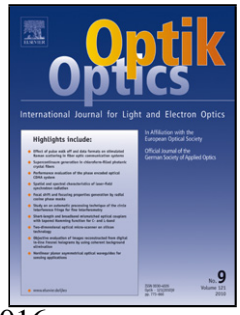

Title: A p system for Hamiltonian cycle problem

Author: Ping Guo Yunlei Dai Haizhu Chen

PII: S0030-4026(16)30637-4
DOI: http://dx.doi.org/doi:10.1016/j.ijleo.2016.06.016
Reference: IJLEO 57802

To appear in:

Received date: 11-4-2016
Accepted date: 1-6-2016

Please cite this article as: Ping Guo, Yunlei Dai, Haizhu Chen, A p system for Hamiltonian cycle problem, Optik - International Journal for Light and Electron Optics http://dx.doi.org/10.1016/j.ijleo.2016.06.016

A p system for Hamiltonian cycle problem

Ping Guo[a], Yunlei Dai[a], Haizhu Chen[b]

[a] College of Computer Science, Chongqing University, Chongqing 400030, China
[b] Department of Software Engineering, Chongqing College of Electronic Engineering, Chongqing 401331, China

Abstract. P system is a new kind of distributed parallel computing model. In the P system, objects in each membrane can follow the evolution of the maximum parallelism principle, so we can solve NP-hard problem in polynomial time. In this paper, we design a family of P system with to judge whether there is a Hamiltonian cycle and find all Hamiltonian cycle in an undirected graph, and then an instance is given to illustrate the feasibility and effectiveness of our designed P systems.

Keywords: P system, Hamiltonian cycle problem, Membrane Computing, Natural Computing

1. Introduction

Membrane computing is a branch of natural computing which abstract computing models from the architecture and the functioning of living cells, as well as from the organization of cells in tissues, organs (brain included) or other higher order structures such as colonies of cells. The computing models in the framework of membrane computing are usually called P systems, which are distributed and parallel computing models. Many variants of P systems, including cell-like P systems [1-2], tissue-like P systems [3-4] and neural-like P systems [5-6], have been investigated and most of them are proved to be universal and efficient. It has been proved that membrane computing has the same equivalent computing power as Turing machine.

Until now, many kinds of P system have been proposed to solve NP-complete problems, such as SAT [9-13], HPP [14-15], Subset [16], Knapsack Problem [17], optimization problem [18-20]. There are two main ways for P systems solving NP-complete problem: the semi-uniform way, which associates with each instance of the problem one P system solving it, and the uniform way, which associates with each possible size of the instances of the problem one P system that can solve all instances of that size [21].

Based on cell-like P systems which are one kind of common systems in membrane computing, Ref. [14] present an algorithm for deterministically deciding HPP in polynomial time by a uniform family of P systems with separation rules, this algorithm could just determine whether there is a Hamiltonian path between two specified nodes. Ref. [15] propose an asynchronous P system that solves the Hamiltonian cycle problem with n node in $O(n!)$ sequential steps, but it only answer "TRUE" or "FALSE" for an input graph. In this paper, we designed a family of P system in a uniform way to not only determine whether there is a Hamiltonian cycle path but also find the complete solution. This P system is constituted by a series of membranes and they are nested one by one. The vertices and edges in an undirected graph will be sent to these membranes. By applying the designed rules to evolve objects, Hamiltonian cycle problem can be final solved. The organization of the rest part of this paper is as follows: section 2 introduces the foundation of P systems, section 3 describes the parallel computing method for solving Hamiltonian cycle problem and section 4 proposes our P systems with the multi-membrane structure and rules with priority designed. In section 5, we give an instance to show how to solve Hamiltonian cycle problem in our P systems. The conclusions are drawn in the final section.

2. Foundations

Our work in this paper is based on cell-like P systems, so we give the basic concepts about cell-like P systems firstly.

The structure of cell-like P systems is illustrated in Fig.1. As suggested by Fig.1, the structure is a hierarchically arranged set of membranes which are usually identified by labels from a given set and contained in a distinguished external membrane. If a membrane does not contain any other membrane, it is called elementary. The membrane that contains all the other membranes is referred as the skin. Each membrane determines a region delimited from above by it and from below by the membranes placed directly inside, if any exists.

Formally, a cell-like P system (of degree $m \geq 1$) can be defined as form [7, 8]:

$$\Pi = (O, \mu, \omega_1, \omega_2 \quad \ldots \omega_m, R_1, R_2, \cdots, R_m, i_o) \tag{1}$$

Where,

(i) O is the alphabet. Each symbol represents one kind of objects in the systems, O^* is finite and non-empty multiset over O, where λ is empty string, $O^+ = O^* - \{\lambda\}$.

(ii) μ is a membrane structure with m membrane, labeled by 1, 2, $\cdots$,m.

