Int. J. of Computers, Communications & Control, ISSN 1841-9836, E-ISSN 1841-9844
Vol. V (2010), No. 4, pp. 540-550

Solving `Vertex Cover` Problem by Means of Tissue P Systems with Cell Separation

C. Lu, X. Zhang

Chun Lu
Key Laboratory of Image Processing and Intelligent Control
Department of Control Science and Engineering
Huazhong University of Science and Technology
Wuhan 430074, Hubei, People's Republic of China
E-mail: luchun.et@gmail.com(corresponding author)

Xingyi Zhang
School of Computer Science and Technology, Anhui University
Hefei 230601, Anhui, People's Republic of China
E-mail: xyzhanghust@gmail.com

Abstract: Tissue P systems is a computing model in the framework of membrane computing inspired from intercellular communication and cooperation between neurons. Many different variants of this model have been proposed. One of the most important models is known as tissue P systems with cell separation. This model has the ability of generating an exponential amount of workspace in linear time, thus it allows us to design cellular solutions to NP-complete problems in polynomial time. In this paper, we present a solution to the `Vertex Cover` problem via a family of such devices. This is the first solution to this problem in the framework of tissue P systems with cell separation.

Keywords: Membrane Computing, Tissue P System, Cell Separation, `Vertex Cover`

1 Introduction

Membrane computing is an emergent branch of natural computing, which is inspired by the structure and the function of living cells, as well as the organization of cells in tissues, organs and other higher order structures. The devices in membrane computing, called *P systems*, provide distributed parallel and non-deterministic computing models. Since Gh. Păun introduced the P system in [10], this area has received important attention from the scientific community, such as computer scientists, biologists, formal linguists and complexity theoreticians.

In the last years, many different models of P systems have been proposed (a comprehensive bibliography can be found in [14]). The most studied variants are the *cell-like* models of P systems, where membranes are hierarchically arranged in a tree-like structure. Various models of cell-like P systems have been successfully used to design solutions to NP-complete problems in polynomial time (see [4]). These solutions are obtained by generating an exponential amount of workspace in polynomial time and using parallelism to check simultaneously all the candidate solutions. In general, cell division, cell creation and cell separation are the three efficient ways to obtain exponential workspace in polynomial time, thus obtaining three corresponding variants of P systems: *cell division*, where the new workspace is generated by membrane division, *cell creation*, where the new membranes are created from objects, and *cell separation*, where the new workspace is generated by membrane separation. It has been proved that all of the three models can efficiently solve NP-complete problems, but technically they are pretty different in the way of designing solutions.

Another interesting class of P systems is known as *tissue P systems*, where membranes are placed in the nodes of a graph. This variant has two biological inspirations (see [6]): intercellular communication and cooperation between neurons. The common mathematical model of these two mechanisms is a net of processors dealing with symbols and communicating these symbols along channels specified in advance, based on symport/antiport rules [9]. Tissue P systems can also efficiently solve NP-complete problems provided that some ingredients are added into such systems, as in the case of cell-like P systems. The first attempt in this respect is to consider cell division in tissue P systems, yielding *tissue P systems with cell division* [12]. In this model, the two new cells generated by a division rule have exactly the same objects except for at most a pair of different objects. This model was shown to efficiently solve NP-complete: SAT [12], 3-coloring [1], Subset Sum [2], Vertex Cover [3], etc.

Recently, another class of tissue P systems is proposed based on cell separation, that is, *tissue P systems with cell separation*, and a polynomial-time solution to the NP-complete problem SAT is given in [8]. In this model, the contents of the two new cells evolved from a cell by separation rules can be different, thus leading to a significant difference in specific techniques for designing solutions to concrete NP-complete problems. In this paper, we shall explore the possibility of using such a model to solve another NP-complete problem–`Vertex Cover`. Specifically, a family of tissue P systems with cell separation is constructed, in which each system can solve all instances of `Vertex Cover` of a fixed size in a polynomial time. Although the `Vertex Cover` problem has been considered in the framework of other models in membrane computing (for instance, cell-like P systems with active membrane, tissue P systems with cell division, and so on), here the first solution for this problem is presented in the framework of tissue P systems with cell separation.

The paper is organized as follows: in Sections 2 and 3 preliminaries and the definition of tissue-like P systems with cell separation are recalled, respectively. In Section 4, recognizer tissue P systems are briefly described. A polynomial-time solution to `Vertex Cover` problem is presented in Section 5, including a short overview of the computation and of the necessary resources. Finally, some conclusions and new open research lines are presented.

2 Preliminaries

An *alphabet*, Σ, is a finite and non-empty set of abstract symbols. An ordered sequence of symbols is a *string*. Let Σ be a (finite) alphabet; then Σ^* is the set of all strings over Σ. The number of symbols in a string u is the *length* of the string, and it is denoted by $|u|$. As usual, empty string (with length 0) is denoted by λ. The set of strings of length n built with symbols from the alphabet Σ is denoted by Σ^n and $\Sigma^* = \cup_{n \geq 0} \Sigma^n$.

Let A be a (finite) set, $A = \{a_1, \cdots, a_n\}$. Then a finite multiset m over A is a function $f : A \to \mathbb{N}$. If $m = (A, f)$ is a multiset then its *support* is defined as $supp(m) = \{x \in A \,|\, f(x) > 0\}$. The size of the multiest m is $|m| = \Sigma_{x \in A} f(x)$. A multiset is empty (resp. finite) if its support is the empty set (resp. finite).

