International Journal of Networking and Computing – www.ijnc.org, ISSN 2185-2847
Volume 10, Number 2, pages 159–173, July 2020

P systems with branch and bound for solving two hard graph problems

Kotaro Umetsu and Akihiro Fujiwara
Graduate School of Computer Science and Systems Engineering,
Kyushu Institute of Technology
Iizuka, Fukuoka, 820-8502, Japan

Received: February 12, 2020
Revised: April 10, 2020
Accepted: May 30, 2020
Communicated by Susumu Matsumae

Abstract

Membrane computing is a computational model based on activity of cells. Using the membrane computing, a number of computationally hard problems have been solved in a polynomial number of steps using an exponential number of membranes. However, the number of membranes denotes the number of cells from practical point of view, and the reduction of the number of membranes must be considered for using the membrane computing in real world.

In this paper, we propose asynchronous P systems with branch and bound for reducing the number of membranes for two computationally hard graph problems. We first propose an asynchronous P system that solves Hamiltonian cycle problem for a graph with n vertices, and show that the proposed P system works in $O(n^2)$ parallel steps. We next propose an asynchronous P system that solves the minimum graph coloring for a graph with n vertices, and also show that the P system works in $O(n^2)$ parallel steps.

In addition, we evaluate validity of the proposed P systems using computational simulations. The experimental results show the validity and efficiency of the proposed P systems with branch and bound.

Keywords: membrane computing, Hamiltonian cycle, graph coloring, branch and bound

1 Introduction

Natural computing is one of the next-generation computing paradigms. The membrane computing, which has been initially introduced in [7] as a P system, is a representative computational model in the natural computing. Definition of the P system is based on a feature of cells, and a membrane and an object in the P system denote a computing cell and a storage of data, respectively. In addition, each object evolves according to evolution rules, which is associated with the membrane.

Since the exponential number of membranes can be created in the polynomial number of steps by a division rule, which is one of evolution rules on the P system, the computationally hard problem can be solved in a polynomial number of steps. Therefore, there are a number of P systems that solve computationally hard problems [1, 2, 3, 5, 6, 8, 9, 10, 11, 12, 13, 14].

In addition, asynchronous parallelism, which assumes asynchronous application of evolution rules, has been considered for the P system. The asynchronous parallelism means that all objects may react to rules with different speeds on the P system. The asynchronous parallelism makes a P system a more realistic computational model because livings cells work independently according to environment.

A number of asynchronous P systems have also been proposed for NP problems [1, 3, 12, 13]. For example, an asynchronous P system for Hamiltonian cycle has been proposed in [12]. The P system for the Hamiltonian cycle works in $O(n!)$ sequential steps or $O(n^2)$ parallel steps, for a graph with n vertices. For another example, an asynchronous P system for finding the minimum graph coloring has been proposed in [13] . The P system for a graph with n vertices works in $O(n^n)$ sequential steps or $O(n^2)$ parallel steps.

All of the above P systems solves the computationally hard problems in polynomial numbers of steps using exponential numbers of membranes. The number of membranes means the number of cells, and reduction of the number of membranes must be considered in case that the P system is implemented using living cells because the cells cannot be created exponentially.

Recently, an asynchronous P system for SAT using branch and bound [4] has been proposed for reducing the number of membranes. Branch and bound is a well-known optimization technique, and the number of membranes is reduced in the P system using branch and bound by omitting partial value assignments that cannot satisfy a given Boolean formula.

In the present paper, we propose two asynchronous P systems with branch and bound for reducing the number of membranes for computationally hard graph problems, which are Hamiltonian cycle and the minimum graph coloring.

We first propose an asynchronous P system with branch and bound for Hamiltonian cycle. We show that the proposed P system solves the Hamiltonian cycle problem for a graph with n vertices with the same complexity in [12], that is, the P system works in $O(n!)$ sequential steps or $O(n^2)$ parallel steps. In the proposed P system, a partial permutation of vertices is created, and then, existence of edges between vertices is checked for the partial permutation. If there is no edge between the vertices, the partial permutation is discarded as a bounding operation. Since the number of membranes increases according to the number of created permutations of vertices, the number can be reduced by omitting permutations that must not be a Hamiltonian cycle.

We next propose an asynchronous P system with branch and bound for solving the minimum graph coloring problem. In the proposed P system, vertices are colored one by one, and the adjacent vertices are checked for the partial color assignment. If the adjacent vertices have the same color, the partial color assignment of vertices is discarded as a bounding operation. Since the number of membranes increases according to the number of color assignments of vertices, the number can be reduced by omitting partial assignments that are not possible for the graph coloring. We show that the proposed P system computes the minimum graph coloring with the same complexity in [13], that is, the P system solves the minimum graph coloring problem for a graph with n vertices in $O(n^n)$ sequential steps or $O(n^2)$ parallel steps.

Finally, we show the validity of the proposed systems through experimental simulations. In the simulation, various instances are executed on the previous P systems [12, 13] and the proposed P systems, and the numbers of membranes are compared for the same instances. The results show validity and efficiency of the proposed P systems.

The remainder of the paper is organized as follows. In Section 2, we describe the computational model for the membrane computing. In Section 3 and Section 4, we propose two P systems with branch and bound for Hamiltonian cycle and the minimum graph coloring, respectively. In Section 5, we show experimental results for the previous P systems and the proposed P systems. Finally, Section 6 concludes the paper.

2 Preliminaries

In this paper, we assume an asynchronous P system proposed in [12]. We briefly explain definition of the P system in the following.

The P system mainly consists of membranes and objects. A membrane, which is a computing cell in the P system, may contain objects and other membranes, and each membrane is labeled with an integer. An object stores data as a memory storage in the P system. We assume that each

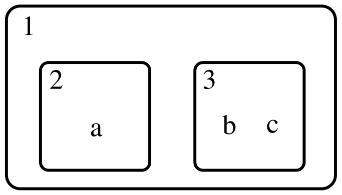

Figure 1: An example of membranes and objects

object is a finite strings over a given set of alphabet. Objects may be evolved into another objects or pass through membranes according to the evolution rules, which are defined later. In addition, dissolution and division of a membrane may be triggered by contained objects.

For example of the membrane and the object, the following expression and Fig. 1 denote the same membrane structure that consists of three membranes and three objects.

$$[\ [\ a\ ]_2\ [\ b\ c\ ]_3\ ]_1$$

In the example, the membrane labeled 1 contains two membranes labeled 2 and 3, and the membrane labeled 2 and 3 contain sets of objects $\{a\}$ and $\{b, c\}$, respectively.

Computation of P systems is defined as evolution rules. Each evolution rule is a rewriting rule for membranes and objects. According to the applicable evolution rules, objects and membranes are transformed in parallel in each step of computation. The system stops computation if there is no applicable evolution rule for membranes and objects.

A number of types of evolution rules are assumed in the membrane computing. In this paper, we assume the following five rules in [12].

