

Received May 30, 2020, accepted June 16, 2020, date of publication June 29, 2020, date of current version July 28, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3005459

Timed Homeostasis Tissue-Like P Systems With Evolutional Symport/Antiport Rules

YUEGUO LUO¹, PING GUO², (Member, IEEE), YUN JIANG³, AND YING ZHANG¹

¹College of Big Data and Intelligent Engineering, Yangtze Normal University, Chongqing 408100, China

²College of Computer Science, Chongqing University, Chongqing 400044, China

³School of Artificial Intelligence, Chongqing Technology and Business University, Chongqing 400067, China

Corresponding author: Ping Guo (guoping@cqu.edu.cn)

This work was supported in part by the Natural Science Foundation of Chongqing China under Grant cstc2019jcyj-msxmX0385 and Grant cstc2018jcyjAX0057, in part by the Science and Technology Research Program of Chongqing Municipal Education Commission under Grant KJQN201801426 and Grant KJQN201800814, and in part by the Science and Technology Project of Fuling China under Grant FLKJ2018BBB3021.

ABSTRACT Tissue-like P systems are a type of distributed parallel computing models inspired by actual biological tissue. In this paper, we consider a new variant of tissue-like P systems, which is called tissue-like P systems with evolutional symport/antiport rules. Unlike traditional models of this type, in the new P systems, objects can change during transmission. In biology, an organism that is in “homeostasis” reduces its dependence on external conditions, thereby keeping it relatively constant and maintaining a relatively stable internal environment. In our work, we remove the assumption that the quantity of objects in the environment is infinite, reducing the influence of the environment on this system, so the environment no longer provides powerful energy for cells. Moreover, the time-free mode is introduced into this type of P systems, which makes the constructed systems more robust. We investigate the computational power and computational efficiency of the constructed system. Specifically, by simulating register machines, such a P system can generate any Turing computable set of numbers. In addition, we prove that this constructed system can efficiently solve the *SAT* problem. Although we restrict P systems and consider time-free manner, the results show that this system is not only Turing universal, but also can solve **NP**-complete problem.


INDEX TERMS Membrane computing, homeostasis, time-free, tissue P system, evolutional symport/antiport rules.

I. INTRODUCTION

Membrane computing is a new research area emerging rapidly in biocomputing, which is founded by Păun [1], an academician of the European Academy of Sciences. Due to his pioneering work, membrane systems can also be called P systems, which are highly parallel computing systems. Membrane computing is a cutting-edge research field involving computer science, mathematics and biology in recent years, and it has many outstanding advantages compared with traditional computing models. Theoretically, the computational efficiency of membrane systems can be higher than that of current electronic computers. P systems are mainly composed of three important ingredients: the membrane structure, objects in various regions, and rules [2]. At present, scholars mainly focus on three types of P systems: cell-like P systems [1], [3], tissue P systems [4],

and neural-like P systems [5]. Recently, some scholars have proposed many new models of these basic systems [6]–[12]. In terms of computing efficiency, many scholars have proved that some **NP**-hard problems can be solved by P systems and their variants with reasonable computing resources in polynomial time (even linear time), such as Hamiltonian cycle problem [13], the 3-coloring problem [14], [15], the vertex cover problem [16], the *SAT* problem [17]–[20], and have obtained some important results. In the application field of membrane computing, it is worth noting that there have been many practical applications [21]–[26].

In this paper, our work is based on tissue P systems, which are inspired by biological tissue. In Ref. [4], tissue P systems have proved to be Turing universal. In recent years, many literatures have studied the computational efficiency of this type of system [27], [28]. So far, scholars have proposed some variants of this model [29]–[31]. For these tissue P systems, objects can be transferred between the environment and the cells, or between the cells, but objects cannot

The associate editor coordinating the review of this manuscript and approving it for publication was Chua Chin Heng Matthew .

change during the process of moving from one region to another. In addition, an important feature is that endless objects in the environment can be used when rules are applied. Therefore, the environment provides powerful material and energy for cells.

