Applied Mathematics & Information Sciences
An International Journal

http://dx.doi.org/10.12785/amis/080141

Solving Vertex Cover Problem by Tissue P Systems with Cell Division

Tao Song[1,3], *Hongjiang Zheng*[1,2,*] *and Juanjuan He*[3]

[1] School of Information Engineering, Wuhan University of Technology, Wuhan, 430070, Hubei, China
[2] School of Information Engineering, Tarim University, Akesu, 843300, Xinjiang, China
[3] School of Automation, Huazhong University of Science and Technology, Wuhan 430074, Hubei, China

Received: 24 Jun. 2013, Revised: 3 Nov. 2013, Accepted: 5 Nov. 2013
Published online: 1 Jan. 2014

Abstract: Tissue P systems are a class of distributed and parallel computing models investigated in membrane computing, which are inspired from the structure and functioning of communication cells in tissues. Such systems with cell division (corresponding to the mitosis behavior of living cells) can theoretically generate exponential working space in linear time, therefore providing a possible way to solve computational hard problems in feasible time by a space-time trade-off. In this work, we construct a family of tissue P systems with cell division to solve the vertex cover problems, and achieve a linear time solution (with respect to the size of the problems). Furthermore, we prove that the systems are constructed in a uniform manner and work in a confluent way.

Keywords: membrane computing, tissue P system, decision problem, vertex cover problem, linear solution.

1 Introduction

Membrane computing initialed by G. Păun [9] in 2000, which is now known as a hot and new branch in natural computing. In membrane computing, a class of computing models, usually called P systems, are investigated by abstracting ideas from the structure and the functioning of a single cell and tissues, even from human brain. Till now, there are three mainly investigated classes of P systems: cell-like P systems (generally called P systems) [9], tissue-like P systems (called tissue P systems) [2], and neural-like P systems (specified by spiking neural P systems) [10].

Many variants of all the three classes of P systems, as well as their computational properties have been considered [1,3,4,5,6,7,8,15,16,17,18,19]; an overview of membrane computing or P systems can be found in [10] and [12], with up-to-date information available at the membrane computing website (`http://ppage.psystems.eu`). For an introduction to membrane computing, one may consult [12] and [13]. In the present work, we deal with a variant of tissue P systems, called tissue P systems with cell division [11].

Generally speaking, a tissue P system with cell division can be represented by a graph, where cells are placed in the nodes of a directed graph, and the edges correspond to communication channels among the cells. Each cell contains several objects (represented by multiset on a given alphabet), communication rules and division rules. A cell can communicate with other cells (or with the environment) along channels specified in advance, and can also divide into two new cells. The communication between each pair of cells is achieved by the so called communication rules (also called symport/antiport rules), which was proposed in [2]. Specifically, by using symport rules, the objects can be moved across a membrane in one direction, whereas by using antiport rules, the objects may move across a membrane in opposite directions. Inspired from the behavior of mitosis of living cells, a cell can divide into two new cells by using division rules. The two children cells have the same label with their mother cell (the label of the cell precisely identifies the available rules). In each time unit (as in usual P systems, it is assumed a global clock in the system, marking the time for the system), objects can evolve by using the evolution rules in a non-deterministic maximally parallel manner, with the restriction that if a cell can perform the division operation, i.e, a division rule is enabled at that moment,

* Corresponding author e-mail: zhjwhut@gmail.com

then the cell does not participate in any other evolution rules. It indicates that when a cell is dividing, the interaction of the cells will be blocked. Hence, during the division, the rules and the objects in the mother cell remain unchanged and inherit in each of the two child cells. We should stress that, if there are rules that can be applied in a cell, then one of the rules must be applied that is, each cell in the systems works in synchronized mode, but for different cells, they work in a parallel way.

In tissue P systems with cell division, by using cell division rules, we can theoretically generate exponential working space in linear time during the computation, thus provide a possible way to solve computational hard problems in feasible (in polynomial or even in linear) time by a space-time trade-off strategy.

