13th International Conference on Membrane Computing, CMC13,
Budapest, Hungary, August 28 - 31, 2012. Proceedings, pages 419 - 432.

Limits of the Power of Tissue P Systems with Cell Division

Petr Sosík[1,2]

[1] Departamento de Inteligencia Artificial, Facultad de Informática,
Universidad Politécnica de Madrid, Campus de Montegancedo s/n,
Boadilla del Monte, 28660 Madrid, Spain,
[2] Research Institute of the IT4Innovations Centre of Excellence,
Faculty of Philosophy and Science, Silesian University in Opava
74601 Opava, Czech Republic, petr.sosik@fpf.slu.cz

Abstract. Tissue P systems generalize the membrane structure tree usual in original models of P systems to an arbitrary graph. Basic operations in these systems are communication rules, enriched in some variants with cell division or cell separation. Several variants of tissue P systems were recently studied, together with the concept of uniform families of these systems. Their computational power was shown to range between **P** and **NP** $\cup$ **co-NP**, thus characterizing some interesting borderlines between tractability and intractability. In this paper we show that computational power of these uniform families in polynomial time is limited by the class **PSPACE**. This class characterizes the power of many classical parallel computing models.

1 Introduction

P systems (also membrane systems) can be described as bio-inspired computing models trying to capture information and control aspects of processes in living cells. P systems are focusing, e.g., on molecular synthesis within cells, selective particle recognition by membranes, controlled transport through protein channels, membrane division, membrane dissolution and many others. These processes are modeled in P systems by means of operations on multisets in separate cell-like regions.

Tissue P systems were introduced first in [9] where they were described as a kind of abstract neural nets. Instead of considering a hierarchical arrangement usual in previous models of P systems, membranes/cells are placed in the nodes of a virtual graph. Biological justification of the model (see [10]) is the intercellular communication and cooperation between neurons and, generally, between tissue cells. The communication among cells is based on symport/antiport rules which were introduced to P systems in [14]. Symport rules move objects across a membrane together in one direction, whereas antiport rules move objects across a membrane in opposite directions. In tissue P systems these two variants were unified as a unique type of rule. From the original definitions of tissue P systems

[9, 10], several research lines have been developed and other variants have arisen (see, for example, [1, 2, 5, 7, 8, 11, 12]).

An interesting variant of tissue P systems was presented in [15] and named *tissue P systems with cell division*. The model is enriched with the operation of cell replication, that is, two new cells are generated from one original cell by a division rule. The new cells have exactly the same objects except for at most a pair of different objects. The following results were obtained: (a) only tractable problems can be efficiently solved when the length of communication rules is restricted to 1, and (b) an efficient (uniform) solution to the `SAT` problem exists when using communication rules with length at most 3 (and, of course, division rules). Hence, in the framework of recognizer tissue P systems with cell division, the length of the communication rules provides a borderline between efficiency and non-efficiency.

In this paper we impose an upper bound on the power of several types of tissue P systems. Specifically, we show that tissue systems with cell division can be simulated in polynomial space. As a consequence, the class of problems solvable by uniform families of these systems in polynomial time is limited by the class PSPACE.

The paper is organized as follows: first, we recall some preliminaries, and then the definition of tissue P systems with cell division is given. Next, recognizer tissue P systems and computational complexity classes in this framework are briefly described. In Section 3 we demonstrate that any such tissue P system can be simulated by a classical computer (and, hence, also by Turing machine) in polynomial space. The last section contains conclusions and some open problems.

2 Tissue P Systems with Cell Division

We fix some notation first. A *multiset* m with underlying set A is a pair (A, f) where $f : A \to \mathbb{N}$ is a mapping. If $m = (A, f)$ is a multiset then its *support* is defined as $supp(m) = \{x \in A \mid f(x) > 0\}$. The total number of elements in a multiset, including repeated memberships, is the *cardinality* of the multiset. A multiset is empty (resp. finite) if its support is the empty set (resp. a finite set). If $m = (A, f)$ is a finite multiset over A, and $supp(m) = \{a_1, \ldots, a_k\}$ then it can also be represented by the string $a_1^{f(a_1)} \ldots a_k^{f(a_k)}$ over the alphabet $\{a_1, \ldots, a_k\}$. Nevertheless, all permutations of this string precisely identify the same multiset m. Throughout this paper, we speak about "the finite multiset m" where m is a string, and meaning "the finite multiset represented by the string m".

