Hindawi
Mathematical Problems in Engineering
Volume 2019, Article ID 5793234, 8 pages
https://doi.org/10.1155/2019/5793234

Research Article

Solution to PSPACE-Complete Problem Using P Systems with Active Membranes with Time-Freeness

Bosheng Song[1] **and Yuan Kong**[2]

[1]*College of Information Science and Engineering, Hunan University, Changsha 410082, Hunan, China*
[2]*College of Mathematics and Systems Science, Shandong University of Science and Technology, Qingdao, 266590, China*

Correspondence should be addressed to Yuan Kong; kongyuan@sdust.edu.cn

Received 12 May 2019; Revised 13 June 2019; Accepted 18 June 2019; Published 27 June 2019

Academic Editor: Francesca Vipiana

P systems with active membranes are powerful parallel natural computing models, which were inspired by cell structure and behavior. Inspired by the parallel processing of biological information and with the idealistic assumption that each rule is completed in exactly one time unit, P systems with active membranes are able to solve computational hard problems in a feasible time. However, an important biological fact in living cells is that the execution time of a biochemical reaction cannot be accurately divided equally and completed in one time unit. In this work, we consider time as an important factor for the computation in P systems with active membranes and investigate the computational efficiency of such P systems. Specifically, we present a time-free semiuniform solution to the quantified Boolean satisfiability problem (`QSAT` problem, for short) in the framework of P systems with active membranes, where the solution to such problem is correct, which does not depend on the execution time for the used rules.

1. Introduction

Membrane computing is a member of natural computing that seeks to discover new computational models from the study of biological cells [1, 2]. Since the first computational model was proposed in 1998, many variants of models were constructed [3–7], and all computational models considered in such framework are called *P systems*. Inspired by the biological structure of cells, there are two kinds of P systems: *neural-like P systems* [8, 9] or *tissue-like P systems* [10, 11] (arbitrary graph structure) and *cell-like P systems* [2, 12] (tree structure). For more information about the area of membrane computing, we refer the reader to [13–18].

A *P system with active membranes* is one of basic cell-like P systems; all membranes are embedded in a skin membrane [19, 20]. In such P systems, each membrane delimits a *region* (also called *compartment*), which contains *multisets of objects* and *evolution rules*. If a membrane does not contain any other membrane, then it is called *elementary*; if a membrane contains at least an inner membrane, then it is called *nonelementary*. An interesting feature of P systems with active membranes is that there exists one of electrical charge (positive (+), negative (−), or neutral (0)) on each membrane, and such P systems are evolved according to the following types of rules: (a) *send-in* communication rules, (b) *send-out* communication rules, (c) evolution rules, (d) division rules, and (e) dissolving rules. When one of such rules is applied, the charge on the involved membrane may be changed.

There are many interesting results obtained by P systems with active membranes and their variants [19, 21, 22]. Because P systems with active membranes contain membrane division rules, which can generate arbitrary numbers of membranes, that is, an exponential workspace in polynomial time has been generated, they provide a way to theoretically solve *NP*-complete problems or *PSPACE*-complete problem in feasible time (in polynomial time or in linear time) by a time-space trade-off [23–27]. In [28], the `QSAT` problem is solved in linear time by P systems with restricted active membranes; however, in [29], the `QSAT` problem is solved in polynomial time by P systems with active membranes and without polarizations.

Inspired by biological fact that the execution time of a biochemical reaction cannot be accurately divided equally and completed in one time unit, in [30], *timed P systems* were proposed such that a natural number representing the execution time is associated with each rule. The notion of *time-free* is also proposed in [30], which is considered to

solve hard computational problems [31–37]. In all these time-free solutions to hard computational problems, the correct computation results are obtained when any execution time is given for each rule. Note that, in [32, 36], the QSAT problem was solved by P systems in a time-free uniform way.

In this work, the notion of time-freeness is incorporated into P systems with active membranes, and a time-free semi-uniform solution to the *PSPACE*-complete problem, QSAT problem, is presented, where the solution to such problem is correct, which does not depend on the execution time for the used rules.

2. Preliminary and Model Description

In this section, we first give the basic conception of formal language theory and the notion of P systems with active membranes [20, 38].

For an alphabet Γ (a finite nonempty set of symbols), we denote by Γ^* the set of all strings over Γ, and by $\Gamma^+ = \Gamma^* \backslash \{\lambda\}$ we denote the set of nonempty strings. The *length* of a string u, denoted by $|u|$, is the total number of symbols in the string.

We denote by (Γ, f) a *multiset* m over an alphabet Γ, where f is a mapping from Γ onto $\mathbb{N}$ (the set of natural numbers). If $\Gamma = \{a_1, \ldots, a_k\}$, then the multiset of m is denoted by $m = \{a_1^{f(a_1)}, \ldots, a_k^{f(a_k)}\}$, which can also be represented by $m = a_1^{f(a_1)} \ldots a_k^{f(a_k)}$ or by any permutation of this string.

Definition 1. A P system with active membranes is a tuple

$$\Pi = \left(O, H, \mu, w_1, \ldots, w_m, R\right), \tag{1}$$

where

(i) O is an alphabet which includes all objects used in the system;

(ii) H is a label set which marks for each membrane;

(iii) μ is an initial membrane structure;

(iv) w_i, $1 \leq i \leq m$, are multisets of objects that are placed in m membranes at the initial configuration;

(v) R is a finite set of rules with the following forms:

(a) Object evolution rules: $[a \longrightarrow v]_h^e$, for $h \in H, e \in C, a \in O, v \in O^*$.

