Hindawi
Mathematical Problems in Engineering
Volume 2017, Article ID 1567378, 8 pages
https://doi.org/10.1155/2017/1567378

Research Article

Time-Free Solution to SAT Problem by Tissue P Systems

Yueguo Luo,[1,2] Zhongyang Xiong,[1] and Guanghua Zhang[1]

[1]*College of Computer Science, Chongqing University, Chongqing 400030, China*
[2]*College of Computer Engineering, Yangtze Normal University, Chongqing 408100, China*

Correspondence should be addressed to Yueguo Luo; ygluo@cqu.edu.cn

Received 15 December 2016; Revised 15 January 2017; Accepted 19 January 2017; Published 23 February 2017

Academic Editor: Nicolas Hudon

Tissue P systems are a class of computing models inspired by intercellular communication, where the rules are used in the nondeterministic maximally parallel manner. As we know, the execution time of each rule is the same in the system. However, the execution time of biochemical reactions is hard to control from a biochemical point of view. In this work, we construct a uniform and efficient solution to the SAT problem with tissue P systems in a time-free way for the first time. With the P systems constructed from the sizes of instances, the execution time of the rules has no influence on the computation results. As a result, we prove that such system is shown to be highly effective for NP-complete problem even in a time-free manner with communication rules of length at most 3.

1. Introduction

As a well-known NP-complete problem, SAT problem has been widely used in artificial intelligence and electronic design automation. Many applications can be considered as a decision procedure to determine if a specific instance is SAT or UNSAT. As we know, some traditional SAT algorithms, such as DP [1] and Wu's method [2], have an exponential time complexity. Compared with these algorithms, solving the SAT problem in a distributed and parallel manner may be an efficient method. Hence, we naturally consider the computing models in the framework of membrane computing, which can solve computationally hard problems efficiently.

Membrane computing is a new area of computer science in recent years. It is introduced by Păun in [3]. Since then, this research area has been widely concerned by researchers. The computing models are inspired by biological phenomena and biological characteristics like other methods of natural computing, such as particle swarm optimization algorithms, ant colony algorithms, and genetic algorithms. The models abstract computing ideas from the structure and function of individual cells and from complexes of cells, such as tissues and organs (including the brain). Based on these biological facts, there are mainly three types of P systems which have been introduced: cell-like P systems, tissue P systems, and neural-like P systems. In traditional tissue P systems, rules are used in a synchronous, parallel, and nondeterministic way. Note that evolution rules in each cell follow the principle of maximal parallelism. Therefore, NP-hard problems can be solved efficiently. Until now, many kinds of P systems can solve some NP-complete problems, such as SAT [4–10], 3-coloring problem [11], and Hamiltonian cycle problem [12]. In addition, many new variants of P systems have equivalent computing power with Turing machine and to be Turing universal as language generators [13–15]. Inspired by some biological facts, several methods were proposed to produce new membranes in living cells: membrane division [5, 16], membrane creation [4], and membrane separation [7]. In this paper, we pay close attention to tissue P systems with cell division. In [16], tissue P systems with cell division were proposed. This computing model is inspired by the biological fact that cells are duplicated via mitosis. By this method, an exponential amount of cells can be generated. It has been proved that tissue P systems can solve some computational hard problems in polynomial time (even linear time). The first attempt in this topic was done in [16], where SAT problem was solved by tissue P systems with cell division in polynomial time.

SAT problem has been widely investigated by cell-like P systems [4–8]. However, it is rarely investigated by tissue P systems. In [9, 16], SAT problem was solved by tissue P systems in polynomial time. However, in these works, each rule is assumed to be completed in one accurate unit time. Hence, every rule in these systems has the same execution time. However, the execution time of biochemical reactions is uncontrollable. Therefore, it is necessary to construct a system that works independently of the execution time of rules. The concept of time-free was first introduced in [17], and the open problems with solutions for NP-complete problems were formulated in [18]. Recently, some NP-complete problems have been solved in a time-free manner [19–23]. However, the model of these P systems is based on cell-like P systems. In this work, we solve SAT problem with timed tissue P systems by cell division in a uniform way. As far as we know, there are no other relevant papers on solving SAT problem with tissue P systems in a time-free manner.

The paper is structured as follows: firstly, we give brief descriptions of the basic model of tissue P systems with cell division. Then, timed P systems with tissue P systems to decision problems will be proposed. In Section 3, we prove that SAT problem can be solved by tissue P systems in a time-free way. Finally, some formal details and conclusions are presented.

2. Tissue P Systems with Cell Division

2.1. Tissue P Systems with Cell Division. The membrane structures of tissue P systems are described by general graphs, where cells correspond to nodes of a graph, and communication channels correspond to edges between two nodes. If there exists a communication channel, objects can communicate between two cells (or a cell and the environment) with communication rules.