If $m_1 = (A, f_1)$, $m_2 = (A, f_2)$ are multisets over A, then we define the union of m_1 and m_2 as $m_1 + m_2 = (A, g)$, where $g = f_1 + f_2$.

For any sets A and B the *relative complement* $A \setminus B$ of B in A is defined as follows:

$$A \setminus B = \{x \in A \mid \ x \notin B\}$$

In what follows, we assume the reader is already familiar with the basic notions and the terminology of P systems. For details, see [16].

2.1 Basic Definition

Tissue P Systems with cell division is based on the cell-like model of P systems with active membranes [13]. The biological inspiration is the following: alive tissues are not static network of cells but new cells are produced by membrane division in a natural way. In these models, the cells are not polarized; the two cells obtained by division have the same labels as the original cell, and if a cell is divided, its interaction with other cells or with the environment is blocked during the division process.

Definition 1. *A tissue P system with cell division of degree $q \geq 1$ is a tuple*

$$\Pi = (\Gamma, \mathcal{E}, \mathcal{M}_1, \ldots, \mathcal{M}_q, \mathcal{R}, i_{out}),$$

where:

1. *Γ is a finite* alphabet *whose elements are called* objects*;*
2. *$\mathcal{E} \subseteq \Gamma$ is a finite alphabet representing the set of objects initially in the environment of the system, and 0 is the label of the environment (the environment is not properly a cell of the system); let us assume that objects in the environment appear in inexhaustibly many copies each;*
3. *$\mathcal{M}_1, \ldots, \mathcal{M}_q$ are strings over Γ, representing the finite multisets of objects placed in the q cells of the system at the beginning of the computation; $1, 2, \cdots, q$ are labels which identify the cells of the system;*
4. *$\mathcal{R}$ is a finite set of rules of the following forms:*
   - **(a)** Communication rules*: $(i, u/v, j)$, for $i, j \in \{0, 1, 2, \ldots, q\}, i \neq j$, $u, v \in \Gamma^*$, $|uv| > 0$. When applying a rule $(i, u/v, j)$, the objects of the multiset represented by u are sent from region i to region j and, simultaneously, the objects of the multiset v are sent from region j to region i;*
   - **(b)** Division rules*: $[a]_i \to [b]_i[c]_i$, where $i \in \{1, 2, \ldots, q\}$ and $a, b, c \in \Gamma$, and $i \neq i_{out}$. In reaction with an object a, the cell i is divided into two cells with the same label; in the first cell the object a is replaced by b; in the second cell the object a is replaced by c; the output cell i_{out} cannot be divided;*
5. *$i_{out} \in \{0, 1, 2, \ldots, q\}$ is the output cell.*

A communication rule $(i, u/v, j)$ is called a *symport rule* if $u = \lambda$ or $v = \lambda$. A symport rule $(i, u/\lambda, j)$, with $i \neq 0, j \neq 0$, provides a virtual arc from cell i to cell j. A communication rule $(i, u/v, j)$ is called an *antiport rule* if $u \neq \lambda$ and $v \neq \lambda$. An antiport rule $(i, u/v, j)$, with $i \neq 0, j \neq 0$, provides two arcs: one from cell i to cell j and another one from cell j to cell i. Thus, every tissue P systems has an underlying directed graph whose nodes are the cells of the system and the arcs are obtained from communication rules. In this context, the environment can be considered as a virtual node of the graph such that their connections are defined by the communication rules of the form $(i, u/v, j)$, with $i = 0$ or $j = 0$. Let us agree that no symport rule is permissible which would send an infinite number of objects from the environment to some cell. The length of the communication rule $(i, u/v, j)$ is defined as $|u| + |v|$.

The rules of a system like the above one are used in the non-deterministic maximally parallel manner as customary in Membrane Computing. At each step, all cells which can evolve must evolve in a maximally parallel way (at each step we apply a multiset of rules which is maximal, no further rule can be added being applicable). There is one important restriction: when a cell is divided, the division rule is the only one which is applied for that cell at that step; thus, the objects inside that cell do not evolve by means of communication rules. The label of a cell precisely identify the rules which can be applied to it.

A *configuration* of a tissue P system with cell division at any instant is described by all multisets of objects over Γ associated with all the cells present in the system, and the multiset of objects over $\Gamma - \mathcal{E}$ associated with the environment at that moment. Bearing in mind the objects from $\mathcal{E}$ have infinite copies in the environment, they are not properly changed along the computation. The *initial configuration* is $(\mathcal{M}_1, \cdots, \mathcal{M}_q; \emptyset)$. A configuration is a *halting configuration* if no rule of the system is applicable to it.

