An optimal solution to the SAT problem with tissue P systems

David Orellana-Martín, Luis Valencia-Cabrera, Mario J. Pérez-Jiménez

Research Group on Natural Computing
Department of Computer Science and Artificial Intelligence
Universidad de Sevilla
Avda. Reina Mercedes s/n, 41012 Sevilla, Spain
E-mail: {dorellana, lvalencia, marper}@us.es

Summary. In the framework of membrane computing, several frontiers of efficiency have been found with respect to the resources that different families of P systems take to solve a decision problem. Each of these frontiers provides a new way to tackle the **P** versus **NP** problem. In this sense, optimal frontiers are needed in order to separate close variants of P systems. In a previous work, an efficient solution to SAT was given in the framework of P systems from $\mathcal{TDC}(3)$. In this work, we will provide an optimal solution to the SAT problem in terms of length of the rules.

Key words: Membrane Computing, tissue P systems, symport/antiport rules, SAT problem.

1 Introduction

One of the most prolific fields within the framework of Membrane Computing is computational complexity theory. The search for frontiers of efficiency has produced several results in terms of correspondences between classic computational complexity classes and membrane computing complexity classes. In particular, several results with tissue P systems [5] have been achieved. Recognizer tissue P systems were introduced in [6]. In [3], it was demonstrated that the complexity class of tissue P systems from $\mathcal{TDC}(1)$ are non-efficient systems through the dependency graph technique. In [6], an efficient solution to the SAT problem is given by means of a family of tissue P systems from $\mathcal{TDC}(5)$. In this work, we present an optimal solution for SAT with a family of tissue P systems from $\mathcal{TDC}(2)$. This implies that $\mathbf{NP} \cup \mathbf{co-NP} \subseteq \mathbf{PMC}_{\mathcal{TDC}(2)}$. However, this last result is not new, since in [9] it was presented a solution to HAM-CYCLE, a well-known **NP**-complete problem, with a family of P systems from $\mathcal{TDC}(2)$. The novelty of this work is to present an efficient SAT solver, given the relevance of the problem. The paper is organized as follows. In Section 2, some prerequisites about languages and sets are given. The next section is devoted to introduce tissue P systems, and the complexity classes associated to them. In Section 4, an efficient solution to SAT by means of tissue P systems from $\mathcal{TDC}(2)$ is given, and in the next section an overview of the computations is given. Finally, the work finishes with some conclusions.

2 Preliminaries

An *alphabet* Γ is a non-empty set whose elements are called *symbols*. A *string* u over $\Gamma$is an ordered finite sequence of symbols; that is, a mapping from a natural number $n \in \mathbb{N}$ onto Γ The number $n$is called the *length* of the string u and it is denoted by $|u|$ The empty string (with length 0) is denoted by λ. The set of all strings over an alphabet Γ is denoted by Γ^*. A *language* over Γ is a subset of Γ^*.

A *multiset* over an alphabet Γ is an ordered (Γ, f) where $f$is a mapping from Γ onto the set of natural numbers $\mathbb{N}$. The *support* of a multiset $m = (\Gamma, f)$ is defined as $supp(m) = \{x \in \Gamma \mid f(x) > 0\}$. A multiset is finite (respectively, empty) if its support is a finite (respectively, empty) set. We denote by $\emptyset$ the empty multiset. Let $m_1 = (\Gamma, f_1), m_2 = (\Gamma, f_2)$ be multisets over Γ, then the union of m_1 and m_2, denoted by $m_1 + m_2$, is the multiset (Γ, g) where $g(x) = f_1(x) + f_2(x)$ for each $x \in \Gamma$. The difference of m_1 and m_2, denoted by $m_1 \setminus m_2$, is the multiset (Γ, g) where $g(x) = f_1(x) - f_2(x)$ for each $x \in \Gamma$. We denote by $M(\Gamma)$ the set of all multisets over Γ.