Definition 1. Formally, a tissue P system (of degree $m \geq 1$) with cell division can be defined as the form

$$\Pi = \left(O, E, w_1, \ldots, w_m, R, i_{\text{out}}\right), \tag{1}$$

where

(i) O is the finite nonempty alphabet of objects,

(ii) $E \subseteq O$ is the set of objects, which are initially placed in the environment with arbitrary number of copies,

(iii) w_i $(1 \leq i \leq m)$ are finite multisets over O,

(iv) i_{out} is output region of membrane structure, and it saves the results,

(v) R is a finite set of rules. There are mainly the following two forms:

(a) communication rules: $(i, u/v, j)$, where $i, j \in \{0, 1, 2, \ldots, m\}$, $i \neq j$, $u, v \in O^*$, $|uv| > 0$, and i and j correspond to the cell i and cell j, respectively. When $i = 0$ or $j = 0$, it corresponds to the environment. If $u \neq \lambda$ and $v \neq \lambda$, a communication rule is called an antiport rule; otherwise it is a symport rule. When there are objects represented by multiset u in cell i and objects represented by multiset v in cell j, a communication rule can be applied. With applying this rule, the objects denoted by u are sent into region j; at the same time, the objects denoted by v are sent into region i in the opposite direction. These objects can evolve by given rules in a synchronous, parallel, and nondeterministic way,

(b) division rules: $[a]_i \rightarrow [b]_i[c]_i$, where $i \in \{0, 1, 2, \ldots, m\}$; $a, b, c \in O$; $i \neq i_{\text{out}}$. Under the influence of the object present in a cell, this rule can be applied to divide the cell into two copies of cells (with the same label with i); object b and object c in the two cells can be generated, respectively. The remaining objects in the original cell enter the newly generated cells, respectively. When a cell is being divided, the division rule is the only one which is applied for that cell at that step.

Before the system begins to run, the initial configuration is represented by $(w_1, \ldots, w_m)$; that is, objects denoted by $w_1, \ldots, w_m$ are placed in the corresponding cells. We can get transitions between the configurations by applying rules of the system as described above. The current configuration is described by multisets of objects. Finally, the system will stop running. At that moment, there is no rule which is being applied, and no rule can be applied. In each cell, it must be stressed that the system works with the following principles:

(i) Nondeterminism: for each computation step, rules and objects are nondeterministically chosen; that is, any possible execution of rules can start randomly.

(ii) Maximal parallelism: at one computation step, if no other rules can be added to be applied, all rules which can be applied have to be applied to all possible objects.

2.2. Recognizer Timed Tissue P Systems with Cell Division. A P system that generates (or accepts) the same family of vectors of natural numbers, independently of the value assigned to the execution time of each rule r, is called time-free [18].

Definition 2. A recognizer timed tissue P system can be defined as the form

$$\Pi = \left(O, \Sigma, E, w_1, \ldots, w_m, R, e, i_{\text{in}}, i_{\text{out}}\right), \tag{2}$$

where

(i) O is the finite nonempty alphabet of objects;

(ii) Σ is an (input) alphabet strictly contained in O, and $\Sigma \cap E = \emptyset$;

(iii) $E \subseteq O$ is the set of objects, which are initially placed in the environment with arbitrary number of copies;

(iv) w_i $(1 \leq i \leq m)$ are finite multisets over O;

(v) R is a finite set of rules;

(vi) we can specify the execution time of each rule by a mapping e: $R \to \mathbb{N}$, where $\mathbb{N}$ is the set of nonpositive integers;

(vii) $i_{\text{in}} \in \{1, 2, \ldots, m\}$ is the input cell;

(viii) i_{out} is the environment, which saves the results;

(ix) the working alphabet contains two distinguished elements `yes` and `no`;

(x) all the computations halt;

(xi) when the system halts, either object `yes` or `no` (but not both) must appear in the environment.

The input multiset has been added to the contents of the input cell i_{in}. Thus, we have an initial configuration associated with each input multiset. For the multiset w over the input alphabet Σ, the computation of the system with input w starts as the form

$$\left(w_1, \ldots, \left(w_{i_{\text{in}}} + w\right), \ldots, w_m\right). \tag{3}$$

At one step, if no rules can be applied to the current configuration and no rules are being applying, the system will stop running. In this case, the object of output cell is computing result in the stopping configuration. If the object `yes` appears in the environment when the system stops, it is an *accepting computation*; on the contrary, if the object `no` appears, it is a *rejecting computation*.

Definition 3. A timed tissue P system with cell division is a pair (Π, e), where Π is a tissue P system with cell division and e represent the execution times of the rules. We denote by $\Pi(e)$ the timed tissue P system with cell division.

$\Pi(e)$ works in the following way: an external clock is supposed to mark time units of equal length, starting from time 0. If a rule r from R is selected to be executed, we denote $e(r)$ as the time which the rule r lasts. If the application of a rule r starts at time j, the execution of this rule terminates at time $j+e(r)$. It means that the rule lasts $e(r)$ steps. The objects and the cells produced by the rule are not available until the beginning of step $j + e(r) + 1$.

2.3. Polynomial Complexity Classes of Recognizer Timed Tissue P Systems with Cell Division

Definition 4. Let $X = (I_X, \theta_X)$ be a decision problem (with `yes`/`no` answer), where I_X is a set of instances and θ_X is a predicate over I_X, and let $\mathbf{\Pi} = \{\Pi(n) \mid n \in \mathbb{N}\}$ be a family of recognizer P systems. We use *rule starting step* (RS-step, for short) to define the computation step in timed P systems [18].

Because the system works in a time-free manner, the execution time of each rule is no longer one accurate unit time. $\mathbf{\Pi}$ computes a solution of an instance, and the system works independently of any time mapping e. Hence, the systems $\mathbf{\Pi}$ can solve problem X in a time-free manner.