We say that configuration C_1 yields configuration C_2 in one *transition step*, denoted $C_1 \Rightarrow_\Pi C_2$, if we can pass from C_1 to C_2 by applying the rules from $\mathcal{R}$ as specified above. A *computation* of Π is a (finite or infinite) sequence of configurations such that:

1. the first term of the sequence is the initial configuration of the system;
2. each non-initial configuration of the sequence is obtained from the previous configuration by applying rules of the system in a maximally parallel manner with the restrictions previously mentioned; and
3. if the sequence is finite (called *halting computation*) then the last term of the sequence is a halting configuration.

Halting computations give a result which is encoded by the objects present in the output cell i_{out} in the halting configuration.

2.2 Recognizer Tissue P Systems with Cell Division

Let us denote a *decision problem* as a pair (I_X, θ_X) where I_X is a language over a finite alphabet (whose elements are called *instances*) and θ_X is a total boolean function over I_X. A natural correspondence between decision problems and languages over a finite alphabet can be established as follows. Given a decision problem $X = (I_X, \theta_X)$, its associated language is $L_X = \{w \in I_X : \theta_X(w) = 1\}$. Conversely, given a language L over an alphabet Σ, its associated decision problem is $X_L = (I_{X_L}, \theta_{X_L})$, where $I_{X_L} = \Sigma^*$, and $\theta_{X_L} = \{(x, 1) : x \in L\} \cup \{(x, 0) : x \notin L\}$. The solvability of decision problems is defined through the recognition of the languages associated with them, by using languages recognizer devices.

In order to study the computational efficiency of membrane systems, the notions from classical *computational complexity theory* are adapted for Membrane Computing, and a special class of cell-like P systems is introduced in [18]: *recognizer P systems*. For tissue P systems, with the same idea as recognizer cell-like P systems, *recognizer tissue P systems* is introduced in [15].

Definition 2. *A* recognizer tissue P system with cell division *of degree* $q \geq 1$ *is a tuple*

$$\Pi = (\Gamma, \Sigma, \mathcal{E}, \mathcal{M}_1, \ldots, \mathcal{M}_q, \mathcal{R}, i_{in}, i_{out})$$

where:

1. $(\Gamma, \mathcal{E}, \mathcal{M}_1, \ldots, \mathcal{M}_q, \mathcal{R}, i_{out})$ *is a tissue P system with cell division of degree* $q \geq 1$ *(as defined in the previous section).*
2. *The working alphabet* Γ *has two distinguished objects* `yes` *and* `no` *being, at least, one copy of them present in some initial multisets* $\mathcal{M}_1, \ldots, \mathcal{M}_q$*, but none of them are present in* $\mathcal{E}$*.*
3. Σ *is an (input) alphabet strictly contained in* Γ*, and* $\mathcal{E} \subseteq \Gamma \setminus \Sigma$*.*
4. $\mathcal{M}_1, \ldots, \mathcal{M}_q$ *are strings over* $\Gamma \setminus \Sigma$*;*
5. $i_{in} \in \{1, \ldots, q\}$ *is the input cell.*
6. *The output region* i_{out} *is the environment.*
7. *All computations halt.*
8. *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 $w \in \Sigma^*$, the *computation of the system* Π *with input* $w \in \Sigma^*$ starts from the configuration of the form $(\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_{i_{in}} + w, \ldots, \mathcal{M}_q; \emptyset)$, that is, the input multiset w has been added to the contents of the input cell i_{in}. Therefore, we have an initial configuration associated with each input multiset w (over the input alphabet Σ) in this kind of systems.

Given a recognizer tissue P system with cell division, we say that a computation $\mathcal{C}$ is an *accepting computation* (respectively, *rejecting computation*) if object `yes` (respectively, 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 $\mathcal{C}$.

For each natural number $k \geq 1$, we denote by $\mathbf{TDC}(k)$ the class of recognizer tissue P systems with cell division and communication rules of length at most k. We denote by $\mathbf{TDC}$ the class of recognizer tissue P systems with cell division and without restriction on the length of communication rules. Obviously, $\mathbf{TDC}(k) \subseteq \mathbf{TDC}$ for all $k \geq 1$.

2.3 Polynomial Complexity Classes of Tissue P Systems

Next, 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 numerable family of tissue P systems will be necessary in order to solve a decision problem.

