International Journal of Networking and Computing – www.ijnc.org, ISSN 2185-2847
Volume 8, Number 2, pages 141–152, July 2018

An asynchronous P system with branch and bound for solving the satisfiability problem

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

Received: January 31, 2018
Accepted: March 19, 2018
Communicated by Fukuhito Ooshita

Abstract

Membrane computing, which is a computational model based on cell activity, has considerable attention as one of new paradigms of computations. In the general membrane computing, computationally hard problems have been solved in a polynomial number of steps using an exponential number of membranes. However, reduction of the number of membranes must be considered to make P system more realistic model.

In the paper, we propose an asynchronous P system with branch and bound, which is a well known optimization technique, to reduce the number of membranes. The proposed P system solves the satisfiability problem (SAT) with n variables and m clauses, and works in $O(m2^n)$ sequential steps or $O(mn)$ parallel steps.

In addition, the number of membranes used in the proposed P system is evaluated using a computational simulation. The experimental result shows validity and efficiency of the proposed P system.

Keywords: membrane computing, satisfiability problem (SAT), branch and bound

1 Introduction

A number of next-generation computing paradigms have been considered due to limitation of silicon-based computational hardware. As an example of the computing paradigms, natural computing, which works using natural materials for computation, has considerable attention. Membrane computing, which is a computational model inspired by the structures and behaviors of living cells, is a representative of the natural computing.

A basic feature of the membrane computing was introduced in [9] as a P system. The P system consists mainly of membranes and objects. A membrane is a computing cell, in which independent computation is executed, and may contain objects and other membranes. Each object evolves according to evolution rules associated with a membrane in which the object is contained.

The P system and most variants have been proved to be universal [11], and several P systems have been proposed for solving computationally hard problems [3, 6, 8, 10, 15, 18, 2, 12, 13, 14, 5]. For example, Pan and Alhazov [8] have proposed a P system that solves satisfiability problem (SAT). The P system solves SAT with n variables and m clauses in $O(n+m)$ steps using $O(mn^2)$ kinds of objects and $O(2^n)$ membranes.

In addition, asynchronous parallelism has been considered on the P system. The asynchronous parallelism means that all objects may react on rules with different speed, and evolution rules are

applied to objects independently. Since all objects in a living cell basically works in asynchronous manner, the asynchronous parallelism makes P system a more realistic computational model.

A number of asynchronous P systems have been proposed for the computationally hard problems in [1, 16, 7, 17, 4]. For example, an asynchronous P system has been proposed for solving SAT in [16]. The P system solves the satisfiability with n variables and m clauses in $O(mn2^n)$ sequential steps or $O(mn)$ parallel steps using $O(mn)$ kinds of objects.

In all of the above P systems, the computationally hard problems have been solved in polynomial numbers of steps using exponential numbers of membranes. The number of membranes means the number of living cells, and reduction of the number of membranes must be considered in case that the P system is implemented using living cells because living cells cannot be created exponentially. However, no P system is proposed for considering reduction of the number of membranes.

In the paper, we propose an asynchronous P system for solving SAT with branch and bound, which is a well known optimization technique, to reduce the number of membranes. In the proposed P system, one of two values, true or false, is assigned for variables one by one, and then, the satisfiability is checked for the partial assignment. If all clauses are satisfied or one of the clauses cannot be satisfied for the partial assignment, output of SAT can be determined. Since the number of membranes increases according to the number of assigned values, the number of membrane can be reduced by omitting value assignments for the other variables. We show that the proposed P system solves the satisfiability problem with n variables and m clauses, and works in $O(m2^n)$ sequential steps or $O(mn)$ parallel steps using $O(mn^2)$ kinds of objects.

Since the asymptotic complexities are almost the same as complexities of the previous P system [16], validity of the proposed system is evaluated using a computational simulation. In the simulation, various instances are executed on the previous P system and the proposed P system, and the number of membranes are compared for the same instances. The experimental results show validity and efficiency of the proposed P system with branch and bound.

The remainder of the present paper is organized as follows. In Section 2, we give a brief description of the model for asynchronous membrane computing. In Section 3, we propose the P system with branch and bound for SAT, and experimental results for the proposed P system are shown in Section 4. Section 5 concludes the paper.

2 Preliminaries

2.1 Computational model for membrane computing

Several models have been proposed for membrane computing. We briefly introduce a basic model of the P system in this subsection.

The P system consists mainly of membranes and objects. A membrane is a computing cell, in which independent computation is executed, and may contain objects and other membranes. In other words, the membranes form nested structures. In the present paper, each membrane is denoted by using a pair of square brackets, and the number on the right-hand side of each right-hand bracket denotes the label of the corresponding membrane. An object in the P system is a memory cell, in which each data is stored, and can divide, dissolve, and pass through membranes. In the present paper, each object is denoted by finite strings over a given alphabet, and is contained in one of the membranes.