(iii) $\omega_i (1 \leq i \leq m)$ is a string over O and represents the multiset of objects placed in member i.

(iv) $R_1, R_2, \cdots, R_m$ are finite sets of the evolution rules over O associated with the regions 1, 2, $\cdots$, m of μ. The rules in R_i ($1 \le i \le m$) are of the form $U \rightarrow V|_{\alpha}$, with $a \in O$, $U \in O^+$, $V = V'$ or $V = V'\delta$, $V' \in (O \times Tar)^*$, and $Tar = \{here; out; in_j | 1 \le j \le m\}$. *Here* means V is remained in the same region, *out* means V goes out of the region, and in_j means V goes to inner membrane j. Object a is a promoter in rule $U \rightarrow V|_a$, this rule can only be applied in the presence of object a.

(v) i_o is output region of the system and it saves the final result.

In each membrane, rules are applied according to the following principles:

(1) Non-determinism. Suppose n rules compete for the reactants which can only support m ($m < n$) rules to be applied, then the m rules are chosen non-deterministically.

(2) Maximal parallelism. All of the rules that can be applied must be applied simultaneously.

3 Hamiltonian cycle problem and the parallel algorithm

Hamiltonian cycle problem is an old question in graph theory. Actually, many problems could be converted to Hamiltonian cycle problem, such as the design of material transportation routes and the design of bus line. Consequently, research on the parallel computing methods for Hamiltonian cycle problem has great significance. Because objects in cell-like P systems can be evolved in accordance with the principle of maximal parallelism, the characteristics of its high parallelism makes it possible to solve NP-complete problems in polynomial time. In this section, we firstly give the definition and the current research status of Hamiltonian cycle problem, and then give an algorithm which is suitable for solving Hamiltonian cycle problem in P systems.

The HPP is a well-known NP-complete problem, which asks whether or not a given graph $G=(V, E)$ contains a Hamiltonian path. Let G be an undirected graph, if there is a path P traversed from v_1 to v_2 and pass exactly once through each vertex of G, then path P is a Hamiltonian path from v_1 to v_2 in G. And if $v_1=v_2$, path P is called a Hamiltonian cycle. So, Hamiltonian cycle problem (short called HCP) is to find all Hamiltonian cycles in G. Fig.2 gives an instance of undirected graph, let V_1 be the starting vertex and ending vertex, its Hamiltonian cycles are $\{V_1 \rightarrow V_2 \rightarrow V_3 \rightarrow V_4 \rightarrow V_5 \rightarrow V_1\}$, $\{V_1 \rightarrow V_2 \rightarrow V_3 \rightarrow V_5 \rightarrow V_4 \rightarrow V_1\}$, $\{V_1 \rightarrow V_2 \rightarrow V_5 \rightarrow V_3 \rightarrow V_4 \rightarrow V_1\}$.

There have been some algorithm to solve HCP problem [15], this paper presents an algorithm can not only answer if Hamiltonian cycle problem is solvable (also known as HCP judging problem), but also can give the all solution when it is solvable (namely all-HCP). The Algorithm constructs all legal paths in polynomial time by parallel membrane creation, which is corresponding to path construction. Illegal Hamiltonian paths will be deleted while constructing the paths, which are corresponding to path detection. At last, if the paths still exist after path detection, then HCP problem is solvable, and the paths represent all Hamiltonian cycle paths, otherwise, Hamiltonian cycle path doesn't exist. The PAHCP (Parallel algorithm for Hamiltonian cycle problem) can be described as Algorithm PAHCP.

Algorithm: PAHCP

Input: undirected graph $G=(V, E)$

Output: all Hamiltonian cycle paths or *No* (represent Hamiltonian cycle path doesn't exist in graph G)

Steps:

(1) Initialization: input the edges and vertices of the undirected graph.

(2) Path construction: construct all legal paths in parallel, the steps of constructing one legal path P as follows:

1) Add the starting vertex to the path *P*.

2) If there is an edge from the last vertex on path P to a vertex has not been added, add it to path P.

3) Repeat step 2) until no vertex could be added to path *P*.

4) If each vertex of graph G has been added to path P and there is an edge from the last vertex on path P to the starting vertex, add the starting vertex to path P.

(3) Path detection: delete illegal Hamiltonian cycle paths while constructing the paths.

1) If there is no edge from the last vertex on path *P* to a vertex has not been added, delete path *P*.

2) If each vertex of graph G has been added to path P and there is no edge from the last vertex on path P to the starting vertex, delete path P.

(4) Output: output all Hamiltonian cycle paths or *No*.

End

4. The design of P Systems Π_{HT}

In this section, we design a P system Π_{HT} for solving all-HCP based on the algorithm PAHCP which discussed in subsection 3.