(b) *Send-in* communication rules: $a[]_h^{e_1} \longrightarrow [b]_h^{e_2}$, for $h \in H, e_1, e_2 \in C, a, b \in O$.

(c) *Send-out* communication rules: $[a]_h^{e_1} \longrightarrow []_h^{e_2} b$, for $h \in H, e_1, e_2 \in C, a, b \in O$.

(d) Dissolving rules: $[a]_h^e \longrightarrow b$, for $h \in H, e \in C, a, b \in O$.

(e) Division rules for elementary membranes or nonelementary membranes: $[a]_h^{e_1} \longrightarrow [b]_h^{e_2}[c]_h^{e_3}$, for $h \in H, e_1, e_2, e_3 \in C, a, b, c \in O$.

The current membrane structure (including polarization associates with each membrane) and the multisets of objects in each membrane at a moment are considered a *configuration* of the P system. A P system starts from the initial configuration; using rules in a maximally parallel manner, a sequence of consecutive configurations is archived. If there is no rule used in the system, then *halting* configuration is reached. The result of a system is encoded by the objects present in the output membrane or emitted from the skin membrane when the system reaches halting configuration.

Next we give the definitions of timed P systems with active membranes and the corresponding recognizer version of such model; time-free solutions to decision problems by such P systems are also introduced [33, 39].

Definition 2. A timed P system with active membranes is a pair (Π, e), where Π is a P system with active membranes and e is a mapping from the finite set of rules in the system into the set of natural numbers $\mathbb{N}$ (representing the execution times for rules in such system).

A timed P system with active membranes $\Pi(e)$ works in the following way: an external clock is assumed, which marks time-units of equal length; the step t of computation is defined by the period of time that goes from instant $t-1$ to instant t. If a membrane contains a rule r from types $(a)-(e)$ selected to be executed, then execution of such rule takes $e(r)$ time units to complete. Therefore, if the execution is started at instant j, the rule is completed at instant $j + e(r)$ and the resulting objects and membranes become available only at the beginning of step $j + e(r) + 1$. When a rule r from types (b) – (e) is started, then the occurrences of symbol-objects and the membrane subject to this rule cannot be subject to other rules from types (b)–(e) until the implementation of the rule completes. At one step, a membrane can be subject to several rules of type (a).

In timed P systems with active membranes, the evolution rules and division rules also follow the "bottom-up" manner (one can refer to [33, 39] for more details).

Definition 3. A recognizer timed P system with active membranes of degree $m \geq 1$ is a tuple $\Pi = (O, H, \mu, w_1, \ldots, w_m, R, e, i_{out})$, such that

(i) the tuple $(O, H, \mu, w_1, \ldots, w_m, R)$ is a P system with active membranes, where alphabet O includes two elements, yes and no;

(ii) $i_{out} = 0$: the output zone is the environment;

(iii) e is a time-mapping of Π;

(iv) all the computations halt;

(v) either object yes or object no (but not both) must appear in the output region (environment) when the system reaches the halting configuration.

The present work also uses rule starting steps (RS-steps, for short) as the computation steps; a computation step is called a *RS-step* if at this step at least one rule starts its execution [33, 39].

A *decision problem* X is a pair (I_X, Θ_X) such that I_X is a language over a finite alphabet (whose elements are called *instances*) and Θ_X is a total Boolean function (i.e., predicate) over I_X.

Definition 4. Let $X = (I_X, \Theta_X)$ be a decision problem. We say that X is solvable in time-free polynomial RS-steps by a family of recognizer P systems with active membranes $\mathbf{\Pi} = \{\Pi_u : u \in I_X\}$ if the following items are true:

(i) the family $\mathbf{\Pi}$ is polynomially uniform by a Turing machine; that is, there exists a deterministic Turing machine working in polynomial time which constructs the system Π_u from the instance $u \in I_X$.

(ii) the family $\mathbf{\Pi}$ is time-free sound (with respect to X); that is, for any time-mapping e, the following property holds: if for each instance of the problem $u \in I_X$ such that there exists an accepting computation of $\Pi_u(e)$, we have $\Theta_X(u) = 1$.

(iii) the family $\mathbf{\Pi}$ is time-free complete (with respect to X); that is, for any time-mapping e, the following property holds: if for each instance of the problem $u \in I_X$ such that $\Theta_X(u) = 1$, every computation of $\Pi_u(e)$ is an accepting computation.

(iv) the family $\mathbf{\Pi}$ is time-free polynomially bounded; that is, there exists a polynomial function $p(n)$ such that, for any time-mapping e and for each $u \in I_X$, all the computations in $\Pi_u(e)$ halt in, at most, $p(|u|)$ RS-steps.

We also say that the family $\mathbf{\Pi}$ provides an efficient *time-free semiuniform solution* to the decision problem X.

3. A Time-Free Solution to QSAT Problem by P Systems with Active Membranes

Quantified Boolean formula problem (QBF, in short) is a well-known *PSPACE*-complete problem [40]; it asks whether a quantified sentential form over a set of Boolean variables is true or false. For more details about QBF, one can refer to [28].