A multiset m over A can also be represented by any string x that contains exactly $f_m(a_i)$ symbols a_i for all $1 \leq i \leq n$, e.g., by $a_1^{f(a_1)} a_2^{f(a_2)} \ldots a_k^{f(a_k)}$. Thus, superscripts indicate the multiplicity of each element, and if $f(x) = 0$ for any $x \in A$, then this element is omitted.

We suppose that the reader is already familiar with the basic notions and the terminology of P systems. For details, see [11].

3 Tissue P Systems with Cell Separation

According to the first works on tissue P systems [5, 6] the membrane structure did not change along the computation. A new model based on the cell-like model of *tissue P systems with cell separation* is presented in [7]. The biological inspiration of them is clear: alive tissues are not *static* network of cells, since membrane fission generates new cells in a natural way.

Formally, a *tissue P system with cell separation* of initial degree $q \geq 1$ is a construct

$$\Pi = (\Gamma, O_1, O_2, w_1, \ldots, w_q, \mathcal{E}, \mathcal{R}, i_o),$$

where:

1. Γ is the *alphabet* of *objects*, $\Gamma = O_1 \cup O_2$, $O_1, O_2 \neq \emptyset$, $O_1 \cap O_2 = \emptyset$;
2. $w_1, \ldots, w_q$ are strings over Γ, describing the multisets of objects placed in the cells of the system at the beginning of the computation;
3. $\mathcal{E} \subseteq \Gamma$ is the set of objects present in the environment in arbitrarily copies each;
4. $\mathcal{R}$ is a finite set of rules of the following forms:

   (a) $(i, u/v, j)$, for $i, j \in \{0, 1, 2, \ldots, q\}, i \neq j$, $u, v \in \Gamma^*$;
   Communication rules; $1, 2, \cdots, q$ identify the cells of the system, 0 is used as the label of the environment. This rule $(i, u/v, j)$ can be applied over two cells i and j such that u is contained in cell i and v is contained in cell j. The application of this rule means that the objects of the multisets represented by u and v are interchanged between the two cells;

   (b) $[a]_i \rightarrow [O_1]_i [O_2]_i$, where $i \in \{1, 2, \ldots, q\}$ and $a \in \Gamma$;
   Separation rules; under the influence of object a, the cell with label i is separated into two cells with the same label; at the same time, the object a is consumed; the objects from O_1 are placed in the first cell, those from O_2 are placed in the second cell;

5. $i_o \in \{0, 1, 2, \ldots, q\}$ is the output region.

Rules are used in the non-deterministic maximally parallel manner as customary in membrane computing. In each step, all cells which can evolve must evolve in a maximally parallel way (in each step a multiset of rules which is maximal is applied, no further rule can be added). This way of applying rules has only one restriction: when a cell is separated, the separation rule is the only one which is applied for that cell in that step; the objects inside that cell do not evolve by means of communication rules. The daughter cells will participate to the interaction with other cells or with the environment by means of communication rules in the next step, if they are not separated once again. Their labels precisely identify the rules which can be applied to them.

A sequence of transitions which starts from the initial configuration is called a computation with respect Π. A computation is completed only if it halts and the computations give a result, and result is the multiset of objects present in region i_o in the halting configuration.

4 Recognizer Tissue P Systems with Cell Separation

NP-completeness has been usually studied in the framework of *decision problems*. Let us recall that a decision problem is 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.

The notions from classical *computational complexity theory* are adapted for membrane computing to study the computing efficiency for solving decision problems. *Recognizer tissue P systems* are introduced in [12] for tissue P systems with the same idea of *recognizer P systems* introduced into cell-like P systems [13].

A recognizer tissue P system with cell separation of degree $q \geq 1$ is a construct

$$\Pi = (\Gamma, O_1, O_2, \Sigma, w_1, \ldots, w_q, \mathcal{E}, \mathcal{R}, i_{in}, i_o)$$

where:

- $(\Gamma, O_1, O_2, w_1, \dots, w_q, \mathcal{E}, \mathcal{R}, i_o)$ is a tissue P system with cell separation of degree $q \geq 1$ (as defined in the previous section).
- The working alphabet Γ has two distinguished objects `yes` and `no`, at least one copy of them present in some initial multisets $w_1, \dots, w_q$, but not present in $\mathcal{E}$.
- Σ is an (input) alphabet strictly contained in Γ.
- $i_{in} \in \{1, \dots, q\}$ is the input cell.
- The output region i_o is the environment.
- All computations halt.
- If $\mathcal{C}$ is a computation of Π, in the last step of the computation either the object `yes` or the object `no` (but not both) have to be send out to the environment.

The computations of the system Π with input $w \in \Sigma^*$ start from a configuration of the form $(w_1, w_2, \dots, w_{i_{in}}w, \dots, w_q; \mathcal{E})$, that is, after adding the multiset w to the contents of the input cell i_{in}. We say that the multiset w is *recognized* by Π if and only if the object `yes` is sent to the environment, in the last step of the corresponding computation. We say that $\mathcal{C}$ is an accepting computation (respectively, rejecting computation) if the object `yes` (respectively, `no`) appears in the environment associated to the corresponding halting configuration of $\mathcal{C}$.