4.1 The Definition of Π_{HT}

According to the previous definition of eq. (1), the like-cell P systems Π_{HT} can be defined as:

$$\Pi_{HT} = (O, \mu, \omega, R, \rho, i_o) \quad (2)$$

Where,

(1) O is a finite and non-empty alphabet of objects, it includes:

- objects which represent vertices in the undirected graph: $\{a_i, e_i, u_i, p_i, q_i, f_i | 1 \leq i \leq n\}$

$\{a_1, a_2, ..., a_n\}$ represents the set of vertices of graph, simultaneously, $\{e_1, e_2, ..., e_n\}$, $\{u_1, u_2, ..., u_n\}$, $\{p_1, p_2, ..., p_n\}$, $\{q_1, q_2, ..., q_n\}$, $\{f_1, f_2, ..., f_n\}$ also represents the set of vertices of graph. a_i, e_i, u_i, q_i and f_i represent the ith vertices in the graph ($1 \leq i \leq n$).

- some special objects:

-y, w, z: the existence of y means that all vertices have been visited; w, v represents that Hamiltonian path has been found.

-λ: means that there is no object in the membrane

-τ: when τ is in membrane i, the thickness of membrane i will be increased, objects can't get in and out of membrane i.

-δ: membrane i is in membrane j, when object δ is created in membrane i, membrane i will be dissolved and all objects in membrane j will be sent to membrane j.

Besides the objects mentioned above, there are other objects which are used to control the application of the rules, and we do not interpret them here.

(2) μ is the initial structure of our P system as shown in Fig.3, and the structure will be changed during the evolution of the system.

(3) $\omega = \{\omega_1, \omega_2\}$ is a collection of multisets in the initial configuration, where: $\omega_1 = \phi$, $\omega_2 = \phi$.

(4) $R = R^{I} \cup R^{C} \cup R^{D}$, where, R^{I} is used for initialization, R^{C} for path construction and R^{D} for path detection.

Based on the algorithm PAHCP, the procedure of applying the rules in Π_{HT} is:

- Initialization (see subsection 4.2.1)
- Path construction (see subsection 4.2.2)
- Path detection (see subsection 4.2.3)

In R, there are 5 type rules:

(a) $(u \rightarrow v, k)$,

(b) $(u \rightarrow v|_w, k)$

(c) $(u \rightarrow v_1(\, v_2, in/out), k)$

(d) $(u \rightarrow v_1(\, v_2, in/out)|_w, k)$

(e) $(u \rightarrow v_1[v_2]|_w, k)$

The k indicates the priority; the smaller value k is set, the higher the priority of the rule is. And the w indicates rule can only be applied in the presence of multiset w. Rule of type (a) means object u evolves to object v. Rule of type (b) means object u evolves to object v when multiset w appears in the membrane. Rule of type (c) means object u evolves to object v_1, at the same time generates object v_2 which is sent into or outer the membrane. Rule of type (d) means object u evolves to object v_1 when multiset w appears in the membrane, at the same time generates object v_2 which is sent into or outer the membrane. Rule of type (e) means when multiset w appears in the membrane, object u evolve to object v_1, at the same time create a new sub-membrane and send v_2 to it.

(5) The final result can be found in membrane 1 when the whole system halts, i_0 corresponds to membrane 1 in Π_{HT}.

4.2 The rules in Π_{HT}

4.2.1 Initialization

In P system Π_{HT}, the rules in R^I are applied to input the following objects into the skin membrane: 1) a_i, represent the vertices of graph G; 2) e, represent the end of inputting vertices; 3) f_i, represent the starting and ending vertex of the Hamiltonian cycle. The rules in R^I associated with the initialization are ($1 \le i \le n$):

r_1: $(a_i \rightarrow (\, sa_i\, , in\,), 2)$ r_2: $(e \rightarrow (e, in), 2)$ r_3: $(f_i \rightarrow (\, t^2 a_i f_i, in\,), 2)$

r_4: $(ts \rightarrow \lambda, 2)$ r_5: $(a_i^2 e \rightarrow p_i b, 2)$

4.2.2 Path construction

(1) Visit each vertex of graph

If we want to search for the Hamiltonian cycle path, we need to visit all nodes exactly once from the starting vertex, and finally return to the starting vertex. Firstly, the length of path P is 0, the length will be increased 1 when a vertex has been visited. The rules in R^C associated with the process are ($1 \le i \le n$, $1 \le j \le n$, $1 \le k \le n$):

r_1: $([ba_i]_k \rightarrow [e_i c[t\, p_i]_{k+1}]_k, 1)$ r_2: $(u_i \rightarrow u_i\, (a_i, in)|_c, 2)$ r_3: $(a_i \rightarrow a_i(a_i, in)|_c, 2)$

r_4: $(s \rightarrow s\, (s, in)|_c, 2)$ r_5: $(f_i \rightarrow f_i(f_i, in)|_c, 2)$ r_6: $(cp_i \rightarrow p_i b(q_i \tau, in), 3)$