Moreover, from the perspective of biochemical reactions, the execution time of each rule is uncertain. Based on the biological reality, in our work, we consider the time-free mode, thereby constructing P systems with better fault tolerance. Hence, it is important to study P systems in the time-free mode. Moreover, compared to a standard P system, the execution time of rules in such a system is uncontrollable, which causes the system to be more complicated during running in the time-free mode. Therefore, in this paper, we remove the condition that the execution time of each rule is a unit time, and study the computational power and computational efficiency of this type of system in the time-free mode. In Ref. [32], the concept of time-free is proposed. In Ref. [33], the semi-uniform solution of the SAT has obtained for the first time. Recently, some NP-hard problems have been solved by various P systems in the time-free mode [34]–[36]. It should be noted that if tissue-like P systems work in a time-free manner, we call them timed tissue-like P systems.

In our work, for rules in P systems, we consider the new model proposed by Song [37], that is, objects can be changed during the process of moving from one region to another, which is completely different from the traditional models of tissue-like P systems and it is an interesting model that deserves further study. However, in this model, the objects can not only move from one region to another, but also can be evolved into new multisets, which leads to this model being too powerful. For this reason, we will restrict system running mode and influence of the environment. In biology, substances can be exchanged between the environment and cells. However, when this communication reaches a certain level, cells no longer depend on the external environment as much as possible, keeping it relatively constant and maintaining a relatively stable internal environment. Inspired by the biological reality, in our work, we introduce the concept of “homeostasis” [38] into this type of tissue P systems, that is, there are only a limited quantity of objects in the environment, rather than an infinite number. Therefore, we eliminate the assumption that objects in the environment is infinite, thereby constructing a type of new model that can simulate the biological homeostasis mechanism. Moreover, we consider that P systems run in the time-free mode, which makes the constructed systems more robust.

The first contribution of our paper, a novel type of variant tissue P systems, named timed homeostasis tissue-like P systems (*THTP systems*, for short), are introduced. In this system, we limit the quantity of objects in the environment. Hence, the environment generates weak energy by the cells themselves, so it can simulate the computing process of biological “homeostasis” state. The second contribution of our paper, we consider that this system runs in the time-free mode, which make the constructed systems more robust,

and expand the scope of applying this type of model. The third contribution of our paper is that we investigate the computational power and computational efficiency of the constructed system, and obtain some valuable results.

The structure of this paper is as follows. First, the basic knowledge involved in this paper is briefly reviewed, and then the model of *THTP systems* is put forward in section 3. In section 4, universality of *THTP systems* is proved. Based on the model, the classical SAT problem is solved, and an instance is demonstrated in section 5. Finally, conclusions are put forward, and several open problems are proposed.

II. FOUNDATIONS

A. FORMAL LANGUAGE THEORY

A finite non-empty set of symbols is called an alphabet V , and a sequence of symbols with finite length from V is a string. For a string x , its length can be denoted by $|x|$, which is the quantity of symbols. The empty string (denoted by λ) does not contain a symbol. All strings from V are denoted by V^* , and V^+ represents the set of non-empty strings ($V^* - \lambda$). For two arbitrary strings x and w , the string composed of x and w is called their concatenation, denoted by xw . If $V = a$, then $\{a^*\}$, $\{a^+\}$ can be abbreviated as a^* , a^+ respectively.

A multiset P over an alphabet V is a map $P : V \rightarrow \mathbb{N}$, where \mathbb{N} represents a set of natural numbers. If $a \in V$, $P(a)$ represents the multiplicity of a in P . For a finite set V , $V = \{a_1, \dots, a_n\}$, the multiset is $\{(a_1, P(a_1)), \dots, (a_n, P(a_n))\}$. For a multiset M of finite support, $\{(a_1, P(a_1)), \dots, (a_n, P(a_n))\}$ can be denoted as a string:

$$a_1^{P(a_1)} a_2^{P(a_2)} \dots a_n^{P(a_n)}$$

For more details on formal languages, please refer to Ref. [39].

B. REGISTER MACHINES

We can prove Turing universality to study the computational power of a system, and a general research approach is to simulate register machine or matrix grammar. It has been proved mathematically that as long as a system can simulate the computing process of a register machine, then this system can be proved to be Turing universal. In our work, we obtain the result by simulating register machine.

A register machine is a five tuple:

$$M = (m, H, l_0, l_h, I),$$

where,

m is the number of registers, H is instruction set of labels, $l_0, l_h \in H$, l_0 and l_h represent the initial label and halting label of instructions respectively, I represents the instruction set, and its specific definition is as follows:

$l_i : (ADD(r), l_j, l_k)$: this instruction adds 1 to the value in register r , and then execute instruction l_j or l_k non-deterministically.

