (fork), and J (join) required to construct them. Each place contains zero or one token, except those labelled by ∞, which can contain an arbitrary number of tokens; the place marked by $*$ is labelled by ∞ in order to avoid introducing an extra building block, but will always contain at most one token [2, 4] and thus its output transition will work as a J block.

never contain more than one token is a consequence of the initial marking (i.e., the placement of tokens) [2, 4]; in other words, these places are 1-*bounded* [3]. Notice that, although the highlighted portion of the decrement subnet in Figure 1 presents a conflict between two transitions, the actual placement of tokens enables at most one of them at any given moment; the two transitions thus behave as mutually exclusive join blocks.

It is thus possible to prove the universality of the tissue P systems considered here by showing how each building block can be simulated. We always assume that, when the input places of the transition in a building block contain a token, then the output place not labelled by ∞ is empty; this does not imply a loss of generality, since the initial marking ensures that this is always the case, assuming that no instruction of the simulated register machine immediately jumps back to itself [2, 4].

Each place h of the Petri net containing at most one token is simulated by a single cell with label h in the tissue P system; this cell contains the "dummy" object $\circ$ when the place is empty, and a single object, also called h, when it contains a token. The places h of unbounded content in the Petri net are simulated by $m + 1 + f$ cells with label h, where m is the number of tokens contained initially in place h and f is the number of times that transitions entering h have fired until now. When place h contains n tokens, n of these $m + 1 + f$ cells contain the object h, and the remaining ones (which are always at least one) contain the dummy object $\circ$.

The sequence block S is thus simulated by two cells with labels h and k associated with the communication rules

$$[h]_h \leftrightarrow [\circ]_k \qquad [h]_k \leftrightarrow h' \qquad [h']_k \leftrightarrow k$$

Starting from the (sub-)configuration $[h]_h\ [\circ]_k$, which encodes a token in place h and the empty place k,

these rules produce the computation

$$[h]_h\,[\circ]_k \quad\longrightarrow\quad [\circ]_h\,[h]_k \quad\stackrel{*}{\longrightarrow}\quad [\circ]_h\,[h']_k \quad\longrightarrow\quad [\circ]_h\,[k]_k$$

where the last configuration encodes a token in place k and empty place h, as required. The computation step marked by $*$, apparently redundant, is performed in order to ensure that the simulation of block S requires the same number of steps as blocks F and J.

The fork block F is simulated by two cells labelled by h and ℓ, and any positive number of cells labelled by k, with the rules

$$[h]_h \leftrightarrow [\circ]_k \qquad [h]_k \rightarrow [k']_k\,[\ell]_k \qquad [\ell]_k \leftrightarrow [\circ]_\ell \qquad [k']_k \leftrightarrow k$$

Starting from the (sub-)configuration $[h]_h\,[\circ]_k\,[\circ]_\ell$, encoding one token in h, no tokens in ℓ, and any number of tokens in k (not shown here; recall that a place k with unbounded capacity is associated with a number of cells labelled by k and there always exists one of them containing $\circ$), these rules produce the computation steps

$$[h]_h\,[\circ]_k\,[\circ]_\ell \quad\longrightarrow\quad [\circ]_h\,[h]_k\,[\circ]_\ell \quad\longrightarrow\quad [\circ]_h\,[k']_k\,[\ell]_k\,[\circ]_\ell \quad\longrightarrow\quad [\circ]_h\,[k]_k\,[\circ]_k\,[\ell]_\ell$$

which represent the movement of a token from place h to place ℓ, and the increment of the number of tokens in place k. Notice that these rules preserve the invariant that at least one cell with label k contains $\circ$.