r_7: $(e_i \rightarrow u_i|_c, 2)$ r_8: $(ts \rightarrow \lambda, 2)$ r_9: $(q_i p_j \rightarrow p_j b, 1)$

r_{10}: $(q_i\, p_j \rightarrow d, 1)$

Rules of type r_1 is used to create sub-membrane and the sub-membrane is used to determine whether there is an edge between two specified vertices, rules of types r_2~r_6 is used to copy and move objects to the new membrane, when rule of type r_9 is applied, it means that there is an edge from the vertex v_i to the vertex v_j, when rule of type r_{10} is applied, it means that there is no edge from the vertex v_i to the vertex v_j. In the process of path construction, after rule of type r_1 is applied, a sub-membrane will be created, then rules of type r_2~r_6 will be used to copy and move objects to the new sub-membrane. let v_i be the last vertex of current path, if there is an edge from v_i to v_j, rule of type r_9 will be applied to create object b and the new sub-membrane will be kept, this means that v_j has been added to the current path; if there is no edge from v_i to v_j, then rule of type r_{10} will be applied to create object d, then the new sub-membrane and objects in it will be dissolved, this means that the current path can't be a Hamiltonian cycle path.

(2) Return to the starting vertex

When all the vertices of graph are all added to the path P, let v_i be the last vertex of path P, we need to determine whether there is an edge from v_i to the starting vertex. If so, path P is a Hamiltonian cycle path; if not, path P is not a Hamiltonian cycle path. Rules in R^C are as follows ($1 \le i \le n$):

r_{11}: $(bf_i \rightarrow y[p_i], 3)$ r_{12}: $(yb \rightarrow (w, out), 1)$ r_{13}: $(yd \rightarrow d\delta, 1)$

r_{14}: $(p_i\, y \rightarrow p_i(y\, q_i, in), 1)$

Under the action of rule of type of r_{11}, a new sub-membrane and p_i which represents the starting vertex will be created. Then under the action of rule of type r_{14}, q_i which represents the last vertex of path P will be created and sent to the sub-membrane. By this time, rule of type r_9 will be applied to create object b if there is an edge from the last vertex of path P to the starting vertex, then because of the existence of object b, rule r_{12} will be applied to create and send object w which shows there exists a Hamiltonian cycle path to outer membrane. Or else, rule of type r_{10} will be applied to create object d if there is no edge from the last vertex of path P to the starting vertex, then because of the existence of object d, rule r_{13} will be applied to dissolve the sub-membrane which shows that path P cannot be a Hamiltonian cycle path.

4.2.3 Path detection

1) Judgment

When object w is sent to the outer membrane by applying rule r_{12} in R^C, it means that a Hamiltonian cycle path has been found. So we need to send message to outer membranes to dispose the result. Rules in R^D associated with the process are:

r_1: $(w \rightarrow z\delta, 1)$ r_2: $(tz \rightarrow v(t,out), 3)$ r_3: $(szt \rightarrow vz, 1)$

r_4: $(k \rightarrow h\delta, 2)$ r_5: $(vzh \rightarrow v(t,out), 4)$ r_6: $(szh \rightarrow z, 2)$

r_7: $(zh \rightarrow k, 5)$

Rule r_1 is used to convert w to z and reduce the thickness of membrane. If object s doesn't exist in current membrane (the existence of object s means that all sub-membranes of current membrane hasn't been all disposed), rule r_2 will be used to create v and send object t to outer membrane. In Π_{HT}, the depth of membrane shows the number of vertices which the current path contains, object p_i in a certain membrane shows vertex v_i is on the current path, the number of object v in membrane shows the number of Hamiltonian cycle path that starts from the current path. If object s exists in current membrane, then rule r_3 will be applied to create object v only.

If no Hamiltonian path has been found, rule r_4 will be applied to send object h to outer membranes (the existence of object h exactly means that no Hamiltonian path has been found). If object s exists in outer membrane, rule r_6 will be used; if only object v exists in outer membrane, that is to say all sub membranes have been disposed and Hamiltonian cycle path exists. At this time, rule r_5 will be applied to delete object h and to send t to outer membrane; if outer membrane doesn't exist object s and object v, this means that all sub membranes has been disposed and no Hamiltonian path has been found, then rule r_7 will be applied to create object k to start delete rules.