In this work, we construct a family of tissue P systems with cell division, which can solve vertex cover problems in linear time (with respect to the size of the problems). Specifically, for an instance of vertex cover problem γ consisting of n vertices, there exist a such system solving the instance in at most $2n+7$ steps. The construction of the systems is uniform, that is, a family of tissue P systems with cell division can be constructed by a deterministic Turing machine in a polynomial time, which, by receiving the inputs encoding of instances of vertex covering problems, tell us whether or not these instances have positive answers or not.

2 Tissue P Systems with Cell Division

In this section, we start by recalling some basic notions in formal language and automata theory, and then recalling the tissue P systems with cell division introduced in [11].

By $\mathbb{N}$ we denote the set of non-positive integers. For a set V, by $card(V)$ we denote the size of the set V, which is number of elements in V. Let V be an alphabet of symbols, by V^*, we denotes the set of all finite strings of symbols from V; V^+ denotes the set of all non-empty strings over V.

The definition of tissue P systems with cell division is complete, but familiarity with the basic elements of classic tissue P systems is helpful.

A *tissue P system with cell division* of degree $m \geq 1$ is a construct

$$\Pi = (O, E, w_1, \ldots, w_m, R, i_o), \text{ where:}$$

– O is a finite alphabet of *objects*;
– $E \subseteq O$ is the set of objects, which are contained in the environment with arbitrarily copies;
– $m \geq 1$ is the degree of the system, which means the system has m cells (labeled with $1, 2, \ldots, m$); particular, with 0, we refer to the environment of the system);
– $w_1, w_2, \ldots, w_m$ are strings over alphabet O, respecting multisets of objects initially placed in the m cells;
– R is a finite set of rules, where the rules are of the following two forms:
  – $(i, u/v, j)$, for $i, j \in \{1, 2, \ldots, m\}, i \neq j$ and $u, v \in O^*$;
  – $[a]_i \rightarrow [b]_i[c]_i$, $i \in \{1, 2, \ldots, m\}$ and $a, b, c \in O$;
– $i_o \in \{0, 1, 2, \ldots, m\}$ is the label of output cell, whose objects are considered as the computation result when the system halts.

The rules can be applied as follows. The rules of the form $(i, x/y, j)$ are *communication rules*, where $i, j \in \{1, 2, \ldots, m\}$ identify the cells of the system and 0 is the environment. By using the communication rule, the objects represented by multisets x are sent to cell j from cell i, while the objects represented by multisets y are sent to cell i from cell j. If $i = 0$, it means multisets x are sent to cell j from the environment, while multisets y are sent out to the environment; if $j = 0$, it means multisets y are sent to cell i from the environment, while multisets x are sent out to the environment. For a communication rule $(i, x/y, j)$, by $\max\{|x|, |y|\}$, we denote the weight of the rule.

The division rules are of the form $[a]_i \rightarrow [b]_i[c]_i$ ($i \in \{1, 2, \ldots, m\}$). The rule is enabled to use, only when a cell labeled with i contains object a, then it can divide into two new cells labeled by i. In the two newly generated cells, object a evolves to objects b and c, which are placed in the two new cells, respectively.

The rules in the system are used in a non-deterministic maximally parallel manner. In each step (a global clock is assumed to mark the time of the whole system), all cells that can evolve must evolve as in classic P systems with the following restriction: if a cell is using a division rule, the cell can not do any communication at that moment. The objects in the environment are never exhausted (inspired by the fact that in biological systems, external objects can provide sources of objects).

The *configuration* (also called the "state") of a tissue P system with cell division can be described by the objects present in the cells. With this notion, the initial configuration is tuple $(w_1, w_2, \ldots, w_m)$. By using communication rules and cell division rules, we can define *transitions* of the system among configurations. A series of transitions starting from the initial configuration, halting or not, is called a *computation*. If the computation proceeds to a halting configuration, where no rule can be used in any cell, then it is called a successful computation. With any successful computation, a *result* is associated, which is the number of objects in cell labeled with i_o.