Definition 1. A decision problem $X = (I_X, \theta_X)$ is solvable in polynomial time by a family of recognizer tissue P systems $\Pi = \{\Pi(n) \mid n \in \mathbb{N}\}$ with cell separation, if the following holds:

- The family Π is *polynomially uniform* by Turing machines, that is, there exists a deterministic Turing machine constructing $\Pi(n)$ from $n \in \mathbb{N}$ in polynomial time.
- There exists a polynomial-time coding (cod, s) form I_X to Π such that:
  - for each instance $u \in I_X$, $s(u)$ is a natural number and $cod(u)$ is an input multiset of the system $\Pi(s(u))$;
  - the family Π 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, moreover, it performs at most $p(|u|)$ steps;
  - the family Π 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 $\theta_X(u) = 1$;
  - the family Π 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.

We denote by $\mathbf{PMC}_{TS}$ the set of all decision problems which can be solved by means of recognizer tissue P systems with cell separation in polynomial time.

5 A Solution to the `Vertex Cover` Problem

The vertex cover of a non-directed graph is a subset of its vertices such that for each edge of the graph at least one of its endpoints belongs to that subset. The size of the vertex cover is the number of vertices in the subset. The `Vertex Cover` problem considered in this paper is formulated as follows: given a non-directed graph, $G = (V, E)$, and a natural number $k \leq |V|$, determine whether or not G has a vertex cover of size at most k.

We shall prove that `Vertex Cover` can be solved in linear time (in the number of nodes and edges of the graph) by a family of recognizer tissue-like P systems with cell separation. We construct a family $\Pi = \{\Pi(\langle n,m,k\rangle) \mid n,m,k \in \mathbb{N}\}$ where each system of the family will process every instance u of the problem given by a graph with n vertices and m edges, and by a size k of the vertex cover (that is, $s(u) = \langle n,m,k\rangle$, where $\langle a,b\rangle = \frac{(a+b)(a+b+1)}{2} + a$ and $\langle a,b,c\rangle = \langle\langle a,b\rangle,c\rangle$. In order to provide a suitable encoding of these instances, we will use the objects A_{ij}, with $1 \le i < j \le n$, to represent the edges of the graph, and we will provide $cod(u) = \{A_{ij} \mid 1 \le i < j \le n \wedge (v_i,v_j) \in E\}$ as the initial multiset for the system.

With an instance u of the `VC` problem, the system $\Pi(s(u))$ with input $cod(u)$ decides that instance by a brute force algorithm, implemented in the following four stages:

- *Generation Stage*: The initial cell labeled by 2 is separated into two new cells; the separations are iterated until a cell has been produced for each possible candidate solution.
- *Pre-checking Stage*: After obtaining all possible subsets of vertices encoded in cells labeled by 2, this stage only select the subsets of size k.
- *Checking Stage*: For each of these subsets, it is checked if there exists an edge of the graph for which none of its endpoints is in the subset.
- *Output Stage*: The system sends to the environment the right answer according to the results of the previous stage.

$\Pi(\langle n,m,k\rangle) = (\Gamma(\langle n,m,k\rangle), \Sigma(\langle n,m,k\rangle), w_1, w_2, \mathcal{R}(\langle n,m,k\rangle), \mathcal{E}(\langle n,m,k\rangle), i_{in}, i_0)$, for each $n,m,k \in \mathbb{N}$. The family Π contains the following systems:

- $\Gamma(\langle n,m,k\rangle) = O_1 \cup O_2$,

$$\begin{aligned}
O_1 &= \{c_{i,j}, A_{i,j}, z_{i,j}, P_{i,j} \mid 1 \le i < j \le n\} \cup \{j_i \mid 1 \le i \le 2n+1\}\\
&\cup \{A_i, B_i, B_i', C_i', T_i, F_i' \mid 1 \le i \le n\} \cup \{d_i \mid 1 \le i \le n+1\}\\
&\cup \{D_{i,j} \mid 1 \le i,j \le n\} \cup \{a_{1,i}, b_{1,i}, d_{1,i}, g_i, h_i, l_i, e_i \mid 1 \le i \le n-1\}\\
&\cup \{a_i \mid 1 \le i \le 5n+m+\lceil \lg n\rceil + 9\} \cup \{a_{2,i}, b_{2,i}, d_{2,i} \mid 2 \le i \le n-1\}\\
&\cup \{a_{i,j,k}, b_{i,j,k}, d_{i,j,k} \mid 1 \le i < j \le n, 1 \le k \le n-1\}\\
&\cup \{C_{i,j}, B_{i,j} \mid 1 \le i \le n, 1 \le j \le m\} \cup \{L_i \mid 1 \le i \le m + \lceil \lg n\rceil + 7\}\\
&\cup \{P_i \mid 1 \le i \le m + \lceil \lg n\rceil + 6\} \cup \{H_i \mid 1 \le i \le \lceil \lg m\rceil + 1\}\\
&\cup \{G_i \mid 1 \le j \le \lceil \lg n\rceil + 1\} \cup \{b, z, f_1, y, s, E_0, E_1, E_2, T, N, yes, no\},\\
O_2 &= \{c_{i,j}', A_{i,j}', z_{i,j}'\} \mid 1 \le i < j \le n\} \cup \{T_i', F_i \mid 1 \le i \le n\} \cup \{y', z', f'\}.
\end{aligned}$$

