International Journal of Networking and Computing – www.ijnc.org, ISSN 2185-2847
Volume 14, Number 2, pages 248–261, July 2024

An asynchronous P system with branch and bound for the minimum Steiner tree

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

Received: February 15, 2024
Revised: May 2, 2024
Accepted: June 10, 2024
Communicated by Junya Nakamura

Abstract

Membrane computing, also known as a P system, is a computational model inspired by the activity of living cells. P systems work in a polynomial number of steps, and several have been proposed for solving computationally hard problems. However, most of the proposed algorithms use an exponential number of membranes, and reduction in the number of membranes must be considered in order to make a P system a more realistic model.

In the present paper, we propose an asynchronous P system using branch and bound to solve the minimum Steiner tree problem. The proposed P system solves for the minimum Steiner tree with n vertices and m edges in $O(n^2)$ parallel steps or $O(2^m n^2)$ sequential steps.

We evaluate the number of membranes used in the proposed P system through experimental simulations. Our experimental results show the validity and efficiency of the proposed P system.

Keywords: membrane computing, asynchronous P system, minimum Steiner tree

1 Introduction

Membrane computing, which was introduced in [10] as a P system, is a computational model inspired by the activity of living cells. A P system consists of membranes and objects, which represent computing cells and data storage, respectively. In a P system, each object evolves according to certain evolution rules associated with the membrane.

Since an exponential number of membranes can be created in a polynomial number of steps using a division rule, which is one of evolution rules in a P system, a computationally hard problem can be solved in a polynomial number of steps. Using this feature, a number of P systems have been proposed for solving computationally hard problems [4, 5, 9, 11, 12, 14, 16].

In addition, asynchronous parallelism, which assumes asynchronous application of evolution rules, has been considered for P systems. Asynchronous parallelism means that different objects may react to rules at different speeds on a P system. Asynchronous parallelism makes a P system a more realistic computational model because living cells work independently according to the environment. Using asynchronous parallelism, a number of asynchronous P systems have been proposed for computationally hard problems [1, 12, 13].

However, computationally hard problems have been solved in polynomial numbers of steps using exponential numbers of membranes in all of the abovementioned P systems. The exponential number

of membranes corresponds to the number of living cells, which must be reduced in the case of implementing a P system using living cells because the living cells cannot be created exponentially.

Recently, a number of P systems [2, 6, 7, 8, 15] have been proposed for reducing the number of membranes. For example, a P system for the satisfiability problem (SAT) with a DPLL algorithm, which is a well-known search algorithm for SAT, was proposed in [7]. The proposed P system preferentially assigns a variable in the partial assignment using three rules: a one-literal rule, a pure literal rule, and a splitting rule. Using these three rules repeatedly and intensively, the number of membranes that contain partial assignment can be decreased.

In the present paper, we propose an asynchronous P system for solving the minimum Steiner tree problem. Since the minimum Steiner tree is a recently known computationally hard problem, no optimization technique is known in the membrane computing. Our proposed P system is based on the branch and bound, which is one of optimization techniques for eliminating invalid candidate solutions. We show that the proposed P system works in $O(2^m n^2)$ sequential steps or $O(n^2)$ parallel steps using $O(n^4)$ kinds of objects.

We evaluate the number of membranes used in the proposed P system through experimental simulations. The results show that the numbers of membranes used in the proposed P system are 57% less than the numbers of the membranes obtained by an exhaustive search on average.

The remainder of the paper is organized as follows. In Section 2, we describe the computational model for the membrane computing and definition of the minimum Steiner tree. In Section 3, we propose a P system with branch and bound for the minimum Steiner tree problem and consider the complexity of the P system. In Section 4, we show experimental results for the proposed P system. Finally, Section 5 concludes the paper.

2 Preliminaries

In the present paper, we propose an asynchronous P system for the minimum Steiner tree problem. We briefly define a P system and a minimum Steiner tree.

2.1 Computational model for membrane computing

A P system consists mainly of membranes and objects. A membrane is a computing cell in the P system and may contain objects and other membranes. Each membrane is initially labeled with a distinct integer. An object represents a memory cell that stores data in the P system. According to the evolution rules for the corresponding membrane, objects may evolve into other objects or pass through membranes. Objects may also divide or dissolve membranes in which the objects are stored. We assume that each object is a finite string over a given set of alphabetic characters.

As an example of membranes and objects, the following expression defines a membrane structure comprising two membranes and three objects.

$$[\ [\ \alpha\ ]_2\ [\ \beta\ \gamma\ ]_3\ ]_1$$

In this example, the membrane labeled 1 contains two membranes, labeled 2 and 3, which in turn contain sets of objects $\{\alpha\}$ and $\{\beta, \gamma\}$, respectively.

The computation of P systems is governed by a number of evolution rules. Each evolution rule is a rule for updating membranes and objects. According to the applicable evolution rules, objects and membranes are transformed in parallel in every step of the computation. The system stops the computation if there are no applicable evolution rules for the objects.

Various types of evolution rules are assumed in membrane computing. In the present paper, we assume the following five rules from [4]:

(1) Object evolution rule:

$$[\ \alpha\ ]_h \rightarrow [\ \beta\ ]_h$$

Object α is transformed into object β.

(2) Send-in communication rule:

$$\alpha\ [\ ]_h \rightarrow [\ \beta\ ]_h$$

Object α is moved into inner membrane h and is transformed into object β.

(3) Send-out communication rule:

$$[\ \alpha\ ]_h \rightarrow [\ ]_h\ \beta$$

Object α is sent out from membrane h and is transformed into object β.

(4) Dissolution rule:

$$[\ \alpha\ ]_h \rightarrow \beta$$

The membrane that contains object α is dissolved, and object α is transformed into object β. (Note that the outermost membrane cannot be dissolved.)

(5) Division rule:

$$[\ \alpha\ ]_h \rightarrow [\ \beta\ ]_h[\ \gamma\ ]_h$$

The membrane that contains object α is divided into two membranes with the same label, and object α is transformed into other objects, β and γ, each in one of the created membranes.

The P system consists of the following six components:

O**:** the set of objects used in the system,

μ**:** the structure of the membrane,

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

R_i**:** the set of evolution rules for the membrane labeled i,

i_{in}**:** the label of the input membrane, and

i_{out}**:** the label of the output membrane

Using the above components, a P system Π with m membranes is defined as follows:

$$\Pi = (O, \mu, \omega_1, \omega_2, \cdots, \omega_m, R_1, R_2, \cdots, R_m, i_{in}, i_{out})$$

Under the assumption that each of the evolution rules can be applied in a single step in the computational model, the complexity of the P system is defined as the number of steps executed.

