Tissue P Systems with Cell Division

Gheorghe PĂUN[1,2], Mario PÉREZ-JIMÉNEZ[2], Agustín RISCOS-NÚÑEZ[2]

[1]Institute of Mathematics of the Romanian Academy
PO Box 1-764, 014700 Bucureşti, Romania

[2] Research Group on Natural Computing
Department of Computer Science and Artificial Intelligence
Technical Higher School of Computer Science Engineering
University of Sevilla
Avda. Reina Mercedes s/n, 41012 Sevilla, Spain
E-mail: {gpaun, marper, ariscosn}@us.es

Abstract. In tissue P systems several cells (elementary membranes) communicate through symport/antiport rules, thus carrying out a computation. We add to such systems the basic feature of (cell) P systems with active membranes – the possibility to divide cells. As expected (as it is the case for P systems with active membranes), in this way we get the possibility to solve computationally hard problems in polynomial time; we illustrate this possibility with SAT problem.

1 Introduction

In membrane computing, there are two main classes of P systems: with the membranes arranged hierarchically, inspired from the structure of the cell, and with the membranes placed in the nodes of a graph, all of them at the same level, inspired from the cell inter-communication in tissues. A particularly interesting sub-class of the first class are the systems with active membranes, where the membrane division can be used in order to solve hard problems, e.g., **NP**-complete problems, in polynomial or even linear time, by a space-time trade-off. In the tissue P systems, the communication among cells is performed by means of symport/antiport rules, well-known in biology. Details can be found in [2], [3], as well as in the comprehensive page from the web address http://psystems.disco.unimib.it).

In this paper we combine the two definitions, and consider tissue P systems (with the communication done through symport/antiport rules) with cell division rules of the same form as in P systems with active membranes, but without using polarizations. The rules are used in the non-deterministic maximally parallel way, with the restriction that if a division rule is used for dividing a cell, then this cell does not participate in any other rule, for division or communication (when dividing, the interaction of the cell with other cells or with the environment is blocked); the cells obtained by division have the same labels as the mother cell, hence the rules to be used for evolving them or their objects are inherited (the label precisely identifies the available rules).

This natural extension of tissue P systems provides the possibility of solving SAT problem in polynomial time, in a confluent way: at precise times, one of the objects yes or no is sent to the environment, giving the answer to the question whether the input propositional formula is satisfiable. The construction is uniform: in a polynomial time, a family of recognizing tissue P systems with cell division is constructed, which, receiving as inputs encodings of instances of SAT, tells us whether or not these instances are satisfiable.

2 Tissue P Systems with Cell Division

We assume the reader to be familiar with basic elements of membrane computing and we directly define the class of P systems which is investigated in this paper.

A *tissue P system with cell division* is a construct

$$\Pi = (O, w_1, \ldots, w_m, E, R, i_o),$$

where:

1. $m \geq 1$ (the initial degree of the system; the system contains m cells, labeled with $1, 2, \ldots, m$);
2. O is the alphabet of *objects*;
3. $w_1, \ldots, w_m$ are strings over O, describing the *multisets of objects* placed in the m cells of the system;
4. $E \subseteq O$ is the set of objects present in the environment in arbitrarily many copies each;
5. R is a finite set of *developmental rules*, of the following forms:
   (a) $(i, x/y, j)$, for $i, j \in \{0, 1, 2, \ldots, m\}, i \neq j$, and $x, y \in O^*$;
   communication rules; $1, 2, \ldots, m$ identify the cells of the system, 0 is the environment; when applying a rule $(i, x/y, j)$, the objects of the multiset represented by x are sent from region i to region j and simultaneously the objects of the multiset y are sent from region j to region i;
   (b) $[\ a\ ]_i \rightarrow [\ b\ ]_i [\ c\ ]_i$, where $i \in \{1, 2, \ldots, m\}$ and $a, b, c \in O$;
   division rules; under the influence of object a, the cell with label i is divided in two cells with the same label; in the first copy the object a is replaced by b, in the second copy the object a is replaced by c; all other objects are replicated and copies of them are placed in the two new cells.