Theorem 5. *A family of P systems with active membranes using rules of types (a), (b), (c), and (e′) can solve the* QSAT *problem in polynomial RS-steps in a time-free manner, where the solution to such problem is correct, which does not depend on the execution time for the used rules.*

Proof. Let us consider a quantified Boolean formula:

$$\begin{aligned} &\varphi = \exists x_1 \forall x_2 \ldots \exists x_{2n-1} \forall x_{2n} \left(C_1 \wedge C_2 \wedge \cdots \wedge C_m\right), \\ &C_i = y_{i,1} \vee \cdots \vee y_{i,l_i}, \quad 1 \le i \le m, \end{aligned} \tag{2}$$

where

$$y_{i,k} \in \left\{x_j, \neg x_j \mid 1 \le j \le 2n\right\}, \quad 1 \le i \le m,\ 1 \le k \le l_i. \tag{3}$$

We construct the following P system with active membranes:

$$\Pi = \left(O, H, \mu, w_1, w_2, \ldots, w_{2n+m+1}, R, i_{out}\right), \tag{4}$$

with the following components:

$$\begin{aligned} &O \\ &\quad = \left\{a_i 1, \le a_i^{(1)} i, \le a_i^{(2)} 2, n a_i^{(3)}, t_i, f_i\right\} \\ &\qquad \cup \left\{c_i 1, \le c_i' i, \le r_i m, r_i'\right\} \\ &\qquad \cup \left\{\mathtt{yes}, \mathtt{no}, a_{2n+1}, c_{m+1}, s, t\right\}. \\ &H = \{1, 2, \ldots, 2n + m + 1\}, \\ &\mu = \left[\left[\left[\cdots\left[\left[\left[\right]_{2n+1}^0 \cdots \left[\right]_{2n+m}^0\right]_{2n}^0 \right]_{2n-1}^0 \cdots\right]_2^0\right]_1^0\right]_{2n+m+1}^0, \\ &w_1 = a_1, \\ &w_i = \lambda, \quad 2 \le i \le 2n + m, \\ &w_{2n+m+1} = \mathtt{no}, \\ &i_{out} = 0, \end{aligned} \tag{5}$$

The rules designed for solving the QSAT problem are divided into four parts: generation phase, checking phase, quantifier phase, and output phase. In what follows, we give the set R and its explanation. Let e be any time-mapping from R to $\mathbb{N}$ (set of natural numbers representing the execution times for the rules).

Generation Phase

$G_{1,i} : [a_i]_i^0 \longrightarrow [t_i]_i^0 [f_i]_i^0$, $1 \le i \le 2n$.

$G_{2,i,j} : t_i[]_j^0 \longrightarrow [t_i]_j^0$, $1 \le i \le 2n-1,\ i+1 \le j \le 2n$.

$G_{3,i,j} : f_i[]_j^0 \longrightarrow [f_i]_j^0$, $1 \le i \le 2n-1,\ i+1 \le j \le 2n$.

$G_{4,i} : [t_i \longrightarrow r_{h_{i,1}} \ldots r_{h_{i,j_i}} a_i^{(1)}]_{2n}^0$, $1 \le i \le 2n$, and the clauses $C_{h_{i,1}}, \ldots, C_{h_{i,j_i}}$ contain the literal x_i.

$G_{5,i} : [f_i \longrightarrow r_{h_{i,1}} \ldots r_{h_{i,j_i}} a_i^{(2)}]_{2n}^0$, $1 \le i \le 2n$, and the clauses $C_{h_{i,1}}, \ldots, C_{h_{i,j_i}}$ contain the literal $\neg x_i$.

$G_{6,i,j} : [a_i^{(1)}]_j^0 \longrightarrow []_j^0 a_i^{(1)}$, $1 \le i \le 2n-1,\ i+1 \le j \le 2n$.

$G_{7,i,j} : [a_i^{(2)}]_j^0 \longrightarrow []_j^0 a_i^{(2)}$, $1 \le i \le 2n-1,\ i+1 \le j \le 2n$.

$G_{8,i} : [a_i^{(1)}]_i^0 \longrightarrow []_i^+ a_i^{(1)}$, $1 \le i \le 2n$.

$G_{9,i} : [a_i^{(2)}]_i^0 \longrightarrow []_i^- a_i^{(2)}$, $1 \le i \le 2n$.

$G_{10,i} : a_i^{(1)}[]_i^- \longrightarrow [a_i^{(3)}]_i^+$, $1 \le i \le 2n$.

$G_{11,i} : [a_i^{(3)}]_i^+ \longrightarrow []_i^+ a_i^{(3)}$, $1 \le i \le 2n$.

$G_{12} : [a_1^{(3)} \longrightarrow a_2 a_2]_{2n+m+1}^+$.

$G_{13,i} : [a_i^{(3)} \longrightarrow a_{i+1} a_{i+1}]_{i-1}^0$, $2 \le i \le 2n$.

$$G_{14,i} : a_i[]^+_{i-1} \longrightarrow [a_i]^0_{i-1}, 2 \le i \le 2n+1.$$

$$G_{15,i} : a_i[]^0_i \longrightarrow [a_i]^0_i, 2 \le i \le 2n.$$

Initially, we have object a_1 in membrane 1, which corresponds to variable x_1, and object no in membrane $2n+m+1$. At step 1, rules $G_{1,1}$ and O_1 start to be used at the same step, but they may complete at different steps. By using $G_{1,1}$, membrane 1 is divided, objects t_1 (representing the true value *true*) and f_1 (representing the true value *false*) are generated, and each of the new produced membranes will obtain one of these two objects. Besides, object no exits the membrane $2n+m+1$ by using rule O_1; polarization of membrane $2n+m+1$ is changed to positive. During the execution of rule $G_{1,1}$, the system takes one RS-step.