Finally, the join block J is simulated by any positive number of cells with label h and two cells labelled by k and ℓ, with the associated rules

$$[h]_h \leftrightarrow [k]_k \qquad [k]_h \leftrightarrow \circ \qquad [h]_k \leftrightarrow [\circ]_\ell \qquad [h]_\ell \leftrightarrow \ell$$

Starting from the (sub-)configuration $[h]_h\,[k]_k\,[\circ]_\ell$, encoding at least one token in place h, one token in k and zero tokens in ℓ, these rules produce the computation steps

$$[h]_h\,[k]_k\,[\circ]_\ell \quad\longrightarrow\quad [k]_h\,[h]_k\,[\circ]_\ell \quad\longrightarrow\quad [\circ]_h\,[\circ]_k\,[h]_\ell \quad\longrightarrow\quad [\circ]_h\,[\circ]_k\,[\ell]_\ell$$

where the tokens in place h decrease by one, while the token in k is moved to place ℓ. Notice that this block does not execute at all if place h is empty, that is, if no cell labelled by h contains the object h, since rule $[h]_h \leftrightarrow [k]_k$ (and, consequently, the remaining three rules) cannot be applied.

Each register machine is thus simulated by a constant (i.e., independent of the value of the input register) number of building blocks, each corresponding to a constant number of cell labels and object types; furthermore, each cell always contains exactly one object. The input x of the register machine R (corresponding to the number of tokens in the input place r_1 of the Petri net) is encoded as the number of cells with label r_1 containing the object r_1. The mapping $x \mapsto \Pi_x$, where Π_x is the tissue P system simulating R on input x, can thus be computed in exponential time by a deterministic Turing machine, expanding the binary encoding x in unary as the aforementioned number of cells r_1, which is the only part of the P system depending on x.

Since the number of labels and the size of the alphabet are constant, and each cell always contains exactly one object, the family of tissue P systems has cell volume $v(n) \in O(1)$, where the constant depends only on the register machine R being simulated. The rules used in this simulation are all communication rules of length 2 and division rules. □

Notice, however, that in Theorem 3.2 we are employing a slightly more general definition of "initial configuration" for tissue P systems, where multiple cells can share the same label. Without this stipulation, there can only exist a constant number of distinct configurations consisting of cells of constant volume[2], and thus essentially all work needed in order to establish if an input x is a positive instance or a negative one must be carried out by the Turing machine providing the semi-uniformity condition.

Furthermore, Theorem 3.2 cannot be proved for *uniform* families, since these require the input to be placed inside a single cell (and, once again, a cell of constant volume can only contain a constant number of possible inputs). By allowing the input to be distributed among several cells, the result can be extended to uniform families.

On the other hand, Theorem 3.2 is optimal with respect to the length of communication rules, since tissue P systems using only rules of length 1 (i.e., involving only one object), even together with cell division or separation, can be simulated in polynomial time using dependency graphs, and thus are not universal [5, 12].

4. Simulation of Tissue P Systems with Small Cell Volume

We now turn our attention to the amount of resources needed in order to simulate confluent recogniser tissue P systems with limited cell volume by means of deterministic Turing machines.

We begin by giving an upper bound to the number of cells in a tissue P system, which is proved by assuming that each cell divides at each time step:

Lemma 4.1. A tissue P system with c cells (having distinct labels) in its initial configuration has at most $c \times 2^t$ cells after t computation steps, and more specifically at most 2^t per label. □

This allows us to give an upper bound to the space needed in order to store the configuration of a tissue P system with limited cell volume in a tabular form.

Lemma 4.2. The configuration $\mathcal{C}$ of a tissue P system with cell volume v after t computation steps can be stored in $O(2^v \times t)$ bits.

Proof:
It suffices to construct a table indexed by all possible distinct cell configurations of volume bounded by v, which are at most 2^v in number. For each of them, the table stores the multiplicity of that cell configuration in the current global configuration of the tissue P system, which is at most 2^t (Lemma 4.1), in binary notation. □

We can now give an efficient algorithm for simulating a computation step of a tissue P system with limited cell volume, starting from a configuration given in tabular form.

Lemma 4.3. Given a configuration $\mathcal{C}$ in tabular form, as in the proof of Lemma 4.2, of a tissue P system Π with cell volume v and m rules at time t, it is possible to construct a successor configuration $\mathcal{C}'$ in tabular form in polynomial time with respect to 2^v, t and m.

[2]Specifically, there are exponentially many labels of volume $v(n)$, and each of them can either appear or not appear in the initial configuration; the total number of initial configurations is thus at most doubly exponential in (the constant) $v(n)$.