Therefore, we use antiport rules for communication (for a rule $(i, x/y, j)$ we say that the maximum of the lengths of x and y is the *weight* of the rule), and division rules as in P systems with active membranes.

The rules of a system as above are used in the non-deterministic maximally parallel manner as customary in membrane computing. In each step, all objects and all cells which can evolve must evolve (that is, in each step we apply a set of rules which is maximal, no further rule can be added), with the following important mentioning: if a cell is divided, then the division rule is the only one which is applied for that cell in that step, its objects do not evolve by means of communication rules. This is like saying that a cell which divides first cuts all its communication channels with the other cells and with the environment; the dotter cells will participate to the interaction with other cells or with the environment only in the next step – providing that they are not divided once again.

The computation starts from the initial configuration and proceeds as defined above; only halting computations give a result, and the result is the number of objects present in the halting configuration in cell i_o; the set of numbers computed in this way by the various halting computations in Π is denoted by $N(\Pi)$.

In the present paper we are not interested in the computing power of systems as above – already systems without membrane division are known to be Turing complete (see [2], [1], etc.), but in their computing efficiency. That is why we introduce a variant of tissue P systems with membrane division, namely *recognizing systems with input*. Such a system has the form $\Pi = (O, w_1, \ldots, w_m, E, R, i_{in})$, with the set O containing two distinguished objects, `yes` and `no`, present in at least one copy in $w_1 w_2 \ldots w_m$ but not present in E, and with $i_{in} \in \{1, 2, \ldots, m\}$ being the input cell. The computations of the system Π start from configurations of the form $(w_1, w_2, \ldots, w_{in} w, \ldots, w_m; E)$, where $w \in O^*$ (that is, after adding the multiset w to the contents of the input cell); all computations halt; in the last step of a computation either a copy of the object `yes` or a copy of the object `no` is sent into the environment; we say that the multiset w is recognized/accepted by Π if and only if the object `yes` was sent out.

If the multiset w codifies an instance $Q(n)$ of a decision problem Q, then we say that Π answers the question whether or not $Q(n)$ has an affirmative answer (w is accepted if and only if the codified instance $Q(n)$ has the answer *yes*). Thus, we say that the problem Q is solved in a time bounded by a mapping f if and only if a class of recognizing P systems with input, $\Pi(Q, n)$, can be constructed in polynomial time by a Turing machine starting from Q and n, such that $\Pi(Q, n)$ halts in less than $f(n)$ steps when starting with the input $w(Q, n)$, which codifies $Q(n)$, and provides the answer `yes` if and only if $Q(n)$ has the affirmative answer. (The code $w(Q, n)$ should be obtained in polynomial time by a Turing machine, starting from the instance $Q(n)$.) The set of all decision problems which can be solved as above in a number of steps bounded by a mapping f form the complexity class $\mathbf{MC}_{TD}(f)$. Here we are interested in polynomial time solutions, hence we consider the class $\mathbf{PMC}_{TD}$, obtained as the union of all classes $\mathbf{MC}_{TD}(f)$, for all polynomials f.

More precise definitions of complexity classes in terms of membrane computing can be found in [3]. We close this section with an important remark about the previous way of solving decision problems. Specifically, we have said nothing about the way the computations proceed; in particular, they can be non-deterministic, as standard in membrane computing. It is important however that the systems always stop and always they send out an object which is the correct answer to the input problem. In other terms, the systems are *confluent, sound, and complete*.

3 Solving `SAT` in Polynomial Time

As expected, the possibility to divide cells means the possibility to create an exponential space in a linear time, and this space can be used in order to obtain fast solutions to computationally hard problems.