Definition 5. A decision problem $X = (I_X, \theta_X)$ is solvable in polynomial time and in uniform way by a family $\mathbf{\Pi} = \{\Pi(n) \mid n \in \mathbb{N}\}$ of recognizer timed tissue P systems, if the following holds:

(1) The family Π is polynomially uniform with respect to Turing machines; namely, there exists a deterministic Turing machine which constructs the system $\Pi(n)$ with knowledge involving only the size of the problem X for every instance of X working in polynomial time.

(2) There exists a pair (cod, s) of polynomial time computable functions over I_X such that

(i) for each instance $u \in I_X$, $s(u)$ is a natural number and $\text{cod}(u)$ is an input multiset of the system $\Pi(s(u))$,

(ii) the family $\mathbf{\Pi}$ is time-free sound with regard to (X, cod, s). For each instance $u \in I_X$ such that there exists an accepting computation if $\Pi(s(u), e)$ with input $\text{cod}(u)$, we have $\theta_X(u) = 1$,

(iii) the family $\mathbf{\Pi}$ is time-free complete with regard to (X, cod, s). For each instance $u \in I_X$, if $\theta_X(u) = 1$, then every computation of $\Pi(s(u), e)$ with input $\text{cod}(u)$ is an accepting one for any time mapping e,

(iv) the family $\mathbf{\Pi}$ is time-free polynomially bounded with regard to (X, cod, s); namely, there exists a polynomial function $p(n)$ such that for each $u \in I_X$, when all the computations in $\Pi(s(u), e)$ with input $\text{cod}(u)$ halt, there is at most $p(|u|)$ RS-steps for any time mapping e.

For the set of decision problems, we denote it with $\text{PMC}_{\text{TF-TP}(k)}$ which can be solved by recognizer timed tissue P systems and the length of evolution rules at most k working.

3. A Time-Free Uniform Solution to `SAT` Problem Using Tissue P Systems

Definition 6. The `SAT` problem is, for a Boolean formula of CNF, to look for whether or not there exists an assignment to its variables on which it is evaluated to be true [4].

The `SAT` problem is a well-known NP-complete problem. This problem has been widely used in artificial intelligence and computer theories. Boolean formula containing n variables has 2^n values. By membrane division, we can obtain 2^n membranes in n steps. In addition, by the maximal parallelism, we can obtain polynomial (maybe, linear) solutions to NP-hard problems. In what follows, we will propose a time-free uniform solution to the `SAT` problem; that is, P systems are constructed from the size of instances of the `SAT` problem in a time-free manner.

Theorem 7. `SAT` *problem can be solved by a uniform family of timed tissue P systems by cell division.*

Proof. For a `SAT` formula with n Boolean variables and m clauses, its CNF form is input into P systems. Let us consider

a propositional formula $\gamma = C_1 \wedge C_2 \wedge \cdots \wedge C_m$, consisting of m clauses $C_j = y_{1,j} \wedge \cdots \wedge y_{p_j,j}$, where $y_{i,j} \in \{x_l, \neg x_l \mid 1 \le l \le n\}$. $1 \le i \le p_j$, $1 \le j \le m$.

Let us consider the polynomial–time computable function (the pair function)

$$g(n,k) = \left(\frac{(n+k)(n+k+1)}{2}\right) + n. \tag{4}$$

It is a primitive recursive and bijective function from $\mathbb{N}^2$ to $\mathbb{N}$. □

The instance γ is encoded by multiset as follows:

$$\begin{aligned}&\operatorname{cod}(\gamma)\\ &= \alpha_{1,1}\alpha_{1,2}\cdots\alpha_{1,m}\alpha_{2,1}\alpha_{2,2}\cdots\alpha_{2,m}\cdots\alpha_{n,1}\alpha_{n,2}\cdots\alpha_{n,m}.\end{aligned} \tag{5}$$

We codify γ, where for $1 \le i \le n$, $1 \le j \le m$, by the multiset

$$\alpha_{i,j} = \begin{cases} \beta_{i,j} & \text{if } x_i \text{ appears in } C_j, \\ \overline{\beta_{i,j}} & \text{if } \neg x_i \text{ appears in } C_j, \\ \beta^0_{i,j} & \text{if } x_i \text{ and } \neg x_i \text{ do not appear in } C_j. \end{cases} \tag{6}$$

Next, we define a family of recognizer tissue P systems with cell division, which can process all instances γ provided the appropriate input multiset $\operatorname{cod}(\gamma)$. The instance γ will be processed by the tissue P system with input $\operatorname{cod}(\gamma)$. For the given n variables and m clauses, we construct the recognizer tissue P system as follows:

$$\Pi_{\mathrm{SAT}(m,n)} = (O, \Sigma, E, w_1, w_2, R, e, i_{\mathrm{in}}, i_{\mathrm{out}}), \tag{7}$$

where

(i) O is the finite alphabet of the system,

$O = \Sigma \cup \{a_i 1 \le i \le n+1\} \cup \{t_{i,j}, f_{i,j} 1 \le i \le n,\ 1 \le j \le m+1\} \cup \{e_i 2 \le i \le n+1\} \cup \{b_{i,j}, T_i, F_i, r_j 1 \le i \le n,\ 1 \le j \le m\} \cup \{s_j 2 \le j \le m+1\} \cup \{d, \mathtt{yes}, \mathtt{no}\}$,

the objects in the O mainly contain

(a) $t_{i,j}$ denotes *true* of the Boolean value for the variable;