Definition 3. *We say that a decision problem* $X = (I_X, \theta_X)$ *is solvable in a uniform way and polynomial time by a family* $\mathbf{\Pi} = \{\Pi(n) \mid n \in \mathbb{N}\}$ *of recognizer tissue P systems (with cell division) if the following holds:*

1. *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* $n \in \mathbb{N}$.
2. *There exists a pair* (cod, s) *of polynomial-time computable functions over* I_X *such that:*
   (a) *for each instance* $u \in I_X$, $s(u)$ *is a natural number and* $cod(u)$ *is an input multiset of the system* $\Pi(s(u))$;
   (b) *for each* $n \in \mathbb{N}$, $s^{-1}(n)$ *is a finite set;*
   (c) *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* $\Pi(s(u))$ *with input* $cod(u)$ *is halting and it performs at most* $p(|u|)$ *steps;*
   (d) *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* $\Pi(s(u))$ *with input* $cod(u)$, *then* $\theta_X(u) = 1$;
   (e) *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* $\Pi(s(u))$ *with input* $cod(u)$ *is an accepting one.*

From the soundness and completeness conditions above we deduce that every P system $\Pi(n)$ is *confluent*, in the following sense: every computation of a system with the *same* input multiset must always give the *same* answer.

Let $\mathbf{R}$ be a class of recognizer tissue P systems. We denote by $\mathbf{PMC_R}$ the set of all decision problems which can be solved in a uniform way and polynomial time by means of families of systems from $\mathbf{R}$. The following results have been proved:

Theorem 1 ([6]). $\mathbf{P} = \mathbf{PMC}_{TDC(1)}$

Theorem 2 ([15]). $\mathbf{NP} \cup \mathbf{co\text{-}NP} \subseteq \mathbf{PMC}_{TDC(3)}$

As a consequence, both **NP** and **co-NP** are contained in the class $\mathbf{PMC}_{TDC}$. In this paper we impose an upper bound on $\mathbf{PMC}_{TDC}$.

3 Simulation of Tissue P Systems with Cell Division in Polynomial Space

In this section we demonstrate that any computation of a recognizer tissue P system with cell division can be simulated in space polynomial to its initial size and the number of steps. Instead of simulating a computation of a P system from its initial configuration onwards (which would require exponential space for storing configurations), we create a recursive function which computes content of any cell h after a given number of steps. Thus we do not need to store content of cells interacting with h but we calculate it recursively whenever needed.

Simulated P systems are confluent, hence possibly nondeterministic, but the simulation will be performed in a deterministic way: only one possible sequence of configurations of the P system is traced. This corresponds to a weak priority relation between rules:

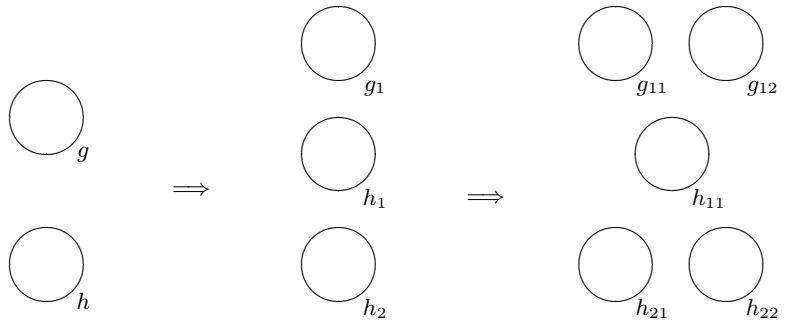

Fig. 1. An example of indexing of cells during first two computational steps.

(i) division rules are always applied prior to communication rules,
(ii) priority between communication rules given by the order they are listed,
(iii) priority between cells to which the rules are applied.

However, the confluency condition ensures that such a simulation is correct as all computations starting from the same initial configuration must lead to the same result.

Each cell of Π is assigned a unique label in the initial configuration. But cells may be divided during computation of Π, producing more membranes with the same label. To identify membranes uniquely, we add to each label a compound index. Each index is an empty string in the initial configuration. If a membrane is not divided in a computational step, digit 1 is attached to its index. If a division rule is applied, the first resulting membrane has attached 1 and the second membrane 2 to its index. After n steps of computation, index of each membrane is an n-tuple of digits from $\{1, 2\}$. Notice that some n-tuples may denote non-existing membranes as membranes need not divide at each step. The situation is illustrated in Fig. 1: membrane h is divided at first step, membranes g_1 and h_2 are divided at second step. Membrane h_{12} does not exist, for instance.