For example, $[[a]_2[b]_3]_1$ and Figure 1 denotes the same membrane structure that consists of three membranes. The membrane labeled 1 contains two membranes labeled 2 and 3, and the two membranes contain objects a and b, respectively.

Computation of P systems is executed according to evolution rules, which are defined as rewriting rules for membranes and objects. All objects and membranes are transformed in parallel according to applicable evolution rules. If no evolution rule is applicable for objects, the system ceases computation.

We now formally define a P system and the sets used in the system as follows.

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

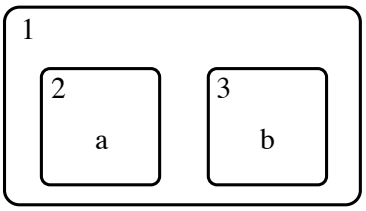

Figure 1: An example of membrane structure

O: O is the set of all objects used in the system.

μ: μ is membrane structure that consists of m membranes. Each membrane in the structure is labeled with an integer.

ω_i: Each ω_i is a set of objects initially contained only in the membrane labeled i.

R_i: Each R_i is a set of evolution rules that are applicable to objects in the membrane labeled i.

In the present paper, we assume that input objects are given from the outside region into the outermost membrane, and computation is started by applying evolution rules. We also assume that output objects are sent from the outermost membrane to the outside region.

In membrane computing, several types of rules are proposed. In the present paper, we consider five basic rules of the following forms.

(1) Object evolution rule:

$$[\ a\ ]_h \to [\ b\ ]_h$$

In the above rule, h is a label of the membrane and $a, b \in O$. Using the rule, an object a evolves into another object b. (We omit the brackets in each evolution rule such as $a \to b$ for cases that a corresponding membrane is obvious.)

(2) Send-in communication rule:

$$a[\ ]_h \to [\ b\ ]_h$$

In the above rule, h is a label of the membrane, and $a, b \in O$. Using the rule, an object a is sent into the membrane, and can evolve into another object b.

(3) Send-out communication rule:

$$[\ a\ ]_h \to [\ ]_h b$$

In the above rule, h is a label of the membrane, and $a, b \in O$. Using the rule, an object a is sent out of the membrane, and can evolve into another object b.

(4) Dissolution rule:

$$[\ a\ ]_h \to b$$

In the above rule, h is a label of the membrane, and $a, b \in O$. Using the rule, the membrane, which contains object a, is dissolved, and the object can evolve into another object b. (The outermost membrane cannot be dissolved.)

(5) Division rule:

$$[\ a\ ]_h \to [\ b\ ]_h[\ c\ ]_h$$

In the above rule, h is a label of the membrane, and $a, b, c \in O$. Using the rule, the membrane, which contains object a, is divided into two membranes that contain objects b and c, respectively.

We assume that each of the above rules is applied in a constant number of biological steps. In the following sections, we consider the number of steps executed in a P system as the complexity of the P system.

2.2 Maximal parallelism and asynchronous parallelism

In the standard model in membrane computing, which is a P system with maximal parallelism, all of the above rules are applied in a non-deterministic maximally parallel manner. In one step of computation of the P system, each object is evolved according to one of applicable rules. (In case there are several possibilities, one of the applicable rules is non-deterministically chosen.) All objects, for which no rules applicable, remain unchanged to the next step. In other words, all applicable rules are applied in parallel in each step of computation.

On the other hand, evolution rules are applied in a fully asynchronous manner on the asynchronous P system, and any number of applicable evolution rules is applied in each step of computation. In other words, the asynchronous P system can be executed sequentially, and also can be executed in the maximal parallel manner.

The reason why we assume the asynchronous parallelism in this paper is based on the fact that every living cell acts independently and asynchronously. Since the standard P system ignores the asynchronous feature of living cells, the asynchronous P system is a more realistic computation model for cell activities.

We now show an example for difference between the P system with maximal parallelism and the asynchronous P system. We define P system Π and the sets used in the system as follows.

$$\Pi = (O, \mu, \omega_1, R_1)$$
$$O = \{a, b, c, d, e\}, \mu = [\ ]_1, \omega_1 = \phi,$$
$$R_1 = \{a \to b, bc \to d, c \to e, ae \to f\}$$

We consider computations of the P system Π. Let us assume that input objects ac are given into the membrane from the outside region. On the standard P system, all applicable evolution rules, which are $a \to b$ and $c \to e$, are applied in parallel, and the objects ac are evolved into be. Since objects be cannot be evolved using evolution rules in R_1, the computation on the P system is halted.