Proof:
In order to compute $\mathcal{C}'$ we temporarily enlarge the alphabet Γ of Π with the new objects $\Gamma' = \{a' : a \in \Gamma\}$ and construct a new table $\mathcal{D}$, initialised in such a way that $\mathcal{D}[c] = \mathcal{C}[c]$ for all cell configurations c that do not include objects in Γ' (i.e., the number of cells having configuration c is identical in both tables), and $\mathcal{D}[c] = 0$ otherwise. Notice that, by Lemma 4.2, table $\mathcal{D}$ can be stored in $O(2^{2v} \times t)$ bits. The new, primed objects in Γ' temporarily replace the objects in Γ when they cannot be subject to any (further) rule in the simulated computation step; namely, the objects will be primed when they have already been subject to a communication rule, or when they belong to a cell that has already divided or separated.

Table $\mathcal{D}$ is updated by first iterating across the set of rules of Π and then across all configurations (or pairs of configurations) of cells possibly involved in the current rule. In order to construct a successor configuration, we can give priority to division and separation rules over communication rules:

1 **for** each division rule $r = [a]_h \to [b_1]_h\ [b_2]_h$ **do**
2 **for** each cell configuration $c = [w]_h$ **do**
3 **if** rule r is applicable to configuration c **then**
4 apply rule r to configuration c
5 **for** each separation rule $r = [a]_h \to [\Gamma_1]_h\ [\Gamma_2]_h$ **do**
6 **for** each cell configuration $c = [w]_h$ **do**
7 **if** rule r is applicable to configuration c **then**
8 apply rule r to configuration c
9 **for** each communication rule $r = [u]_h \leftrightarrow [v]_k$ **do**
10 **for** each cell configuration $c_1 = [w_1]_h$ **do**
11 **for** each cell configuration $c_2 = [w_2]_k$ **do**
12 **if** rule r is applicable to configurations c_1 and c_2 **then**
13 apply rule r to configurations c_1 and c_2

The applicability of rules in lines 12, 3, and 7 requires checking that the multisets u and v (for communication rules) or the object a (for division and separation rules) appear in one of the cell configurations involved, that these cell configurations have the correct label, and (for division and separation rules only) that no primed objects appear in cell configuration c. Multiset inclusion can be checked in polynomial time with respect to v, which is an upper bound on the size in bits of the multisets involved.

Thus, in order to prove the statement of the lemma we only need to show that each kind of rule can be simulated in polynomial time (lines 13, 4, and 8). A division rule $r = [a]_h \to [b_1]_h\ [b_2]_h$ is simulated as follows:

4.1 $tmp \coloneqq \mathcal{D}[c]$
4.2 $\mathcal{D}[c] \coloneqq 0$
4.3 $c'_1 \coloneqq [(w - a + b_1)']_h$
4.4 $\mathcal{D}[c'_1] \coloneqq \mathcal{D}[c'_1] + tmp$
4.5 $c'_2 \coloneqq [(w - a + b_2)']_h$
4.6 $\mathcal{D}[c'_2] \coloneqq \mathcal{D}[c'_2] + tmp$

Rule r is applied to *all* cells having configuration c; as a result, the number of cells having that configuration is set to zero (line 4.2) after storing it in a temporary variable (line 4.1). Each cell having

configuration c produces two cells with configuration c'_1 and c'_2, respectively, computed in lines 4.3 and 4.5, by removing an instance of a, replacing it with an instance of b_1 (resp., b_2) and priming *all* the objects in order to signal that no further rule can be applied to cells having configuration c'_1 or c'_2. The number of cells having configuration c'_1 and c'_2 is updated according to the original number of cells with configuration c (lines 4.4 and 4.6). The time required to simulate the division rule r is polynomial with respect to t (which bounds the number of instances of each cell configuration) and $O(2^v)$ (the size of table $\mathcal{D}$, assuming sequential access to its entries on a Turing machine).