(1) Object evolution rule: $[\ \alpha\ ]_h \rightarrow [\ \beta\ ]_h$

A set of objects α is transformed into another set of objects β. In the above rule, h is a label of the membrane, and α and β are multi-subsets of O. (We omit the brackets in each evolution rule such as $\alpha \rightarrow \beta$ for cases that a corresponding membrane is obvious.)

(2) Send-in communication rule: $\alpha\ [\ ]_h \rightarrow [\ \beta\ ]_h$

A set of objects α is moved into the inner membrane labeled by h, and transformed into another set of objects β. In the above rule, h is a label of the membrane, and α and β are multi-subsets of O.

(3) Send-out communication rule: $[\ \alpha\ ]_h \rightarrow [\ ]_h\ \beta$

A set of objects α is sent out from the membrane labeled by h, and transformed into another set of objects β. In the above rule, h is a label of the membrane, and α and β are multi-subsets of O.

(4) Dissolution rule: $[\ \alpha\ ]_h \rightarrow \beta$

The membrane that contains a set of objects α is dissolved, and α is transformed into another set of objects β. (Note that the outermost membrane cannot be dissolved.) In the above rule, h is a label of the membrane, and α and β are multi-subsets of O. Using the rule, all objects in the dissolved membrane are moved to the outer membrane.

(5) Division rule: $[\ \alpha\ ]_h \rightarrow [\ \beta\ ]_h[\ \gamma\ ]_h$

The membrane that contains a set of objects α is divided into two membranes with the same label, and α is transformed into another two set of objects, β and γ, in each of the divided membranes. In the above rule, h is a label of the membrane, and α, β and γ are multi-subsets of O. Using the rule, all other objects in the membrane, which contains α, are copied into two divided membranes that contain objects β and γ.

In the above five rules, send-in, send-out, dissolution and division rules are blocking rules for the same membrane. In other words, only one of the four rules can be applied for a membrane in each step of computation on the P system.

We now summarize definition of the P system. The P system consists of the following four components.

O**:** The set of objects used in the system.

μ**:** A structure of membrane.

ω_i**:** A set of objects initially contained in the membrane labeled i.

R_i**:** A set of evolution rules for a membrane labeled i.

Using the above components, the P system with m membranes is defined as follows.

$$\Pi = (O, \mu, \omega_1, \omega_2, \cdots, \omega_m . R_1, R_2, \cdots, R_m)$$

We now consider complexity of the P system. We assume that each of evolution rules can be executed in one step on the computational model, and the complexity of the P system is defined as the number of steps executed on the P system.

On the standard P system, all objects, for which there are applicable evolution rules, are transformed with maximally parallel manner. (In case that there are a number of applicable evolution rules for an object, one of the rules is applied non-deterministically.) Using the maximally parallel manner, we can consider an execution on the P system easily because the execution with the maximally parallel manner is unique. The complexity of the P system with maximally parallel manner is the best case complexity, and we call the number of steps executed with maximal parallel manner as *the number of parallel steps.*

In this paper, we also consider asynchronous parallelism [12] in the P system. Under the assumption of the asynchronous parallelism, any number of applicable evolution rules are applied in parallel. In other words, all objects, for which there are applicable evolution rules, can be transformed in parallel, or only one of the applicable evolution rules is applied in each step of computation. We call the number of of steps in the latter case as *the number of sequential steps.* The number of sequential steps denotes the complexity of the P system in the worst case.

3 An asynchronous P system with branch and bound for Hamiltonian cycle

In this section, we propose an asynchronous P system with branch and bound for Hamiltonian cycle. We first explain input and output for the Hamiltonian cycle, and next show an algorithm on the P system. We finally consider complexity of the proposed P system.

3.1 Input and output for the Hamiltonian cycle problem

Hamiltonian cycle is a well known computationally hard problem. An input of the problem is a directed graph $G = (V, E)$, and an output is "true" or "false". In case that there exists a Hamiltonian cycle, which is a simple cycle such that the cycle visits each vertex exactly once, the output is "true", otherwise, the output is "false". In case that a directed graph in Fig. 2 is an input, there is a Hamiltonian cycle $(v_1, v_2, v_3, v_4, v_1)$.

For the proposed P system, the following two sets of objects, O_V and O_E, are given as an input graph.

$$\begin{aligned} O_V &= \{\langle v_i \rangle \mid 1 \le i \le n\} \\ O_E &= \{\langle e_{i,j}, W \rangle \mid 1 \le i \le n, 1 \le j \le n, W \in \{T, F\}\} \end{aligned}$$

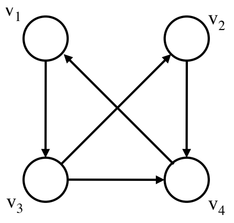

Figure 2: A directed graph containing a Hamiltonian cycle $(v_1, v_3, v_2, v_4, v_1)$.

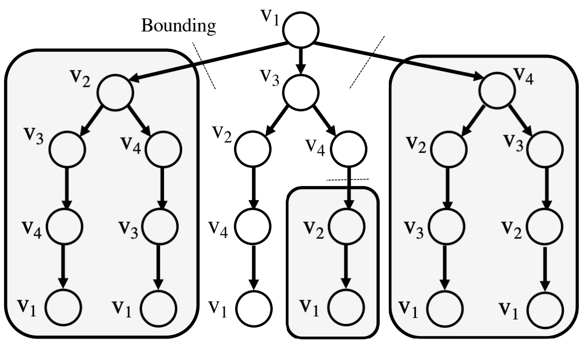

Figure 3: An example of decision tree with branch and bound for Hamiltonian cycle

In the input objects, a vertex v_i and an edge (v_i, v_j) are denoted by objects $\langle v_i \rangle$ and $\langle e_{i,j}, W \rangle$, respectively. A value W is set to T if an edge (v_i, v_j) is in the graph, otherwise, W is set to F.

For example, the following O_V and O_E are given as an input of the P system for a directed graph in Fig. 2.

$$\begin{aligned}
O_V &= \{\langle v_1 \rangle, \langle v_2 \rangle, \langle v_3 \rangle, \langle v_4 \rangle\} \\
O_E &= \{\langle e_{1,1}, F \rangle, \langle e_{1,2}, F \rangle, \langle e_{1,3}, T \rangle, \langle e_{1,4}, F \rangle, \\
&\quad \langle e_{2,1}, F \rangle, \langle e_{2,2}, F \rangle, \langle e_{2,3}, F \rangle, \langle e_{2,4}, T \rangle, \\
&\quad \langle e_{3,1}, F \rangle, \langle e_{3,2}, T \rangle, \langle e_{3,3}, F \rangle, \langle e_{3,4}, T \rangle, \\
&\quad \langle e_{4,1}, T \rangle, \langle e_{4,2}, F \rangle, \langle e_{4,3}, F \rangle, \langle e_{4,4}, F \rangle\}
\end{aligned}$$

We assume that a computation on the P system starts if the above O_V and O_E are given from the outside region into the skin membrane.