In the present paper, we consider asynchronous parallelism [12] in the P system. Under the assumption of 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 the computation. We refer to the numbers of steps in the former and latter cases as the numbers of parallel and sequential steps, respectively. The number of parallel steps is the complexity of the P system in the best case, and the number of sequential steps is the complexity in the worst case.

2.2 Minimum Steiner tree

A Steiner tree for an undirected graph $G = (V, E)$ is defined as a tree that involves a given subset T of vertices, which is called a set of terminals. The minimum Steiner tree problem is that of computing a minimum cost Steiner tree for nontrivial T, and this problem is known to be NP-complete[3]. In the case that the number of vertices in T is 2, the minimum Steiner tree is equivalent to a shortest path connecting the two vertices. On the other hand, in the case of $T = V$, the minimum Steiner tree is equivalent to a spanning tree.

Figure 1 shows an example for a minimum Steiner tree. In the input graph, we assume that $T = \{v_0, v_2\}$ and $e_{i,j}$ denotes the edge between v_i and v_j. Then, eight trees, whose sets of edges are $\{e_{0,1}, e_{0,2}, e_{2,3}\}$, $\{e_{0,1}, e_{1,2}, e_{2,3}\}$, $\{e_{0,2}, e_{1,2}, e_{2,3}\}$, $\{e_{0,1}, e_{0,2}\}$, $\{e_{0,1}, e_{1,2}\}$, $\{e_{0,2}, e_{1,2}\}$, $\{e_{0,2}, e_{2,3}\}$, and $\{e_{0,2}\}$, are Steiner trees for the input graph, and the tree with the set of edges $\{e_{0,2}\}$ is the minimum Steiner tree for this case.

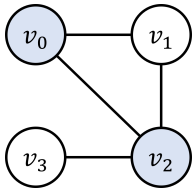

Figure 1: Example of an input graph

3 Asynchronous P system with branch and bound for the minimum Steiner tree

In this section, we present an asynchronous P system with branch and bound for solving the minimum Steiner tree problem. We first show an encoding for the input and the output for the P system, and then show an outline and details of the P system. Finally, we show an example of execution of the proposed P system, and discuss complexity of the proposed P system.

3.1 Input and output for the minimum Steiner tree

We first show an encoding for the input and output for the P system. We assume that the input for the minimum Steiner tree is an undirected graph $G = (V, E)$ with n vertices and m edges.

The input is given by the following set of objects O_{in} in the P system:

$$\begin{aligned} O_{in} \quad = \quad & \{\langle e_{i,j}, B\rangle \mid 0 \leq i < j \leq n-1, B \in \{T, F\}\} \\ & \cup \{\langle v_k, B\rangle \mid 0 \leq k \leq n-1, B \in \{T, F\}\} \end{aligned}$$

In the above set of input objects, an edge (v_i, v_j) is represented by the object $\langle e_{i,j}, B\rangle$. If edge (v_i, v_j) is in the graph, then B in $\langle e_{i,j}, B\rangle$ is set to T, otherwise, B is set to F. In addition, vertex v_k is represented by object $\langle v_k, B\rangle$. If vertex v_k is one of the terminals, then B in $\langle v_k, B\rangle$ is set to T, otherwise, B is set to F.

For example, the following set of objects is given as an input of the P system, O_{in}, for a graph in Fig. 1.

$$\begin{aligned} O_{in} = & \{\langle e_{0,1}, T\rangle, \langle e_{0,2}, T\rangle, \langle e_{0,3}, F\rangle, \langle e_{1,2}, T\rangle, \langle e_{1,3}, F\rangle, \langle e_{2,3}, T\rangle\} \\ & \cup \{\langle v_0, T\rangle, \langle v_1, F\rangle, \langle v_2, T\rangle, \langle v_3, F\rangle\} \end{aligned}$$

We assume that a computation on the P system starts if the above O_{in} is given as input from the outside region into the outermost membrane.

An output of the problem is a set of edges, which represents the minimum tree including terminals. The output of the P system is the following set of objects, which represents a subset of edges:

$$O_{out} = \{\langle e_{i,j}\rangle \mid 0 \leq i < j \leq n-1\}$$

Object $\langle e_{i,j}\rangle$ means that the solution includes an edge $e_{i,j}$.

For example, the following is an output of the P system, which represents the minimum Steiner tree for the graph in Fig. 1.

$$O_{out} = \{\langle e_{0,2}\rangle\}$$

3.2 Branch and bound for the minimum Steiner tree

Branch and bound is a well-known computing paradigm for optimization problems. All candidate solutions are enumerated for m edges to solve the minimum Steiner tree, and 2^m solutions must be created for an exhaustive enumeration. In other words, each edge is examined in a Steiner tree or

not in each partial assignment, which is a set of selected edges for the solution. However, a number of partial assignments can be discarded if all adjacent edges of a terminal are assumed not to be in the partial assignment.

We now give an overview of the asynchronous P system with branch and bound for solving the minimum Steiner tree problem. The proposed P system consists of two membranes $[\,[\,]_2\,]_1$, i.e., an inner membrane labeled 2 contained in an outer membrane labeled 1.

The P system consists of the following 4 steps.

Step 1: Move all input objects in the outer membrane into the inner membrane.

Step 2: In each inner membrane, repeat the following until all edges are examined.

- Divide each inner membrane into two membranes: one that includes an edge and the other that does not.
- Check whether there exists a terminal such that all adjacent edges are assumed not to be in the partial assignment. If such a terminal exists, stop the repetition for the inner membrane.

Step 3: Confirm a tree condition of the graph in each inner membrane is a tree by executing the following (3-1) and (3-2).

(3-1) Confirm whether the number of edges in the graph is equal to the number of vertices minus one.

(3-2) Confirm whether the graph is acyclic and connected using a depth-first search (DFS) technique.

Step 4: Select a membrane representing an optimal solution, and output edges of the solution.

3.3 Details of the P system

We now explain the details of each step of the computation.

Step 1: All input objects are moved into the inner membrane. This step is executed using the sets of evolution rules $R_{1,1}$ and $R_{2,1}$ in Fig. 2.