Rules $G_{2,1,2}$ and $G_{3,1,2}$ start to be used at the same step only when rule $G_{1,1}$ completes; however, due to the fact that execution for rules $G_{2,1,2}$ and $G_{3,1,2}$ may take different times, these two rules may complete at different steps. By using rule $G_{2,1,2}$ (resp., $G_{3,1,2}$), object t_1 (resp., f_1) enters membrane 2. After the execution of rule $G_{2,1,2}$ (resp., $G_{3,1,2}$), system starts to use rule $G_{2,1,3}$ (resp., $G_{3,1,3}$); object t_1 (resp., f_1) enters membrane 3. In this way, rules $G_{2,1,j}$ (resp., $G_{3,1,j}$) ($2 \le j \le 2n$) will be applied one by one, and object t_1 (resp., f_1) will be transferred to membrane $2n$.

If membrane $2n$ has object t_1 (resp., f_1), rule $G_{4,1}$ (resp., $G_{5,1}$) starts to apply; these rules are used to look for the clauses satisfied by the truth-assignment *true* (resp., *false*) of variable x_1. Note that rules $G_{4,1}$ and $G_{5,1}$ may start to be used at different steps.

If membrane $2n$ contains object $a_1^{(1)}$ (resp., $a_1^{(2)}$), then rule $G_{6,1,2n}$ (resp., $G_{7,1,2n}$) is used; object $a_1^{(1)}$ (resp., $a_1^{(2)}$) exits membrane $2n$. When membrane $2n-1$ contains object $a_1^{(1)}$ (resp., $a_1^{(2)}$), rule $G_{6,1,2n-1}$ (resp., $G_{7,1,2n-1}$) starts to apply; object $a_1^{(1)}$ (resp., $a_1^{(2)}$) exits membrane $2n-1$. In this way, rules $G_{6,1,j}$ (resp., $G_{7,1,j}$) ($2 \le j \le 2n$) will be applied one by one, and object $a_1^{(1)}$ (resp., $a_1^{(2)}$) will be transferred to membrane 1. Note that rules $G_{6,1,2n}$ and $G_{7,1,2n}$ may start at different steps; rules $G_{6,1,j}$ and $G_{7,1,j}$ ($2 \le j \le 2n-1$) may start at different steps.

When membrane 1 contains object $a_1^{(1)}$ (resp., $a_1^{(2)}$), system starts to use rule $G_{8,1}$ (resp., $G_{9,1}$), object $a_1^{(1)}$ (resp., $a_1^{(2)}$) exits membrane 1, and the polarization of membrane i is changed from neutral to positive (resp., negative). Note that rules $G_{8,1}$ and $G_{9,1}$ may start to be used at different steps. Rule $G_{10,1}$ can be applied only when membrane $2n+m+1$ has object $a_1^{(1)}$, and there is a membrane 1 having polarization negative; rule $G_{10,1}$ can be applied only when both rules $G_{8,1}$ and $G_{9,1}$ are completed. By applying rule $G_{10,1}$, object $a_1^{(1)}$ evolves to $a_1^{(3)}$, and object $a_1^{(3)}$ enters membrane 1; polarization of membrane 1 is changed to positive. Hence, rule $G_{10,1}$ has a synchronization function because $\Sigma_{2\le j\le 2n} e(G_{2,1,j}) + e(G_{4,1}) + \Sigma_{2\le j\le 2n} e(G_{6,1,j}) + e(G_{8,1})$ may not be equal to $\Sigma_{2\le j\le 2n} e(G_{3,1,j}) + e(G_{5,1}) + \Sigma_{2\le j\le 2n} e(G_{7,1,j}) + e(G_{9,1})$. Hence, after the execution of rule $G_{10,1}$, the system takes at most $8n+1$ RS-steps.

After the execution of rule $G_{10,1}$, the system starts to use rule $G_{11,1}$; object $a_1^{(3)}$ exits membrane 1. Now we have the following two cases:

(i) The execution of rule O_1 has completed; in this case, rule G_{12} starts to be used, the polarization of membrane $2n+m+1$ changes to positive, and two copies of object a_2 are produced. Each membrane 1 will obtain one copy of object a_2, and polarization of the corresponding membrane is changed to neutral.

(ii) The system is still at the execution of rule O_1; rules G_{12} and $G_{14,2}$ will be applied after the execution of rule O_1. After the execution of rule O_1, the system starts to use rules G_{12} and $G_{14,2}$.

After the execution of rule $G_{14,2}$, rule $G_{15,2}$ starts to apply; object a_2 enters membrane 2. In all membranes 1, the system starts to use rule $G_{15,2}$ at the same step, so rule $G_{15,2}$ in all membranes 1 completes at the same step; that is, in each membrane 2, object a_2 is generated at the same step. In general, when membrane 2 has object a_2, the system takes at most $8n+5$ RS-steps.

When membrane 2 has object a_2, the system starts to assign truth-assignment of variable x_2 and looks for clauses satisfied by such variable.

If membrane 2 has object a_2, rule $G_{1,2}$ starts to be used; the system starts to assign truth values *true* and *false* to variable x_2. In all membranes 1, the system starts and completes rule $G_{1,2}$ at the same step. The system starts to use rules $G_{2,2,3}$ and $G_{3,2,3}$ at the same step, but these rules may complete at different steps. Besides, the system starts and completes rules $G_{2,2,j}$ ($4 \le j \le 2n$), $G_{3,2,j}$ ($4 \le j \le 2n$), $G_{4,2}$, $G_{5,2}$, $G_{6,2,j}$ ($3 \le j \le 2n$), $G_{7,2,j}$ ($3 \le j \le 2n$), $G_{8,2}$, and $G_{9,2}$ at different steps. Note that rule $G_{10,2}$ can be used only when all the above rules have completed their executions. Thus, this process takes at most $8n+1$ RS-steps. So we can deduce that if we consider this process in the worse case, then the system assigns from x_i to x_{i+1} ($1 \le i \le 2n-2$); the number of RS-steps decreases by 4. Hence, the system assigns truth-assignment of variable x_{2n-1} and looks for the clauses satisfied by this variable; this process takes at most 13 RS-steps. Therefore, the system takes at most $8n^2 + 14n - 9$ RS-steps for this process.