The Cantor pairing function is used to encode two natural numbers into a single one, and is defined as follows: given $x, y \in \mathbb{N}, \langle x, y\rangle = \frac{(x+y+1)(x+y)}{2} + y$

3 Tissue P systems with symport/antiport rules

Definition 1. *A recognizer tissue P system with symport/antiport rules and division rules 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. *Γ, Σ and $\mathcal{E}$ are finite alphabets such that $\Sigma, \mathcal{E} \subseteq \Gamma$ and $\Sigma \cap \mathcal{E} = \emptyset$.*
2. *$\mathcal{M}_1, \ldots, \mathcal{M}_q$ are multisets over $\Gamma \setminus (\Sigma \cup \mathcal{E})$.*
3. *$\mathcal{R}$ is a set of rules of the following types:*
   (a) Communication rules: $(i, u/v, j)$, for $i, j \in \{0, 1, \ldots, q\}, i \neq j, u, v \in M(\Gamma), |u| + |v| > 0$.
   (b) Division rules: $[\,a\,]_i \to [\,b\,]_i[\,c\,]_i$, for $i \in \{1, \ldots, q\}, a, b, c \in \Gamma$.
4. *$i_{in} \in \{1, \ldots, q\}$ and $i_{out} = env$.*

A recognizer tissue P system with symport/antiport rules and division rules $\Pi = (\Gamma, \Sigma, \mathcal{E}, \mathcal{M}_1, \ldots, \mathcal{M}_q, \mathcal{R}, i_{in}, i_{out})$ of degree $q \geq 1$ can be viewed as a se of q cells, labelled by $1, \ldots, q$ with an environment labelled by 0 such that: (a)

$\mathcal{M}_1, \dots \mathcal{M}_q$are the multisets of objects initially placed in the q cells of the system; (b) $\mathcal{E}$ is the set of objects located initially placed in the environment of the system, all of them appearing in an arbitrary number of copies; and (c) $i_{in} \in \{1, \dots, q\}, i_{out} = env$ represent distinguished cells where objects of the instance are introduced and which will encode the output of the system, respectively.

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 multiset v are sent from region j to region i. The length of the communication rule $(i, u/v, j)$ is defined as $|u| + |v|$, that is, the total number of objects which appear in the rule.

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 communication rules of the form $(i, u/v, j)$, with $i = 0$ or $j = 0$.

When applying a division rule $[\,a\,]_i \rightarrow [\,b\,]_i[\,c\,]_i$, cell i is duplicated into two new cells with the same label. Object a dissapears and an object b is created in the first new cell and an object c is created in the second new cell. The rest of the objects within the cell i are duplicated in the two new cells.

The rules of a system like the one above are used in a non-deterministic maximally parallel manner as it is customary in Membrane Computing. At each step, all communication rules which can be applied will be applied in a maximally parallel way (at each step we apply a multiset of rules which is maximal, no further applicable rule can be added). Only a single division rule can be applied to each cell. If a division rule is applied to a cell, we say that this cell is is "blocked", and no communication rules can be applied to that cell in that computational step.

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

Let us fix a tissue P system with symport/antiport rules Π. We say that configuration $\mathcal{C}_i$ yields configuration $\mathcal{C}_{i+1}$ in one transition step, denoted $\mathcal{C}_i \Rightarrow_{\Pi} \mathcal{C}_{i+1}$, if we can pass from $\mathcal{C}_i$ to $\mathcal{C}_{i+1}$ by applying the rules from $\mathcal{R}$ following the previous remarks. A computation of Π is a (finite or infinite) sequence of configurations such that: (a) the first term of the sequence is an initial configuration of the system; (b) each non-initial configuration of the sequence is obtained from the previous configuration by applying the rules of the system in a maximally parallel manner with the restrictions previously mentioned; and (c) if the sequence is finite (called halting computation), then the last term of the sequence is a halting configuration. All computations start from an initial configuration and proceed as stated above;only halting computations give a result, which is encoded by the objects present in the environment in the halting configuration. Given that they are *recognizer* P systems, all computations halt and return the same output. We denote a recognizer membrane system Π with the input multiset m in the input region as $\Pi + m$.

3.1 Complexity classes associated to tissue P systems

Let $\mathcal{R}$ be a class of recognizer membrane systems. We say that $\mathbf{PMC}_{\mathcal{R}}$ is the class of problems solvable efficiently in a uniform way by means of a family of recognizer membrane systems from $\mathcal{R}$. The class of recognizer tissue P systems with symport/antiport rules with length at most k and division rules is denoted by $\mathcal{TDC}(k), k \geq 1$. For more information about computational complexity theory in the framework of membrane computing, we refer the reader to [7, 8].