- $\Sigma(\langle n,m,k\rangle) = \{c_{i,j}, A_{i,j}, A_{i,j}' \mid 1 \le i < j \le n\}$.
- $w_1 = a_1 a_{1,1} g_1 a_{i,j,1} \texttt{yes no}$.
- $w_2 = c_{i,j} A_{i,j} A_1$.
- $\mathcal{R}(\langle n,m,k\rangle)$ is the set of rules:

  1. **Separation rule:**
     $r_1 \equiv [s]_2 \to [O_1]_2[O_2]_2$.
  2. **Communication rules:**
     $r_{2,i} \equiv (1, a_i/a_{i+1}, 0)$ for $1 \le i \le 5n+m+\lceil \lg n\rceil + 8$;
     $r_{3,i,j,k} \equiv (1, a_{i,j,k}/b_{i,j,k}, 0)$ for $1 \le i < j \le n, 1 \le k \le n-1$;
     $r_{4,i,j,k} \equiv (1, b_{i,j,k}/c_{i,j}^2 d_{i,j,k}^2, 0)$ for $1 \le i < j \le n, 1 \le k \le n-1$;
     $r_{5,i,j,k} \equiv (1, d_{i,j,k}/a_{i,j,k+1}, 0)$ for $1 \le i < j \le n, 1 \le k \le n-2$;

$r_{6,i} \equiv (1, g_i/h_i, 0)$ for $1 \le i \le n-1$;
$r_{7,i} \equiv (1, h_i/l_i^2 A_{i+1}^2, 0)$ for $1 \le i \le n-1$;
$r_{8,i} \equiv (1, l_i/g_{i+1}, 0)$ for $1 \le i \le n-2$;
$r_{9,i} \equiv (1, a_{1,i}/b_{1,i}, 0)$ for $1 \le i \le n-1$;
$r_{10,i} \equiv (1, b_{1,i}/c^2 d_{1,i}^2 e_i^2, 0)$ for $1 \le i \le n-1$;
$r_{11,i} \equiv (1, d_{1,i}/a_{1,i+1}, 0)$ for $1 \le i \le n-2$;
$r_{12,i} \equiv (1, e_i/a_{2,i+1}, 0)$ for $1 \le i \le n-2$;
$r_{13,i} \equiv (1, a_{2,i}/b_{2,i}, 0)$ for $2 \le i \le n-1$;
$r_{14,i} \equiv (1, b_{2,i}/c^2 d_{2,i}^2, 0)$ for $2 \le i \le n-1$;
$r_{15,i} \equiv (1, d_{2,i}/a_{2,i+1}, 0)$ for $2 \le i \le n-2$;
$r_{16,i,j} \equiv (2, c_{i,j} A_{i,j}/z_{i,j} z'_{i,j} A_{i,j} A'_{i,j}, 0)$ for $1 \le i < j \le n$;
$r_{17,i,j} \equiv (2, c_{i,j} A'_{i,j}/z_{i,j} z'_{i,j} A_{i,j} A'_{i,j}, 0)$ for $1 \le i < j \le n$;
$r_{18,i} \equiv (2, cT_i/zz'T_iT_i', 0)$ for $1 \le i \le n-1$;
$r_{19,i} \equiv (2, cT_i'/zz'T_iT_i', 0)$ for $1 \le i \le n-1$;
$r_{20,i} \equiv (2, cF_i/zz'F_iF_i', 0)$ for $1 \le i \le n-1$;
$r_{21,i} \equiv (2, cF_i'/zz'F_iF_i', 0)$ for $1 \le i \le n-1$;
$r_{22} \equiv (2, A_n/T_nF_nf_1f_1's, 0)$;
$r_{23,i} \equiv (2, A_i/T_iF_iyy'zz's, 0)$ for $1 \le i \le n-1$;
$r_{24,i} \equiv (2, y/A_i, 1)$ for $2 \le i \le n$;
$r_{25,i} \equiv (2, y'/A_i, 1)$ for $2 \le i \le n$;
$r_{26} \equiv (2, z/c, 1)$;
$r_{27} \equiv (2, z'/c, 1)$;
$r_{28,i,j} \equiv (2, z_{i,j}/c_{i,j}, 1)$ for $1 \le i < j \le n$;
$r_{29,i,j} \equiv (2, z'_{i,j}/c_{i,j}, 1)$ for $1 \le i < j \le n$;
$r_{30} \equiv (1, z/\lambda, 0)$;
$r_{31} \equiv (1, z'/\lambda, 0)$;
$r_{32,i,j} \equiv (1, z_{i,j}/\lambda, 0)$ for $1 \le i < j \le n$;
$r_{33,i,j} \equiv (1, z'_{i,j}/\lambda, 0)$ for $1 \le i < j \le n$;
$r_{34} \equiv (2, f/j_1d_1, 0)$;
$r_{35} \equiv (2, f'/j_1d_1, 0)$;
$r_{36,i,j} \equiv (2, d_jT_i/D_{i,j}, 0)$ for $1 \le i, j \le n$;
$r_{37,i,j} \equiv (2, d_jT_i'/D_{i,j}, 0)$ for $1 \le i, j \le n$;
$r_{38,i,j} \equiv (2, D_{i,j}/B_id_{j+1}, 0)$ for $1 \le i, j \le n$;
$r_{39,i} \equiv (2, j_i/j_{i+1}, 0)$ for $1 \le i \le 2n$;
$r_{40} \equiv (2, j_{2n+1}d_{k+1}/E_0, 0)$;
$r_{41} \equiv (2, E_0/L_1E_1, 0)$;
$r_{42,i} \equiv (2, L_i/L_{i+1}, 0)$ for $i = 1, \ldots, m + \lceil \lg n \rceil + 6$;
$r_{43} \equiv (2, E_1/P_1E_2, 0)$;
$r_{44} \equiv (2, E_2/G_1H_1, 0)$;
$r_{45,i} \equiv (2, P_i/P_{i+1}, 0)$ for $i = 1, \ldots, m + \lceil \lg n \rceil + 5$;
$r_{46,i} \equiv (2, G_i/G_{i+1}^2, 0)$ for $i = 1, \ldots, \lceil \lg n \rceil$;
$r_{47,i} \equiv (2, H_i/H_{i+1}^2, 0)$ for $i = 1, \ldots, \lceil \lg m \rceil$;
$r_{48,i,j} \equiv (2, A_{i,j}H_{\lceil \lg m \rceil + 1}/P_{i,j}, 0)$ for $1 \le i < j \le n$;
$r_{49,i,j} \equiv (2, A'_{i,j}H_{\lceil \lg m \rceil + 1}/P_{i,j}, 0)$ for $1 \le i < j \le n$;
$r_{50,i} \equiv (2, G_{\lceil \lg n \rceil + 1}B_i/C_i, 0)$ for $i = 1, \ldots, n$;
$r_{51,i} \equiv (2, C_i/C_{i,1}B_{i,1}, 0)$ for $i = 1, \ldots, n$;
$r_{52,i,j} \equiv (2, B_{i,j}/B_{i,j+1}B_i', 0)$ for $i = 1, \ldots, n$ and $j = 1, \ldots, m$;
$r_{53,i,j} \equiv (2, C_{i,j}/C_{i,j+1}C_i', 0)$ for $i = 1, \ldots, n$ and $j = 1, \ldots, m$;
$r_{54,i,j} \equiv (2, B_i'P_{i,j}/\lambda, 0)$ for $1 \le i < j \le n$;