The output of the P system is one of two objects, $\langle TRUE \rangle$ and $\langle FALSE \rangle$, which denote "true" and "false", respectively.

3.2 Branch and bound for Hamiltonian cycle

The branch and bound is an well known technique for optimization. On the P system for solving Hamiltonian cycle [12], all permutation of vertices are created for an input graph with n vertices, and $(n-1)!$ permutations are checked whether the cycle is Hamiltonian. However, a partial permutation of vertices can be discarded if the output is determined to be "false".

Fig. 3 shows a decision tree for illustration of the above idea. Let a directed graph in Fig. 2 be an input graph. In the decision tree, a permutation of vertices is denoted by a path from root to a leaf node. In this case, a path starting with an edge from v_1 to v_2 can be bounded because there is no edge between v_1 and v_2, and the permutation must not be cycle. In another case, a path starting with an edge from v_1 to v_4 is bounded similarly.

The following is an outline of the proposed P system for Hamiltonian cycle using branch and bound. The P system consists of two membrane such that $[\,[\,]_2\,]_1$. The membranes labeled 1 and 2 are called *outer* and *inner* membranes, respectively.

An outline of the proposed P system for Hamiltonian cycle

Step 1: In the outer membrane, move input objects into the inner membrane using send-in communication rules.

Step 2: In each inner membrane, repeat the following (2-1) and (2-2) until "true" or "false" is outputted.

(2-1) Select a vertex, which is not visited, as a next vertex. In case that there is an edge from the last vertex in the permutation to the next vertex, the next vertex is added to the partial permutation of vertices. (The partial permutation denotes a path from a start vertex.) The partial permutation with the next vertex is created by dividing the inner membrane.

In case that there is no edge from the last vertex to the next vertex, the partial permutation is discarded, and an object that denotes "false" is outputted. (The divided membrane stops the computation.)

(2-2) In case that length of the permutation is equal to the number of vertices, check the created permutation of vertices whether the permutation denotes a Hamiltonian cycle. If there exists an edge from the last vertex in the permutation to the start vertex, output an object that denotes "true", otherwise, output an object that denotes "false".

Step 3: Send out a final result, "true" or "false", from the outer membrane using send-out communication rules.

3.3 Details of the proposed P system

We now explain details of each step of the computation for Hamiltonian cycle. In Step 1, all input objects in the outer membrane are moved into the inner membrane. The input objects are moved sequentially in the step because the proposed P system is asynchronous. The Step 1 is executed using the following $R_{OUT,1}$ and $R_{IN,1}$. (In the following, $R_{OUT,i}$ and $R_{IN,i}$ denote sets of evolution rules for outer and inner membranes in Step i, respectively.)

$$\begin{aligned} R_{OUT,1} &= \{\langle e_{1,1},F\rangle[\ ]_2 \rightarrow [\langle M_{2,1}\rangle\langle e_{1,1},F\rangle]_2\} \\ &\quad \cup \{\langle M_{i,j}\rangle\langle e_{i,j},V\rangle[\ ]_2 \rightarrow [\langle M_{i+1,j}\rangle\langle e_{i,j},V\rangle]_2 \mid 1 \le i \le n, 1 \le j \le n, V \in \{T,F\}\}\} \\ &\quad \cup \{\langle M_{i,n+1}\rangle\langle v_i\rangle[\ ]_2 \rightarrow [\langle M_{i+1,n+1}\rangle\langle v_i\rangle]_2 \mid 1 \le i \le n\} \\ &\quad \cup \{\langle M_{n+1,j}\rangle \rightarrow \langle M_{1,j+1}\rangle \mid 1 \le j \le n+1\} \\ &\quad \cup \{\langle M_{1,n+2}\rangle \rightarrow \langle OUT\rangle\langle C\rangle, \langle C\rangle[\ ]_2 \rightarrow [\langle C_{1,1}\rangle]_2\} \\ R_{IN,1} &= \{[\langle M_{i,j}\rangle]_2 \rightarrow [\ ]_2\langle M_{i,j}\rangle \mid\ 2 \le i \le n+1, 1 \le j \le n+1\} \end{aligned}$$

In the above evolution rules, object $\langle e_{1,1},F\rangle$ starts the computation, and two kinds of input objects are moved into the inner membrane using object $\langle M_{i,j}\rangle$. At the end of Step 1, two objects, $\langle OUT\rangle$ and $\langle C\rangle$, are created in the outer membrane. The created object $\langle OUT\rangle$ is used for outputting an object that denotes "true" or "false" at the end of the final step. The other created object $\langle C\rangle$ is sent into the inner membrane as $\langle C_{1,1}\rangle$, and the object triggers computation of Step 2.

Step 2 consists of sub-steps (2-1) and (2-2). At the beginning of Step 2, the following evolution rule is executed for setting status of the start vertex v_1 to *visited.*

$$R_{IN,\text{(2-0)}} = \{\langle C_{1,1}\rangle\langle v_1\rangle \rightarrow \langle C_{2,1}\rangle\langle \lambda_1\rangle\langle z_{1,1}\rangle\}$$

In the evolution rules, object $\langle \lambda_j\rangle$ denotes that vertex v_j is visited, and object $\langle z_{i,j}\rangle$ denotes that the i-th vertex in the permutation is v_j.

In (2-1), an unvisited vertex is selected as a next vertex, and the next vertex is added to the partial permutation of vertices in case that there is an edge from the last vertex in the permutation to the next vertex. (The partial permutation denotes a path from a start vertex.) The (2-1) is executed using the following $R_{IN,(2\text{-}1\text{-}1)}$. The partial permutation with the next vertex is created by dividing the inner membrane with the first set of evolution rules, and the next unvisited vertex is selected using the second set of evolution rules.

$$\begin{aligned} R_{IN,(2\text{-}1\text{-}1)} &= \{[\langle z_{k-1,i}\rangle\langle C_{k,j}\rangle\langle v_j\rangle\langle e_{i,j},T\rangle]_2 \to [\langle z_{k-1,i}\rangle\langle C_{k,j+1}\rangle\langle v_j\rangle\langle e_{i,j},T\rangle]_2 \\ &\qquad [\langle z_{k-1,i}\rangle\langle C_{k+1,1}\rangle\langle z_{k,j}\rangle\langle \lambda_j\rangle]_2 \mid 2 \le k \le n, 1 \le i \le n, 1 \le j \le n\} \\ &\quad \cup\{\langle C_{k,j}\rangle\langle \lambda_j\rangle \to \langle C_{k,j+1}\rangle\langle \lambda_j\rangle \mid 2 \le k \le n, 1 \le j \le n\} \end{aligned}$$

On the other hand, in (2-1), the partial permutation is discarded in case that there is no edge from the last vertex to the next vertex. In this case, membrane division is executed to select the next vertex, and an object that denotes “false” is outputted by applying the following $R_{IN,(2\text{-}1\text{-}2)}$.

$$\begin{aligned} R_{IN,(2\text{-}1\text{-}2)} &= \{[\langle z_{k-1,i}\rangle\langle C_{k,j}\rangle\langle v_j\rangle\langle e_{i,j},F\rangle]_2 \to [\langle z_{k-1,i}\rangle\langle C_{k,j+1}\rangle\langle v_j\rangle\langle e_{i,j},F\rangle]_2[\langle FALSE, n-k\rangle]_2 \\ &\qquad \mid 2 \le k \le n, 1 \le i \le n, 1 \le j \le n\} \\ &\quad \cup\{[\langle FALSE, n-k\rangle]_2 \to [\ ]_2\langle FALSE, n-k\rangle \mid 2 \le k \le n\} \end{aligned}$$

In the above evolution rules, object $\langle FALSE, n-k\rangle$ is created to count the number of “false” in Step 3.