4 A solution to SAT in $\mathcal{TDC}(2)$

In this section, an efficient solution to the SAT problem by means of a family of P systems will cell division and symport/antiport rules of length at most 2 is presented.

For each pair of natural numbers $n, p \in \mathbb{N}$, we will consider the recognizer tissue P system with cell division and symport/antiport rules

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

of degree $np + 3$ defined as follows:

(a) $\Gamma = \Sigma \cup \mathcal{E} \cup \{\texttt{yes}, \texttt{no}, \alpha, \beta_0, \gamma_0\} \cup \{c_j \mid 1 \leq k \leq p\} \cup$
$\{\alpha_k \mid 0 \leq k \leq np-1\} \cup \{a_{i,j} \mid 1 \leq i \leq n, 1 \leq j \leq p\} \cup$
$\{T_{i,j}, F_{i,j} \mid 1 \leq i \leq n, 1 \leq j \leq p\} \cup$
$\{x_{i,j,k}, \overline{x}_{i,j,k} \mid 1 \leq i \leq n, 1 \leq j \leq p, 0 \leq k \leq np\}$

(b) $\mathcal{E} = \{\alpha_k \mid np \leq k \leq 2np+2\} \cup \{\beta_k \mid 1 \leq k \leq 2np+4\} \cup$
$\{\gamma_k \mid 1 \leq k \leq 2np+5\} \cup \{x_{i,j,k}, \overline{x}_{i,j,k} \mid 1 \leq i \leq n, 1 \leq j \leq p, np+1 \leq k \leq 2np\}$

(c) $\Sigma = \{x_{i,j,0}, \overline{x}_{i,j,0} \mid 1 \leq i \leq n, 1 \leq j \leq p\}$

(d) $\mathcal{M}_i = \emptyset$ for $1 \leq i \leq np$, $\mathcal{M}_{np+1} = \{\texttt{yes}, \texttt{no}, \beta_0, \gamma_0\}$, $\mathcal{M}_{np+2} = \{a_{i,j} \mid 1 \leq i \leq n, 1 \leq j \leq p\} \cup \{c_j \mid 1 \leq j \leq p\} \cup \{\alpha\}$, $\mathcal{M}_{np+3} = \{\alpha_0\}$.

(e) The set of rules $\mathcal{R}$ is the following:

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

$$\left.\begin{array}{l} [\, a_{i,j} \,]_{np+2} \rightarrow [\, T_{i,j} \,]_{np+2} [\, F_{i,j} \,]_{np+2} \\ (np+2, T_{i,j} F_{i,j'}, /\lambda, env) \end{array}\right\} \text{for } 1 \leq i \leq n, 1 \leq j, j' \leq p$$

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

$$\left.\begin{array}{l}(np+1, x_{i,j,0}/\lambda, i+n\cdot(j-1))\\ (np+1, \overline{x}_{i,j,0}/\lambda, i+n\cdot(j-1))\end{array}\right\}\text{for } \begin{array}{l}1\le i\le n,\\ 1\le j\le p\end{array}$$

$$\left.\begin{array}{l}[\,x_{i,j,k}\,]_{i+n\cdot(j-1)} \to [\,x_{i,j,k+1}\,]_{i+n\cdot(j-1)}[\,x_{i,j,k+1}\,]_{i+n\cdot(j-1)}\\ [\,\overline{x}_{i,j,k}\,]_{i+n\cdot(j-1)} \to [\,\overline{x}_{i,j,k+1}\,]_{i+n\cdot(j-1)}[\,\overline{x}_{i,j,k+1}\,]_{i+n\cdot(j-1)}\end{array}\right\}\text{for } \begin{array}{l}1\le i\le n,\\ 1\le j\le p,\\ 0\le k\le np-1\end{array}$$

$$\left.\begin{array}{l}(i+n\cdot(j-1), x_{i,j,k}/x_{i,j,k+1}, env)\\ (i+n\cdot(j-1), \overline{x}_{i,j,k}/\overline{x}_{i,j,k+1}, env)\end{array}\right\}\text{for } \begin{array}{l}1\le i\le n,\\ 1\le j\le p,\\ np\le k\le 2np-1\end{array}$$