$r_{55,i,j} \equiv (2, C'_j P_{i,j}/\lambda, 0)$ for $1 \leq i < j \leq n$;
$r_{56,i,j} \equiv (2, P_{m+\lceil \lg n \rceil+5} P_{i,j}/N, 0)$ for $1 \leq i < j \leq n$;
$r_{57} \equiv (2, L_{m+\lceil \lg n \rceil+7} P_{m+\lceil \lg n \rceil+6}/T, 0)$;
$r_{58} \equiv (1, b/T, 2)$;
$r_{59} \equiv (1, a_{5n+m+\lceil \lg n \rceil+9} b/N, 2)$;
$r_{60} \equiv (1, T\ \texttt{yes}/\lambda, 0)$;
$r_{61} \equiv (1, N\ \texttt{no}/\lambda, 0)$;

- $\mathcal{E}(\langle n,m,k \rangle) = \Gamma(\langle n,m,k \rangle) - \{yes, no\}$.
- $i_{in} = 2$ is the *input cell.*
- $i_o = 0$ is the *output region.*

We will show that the family $\Pi = \{\Pi(\langle n,m,k \rangle) \mid n,m,k \in \mathbb{N}\}$ defined above is polynomially uniform by Turing machines. To this aim it will be proved that $\Pi(\langle n,m,k \rangle)$ is built in polynomial time with respect to the size parameter n, m and k of instances of `Vertex Cover` problem.

It is easy to check that the rules of a system $\Pi(\langle n,m,k \rangle)$ of the family are defined recursively from the values n, m and k. The necessary resources to build an element of the family are of a polynomial order, as shown below:

- Size of the alphabet: $n^2 + 5mn + 26n + 7m + 4\lceil \lg n \rceil + \lceil \lg m \rceil + 27 \in O(n^2 + mn)$.
- Initial number of cells: $2 \in O(1)$.
- Initial number of objects: $3m + 6 \in O(m)$.
- Number of rules: $5mn + 3n^2 + 26n + 10m + 4\lceil \lg n \rceil + \lceil \lg m \rceil + 6 \in O(n^2 + mn)$.
- Maximal length of a rule: $6 \in O(1)$.

Therefore, a deterministic Turing machine can build $\Pi(\langle n,m,k \rangle)$ in a polynomial time with respect to n, m and k.

5.1 An Overview of the Computation

A family of recognizer tissue P systems with cell separation is constructed in the previous section. In the following, we informally describe how the recognizer tissue P system with cell separation $\Pi(s(\gamma))$ with input $cod(\gamma)$ works. Let us start with the *generation stage*, where all the possible subsets of the vertices of the graph are generated. This stage has several parallel processes, which we describe in several items.