The computation using the sets of evolution rules is executed as follows. The object $\langle Start\rangle$ is given as ω_1, which is the set of objects initially contained in the inner membrane, and generates objects $\langle V_0\rangle$. Input objects representing vertices $\langle v_i, B\rangle$ in the outer membrane are moved into the inner membrane using object $\langle V_i\rangle$. Then, objects representing degrees of each vertex $\langle K_i, 0\rangle$ are generated. After that, input objects representing edges $\langle e_{i,j}, B\rangle$ in the outer membrane are moved into the inner membrane using object $\langle E_{i,j}\rangle$. Then, if B of an added edge is T, then the degree of the edge $\langle K_i, k_i\rangle$ is incremented in the inner membrane using object $\langle EI_{i,j}\rangle$. After all objects representing vertices and edges are moved, the objects $\langle DIV_{0,1}\rangle$, $\langle NV_n\rangle$, $\langle NE_m\rangle$, and $\langle DC, 0\rangle$ are generated for Step 2.

Step 2: First, each inner membrane is divided into two membranes: one that includes an edge and the other that does not. Next, a check is executed whether there exists a terminal such that all adjacent edges are assumed not to be in the partial assignment. If there is such a terminal, then the division is stopped for the inner membrane, otherwise, the division is repeated for the divided membrane. This step is executed using the set of evolution rules $R_{2,2}$ in Fig. 3.

The computation using the set of evolution rules is executed as follows. First, the inner membrane is divided into two membranes: one includes $\langle e_{i,j}, B\rangle$ and the other does not. In the same way, the membranes are divided repeatedly for each edge. An object $\langle DC, dc\rangle$ represents the number of divisions of the membrane and is incremented every time the membrane is divided.

In the case of the division, objects $\langle K_i, k_i\rangle$ and $\langle K_j, k_j\rangle$, which represent the degrees of $\langle v_i, B\rangle$ and $\langle v_j, B\rangle$, are decremented in the membrane that does not involve $\langle e_{i,j}, B\rangle$. Simultaneously, an object $\langle de_{i,j}\rangle$, which represents an excluded edge, is created.

(Evolution rules for outer membrane)

$$
\begin{aligned}
R_{1,1} \quad = \quad & \{\langle Start\rangle \to \langle V_0\rangle\} \\
& \cup\{\langle V_i\rangle\langle v_i, B\rangle[]_2 \to [\langle V_i\rangle\langle v_i, B\rangle\langle K_i, 0\rangle]_2 \ \mid 0 \le i \le n-1, B \in \{T, F\}\} \\
& \cup\{\langle V_n\rangle \to \langle E_{0,1}\rangle\} \\
& \cup\{\langle E_{i,j}\rangle\langle e_{i,j}, T\rangle[]_2 \to [\langle EI_{i,j}\rangle\langle e_{i,j}, T\rangle]_2 \ \mid 0 \le i < j \le n-1\} \\
& \cup\{\langle E_{i,j}\rangle\langle e_{i,j}, F\rangle[]_2 \to [\langle E_{i,j}\rangle\langle e_{i,j}, F\rangle]_2 \ \mid 0 \le i < j \le n-1\} \\
& \cup\{\langle E_{i,n}\rangle \to \langle E_{i+1,i+2}\rangle \ \mid 0 \le i \le n-2\} \\
& \cup\{\langle E_{n-1,n}\rangle[]_2 \to [\langle DIV_{0,1}\rangle\langle NV_n\rangle\langle NE_m\rangle\langle DC, 0\rangle]_2\}
\end{aligned}
$$

(Evolution rules for inner membranes)

$$
\begin{aligned}
R_{2,1} \quad = \quad & \{[\langle V_i\rangle]_2 \to []_2\langle V_{i+1}\rangle \ \mid 1 \le i \le n-1\} \\
& \cup\{\langle EI_{i,j}\rangle\langle K_i, k_i\rangle\langle K_j, k_j\rangle \to \langle E_{i,j}\rangle\langle K_i, k_i+1\rangle\langle K_j, k_j+1\rangle \\
& \qquad \mid 1 \le i < j \le n, 0 \le k_i \le n-2, 0 \le k_j \le n-2\} \\
& \cup\{[\langle E_{i,j}\rangle]_2 \to []_2\langle E_{i,j+1}\rangle \ \mid 1 \le i < j \le n-1\}
\end{aligned}
$$

Figure 2: Evolution rules for Step 1

(Evolution rules for inner membranes)

$$
\begin{aligned}
R_{2,2} \quad = \quad & \{[\langle DIV_{i,j}\rangle\langle NE_{ne}\rangle\langle K_i, k_i\rangle\langle K_j, k_j\rangle\langle e_{i,j}, T\rangle\langle DC, dc\rangle]_2 \\
& \quad \to [\langle DIV_{i,j+1}\rangle\langle NE_{ne}\rangle\langle K_i, k_i\rangle\langle K_j, k_j\rangle\langle e_{i,j}, T\rangle\langle DC, dc+1\rangle]_2 \\
& \qquad [\langle DIV_{i,j}, D\rangle\langle NE_{ne-1}\rangle\langle K_i, k_i-1\rangle\langle K_j, k_j-1\rangle\langle de_{i,j}\rangle\langle DC, dc+1\rangle]_2 \\
& \qquad \mid 0 \le i < j \le n-1, 1 \le ne \le m, 1 \le k_i \le n-1, 1 \le k_j \le n-1, 0 \le dc \le m-1\} \\
& \cup\{\langle DIV_{i,j}\rangle\langle e_{i,j}, F\rangle \to \langle DIV_{i,j+1}\rangle\langle de_{i,j}\rangle \ \mid 0 \le i < j \le n-1\} \\
& \cup\{\langle DIV_{i,j}, D\rangle\langle K_i, 0\rangle\langle v_i, T\rangle \to \langle DIV, F\rangle \ \mid 0 \le i < j \le n-1\} \\
& \cup\{\langle DIV_{i,j}, D\rangle\langle K_j, 0\rangle\langle v_j, T\rangle \to \langle DIV, F\rangle \ \mid 0 \le i < j \le n-1\} \\
& \cup\{\langle DIV_{i,j}, D\rangle\langle K_i, 0\rangle\langle v_i, F\rangle\langle K_j, 0\rangle\langle v_j, F\rangle\langle NV_{nv}\rangle \to \langle DIV_{i,j+1}\rangle\langle NV_{nv-2}\rangle \\
& \qquad \mid 0 \le i < j \le n-1, 2 \le nv \le n\} \\
& \cup\{\langle DIV_{i,j}, D\rangle\langle K_i, 0\rangle\langle v_i, F\rangle\langle K_j, k_j\rangle\langle NV_{nv}\rangle \to \langle DIV_{i,j+1}\rangle\langle K_j, k_j\rangle\langle NV_{nv-1}\rangle \\
& \qquad \mid 0 \le i < j \le n-1, 2 \le nv \le n, 1 \le k_j \le n-1, B_j \in \{T, F\}\} \\
& \cup\{\langle DIV_{i,j}, D\rangle\langle K_i, k_i\rangle\langle K_j, 0\rangle\langle v_j, F\rangle\langle NV_{nv}\rangle \to \langle DIV_{i,j+1}\rangle\langle K_i, k_i\rangle\langle NV_{nv-1}\rangle \\
& \qquad \mid 0 \le i < j \le n-1, 2 \le nv \le n, 1 \le k_i \le n-1, B_i \in \{T, F\}\} \\
& \cup\{\langle DIV_{i,j}, D\rangle\langle K_i, k_i\rangle\langle K_j, k_j\rangle \to \langle DIV_{i,j+1}\rangle\langle K_i, k_i\rangle\langle K_j, k_j\rangle \\
& \qquad \mid 0 \le i < j \le n-1, 1 \le k_i \le n-1, 1 \le k_j \le n-1\} \\
& \cup\{\langle DIV, F\rangle\langle DC, dc\rangle \to \langle OP_{NG}, 2^{m-dc}\rangle \ \mid 0 \le i < j \le n-1, 0 \le dc \le m\} \\
& \cup\{\langle DIV_{i,n}\rangle \to \langle DIV_{i+1,i+2}\rangle \ \mid 0 \le i \le n-1\} \\
& \cup\{\langle DIV_{n,n+1}\rangle \to \langle VR_1\rangle\}
\end{aligned}
$$