2) Pruning

Pruning is to delete surplus membranes and objects, only membranes and objects which represent Hamiltonian cycle path are remained. In Π_{HT}, pruning is needed in following four situations:

(1) When visiting each vertex of graph, let v_i be a vertex which hasn't been visited. We need to determine whether there is an edge from the last vertex on path P to v_i. In Π_{HT}, we dispose it in a new sub-membrane which is created by applying rule of type r_1 in R^C. When there is no edge from the last vertex on path P to v_i, object d will be created by the action of rule of type r_9 in R^C and delete rules in R^D will be started. As a result, the new sub-membrane and objects in it will be dissolved. Rules in R^D associated with the process are ($1 \le i \le n$):

r_8: $(s \rightarrow \lambda|_d, 1)$ r_9: $(t \rightarrow \lambda|_d, 1)$ r_{10}: $(a_i \rightarrow \lambda|_d, 1)$

r_{11}: $(u_i \rightarrow \lambda|_d, 1)$ r_{12}: $(f_i \rightarrow \lambda|_d, 1)$ r_{13}: $(d \rightarrow k\delta, 2)$

r_{14}: $(p_i \rightarrow \lambda|_k, 1)$

(2)When all sub membranes have been created, objects in current membrane except s, p_i should be deleted. Moreover, the thickness of current membrane should be reduced. Rules in R^D associated with this process are ($1 \le i \le n$):

r_{15}: $(bu_i \rightarrow g, 2)$ r_{16}: $(a_i \rightarrow \lambda|_g, 1)$ r_{17}: $(u_i \rightarrow \lambda|_g, 1)$

r_{18}: $(f_i \rightarrow \lambda|_g, 1)$ r_{19}: $(g \rightarrow z\delta, 2)$

Object b is used to control the creation of new membrane and object u_i ($1 \le i \le n$) represents vertex v_i ($1 \le i \le n$) is

already added to the path. When object a_i ($1 \le i \le n$) doesn't exist in current membrane (namely all vertices have been visited), r_{15}, which has a lower priority than r_1 in R^C, will be applied to generate object g to start rules r_{16}~r_{19} to delete corresponding objects and membrane.

(3) When each vertex of graph has been added to path P, we need to determine whether the staring vertex could be added to path P. If not, by applying rule r_{13} in R^D, object d will be created. Then by the action of rule r_{13} and rule r_4 in R^D, the sub-membrane and objects in it will be dissolved, it means that path P is not a Hamiltonian cycle path. Rules in R^D associated with this process are ($1 \le i \le n$):

r_{13}: $(d \rightarrow k\delta, 2)$ r_{14}: $(p_i \rightarrow \lambda|_k, 1)$ r_4: $(k \rightarrow h\delta, 2)$

(4) All sub membranes have been detected and no Hamiltonian path has been found in these sub membranes, then the current membrane should be dissolved. Rules in R^D associated with this process are ($1 \le i \le n$):

r_{14}: $(p_i \rightarrow \lambda|_k, 1)$ r_7: $(zh \rightarrow k, 5)$ r_4: $(k \rightarrow h\delta, 2)$

5. Calculate instance

In this section, we will give an instance to show how to solve Hamiltonian cycle problem in P systems Π_{HT}.

The undirected graph $G=(V, E)$ is shown in Fig.2, let V_1 be the starting and ending vertex, the rules in P systems can be applied as follows:

1) Initialization

First input multiset $a_1a_2a_3a_4a_5$, then input object e, last input f_1. As the result, multiset $a_1a_2a_3a_4a_5ef_1$ is put in skin membrane. The initial membrane structure is shown in Fig.3. For the input multiset, rules in R^I will be applied to send these objects to membrane 2. The available rules in R^I are applied in the order of $\{r_1\}\rightarrow\{r_2\}\rightarrow\{r_3\}\rightarrow\{r_1\}\rightarrow\{r_4, r_5\}$. Rule of type r_1 is used to send a_i and create object s to membrane 2, at this time, membrane 2 has multiset $s^5a_1a_2a_3a_4a_5$. Rule r_2 is applied to send object e to membrane 2, rule r_3 is applied to send object f_1 and create multiset t^2a_1 to membrane 2. Initialization is complete when rules $\{r_4, r_5\}$ are applied successfully. At this time, membrane 2 has multiset $s^3p_1ba_2a_3a_4a_5f_1$.

2) Path construction

After the initialization, legal paths will be constructed by membrane creation. The available rules in R^C are applied in the order of $\{r_1\}\rightarrow\{r_2, r_3, r_4, r_5, r_6\}\rightarrow\{r_{10}\}\rightarrow\{r_{11}\}\rightarrow\{r_{14}\}\rightarrow\{r_{10}\}\rightarrow\{r_{12}\}$. Membrane 2 has the multiset $s^3p_1ba_2a_3a_4a_5f_1$, rule of type r_1 is used to create sub-membrane, rules of type r_2~r_6 are applied to copy objects to new sub-membrane. Firstly, the length of current path is 1, object p_1 shows that V_1 as the starting vertex has been added to path. We need to determine whether there is an edge from V_1 to V_2, V_3, V_4 or V_5. After rule of type r_1 and rules of type r_2~r_6 in R^C are applied, a sub-membrane will be created and have multiset $s^3tq_1p_2a_3a_4a_5f_1\tau$. As shown in Fig.2, there is an edge from V_1 to V_2. So rule of type r_{10} in R^C will be applied, multiset q_1p_2 are converted to p_2b and the new sub-membrane won't be dissolved, this means V_2 has been added to current path. Objects in membrane 2 continue to evolve and this process is corresponding to determine whether there is an edge from V_1 to V_3, V_4 or V_5. After rules of type r_2~r_6 are in R^C applied in the new sub-membrane mentioned above, it has multiset $s^2p_2ba_3a_4a_5f_1\tau$ which shows that V_2 is the last vertex on current path and V_3, V_4 and V_5 haven't been visited. Above described process of extending the current path can cycle in the new sub-membrane.