When membrane $2n$ contains object a_{2n}, the system starts to execute rule $G_{1,2n}$. By using rule $G_{1,2n}$, objects t_1 and f_1 are generated, which will be placed in two separate copies of membrane $2n$. Note that rule $G_{1,2n}$ in all membranes $2n-1$ starts and completes at the same step. Rules $G_{4,2n}$ and $G_{5,2n}$ start to be used at the same step, but they may complete at different steps. Rules $G_{8,2n}$ and $G_{9,2n}$ may start and complete at different steps. When both rules $G_{8,2n}$ and $G_{9,2n}$ have been completed, rule $G_{10,2n}$ starts to apply. So, rule $G_{10,2n}$ has a synchronization function as $e(G_{4,2n}) + e(G_{8,2n})$ may not be equal to $e(G_{5,2n}) + e(G_{9,2n})$. If rule $G_{10,2n}$ completes, rules $G_{11,2n}$, $G_{13,2n}$, and $G_{14,2n+1}$ start to apply one by one. Hence, the system assigns truth-assignment of variable x_{2n} and looks for the clauses satisfied for such variable, this process takes at most 8 RS-steps.

So this phase takes at most $8n^2 + 14n - 1$ RS-steps.

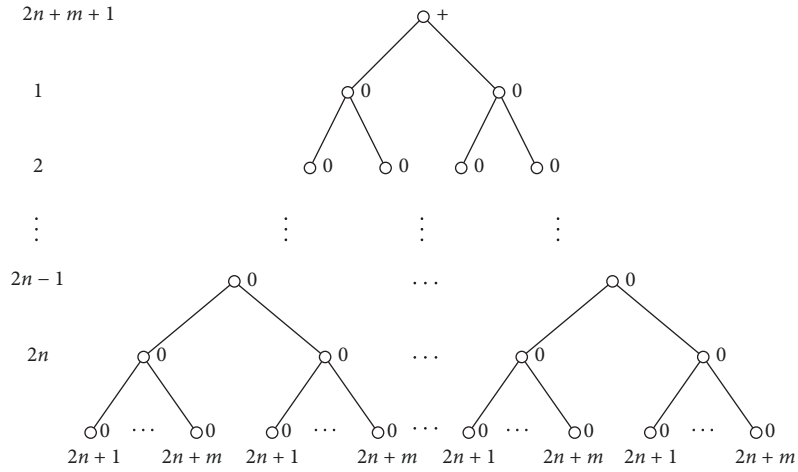

Figure 1: The membrane structure of the system Π at the moment when the generation phase completes.

The membrane structure is described in Figure 1 when generation phase finishes.

Checking Phase

$$C_1 : [a_{2n+1} \longrightarrow c_1]^0_{2n}.$$
$$C_{2,j} : c_j[]^0_{2n+j} \longrightarrow [c'_j]^+_{2n+j}, 1 \le j \le m.$$
$$C_{3,j} : r_j[]^+_{2n+j} \longrightarrow [r'_j]^0_{2n+j}, 1 \le j \le m.$$
$$C_{4,j} : [c'_j]^0_{2n+j} \longrightarrow []^0_{2n+j}c_{j+1}, 1 \le j \le m.$$
$$C_5 : [c_{m+1} \longrightarrow t]^0_{2n}.$$

After the execution of rule $G_{13,2n}$, all objects a_{2n+1} in each membrane $2n-1$ enter membrane $2n$; polarization for this membrane is changed to neutral at the same step. At that moment, rule C_1 is applied; object a_{2n+1} evolves to object c_1 in membrane $2n$. After the execution of rule C_1, the system starts to check whether each membrane $2n$ has object r_1 or not.

By applying rule $C_{2,1}$, object c_1 evolves to c'_1 and object c'_1 enters membranes $2n+1$; polarization of such membrane is changed from neutral to positive. After the execution of rule $C_{2,1}$, rule $C_{3,1}$ starts to be used, object r_1 evolves to r'_1, and object r'_1 enters membrane $2n+1$; polarization of such membrane is changed from positive to neutral. When the charge of membrane $2n+1$ changes to neutral, rule $C_{4,1}$ is used, object c'_1 evolves to c_2, and object c_2 exits membrane $2n+1$. When membrane $2n$ has object c_2, it means clause C_1 is satisfied, and the system starts to check whether membrane $2n$ has object r_2 or not.

If membrane $2n$ has object c_2, rule $C_{2,2}$ starts to be used. When membrane $2n$ has object r_2, object c_3 will be obtained. If a membrane $2n$ does not contain an object r_j, then this membrane will stop evolving when $C_{3,j}$ is supposed to be used. In general, the system takes at most $3m+1$ RS-steps.

If there exists a membrane $2n$ which contains all objects $r_1, r_2, \ldots, r_m$, then rule C_5 starts to apply; object c_{m+1} evolves to t in membrane $2n$.

This phase takes at most $3m+2$ RS-steps.