In the present work, we do not consider the computational power of tissue P systems with cell division, but the efficiency of the systems by solving computational hard problems. A recognizer tissue P system with cell division can be defined as follows.

$$\Pi' = (O, \Sigma, E, w_1, \ldots, w_m, R, i_o, i_{in}), \text{ where:}$$

• O is the finite alphabet of objects, containing two distinguished objects `yes` and `no`;

- $\Pi = (O, E, w_1, \ldots, w_m, R, i_o)$ is a tissue P system with cell division defined as above with $i_o = 0$;
- $\Sigma \subseteq O/\{\texttt{yes}, \texttt{no}\}$ is a finite alphabet of input objects;
- i_{in} is the input cell.

All the computations of Π' halt, and for any computation C of Π', either object `yes` or `no` (but not both) is present in the environment when the system halts.

The recognizer tissue P systems with cell division can be used to solve decision problems as follows. For an instance of the problem X, the computations of the systems start from a initial configuration by adding the code of the problem $cod(\gamma)$ to the input cell i_{in}. If the computations of the systems finally halt with object `yes` (`no`) present in the environment, then we say the problem has a positive (negative) answer. A computation C is called accepting one (or a rejecting one) if object `yes` (or object `no`) presents in the environment when the system proceeds to the halting configuration of. In the following, we will formally define the solution to a decision problem by recognizer tissue P systems with cell division.

Let $X = (I_X, \Theta_X)$ be a decision problem, where I_X is a set of instances and Θ_X is a predicate over I_X, and $\Pi = \{\Pi(n) \mid n \in \mathbb{N}$ be a family of recognizer tissue P systems with cell division. We say that problem X can be solved in polynomial time by tissue P systems Π, if the following conditions hold.

- The family Π can be constructed in polynomial time by Turing machine.
- There exits a pair of functions (cod, s) over X such that
  - –for any $u \in I_X$, by $cod(u)$, we can input the instance into the system Π, where $s(u) \in \mathbb{N}$;
  - –the family of systems $\Pi(s(u))$ is said to be*sound* with respect to X, cod, s, that is, for any $u \in I_X$, if there exists an accepting computation of $\Pi(s(u))$ with input $cod(u)$, then we have $\Theta_X(u) = 1$;
  - –the family of systems Π is called to be *complete* with respect to X, cod, s, that is, for any $u \in I_X$, if $\Theta_X(u) = 1$, then every computation of system Π_u is an accepting computation;
  - –the family Π is *polynomial bounded*, if there exists a polynomial function $p(n)$ such that, for each $u \in I_X$, all computations in $\Pi(s(u))$ halt in, at most, $p(|u|)$ steps.

The construction of the family of systems is called uniform, if a family of recognizer tissue P systems with cell division are constructed in a polynomial time by a deterministic Turing machine, which give the result of the instance of the problem by receiving as inputs encoding of instances of the problems. The system $\Pi = \{\Pi(n) \mid n \in \mathbb{N}\}$ is confluent, if every computation of Π always gives the same answer with the same input instance of the problem.

3 Solving Vertex Cover Problem in Linear Time

This section is started by recalling the definition of vertex cover problems, and then we construct a family of tissue P systems with cell division in a uniform manner to solve the vertex cover problems.

The Vertex Cover Problem

INSTANCE: A directed graph $\gamma = (V, E)$ with n vertices $v_1, v_2, \ldots, v_n$, and a positive integer $k \leq card(V)$.

QUESTION: Is there a subset $V' \subseteq V$ with $card(V') \leq k$ such that for all $(v_i, v_j) \in E, 1 \leq i, j \leq n$, at least one of v_i, v_j is in V'?

Theorem 1. *Vertex cover problems can be solved in linear time (with respect to the size of the instance) by tissue P systems with cell division.*

Proof Let us consider a vertex cover problem $\gamma = (V, E)$, where $V = \{v_1, v_2, \ldots, v_n\}$ is the set of vertices, and E is the set of edges with elements of the form (v_i, v_j) with $v_i, v_j \in V$ and $i \neq j$.