(b) $f_{i,j}$ denotes *false* of the Boolean value for the variable;

(c) the object `yes` means that the CNF is satisfiable;

(d) the object `no` means that the CNF is unsatisfiable,

(ii) $\Sigma = \{\beta_{i,j}, \beta^0_{i,j}, \overline{\beta}_{i,j} \mid 1 \le i \le n,\ 1 \le j \le m\}$ is the input alphabet contained in O,

(iii) E is the arbitrary copies of objects in the environment,

$E = \{a_i 2 \le i \le n+1\} \cup \{t_{i,j}, f_{i,j} 1 \le i \le n,\ 2 \le j \le m+1\} \cup \{e_i 2 \le i \le n+1\} \cup \{b_{i,j}, T_i, F_i, r_j 1 \le i \le n,\ 1 \le j \le m\} \cup \{s_j 2 \le j \le m+1\} \cup \{d\}$,

(iv) w_i $(1 \le i \le m)$ are finite multisets over O, $w_1 = \{\mathtt{yes}, \mathtt{no}\}$, $w_2 = a_1$,

(v) $i_{\mathrm{in}} = 2$ is the input cell; $i_{\mathrm{out}} = 0$ is the output region which saves the results,

(vi) We can specify the execution time of each rule by a mapping e: $R \to \mathbb{N}$, where R is the set of rules of $\Pi_{\mathrm{SAT}(m,n)}$. R is the following set of rules.

(a) Generation Phase

$$\begin{aligned}
&R_{1,i}\colon\ [a_i]_2 \longrightarrow [t_{i,1}]_2 [f_{i,1}]_2, \quad 1 \le i \le n,\\
&R_t \begin{cases} R_{2,i,j}\colon \left\{\left(2, t_{i,j}\beta_{i,j}/b_{i,j}, 0\right), \left(2, b_{i,j}/r_j t_{i,j+1}, 0\right),\ 1 \le i \le n,\ 1 \le j \le m\right\}, \\ R_{3,i,j}\colon \left(2, t_{i,j}\overline{\beta_{i,j}}/t_{i,j+1}, 0\right), \quad 1 \le i \le n,\ 1 \le j \le m, \\ R_{4,i,j}\colon \left(2, t_{i,j}\beta^0_{i,j}/t_{i,j+1}, 0\right), \quad 1 \le i \le n,\ 1 \le j \le m, \end{cases}\\
&R_f \begin{cases} R_{5,i,j}\colon \left\{\left(2, f_{i,j}\overline{\beta_{i,j}}/b_{i,j}, 0\right), \left(2, b_{i,j}/r_j f_{i,j+1}, 0\right),\ 1 \le i \le n,\ 1 \le j \le m\right\}, \\ R_{6,i,j}\colon \left(2, f_{i,j}\beta_{i,j}/f_{i,j+1}, 0\right), \quad 1 \le i \le n,\ 1 \le j \le m, \\ R_{7,i,j}\colon \left(2, f_{i,j}\beta^0_{i,j}/f_{i,j+1}, 0\right), \quad 1 \le i \le n,\ 1 \le j \le m, \end{cases}\\
&R_{8,i}\colon \left(2, t_{i,m+1}/T_i d, 0\right), \quad 1 \le i \le n,\\
&R_{9,i}\colon \left(2, f_{i,m+1}/F_i d, 0\right), \quad 1 \le i \le n,\\
&R_{10,i}\colon \left(2, T_i/\lambda, 1\right), \quad 1 \le i \le n,\\
&R_{11,i}\colon \left(2, F_i/\lambda, 1\right), \quad 1 \le i \le n,\\
&R_{12,i}\colon \left(1, T_i F_i/e_{i+1}, 0\right), \quad 1 \le i \le n,\\
&R_{13,i}\colon \left(1, e_i/a_i^2, 0\right), \quad 2 \le i \le n+1,\\
&R_{14,i}\colon \left(2, d/a_i, 1\right), \quad 2 \le i \le n+1.
\end{aligned} \tag{8}$$

(b) Checking Phase

$$R_{15}\colon\ (2, a_{n+1}r_1/s_2, 0)\,,$$
$$R_{16,j}\colon\ \left(2, s_j r_j/s_{j+1}, 0\right),\quad 2 \le j \le m. \tag{9}$$

(c) Output Phase

$$R_{17}\colon\ (1, \mathtt{no}/\lambda, 0)\,,$$
$$R_{18}\colon\ (2, s_{m+1}/\mathtt{yes}, 1)\,, \tag{10}$$
$$R_{19}\colon\ (2, \mathtt{yes}/\mathtt{no}, 0)\,.$$

The computing process consists of the following phases:

(a) Generation phase: all assignments of variables can be generated by using membrane division. Finally, 2^n copies of membranes can be generated.

(b) Checking phase: the system checks whether or not the formula has a truth assignment. If all assignments of variables cannot satisfy the clause, it means that there is no satisfiable solution; or it means that there is at least one satisfiable solution.

(c) Output phase: the object yes/no of output membrane is computing result in the stopping configuration.