In (2-2), the created permutation of vertices is checked whether the permutation denotes a Hamiltonian cycle. In the following $R_{IN,(2\text{-}2)}$, the first set of evolution rules is applied for outputting an object that denotes “true” in case that there exists an edge from the last vertex in the permutation to the start vertex. Otherwise, the second set of evolution rules is applied for outputting an object that denotes “false”.

$$\begin{aligned} R_{IN,(2\text{-}2)} &= \{[\langle C_{n+1,1}\rangle\langle z_{n,i}\rangle\langle e_{i,1},T\rangle]_2 \to [\ ]_2\ \langle TRUE\rangle \mid 1 \le i \le n\} \\ &\quad \cup\{[\langle C_{n+1,1}\rangle\langle z_{n,i}\rangle\langle e_{i,1},F\rangle]_2 \to [\ ]_2\ \langle FALSE, 0\rangle \mid 1 \le i \le n\} \end{aligned}$$

We now summarize the set of evolution rules for Step 2 as follows.

$$R_{IN,2} = R_{IN,(2\text{-}0)} \cup R_{IN,(2\text{-}1\text{-}1)} \cup R_{IN,(2\text{-}1\text{-}2)} \cup R_{IN,(2\text{-}2)}$$

In Step 3, a final result is sent out from the outer membrane. The Step 3 is executed applying the following $R_{OUT,3}$.

$$\begin{aligned} R_{OUT,3} &= \{\{[\langle TRUE\rangle\langle OUT\rangle]_1 \to [\ ]_1\langle TRUE\rangle\} \\ &\quad \cup\{\langle FALSE, k\rangle^{k+1} \to \langle FALSE, k+1\rangle \mid 0 \le k \le n-2\} \\ &\quad \cup\{\{[\langle FALSE, n-1\rangle\langle OUT\rangle]_1 \to [\ ]_1\langle FALSE\rangle\} \end{aligned}$$

If object $\langle TRUE\rangle$ is in the outer membrane, there is a Hamiltonian cycle in the input graph, and the object is sent out from the outer membrane immediately applying the first evolution rules. On the other hand, objects that denotes “false” is outputted from all inner membranes in case that the final result is “false”. Therefore, the sum of “false” is counted asynchronously using the second set of evolution rules, and final object $\langle FALSE\rangle$ is outputted if and only if all outputs of inner membranes are “false”.

We now summarize the asynchronous P system $\Pi_{\mathrm{BB-HC}}$ for solving Hamiltonian cycle as follows.

$$\Pi_{\mathrm{BB-HC}} = (O, \mu, \omega_1, \omega_2, R_{OUT}, R_{IN})$$

- $O = O_V \cup O_E \cup O_{TRUE} \cup O_{FALSE} \cup O_M \cup O_C \cup O_\lambda \cup O_z$
- $O_{TRUE} = \{\langle TRUE\rangle\}$
- $O_{FALSE} = \{\langle FALSE\rangle\} \cup \{\langle FALSE, k\rangle \mid 1 \le k \le n\}$

- $O_M = \{\langle M_{i,j}\rangle \mid 1 \leq i \leq n+1, 1 \leq j \leq n+1\}$
- $O_C = \{\langle C_{i,j}\rangle \mid 1 \leq i \leq n+1, 1 \leq j \leq n+1\}$
- $O_\lambda = \{\langle \lambda_j\rangle \mid 1 \leq j \leq n\}$
- $O_z = \{\langle z_{i,j}\rangle \mid 1 \leq i \leq n, 1 \leq j \leq n\}$
- $\mu = [\ [\ ]_2\ ]_1$
- $\omega_1 = \omega_2 = \phi$
- $R_{OUT} = R_{OUT,1} \cup R_{OUT,3}$, $R_{IN} = R_{IN,1} \cup R_{IN,2}$

3.4 Complexity of the P system

We now consider the complexity of asynchronous P system Π_{BB-HC}. Since $O(n^2)$ objects are moved from the outer membrane into the inner membrane sequentially, Step 1 can be executed in $O(n^2)$ parallel or sequential steps using $O(n^2)$ kinds of objects and $O(n^2)$ kinds of evolution rules.

In Step 2, $O((n-1)!)$ membranes are created in the worst case, and evolution rules are applied sequentially at most $O(n)$ times in each membrane. Therefore, Step 2 can be executed in $O(n^2)$ parallel steps and $O(n!)$ sequential steps using $O(n^2)$ kinds of objects and $O(n^3)$ kinds of evolution rules.

In the final step, $O(n)$ evolution rules are applied sequentially in cast that the output is "false", and Step 3 can be executed in $O(n)$ parallel step or $O(n)$ sequential step using $O(n)$ kinds of objects and $O(n)$ kinds of evolution rules.

We obtain the following theorem for the above asynchronous P system Π_{BB-HC}.

Theorem 1 *The asynchronous P system Π_{BB-HC}, which computes Hamiltonian cycle for a directed graph with n vertices, works in $O(n!)$ sequential steps or $O(n^2)$ parallel steps by using $O(n^2)$ kinds of objects and evolution rules of size $O(n^3)$.* □

4 An asynchronous P system with branch and bound for minimum graph coloring

In this section, we explain our proposed asynchronous P system for the minimum graph coloring. An input and an output of the problem on the P system is shown, and then, an outline and details of the P system are presented. Finally, the complexity of the proposed P system is discussed.

4.1 Input and output for minimum graph coloring

The minimum graph coloring is a well known computationally hard problem. An input of the problem is an undirected graph $G = (V, E)$. An output of the problem is a coloring of vertices, which is an assignment of colors to all vertices such that no pair of adjacent vertices with the same color. For example, let the graph in Fig. 4 be an input graph. Then, both of two assignments of colors, $c_a = \{v_1\}, c_b = \{v_2\}, c_c = \{v_3, v_4\}$ and $c_a = \{v_1\}, c_b = \{v_2\}, c_c = \{v_3\}, c_d = \{v_4\}$, are assignments of colors for the graph. In case of the graph, the former assignment is the minimum graph coloring.