When each vertex of graph has been added to current path P, we need to determine whether there is an edge from the last vertex of current path to the starting vertex. Let V_5 be the last vertex on path P, as shown in Fig.2, there is an edge from V_5 to V_1, so after rule of type r_{11} and r_{14} in R^C is applied, rule r_{12} will be applied which means V_1 could be added to path P. So path P is a Hamiltonian cycle path.

3) Path detection

(1) Judgment

As shown in Fig.4, after rule r_{12} in R^C are applied, object w will be created and sent to membrane 4. The existence

of object *w* means that there exists a Hamiltonian cycle path. At this time, membrane 5 has multiset $tp_5\tau w$. Because of object τ exists, so after rule r_1 in R^D is applied, membrane 5 will have multiset tp_5z. Then rule r_2 in R^D applied to create object *v* and send object *t* to membrane 4. Then rule r_3 will be applied to evolve multiset *szt* to *vz*.

As shown in Fig.5, when object *d* is created in membrane 6, it means that there exists no Hamiltonian cycle path. At this time, Rule r_{13} will be applied to dissolve membrane 6 and object *d* will be sent to membrane 5. Because the existence of object *d*, object *t* in membrane 5 will be dissolved. Then rule r_{13} in R^D will be applied to reduce the thickness of membrane 5 and convert object *d* to object *k*. Because of the existence object *k*, rule r_{14} in R^D will be applied to dissolve object p_3 in membrane 5.

(2) Pruning

Surplus membranes and objects will be dissolved, only membranes and objects that represent Hamiltonian cycle path are remained. Relevant rules are applied in this instance as follows:

- When visiting each vertex of graph, let v_i be a vertex which we haven't visited. We need to determine whether there is an edge from the last vertex on path *P* to v_i. As shown in Fig.6, rule of type r_9 will be applied to create object *d* because there is no edge from V_1 to V_3. Then rules in R^D will be applied to dissolve membrane 3 and all objects in it.
- As shown in Fig.7, all object a_i has evolved to u_i which means that we have tried adding each vertex to current path. Then we need to dissolve objects in membrane 2 except *s*, p_i. Rules in R^D are applied in the order of $\{r_{15}\}\rightarrow\{r_{17}, r_{18}\}\rightarrow\{r_{19}\}$.
- As shown in Fig.8, each vertex has been added to current path. But because there is no edge from V_3 to V_1, rule of type r_9 in R^D and rule r_{13} in R^C are applied to dissolve membrane 6 and object *d* will be sent into membrane 5, and then object *d* will evolve to object *k* under the action of rule r_{13} in R^D.
- As shown in Fig.9, all sub membranes of membrane 4 have been disposed and there exists no Hamiltonian cycle path. Membrane 4 and object p_2 in it will be dissolved by the action of rule r_7 and rule of type r_{14} in R^D. After rules in R^D are applied, object *h* which shows there exist no Hamiltonian cycle path will be sent into membrane 3.

When the whole system halts, the final membrane structure is shown in Fig.10. As we can see in Fig.10, only object p_i that represents the order of vertices and object *v* that represent the number of the Hamiltonian path are remained. From the membrane structure shown in Fig.10, we can detect all Hamiltonian paths that start from V_1 and end with V_1. According to the number of object *v* in membrane 1, we know that there are three Hamiltonian paths. By detect object p_i in each membranes, we can find these paths are :$\{V_1 \rightarrow V_2 \rightarrow V_3 \rightarrow V_4 \rightarrow V_5 \rightarrow V_1\}$, $\{V_1 \rightarrow V_2 \rightarrow V_3 \rightarrow V_5 \rightarrow V_4 \rightarrow V_1\}$, $\{V_1 \rightarrow V_2 \rightarrow V_5 \rightarrow V_3 \rightarrow V_4 \rightarrow V_1\}$.