On the other hand, five computations given below can be considered on the asynchronous P system according to the order of application of the evolution rules.

$$ac \to be$$
$$ac \to ae \to be$$
$$ac \to ae \to f$$
$$ac \to bc \to be$$
$$ac \to bc \to d$$

Therefore, a number of executions are possible in the asynchronous P system, and the evolution rules in the standard P system, which assumes the maximal parallel manner, may not work in the asynchronous parallel manner.

In the asynchronous P system, all evolution rules can be applied completely in parallel, which is the same as the conventional P system, or all evolution rules can be applied sequentially. We define the number of steps executed in the asynchronous P system in the maximal parallel manner as the number of parallel steps. We also define the number of steps in the case that the applicable evolution rules are applied sequentially as the number of sequential steps. The numbers of parallel and sequential steps indicate the best and worst case complexities for the asynchronous P system. In addition, the proposed asynchronous P system must be guaranteed to output a correct solution in any asynchronous execution.

3 An asynchronous P system with branch and bound for SAT

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

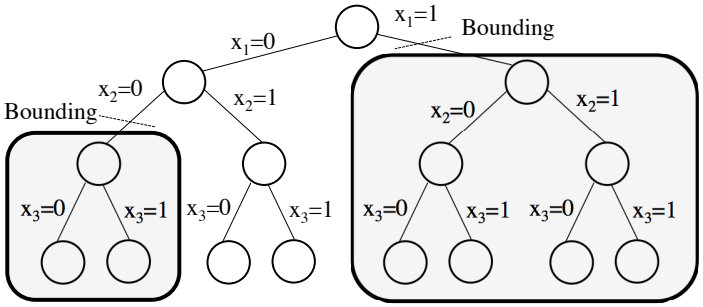

Figure 2: An example of branch and bound for SAT

3.1 Input and output for SAT

The satisfiability problem (SAT) is a well-known problem that determines if there exists a truth assignment for a given Boolean formula. We assume that an input formula of SAT is given in the conjunctive normal form (CNF) with n Boolean variables and m clauses. We also assume that an output of SAT is one of two values, "TRUE" and "FALSE". The output is "TRUE" if there exists a truth assignment for satisfying the formula, otherwise, the value is "FALSE".

The following is an example of an input formula with 3 variables and 3 clauses. Since a truth assignment, $x_1 = 0, x_2 = 1$ and $x_3 = 0$, satisfies the input formula, an output of SAT for the instance is "TRUE".

$$(x_1 \vee x_2) \wedge (\neg x_2 \vee \neg x_3) \wedge (\neg x_1)$$

The above input is given by the following set of objects O_L in the P system.

$$O_L = \{\langle X_{i,j}, V\rangle \mid 1 \leq i \leq n, 1 \leq j \leq m, V \in \{0, 1, N\}\}$$

Each object $\langle X_{i,j}, V\rangle$ denotes a Boolean value of variable x_i in the j-th clause. In addition, V is set to N if neither x_i nor $\bar{x}_i$ is in the j-th clause.

For example, the following set of objects denotes the above input formula.

$$\begin{aligned} O_L \;=\; \{\; & \langle X_{1,1}, 1\rangle, \langle X_{1,2}, N\rangle, \langle X_{1,3}, 0\rangle, \\ & \langle X_{2,1}, 1\rangle, \langle X_{2,2}, 0\rangle, \langle X_{2,3}, N\rangle, \\ & \langle X_{3,1}, N\rangle, \langle X_{3,2}, 0\rangle, \langle X_{3,3}, N\rangle \;\} \end{aligned}$$

We assume that input set O_L is given from the outside region into the outer membrane.

The output of the P system is one of two objects, $\langle TRUE\rangle$ and $\langle FALSE\rangle$. The object $\langle TRUE\rangle$ is sent out from the outer membrane to the outside region if the input formula is satisfiable, otherwise, the object $\langle FALSE\rangle$ is sent out to the outside region.

3.2 Branch and bound technique for SAT

Branch and bound is a well known computing paradigm for optimization problems. For solving SAT, all assignments are enumerated for n Boolean variables, and 2^n solutions must be created for the exhaustive enumeration. However, a number of assignments can be discarded if the output is determined to be "TRUE" or "FALSE" by partial assignments for the variables.

Figure 2 shows a search tree for illustration of the above idea. Let $(x_1 \vee x_2) \wedge (\neg x_2 \vee \neg x_3) \wedge (\neg x_1)$ be an input formula. In this case, the first clause cannot be satisfied in case of $x_1 = 0, x_2 = 0$, and assignments for x_3 can be ignored. In another case, the third clause cannot be satisfied in case of $x_1 = 1$, and assignments for x_2 and x_3 can be ignored.