For the proposed P system, the following set of objects, O_E, are given as an input graph.

$$O_E \quad = \quad \{\langle e_{i,j}, W\rangle \mid 1 \leq i \leq n, 1 \leq j \leq n, W \in \{T, F\}\}$$

In the above set of input objects, an edge (v_i, v_j) is denoted by an objects $\langle e_{i,j}, W\rangle$. A value W is set to T if an edge (v_i, v_j) is in the graph, otherwise, W is set to F.

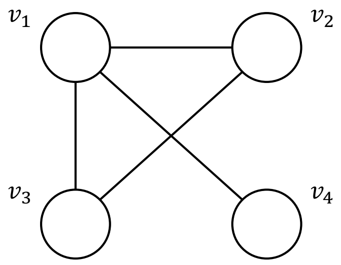

Figure 4: Example of an input undirected graph

For example, the following O_E is given as an input of the P system for a graph in Fig. 4.

$$\begin{aligned} O_E \;=\; & \{\langle e_{1,1}, F\rangle, \langle e_{1,2}, T\rangle, \langle e_{1,3}, T\rangle, \langle e_{1,4}, T\rangle, \\ & \langle e_{2,1}, T\rangle, \langle e_{2,2}, F\rangle, \langle e_{2,3}, T\rangle, \langle e_{2,4}, F\rangle, \\ & \langle e_{3,1}, T\rangle, \langle e_{3,2}, T\rangle, \langle e_{3,3}, F\rangle, \langle e_{3,4}, F\rangle, \\ & \langle e_{4,1}, T\rangle, \langle e_{4,2}, F\rangle, \langle e_{4,3}, F\rangle, \langle e_{4,4}, F\rangle\} \end{aligned}$$

We assume that a computation on the P system starts if the above O_E are given from the outside region into the skin membrane.

The output of the P system is the following set of objects, O_C, which denotes color assignments of all vertices.

$$O_C \;=\; \{\langle V_i, h\rangle \mid 1 \le i \le n, 1 \le h \le n\}$$

In an object $\langle V_i, h\rangle$, h denotes a color of vertex v_i.

For example, the following O_C is an output of the P system, which denotes minimum color assignment for the graph in Fig. 4.

$$O_C \;=\; \{\langle V_1, 1\rangle, \langle V_2, 2\rangle, \langle V_3, 3\rangle, \langle V_4, 3\rangle\}$$

4.2 Branch and bound for minimum graph coloring

Branch and bound is a well-known computing paradigm for the optimization problem. On the existing P system for solving minimum graph coloring [13], all assignments of colors for vertices are created for an input graph with n vertices, and n^n assignments are checked as to whether each assignment of colors is valid. However, a partial assignment of vertices can be discarded if the valid assignment of vertices is determined.

Fig. 5 shows an example of a search tree for the above concept. Let the graph in Fig. 4 be an input graph, and we consider a 3-coloring for the graph. In the search tree, a color assignment of vertices is denoted by a path from a root node to a leaf node. In this case, a path starting from v_1, which is colored with C_1, to v_2, which is also colored with C_1, can be bounded because there is an edge between v_1 and v_2 in the input graph.

We now explain an overview of the asynchronous P system with branch and bound for finding the minimum graph coloring. An initial membrane structure for the computation is $[\,[\,]_1\,[\,]_2\,\cdots\,[\,]_n\,]_0$. We refer to the membrane labeled 0 as the *outer membrane* and refer to the membranes labeled 1 through n as *inner membranes.*

The computation of the P system mainly consists of the following four steps.

Step 1: Move modified copies of input objects into all inner membranes.

Step 2: In each inner membrane labeled h, create an assignment of colors with h colors by dividing the inner membranes. Then, check whether adjacent vertices are colored with different colors with branch and bound. If adjacent vertices are colored with the same color in the assignment, stop assignment for vertices.

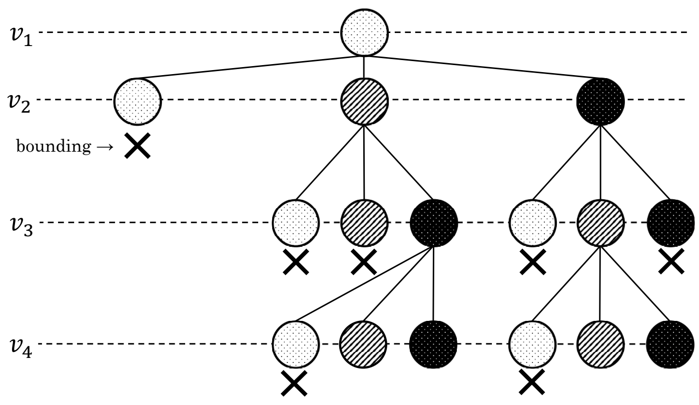

Figure 5: Example of branch and bound for minimum graph coloring

Step 3: Check whether an assignment of colors is completed for all vertices in each divided inner membrane, and then, output an object that denotes successful assignment of colors to the outer membrane without dissolving the inner membrane. Otherwise, dissolve the inner membrane and output an object that denotes failure of the coloring assignment. Then, determine the minimum number of colors in successful assignments in the outer membrane.

Step 4: Send out the minimum number of colors from the outer membrane.

4.3 Details of the proposed P system

We explain details of each step of the computation for minimum graph coloring. In Step 1, modified copies of the input objects are moved into all inner membranes. Each input object is modified so as to contain the label of a stored membrane.

Since the P system considered in the present paper is asynchronous, we cannot move the input objects in parallel, and input objects are moved one by one applying the following sets of evolution rules. (In the following description, a set of evolution rules $R_{i,j}$ indicates that the set of rules is used for membrane i in Step j.)

$$\begin{aligned}
R_{0,1} \quad = \quad & \{\langle e_{i,j}, W\rangle \rightarrow \langle e_{i,j}, W, 1\rangle\langle f_{i,j}, W, 1\rangle \mid 1 \leq i \leq n, 1 \leq j \leq n, W \in \{F, T\}\} \\
& \cup\{\langle f_{i,j}, W, h\rangle \rightarrow \langle e_{i,j}, W, h+1\rangle\langle f_{i,j}, W, h+1\rangle \mid 1 \leq i \leq n, 1 \leq j \leq n, 1 \leq h \leq n-2, \\
& \quad W \in \{F, T\}\} \\
& \cup\{\langle f_{i,j}, W, n-1\rangle \rightarrow \langle e_{i,j}, W, n\rangle \mid 1 \leq i \leq n, 1 \leq j \leq n, W \in \{F, T\}\} \\
& \cup\{\langle e_{1,1}, F, h\rangle[]_h \rightarrow [\langle M_{2,1}, h\rangle\langle e_{1,1}, F, h\rangle]_h \mid 1 \leq h \leq n\} \\
& \cup\{\langle M_{i,j}, h\rangle\langle e_{i,j}, W, h\rangle[]_h \rightarrow [\langle M_{i+1,j}, h\rangle\langle e_{i,j}, W, h\rangle]_h \mid 1 \leq i \leq n, 1 \leq j \leq n, 1 \leq h \leq n, \\
& \quad W \in \{F, T\}\} \\
& \cup\{\langle M_{n+1,j}, h\rangle \rightarrow \langle M_{1,j+1}, h\rangle \mid 1 \leq j \leq n, 1 \leq h \leq n\} \\
& \cup\{\langle M_{1,n+1}, h\rangle[]_h \rightarrow [\langle S_1, h\rangle]_h \mid 1 \leq h \leq n\} \\
R_{h,1} \quad = \quad & \{[\langle M_{i,j}, h\rangle]_h \rightarrow []_h\langle M_{i,j}, h\rangle \mid 1 \leq i \leq n+1, 1 \leq j \leq n+1\}
\end{aligned}$$