The multiset of an instance is introduced into input cell of the system, and we can get the initial configuration. In what follows, we will give an overview of computing process to show how the system works. In order to have a global understanding of the subsequent process of proof, we classify all the rules as follows:

(i) $R_{1,i}$ $(1 \le i \le n)$ are division rules which can be used to produce an exponential space.

(ii) The rules R_t and R_f are used to check whether the corresponding clauses are satisfied by the assignment *true* or *false* of variable x_i $(1 \le i \le n)$.

(iii) The rules from $R_{8,i,j}$ to $R_{14,i}$ are used for the next iteration of variable x_i or preparing for checking phase.

(iv) With the rules R_{15} and $R_{16,i}$, the final detection results can be generated in each cell.

(v) With the rules from R_{17} to R_{19}, the system sends to the environment the final answer.

(a) Generation Phase. Let e be the execution time of the rules. Initially, cell 1 contains objects yes and no; cell 2 contains object a_1, cod(γ). Object a_1 in cell 2 corresponds to variable x_1.

In general, objects a_i in the cell with label 2 correspond to variable x_i, $1 \le i \le n$. Boolean formula containing n variables has 2^n values. By cell division, 2^n copies of cells can be generated in n steps. Hence, we can obtain all the possible solutions of SAT problem in the end. At step 1, the system starts running, and rules $R_{1,1}$ and R_{17} can be applied at the same time. Rule $R_{1,1}$ is a division rule, which can be applied to divide this cell into new cells with the same label. At the same time, object a_1 evolves to objects $t_{1,1}$ and $f_{1,1}$ in the two newly generated cells: object $t_{1,1}$ corresponds to assignment *true* of variable x_1, and object $f_{1,1}$ corresponds to assignment *false* of variable x_1. In this process, there is only one RS-step because no other rules can be applied except for the rules $R_{1,1}$ and R_{17}. When the execution of rule $R_{1,1}$ finishes, a rule in R_t (resp., R_f) can be enabled. These rules correspond to check whether the assignment (*true* or *false*) to the variable x_1 in each clause is satisfiable.

When object $t_{1,j}$ and object $\beta_{1,j}$ appear in cell 2 or object $f_{1,j}$ and object $\overline{\beta_{1,j}}$ appear in cell 2, it means that the assignment of the current variable x_1 can satisfy the clause. At this moment, by the application of the first rule in $R_{2,i,j}$ (resp., $R_{5,i,j}$), object $b_{i,j}$ can be generated. When the execution of this rule has completed, the second rule in $R_{2,i,j}$ (resp., $R_{5,i,j}$) can be applied. By using the rule, object $b_{i,j}$ in cell 2 can be sent out the cell, and object r_j can be generated, which represent that this clause can be satisfied by the assignment. For object $t_{1,j}$ (resp., $f_{1,j}$), $1 \le i \le n$, by using the rule $R_{2,i,j}$, $R_{3,i,j}$, or $R_{4,i,j}$ (resp., $R_{5,i,j}$, $R_{6,i,j}$, or $R_{7,i,j}$), the second component of the subscript will increase one by one. In general, for the current variable x_i $(1 \le i \le n)$, there are m clauses; that is, the second component of the subscript can reach to $m + 1$ in the end. Thus, when the application of all the rules in R_t (resp., R_f) completes, object $t_{i,m+1}$ (resp., $f_{i,m+1}$) can be generated. Because the first two rules in R_t and R_f are started at the same time, these two rules take only one RS-step. In addition, we consider maximum number of rules in R_t and R_f, namely, $R_{2,i,j}$ and $R_{5,i,j}$. Note that the two rules ($R_{2,i,j}$ and $R_{5,i,j}$) cannot be executed together because the same variable in the same clause is either x_i or $\neg x_i$. Therefore, it takes at most $3m - 1$ RS-steps to complete this process.

Figure 1 shows the division process of n variables, which can be represented by a binary tree. Each variable of a CNF formula is divided into two parts in an iterative manner. Each node of the binary tree is assigned to *true* or *false*, which represents the assignment of a variable. Left and right subtree branches indicate the *true* value and the *false* value, respectively. For every iteration of variable x_i, when the execution of rule R_t (resp., R_f) finishes, at that moment, object $t_{i,m+1}$ (resp., $f_{i,m+1}$) can be generated in every cell with label 2. For example, for the iteration of variable x_3 in Figure 1, there are four copies of object $t_{i,m+1}$ (resp., $f_{i,m+1}$) generated at the same time. When division rule completes, the rules in R_t and R_f start simultaneously in each cell 2. For every cell containing object $t_{i,m+1}$ (resp., $f_{i,m+1}$), in which the same rules are applied, object $t_{i,m+1}$ (resp., $f_{i,m+1}$) in each cell with label 2 is generated at the same time. But $t_{i,m+1}$ and $f_{i,m+1}$ cannot be generated at the same time because the execution time of rules in R_t and R_f is different.