– In the cells with label 2, in the presence of $c_{i,j}$, by the rules $r_{16,i,j}$, $r_{17,i,j}$, the objects $c_{i,j}A_{i,j}$, $c_{i,j}A'_{i,j}$ introduce the objects $z_{i,j}z'_{i,j}A_{i,j}A'_{i,j}$, respectively. In the next step, primed objects and non-primed objects are separated into the new daughter cells with label 2. The objects $z_{i,j}$ and $z'_{i,j}$ in cells with label 2 are exchanged with the objects $c_{i,j}$ in the cell with label 1 by the rules $r_{28,i,j}$ and $r_{29,i,j}$. In this way, the cycle of duplication-separation can be iterated.

– In parallel with the above duplication-separation process, the objects c are used to duplicate the objects T_i, T_i', F_i and F_i' by the rules $r_{18,i} - r_{21,i}$ (in general $T_i(T_i')$ and $F_i(F_i')$ correspond to the values $true$ and $false$ of vertex A_i); the rules r_{26} and r_{27} take care of introducing the object c from the cell with label 1 to cells with label 2.

- In the initial configuration of the system, the cell with label 2 contains an object A_1 (A_i encodes the i-th variable in the propositional formula). The objects T_1, F_1', z, z', y, y' and s are brought in the cell with label 2, in exchange of A_1, by the rule $r_{23,i}$. In the next step they are separated into the new daughter cells with label 2 by separation rule, because $(T_1, F_1') \in O_1$ and $(F_1, T_1') \in O_2$. The object s is used to activate the separation rule r_1, and is consumed during the application of this rule. The objects y and y' are used to introduce A_2 from the cell with label 1, and the process of truth-assignment for variable v_2 can continue. In this way, in $3n-1$ steps, we get 2^n cells with label 2, and each one contains one of the 2^n possible truth-assignments for the n variables.
- In parallel with the operations in the cells with label 2, the objects $a_{i,j,k+1}$ from the cell with label 1 are traded for objects $b_{i,j,k+1}$ from the environment at the step $3k+1$ $(0 \le k \le n-3)$ by the rule $r_{2,i,j,k}$. In the next step, each object $b_{i,j,k+1}$ is traded for two copies of objects $c_{i,j}$ and $d_{i,j,k+1}$ by the rule $r_{3,i,j,k}$. At step $3k+3$ $(0 \le k \le n-3)$, the object $d_{i,j,k}$ is traded for object $a_{i,j,k+2}$ by the rule $r_{4,i,j,k}$. Especially, at step $3n-5$, $a_{i,j,n-1}$ is traded for $b_{i,j,n-1}$ by the $r_{2,i,j,k}$, at step $3n-4$, each copy of object $b_{i,j,n-1}$ is traded for two copies of $c_{i,j}$ by the $r_{4,i,j}$. After step $3n-4$, there is no object $a_{i,j,k}$ appears in the cell with label 1, and the group of rules $r_{3,i,j,k}$ – $r_{5,i,j,k}$ will not be used again. Note that the subscript k of the object $a_{i,j,k}$ grows by 1 in every 3 steps until reaching the value $n-1$, and the number of copies of $a_{i,j,k}$ is doubled in every 3 steps. At step $3k+3$ $(0 \le k \le n-2)$, the cell with label 1 contains 2^{k+1} copies of object $c_{i,j}$. At the same time, we have 2^{k+1} cells with label 2, and each cell with label 2 contains one copy of object $z_{i,j}$ (or $z'_{i,j}$). Due to the maximality of the parallelism of using the rules, each cell with label 2 gets exactly one copy of $c_{i,j}$ from the cell with label 1 by the rules $r_{28,i,j}$ and $r_{29,i,j}$. The object $c_{i,j}$ in cell with label 2 is used for duplication as described above.
- The objects $a_{1,i}$ and $a_{2,i}$ in the cell with label 1 has a similar role as object $a_{i,j,k}$ in cell 1, which introduces appropriate copies of object c for the duplication of objects T_i, T_i', F_i and F_i' by the rules $r_{9,i}$ – $r_{15,i}$. Note that at step $3k+3$ $(0 \le k \le n-2)$, there are $(k+1)2^{k+1}$ copies of object c which, by the maximality of the parallelism of using the rules, ensures that each cell with label 2 gets $k+1$ copies of object c .
- The object g_{i+1} in the cell with label 1 is traded for h_{i+1} from the environment at step $3i+1$ $(0 \le i \le n-3)$ by the rule $r_{6,i}$. In the next step, the object h_{i+1} is traded for two copies of objects l_{i+1} and A_{i+2} by the rule $r_{13,i}$. At the step $3i+3$ $(0 \le i \le n-3)$, the object l_{i+1} is traded for two copies of g_{i+2}, so that the process can be iterated, until the subscript i of g_i reaches $n-1$. At step $3n-5$, object g_{n-1} is traded for h_{n-1} by the rule $r_{6,i}$. At step $3n-4$, each object h_{n-1} is traded for two copies of A_n. After step $3n-4$, no object g_i appears in the cell with label 1, and the group of rules $r_{15,i}$ – $r_{18,i}$ will not be used again. At the step $3i+3$ $(0 \le i \le n-2)$, the cell with label 1 contains 2^{i+1} copies of A_{i+2}, and we have 2^{i+1} cells with label 2, each of them containing one copy of object y or one copy of object y'. Due to the maximality of the parallelism of using the rules, each cell with label 2 gets exactly one copy of A_{i+2} from cell 1 by the rules $r_{24,i}$ and $r_{25,i}$. In this way, the truth-assignment for the vertex A_{i+1} can continue.
- The objects $z_{i,j}$, $z'_{i,j}$, y, y', z and z' in the cell with label 1 are removed by the rules $r_{28,i,j}$, $r_{29,i,j}$, r_{30}, r_{31}.