We can codify γ by function $cod(\gamma) = a_1 a_2 \ldots a_n$, where variable a_i represents vertex v_i. The process of coding γ to $cod(\gamma)$ can be finished in linear steps (with respect to n). In the follosing, we deal with the instance γ by the tissue P systems with cell division $\Pi(s(\gamma))$ with input $cod(\gamma)$, where $s(\gamma) = \langle n, k \rangle = \frac{(n+k)(n+k+1)}{2} + n$.

We construct a family of recognizing tissue P systems with cell division of degree 4

$$\Pi(\langle n, k \rangle) = (O, \Sigma, E, w_1, w_2, w_3, w_4, R, i_{in}), \text{ where:}$$

- $O = \{a_i, a_i', a_i'' \mid i = 1, 2, \ldots, n\} \cup \{c_i \mid i = 1, 2, \ldots, 2n+3\} \cup \{d_i \mid i = 1, 2, \ldots, 2n+5\} \cup \{\texttt{yes}, \texttt{no}, c, t\}$;
- $\Sigma = \{a_i \mid i = 1, 2, \ldots, n\} \cup \{yes, no, c, t\}$;
- $E = O - \{\texttt{yes}, \texttt{no}\}$;
- $w_1 = \texttt{yes}\ \texttt{no}\ a_1 a_2 \ldots a_n, w_2 = \lambda, w_3 = c_1, w_4 = d_1$;
- $R = R_d \cup R_c$ is the set of evolution rules, where R_d is the set of division rules, and R_c is the set of communication rules;
  - $(1) R_d = \{[a_i]_2 \to [a_i']_2 [a_i'']_2 \mid i = 1, 2, \ldots, n\}$;
  - $(2) R_c = \{(1, no/t, 0), (1, t a_1 a_2 \ldots a_n/\lambda, 2)\}$
    $\cup \{(2, a_i'/c, 0) \mid i = 1, 2, \ldots, n\}$
    $\cup \{(3, c_{2n+3}/\lambda, 2), (2, c_{2n+3}/d_{2n+5}, 4)\}$
    $\cup \{(3, c_j/c_{j+1}^2, 0) \mid j = 1, 2, \ldots, 2n+2\}$
    $\cup \{(4, d_h/d_{h+1}^2, 0) \mid h = 1, 2, \ldots, 2n+4\}$
    $\cup \{(2, c_{2n+2} c^{k+1}/\lambda, 0), (1, \texttt{yes}/\lambda, 4), \ (4, \texttt{yes}/\texttt{no}, 0)\}$
    $\cup \{(2, c_{2n+2} a_i'' a_j''/\lambda, 0) \mid (v_i, v_j) \in E, 1 \leq i, j \leq n\}$;
- $i_{in} = 1$ is the input cell.

The necessary resources to build the system are as follows:

- –the size of the alphabet: $7n + 12$;
- –the initial number of cells: 4;
- –the initial number of objects: $n + 4$;

–the number of rules: $n^2+6n+12$;
–the maximal length of a rule: $\max\{3,n+1\}$.

Therefore, $\Pi(\langle n,k\rangle)$ can be built by a deterministic Turing machine in polynomial time with respect to the size parameters n and k of the instance of vertex cover problems.

In the following, we describe how the system $\Pi(\langle n,k\rangle)$ with input $cod(\gamma)$ works. Initially, cell 1 (the input cell) contains objects $w_1 = \texttt{yes}\ \texttt{no}\ a_1\ a_2 \ldots a_n$, so in the first step only the communication rules $(1,\texttt{no}/t,0)$ can be used sending object no to the environment. (Since at that moment the system does not halt, object no can not be taken as the answer to the instance.) In the next step, cell 1 can use the communication rule $(1,ta_1a_2\ldots a_n/\lambda,2)$ to send objects $ta_1a_2\ldots a_n$ to cell 2. In this way, cell 2 will begin to use division rule $[a_i]_2 \to [a_i']_2[a_i'']_2$ $(i=1)$ at step 3.