6. Conclusions

This paper presents a family of P systems to solve Hamiltonian cycle problem in polynomial time. In this P system, all legal paths are constructed by membrane creation, and illegal paths are deleted by path detection while constructing paths. An instance is given to illustrate the feasibility and effectiveness of our designed P system. Because too many new membranes and objects are created in solving Hamiltonian cycle problem, space complexity of the P system is a little high. Therefore, reducing space complexity of Π_{HT} is our main future research work.

Reference

[1] A. Vitale, G. Mauria, C. Zandrom, Simulation of a bounded symport/antiport P system with Brane calculi, Biosystems, 91(3) (2008) 558-571.

[2] C. Martin-Vide, Gh. Păun, A. Rodríguez-Patón, On P systems with membrane creation, Computer Science Journal

of Moldova, 9(2) (2001) 134-145.

[3] C. Mart, Gh. Păun, J. Pazos, Tissue P systems, Theoretical Computer Science, 296(2) (2003) 295-326.

[4] R. Freund, Gh. Păun, M. J. Pérez-Jiménez, Tissue P systems with channel states, Theoretical Computer Science, 330(1) (2003) 101-116.

[5] X.-Y. Zhang, X.-X. Zeng, B. Luo, and J.-B. Xu, Several Applications of Spiking Neural P Systems with Weights, Journal of Computational and Theoretical Nanoscience, 9(6) (2012) 769-777.

[6] T. Song, Y. Jiang, X.-L. Shi, and X.-X. Zeng, Small Universal Spiking Neural P Systems with Anti-Spikes, Journal of Computational and Theoretical Nanoscience, 10(4) (2013). 999-1006.

[7] Gh. Păun, Computing with membranes, Journal of Computer and System Sciences, 61(2000): 108–143.

[8] Gh. Păun, Membrane Computing, An Introduction, Springer-Verlag, Berlín, Heidelberg, 2002.

[9] M.A. Gutiérrez-Naranjo, M.A. Pérez-Jiménez, F.J. Romero-Campero, A uniform solution to SAT using membrane creation. Theoretical Computer Science, 371(1-2) (2007) 54–61.

[10] Z. Gazdag, Solving SAT by P Systems with Active Membranes in Linear Time in the Number of Variables. In: 14th International Conference on Membrane Computing (CMC 2013), LNCS 8340, Springer-Verlag Berlin Heidelberg 2014, pp. 189-205.

[11] P. Guo, Y.-W. Zhong, H.-Z Chen, R. Liu, A P system for finding all solutions of the vertex cover problem, Journal of Computational and Theoretical Nanoscience, 12(12) (2015) 5229-5235.

[12] P. Guo, J.-F. Ji, H.-Z. Chen, R. Liu, Solving All-SAT Problems by P Systems, Chinese Journal of Electronics, 24(4) (2015) 744-749.

[13] P. Guo, J. Zhu, M.-Q. Zhou, A family of uniform P systems for All-SAT problem, Journal of Computational and Theoretical Nanoscience, 13(1) (2016) 319-326.

[14] L.-Q Pan, A. Alhazov, Solving HPP and SAT by P systems with active membranes and separation rules, Acta Informatica, 43(2) (2006) 13 l-145.

[15] K Ishii, A. Fujiwara, Asynchronous P systems for SAT and Hamiltonian Cycle Problem, in: 2010 Second World Congress on Nature & Biologically Inspired Computing, IEEE, 2010, pp. 513-519.

[16] A. Leporati, M.A. Gutiérrez-Naranjo, Solving Subset Sum by spiking neural P systems with pre-computed resources, Fundamenta Informaticae, 87(1) (2008) 61–77.

[17] J. He, J. Xiao, X. Shi, A membrane-inspired algorithm with a memory mechanism for knapsack problems, Journal of Zhejiang University Science C, 14 (8) (2013) 612-622.

[18] J.-J. He, J.-H Xiao, X. Liu, T.-F. Wu, T. Song, A novel membrane-inspired algorithm for optimizing solid waste transportation, Optik, 126(23) (2015) 3883-3888.

[19] J.-H. Xiao, Y.-F. Huang, Z. Cheng, J.-J. He, Y.-Y. Niu, A hybrid membrane evolutionary algorithm for solving constrained optimization problems, Optik, 125(2) (2014) 897-902.

[20] C. Liu, D.-L Chen, F.-C. Wan, Multiobjective learning algorithm based on membrane systems for optimizing the parameters of extreme learning machine, Optik, 127(4) (2016) 1909-1917.

[21] M.J. Perez-Jiménez, A. Romero–Jiménez, F. Sancho-Caparrini, Computationally hard problems addressed through P systems, in: Applications of membrane computing, Springer, Berlin Heidelberg, 2006, pp. 313-346.