$l_i : (SUB(r), l_j, l_k)$: unlike ADD instruction, a register machine will check the value of register r . If this value is non-zero, the register machine will subtract 1, and then the

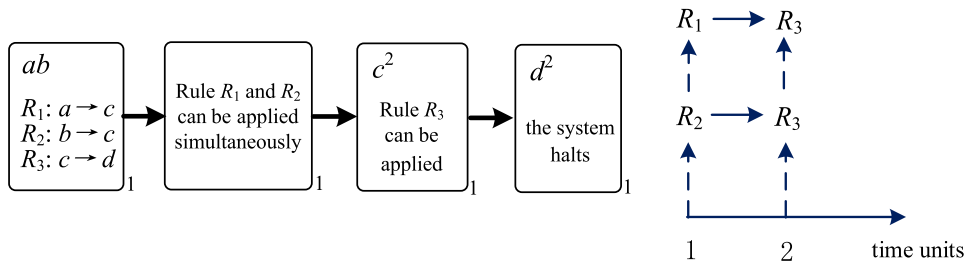


FIGURE 1. The execution process of rules in a standard P system.

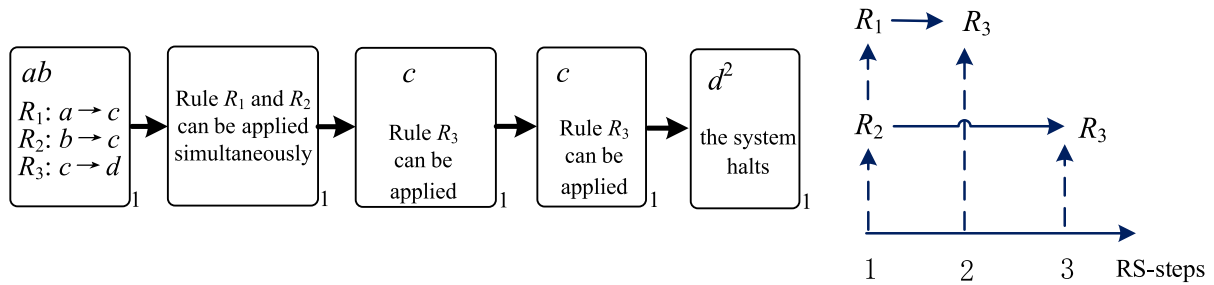


FIGURE 2. The execution process of rules in a timed P system.

instruction l_j is executed; if this value is zero, register machine will execute the instruction l_j ;

l_h : \mathcal{HALT} : the halt instruction.

Initially, the values stored in registers are all empty. Then, at the first step, the system executes the instruction l_0 (ADD instruction). At each subsequent step, the system executes one of the above three types of instructions. In this way, the contents of the register can be changed. Finally, the system stops running, and the value stored in the register 1 is called the number generated by the register machine.

The set of all numerical sequences generated by the register machine M is denoted by $N(M)$. It has been proven that the register machine can generate Turing computable numbers [40].

C. TIMED P SYSTEMS

If a P system works in the time-free mode, we call it timed P system. Timed P systems, compared with standard membrane systems, the execution time of each rule in a system is uncertain. For a rule R , $e(R)$ represents the execution time of this rule. If R is applied at the instant T , the rule will stop running at $T + e(R)$. During the execution process, objects using in this rule cannot be used by any other rules until $T + e(R) + 1$. If at least one rule starts to execute at one step, this step is called a *rule starting step* (RS-step [30], for short), which can characterize the computational efficiency of membrane systems.

Timed P systems and standard membrane systems have completely different running modes. In what follows, we will use a simple example to illustrate them with cell-like P systems. As shown in Fig. 1, there are a multiset ab , rules R_1 , R_2 and R_3 in membrane 1. For standard P systems, at the first computation step, the system executes the rules R_1 and R_2 at the same time to generate two copies of c . At the second

computation step, by executing rule R_3 , two copies of c are consumed, and two copies of d can be generated. In this way, the system will halt after two time units. Next, let's consider the time-free mode, the execution process of rules is shown in Fig. 2. When the system starts running, rules R_1 and R_2 are executed at the same time. However, the execution time of these two rules is uncertain. Here, it is assumed that the execution of rule R_1 ends first and generates a copy of c . At this time, by executing rule R_3 , object c is consumed and generates a copy of d . Similarly, when the execution of rule R_2 ends, the system executes rule R_3 again, and object c consumes and generates another copy of d at the same time. At this time, the system stops working after three RS-steps. It can be seen that timed P systems are generally more complicated than standard P systems.