Quantifier Phase

$$Q_1 : [t]^0_{2n} \longrightarrow []^0_{2n}t.$$
$$Q_2 : [t]^0_i \longrightarrow []^+_i t, i = 2k,\ 1 \le k \le n-1.$$
$$Q_3 : [t]^0_i \longrightarrow []^+_i s, i = 2k-1,\ 1 \le k \le n.$$
$$Q_4 : [t]^+_i \longrightarrow []^0_i t, i = 2k-1,\ 1 \le k \le n.$$

After the execution of rule C_5 (if membranes $2n$ have object c_{m+1}), object t presents in membranes $2n$ at the same step. At that moment, rule Q_1 starts to be used; object t (if it exists) exits membranes $2n$. Note that in membranes $2n$ rule Q_1 starts and completes at the same step.

The quantifier $\exists$ is simulated by transferring only one copy of object t to an upper level membrane; that is, in membrane $2k$ $(1 \le k \le n-1)$, there exists at least one copy of object t. Specifically, if membranes $2k$ $(1 \le k \le n-1)$ have object t, rule Q_2 starts to be used, object t exits membranes $2k$, and the polarization of such membrane is changed from neutral to positive. Note that, in each of the immediately lower level membranes, if there exist two copies of object t in a membrane $2k$ $(1 \le k \le n-1)$, then in membranes $2k$ two copies of object t appear at the same step. Obviously, there are $n-1$ levels of membranes $2k$ $(1 \le k \le n-1)$, and rule Q_2 starts to be used at different steps for each level of membranes $2k$. Hence, the simulation of all quantifiers $\exists$ takes $n-1$ RS-steps.

The quantifier $\forall$ is simulated by transferring only one copy of t to an upper level membrane; that is, there are two copies of object t in membrane $2k-1$ $(1 \le k \le n)$. Specifically, if membranes $2k-1$ $(1 \le k \le n)$ contain two copies of t, rule Q_3 starts to apply, object t evolves to s, and object s exits membrane; the charge of such membrane is changed from neutral to positive. After the execution of rule Q_3, rule Q_4 starts to be used, object t exits this membrane, and the charge is changed from positive to neutral. Note that if each of the immediately lower level membranes of a membrane $2k-1$ $(1 \le k \le n)$ has one copy of object t, then two copies of t appear in membranes $2k-1$ at the same step. There are n levels

of membranes $2k-1$ ($1 \le k \le n$), and rules Q_3, Q_4 start to be used at different steps for each level of the membranes $2k-1$; thus, the simulation of all quantifiers $\forall$ takes $2n$ RS-steps.

In general, this phase takes $3n$ RS-steps.

Output Phase

$$O_1 : [\mathtt{no}]^0_{2n+m+1} \longrightarrow []^+_{2n+m+1}\mathtt{no}.$$
$$O_2 : \mathtt{no}[]^-_{2n+m+1} \longrightarrow [\mathtt{no}]^-_{2n+m+1}.$$
$$O_3 : [t]^+_{2n+m+1} \longrightarrow []^-_{2n+m+1}\mathtt{yes}.$$

At step 1, object no is sent out of the system by using rule O_1. Note that this operation takes no RS-step. After the quantifier phase, we have the following two cases.

If positive membrane $2n+m+1$ does not contain object t, in this case, rules O_3, O_2 cannot be applied. Hence when computation halts, the environment has object no, which means the formula is not satisfiable.

If positive membrane $2n+m+1$ contains object t, in this case, rule O_3 will be applied, object t evolves to yes, which will be sent to the environment, and the polarization of membrane $2n+m+1$ is changed from positive to negative. After the execution of rule O_3, rule O_2 starts to be used; object no enters membrane $2n+m+1$. Hence when computation halts, the environment has one copy of yes, which means the formula is satisfiable. The output phase takes two RS-steps.

According to the constructed P systems, for any time-mapping $e : R \longrightarrow \mathbb{N}$, if the computation halts, object yes (resp., no) appears in the environment if and only if the formula φ is satisfiable (resp., not satisfiable). Thus, the system Π is time-free sound and time-free complete.

For any time-mapping $e : R \longrightarrow \mathbb{N}$, the computation takes at most $8n^2 + 17n + 3m + 3$ RS-steps when formula φ is satisfiable, and the computation takes at most $8n^2 + 17n + 3m + 1$ RS-steps when formula φ is not satisfiable. Therefore, the family of P systems with active membrane is time-free polynomially bounded.

The family $\mathbf{\Pi} = \{\Pi_\varphi \mid \varphi$ is an instance of the QSAT problem$\}$ is polynomially uniform:

(i) total number of objects: $12n + 4m + 6$;

(ii) number of initial membranes: $2n + m + 1$;

(iii) cardinality of the initial multisets: $2n + m + 1$;

(iv) total number of rules: $8n^2 + 19n + 3m + 4$;

(v) maximal length of a rule: $m + 4$.

Therefore, P systems with active membranes can solve the QSAT problem in polynomial RS-steps in a time-free manner; this concludes the proof. □

4. Conclusions and Future Work

In this work, a time-free way of using rules is considered into P systems with active membranes, and a time-free solution to the QSAT problem by using P systems with active membranes has been given, where the solution to such problem is correct, which does not depend on the execution time for the used rules.

P systems with active membranes presented in this work are semiuniform; that is, the P systems are designed from the instances of the problem. It remains open how we can construct a family of P systems to solve the QSAT problem in a time-free manner in the sense that P systems are designed from the size of instances.

The P system constructed in Section 3 has the polarization on membranes. It is of interest to investigate whether P systems with active membranes without polarization on membranes can still solve the QSAT problem in a time-free context.