We now explain an overview of the asynchronous P system with branch and bound for solving SAT. An initial membrane structure for the computation is $[\,[\,]_2\,]_1$. We call the membranes labeled 1 and 2 *outer* and *inner* membranes, respectively.

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

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

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

(2-1) Create a truth assignment for a variable by dividing the inner membrane.

(2-2) Check satisfiability for a truth assignment in each divided membrane. If all clauses are satisfied with the assignment, an object denoting "TRUE" is outputted to the outer membrane. On the other hand, if one of the clauses cannot be satisfied by the truth assignment, an object denoting "FALSE" is outputted to the outer membrane.

Step 3: Send out a final result, "TRUE" or "FALSE", from the outer membrane.

3.3 Details of the P system

We now explain details of each step of the computation. In Step 1, all input objects in the outer membrane are moved into the inner membrane. Since the P system used in the paper is asynchronous, we cannot move the input objects in parallel, and input objects are moved one by one applying following two sets of evolution rules.

(Evolution rules for the outer membrane)

$$\begin{aligned} R_{1,1} \;=\; & \{\langle X_{1,1},V\rangle[\,]_2 \;\to [\langle M_{2.1},0\rangle\langle X_{1,1},V\rangle]_2 \mid V\in\{0,1,N\}\} \\ & \cup\{\langle M_{i,j},k\rangle\langle X_{i,j},V\rangle[\,]_2 \;\to [\langle M_{i+1,j},0\rangle\langle X_{i,j},V\rangle]_2 \\ & \quad \mid 1\le i\le n, 1\le j\le m, 0\le k\le n-1, V\in\{0,1\}\} \\ & \cup\{\langle M_{i,j},k\rangle\langle X_{i,j},N\rangle[\,]_2 \;\to [\langle M_{i+1,j},k+1\rangle\langle X_{i,j},N\rangle]_2 \\ & \quad \mid 1\le i\le n, 1\le j\le m, 0\le k\le n\} \end{aligned}$$

(Evolution rules for the inner membrane)

$$\begin{aligned} R_{2,1} \;=\; & \{[\langle M_{i,j},k\rangle]_2 \;\to\; [\,]_2\langle M_{i,j},k\rangle \mid 2\le i\le n, 1\le j\le m\} \\ & \cup\{\langle M_{n+1,j},k\rangle \;\to\; \langle M_{1,j+1},0\rangle\langle F_j,n-k,0\rangle \mid 1\le j\le m, 0\le k\le n-1\} \\ & \cup\{\langle M_{1,m+1},0\rangle \;\to \langle S_1\rangle\langle m\rangle\} \end{aligned}$$

In the above evolution rules, object $\langle X_{1,1},V\rangle$ is moved from the outer membrane to the inner membrane, and then, object $\langle M_{2,1},0\rangle$ is created and moved to the inner membrane. (Object $\langle M_{i,j},k\rangle$ moves object $\langle X_{i,j},V\rangle$ from the outer membrane to the inner membrane.)

In addition, object $\langle F_j,k,f\rangle$ is created for the j-th clause. In the object, k denotes the order of division for the clause, and $f\in\{0,1\}$ is a flag that is set to 1 if j-th clause is satisfied. The object $\langle F_j,k,f\rangle$ is also moved to the inner membrane.

After all input objects are moved to the inner membrane, object $\langle M_{1,m+1},0\rangle$ is created, and then, object $\langle S_1\rangle$ and $\langle m\rangle$ are created in the inner membrane. Object $\langle m\rangle$ denotes the the number of unsatisfied clauses, and object $\langle S_1\rangle$ triggers Step 2.

In Step 2, object $\langle S_i\rangle$ is used as a trigger for membrane division, and object $\langle A_i,0\rangle$ and $\langle A_i,1\rangle$ are used for assignments such that $x_i=0$ and $x_i=1$, respectively. In addition, object $\langle C_{i,j},i\rangle$ is used for checking if an assignment for the i-th literal satisfies the j-th clause.

In (2-1), a truth assignment for a variable is created by dividing the inner membrane. The sub-step is executed by applying the following set of evolution rules.

(Evolution rules for the outer membrane)

$$R_{2,2,1} \;=\; \{[\langle S_i\rangle]_2 \;\to [\langle A_i,0\rangle\langle C_{i,1},i\rangle]_2[\langle A_i,1\rangle\langle C_{i,1},i\rangle]_2 \mid 1\le i\le n\}$$

In (2-2), satisfiability for a truth assignment in each divided membrane is checked. The sub-step is executed by applying the following sets of evolution rules.