In the above evolution rules, two kinds of objects, $\langle e_{i,j}, W, h\rangle$ and $\langle f_{i,j}, W, h\rangle$, are created from input object $\langle e_{i,j}, W\rangle$. The first objects, $\langle e_{i,j}, W, h\rangle$, are moved into a membrane labeled h using objects $\langle M_{i,j}, h\rangle$. At the end of Step 1, object $\langle S_1, h\rangle$ is created in the membrane labeled h, and the object triggers the computation of Step 2.

In Step 2, in each inner membrane labeled h, assignments of colors are created with h colors by dividing the inner membranes. Then, the partial assignment is checked as to whether adjacent vertices are colored with different colors using the bounding condition. If the adjacent vertices are colored with the same color, the membrane stops the partial assignment. The above step is executed by applying the following set of evolution rules.

$$\begin{aligned}
R_{h,2} \;=\; & \{\langle S_1,h\rangle \to \langle S_2,h\rangle\langle V_1,h\rangle\} \\
& \cup\{[\langle S_i,k\rangle]_h \to [\langle L_{i,1}\rangle\langle V_i,k\rangle]_h[\langle S_i,k-1\rangle]_h \mid 2\le i\le n, 2\le k\le h\} \\
& \cup\{\langle S_i,1\rangle \to \langle L_{i,1}\rangle\langle V_i,1\rangle \mid 2\le i\le n\} \\
& \cup\{\langle L_{i,j}\rangle\langle V_i,a\rangle\langle V_j,b\rangle\langle e_{i,j},F,h\rangle\langle e_{j,i},F,h\rangle \to \langle L_{i,j+1}\rangle\langle V_i,a\rangle\langle V_j,b\rangle \mid 2\le i\le n, 1\le j\le i, \\
& \quad 1\le a,b\le h\} \\
& \cup\{\langle L_{i,j}\rangle\langle V_i,a\rangle\langle V_j,b\rangle\langle e_{i,j},T,h\rangle\langle e_{j,i},T,h\rangle \to \langle L_{i,j+1}\rangle\langle V_i,a\rangle\langle V_j,b\rangle \mid 2\le i\le n, 1\le j\le i, \\
& \quad 1\le a,b\le h, a\ne b\} \\
& \cup\{\langle L_{i,j}\rangle\langle V_i,a\rangle\langle V_j,a\rangle\langle e_{i,j},T,h\rangle\langle e_{j,i},T,h\rangle \to \langle Z_h,n-i\rangle\langle G_{i-1}\rangle\langle V_j,a\rangle \mid 2\le i\le n, \\
& \quad 1\le j\le i, 1\le a\le h\} \\
& \cup\{\langle L_{i,i}\rangle \to \langle S_{i+1},h\rangle \mid 2\le i\le n\}
\end{aligned}$$

In the above evolution rules, vertex v_i is colored with color k in the second set of rules, and object $\langle V_i,k\rangle$ denotes the color. Then, the vertices of v_j $(1\le j\le i)$ are checked for v_i using object $\langle L_{i,j}\rangle$. If v_i and v_j are not adjacent or their colors are different, then object $\langle L_{i,j+1}\rangle$ is created. After all checked vertices are passed for v_i, object $\langle L_{i,i}\rangle$ is created, and object $\langle S_{i+1},h\rangle$, which is an object for checking the next vertex v_{i+1}, is created.

On the other hand, for the case in which v_i and v_j are adjacent and are colored with the same color, objects $\langle Z_h,n-i\rangle$ and $\langle G_{i-1}\rangle$, which denote a failure of coloring, are created. The object $\langle Z_h,n-i\rangle$ denotes that $n-i$ colors remain in the case of coloring failure, and $\langle G_{i-1}\rangle$ is an object used in Step 3 for deleting partial assignment of colors in the membrane. In addition, the two objects trigger the computation of Step 3.

In Step 3, the assignment in each membrane is checked as to whether colors are assigned for all vertices. For the case in which an assignment of colors is completed for all vertices, an object that denotes the success of the assignment of colors is sent to the outer membrane without dissolving the inner membrane. Otherwise, the inner membrane is dissolved, and an object that denotes a failure of coloring assignment is output to the outer membrane. Then, the minimum number of colors in successful assignments is determined in the outer membrane. Step 3 is executed by applying the following sets of evolution rules.

$$\begin{aligned}
R_{0,3} \;=\; & \{\langle S_{n+1},h\rangle \to \langle S_{n+1},h,0\rangle \mid 1\le h\le n\} \\
& \cup\{\langle S_{n+1},h,k\rangle^h \to \langle S_{n+1},h,k+1\rangle \mid 1\le h\le n, 0\le k\le n-2\} \\
& \cup\{\langle Z_h,k\rangle^h \to \langle Z_h,k+1\rangle \mid 1\le h\le n, 0\le k\le n-2\} \\
& \cup\{\langle S_{n+1},h,k\rangle^{h-d}\langle Z_h,k\rangle^d \to \langle S_{n+1},h,k+1\rangle \mid 1\le h\le n, 0\le k\le n-2, 1\le d\le h-1\} \\
& \cup\{\langle S_{n+1},h,n-1\rangle\langle Z_{h-1},n-1\rangle \to \langle A_h\rangle \mid 1\le h\le n\} \\
R_{h,3} \;=\; & \{[\langle S_{n+1},h\rangle]_h \to []_h\langle S_{n+1},h\rangle\} \\
& \cup\{\langle V_i,k\rangle\langle G_i\rangle \to \langle G_{i-1}\rangle \mid 2\le i\le n, 1\le k\le h\} \\
& \cup\{[\langle V_1,h\rangle\langle G_1\rangle]_h \to \langle G_0\rangle\}
\end{aligned}$$

In Step 3, $R_{h,3}$ is applied in each divided membrane. For the case in which the object that denotes success of the assignment of colors, $\langle S_{n+1},h\rangle$, is in the membrane, the object is sent to the outer membrane without dissolving the membrane. On the other hand, for the case in which the object that denotes failure of the assignment of colors, $\langle G_i\rangle$, is in the membrane, the membrane is dissolved using $\langle G_i\rangle$, and objects $\langle S_{n+1},h,k\rangle$ and $\langle Z_h,k\rangle$ are in the outer membrane.