According to the analysis above, the applications of rules $R_{8,1}$ and $R_{9,1}$ start at different steps. When an object $t_{1,m+1}$ (resp., $f_{1,m+1}$) is generated, rule $R_{8,1}$ (resp., $R_{9,1}$) can be applied, by which cell 2 sends out object $t_{1,m+1}$ (resp., $f_{1,m+1}$) to the environment and receives multiset $T_1 d$ (resp., $F_1 d$) from the environment. Thus, the application of rule $R_{8,i}$ (resp., $R_{9,i}$) can ensure that only one copy of object d can be generated in each cell with label 2. When the object T_1 (resp.,

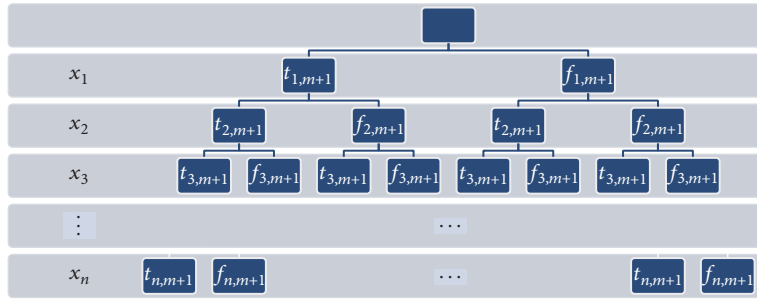

Figure 1: The division process of variables.

F_1) is generated in a cell 2, rule $R_{10,1}$ (resp., $R_{11,1}$) is applied, by which object T_1 (resp., F_1) can be sent to cell 1. When the execution of both rule $R_{10,1}$ and rule $R_{11,1}$ finishes, rule $R_{12,1}$ can be applied. At this moment, object T_1 and F_1 can appear in the cell 1. By using the rule $R_{12,1}$, object T_1 and F_1 in cell 1 can be sent out that cell. At the same time, object e_2 enters the cell 1.

As we know, it is obvious that $e(R_t) + e(R_{8,1}) + e(R_{10,1})$ may not be equal to $e(R_f) + e(R_{9,1}) + e(R_{11,1})$ because the system works independently of the execution time of rules. Hence, the rule $R_{12,1}$ is applied to solve synchronization problem of the rules. When the application of rule $R_{12,1}$ finishes, the execution of all the aforementioned rules has completed. At that moment, cell 1 has one copy of object e_2. By using the rule $R_{13,2}$, two copies of object a_2 enter the cell 1. Note that each cell 2 has an object d; by using the rule $R_{14,2}$, each cell 2 will obtain one copy of object a_2 simultaneously because of the principle of maximal parallelism; that is, all of the rules that can be applied must be applied simultaneously. Note that only one copy of object d appears in each cell with label 2; eventually, each cell 2 contains one copy of object a_{i+1}.

The system runs in this way, and object a_2 must appear in each cell 2 simultaneously. When the application of rule $R_{14,2}$ is completed, note that the execution of all the aforementioned rules has been completed for the first iteration of variable x_i ($1 \le i \le n$). These available rules are applied in the order

$$\{R_{1,1}\} \to \{\{R_{2,1,j}, R_{3,1,j}, R_{4,1,j}\} \to \{R_{8,1}\} \to \{R_{10,1}\} \text{ (resp., } \{R_{5,1,j}, R_{6,1,j}, R_{7,1,j}\} \to \{R_{9,1}\} \to \{R_{11,1}\})\} \to \{R_{12,1}\} \to \{R_{13,2}\} \to \{R_{14,2}\}.$$

When the application of all rules above is completed, it takes at most $3m + 7$ RS-steps.

When the first iteration of variable x_i ($1 \le i \le n$) is completed, every cell 2 contains object a_2. Under the influence of object a_2, rule $R_{1,2}$ can be applied simultaneously. With the application of the division rule, four cells with label 2 can be generated. Similarly, the system executes the above process in an iterative manner till the assignments of all the variables are obtained.

The available rules from $R_{1,1}$ to $R_{14,n}$ are applied in the following order:

The application of rules for the first iteration:

$$\{R_{1,1}\} \to \{\{R_{2,1,j}, R_{3,1,j}, R_{4,1,j}\} \to \{R_{8,1}\} \to \{R_{10,1}\} \text{ (resp., } \{R_{5,1,j}, R_{6,1,j}, R_{7,1,j}\} \to \{R_{9,1}\} \to \{R_{11,1}\})\} \to \{R_{12,1}\} \to \{R_{13,2}\} \to \{R_{14,2}\}.$$

The application of rules for the second iteration:

$$\{R_{1,2}\} \to \{\{R_{2,2,j}, R_{3,2,j}, R_{4,2,j}\} \to \{R_{8,2}\} \to \{R_{10,2}\} \text{ (resp., } \{R_{5,2,j}, R_{6,2,j}, R_{7,2,j}\} \to \{R_{9,2}\} \to \{R_{11,2}\})\} \to \{R_{12,2}\} \to \{R_{13,3}\} \to \{R_{14,3}\}.$$

$\vdots$

The application of rules for the last iteration:

$$\{R_{1,n}\} \to \{\{R_{2,n,j}, R_{3,n,j}, R_{4,n,j}\} \to \{R_{8,n}\} \to \{R_{10,n}\} \text{ (resp., } \{R_{5,n,j}, R_{6,n,j}, R_{7,n,j}\} \to \{R_{9,n}\} \to \{R_{11,n}\})\} \to \{R_{12,n}\} \to \{R_{13,n+1}\} \to \{R_{14,n+1}\}.$$

In general, when the application of rules from $R_{1,1}$ to $R_{14,n}$ is completed, this computation process takes at most $3mn + 7n$ RS-steps.