(Evolution rules for the inner membrane)

$$
\begin{aligned}
R_{2,2,2} &= R_{2,2,2,1} \cup R_{2,2,2,2} \cup R_{2,2,2,3} \\
R_{2,2,2,1} &= \{\langle C_{i,j}, i\rangle\langle X_{i,j}, V\rangle\langle A_i, V'\rangle\langle F_j, k, f\rangle \to \langle C_{i,j+1}, i\rangle\langle X_{i,j}, V\rangle\langle A_i, V'\rangle\langle F_j, k, f\rangle \\
&\quad \mid 1 \le i \le n, 1 \le j \le m, i \ne k, f \in \{0,1\}, V \in \{0,1,N\}, V' \in \{0,1\}, V \ne V'\} \\
&\quad \cup \{[\langle C_{i,j}, i\rangle\langle X_{i,j}, V\rangle\langle A_i, V'\rangle\langle F_j, i, 0\rangle]_2 \to [\ ]_2\langle FALSE, 2^{n-i}\rangle \\
&\quad \mid 1 \le i \le n, 1 \le j \le m, V \in \{0,1,N\}, V' \in \{0,1\}, V \ne V'\} \\
&\quad \cup \{\langle C_{i,j}, i\rangle\langle X_{i,j}, V\rangle\langle A_i, V\rangle\langle F_j, k, 0\rangle\langle l\rangle \to \langle C_{i,j+1}, i\rangle\langle X_{i,j}, V\rangle\langle A_i, V\rangle\langle F_j, k, 1\rangle\langle l-1\rangle \\
&\quad \mid 1 \le i \le n, 1 \le j \le m, 1 \le l \le m, 1 \le k \le n, V \in \{0,1\}\} \\
&\quad \cup \{\langle C_{i,j}, i\rangle\langle X_{i,j}, V\rangle\langle A_i, V\rangle\langle F_j, k, 1\rangle \to \langle C_{i,j+1}, i\rangle\langle X_{i,j}, V\rangle\langle A_i, V\rangle\langle F_j, k, 1\rangle \\
&\quad \mid 1 \le i \le n, 1 \le j \le m, 0 \le k \le n-1, i \ne k, V \in \{0,1\}\} \\
R_{2,2,2,2} &= \{\langle C_{i,m+1}, i\rangle\langle l\rangle \to \langle S_{i+1}\rangle\langle l\rangle \mid 1 \le i \le n, 1 \le j \le m, 1 \le l \le m\} \\
&\quad \cup \{[\langle C_{i,m+1}, i\rangle\langle 0\rangle]_2 \to [\ ]_2\langle TRUE\rangle \mid 1 \le i \le n\} \\
R_{2,2,2,3} &= \{[\langle S_{n+1}\rangle]_2 \to [\ ]_2\langle FALSE, 1\rangle\}
\end{aligned}
$$

In the above evolution rules, satisfiability of each clause is checked in order of $X_{i,1}, X_{i,2}, \cdots X_{i,m}$, and the number of satisfied clauses is computed using evolution rules in $R_{2,2,2,1}$. If all clauses are satisfied, $\langle TRUE\rangle$ is outputted to the outer membrane, and membrane division is terminated. On the other hand, if a clause cannot be satisfied by a truth assignment in the divided membrane, object $\langle FALSE, 2^{n-i}\rangle$ is outputted to the outer membrane, and membrane division is also terminated.

Otherwise, if truth assignments are created for all variables, object $\langle S_{n+1}\rangle$, which indicates that all checks for variables are completed, is created, and object $\langle FALSE, 1\rangle$ is outputted to the outer membrane. If the check is not completed for all variables, object $\langle C_{i,m+1}, i\rangle$ is changed into object $\langle S_{i+1}\rangle$, and Step 2 is repeated by executing evolution rules in $R_{2,2,2,1}$.

In Step 3, a final result is sent out from the outer membrane. The Step 3 is executed applying the following set of evolution rules.

(Evolution rules for the outer membrane)

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

If object $\langle TRUE\rangle$ is in the outer membrane, there is an assignment that satisfies the input formula, and the object is sent out from the membrane immediately applying $[\langle TRUE\rangle]_1 \to [\ ]_1\langle TRUE\rangle$. On the other hand, 2^n objects, $\langle FALSE, 1\rangle$, are sent out from the divided inner membranes to the outer membrane in case that the formula cannot be satisfied in Step 2. Since we assume the P system is asynchronous, the sum of the objects must be counted asynchronously, and a final $\langle FALSE\rangle$ is outputted if 2^n FALSE objects are obtained.