Figure 3: Evolution rules for Step 2

After each division, object $\langle DIV_{i,j}, D\rangle$ confirms whether the graph represented by the membrane involves terminals, and then objects representing the numbers of edges and vertices in the membrane, $\langle NE_{ne}\rangle$ and $\langle NV_{nv}\rangle$, are decremented as necessary.

Object $\langle DIV, F\rangle$ indicates the membrane is not appropriate as a Steiner tree, because a terminal exists such that no adjacent edge is in the graph in the membrane. Division of the membrane involving the object is stopped and the membrane is dissolved in Step 4. Once each membrane has finished all its divisions, object $\langle VR_1\rangle$ is generated for Step 3.

Step 3: A graph in each membrane is confirmed to be a tree by executing the following (3-1) and (3-2) in Step 3. In (3-1), the condition that the number of edges is equal to the number of vertices minus one is confirmed. In (3-2), the condition that a graph in the membrane is acyclic and connected is checked using a DFS technique.

First, (3-1) is executed using the set of evolution rules in Fig. 4. The computation using the evolution rules is executed as follows. The condition that the number of edges is equal to the number of vertices minus one is confirmed using objects $\langle VR_1\rangle$, $\langle NV_{nv}\rangle$, and $\langle NE_{ne}\rangle$. Object $\langle OP_{NG}\rangle$ indicates that the membrane does not satisfy the condition, and the membrane is dissolved in Step 4. If the membrane satisfies the condition, then object $\langle VR_2\rangle$ is generated for (3-2).

Next, (3-2) is executed using the set of evolution rules in Fig. 5. The computation using the evolution rules in Fig. 5 is executed as follows. First, a start vertex is selected at random from a set of vertices $\{v_0, v_1, \cdots v_{n-1}\}$. $\langle Start_i\rangle$ indicates that vertex v_i is a vertex for starting DFS, and $\langle R_m\rangle$ represents the number of unexplored edges. DFS takes place using objects $\langle DFS_{i,j}\rangle$, where object $\langle DFS_{i,j}\rangle$ represents an exploration from vertex v_i to v_j. Objects $\langle sv_i\rangle$ and $\langle se_{i,j}\rangle$ represent explored vertices and edges, respectively. In case that the exploration is impossible from v_i to v_j, that is, there is no edge between v_i and v_j, the fourth or fifth rule in Fig. 5 is applied, and object $\langle DFS_{i,j+1}\rangle$, which denote an exploration to v_{j+1}, is generated. In another case, if a cycle is found in the exploration, the sixth or seventh rule is applied. If all adjacent edges are checked for v_i, object $\langle DFS_{i,n}\rangle$ is generated, and the object triggers backtracking from v_i.

Once the whole graph has been explored and confirmed to be a tree, object $\langle OP_1\rangle$ is generated for Step 4.

Step 4: A membrane representing an optimal solution is selected, and the edges of the solution are outputted. This step is executed using sets of evolution rules, $R_{1,4}$ and $R_{2,4}$, in Fig. 6.

The computation using evolution rules in Fig. 6 is executed as follows. Objects representing each weight of the membrane $\langle W_i\rangle$ are moved to an outer membrane and changed into $\langle W_i, 1\rangle$ using objects $\langle OP_1\rangle$ and $\langle OP_2\rangle$. The variable k in $\langle W_i, k\rangle$ represents the number of weights already compared. The weights are compared to each other in the outer membrane, and all object representing weights are dissolved except for an object representing the smallest weight. Once the comparison of weights for all membranes are finished, that is, $\langle W_i, 2^m\rangle$ is generated, object $\langle Output_i\rangle$, where i represents the weight of the membrane whose weight is the smallest, is generated. The object $\langle Output_i\rangle$ is moved into an inner membrane randomly, and if the optimal weight is equal to the weight of the membrane, then the edges in the membrane are outputted using objects $\langle OutE_i\rangle$, otherwise, the membrane is dissolved, and $\langle Output_i\rangle$ is moved into another inner membrane.