3. Rules to check which clauses are satisfied by the truth assignment

$$\left.\begin{array}{l}(np+2, T_{i,j}/x_{i,j,2np}, i+n\cdot(j-1))\\ (np+2, F_{i,j}/\overline{x}_{i,j,2np}, i+n\cdot(j-1))\end{array}\right\}for \begin{array}{l}1\le i\le n,\\ 1\le j\le p\end{array}$$

$$\left.\begin{array}{l}(np+2, c_j x_{i,j,2np}/\lambda, env)\\ (np+2, c_j \overline{x}_{i,j,2np}/\lambda, env)\end{array}\right\}for\ 1\le i\le n, 1\le j\le p$$

$[\,\alpha_k\,]_{np+3} \to [\,\alpha_{k+1}\,]_{np+3}\,[\,\alpha_{k+1}\,]_{np+3}\,\}\,for\ 0\le k\le np-1$
$(np+3, \alpha_{np+k}/\alpha_{np+k+1}, env)\ for\ 0\le k\le np+1$
$(np+1, \beta_k/\beta_{k+1}, env)\ for\ 0\le k\le 2np+3$
$(np+1, \gamma_k/\gamma_{k+1}, env)\ for\ 0\le k\le 2np+4$
$(np+2, \alpha/\alpha_{2np+2}, np+3)$
$(np+2, \alpha_{2np+2}c_j/\lambda, env)\ for\ 1\le j\le p$

4. Rules to return a negative answer
$(np+1, \beta_{2np+4}, \gamma_{2np+5}/\lambda, np+3)$
$(np+1, \texttt{no}/\beta_{2np+4}, np+3)$
$(np+3, \texttt{no}/\lambda, env)$

5. Rules to return a positive answer
$(np+1, \beta_{2np+4}/\alpha_{2np+2}, np+2)$
$(np+1, \alpha_{2np+2}\texttt{yes}/\lambda, env)$

(f) $i_{in} = np+1$ and $i_{out} = env$

5 An overview of the computations

In this section, we will explain a brief overview of the computations.

Let φ be a propositional logic formula in conjunctive normal form, where $Var(\varphi) = \{x_1,\dots,x_n\}$. Then, φ is of the form $\varphi = C_1 \wedge \dots \wedge C_p$, where C_j is a clause such that $C_j = l_{1,j} \vee \dots l_{p_j,j}, l_{i,j} \in \{x_i, \neg x_i\}$. The pair (cod, s) for this family is $cod(\varphi) = \{x_{i,j} \mid x_i \in C_j\} \cup \{\overline{x}_{i,j} \mid \neg x_i \in C_j\}$, and $s = \langle n,p\rangle$.

For that, let us remember that the input is an instance of the SAT problem. Let φ the input formula with n variables and p clauses. The tissue P system that will give the answer to the instance is $\Pi(\langle n,p\rangle) + cod(\varphi)$.

5.1 Generation stage

The first $2np$ steps will be devoted to generate both the 2^n possible truth assignments. For this, rules from **1** and **2** are used. On the one hand, With rules from **1**, 2^{np} cells with label $np+2$ with different truth assignments. The second rule is fired in order to remove incompatible truth assignments in the following sense: if two different assignments are given to the same variable, then the corresponding objects are sent to the environment. In this way, the incompatible assignments are removed and the remaining objects will be equivalent to a valid truth assignment.

On the other hand, rules from **2** will create 2^{np} copies of each object from $cod(\varphi)$. Being $l_{i,j,0} \in cod(\varphi)$, at the end of the stage, there will be 2^{np} cells $i+n\cdot(j-1)$ $(1 \leq i \leq n, 1 \leq j \leq p)$ with an object $l_{i,j,2np}$ in the corresponding cell.

Besides these rules, 2^{np} cells with label $np+3$ with an object α_{2np+2} will be generated with rules from **3**. This stage will take $2np$ steps.