When object a_{n+1} occurs in an arbitrary cell 2, it means that the execution of all the aforementioned rules has been completed. Note that the application of $R_{13,n+1}$ and $R_{14,n+1}$ in each cell follows the principle of maximal parallelism. Therefore, objects a_{n+1} in each cell 2 can be generated simultaneously.

(b) Checking Phase. When the execution of rule $R_{14,n+1}$ is completed, there are some objects (or no object) from the set $\{r_1, r_2, \ldots, r_n\}$ in every cell 2. These objects represent that the corresponding clauses can be satisfied. Thus, if at least one cell contains all objects including $r_1, r_2, \ldots, r_n$, it means that there is a satisfiable solution; or it means that there is no satisfiable solution. At this moment, rule R_{15} is applied in each cell 2 simultaneously. With the appearance of object a_{n+1} and object r_1 (if this object exists in a cell 2) in cells 2, by using the rule R_{15}, the object r_1 is checked and s_2 can be sent to cell 2 from the environment. If the object s_i ($2 \le i \le m$) appears in a cell 2, rule $R_{16,i}$ can be applied. Eventually, if the object s_m appears in a cell 2, rule $R_{16,m}$ can be applied. Thus, object s_{m+1} can be generated in that cell. Please note that the rules are used in a time-free manner. In addition, although there may be more than one copy of object r_i ($1 \le i \le m$), only one copy

can be applied by the rules of R_{15} and $R_{16,i}$. In general, this computation process takes at most m RS-steps.

(c) Output Phase. When the checking phase and the execution of rule R_{17} are complete, there are the following two cases:

(i) *Affirmative answer*: in this case, object s_{m+1} is generated in a cell 2, telling us that the formula is satisfiable; by using the rule R_{18}, objects s_{m+1} can be sent out to cell 1; at the same time, object `yes` enters the cell 2. Eventually, by using the rule R_{19}, the object `yes` in one cell 2 can be sent out to the environment; at the same time, object `no` enters the cell 2. Thus, if the system halts, object `yes` is in the environment; we can draw a conclusion that there is at least one satisfiable solution. Hence, in this case, the formula is satisfiable. This process takes 2 RS-steps.

(ii) *Negative answer*: in this case, object s_{m+1} does not appear in a cell with label 2. Therefore, the rule R_{18} and rule R_{19} cannot be applied. Eventually, when the system halts, if object `no` is still in the environment, we can draw a conclusion that there is no satisfiable solution. Hence, in this case, the formula is not satisfiable. This process takes no RS-step.

4. Some Formal Details

For a `SAT` instance with n Boolean variables and m clauses, the necessary resources to construct $\Pi_{SAT(m,n)}$ can be drawn as follows:

(i) Size of the set O: $6mn + 6n + 2m + 4$.

(ii) Initial number of cells: 2.

(iii) Initial number of objects: 3.

(iv) The total number of rules: $8mn + 8n + m + 3$.

(v) The maximal length of a rule: 3.

$\Pi_{SAT(m,n)}$ always halts till the output is `yes` or `no` in the end. As a recognizer tissue P system, the system stops working within a certain RS-steps. Hence, eventually, object `yes` or `no` must appear in the environment in a feasible time. According to the analysis in Section 3, if the formula γ is satisfiable, then after at most $3mn + 7n + m + 2$ RS-steps the P systems will stop working; if the formula γ is not satisfiable, then after at most $3mn + 7n + m$ RS-steps the P systems will stop working. For both cases, the generation phase takes at most $3mn + 7n$ RS-steps, and the checking phase takes at most m RS-steps; for output phase, the former takes at most 2 RS-steps, and the latter takes no RS-step.

Obviously, there is a polynomial bound of the computation for the system. The `SAT` problem can be solved in polynomial RS-steps by a family of recognizer timed tissue P systems. Hence, the constructed P systems have high computational efficiency to solve the NP-hard problems even in the time-free manner.

Theorem 8. *A family of a timed tissue P system can be constructed as a uniform solution to the* `SAT` *problem in a time-free way. For any time mapping e, the execution time of the rules does not influence the results of the system.*

Proof. $\Pi_{SAT(m,n)}$ always halts till the output is `yes` or `no` in the end. According to the analysis above, the `SAT` can be solved by a family of recognizer tissue P systems in polynomial RS-steps. Hence, the $\Pi_{SAT(m,n)}$ have the powerful computational efficiency to solve the computationally hard problem. Furthermore, the execution time of rules has no influence on the computation results.

It is easy to prove that $\Pi_{SAT(m,n)}$ is time-free sound, time-free complete, and time-free polynomially bounded. Hence, we can draw a conclusion that it is a time-free solution to the `SAT` problem. □

Theorem 9. `SAT` $\in \mathrm{PMC}_{\text{TF-TP(3)}}$.

Proof. It suffices to remark that the `SAT` problem is NP-complete. Note that the family of recognizer P systems given in Section 3 has the length of communication rules which is no more than 3. According to Definition 5 and from the discussion in the previous subsections, it is easy to prove $\mathrm{PMC}_{\text{TF-TP(3)}}$ and that this complexity class is closed under polynomial time reduction and under complement. □

Corollary 10. $\mathrm{NP} \cup \text{co-NP} \subseteq \mathrm{PMC}_{\text{PF(3)}}$.