Theorem 3.1 *Tissue P systems with active membranes can solve* `SAT` *in polynomial time. (Otherwise stated,* `SAT` $\in \mathbf{PMC}_{TD}$*.)*

Proof. Let us consider a propositional formula $\gamma = C_1 \wedge \ldots \wedge C_m$, consisting of m clauses $C_j = y_{j,1} \vee \ldots \vee y_{j,k_j}$, $1 \leq j \leq m$, where $y_{j,i} \in \{x_l, \neg x_l \mid 1 \leq l \leq n\}$, $1 \leq i \leq k_j$ (there are used n variables). Without loss of generality, we may assume that no clause

contains two occurrences of some x_i or two occurrences of some $\neg x_i$ (the formula is not redundant at the level of clauses), or both x_i and $\neg x_i$ (otherwise such a clause is trivially satisfiable, hence can be removed).

We codify γ, which is an instance of SAT with size parameters n and m, by the multiset

$$\begin{aligned} w(\gamma) &= \{s_{i,j} \mid y_{j,r} = x_i,\ 1 \leq i \leq n, 1 \leq j \leq m, 1 \leq r \leq k_j\} \\ &\cup \{s'_{i,j} \mid y_{j,r} = \neg x_i,\ 1 \leq i \leq n, 1 \leq j \leq m, 1 \leq r \leq k_j\}. \end{aligned}$$

(We replace each variable x_i from each clause C_j with $s_{i,j}$ and each negated variable $\neg x_i$ from each clause C_j with $s'_{i,j}$, then we remove all parentheses and connectives. In this way we pass from γ to $w(\gamma)$ in a number of steps which is linear with respect to $n \cdot m$.)

We construct the recognizing tissue P system (of degree 2) with input

$$\Pi = (O, w_1, w_2, E, R, 2),$$

with the following components:

$$\begin{aligned} O &= \{a_i, t_i, f_i \mid 1 \leq i \leq n\} \cup \{r_i \mid 1 \leq i \leq m\} \\ &\cup \{T_i, F_i \mid 1 \leq i \leq n\} \cup \{T_{i,j}, F_{i,j} \mid 1 \leq i \leq n, 1 \leq j \leq m+1\} \\ &\cup \{b_i \mid 1 \leq i \leq 3n+m+1\} \cup \{c_i \mid 1 \leq i \leq n+1\} \\ &\cup \{d_i \mid 1 \leq i \leq 3n+nm+m+2\} \cup \{e_i \mid 1 \leq i \leq 3n+nm+m+4\} \\ &\cup \{f, g, \texttt{yes}, \texttt{no}\}, \\ w_1 &= \texttt{yes no } b_1 c_1 d_1 e_1, \\ w_2 &= f g a_1 a_2 \ldots a_n, \\ E &= O - \{\texttt{yes}, \texttt{no}\}, \end{aligned}$$

and the following rules.

1. **Division rules:**

   $[\ a_i\ ]_2 \rightarrow [\ T_i\ ]_2 [\ F_i\ ]_2$, for all $i = 1, 2, \ldots, n$.

   (Membrane 2 is repeatedly divided, each time expanding one object a_i, corresponding to a variable x_i, into T_i and F_i, corresponding to the values *true* and *false* which this variable may assume. In this way, in n steps, we get 2^n cells with label 2, each one containing one of the 2^n truth-assignments possible for the n variables. The objects f, g are duplicated, hence a copy of each of them will appear in each cell.)

2. **Communication rules:**

   $(1, b_i / b_{i+1}^2, 0)$, for all $i = 1, 2, \ldots, n+1$,
   $(1, c_i / c_{i+1}^2, 0)$, for all $i = 1, 2, \ldots, n+1$,
   $(1, d_i / d_{i+1}^2, 0)$, for all $i = 1, 2, \ldots, n+1$,
   $(1, e_i / e_{i+1}, 0)$, for all $i = 1, 2, \ldots, 3n+nm+m+3$.