The object $\langle S_{n+1},h,k\rangle$ denotes that there are h^k successful assignments of colors with h colors, and the object $\langle Z_h,k\rangle$ denotes that there are h^k failure assignments of colors with h colors. Then, if objects $\langle S_{n+1},h,n-1\rangle$ and $\langle Z_{h-1},n-1\rangle$ exist simultaneously in the outer membrane, coloring with

h colors is successful, and coloring with $h-1$ colors has failed. Therefore, the object $\langle A_h \rangle$, which indicates that the minimum number for assignment of colors is h, is created in the outer membrane, and the object triggers the computation of Step 4.

In Step 4, one of the assignments that denote the minimum coloring is sent from the outer membrane. Step 4 is executed by applying the following sets of evolution rules.

$$\begin{aligned}
R_{0,4} &= \{\langle B_i \rangle \to \langle C_i \rangle \langle B_{i+1} \rangle \mid 1 \leq i \leq n-1\} \\
&\quad \cup \{\langle B_n \rangle \to \langle C_n \rangle\} \\
&\quad \cup \{[\langle C_i \rangle \langle V_i, h \rangle]_0 \to []_0 \langle V_i, h \rangle \mid 1 \leq i \leq n, 1 \leq h \leq n\} \\
R_{h,4} &= \{\langle A_h \rangle []_h \to [\langle A_h \rangle]_h\} \cup \{[\langle A_h \rangle]_h \to []_h \langle B_1 \rangle\}
\end{aligned}$$

The objects $\langle V_i, h \rangle$ are sent out from the outer membrane, and the computation is then finished.

We now summarize the asynchronous P system $\Pi_{\mathrm{MIN\_COL}}$ for finding the minimum graph coloring as follows:

$$\Pi_{\mathrm{MIN\_COL}} = (O, \mu, \omega_1, \omega_2, R_0, R_1, \cdots, R_n)$$

$$\begin{aligned}
O &= \{\langle e_{i,j}, W \rangle \mid 1 \leq i \leq n, 1 \leq j \leq n, W \in \{F, T\}\} \\
&\quad \cup \{\langle e_{i,j}, W, h \rangle \mid 1 \leq i \leq n, 1 \leq j \leq n, 1 \leq h \leq n, W \in \{F, T\}\} \\
&\quad \cup \{\langle f_{i,j}, W, h \rangle \mid 1 \leq i \leq n, 1 \leq j \leq n, 1 \leq h \leq n, W \in \{F, T\}\} \\
&\quad \cup \{\langle M_{i,j}, h \rangle \mid 1 \leq i \leq n, 1 \leq j \leq n, 1 \leq h \leq n\} \\
&\quad \cup \{\langle S_i, h \rangle \mid 1 \leq i \leq n+1, 1 \leq h \leq n\} \\
&\quad \cup \{\langle V_i, h \rangle \mid 1 \leq i \leq n, 1 \leq h \leq n\} \\
&\quad \cup \{\langle L_{i,j} \rangle \mid 1 \leq i \leq n, 1 \leq j \leq n\} \\
&\quad \cup \{\langle Z_h, k \rangle \mid 1 \leq h \leq n, 0 \leq k \leq n-2\} \\
&\quad \cup \{\langle G_i \rangle \mid 0 \leq i \leq n-1\} \\
&\quad \cup \{\langle S_{n+1}, h, k \rangle \mid 1 \leq h \leq n, 0 \leq k \leq n-1\} \\
&\quad \cup \{\langle A_h \rangle \mid 1 \leq h \leq n\} \\
&\quad \cup \{\langle B_i \rangle \mid 1 \leq i \leq n\} \\
&\quad \cup \{\langle C_i \rangle \mid 1 \leq i \leq n\} \\
\mu &= [\,[\,]_1\,[\,]_2\,\cdots\,[\,]_n\,]_0 \\
\omega_1 &= \omega_2 = \cdots = \omega_n = \phi \\
R_0 &= R_{0,1} \cup R_{0,3} \cup R_{0,4} \\
R_h &= R_{h,1} \cup R_{h,2} \cup R_{h,3} \quad (1 \leq h \leq n)
\end{aligned}$$

4.4 Complexity

The complexity of the proposed P system $\Pi_{\mathrm{MIN\_COL}}$ is considered as follows. In Step 1, a set of $O(n^2)$ objects are copied into n sets, and each set is moved from the outer membrane into one of inner membranes. Then, Step 1 works in $O(n^2)$ parallel steps or $O(n^3)$ sequential steps. The size of the evolution rules and the number of objects are both $O(n^3)$.

In Step 2, $O(n^n)$ membranes are created and evolution rules are applied $O(n)$ times in the worst case. Therefore, Step 2 works in $O(n^2)$ parallel steps and $O(n^n)$ sequential steps. The size of the evolution rules and the number of objects are $O(n^4)$ and $O(n^3)$, respectively.

Step 3 is executed in each divided membrane. Therefore, Step 3 works in $O(n)$ parallel steps and $O(n^n)$ sequential steps. The size of the evolution rules and the number of objects are both $O(n^3)$.

In the final step, the computation is executed in the outer membrane, and Step 4 works in $O(n)$ parallel or sequential steps. The number of objects is $O(n)$, and the size of the evolution rules is $O(n^2)$.

From the above discussion, the following theorem is obtained for the proposed asynchronous P system $\Pi_{\mathrm{MIN\_COL}}$.

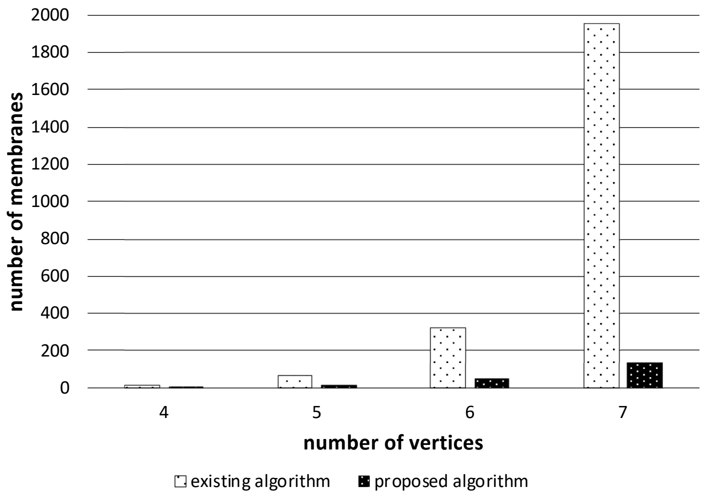

Figure 6: Experimental results for Hamiltonian cycle

Theorem 2 *The asynchronous P system $\Pi_{\text{MIN\_COL}}$, which solves the minimum graph coloring for a graph with n vertices, works in $O(n^n)$ sequential steps or $O(n^2)$ parallel steps by using $O(n^3)$ kinds of objects and evolution rules of size $O(n^4)$.* □

5 Experimental simulations

Although asymptotic complexities of the proposed two P systems are the same as the complexity of the existing P systems [12, 13], the number of membranes can be reduced by the branch and bound. We discuss concrete reduction rate of the number of membranes in this section using experimental simulations.