Consider a confluent recognizer tissue P system with cell division of degree $q \geq 1$, described formally as

$$\Pi = (\Gamma, \Sigma, \mathcal{E}, \mathcal{M}_1, \ldots, \mathcal{M}_q, \mathcal{R}, i_{in}, i_{out}).$$

For any cell of Π we denote the multiset of objects contained in it at any instant simply as its *content.* We construct function `Content` which computes recursively the content of any cell labeled h with index *ind* of Π after $n \geq 0$ steps of computation as follows:

1. verify whether the ancestor of cell h existed at previous computational step; if not, the cell does not exist;
2. for all rules in a fixed order: for all copies of cells affected by that rule:

   (a) subsequently and recursively calculate contents of these cells in previous step;
   (b) calculate the number of applications of the rule in the maximally parallel way;
   (c) if one of the affected cells is h, record the multiset of rules applied to it;
3. Re-calculate content of cell h in previous step of computation and apply the recorded rules to obtain new content of the cell.

When applying a rule to a particular cell in phase 2, one must start with the multiset of objects remaining in that cell after rules already applied in the same step n. Fortunately enough, it is not necessary to store contents of all cells or all multisets of rules applied to each cell in step n. Recall that the order of application of rules in $\mathcal{R}$ is fixed and so is the order of cells to which these rules are applied in a maximally parallel way. Then the multiset of rules already applied to a particular cell in step n can be always re-calculated when the cell is affected by another rule in the same step. The only value which must be stored is the total multiset of rules already applied in step n. Assume for simplicity that an input multiset of objects w is already included in the initial multiset $\mathcal{M}_{i_{in}}$.

`function` **content**

Parameters: $\ell \in \{1, \ldots, q\}$ – label of a cell
$i_1 i_2 \ldots i_n$ – a compound index
n – a number of step

Returns: the content of cell labeled ℓ with compound index $i_1 i_2 \ldots i_n$ after n steps of computation, or `null` if such a cell does not exist.

Auxiliary variables:
`rulesAppliedTo`ℓ`, rulesAppliedTotal, rulesForCell1, rulesForCell2;`
(Multisets of applicable or applied rules with underlying set $\mathcal{R}$)

`contentCell1, contentCell2, contentFinal;`
(Multisets storing contents of cells)

`if` $n = 0$ `then return` $\mathcal{M}_\ell$`;` *(return the initial multiset of cell ℓ)*
`set multiplicity of all elements in rulesAppliedTotal to 0;`
`set multiplicity of all elements in rulesAppliedTo`ℓ `to 0;`

`for each communication rule` $(j, u/v, k)$ `in` $\mathcal{R}$ `do begin`
(Now we scan all existing copies of cells labeled j and k affected by the rule.)

`rulesForCell1 := rulesAppliedTotal;`

`for each possible compound index` $j_1 j_2 \ldots j_{n-1}$ `do begin`
`contentCell1 = content(`j`,` $j_1 j_2 \ldots j_{n-1}$`,` $n-1$`);`

(Calculate the content of cell j with index $j_1 j_2 \dots j_{n-1}$ in previous step)

```
    if (contentCell1 = null) or (cell can apply a division rule)
      then skip the rest of the cycle;

    calculate the maximal multiset of rules in rulesForCell1
      applicable to cell j with objects contentCell1;
    remove these rules from multiset rulesForCell1;
    remove the corresponding objects from contentCell1;

    rulesForCell2 := rulesAppliedTotal;

    for each possible compound index k1k2...k_{n-1} do begin
      contentCell2 = content(k, k1k2...k_{n-1}, n-1);
      (Calculate the content of cell k with index k1k2...k_{n-1} in previous step)

      if contentCell2 = null or cell can apply a division rule
        then skip the rest of the cycle;

      calculate the maximal multiset of rules in rulesForCell2
        applicable to cell k with contentCell2;
      remove these rules from multiset rulesForCell2;
      remove the corresponding objects from contentCell2;

      (Now contentCell1 and ContentCell2 contain objects remaining in cell j
      with index j1j2...j_{n-1} and in cell k with index k1k2...k_{n-1},
      respectively, after application of previously scanned rules in step n.)

      let x = maximum copies of rule (j, u/v, k) applicable to cells
        j, k with contentCell1 and contentCell2, respectively;

      remove x copies of u from contentCell1;
      add x copies of rule (j, u/v, k) to rulesAppliedTotal;

      if one of the cells j or k is identical with cell ℓ
        with index i1i2...i_{n-1} then
          add x occurrences of rule (j, u/v, k) to rulesAppliedToℓ;

    end cycle; (cell k with index k1k2...k_{n-1})
  end cycle; (cell j with index j1j2...j_{n-1})
end cycle; (rule (j, u/v, k))
```