The simulation of a separation rule $[a]_h \to [\Gamma_1]_h\ [\Gamma_2]_h$ is identical to the simulation of a division rule, except that the configurations of the resulting cells are computed in a slightly different way:

8.1 $tmp \coloneqq \mathcal{D}[c]$
8.2 $\mathcal{D}[c] \coloneqq 0$
8.3 $c'_1 \coloneqq [((w-a) \cap \Gamma_1)']_h$
8.4 $\mathcal{D}[c'_1] \coloneqq \mathcal{D}[c'_1] + tmp$
8.5 $c'_2 \coloneqq [((w-a) \cap \Gamma_2)']_h$
8.6 $\mathcal{D}[c'_2] \coloneqq \mathcal{D}[c'_2] + tmp$

Notice that the simulation of division and separation rules primes all objects contained in the resulting cells. Hence, no communication rule will ever be applicable (line 12) to a cell where a division or a separation rule has already been applied.

For two multisets w, u over Γ, let us denote by $w(u)$ the number of disjoint occurrences of u in w. Communication rules $r = [u]_h \leftrightarrow [v]_k$ can be simulated as follows:

13.1 $k \coloneqq \min\big(\mathcal{D}[c_1] \times w_1(u), \mathcal{D}[c_2] \times w_2(v)\big)$
13.2 $c'_1 \coloneqq [w_1 - w_1(u) \times u + w_1(u) \times v']_h$
13.3 $c'_2 \coloneqq [w_2 - w_2(v) \times v + w_2(v) \times u']_h$
13.4 $m_1 \coloneqq \lfloor k / w_1(u) \rfloor$
13.5 $\mathcal{D}[c_1] \coloneqq \mathcal{D}[c_1] - m_1$
13.6 $\mathcal{D}[c'_1] \coloneqq \mathcal{D}[c'_1] + m_1$
13.7 $k_1 \coloneqq k - m_1 \times w_1(u)$
13.8 **if** $k_1 > 0$ **then**
13.9 $\quad c''_1 \coloneqq [w_1 - k_1 \times u + k_1 \times v']_h$
13.10 $\quad \mathcal{D}[c_1] \coloneqq \mathcal{D}[c_1] - 1$
13.11 $\quad \mathcal{D}[c''_1] \coloneqq \mathcal{D}[c''_1] + 1$
13.12 $m_2 \coloneqq \lfloor k / w_2(v) \rfloor$
13.13 $\mathcal{D}[c_2] \coloneqq \mathcal{D}[c_2] - m_2$
13.14 $\mathcal{D}[c'_2] \coloneqq \mathcal{D}[c'_2] + m_2$
13.15 $k_2 \coloneqq k - m_2 \times w_2(v)$
13.16 **if** $k_2 > 0$ **then**
13.17 $\quad c''_2 \coloneqq [w_2 - k_2 \times v + k_2 \times u']_h$
13.18 $\quad \mathcal{D}[c_2] \coloneqq \mathcal{D}[c_2] - 1$
13.19 $\quad \mathcal{D}[c''_2] \coloneqq \mathcal{D}[c''_2] + 1$

The number k computed in line 13.1 is the number of pairs of multisets (u, v) appearing in cell configurations c_1 and c_2, respectively, when the P system is in configuration $\mathcal{C}$ (there might remain unpaired instances of u or v, but not both). This is, hence, the maximum number of times that rule $r = [u]_h \leftrightarrow [v]_k$ can be applied in configuration $\mathcal{C}$ to pairs of cells of the form c_1 and c_2 (note that rule r might also be applicable to other pairs of cell configurations containing u and v). In the configuration $\mathcal{C}'$ we are building, this is indeed the actual number of applications.

If a cell with configuration c_1 applies rule r as many times as possible, that is, exchanging *each* copy of multiset u it contains with a copy of v, the result is cell configuration c_1', as computed in line 13.2 by removing $w_1(u)$ copies of u from the original multiset w_1 and adding an equal number of copies of v', that is, the multiset v where each object has been primed. The analogous updated configuration c_2' is computed in line 13.3.

The number of cells having initial configuration c_1 where rule r can be applied to *all* instances of u (thus leading to cell configuration c_1') is computed in line 13.4. The temporary configuration $\mathcal{D}$ is updated accordingly in lines 13.5 and 13.6.