We summarize the asynchronous P system Π_{MST} as follows.

$$\Pi_{\mathrm{MST}} = (O, \mu, \omega_1, \omega_2, R_1, R_2, R_{3-1}, R_{3-2}, R_4, i_{in}, i_{out})$$

$$\begin{aligned} O \;=\; & \{\langle Start\rangle\} \\ & \cup\{\langle e_{i,j}, B\rangle \mid 0 \le i < j \le n-1, B \in \{T, F\}\} \cup \{\langle de_{i,j}\rangle \mid 0 \le i < j \le n-1\} \end{aligned}$$

(Evolution rules for inner membranes)

$$
\begin{aligned}
R_{2,3\text{-}1} \quad = \quad & \{\langle VR_1\rangle\langle NE_{i-1}\rangle\langle NV_i\rangle \to \langle VR_2\rangle\langle NE_{i-1}\rangle\langle NV_i\rangle\} \mid 2 \le i \le n\} \\
& \cup\{\langle VR_1\rangle\langle NV_i\rangle\langle NE_j\rangle \to \langle OP_{NG}\rangle\langle NV_i\rangle\langle NE_j\rangle\} \mid 0 \le i \le n, 0 \le j \le m, j \ne i-1\}
\end{aligned}
$$

Figure 4: Evolution rules for (3-1)

(Evolution rules for inner membranes)

$$
\begin{aligned}
R_{2,3\text{-}2} \quad = \quad & \{\langle VR_2\rangle\langle NE_{ne}\rangle\langle v_i, B\rangle \to \langle DFS_{i,0}\rangle\langle NE_{ne}\rangle\langle R_{ne}\rangle\langle sv_i\rangle\langle Start_i\rangle \\
& \quad \mid 0 \le i \le n-1, 1 \le ne \le m, B \in \{T, F\}\} \\
& \cup\{\langle DFS_{i,j}\rangle\langle R_r\rangle\langle e_{i,j}, T\rangle\langle v_j, B_j\rangle \to \langle DFS_{j,0}\rangle\langle R_{r-1}\rangle\langle de_{i,j}\rangle\langle se_{i,j}\rangle\langle sv_j\rangle \\
& \quad \mid 0 \le i < j \le n-1, 1 \le r \le m, B_j \in \{T, F\}\} \\
& \cup\{\langle DFS_{j,i}\rangle\langle R_r\rangle\langle e_{i,j}, T\rangle\langle v_i, B_i\rangle \to \langle DFS_{i,0}\rangle\langle R_{r-1}\rangle\langle de_{i,j}\rangle\langle se_{i,j}\rangle\langle sv_i\rangle \\
& \quad \mid 0 \le i < j \le n-1, 1 \le r \le m, B_i \in \{T, F\}\} \\
& \cup\{\langle DFS_{i,j}\rangle\langle de_{i,j}\rangle \to \langle DFS_{i,j+1}\rangle\langle de_{i,j}\rangle \mid 0 \le i < j \le n-1\} \\
& \cup\{\langle DFS_{j,i}\rangle\langle de_{i,j}\rangle \to \langle DFS_{j,i+1}\rangle\langle de_{i,j}\rangle \mid 0 \le i < j \le n-1\} \\
& \cup\{\langle DFS_{i,j}\rangle\langle e_{i,j}, T\rangle\langle sv_j\rangle \to \langle OP_{NG}\rangle \mid 0 \le i < j \le n-1\} \\
& \cup\{\langle DFS_{j,i}\rangle\langle e_{i,j}, T\rangle\langle sv_i\rangle \to \langle OP_{NG}\rangle \mid 0 \le i < j \le n-1\} \\
& \cup\{\langle DFS_{i,i}\rangle \to \langle DFS_{i,i+1}\rangle \mid 0 \le i \le n-1\} \\
& \cup\{\langle DFS_{j,n}\rangle\langle sv_i\rangle\langle sv_j\rangle\langle se_{i,j}\rangle \to \langle DFS_{i,0}\rangle\langle sv_i\rangle\langle sv_j\rangle\langle te_{i,j}\rangle \mid 0 \le i < j \le n-1\} \\
& \cup\{\langle DFS_{i,n}\rangle\langle sv_i\rangle\langle sv_j\rangle\langle se_{i,j}\rangle \to \langle DFS_{j,0}\rangle\langle sv_i\rangle\langle sv_j\rangle\langle te_{i,j}\rangle \mid 0 \le i < j \le n-1\} \\
& \cup\{\langle DFS_{i,n}\rangle\langle R_0\rangle\langle Start_i\rangle \to \langle OP_1\rangle \mid 0 \le i \le n-1\} \\
& \cup\{\langle DFS_{i,n}\rangle\langle R_r\rangle\langle Start_i\rangle \to \langle OP_{NG}\rangle \mid 0 \le i \le n-1, 1 \le r \le m\}
\end{aligned}
$$

Figure 5: Evolution rules for (3-2)

$$
\begin{aligned}
& \cup\{\langle se_{i,j}\rangle \mid 0 \le i < j \le n-1\} \cup \{\langle e_{i,j}\rangle \mid 0 \le i < j \le n-1\} \cup \{\langle te_{i,j}\rangle \mid 0 \le i < j \le n-1\} \\
& \cup\{\langle v_i, B\rangle \mid 0 \le i \le n-1, B \in \{T, F\}\} \cup \{\langle sv_i\rangle \mid 0 \le i \le n-1\} \cup \{\langle V_i\rangle \mid 0 \le i \le n\} \\
& \cup\{\langle DC_i\rangle \mid 0 \le i \le m\} \cup \{\langle E_{i,j}\rangle \mid 0 \le i < j \le n+1\} \cup \{\langle EI_{i,j}\rangle \mid 0 \le i < j \le n\} \\
& \cup\{\langle Eout_{i,j}\rangle \mid 0 \le i < j \le n\} \cup \{\langle NV_{nv}\rangle \mid 0 \le nv \le n\} \cup \{\langle NE_{ne}\rangle \mid 0 \le ne \le m\} \\
& \cup\{\langle DIV_{i,j}\rangle \mid 0 \le i < j \le n+1\} \cup \{\langle DIV_{i,j}, D\rangle \mid 0 \le i < j \le n-1\} \cup \{\langle DIV, F\rangle\} \\
& \cup\{\langle K_i, k_i\rangle \mid 0 \le i \le n-1, 0 \le k_i \le n-1\} \cup \{\langle OP_{NG}\rangle\} \cup \{\langle OP_{NG}, 2^i\rangle \mid 0 \le i \le m\} \\
& \cup\{\langle VR_1\rangle\} \cup \{\langle VR_2\rangle\} \cup \{\langle Start_i\rangle \mid 0 \le i \le n-1\} \cup \{\langle DFS_{i,j}\rangle \mid 0 \le i \le n, 0 \le j \le n\} \\
& \cup\{\langle R_m\rangle \mid 0 \le m \le m\} \cup \{\langle OP_1\rangle\} \cup \{\langle OP_2\rangle\} \cup \{\langle W_i, k\rangle \mid 1 \le i \le m, k = 2^h, 0 \le h \le m\} \\
& \cup\{\langle W_{NG}, k\rangle \mid k = 2^h, 0 \le h \le m\} \cup \{\langle Output_i\rangle \mid 1 \le i \le m\} \cup \{\langle NoSolution\rangle\} \\
& \cup\{\langle OPE_{i,j}\rangle \mid 0 \le i < j \le n-1\} \cup \{\langle OPSol, i\rangle \mid 1 \le i \le m\} \\
& \cup\{\langle OPS_{i,j}\rangle \mid 0 \le i < j \le n-1\} \cup \{\langle OutE_i\rangle \mid 0 \le i \le m\}
\end{aligned}
$$

$$
\begin{aligned}
\mu \quad &= \quad [[]_2]_1 \\
\omega_1 \quad &= \quad \{\langle Start\rangle\} \\
\omega_2 \quad &= \quad \phi \\
R_1 \quad &= \quad R_{1,1} \cup R_{1,4} \\
R_2 \quad &= \quad R_{2,1} \cup R_{2,2} \cup R_{2,3-1} \cup R_{2,3-2} \cup R_{2,4} \\
i_{in} \quad &= \quad 1 \\
i_{out} \quad &= \quad 1
\end{aligned}
$$