QSAT problem is one of the most important issues in many application areas, such as artificial intelligence aspect (e.g., planning, nonmonotonic reasoning, scheduling, model checking, and verification formal verification can be reduced to QSAT [41–45]) and fault localization in digital circuits aspect [46–51]. It is interesting to design new algorithms based on QSAT which can be used in the above-mentioned areas and other areas.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (61602192 and 61602188), the Fundamental Research Funds for the Central Universities (531118010355), and Scientific Research Foundation of Shandong University of Science and Technology for Recruited Talents (2017RCJJ068 and 2017RCJJ069).

References

[1] G. Păun, "Computing with membranes," *Journal of Computer and System Sciences*, vol. 61, no. 1, pp. 108–143, 2000.

[2] G. Păun, G. Rozenberg, and A. Salomaa, "DNA computing," in *New Computing Paradigms*, Springer, Berlin, Germany, 1998.

[3] P. Bottoni, A. Labella, and G. Rozenberg, "Reaction systems with infuence on?environment," *Journal of Membrane Computing*, vol. 1, pp. 3–19, 2019.

[4] M. À. Colomer, A. Margalida, D. Sanuy, and M. J. Pérez-Jiménez, "A bio-inspired computing model as a new tool for modeling ecosystems: The avian scavengers as a case study," *Ecological Modelling*, vol. 222, no. 1, pp. 33–47, 2011.

[5] G. Román, "Inference of bounded L systems with polymorphic P systems," *Journal of Membrane Computing*, vol. 1, pp. 52–57, 2019.

[6] M. García-Quismondo, M. Levin, and D. Lobo, "Modeling regenerative processes with membrane computing," *Information Sciences*, vol. 381, pp. 229–249, 2017.

[7] T. Wu, A. Paun, Z. Zhang, and L. Pan, "Spiking neural P systems with polarizations," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 8, pp. 3349–3360, 2018.

[8] M. Ionescu, G. Păun, and T. Yokomori, "Spiking neural P systems," *Fundamenta Informaticae*, vol. 71, no. 2-3, pp. 279–308, 2006.

[9] T. Wu, F.-D. Bîlbîe, A. Păun, L. Pan, and F. Neri, "Simplified and yet turing universal spiking neural P systems with communication on request," *International Journal of Neural Systems*, vol. 28, no. 8, Article ID 1850013, 2018.

[10] C. Martín-Vide, G. Păun, J. Pazos, and A. Rodríguez-Patón, "Tissue P systems," *Theoretical Computer Science*, vol. 296, no. 2, pp. 295–326, 2003.

[11] L. Pan, B. Song, L. Valencia-Cabrera, and M. J. Pérez-Jiménez, "The computational complexity of tissue P systems with evolutional symport/antiport rules," *Complexity*, vol. 2018, Article ID 3745210, 21 pages, 2018.

[12] D. Orellana-Martín, L. Valencia-Cabrera, A. Riscos-Núñez, and M. J. Pérez-Jiménez, "P systems with proteins: a new frontier when membrane division disappears," *Journal of Membrane Computing*, vol. 1, no. 1, pp. 29–39, 2019.

[13] G. Ciobanu, M. J. Pérez-Jiménez, and G. Paun, Eds., *Applications of Membrane Computing*, Natural Computing Series, Springer, Berlin, Germany, 2006.

[14] D. Díaz-Pernil, M. A. Gutiérrez-Naranjo, and H. Peng, "Membrane computing and image processing: a short survey," *Journal of Membrane Computing*, vol. 1, no. 1, pp. 58–73, 2019.

[15] P. Frisco, M. Gheorghe, and M. J. Pérez-Jiménez, *Applications of Membrane Computing in Systems and Synthetic Biology*, Springer, Switzerland, 2013.

[16] R. Mayne, N. Phillips, and A. Adamatzky, "Towards experimental P-systems using multivesicular liposomes," *Journal of Membrane Computing*, vol. 1, no. 1, pp. 20–28, 2019.

[17] Gh. Paun, G. Rozenberg, and A. Salomaa, Eds., *The Oxford Handbook of Membrane Computing*, Oxford University Press, New York, NY, USA, 2010.

[18] E. Sánchez-Karhunen and L. Valencia-Cabrera, "Modelling complex market interactions using PDP systems," *Journal of Membrane Computing*, vol. 1, pp. 40–51, 2019.

[19] A. Paun, "On P systems with active membranes," in *Proceedings of the Second International Conference on Unconventional Models of Computation*, pp. 187–201, Springer, London, UK, 2000.

[20] G. Păun, "P systems with active membranes attacking NP-complete problems," *Journal of Automata, Languages and Combinatorics*, vol. 6, no. 1, pp. 75–90, 2001.

[21] A. Alhazov, L. Pan, and G. Păun, "Trading polarizations for labels in P systems with active membranes," *Acta Informatica*, vol. 41, no. 2-3, pp. 111–144, 2004.

[22] G. Ciobanu, L. Pan, G. Păun, and M. J. Pérez-Jiménez, "P systems with minimal parallelism," *Theoretical Computer Science*, vol. 378, no. 1, pp. 117–129, 2007.

[23] L. Pan, D. P. Daniel, and M. J. Pérez-Jiménez, "Computation of ramsey numbers by P systems with active membranes," *International Journal of Foundations of Computer Science*, vol. 22, no. 1, pp. 29–38, 2011.

[24] L. Pan and C. Martín-Vide, "Solving multidimensional 0-1 knapsack problem by P systems with input and active membranes," *Journal of Parallel and Distributed Computing*, vol. 65, no. 12, pp. 1578–1584, 2005.

[25] L. Pan and C. Martín-Vide, "Further remark on P systems with active membranes and two polarizations," *Journal of Parallel and Distributed Computing*, vol. 66, no. 6, pp. 867–872, 2006.