Figure Caption

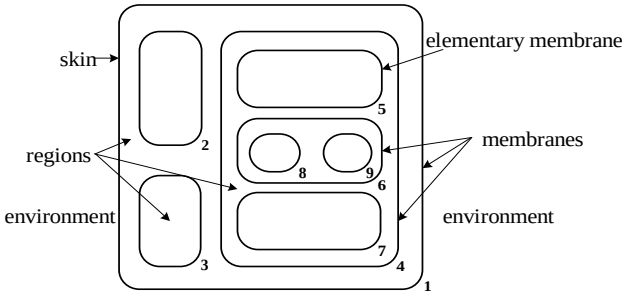

Fig.1. The structure of cell-like P system [8]

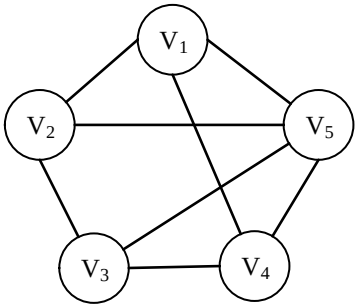

Fig.2. Undirected graph *G*

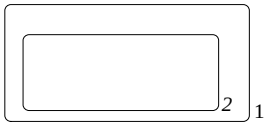

Fig.3. The initial structure of the P system Π_{HT}

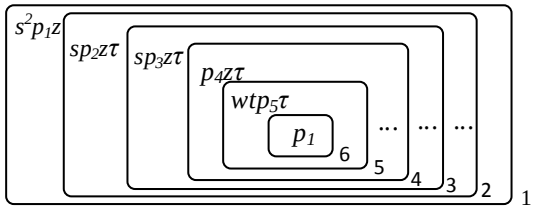

Fig.4. Exist a Hamiltonian cycle path

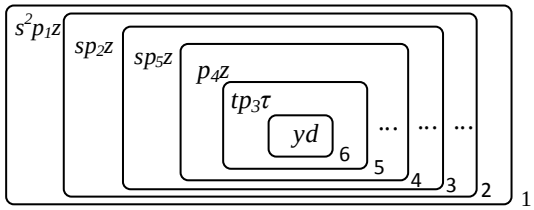

Fig.5. Doesn't exist a Hamiltonian cycle path

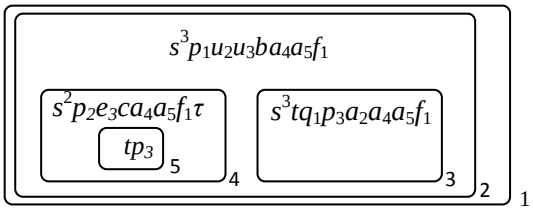

Fig.6. Visiting vertex of graph *G*

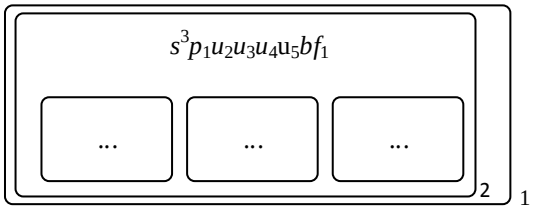

Fig.7. All object a_i has evolved to u_i in membrane 2

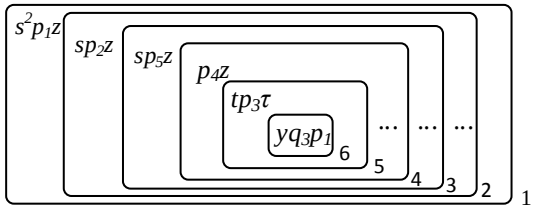

Fig.8. All vertex has been added to current path

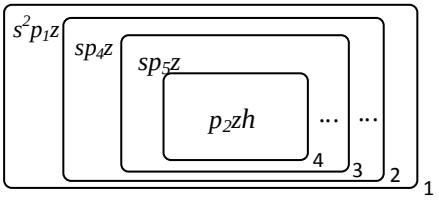

Fig.9. After dispose membrane 5 from Fig.8

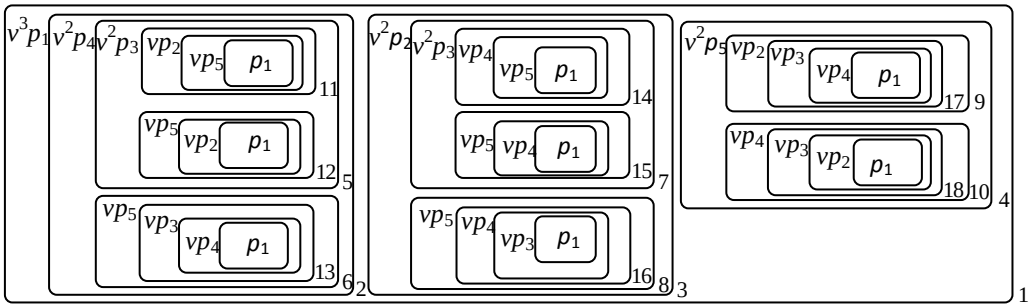

Fig.10. The final membrane structures