   (In parallel with the operation of dividing cell 2, the counters b_i, c_i, d_i, e_i from cell 1 grow their subscripts. In each step, the number of copies of objects of the first three types is doubled, hence after n steps we get 2^n copies of b_{n+1}, c_{n+1}, and d_{n+1}. Objects b_i will check which clauses are satisfied by a given truth-assignment, objects c_i are used in order to multiply the number of copies of t_i, f_i as we will see immediately, d_i are used to check whether there is at least one truth-assignment which

satisfies all clauses, and e_i will be used in order to produce the object `no`, if this will be the case, in the end of the computation.)

$(1, b_{n+1}c_{n+1}/f, 2)$,
$(1, d_{n+1}/g, 2)$.

(In step $n+1$, the counters $b_{n+1}, c_{n+1}, d_{n+1}$ are brought in cells with label 2, in exchange of f and g. Because we have 2^n copies of each object of these types and 2^n cells 2, each one containing exactly one copy of f and one of g, due to the maximality of the parallelism of using the rules, each cell 2 gets precisely one copy of each of $b_{n+1}, c_{n+1}, d_{n+1}$. Note that cells 2 cannot divide any more, because the objects a_i were exhausted.)

$(2, c_{n+1}T_i/c_{n+1}T_i, 0)$,
$(2, c_{n+1}F_i/c_{n+1}F_{i,1}, 0)$, for each $i = 1, 2, \ldots, n$,
$(2, T_{i,j}/t_iT_{i,j+1}, 0)$,
$(2, F_{i,j}/f_iF_{i,j+1}, 0)$, for each $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, m$.

(In the presence of c_{n+1}, the objects T_i, F_i introduce the objects $T_{i,1}$ and $F_{i,1}$, respectively, which initiates the possibility of introducing m copies of each t_i and f_i in each cell 2. The idea is that because we have m clauses, in order to check their values for a given truth-assignment of variables, it is possible to need one value for each valiable for each clause. Note that this phase needs $2n$ steps for introducing the double-subscripted objects $T_{i,1}, F_{i,1}$ – for each one we need one step, because we have only one copy of c_{n+1} available – then further m steps are necessary for each $T_{i,1}, F_{i,1}$ to grow its second subscript; all these steps are done in parallel, but for the last introduced $T_{i,1}, F_{i,1}$ we have to continue m steps after the $2n$ necessary for priming. In total, we perform $2n+m$ steps.)

$(2, b_i/b_{i+1}, 0)$,
$(2, d_i/d_{i+1}, 0)$, for all $i = n+1, \ldots, (n+1)+(2n+m)-1$.

(In parallel with the previous operations, the counters b_i and d_i increase their subscripts, until reaching the value $3n+m+1$. This is done in all cells 2 at the same time. Simultaneously, e_i increases its subscript in cell 1.)

$(2, b_{3n+m+1}t_is_{i,j}/b_{3n+m+1}r_j, 0)$,
$(2, b_{3n+m+1}f_is'_{i,j}/b_{3n+m+1}r_j, 0)$, for all $1 \leq i \leq n$ and $1 \leq j \leq m$,
$(2, d_i/d_{i+1}, 0)$, for all $i = 3n+m+1, \ldots, (3n+m+1)+nm-1$.

(In the presence of b_{3n+m+1} – and not before – we check the values assumed by clauses for the truth-assignments from each cell 2. We have only one copy of b_{3n+m+1} in each cell, hence we need at most nm steps for this: each clause contains at most n literals, and we have m clauses. In parallel, d increases the subscript, until reaching the value $3n+nm+m+1$.)

$(2, d_{3n+nm+m+i}r_i/d_{3n+nm+m+i+1}, 0)$, for all $i = 1, 2, \ldots, m$.

(In each cell with label 2 we check whether or not all clauses are satisfied by the corresponding truth-assignment. For each clause which is satisfied, we increase by one the subscript of d, hence the subscript reaches the value $3n+nm+2m+1$ if and only if all clauses are satisfied.)

$(2, d_{3n+nm+2m+1}/f$ `yes`$, 1)$.