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

$$\Pi_{\mathrm{BB-SAT}} = (O, \mu, \omega_1, \omega_2, R_1, R_2)$$

- $O = O_L \cup \{\langle TRUE\rangle, \langle FALSE\rangle\}$

$$O_L = \{\langle X_{i,j}, V\rangle \mid 1 \le i \le n, 1 \le j \le m, V \in \{0,1,N\}\}$$

- $\mu = [\ [\ ]_2\ ]_1$
- $\omega_1 = \omega_2 = \phi$
- $R_1 = R_{1,1} \cup R_{1,3}$, $R_2 = R_{2,1} \cup R_{2,2,1} \cup R_{2,2,2}$

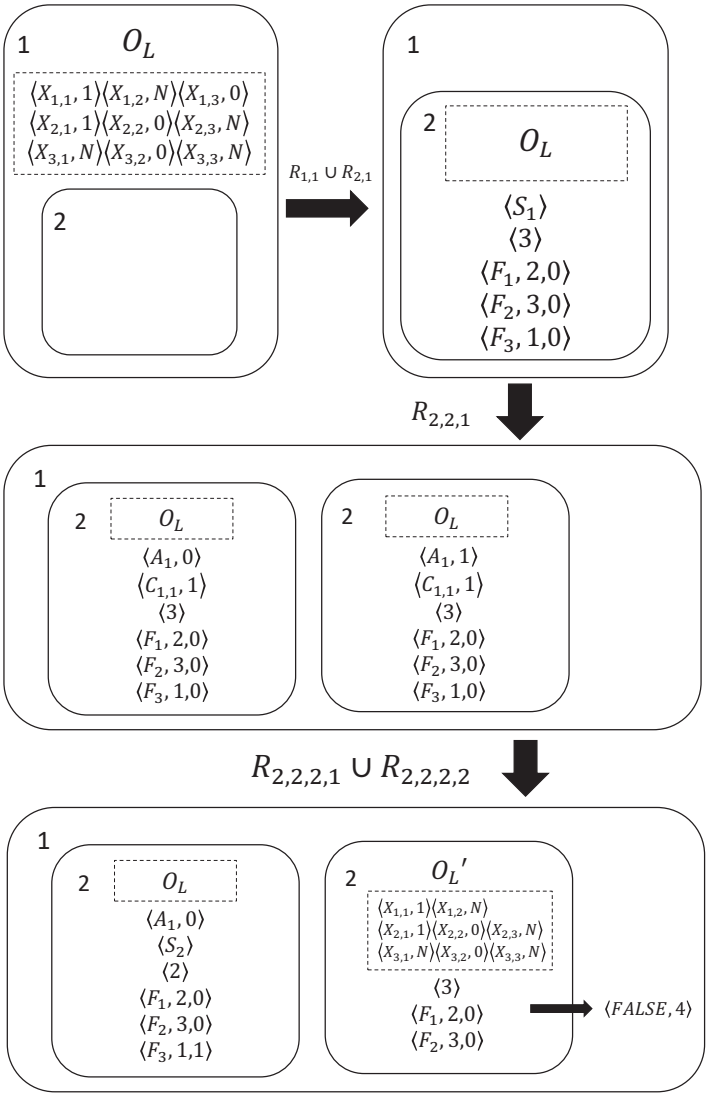

Figure 3: An example of execution of Π_{BB-SAT} (Step 1 and Step 2)

3.4 An example of P system

Figure 3 and Figure 4 illustrate an example execution of the proposed P system Π_{BB-SAT}. An input formula for the example is $(x_1 \vee x_2) \wedge (\neg x_2 \vee \neg x_3) \wedge (\neg x_1)$, and $n = 3$, $m = 3$.

Figure 3 illustrates an execution of Step 1 and Step 2. A set of objects O_L is given from the outside region into the outer membrane. By applying evolution rules $R_{1,1}$ and $R_{2,1}$, objects in O_L and other objects are moved to the inner membrane.

Next, by applying the evolution rule in $R_{2,2}$, $\langle A_1, 0\rangle$ and $\langle A_1, 1\rangle$, which indicate a truth assignment for the first variable x_1, are created. Then, satisfiability for a truth assignment, $x_1 = 0$ or $x_1 = 1$, is checked by applying the evolution rule in $R_{2,2,2}$. Since the last clause cannot be satisfied in case of $x_1 = 1$, object $\langle FALSE, 4\rangle$ is sent out to the outer membrane, and membrane division is terminated for the membrane.