There might remain cells with configuration c_1 where rule r is applied, but not involving *all* instances of u. For example, consider the configuration $\mathcal{C} = [u\ u]_h\ [u\ u]_h\ [v\ v\ v]_k$; after executing line 13.6 we have $\mathcal{D} = [v'\ v']_h\ [u\ u]_h\ [v\ v\ v]_k$, and we still need to replace an instance of u in the second cell (but not the other instance) with v', obtaining $\mathcal{D} = [v'\ v']_h\ [v'\ u]_h\ [v\ v\ v]_k$. The remaining number k_1 of instances of u to be exchanged by rule r is computed in line 13.7. This number, if nonzero, is strictly less than $w_1(u)$ (otherwise there would exist at least another cell with configuration c_1 where r can be applied to every instance of u). We can thus choose all remaining instances of u *from the same cell* with configuration c_1, and obtain the updated configuration c_1'' computed in line 13.9. One cell with configuration c_1'' then replaces one with configuration c_1 in the temporary configuration $\mathcal{D}$ (lines 13.10 and 13.11). The configurations of the cells involved in the right-hand side of the rule $r = [u]_h \leftrightarrow [v]_k$ are updated symmetrically in lines 13.12–13.19.

Notice that one of the two multisets (say, u) appearing in a communication rule $r = [u]_h \leftrightarrow [v]_k$ can be empty. In that case, we have $w_1(u) = \infty$. The above algorithm works even in this case, with the stipulations that $n \times \infty = \infty$ for $n > 0$ (line 13.1), but $0 \times \infty = 0$ (line 13.7), that $\infty \times v = \epsilon$ for each multiset v (line 13.2), and that $n/\infty = 0$ for $n > 0$ (line 13.4). The (admittedly artificial) assumption that $\infty \times v = \epsilon$ is specifically made in order to simplify the algorithm, since this way c_1' is indeed equal to c_1 (line 13.2), but $m_1 = 0$ (line 13.4) and thus $\mathcal{D}[c_1]$ is left untouched (lines 13.5 and 13.6).

The simulation of communication rules (lines 13.1–13.19) can be executed in polynomial time with respect to t and $O(2^v)$, like the simulation of division and separation rules.

After having iterated across all rules of Π and applied them as described above, no further rule can be applied to any unprimed object of the temporary configuration $\mathcal{D}$. The desired configuration $\mathcal{C}'$ can be obtained from $\mathcal{D}$ by merging (by adding their multiplicities) all configurations that are identical when the primes are removed from the objects. For instance, if $c_1 = [a\ b]_h$, $c_2 = [a'\ b]_h$, and $c_3 = [a'\ b']_h$ with $\mathcal{D}[c_1] = 2$, $\mathcal{D}[c_2] = 3$, and $\mathcal{D}[c_3] = 1$, then we set $\mathcal{C}'[c_1] = 6$; note that table $\mathcal{C}'$ does not have the entries for c_2 and c_3, since they contain primed objects. This merging operation can be performed in polynomial time with respect to $O(2^{2v} \times t)$, the size of table $\mathcal{D}$. The total time required by the simulation algorithm is given by the most expensive rule simulation (line 13, requiring polynomial time with respect to 2^v and t) which is executed $O(m \times 2^v \times 2^v)$ times (loop of line 9). □

As a consequence, the whole simulation can be carried out efficiently with respect to the number of steps to be simulated.

Theorem 4.4. A semi-uniform family of confluent recogniser tissue P systems with cell volume $v(n)$ constructed in polynomial time $p(n)$ by a Turing machine and working in time $t(n)$ can be simulated by a deterministic Turing machine in polynomial time with respect to $p(n)$, $t(n)$, and $2^{v(n)}$.