5.2 First checking stage

In this stage, clauses validated by the truth assignments are going to be analyzed. For this, rules from **3** are used. In particular, in the first step of the stage objects $T_{i,j}$ and $F_{i,j}$ are interchanged with objects $x_{i,j,2np}$ and $\overline{x}_{i,j,2np}$, respectively. Then, objects $x_{i,j,2np}$ and $\overline{x}_{i,j,2np}$ will represent that the literal l_i makes true clause C_j. Then, if an object $x_{i,j,2np}$ or $\overline{x}_{i,j,2np}$ exists in a cell labelled by $np+2$, it will "remove" the corresponding object c_j, sending it to the environment. When the stage is over, cells labelled by $np+2$ will contain only the objects c_j such that the clause C_j has not been satisfied by the corresponding truth assignment. This stage takes 2 steps.

5.3 Second checking stage

The satisfiability of the input formula φ will be analyzed in this stage. Last two rules from **3** will be used. In the first step, object α from cells labelled by $np+2$ will be interchanged with object α_{2np+2} from cells labelled by $np+3$. Object α_{2np+2} in cells labelled by $np+3$ is a mark to know if there are remaining objects c_j. With the last rule, any remaining object c_j will "remove" object α_{2np+2} from the corresponding cell $np+2$. Therefore, if a truth assignment does not satisfy the whole formula φ, object α_{2np+2} will not be present in the corresponding cell $np+2$. This stage takes 2 steps.

5.4 Output stage

The output stage starts at the $2np+5$ step, and takes 4 steps in the negative case and 2 steps in the affirmative case.

- *Affirmative answer*: In this case, there will exist a cell labelled by $np+2$ that will have an object α_{2np+2}, as it represents a truth assignment that makes true the input formula φ. With the application of the first rule from **5**, it will be interchanged by the object β_{2np+4} from cell $np+1$. At the same time, object γ_{2np+5} will go inside cell labelled by 1. Since object β_{2np+4} has been moved from cell $np+2$, they will not interact with each other. In the next step, as object α_{2np+2} is present in the cell $np+1$, it will send object `yes` to the environment. Then, the computation ends.
- *Negative answer*: In this case, all objects α_{2np+2} will be in the environment, as there is no truth assignment such that it makes true all clauses from φ. Therefore, in the previous stage there was at least one object c_j in each cell labelled by $np+2$ and it will send object α_{2np+2} to the environment. In the first step of this stage, object γ_{2np+5} will go into cell labelled by $np+1$. In the next step, objects β_{2np+4} and γ_{2np+5} will interact and be sent to a cell labelled by $np+3$. Following that, object `no` will be interchanged with the object β_{2np+4}, and then object `no` will be sent to the environment. The computation stops here.

Theorem 1. $\texttt{SAT} \in \mathbf{PMC}_{\mathcal{TDC}(2)}$

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

- Every system of the family $\mathbf{\Pi}$ is a recognizer P system from $\mathcal{TDC}(2)$.
- The family $\mathbf{\Pi}$ is polynomially uniform by Turing machines because for each $n, p \in \mathbb{N}$, the rules of $\Pi(\langle n, p\rangle)$ of the family are recursively defined from $n, p \in \mathbb{N}$, and the amount of resources needed to build an element of the family is of a polynomial order in n and p, as shown below:
  - Size of the alphabet: $4n^2p^2 + 10np + p + 17 \in \Theta(n^2p^2)$.
  - Initial number of cells: $np + 3 \in \Theta(np)$.
  - Initial number of objects in cells: $np + p + 6 \in \Theta(np)$.
  - Number of rules: $2n^2p^2 + np^2 + 11np + p + 7 \in \Theta(n^2p^2)$.
  - Maximal number of objects involved in any rule: $2 \in \Theta(1)$.
- The pair (cod, s) of polynomial-time computable functions defined fulfills the following: for each input formula φ of SAT problem, $s(\varphi)$ is a natural number, $cod(\varphi)$ is an input multiset for the system $\Pi(s(\varphi))$, and for each $n \in \mathbb{N}$, $s^{-1}(n)$ is a finite set.
- The family $\mathbf{\Pi}$ is polynomially bounded: indeed, for each input formula φ of SAT problem, the deterministic P system $\Pi(s(\varphi)) + cod(\varphi)$ takes exactly $np+7$ steps, being n the number of variables in φ and p its number of clauses.
- The family $\mathbf{\Pi}$ is sound with regard to (X, cod, s): for each formula φ, if the computation of $\Pi(s(\varphi)) + cod(\varphi)$ is an accepting computation, then φ is satisfiable.
- The family $\mathbf{\Pi}$ is complete with regard to (X, cod, s): for each input formula φ such that it is satisfiable, the computation of $\Pi(s(\varphi)) + cod(\varphi)$ is an accepting computation.