Figure 4 illustrates an execution of Step 2 and Step 3. The evolution rules of Step 2 are applied to the inner membranes repeatedly until inner membrane outputs all solutions to the outer membrane. Then, the final solution object $\langle TRUE\rangle$ is outputted from the outer membrane to the outside region.

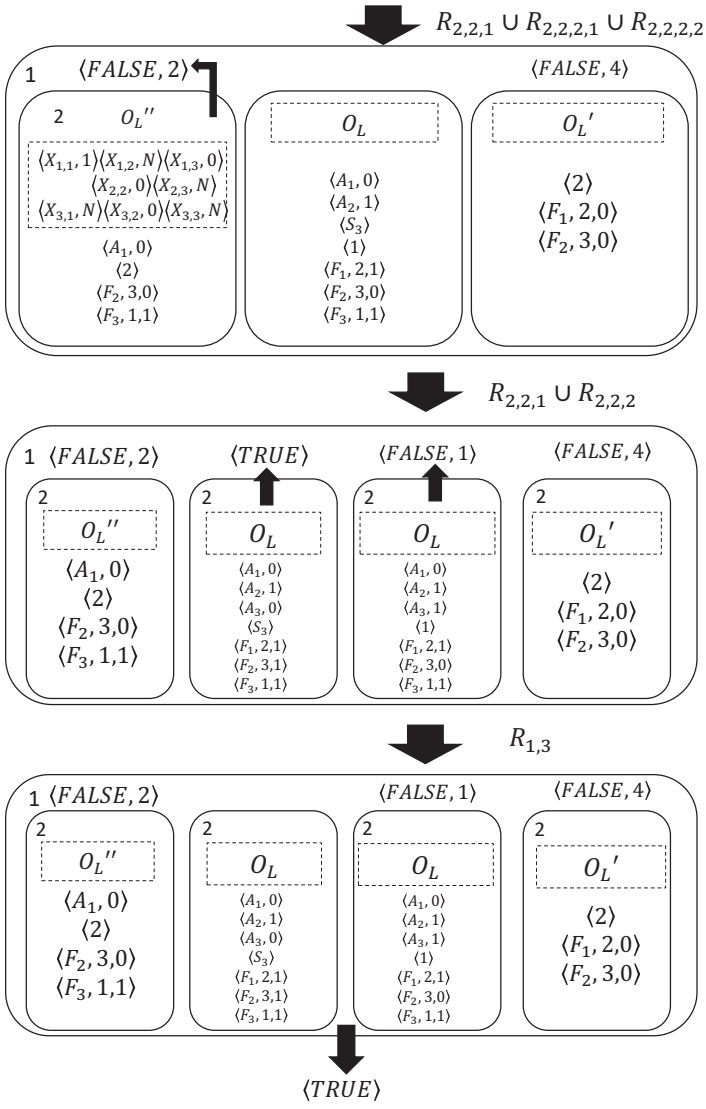

Figure 4: An example of execution of Π_{BB-SAT} (Step 2 and Step 3)

3.5 Complexity of the P system

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

In Step 2, $O(2^n)$ membranes are created in the worst case, and evolution rules are applied sequentially at most $O(m)$ times in each membrane. Therefore, Step 2 can be executed in $O(mn)$ parallel steps and $O(m2^n)$ sequential steps using $O(mn^2)$ kinds of objects and $O(m^2n^2)$ kinds of evolution rules.

In the final step, $O(n)$ evolution rules are applied sequentially in case that the output is "FALSE", and Step 3 can be executed in $O(n)$ parallel steps and $O(2^n)$ sequential steps using $O(n)$ kinds of objects and $O(n)$ kinds of evolution rules.

Therefore, we obtain the following theorem for the asynchronous P system Π_{BB-SAT}.

Theorem 1 *The asynchronous P system Π_{BB-SAT}, which computes the satisfiability problem,*

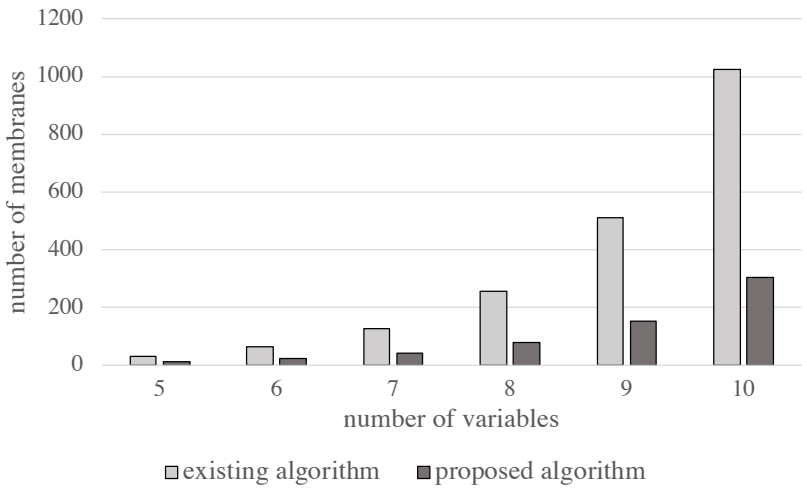

Figure 5: Experimental results

works in $O(m2^n)$ sequential steps or $O(mn)$ parallel steps by using $O(mn^2)$ types of objects and evolution rules of size $O(m^2n^2)$. □