Proof:
Each P system of the family has at most $p(n)$ rules; hence, by Lemma 4.3, one of its possible configurations at time t can be constructed in polynomial time with respect to $2^{v(n)}$, t, and $p(n)$. Since the P systems are confluent, any computation constructed in this way suffices to establish their result. The total time required to simulate the system is the summation for all t from 0 to $t(n)$, i.e., the sum of a polynomial number of polynomial terms. □

The exponential time hypothesis (ETH) [6] is the claim that there does not exist a sub-exponential time algorithm for k-SAT if $k \geq 3$ or, more formally, that

$$s_k = \inf \left\{ \delta > 0 : \text{there exists a } O(2^{\delta n}) \text{ algorithm for } k\text{-SAT} \right\} > 0$$

for all $k \geq 3$. Clearly this statement implies $\mathbf{P} \neq \mathbf{NP}$, but it is possibly stronger. This is connected to the ability of families of tissue P systems to solve $\mathbf{NP}$-problems in polynomial time with sub-polynomial cell volume.

Corollary 4.5. No semi-uniform family $\mathbf{\Pi}$ of confluent recogniser tissue P systems with sub-polynomial cell volume $v(n)$ can solve an $\mathbf{NP}$-complete problem in polynomial time, unless the exponential time hypothesis is false.

Proof:
Suppose $v(n)$ sub-polynomial, that is, $v(n) \in o(n^{\epsilon})$ for all reals $\epsilon > 0$. Let $\mathbf{\Pi}$ be constructed in time $p(n)$ for some polynomial p. If $\mathbf{\Pi}$ solved an $\mathbf{NP}$-complete problem in polynomial time, then it would also solve 3-SAT in polynomial time $t(n)$, by changing the machine providing the uniformity condition with one that performed the necessary reduction from 3-SAT before constructing the P systems. Then, by Theorem 4.4, the family $\mathbf{\Pi}$ could be simulated in polynomial time with respect to $2^{v(n)}$, $t(n)$, and $p(n)$. In particular, this means that the simulation time would be bounded above by $\left(2^{v(n)} \times t(n) \times p(n)\right)^{\ell}$ for some ℓ. This would imply

$$\lim_{n\to\infty} \frac{\left(2^{v(n)} \times t(n) \times p(n)\right)^{\ell}}{2^{\delta n}} = \lim_{n\to\infty} 2^{n\left(\ell \frac{v(n)}{n} + \ell \frac{\log t(n)}{n} + \ell \frac{\log p(n)}{n} - \delta\right)} = \lim_{n\to\infty} 2^{-\delta n} = 0$$

for all $\delta > 0$, falsifying the exponential time hypothesis. □

In particular, the standard complexity theory assumption $\mathbf{P} \neq \mathbf{NP}$ implies that no $\mathbf{NP}$-complete problem can be solved by tissue P systems in polynomial time using only logarithmic cell volume.

Corollary 4.6. Let $\mathcal{D}$ be the class of tissue P systems with cell division, cell separation, or both and having logarithmic cell volume. Then $\mathbf{PMC}_{\mathcal{D}} = \mathbf{PMC}^{\star}_{\mathcal{D}} = \mathbf{P}$. As a consequence, no uniform or semi-uniform family of confluent recogniser tissue P systems with logarithmic cell volume can solve an $\mathbf{NP}$-complete problem in polynomial time, unless $\mathbf{P} = \mathbf{NP}$.

Proof:
Similar to the proof of Corollary 4.5, a family $\mathbf{\Pi}$ solving an $\mathbf{NP}$-complete problem in polynomial time $t(n)$ with volume $v(n)$ can be simulated in time bounded by $(2^{v(n)} \times t(n) \times p(n))^{\ell}$ for some ℓ. This amount is polynomial with respect to n, since $v(n) \in O(\log n)$ implies $2^{v(n)} \in O(n^k)$ for some k. □

5. Conclusions and Open Problems

We showed that families of tissue P systems with constant cell volume are powerful enough to be universal (assuming an exponential-time uniformity condition), but that a sub-polynomial cell volume does not allow them (under standard complexity theory assumptions) to solve intractable problems in polynomial time, *even if* exponentially many cells can be created in polynomial time by cell division or separation.

The proof of universality for families of tissue P systems of constant cell volume employs an exponential time semi-uniformity condition. We conjecture that this is indeed unavoidable and, in particular, that no universal (i.e., simulating a universal machine) family of tissue P systems with constant cell volume can be constructed in less than exponential time.