In the following $2n-1$ steps, cell 2 will use rules $[a_i]_2 \to [a_i']_2[a_i'']_2$ $(i=2,3,\ldots,n)$ in $n-1$ steps (in this way, we can generate all the possible partitions of V in 2^n cells with label 2), and using rules $(2,a_i'/c,0), i=1,2,\ldots,n$ in another n steps. (The single priming objects a_i' correspond to the elements in the subset V' we look for. For any cell 2, the number of objects a_i' corresponding to the number of vertices contained in a partition of V can be counted by the number of object c.) Although this processes are performed non-deterministically, at step $2n+2$, all the rules $[a_i]_2 \to [a_i']_2[a_i'']_2$ and $(2,a_i'/c,0)$ with $i=1,2,3,\ldots,n$ will complete their applications.

Cell 3 continuously uses the rules $(3,c_j/c_{j+1}^2,0), j=1,2,\ldots,n+2$ in $2n+2$ steps, thus it contains 2^n objects c_{2n+3} at step $2n+3$. At that moment, the communication rule $(3,c_{2n+3}/\lambda,2)$ can be used to send one object c_{2n+3} to each of the cells with label 2. The cells with label 2 that contains more than k object c, will use the rule $(2,c_{2n+2}c^{k+1}/\lambda,0)$ to send object c_{2n+3} to the environment (that is the cell 2 corresponding to subset V' of V that contains more than k vertices will not contain object c_{2+3}). Each of cell 2 contains neither a_i' nor a_j' with $(v_i,v_j)\in E, 1\le i,j\le n$ will use the rule $(2,c_{2n+2}a_i''a_j''/\lambda,0)$ to send out the object c_{2n+3}. Hence, the cells with label 2 corresponding to partitions of V that contain more than k vertices or do not contain at least one vertex of every edge in E will have no object c_{2n+3} inside. This process will complete in one step, since each cell 2 only contains one c_{2n+3}. In other words, at step $2n+4$, the cells with label 2 of the partitions of V having less than k vertices and covering at least one vertices of each edge in E can hold object c_{2n+3}. If there exists no such cell 2, the system halts with object no presenting in the environment, thus gives a negative answer to the instance. In this case, the system halts in $2n+4$ steps.

If there exists at least one cell 2 having object c_{2n+3} (this means the instance has a positive answer), the communication rule $(2,c_{2n+3}/d_{2n+5},4)$ can be used at step $2n+5$, by which at least one object c_{2n+3} appears in cell 4. In $2n+4$ steps, cell 4 continuously uses the rule $\{(4,d_h/d_{h+1}^2,0) \mid h=1,2,\ldots,2n+4\}$ to generate d_{2n+5}. With object c_{2n+3}, cell 4 can use the rules $(2,c_{2n+2}c^{k+1}/\lambda,0),(1,\texttt{yes}/\lambda,4)$ and $(4,\texttt{yes}/\texttt{no},0)$. As results, object yes is sent out to the environment and object no comes into cell 4 when the system halts, therefore giving a positive answer to the instance of the problems. In this case, the system halts in $2n+7$ steps.

It is easy to check that the family of systems $\Pi\langle n,k\rangle$ satisfy that (i) all the computations halt; (ii) for each instance of the problem, if the answer of the problem is positive, then the object yes will be present in the environment when the system halts; if the answer of the problem is negative, then object no will present in the environment when the system halts. So, the family of P systems $\Pi\langle n,k\rangle$ are sound and complete. The systems will halt in at most $2n+7$ steps, so they are also linear bounded. Also, we can easily obtain the construction of the family of systems is uniform and the systems work in a confluent way.

As explained above, we can conclude that the family of P systems $\Pi(\langle n,k\rangle)$ with cell division is a uniform solution for vertex cover problem. □

4 Conclusion

Tissue P systems with cell division can generate exponential working space by using cell division rules in linear steps, thus can solve computational hard problems by a space-time trade off strategy. In this work, a uniform linear solution for vertex cover problem is achieved by means of tissue P systems with cell division.

For future work, there are many interesting open problems to consider, such that obtaining solutions in feasible (polynomial or linear) time to more complex decision problems, as well as solutions to **PSPACE** problems.