Existing P systems [12, 13] and our proposed P system for two graph problems are implemented on our original simulator of P systems programmed in Python.

Since the simulator executes P systems with asynchronous parallelism, all of the evolution rules are applied in fully asynchronous manner, in other words, any number of applicable evolution rules are applied in each step of executions on the simulator. Therefore, applied evolution rules are different among executions on the simulator, and output of the simulation may be different for the same input. We first implement the existing P systems and the proposed P systems for the two graph problems on the simulator, and execute simulations for various inputs. In the simulation, valid results are obtained for all inputs, that is, all outputs are the same for the same input on all executions.

Next, we compare the numbers of membranes used on the existing P systems and the proposed P systems for the two graph problems. For each number of vertices, five random graphs are created so that each edge is connected with probability $\frac{1}{2}$, and the same input graphs are given to the existing P systems and the proposed P systems.

Fig. 6 and Fig. 7 show average values of the number of membranes for Hamiltonian cycle and the minimum graph coloring. The numbers of membranes of the existing P systems increase exponentially with respect to the number of vertices n. Although the numbers of membranes in the proposed P systems also increases according to n, the numbers of membranes on the proposed P systems are more than 92 percent and 72 percent less than the numbers of membranes on the existing P systems for Hamiltonian cycle and the graph coloring, respectively.

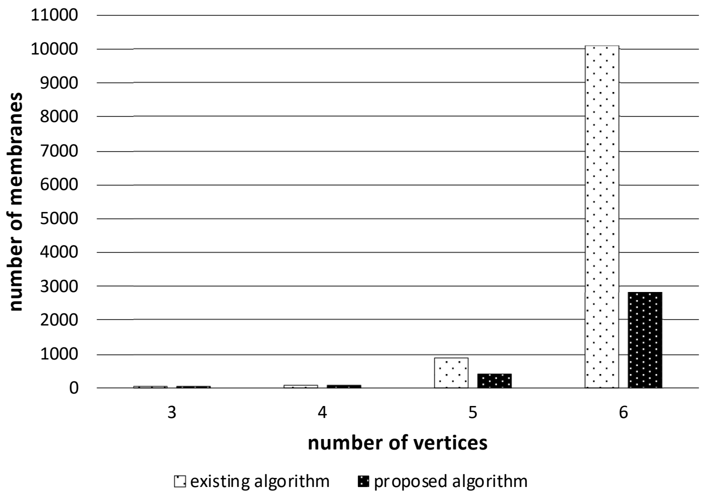

Figure 7: Experimental results for minimum graph coloring

6 Conclusions

In this paper, we proposed two asynchronous P systems with branch and bound for two computationally hard graph problems. The first P system with branch and bound for Hamiltonian cycle reduces the number of membranes by discarding partial permutations that cannot be Hamiltonian cycle. The second P system with branch and bound for the minimum graph coloring reduces the number of membranes by discarding partial assignments of colors in which the colors of neighboring vertices are the same.

The proposed P systems are fully asynchronous and works in a polynomial number of steps in a maximally parallel manner and also works sequentially. Although the number of sequential steps is exponential, the result indicates that the proposed P systems work for any combination of sequential and asynchronous applications of evolution rules and guarantee that correct solutions are output for any case in which any number of evolution rules are synchronized.

We showed that the proposed P systems outputs valid results, and also showed that the number of membranes in the proposed P systems are effectively less than the number of membranes in the existing P systems for Hamiltonian cycle and the minimum graph coloring.

In our future research, we intend to consider a reduction in the number of objects and the size of evolution rules used in the proposed P systems. We also intend to consider asynchronous P systems with branch and bound for other computationally hard problems.

Acknowledgments

The work was supported by JSPS KAKENHI Grant Number 16K00021. The authors would also like to thank the anonymous reviewers for giving us detailed comments that helped us improve quality of the paper.

References

[1] R. Freund. Asynchronous P systems and P systems working in the sequential mode. In *International workshop on Membrane Computing*, pages 36–62, 2005.

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

[3] J. Imatomi and A. Fujiwara. An asynchronous P system for MAX-SAT. In *8th International Workshop on Parallel and Distributed Algorithms and Applications*, pages 572–578, 2016.

[4] Y. Jimen and A. Fujiwara. Asynchronous P systems for solving the satisfiability problem. *International Journal of Networking and Computing*, 8(2):141–152, 2018.

[5] A. Riscos-Núnez M. J. Pérez-Jiménez. A linear-time solution to the knapsack problem using P systems with active membranes. *Proc. International Workshop on Membrane Computing*, pages 250–268, 2003.

[6] L. Q. Pan and A. Alhazov. Solving HPP and SAT by P systems with active membranes and separation rules. *Acta Informatica*, 43(2):131–145, 2006.

[7] G. Păun. Computing with membranes. *Journal of Computer and System Sciences*, 61(1):108–143, 2000.

[8] G. Păun. P systems with active membranes: Attacking NP-complete problems. *Journal of Automata, Languages and Combinatorics*, 6(1):75–90, 2001.

[9] M. J. Pérez-Jiménez and A. Riscos-Núñez. Solving the subset-sum problem by P systems with active membranes. *New Generation Computing*, 23(4):339–356, 2005.

[10] M. J. Pérez-Jiménez and F.J. Romero-Campero. Solving the BIN PACKING problem by recognizer P systems with active membranes. In *The Second Brainstorming Week on Membrane Computing*, pages 414–430, 2004.

[11] M. J. Pérez-Jiménez, A. Romero-Jiménez, and F. Sancho-Caparrini. A polynomial complexity class in P systems using membrane division. *Journal of Automata, Languages and Combinatorics*, 11(4):423–434, 2003.

[12] H. Tagawa and A. Fujiwara. Solving SAT and Hamiltonian cycle problem using asynchronous p systems. *IEICE Transactions on Information and Systems (Special section on Foundations of Computer Science)*, E95-D(3), 2012.

[13] K. Tanaka and A. Fujiwara. Asynchronous P systems for hard graph problems. *International Journal of Networking and Computing*, 4(1):2–22, 2014.

[14] C. Zandron, C. Ferretti, and G. Mauri. Solving NP-complete problems using P systems with active membranes. In *Unconventional Models of Computation*, pages 289–301, 2000.