(At this moment, variable `rulesAppliedTo`ℓ *contains the complete multiset of rules applied in step n to cell ℓ with indices $i_1 i_2 \dots i_{n-1}$.)*

```
contentFinal = content(ℓ, i1i2...in−1, n − 1);
(Calculate the content of cell ℓ with index i1i2...in−1 in previous step)

if contentFinal = null then return null and exit;

if a division rule [a]ℓ → [b]ℓ[c]ℓ exists such that
contentFinal contains a then
  if in = 1 then
    remove a from contentFinal and add b;
  else
    remove a from contentFinal and add c;
  (Cell ℓ with index i1i2...in−1 divides in step n)

else
  if in = 2 then
    return null and exit;
  (The last element in of compound index corresponds to a copy of cell ℓ
  dividing in step n which is not the case, hence this copy does not exist.)

  else
    apply all rules in rulesAppliedToℓ to contentFinal, i.e.,
      add/remove multisets of objects corresponding to cell ℓ
      in rules to/from contentFinal;

return contentFinal;
```

We defined explicitly internal variables with largest memory demands in function `content` in its preamble. Other variables are used implicitly. This is necessary for the following result.

Theorem 3. *A result of any computation consisting of n steps of a confluent tissue P system with cell division can be computed with Turing machine in space polynomial to n.*

Proof. Consider a confluent tissue P system with cell division

$$\Pi = (\Gamma, \Sigma, \mathcal{E}, \mathcal{M}_1, \ldots, \mathcal{M}_q, \mathcal{R}, i_{in}, i_{out}).$$

The function `content` described above evaluates the content of a particular cell after n steps, but simultaneously also an application of all possible rules during n-th step in all cells is also simulated. Hence, it is very easy to check whether any rule is applied or, on the contrary, whether the computation stops (the multiset `rulesAppliedTotal` is empty). The result of computation of Π with an input w is obtained as follows:

1. Prepare the initial configuration of Π, add w to $\mathcal{M}_{i_{in}}$.

2. Subsequently compute `content`$(i_{out}, 11\ldots1, n)$ for $n = 0, 1, 2, ...$ until no rule is applicable. Note that the output membrane never divides and hence its index contains only 1's. Record in each step the presence of objects `yes` or `no`.
3. If one of objects `yes` or `no` appeared in the output membrane and only in the last step, return the result of computation.

Space complexity of the function `content`$(\ell, index, n)$ is determined by variables storing multisets of objects and applicable rules. The first type represents a multiset of objects contained in a particular cell. Its cardinality is limited from above by the total number of objects in the system after n steps. Denote this number by o_n. Therefore,

$$o_0 = \sum_{i=1}^{q} card(\mathcal{M}_i) + |w|. \tag{1}$$

At each step each cell can divide (which does not increase the number of its objects) or it can introduce new object to the system from the environment via antiport rules. Denote $\mathcal{R}_a$ the set of antiport rules in $\mathcal{R}$. Hence, we can write that $o_n \leq co_{n-1}$ for $n \geq 1$ and a constant c, where

$$c = \max\{\max_{(i,u/v,j)\in\mathcal{R}_a}\{|u|/|v|\}, \max_{(i,u/v,j)\in\mathcal{R}_a}\{|v|/|u|\}\}. \tag{2}$$

After n step we have

$$o_n \leq o_0 c^n \tag{3}$$

which is a value representable by dn bits for a constant

$$d \leq \log o_0 + \log c. \tag{4}$$

Finally, $|\Gamma|dn$ bits are necessary to describe any multiset with cardinality dn and with the underlying set Γ. This is also the maximum size of any variable of this type.

The situation is similar for multisets of applicable rules. The cardinality of each such multiset at n-th computational step is limited by the number o_n of objects in the system. Hence the size of each such variable is at most $|\mathcal{R}|dn$.