Note that this non-deterministic generation stage is performed by the successive application of the separation rules, and at the end of the stage the same configuration is always reached. Thus, the system is confluent in this stage and performs $3n+1$ steps.

Now that all the subsets of vertices of the graph are generated, the *pre-checking stage* selects only those of size k. This stage is activated by rules r_{34} and r_{35}, which interchange the object f (or f') of each 2-cell (recall that there are 2^n of them) from the environment, and then each of the latter in each 2-cell introduces an object d_1 and an object j_1 from the environment (recall that there are infinitely many of them).

The objects d_1 and j_1 start two processes of counting in each 2-cell. The first process counts the steps of this stage with counter j_i using rules $r_{39,i}$.

The second process counts the number of vertices in the subset. It is performed using rules $r_{36,i,j}$ and $r_{37,i,j}$, which interchange the objects T_i in the 2-cells by objects B_i (indicating this way that the corresponding vertex has been counted) and increase the counter d_j (the only purpose of the objects D_{ij} is to reduce the length of the rules). Note that this is a non-deterministic process, since the vertex "counted" in each step is chosen in a non-deterministic way. However, as the size of the subsets of vertices is bounded by n, after $2n$ steps of this process, the same configuration is always reached, so the system is also confluent in this stage.

For the counter d_j of a 2-cell to increase, it is necessary and sufficient that in that cell there exist objects B_i left. This means that at the end of the process explained in the previous paragraph, the only 2-cells that contain objects encoding subsets of vertices of size k are those containing the object d_{k+1}. At this moment, those cells also contain the counter j_{2n+1}, which then in two steps cause (using rules r_{40} and r_{41}, and the intermediate object E_0 for rules size reduction) the object d_{k+1} to be interchanged by objects L_1 and E_1 from the environment.

The total number of steps of the *pre-checking stage* is $2n+2$.

The *checking stage* starts now, but before checking if any of the subsets of vertices of size k selected in the previous stage is a vertex cover of the graph, we need some preparation steps. First of all, the objects L_i will be used as a counter, controlled by rules $r_{42,i}$, of the number of steps performed. On the other hand, rule r_{43} introduces another counter P_i, controlled by rules $r_{45,i}$, which runs in parallel, but with a delay of one step. Also, in each 2-cell encoding a subset of vertices of size k objects G_1 and H_1 are introduced by rules r_{43} and r_{44}, and are then multiplied by rules $r_{45,i}$ and $r_{46,i}$ until obtaining n copies of the former and m copies of the latter.

The objects $H_{\lceil \lg m \rceil +1}$ are used by rules $r_{48,i,j}$ and $r_{49,i,j}$ to change into objects P_{ij} encoding the edges of the graph. On the other hand, rules $r_{50,i}$, $r_{51,i}$, $r_{52,i,j}$ and $r_{53,i,j}$ produce, from objects $G_{\lceil \lg n \rceil +1}$ and B_i and by successive interchanges of objects between the 2-cells and the environment, m copies of objects B'_i and C'_i for each and all of the vertices in the subset encoded into the 2-cell.

As the copies of objects B'_i and C'_i are being produced, rules $r_{54,i,j}$ and $r_{55,i,j}$ eliminate from the 2-cell, in a non-deterministic way, edges of the graph (encoded by objects P_{ij}) such that at least one of its endpoints is contained in the subset encoded in the corresponding 2-cell. Once this stage has performed $m+\lceil \lg n \rceil +6$ steps, we are sure that if there is any object P_{ij} left in the 2-cell, then the subset of vertices encoded in that cell is not a vertex cover of the graph, and rule $r_{56,i,j}$ eliminates the counter P_i in an additional step.

The *answer stage* starts at step $5n+m+\lceil \lg n \rceil +9$, when the object $l_{m+\lceil \lg n \rceil +7}$ appears in every 2-cell encoding a subset of vertices of size k. If the counter P has survived in any of these 2-cells, it means that it encoded a vertex cover of the graph, and rule r_{57} interchanges the two counters with an object T from the environment, which is then sent to the 1-cell of the system by rule r_{58}. Then, rules r_{59}, r_{60} and r_{61} control if this cell has received at least one object T from any of the 2-cells of the system. If this is the case, it is detected at step $5n+m+\lceil \lg n \rceil +9$, when an object `yes` is sent to the environment and the system halts. Otherwise, it is detected at step $5n+m+\lceil \lg n \rceil +10$, when an object `no` is sent to the environment and the system halts.

5.2 Main Results

From the discussion in the previous section, the family Π is polynomially bounded, sound and complete with regard to (`VC`,*cod*,*s*). We have the following result:

Theorem 5.1. `Vertex Cover` $\in \mathbf{PMC}_{TS}$.