5. Conclusions

In this work, time-free method is introduced into tissue P systems. We consider that the lasting time of the chemical communication between living cells is often different, because it is influenced by many factors. Thus, a time mapping is added to specify the execution time of rules. Compared with traditional tissue P systems, it is a more realistic model from a biological point of view. The output of the timed tissue P system has no influence on the correctness of the solutions. We prove that `SAT` problem can be efficiently solved by a family of timed tissue P systems with cell division and communication rules with length at most 3, which is a relatively good result from the computational complexity point of view in the time-free way. Our work provides a new and effective solution to `SAT` problem in a distributed and parallel manner.

Finally, the reader can consider further problems. For instance, in Section 3, we prove that NP-complete problem can be solved by timed tissue P systems with communication rules with length at most 3. We can consider what will happen if only symport rules or antiport rules are allowed in timed tissue P systems.

Competing Interests

The authors declare that they have no competing interests.

Acknowledgments

This research is supported by the Educational Science Foundation of Chongqing China (KJ15012016), the Big Data Agricultural Research Center of Southeast Chongqing (2015XJPT02), and the Science Foundation of Yangtze Normal University (2014QN023).

References

[1] M. Davis and H. Putnam, "A computing procedure for quantification theory," *Journal of the ACM*, vol. 7, no. 3, pp. 201–215, 1960.

[2] S. He and B. Zhang, "Solving SAT by algorithm transform of Wu's method," *Journal of Computer Science & Technology*, vol. 14, no. 5, pp. 468–480, 1999.

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

[4] M. A. Gutierrez-Naranjo, M. J. Pérez-Jiménez, and F. J. Romero-Campero, "A uniform solution to SAT using membrane creation," *Theoretical Computer Science*, vol. 371, no. 1-2, pp. 54–61, 2007.

[5] G. Păun, Y. Suzuki, H. Tanaka, and T. Yokomori, "On the power of membrane division in P systems," *Theoretical Computer Science*, vol. 324, no. 1, pp. 61–85, 2004.

[6] Z. Gazdag and G. Kolonits, "A new approach for solving SAT by p systems with active membranes," in *Membrane Computing: 13th International Conference, CMC 2012, Budapest, Hungary, August 28–31, 2012, Revised Selected Papers*, vol. 7762 of *Lecture Notes in Computer Science*, pp. 195–207, Springer, Berlin, Germany, 2013.

[7] M. J. Pérez-Jiménez and P. Sosík, "An optimal frontier of the efficiency of tissue P systems with cell separation," *Fundamenta Informaticae*, vol. 138, no. 1-2, pp. 45–60, 2015.

[8] P. Guo, J. Ji, H. Chen, and R. Liu, "Solving all-SAT problems by P systems," *Chinese Journal of Electronics*, vol. 24, no. 4, pp. 744–749, 2015.

[9] L. Pan and M. J. Pérez-Jiménez, "Computational complexity of tissue-like P systems," *Journal of Complexity*, vol. 26, no. 3, pp. 296–315, 2010.

[10] A. E. Porreca, G. Mauri, and C. Zandron, "Non-confluence in divisionless P systems with active membranes," *Theoretical Computer Science*, vol. 411, no. 6, pp. 878–887, 2010.

[11] D. Díaz-Pernil, M. A. Gutiérrez-Naranjo, M. J. Pérez-Jiménez, and A. Riscos-Núñez, "A uniform family of tissue P systems with cell division solving 3-COL in a linear time," *Theoretical Computer Science*, vol. 404, no. 1-2, pp. 76–87, 2008.

[12] P. Guo, Y. Dai, and H. Chen, "A P system for Hamiltonian cycle problem," *Optik*, vol. 127, no. 20, pp. 8461–8468, 2016.

[13] F. Bernardini and M. Gheorghe, "Cell communication in tissue P systems: universality results," *Soft Computing*, vol. 9, no. 9, pp. 640–649, 2005.

[14] B. Song, C. Zhang, and L. Pan, "Tissue-like P systems with evolutional symport/antiport rules," *Information Sciences*, vol. 378, pp. 177–193, 2017.

[15] B. Song, L. Pan, and M. J. Perez-Jimenez, "Cell-like P systems with channel states and symport/antiport rules," *IEEE Transactions on NanoBioscience*, vol. 15, no. 6, pp. 555–566, 2016.

[16] G. Păun, M. J. Pérez-Jiménez, and A. Riscos-Núñez, "Tissue P systems with cell division," *International Journal of Computers, Communications and Control*, vol. 3, no. 3, pp. 295–303, 2008.

[17] M. Cavaliere and D. Sburlan, "Time—independent P systems," in *Membrane Computing*, Lecture Notes in Computer Science, pp. 239–258, Springer, 2005.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] Y. Niu, Y. Jiang, and J. Xiao, "Time-free solution to 3-coloring problem using tissue P systems," *Chinese Journal of Electronics*, vol. 25, no. 3, pp. 407–412, 2016.

[23] X. Zeng, N. Ding, F. Xing, and X. Liu, "Time-free tissue P systems for solving the hamilton path problem," *Communications in Computer & Information Science*, vol. 472, pp. 562–565, 2014.