III. TIMED HOMEOSTASIS TISSUE-LIKE P SYSTEMS WITH EVOLUTIONAL SYMPORT/ANTIPOINT RULES

In biology, a tissue (also called biological tissue), is the cellular architecture between cells and organs, which is composed of many cells with similar structure and function, and it is the constituent element of organs. Specifically, the tissue includes plasma, lymph, tissue fluid and cells, where, the tissue fluid corresponds to the environment in our model. The actual biological structure of tissue is shown in Fig. 3. In biochemical reactions, an organism that is in "homeostasis" reduces its dependence on external conditions, thereby keeping it relatively constant and maintaining a relatively stable internal environment. Inspired by the biological reality, we construct the biological model shown in Fig. 4. In this model, we introduced the concept of "homeostasis" into tissue P system, eliminating the assumption that the number of objects in the environment is infinite. In the system, there is very little material exchange between cells and

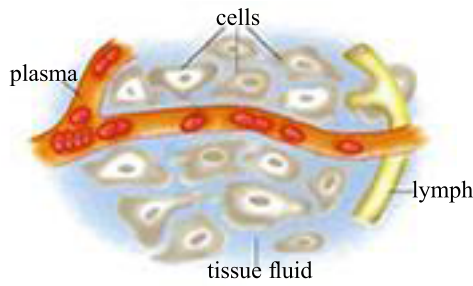


FIGURE 3. The actual biological structure of tissue.

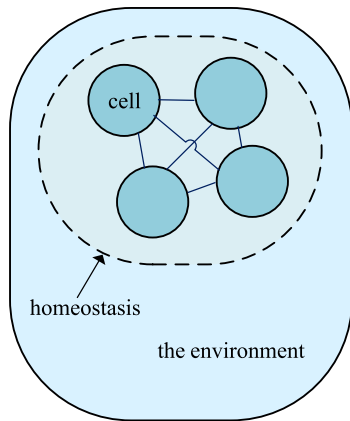


FIGURE 4. Homeostasis tissue structure model.

the environment, so the system greatly reduces the dependence on the environment, thereby simulating the computing process of biological “homeostasis” state.

Moreover, as the key element of membrane systems, rules play a vital role in the running of a system. In this paper, we remove the idealized assumption that the execution time of each rule is a unit time. Hence, the time-free method is considered during the execution of rules, that is, the execution time of any rule can be arbitrary. Obviously, even if such a system has some errors during system running, for example, some rules are executed incorrectly and lead to long-term execution, but for our model, which has no influence on the final execution result. Therefore, such a system obviously has better performance in terms of fault tolerance, thereby constructing a robust membrane system that can overcome the rule execution time.

In Ref. [37], a new model of tissue P systems is proposed by Song, which is based on evolutionary symport/antiport rules. For the model, objects can be changed during the process of moving from one region to another, which is completely different from the traditional models of tissue-like P systems. When apply such a rule of this type of system, a multiset of objects can not only change its location between two cells (or between a cell and the environment), but also can change the multiset itself to other multisets during the transfer process. That is to say, when evolutionary symport rules are applied, objects will go to another region and can be changed to other multisets (may also be empty); when evolutionary

antiport rules are applied, two objects in different region will move to the region where the other object is located, and these two objects can be changed to other multisets (may also be empty).

In our work, based on the new model, a type of novel variant tissue P systems are presented, called timed homeostasis tissue-like P systems with evolutionary symport/antiport rules. For this model, we limit the quantity of objects in the environment, and remove the assumption that the quantity of objects in the environment is infinite. Hence, the environment no longer provides powerful energy for cells, so the cell and the external environment (i.e. the environment) maintain relative independence, thereby simulating the computing process of biological “homeostasis” state. Moreover, we consider the time-free mode, which is introduced into this type of P systems, thereby constructing a more robust computing system.