(Evolution rules for outer membranes)

$$
\begin{aligned}
R_{1,4} \;=\; & \{\langle W_i,k\rangle\langle W_j,k\rangle \to \langle W_i,2k\rangle \mid 0\le i\le j\le m, k=2^h, 0\le h\le m-1\}\\
& \cup\{\langle W_i,k\rangle\langle W_{NG},k\rangle \to \langle W_i,2k\rangle \mid 0\le i\le m, k=2^h, 0\le h\le m-1\}\\
& \cup\{\langle W_{NG},k\rangle^2 \to \langle W_{NG},2k\rangle \mid k=2^h, 0\le h\le m-1\}\\
& \cup\{\langle W_i,2^m\rangle \to \langle Output_i\rangle\langle OPSol,i\rangle \mid 0\le i\le m\}\\
& \cup\{\langle Output_i\rangle[\,]_2 \to [\langle Output_i\rangle]_2 \mid 0\le i\le m\}\\
& \cup\{[\langle W_{NG},2^m\rangle]_1 \to [\,]_1\langle NoSolution\rangle\}\\
& \cup\{\langle OPSol,k\rangle\langle OPE_{i,j}\rangle \to \langle OPSol,k-1\rangle\langle OPS_{i,j}\rangle \mid 0\le i<j\le n-1, 0\le k\le m\}\\
& \cup\{\langle OPS_{i,j}\rangle \to \langle e_{i,j}\rangle \mid 0\le i<j\le n-1\}
\end{aligned}
$$

(Evolution rules for inner membranes)

$$
\begin{aligned}
R_{2,4} \;=\; & \{\langle OP_1\rangle\langle NE_{ne}\rangle \to \langle OP_2\rangle\langle NE_{ne}\rangle\langle W_{ne}\rangle \mid 1\le ne<m\}\\
& \cup\{[\langle OP_2\rangle\langle W_{ne}\rangle]_2 \to [\,]_2\langle W_{ne},1\rangle \mid 1\le ne<m\}\\
& \cup\{[\langle OP_{NG}\rangle\langle W_{ne}\rangle]_2 \to \langle W_{NG},1\rangle \mid 1\le ne<m\}\\
& \cup\{[\langle OP_{NG},i\rangle]_2 \to \langle W_{NG},1\rangle^i \mid i=2^h, 0\le h\le m-1\}\\
& \cup\{\langle Output_i\rangle\langle NE_i\rangle \to \langle OutE_i\rangle \mid 0\le i\le m\}\\
& \cup\{[\langle Output_i\rangle\langle NE_j\rangle]_2 \to \langle Output_i\rangle \mid 1\le i<j\le m\}\\
& \cup\{\langle OutE_k\rangle\langle te_{i,j}\rangle \to \langle OutE_{k-1}\rangle\langle OPE_{i,j}\rangle \mid 0\le i<j\le n-1, 0\le k\le m\}\\
& \cup\{[\langle OPE_{i,j}\rangle]_2 \to [\,]_2\langle OPE_{i,j}\rangle \mid 0\le i<j\le n-1\}
\end{aligned}
$$

Figure 6: Evolution rules for Step 4

3.4 Example of executions of the proposed P system

We now show an example of executions of Π_{MST}. The behaviors for the input graph in Fig. 1 are shown in Figs. 7 to 12.

Figure 7 shows an example of Step 1. First, a set of input objects O_{in} is given in membrane 1, and then O_{in} is moved into membrane 2 and the objects representing the degrees of each vertex, the number of edges, and the number of vertices are generated.

Figures 8 and 9 show an example of Step 2. In Fig. 8, the membrane is divided into two membranes; one membrane includes the object $\langle e_{0,1}, T\rangle$ and the other membrane does not include the object $\langle e_{0,1}, T\rangle$. Figure 9 shows a bounding operation in Step 2. In the membrane, the vertex v_0, which is a terminal, is isolated since the degree is 0. Therefore, the membrane does not contain a Steiner tree, and the membrane is dissolved in Step 4.

Figures 10 and 11 show an example of Step 3. Figure 10 shows an example of (3-1). In the membrane shown in the figure, the number of edges is equal to the number of vertices. Therefore, the membrane proceeds to the next step. Figure 11 shows an example of (3-2). In this step, depth-first search is performed. In this figure, all vertices are traversed, which means that the graph is acyclic and connected. (Note that some detailed substeps are omitted in the figure.) Then, the membrane proceeds to the next step.

Figure 12 shows an example of Step 4. Each membrane with the object $\langle OP_i\rangle$ outputs the cost of the membrane. All costs are compared and the minimum cost remains. After that, the membrane with the minimum cost is selected and the edges of the membrane are outputted to membrane 1. Some inner membranes are omitted due to space limitation.

3.5 Complexity of P system

We now consider the complexity of the proposed P system.