Finally, let us analyze the space complexity of function `content`. Function `content` with parameter n performs recursive calls of itself with parameter $n-1$. It uses three variables storing multisets of objects and four variables with multisets of rules. For its space complexity $C(n)$ we can therefore write:

$$C(0) = \log o_0 \tag{5}$$

$$C(n) \leq C(n-1) + 3|\Gamma|dn + 4|\mathcal{R}|dn, \qquad n \geq 1. \tag{6}$$

The solution to this recurrence is

$$C(n) = \mathcal{O}((|\Gamma| + |\mathcal{R}|)dn^2 + \log o_0). \tag{7}$$

Hence, with the aid of the function `content` described above, a conventional computer can simulate n steps of computation of the systems Π in space polynomial to n, and as the space necessary for Turing machine performing the same computation is asymptotically the same, the statement follows. □

Theorem 4. $\mathbf{PMC}_{TDC} \subseteq \mathbf{PSPACE}$

Proof. Consider a family $\mathbf{\Pi} = \{\Pi(n) \mid n \in \mathbb{N}\}$ of recognizer tissue P systems with cell division satisfying conditions of Definition 3, which solves in a uniform way and polynomial time a decision problem $X = (I_X, \theta_X)$. For each instance $u \in I_X$, denote

$$\Pi(s(u)) = (\Gamma, \Sigma, \mathcal{E}, \mathcal{M}_1, \ldots, \mathcal{M}_q, \mathcal{R}, i_{in}, i_{out})$$

and let $w = cod(u)$ be the corresponding input multiset. By Definition 3, paragraphs 1 and 2(a), the values of $card(w)$, $card(\mathcal{M}_1), \ldots, card(\mathcal{M}_q)$, and lengths of rules in $\mathcal{R}$ are exponential with respect to $|u|$ (they must be constructed by a deterministic Turing machine in polynomial time). Furthermore, values of $|\Gamma|$ and $|\mathcal{R}|$ are polynomial to $|u|$. (Actually, the alphabet Γ could possibly have exponentially many elements but only polynomially many of them could appear in the system $\Pi(s(u))$ during its computation and the rest could be ignored.)

By Definition 3, paragraph 2(c), also the number of steps n of any computation of system $\Pi(s(u))$ is polynomial to $|u|$. Then by (1)–(4) the value of d is polynomial to $|u|$ and, by (7), so is the space complexity of function `content`. Therefore, each instance $u \in I_X$ can be solved with a Turing machine in space polynomial to $|u|$. □

4 Discussion

The results presented in this paper establish a theoretical upper bound on the power of confluent tissue P systems with cell division. Note that the characterization of power of non-confluent (hence non-deterministic) tissue P systems with cell division remains open. The presented proof cannot be simply adapted to this case by using a non-deterministic Turing (or other) machine for simulation. Observe that in our recursive algorithm the same configuration of a P system is typically re-calculated many times during one simulation run. If the simulation was non-deterministic, we could obtain different results for the same configuration which would make the simulation non-consistent.

If we defined a descriptional complexity (i.e., a size of description) of any tissue P system with cell division, Theorem 3 could be rephrased as follows: any computation of such a P systems can be simulated in space polynomial to the size of description of that P system and to the number of steps of its computation.

Another variant one could consider is the case when a cell can divide using a rule of type $[a]_\ell \to [b]_\ell[c]_\ell$ and it can communicate in the same step. To be consistent, one should perform communication first (preserving the object a) and then divide the resulting cell to two membranes, replacing a with b or c, respectively. The presented proofs can be simply adapted to this variant.

The presented result is related to two other results which also deals with the relation of the class **PSPACE** to the computational power of certain families of P systems. The first of them is the result presented in [20] which deals with P systems with active membranes, equipped with a similar division of membranes as here. The P systems with active membranes, however, use an acyclic communication graph (a tree of membrane structure), while here we work with an arbitrary graph which makes the structure of the proof different. It was shown in [20] that the class **PSPACE** characterizes precisely the computational power of P systems with active membranes. The second related result [19] studies the model very similar to that used here: tissue P systems with cell separation. The upper bound **PSPACE** to their computational power is proven in [19]. It remains open whether this upper bound on the power of polynomially uniform families of tissue P systems with cell division or cell separation can be still improved or not.

Acknowledgements

This work was supported by the European Regional Development Fund in the IT4Innovations Centre of Excellence project (CZ.1.05/1.1.00/02.0070), and by the Silesian University in Opava under the Student Funding Scheme, project no SGS/7/2011.