Corollary 2. $\mathbf{NP} \cup \mathbf{co}-\mathbf{NP} \subseteq \mathbf{PMC}_{TS}$.

Proof: It suffices to make the following observations: the `Vertex Cover` problem is NP-complete, `Vertex Cover`$\in$ $\mathbf{PMC}_{TS}$ and this complexity class is closed under polynomial-time reduction and under complement.

6 Discussion

The main purpose of this paper is to provide a polynomial time solution for `Vertex Cover` problem based on tissue P systems with cell separation. We showed that the membrane separation is an important feature that could hold the power to solving computationally hard problems in polynomial time. Following this direction, it remains as further work to describe classical complexity classes below PSPACE with this framework.

7 Acknowledgements

The authors acknowledge the support of National Natural Science Foundation of China (60674106, 30870826, 60703047, and 60533010), Program for New Century Excellent Talents in University (NCET-05-0612), Ph.D. Programs Foundation of Ministry of Education of China (20060487014), Chenguang Program of Wuhan (200750731262), HUST-SRF (2007Z015A), and Natural Science Foundation of Hubei Province (2008CDB113 and 2008CDB180).

Bibliography

[1] D. Díaz-Pernil, M. A. Gutiérrez-Naranjo, M. J. Pérez-Jiménez, A. Riscos-Núñez. A Linear–time Tissue P System Based Solution for the 3–coloring Problem. *Electronic Notes in Theoretical Computer Science*, Vol. 171, pp. 81–93, 2007.

[2] D. Díaz-Pernil, M. A. Gutiérrez-Naranjo, M. J. Pérez-Jiménez, A. Riscos-Núñez. Solving Subset Sum in Linear Time by Using Tissue P Systems with Cell Division. In: J. Mira, J. R. Alvarez, J. R. Ivarez (Eds.) *2nd International Work-Conference*, IWINAC 2007, *Interplay between natural and artificial computation Lecture Notes in Computer Science*, Vol. 4527, pp. 170–179, 2007.

[3] D. Díaz-Pernil, M. J. Pérez-Jiménez, A. Riscos-Núñez, A. Romero. Computational Efficiency of Cellular Division in Tissue-like Membrane Systems. *Romanian Journal of Information Science and Technology*, Vol. 11(3), pp. 229–241, 2008.

[4] M. A. Gutiérrez-Naranjo, M. J. Pérez-Jiménez, F. J. Romero-Campero. A Linear solution for QSAT with Membrane Creation. *Lecture Notes in Computer Science*, Vol. 3850, pp. 241–252, 2006.

[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*, Vol. 2387, pp. 290–299, 2002.

[6] C. Martín Vide, J. Pazos, Gh. Păun, A. Rodríguez-Patón. Tissue P Systems. *Theoretical Computer Science*, Vol. 296, pp. 295–326, 2003.

[7] L. Pan, T.-O. Ishdorj. P Systems with Active Membranes and Separation Rules. *Journal of Universal Computer Science*, Vol. 10(5), pp. 630–649, 2004.

[8] L. Pan, M. J. Pérez-Jiménez. Efficiency of Tissue P Systems with Cell Separation. In M. A. Martínez-del-Amor, E. F. Orejuela-Pinedo, Gh. Păun, I. Pérez-Hurtado, A. Riscos-Núñez, *Seventh Brainstorming Week on Membrane Computing*, Sevilla, Report RGNC 02/2009, 169–196, 2009.

[9] A. Păun, Gh. Păun. The Power of Communication: P Systems with Symport/Antiport. *New Generation Computing*, Vol. 20(3), pp. 295–395, 2002.

[10] Gh. Păun. Computing with Membranes. *Journal of Computer and System Sciences*, Vol. 61(1), 108–143, 2000.

[11] Gh. Păun. *Membrane Computing, An Introduction*, Springer–Verlag, Berlin, 2002.

[12] Gh. Păun, M. J. Pérez-Jiménez, A. Riscos-Núñez. Tissue P System with Cell Division. In Gh. Păun, A. Riscos-Núñez, A. Romero-Jiménez, F. Sancho-Caparrini (eds.), *Second Brainstorming Week on Membrane Computing*, Sevilla, Report RGNC 01/2004, 380–386, 2004.

[13] M. J. Pérez-Jiménez, A. Romero-Jiménez and F. Sancho-Caparrini, A Polynomial Complexity Class in P Systems Using Membrane Division, In E. Csuhaj-Varjú, C. Kintala, D. Wotschke and Gy. Vaszyl (eds.), *Proceedings of the 5th Workshop on Descriptional Complexity of Formal Systems, DCFS 2003*, pp. 284–294, 2003.

[14] The P System Web Page: http://ppage.psystems.eu

Chun Lu is a Ph.D candidate in Huazhong University of Science and Technology, Wuhan, China. He received his master degree in Systems Engineering from Huazhong University of Science and Technology in 2008. Currently, his main research interests cover membrane computing, neural computing, automata theory and its application.

Xingyi Zhang was born in China on June 6, 1982. He received his doctor degree at Huazhong University of Science and Technology in 2009. Currently, he works in School of Computer Science and Technology, Anhui University. His main research fields are formal language theory and its applications, unconventional models of computation, especially, membrane computing. He has published several scientific papers in international journals.