(If one of the truth-assignments from a cell 2 has satisfied all clauses, then we reach $d_{3n+nm+2m+1}$, which is sent to cell 1 in exchange of the objects `yes` and f.)

$(2, \texttt{yes}/\lambda, 0)$.

(In the next step, the object `yes` leaves the system, signaling the fact that the formula is satisfiable. In cell 1, the counter e will increase one more step its subscript, but after that it will remain unchanged – it can leave cell 1 only in the presence of f, but this object was already moved to cell 2.)

$(1, e_{3n+nm+2m+2} f\ \texttt{no}/\lambda, 2)$,
$(2, \textbf{no}/\lambda, 0)$.

(If the counter e reaches the subscript $3n + nm + 2m + 2$ and the object f is still in cell 1, then the object `no` can be moved to a cell 2, randomly chosen, and from here it exits the system, signaling that the formula is not satisfiable.)

From the previous explanations, one can see that, starting with the multiset $w(\gamma)$ added to cell 2, which is the input cell, the system correctly answers the question whether or not γ is satisfiable. The duration of the computation is polynomial in terms of n and m: the answer `yes` is sent out in step $3n + nm + 2m + 2$, while the answer `no` is sent out in step $3n + nm + 2m + 4$. This concludes the proof. □

The antiport rules from the previous construction are of weight at most 3, but the weight can be reduced to two, at the expense of some slowdown of the system. For instance, instead of the rule $(1, e_{3n+nm+2m+2} f\ \texttt{no}/\lambda, 2)$ we can consider the rules $(1, e_{3n+nm+2m+2} f/h, 0)$, $(1, h\ \texttt{no}/\lambda, 2)$, where h is a new object. We can proceed in the same way with the rules $(2, b_{3n+m+1} t_i s_{i,j}/b_{3n+m+1} r_j, 0)$, $(2, b_{3n+m+1} f_i s'_{i,j}/b_{3n+m+1} r_j, 0)$, for $1 \leq i \leq n$ and $1 \leq j \leq m$, but in this way instead of at most nm steps for finding the satisfied clauses we will need at most $2nm$ steps. The details are left to the reader.

4 Final Remarks

We have proven that by adding the membrane division feature to tissue P systems (with the communication done by antiport rules of a small weight) we can solve **NP**-complete problems in polynomial time. We exemplify this possibility with `SAT` problem.

It remains as a research topic to consider the same extension for other types of systems, for instance, for cell P systems with symport/antiport rules, or for neural P systems (with states associated with cells and multiset rewriting rules for processing the objects. The difficulty in the case of cell P systems with symport/antiport comes from the fact that only the skin membrane can communicate with the environment; on the other hand, the skin membrane cannot be divided, hence we need exponentially many objects for communication with inner membranes, and such objects should be brought in from the environment. In turn, neural P systems with the maximal use of rules and replicated communication are already known to be able to solve **NP**-complete problems in polynomial time; the challenge now is not to use replication). In spite of these difficulties, we expect results similar to the above one also these cases.

Another problem which remains open is to consider tissue P systems with the communication using only symport rules.

Acknowledgements. The support of this research through the project TIC2002-04220-C03-01 of the Ministerio de Ciencia y Tecnología of Spain, cofinanced by FEDER funds, is gratefully acknowledged.

References

[1] P. Frisco, H.J. Hoogeboom, Simulating counter automata by P systems with symport/antiport. In Gh. Păun, G. Rozenberg, A. Salomaa, C. Zandron, eds., *Membrane Computing. International Workshop WMC 2002, Curtea de Argeş, Romania, Revised Papers. Lecture Notes in Computer Science* 2597, Springer-Verlag, Berlin, 2003, 288–301.

[2] Gh. Păun, *Computing with Membranes: An Introduction.* Springer-Verlag, Berlin, 2002.

[3] M. Pérez-Jiménez, A. Romero-Jiménez, F. Sancho-Caparrini, *Teoría de la Complejidad en Modelos de Computatión Celular con Membranas.* Editorial Kronos, Sevilla, 2002.