References

1. Alhazov, A., Freund, R. and Oswald, M. Tissue P Systems with Antiport Rules ans Small Numbers of Symbols and Cells. *Lecture Notes in Computer Science* **3572**, (2005), 100–111.
2. Bernardini, F. and Gheorghe, M. Cell Communication in Tissue P Systems and Cell Division in Population P Systems. *Soft Computing* **9**, 9, (2005), 640–649.
3. Christinal, H.A., Díaz-Pernil, D., Gutiérrez-Naranjo, M.A. and Pérez-Jiménez, M.J. Tissue-like P systems without environment. In M.A. Martínez-del-Amor, Gh. Păun, I. Pérez-Hurtado, A. Riscos-Núñez (eds.) *Proceedings of the Eight Brainstorming Week on Membrane Computing*, Sevilla, Spain, February 1-5, 2010, Fénix Editora, Report RGNC 01/2010, pp. 53–64.
4. Díaz-Pernil, D., Gutiérrez-Naranjo, M.A., Pérez-Jiménez, M.J., Riscos-Núñez , A. and Romero–Campero, F.J. Computational efficiency of cellular division in tissue-like P systems. *Romanian Journal of Information Science and Technology* **11**, 3, (2008), 229–241.
5. Freund, R., Păun, Gh. and Pérez-Jiménez, M.J. Tissue P Systems with channel states. *Theoretical Computer Science* **330**, (2005), 101–116.
6. Gutiérrez-Escudero, R., Pérez-Jiménez, M.J. and Rius–Font, M. Characterizing tractability by tissue-like P systems. *Lecture Notes in Computer Science* **5957**, (2010), 289–300.
7. Krishna, S.N., Lakshmanan K. and Rama, R. Tissue P Systems with Contextual and Rewriting Rules. *Lecture Notes in Computer Science* **2597**, (2003), 339–351.

8. Lakshmanan K. and Rama, R. On the Power of Tissue P Systems with Insertion and Deletion Rules. In A. Alhazov, C. Martín-Vide and Gh. Păun (eds.) *Preproceedings of the Workshop on Membrane Computing*, Tarragona, Report RGML 28/03, (2003), pp. 304–318.
9. Martín Vide, C. Pazos, J. Păun, Gh. and Rodríguez Patón, A. A New Class of Symbolic Abstract Neural Nets: Tissue P Systems. *Lecture Notes in Computer Science* **2387**, (2002), 290–299.
10. Martín Vide, C. Pazos, J. Păun, Gh. and Rodríguez Patón, A. Tissue P systems. *Theoretical Computer Science*, **296**, (2003), 295–326.
11. Pan, L. and Ishdorj, T.-O. P systems with active membranes and separation rules. *Journal of Universal Computer Science*, **10**, 5, (2004), 630–649.
12. Pan, L. and Pérez-Jiménez, M.J. Computational complexity of tissue–like P systems. *Journal of Complexity*, **26**, 3 (2010), 296–315.
13. Gh. Păun, P systems with active membranes: attacking NP complete problems, J. Automata, Languages and Combinatorics, 6, 1 (2001), 75–90.
14. Păun, A. and Păun, Gh. The power of communication: P systems with symport/antiport. *New Generation Computing*, **20**, 3, (2002), 295–305.
15. Păun, Gh., Pérez-Jiménez, M.J. and Riscos-Núñez, A. Tissue P System with cell division. *In. J. of Computers, Communications and Control*, **3**, 3, (2008), 295–303.
16. Gh. Păun, G. Rozenberg and A. Salomaa. *The Oxford Handbook of Membrane Computing*, Oxford University Press, 2009.
17. Pérez-Jiménez, M.J., Romero-Jiménez, A. and Sancho-Caparrini, F. Complexity classes in models of cellular computing with membranes. *Natural Computing*, **2**, 3 (2003), 265–285.
18. Pérez-Jiménez, M.J., Romero-Jiménez, A. and Sancho-Caparrini, F. A polynomial complexity class in P systems using membrane division. *Journal of Automata, Languages and Combinatorics*, **11**, 4, (2006), 423-434.
19. Sosík, P., Cienciala, L.: Tissue P Systems with Cell Separation: Upper Bound by PSPACE. In: *Proceedings of the 1st International Conference on the Theory and Practice of Natural Computing (TPNC 2012)*. To appear (2012).
20. Sosík, P., Rodríguez-Patón, A.: Membrane computing and complexity theory: A characterization of PSPACE. *J. Comput. System Sci.* 73(1), 137–152 (2007)
21. The P Systems Web Page, `http://ppage.psystems.eu/`, [cit. 2012-5-29]