Definition 1: Formally, a THTP system can be defined as follows:

$$\Pi = (V, w_0, w_1, \dots, w_m, R, e, i_{out}),$$

where

- (1) V is the set of objects;
- (2) $w_0 (w_0 \subseteq V)$, whose quantity is finite, represents objects in the environment;
- (3) $w_i, 1 \leq i \leq m$, is a finite multiset contained in V in the corresponding region;
- (4) e represents the execution time of rules;
- (5) i_{out} is output region;
- (6) R represents rules, which can be expressed as follows.

A. EVOLUTIONAL SYMPORT RULES

$$[u]_i []_j \rightarrow []_i [u']_j,$$

where, $i, j \in \{0, 1, \dots, m\}, i \neq j, u \in V^+, u' \in V^*$.

When evolutionary symport rules are applied, objects have one direction (from region i to region j). If a multiset u appears in regions i (may be a cell or the environment), this type of rule will start to execute, and the multiset u will move along the corresponding direction and be evolved to new multiset in target region. The execution process of evolutionary symport rules is shown in Fig. 5, where, for the case of “from the environment to a cell”, the quantity of u in the environment is placed beforehand, which is finite in this region. This model makes cells less dependent on the environment, thereby keeping cells in “homeostasis” state. Unlike the original model of tissue P system, the quantity of each object existing in the environment is assumed to be finite in the new model. The length of such a rule can be represented by $|u| + |u'|$;

B. EVOLUTIONAL ANTIPORT RULES

$$[u]_i [w]_j \rightarrow [w']_i [u']_j,$$

where, $i, j \in \{0, 1, \dots, m\}, i \neq j, u, v \in V^+, u', w' \in V^*$.

When evolutionary antiport rules are applied, objects have two directions: one is from region i to region j , and the other is in the opposite direction. If multisets u and w appear in regions i and j respectively, this type of rule will start to execute, and then these two multisets (u and w) will move

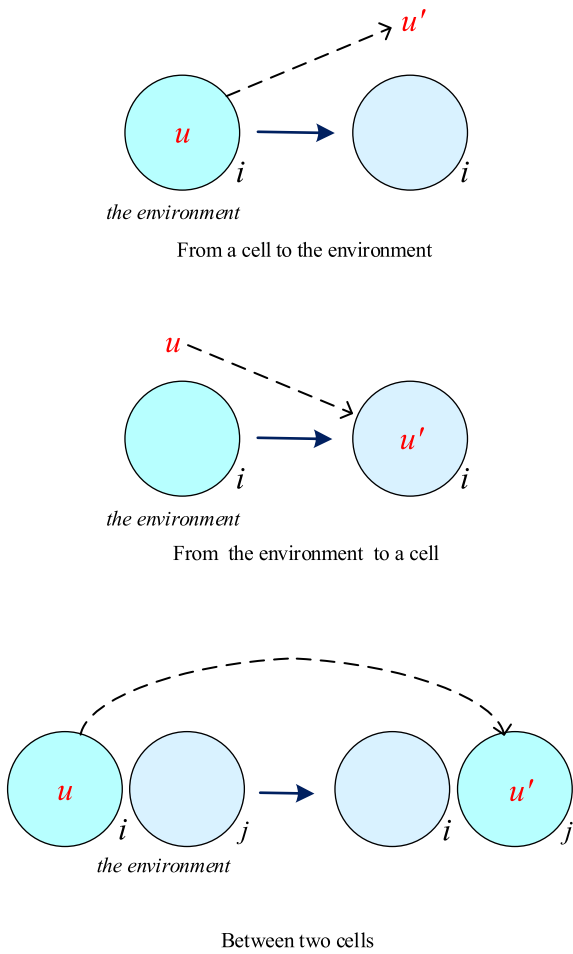


FIGURE 5. The execution process of evolutionary symport rules.

along corresponding directions and be evolved to new multisets in target regions. The execution process of evolutionary antiport rules is shown in Fig. 6. In this case, the quantity of corresponding objects in the environment are placed beforehand, which is finite in this region, thereby keeping cells in “homeostasis” state. The length of this type rule is denoted by $|u| + |w| + |u'| + |w'|$;

C. DIVISION RULES

$[a]_i \rightarrow [b]_i[c]_i$,
 where, $i \in \{1, 2, \dots, m\}$, $a, b, c \in V$, $i \neq i_{out}$.