Acknowledgment

This work is supported National Natural Science Foundation of China (51268051, 61074169, 61033003, 911300 34, 61100145 and 61272071), Ph.D. Programs Foundation of Ministry of Education of China (20100142110072), Fundamental Research Funds for the Central Universities (2011 TS005 and 2011TS006), and Natural Science Foundation of Hubei Province (2011CDA027).

References

[1] T. O. Ishdorj, A. Leporati, L. Pan, X. Zeng, X. Zhang, Deterministic Solutions to QSAT and Q3SAT by Spiking Neural P Systems with Pre-Computed Resources, Theor. Comput. Sci., 411 (2010).

[2] C. Martin-Vide, J. Pazos, G. Păun, A. Rodriguez-Patón, Tissue P Systems. Theor. Comput. Sci., **296**, 2 (2003).
[3] L. Pan, D. P. Daniel, M. J. Pérez-Jiménez, Computation of Ramsey Numbers by P System with Active Membranes. Int. J. Found. of Comput. Sci., **22**, 1 (2011).
[4] L. Pan, G. Păun, Spiking Neural P Systems with Anti-Spikes. Int. J. Comput. Commun., 3 (2009).
[5] L. Pan, G. Păun, Spiking Neural P Systems: An Improved Normal Form. Theor. Comput. Sci., 411 (2010).
[6] L. Pan, G. Păun, M. J. Pérez-Jiménez, Spiking Neural P Systems with Neuron Division and Budding. Sci. China Inform. Sci., **54**, 8 (2011).
[7] L. Pan, X. Zeng, X. Zhang, Time-Free Spiking Neural P Systems. Neural Comput., 23 (2011).
[8] L. Pan, M. J. Pérez-Jiménez, Computational Complexity of Tissue-like P Systems. J. Complexity. **26**, 3 (2010).
[9] G. Păun, Computing with Membranes. J. Comput. Syst. Sci., **61**, 1 (2000).
[10] G. Păun, Membrane Computing. An Introduction. Berlin, Springer-Verlag, (2002).
[11] G. Păun, M. J. Perez-Jimenez, A. Riscos-Nunez, Tissue P Systems with Cell Division. Int. J. Comput. Commun., **3**, 3 (2008).
[12] G. Păun, G. Rozenberg, A. Salomaa (eds.), Handbook of Membrane Computing. Oxford, Oxford University Press, (2010).
[13] G. Păun, G. Rozenberg, A Guide to Membrane Computing. Theor. Comput. Sci., **287**, 1 (2002).
[14] G. Rozenberg, A. Salomaa (eds.), Handbook of Formal Languages. Berlin: Springer-Verlag, (1991).
[15] J. Wang, H. J. Hoogeboom, L. Pan, G. Păun, M. J. Pérez-Jiménez, Spiking Neural Systems with Weights. Neural Comput., **22**, 10 (2010).
[16] X. Zhang, J. Wang, L. Pan, A Note on the Generative Power of AxonSystems. Int. J. Comput. Commun., **4**, 1 (2009).
[17] X. Zhang, X. Zeng, L. Pan, On String Languages Generated by Asynchronous Spiking Neural P Systems. Theor. Comput. Sci., 410 (2009).
[18] X. Zhang, Y. Jiang, L. Pan, Small Universal Spiking Neural P Systems with Exhaustive Use of Rules. Int. J. Comput. Commun., **7**, 5 (2010).
[19] X. Zhang, B. Luo, X. Fang, L. Pan, Sequential Spiking Neural P Systems with Exhaustive Use of Rules, BioSystems, 108 (2012).

Tao Song received his Ph.D degree from Huazhong University of Science and Technology in 2013. He is now working as a post-doctor in Huazhong University of Science and Technology. His research interests include DNA computing, DNA encoding and membrane computing.

Hongjiang Zheng received his Master degree from Wuhan University of Technology in 2008. He is a Ph.D student in Wuhan University of Technology. He is a now lecturer with Tarim University, Akesu. His research interests include cooperative communications and wireless networks and computational theory.

Juanjuan He received Bachelor degree from Huazhong University of Science and Technology in 2008. She is recently a Ph.D student in Huazhong University of Science and Technology. Her research interests include membrane computing, and membrane algorithm.