[26] P. Sosík, "The computational power of cell division in P systems: Beating down parallel computers?" *Natural Computing*, vol. 2, no. 3, pp. 287–298, 2003.

[27] P. Sosík, A. Rodríguez-Patón, and L. Ciencialová, "Standardized proofs of PSPACE-completeness of P systems with active membranes," in *Proceedings of the Eighth Brainstorming Week on Membrane Computing*, pp. 301–310, 2010.

[28] A. Alhazov, C. Martín-Vide, and L. Pan, "Solving a PSPACE-complete problem by recognizing P systems with restricted active membranes," *Fundamenta Informaticae*, vol. 58, no. 2, pp. 67–77, 2003.

[29] A. Alhazov and M. J. Pérez-Jiménez, "Uniform solution of QSAT using polarizationless active membranes," *Lecture Notes in Computer Science*, vol. 4664, pp. 122–133, 2007.

[30] M. Cavaliere and D. Sburlan, "Time-independent P systems," *Lecture Notes in Computer Science*, vol. 3365, pp. 239–258, 2005.

[31] M. Gheorghe, G. Păun, M. J. Pérez-Jiménez, and G. Rozenberg, "Research frontiers of membrane computing: open problems and research topics," *International Journal of Foundations of Computer Science*, vol. 24, no. 5, pp. 547–623, 2013.

[32] Y. Y. Niu and Z. G. Wang, "Time-free solution for QSAT by using timed Tissue P systems," *Applied Mechanics and Materials*, vol. 568-570, pp. 812–816, 2014.

[33] B. Song and L. Pan, "Computational efficiency and universality of timed P systems with active membranes," *Theoretical Computer Science*, vol. 567, pp. 74–86, 2015.

[34] B. Song, M. J. Pérez-Jiménez, and L. Pan, "Computational efficiency and universality of timed P systems with membrane creation," *Soft Computing*, vol. 19, no. 11, pp. 3043–3053, 2015.

[35] B. Song, M. J. Pérez-Jiménez, and L. Pan, "An efficient time-free solution to SAT problem by P systems with proteins on membranes," *Journal of Computer and System Sciences*, vol. 82, no. 6, pp. 1090–1099, 2016.

[36] B. Song, M. J. Pérez-Jiménez, and L. Pan, "An efficient time-free solution to QSAT problem using P systems with proteins on membranes," *Information and Computation*, vol. 256, pp. 287–299, 2017.

[37] B. Song, T. Song, and L. Pan, "A time-free uniform solution to subset sum problem by tissue P systems with cell division," *Mathematical Structures in Computer Science*, vol. 27, no. 1, pp. 17–32, 2017.

[38] G. Paun, *Computing with Membranes: An Introduction*, Springer-Verlag, Berlin, Germany, 2002.

[39] B. Song, T. Song, and L. Pan, "Time-free solution to SAT problem by P systems with active membranes and standard cell division rules," *Natural Computing*, vol. 14, no. 4, pp. 673–681, 2015.

[40] C. H. Papadimitriou, *Computational Complexity*, Addison-Wesley, Massachusetts, USA, 1994.

[41] M. Cashmore, M. Fox, and E. Giunchiglia, "Planning as quantified boolean formula," *Frontiers in Artificial Intelligence and Applications*, vol. 242, pp. 217–222, 2012.

[42] X. Gu and W. Ma, "On a class of coupled Hamiltonian operators and their integrable hierarchies with two potentials," *Mathematical Methods in the Applied Sciences*, vol. 41, no. 10, pp. 3779–3789, 2018.

[43] Y. Li, Y. Sun, F. Meng, and Y. Tian, "Exponential stabilization of switched time-varying systems with delays and disturbances," *Applied Mathematics and Computation*, vol. 324, pp. 131–140, 2018.

[44] H. Mangassarian, A. Veneris, and M. Benedetti, "Robust QBF encodings for sequential circuits with applications to verification, debug, and test," *IEEE Transactions on Computers*, vol. 59, no. 7, pp. 981–994, 2010.

[45] P. Marin, M. Narizzano, L. Pulina et al., "Twelve years of QBF evaluations: QSAT is PSPACE-hard and it shows," *Fundamenta Informaticae*, vol. 149, no. 1-2, pp. 133–158, 2016.

[46] B. Hu, T. Xia, and W. Ma, "Riemann–Hilbert approach for an initial-boundary value problem of the two-component modified Korteweg-de Vries equation on the half-line," *Applied Mathematics and Computation*, vol. 332, pp. 148–159, 2018.

[47] J. Ha, H. Zhang, and Q. Zhao, "Exact solutions for a Dirac-type equation with N-fold Darboux transformation," *Journal of Applied Analysis and Computation*, vol. 9, no. 1, pp. 200–210, 2019.

[48] L. Liu, F. Deng, and T. Hou, "Almost sure exponential stability of implicit numerical solution for stochastic functional differential equation with extended polynomial growth condition," *Applied Mathematics and Computation*, vol. 330, pp. 201–212, 2018.

[49] H. Riener and G. Fey, "Exact diagnosis using boolean satisfiability," in *Proceedings of the the EEE/ACM International Conference on Computer-Aided Design*, pp. 1–8, Austin, Tex, USA, November 2016.

[50] Z. Tu and X. Zeng, "Two classes of permutation trinomials with Niho exponents," *Finite Fields and Their Applications*, vol. 53, pp. 99–112, 2018.

[51] H. Wu and J. Song, "Mixed lump–stripe soliton solutions to a dimensionally reduced generalized Jimbo–Miwa equation," *Applied Mathematics Letters*, vol. 90, pp. 181–187, 2019.