4 Experimental simulation

We develop an original simulator for asynchronous P systems using C++ language, and compare the number of membranes used in executions of an existing P system for SAT [16] and our proposed P system.

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 system and the proposed P system for SAT on the simulator, and execute simulations for various inputs. In the simulation, valid results are obtained for all inputs, that is, both P systema output same objects for all inputs.

Next, we compare the number of membranes used on the existing P system and the proposed P system for SAT. Inputs of the simulation are CNFs with n variable and 3 clauses. For example, the following is an input of the simulation in case of $n = 3$.

$$\begin{aligned} O_L \quad = \quad & \{\ \langle X_{1,1}, V_{1,1}\rangle, \langle X_{1,2}, V_{1,2}\rangle, \langle X_{1,3}, V_{1,3}\rangle, \\ & \langle X_{2,1}, V_{2,1}\rangle, \langle X_{2,2}, V_{2,2}\rangle, \langle X_{2,3}, V_{2,3}\rangle, \\ & \langle X_{3,1}, V_{3,1}\rangle, \langle X_{3,2}, V_{3,2}\rangle, \langle X_{3,3}, V_{3,3}\rangle\ \} \end{aligned}$$

In the above input, each $V_{i,j}$ $(1 \leq i \leq n, 1 \leq j \leq 3)$ is set to 0, 1, or N with equal probability.

For the simulations, 100 inputs are randomly created for each n $(5 \leq n \leq 10)$, and the existing P system and the proposed P system are simulated for the given inputs.

Figure 5 shows that average values of the number of membranes on the simulation. The number of membranes of the existing P system increases exponentially to the number of variables n. Although the number of membranes on the proposed P system also increases according to n, the number of membranes on the proposed P system is 75 percent less than the number of membranes on the existing P system.

5 Conclusions

In this paper, we proposed the asynchronous P system with branch and bound for solving SAT. The P system with branch and bound reduces the number of membranes by omitting value assignments if all clauses are satisfied or one of clauses cannot be satisfied for the partial assignment.

The proposed P systems are fully asynchronous, i.e. any number of applicable rules may be applied in one step of the P systems. The proposed P systems works in a polynomial number of steps in the maximal parallel manner and also works sequentially. Although the number of sequential steps is exponential, the result means that the proposed P system works for any combinations of sequential and asynchronous applications of evolution rules, and guarantees that the P systems can output a correct solution in the case that any number of evolution rules are synchronized.

We experimented with the proposed P system and the existing P system. The experimental results show that the proposed P system outputs valid results, and also show that the number of the membrane on the proposed P system is at most 75 percent less than the number of membranes on the existing P system for SAT.

In our future research, we are considering reduction of the number of objects and evolution rules used in the proposed P system. We are also considering asynchronous P systems with branch and bound for other computationally hard problems.

Acknowledgments

This research was partially supported by JSPS KAKENHI, Grand-in-Aid for Scientific Research (C), 24500019. The authors would also like to thank the anonymous reviewers for giving us detailed comments that helped improve the readability 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 A. Riscos-Núñez. A fast P system for finding a balanced 2-partition. *Soft Computing*, 9(9):673–678, 2005.

[3] 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.

[4] 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.

[5] A. Leporati, C. Zandron, and G. Mauri. Solving the factorization problem with P systems. *Progress in Natural Science*, 17(4):471–478, 2007.

[6] V. Manca. DNA and membrane algorithms for SAT. *Fundamenta Informaticae*, 49(1-3):205–221, 2002.

[7] T. Murakawa and A. Fujiwara. Arithmetic operations and factorization using asynchronous P systems. *International Journal of Networking and Computing*, 2(2):217–233, 2012.

[8] 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.

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

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

[11] G. Păun. *Introduction to Membrane Computing*. Springer, 2006.

[12] M. J. Pérez-Jiménez and A. Riscos-Núñez. A linear-time solution to the knapsack problem using P systems with active membranes. *Membrane Computing*, 2933:250–268, 2004.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

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

[18] 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.