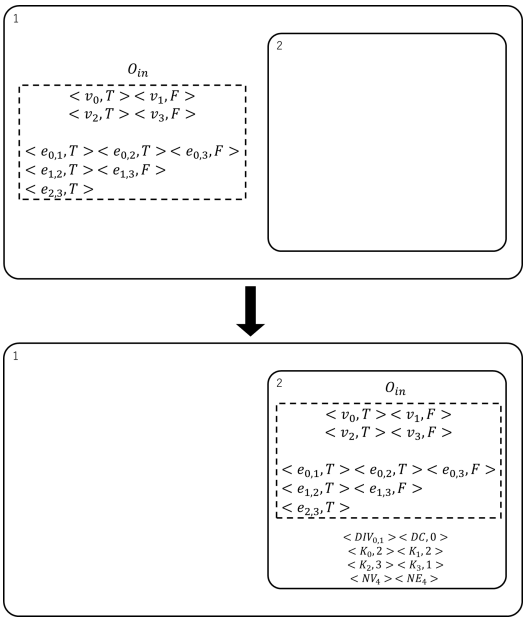

Figure 7: Example of Step 1

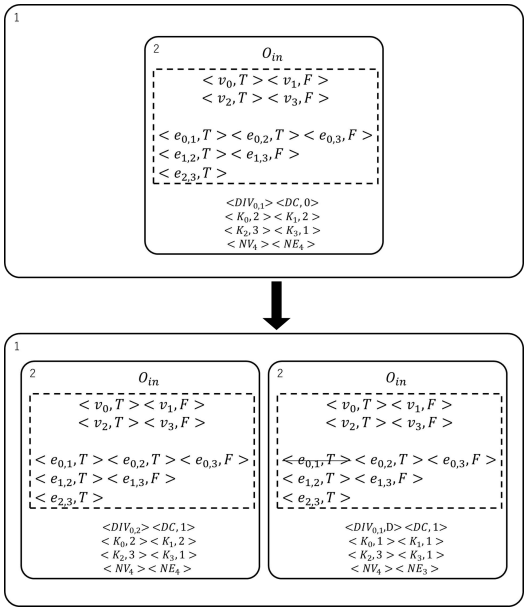

Figure 8: Example of Step 2 (a)

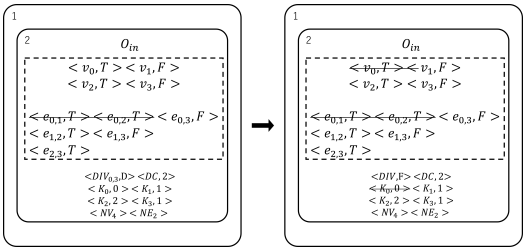

Figure 9: Example of Step 2 (b)

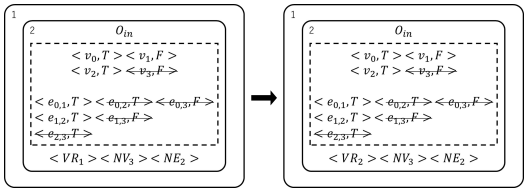

Figure 10: Example of Step 3: (3-1)

In Step 1, $O(m+n)$ objects are moved sequentially. Thus, Step 1 is executed in $O(n^2)$ sequential or parallel steps. The size of evolution rules is $O(n^4)$, and $O(n^2)$ kinds of objects are used.

In Step 2, the membrane is divided for each edge in the worst case. Thus, Step 2 is executed in $O(m)$ parallel steps or $O(2^m)$ sequential steps. The size of the evolution rules is $O(n^4m^2)$, and $O(n^4)$ kinds of objects are used.

In Step 3, a is executed in each membrane. Thus, Step 3 is executed in $O(n^2)$ parallel steps or $O(2^m n^2)$ sequential steps. The size of the evolution rules is $O(n^2m)$, and $O(n^2)$ kinds of objects are used.

In Step 4, weights of membranes are compared in the outer membrane. Thus, Step 4 is executed in $O(m)$ parallel steps or $O(2^m)$ sequential steps. The size of the evolution rules is $O(m^3)$, and $O(m^2)$ kinds of objects are used.

From the above, we obtain the following theorem regarding the complexity of the proposed asynchronous P system, Π_{MST}.

Theorem 1 *The asynchronous P systems,* Π_{MST}*, solve for the minimum Steiner tree with n vertices and m edges and operate in* $O(n^2)$ *parallel steps or* $O(2^m n^2)$ *sequential steps using* $O(n^4)$ *kinds of objects and evolution rules of* $O(n^4m^2)$*.* □

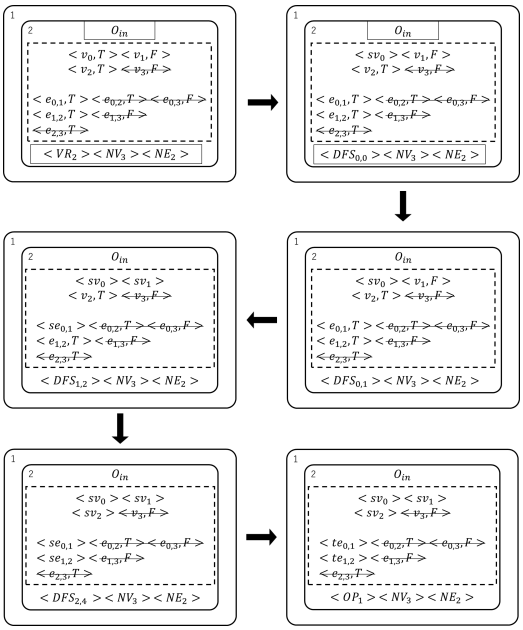

Figure 11: Example of Step 3: (3-2)

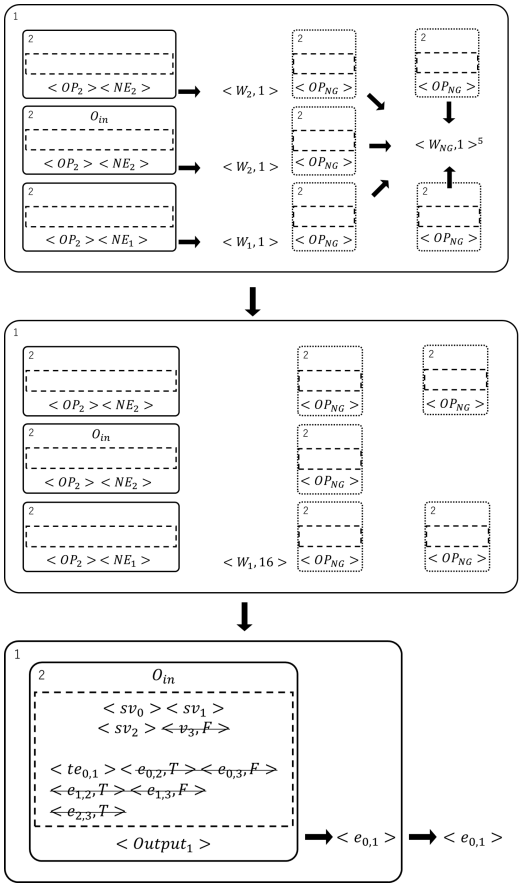

Figure 12: Example of Step 4

4 Experimental simulations

Although the asymptotic complexity of the proposed P system is the same as that of an exhaustive search, the number of membranes can be reduced by the branch and bound technique. We discuss the concrete reduction rate for the number of membranes in this section through experimental simulations.