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

6 Conclusions

In the framework of membrane computing, as mentioned above, we search for frontiers of efficiency as new ways to attack the **P** versus **NP** problem. For that, it is necessary to have both results of non-efficient classes and efficient classes of membrane systems. The thinner the frontier, the easier would be to try to adapt an efficient solution from the efficient model to the non-efficient model.

The SAT problem is the best-known **NP**-complete problem, since it was the first problem to be demonstrated to be **NP**-complete [1, 4], and therefore is one of the most studied problems to solve the conjecture. SAT solvers are systems implemented to give an answer to an input SAT instance [2]. Implementations on high-performance computing platforms could be useful to simulate this solution since it could provide a good alternative to current industrial SAT solvers. More precisely, this solution only requires of 2 objects at most in communication rules, that could be an advantage in the implementation. As future work we will continue studying the efficiency of different variants of recognizer membrane systems.

Acknowledgements

This work was supported in part by the research project TIN2017-89842-P, co-financed by Ministerio de Economía, Industria y Competitividad (MINECO) of Spain, through the Agencia Estatal de Investigación (AEI), and by Fondo Europeo de Desarrollo Regional (FEDER) of the European Union.

References

1. S. Cook. The complexity of theorem proving procedures. *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, 1971, pp. 151–158. (doi:10.1145/800157.805047).
2. C.P. Gomes, H.A. Kautz, A. Sabharwal, B. Selman. Satisfiability Solvers. *Handbook of Knowledge Representation* (2008).
3. R. Gutiérrez-Escudero, M.J. Pérez-Jiménez, M. Rius-Font. Characterizing tractability by tissue-like P systems. *Membrane Computing, 10th International Workshop, WMC 2009, Curtea de Arges*, Romania, August 24-27, 2009, Revised Selected and Invited Papers. Lecture Notes in Computer Science, **5957** (2010), 289-300 (doi:10.1007/978-3-642-11467-0_21) A preliminary version in Gh. Paun, M.J. Pérez-Jiménez, A. Riscos (eds.) *Proceedings of the Tenth Workshop on Membrane Computing*, Curtea de Arges (Romania), August 24-27, 2009, pp. 269-281.
4. R.M. Karp. Reducibility Among Combinatorial Problems. In R.E. Miller; J.W. Thatcher (eds.) *Complexity of Computer Computations* (1972). New York: Plenum. pp. 85-103. ISBN 0-306-30707-3.
5. C. Martín-Vide, J. Pazos, Gh. Păun, A. Rodríguez Patón. A new class of symbolic abstract neural nets: tissue P systems. *Lecture Notes in Computer Science*, **2387** (2002), pp. 290-299

6. Gh. Păun, M.J. Pérez-Jiménez, A. Riscos-Núñez. Tissue P systems with cell division. *International Journal of Computers, Communications & Control*, **3**, 3 (2008), pp. 295-303
7. M.J. Pérez-Jiménez. An Approach to Computational Complexity in Membrane Computing. In G. Mauri, Gh. Păun, M.J. Pérez-Jiménez, G. Rozenberg, A. Salomaa (eds.) *Membrane Computing. WMC 2004. Lecture Notes in Computer Science*, 3365 (2005). Springer, Berlin, Heidelberg, pp. 125-148.
8. M.J. Pérez–Jiménez. A Computational Complexity Theory in Membrane Computing. In Gh. Păun, M.J. Pérez-Jiménez, A. Riscos-Núñez, G. Rozenberg, A. Salomaa (eds.) *Membrane Computing. WMC 2009. Lecture Notes in Computer Science*, 5957 (2010). Springer, Berlin, Heidelberg, pp. 85-109.
9. A.E. Porreca, N. Murphy, M.J. Pérez-Jiménez. An Optimal Frontier of the Efficiency of Tissue P Systems with Cell Division. In M.Á. Martínez-del-Amor, Gh. Păun, I. Pérez-Hurtado, F.J. Romero-Campero (eds.) *Proceedings of the Tenth Brainstorming Week on Membrane Computing*, Report RGNC 01/2012, 2012, pp. 141-166.