Another interesting open problem is related to the volume of membranes in cell-like P systems, such as P systems with active membranes [13]. It is easy to define a natural notion of "membrane volume" for these kinds of P systems, where a membrane recursively contains all its children membranes. In that case, what is the parameter whose value allows cell-like P systems to be universal or to solve $\mathbf{NP}$-complete problems in polynomial time?

Acknowledgements This work was partially supported by Fondo d'Ateneo 2016 of Università degli Studi di Milano-Bicocca, project 2016-ATE-0492 "Sistemi a membrane: classi di complessità spaziale e temporale".

References

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

[2] Burkhard, H.-D.: Ordered firing in Petri nets, *Elektronische Informationsverarbeitung und Kybernetik (Journal of Information Processing and Cybernetics)*, **17**(2/3), 1981, 71–86.

[3] Desel, J., Reisig, W.: Place/transition Petri nets, in: *Lectures on Petri nets I: Basic models* (W. Reisig, G. Rozenberg, Eds.), vol. 1491 of *Advances in Petri Nets*, Springer, 1998, 122–173.

[4] Frisco, P.: P systems, Petri nets, and program machines, in: *Membrane Computing, 6th International Workshop, WMC 2005* (R. Freund, Gh. Păun, G. Rozenberg, A. Salomaa, Eds.), vol. 3850 of *Lecture Notes in Computer Science*, Springer, 2006, 209–223.

[5] Gutiérrez-Escudero R., Pérez-Jiménez M.J., Rius-Font M.: Characterizing tractability by tissue-like P systems in: *Membrane Computing 10th International Workshop, WMC 2009* (Gh. Păun, M. J. Pérez-Jiménez, A. Riscos-Nuñez, G. Rozenberg, A. Salomaa, Eds.), vol. 5957 of *Lecture Notes in Computer Science*, Springer, 2010, 289–300.

[6] Impagliazzo, R., Paturi, R.: On the complexity of k-SAT, *Journal of Computer and System Sciences*, **62**(2), 2001, 367–375.

[7] Leporati, A., Manzoni, L., Mauri, G., Porreca, A. E., Zandron, C.: Constant-space P systems with active membranes, *Fundamenta Informaticae*, **134**(1–2), 2014, 111–128.

[8] Leporati, A., Manzoni, L., Mauri, G., Porreca, A. E., Zandron, C.: Tissue P systems can be simulated efficiently with counting oracles, in: *Membrane Computing, 16th International Conference, CMC 2015* (G. Rozenberg, A. Salomaa, J. M. Sempere, C. Zandron, Eds.), vol. 9504 of *Lecture Notes in Computer Science*, Springer, 2015, 251–261.

[9] Martín-Vide, C., Păun, Gh., Pazos, J., Rodríguez-Patón, A.: Tissue P systems, *Theoretical Computer Science*, **296**(2), 2003, 295–326.

[10] Minsky, M.: *Computation: Finite and Infinite Machines*, Prentice-Hall, 1967.

[11] Murphy, N., Woods, D.: The computational power of membrane systems under tight uniformity conditions, *Natural Computing*, **10**(1), 2011, 613–632.

[12] Pan, L., Pérez-Jiménez, M. J.: Computational complexity of tissue-like P systems, *Journal of Complexity*, **26**(3), 2010, 296–315.

[13] Păun, Gh.: P systems with active membranes: Attacking NP-complete problems, *Journal of Automata, Languages and Combinatorics*, **6**(1), 2001, 75–90.

[14] Păun, Gh., Pérez-Jiménez, M. J., Riscos-Núñez, A.: Tissue P systems with cell division, *International Journal of Computers, Communications & Control*, **3**(3), 2008, 295–303.

[15] Zandron, C., Ferretti, C., Mauri, G.: Solving NP-complete problems using P systems with active membranes, in: *Unconventional Models of Computation, UMC'2K, Proceedings of the Second International Conference* (I. Antoniou, C. S. Calude, M. J. Dinneen, Eds.), Springer, 2001, 289–301.