For the simulations, we use our original simulator for the asynchronous P system. The simulator is built using Python 3. The input graph is randomly created for given numbers of n vertices and m edges.

We compare the number of membranes on the proposed P system and the number of membranes with an exhaustive search. In the simulations, we simulate a P system for graphs with from 50% to 70% density and 3 terminals.

Figure 13 shows the average number of membranes in 10 trials on the proposed P system and the number of membranes with an exhaustive search. As shown in the figure, the number of membranes on the proposed P system is smaller than the number of membranes with an exhaustive search, and the number of membranes used on the proposed P system is up to 57% less than the number of membranes with an exhaustive search.

In addition, we evaluate the number of membranes on the proposed P system for varying number of terminals or density of input graph. Figure 14 shows that the number of membranes on the proposed P system in the case that $n = 7$ and $m = 8$. The number of membranes becomes smaller according to the number of terminals. Figure 15 shows the number of membranes on the proposed P system in the case that $m = 8$ and the number of terminals is 3. The number of membranes becomes smaller with increasing density of input graph.

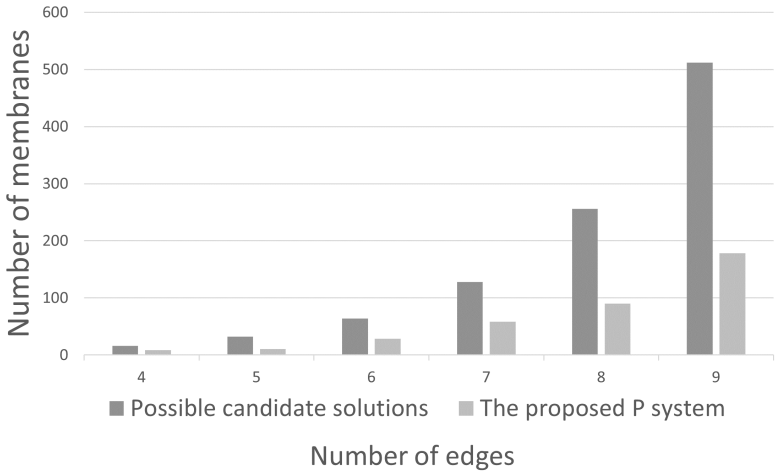

Figure 13: Number of membranes on the proposed P system and possible candidate solutions

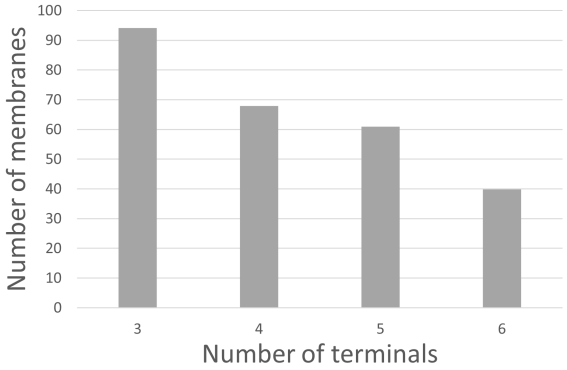

Figure 14: Number of membranes on the proposed P system with a varying number of terminals

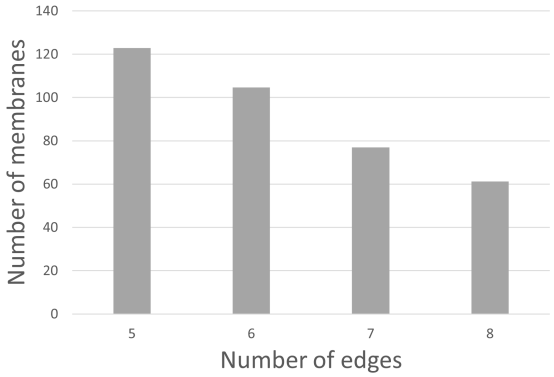

Figure 15: Number of membranes on the proposed P system a varying density of input graph

5 Conclusion

In the present paper, we proposed an asynchronous P system for solving the minimum Steiner tree problem using branch and bound. The results of simulations show that the proposed P system reduces the number of candidate solutions by 57%.

In our future research, we intend to consider reducing the number of membranes for the minimum Steiner tree problem using other optimization techniques.

Acknowledgment

This research was supported in part by JSPS KAKENHI through a Grant-in-Aid for Scientific Research (C) (No. 20K11681).

References

[1] R. Freund. Asynchronous P systems and P systems working in the sequential mode. *WMC'04 Proceedings of the 5th International Conference on Membrane Computing*, 3365:36–62, 2005.

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

[3] R. M. Karp. Reducibility among combinatorial problems. *Plenum Press, New York*, 1972.

[4] A. Leporati and C. Zandron. P systems with input in binary form. *International Journal of Foundations of Computer Science*, 17:127–146, 2006.

[5] T. Murakawa and A. Fujiwara. Asynchronous P system for arithmetic operations and factorization. *Proceedings of 3rd International Workshop on Parallel and Distributed Algorithms and Applications*, 2011.

[6] Y. Nakano and A. Fujiwara. An asynchronous P system with branch and bound for solving the knapsack problem. In *Workshop on Parallel and Distributed Algorithms and Applications*, pages 242–248, 2020.

[7] T. Noguchi and A. Fujiwara. An asynchronous P system with a dpll algorithm for solving sat. *International Journal of Networking and Computing*, 12(2):238–252, 2022.

[8] T. Noguchi and A. Fujiwara. An asynchronous P system with the Bron-Kerbosch algorithm for solving the maximum clique. *International Journal of Networking and Computing*, 13(2):131–148, 2023.

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

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

[11] G. Păun. P system with active membranes: Attacking NP-complete problems. *Journal of Automata, Languages and Combinatorics*, 6(1):75–95, 2001.

[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] T. Tateishi and A. Fujiwara. Logic and arithmetic operations with a constant number of steps in membrane computing. *International Journal of Foundations of Computer Science*, 22(3):547–564, 2011.

[15] K. Umetsu and A. Fujiwara. P systems with branch and bound for solving two hard graph problems. *International Journal of Networking and Computing*, 10(2):159–173, 2020.

[16] C. Zandron, G. Rozenberg, and G. Mauri. Solving NP-complete problems using P systems with active membranes. *Proceedings of the Second International Conference on Uncoventional Models of Computation*, pages 289–301, 2000.