When an object a exists in a cell i , the system will apply a division rule to divide the cell into two cells with the same label, and generate object b and object c in the two new cells, respectively, and other objects in the cell are copied into the two newly generated cells. It should be emphasized that the objects a, b , and c are all single symbols. The execution process of such a division rule is shown in Fig. 7.

A *THTP system* can be denoted by $\Pi(e)$, where Π is a *THTP system*, and e is the execution time of the rules. We describe the membrane system by using configuration. The current configuration is denoted with cells and multisets contained in cells, namely, (w_1, \dots, w_m) . Initially, rules have not yet been applied. At this moment,

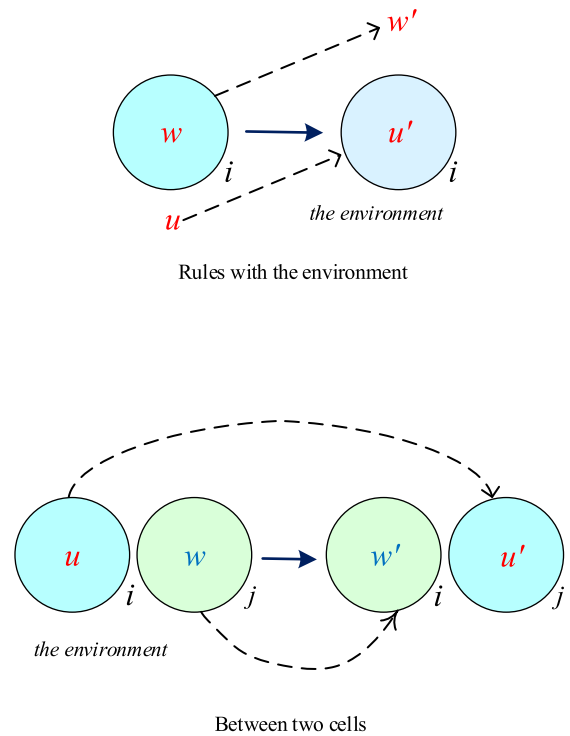


FIGURE 6. The execution process of evolutionary antiport rules.

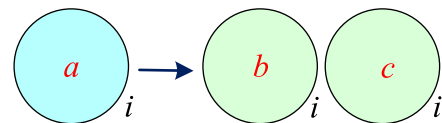


FIGURE 7. The execution process of division rules.

the current configuration is called the *initial configuration*. Next, *transitions* can be generated by applying rules in the *THTP system*. A *computation* is obtained by a sequence of transitions. If there are finitely many transitions, the system will eventually stop running and generate a computing result in the output area, which can be called *halting computation* at that moment; if the sequence is infinite, the system does not stop. A timed system works as follows: suppose the global clock starts from time 0, which divides the entire execution time into equal-length time units. During *THTP systems* running, all rules are applied with maximal parallelism: at an arbitrary moment, in each region, the system has to use a maximal multiset of rules. For these rules, they can be applied many times simultaneously. If a rule R from type (a) to type (c) starts to be applied at instant T , then this rule will halt at instant $T + e(R) + 1$. During the execution process, objects applied by the rule cannot be used by other rules until the execution of the rule halts. For the cell used by the rule, it can be applied by other rules at the same time. Moreover, rules (including objects) are chosen non-deterministically [1].

The advantage of timed membrane systems is that they have strong fault tolerance. Even though the application of a rule does not finish within predetermined time, the entire system can still obtain the correct computing result.

Therefore, the time-free mode enhances the fault tolerance of *THTP systems*.

Definition 2: In the time-free mode, a recognizer *THTP system* can be defined as follows:

$$\Pi = (V, \Sigma, w_0, w_1, \dots, w_m, R, e, i_{in}, i_{out}),$$

where, Σ is an input alphabet from set V , i_{in} is the input cell. The other symbols are similar to Definition 1.

We define a recognizer *THTP system* to solve decision problems. In this way, the purpose is to investigate the computational efficiency of a *THTP system*.

The initial configuration can be denoted by $(w_1, \dots, (w_{in} + w), \dots, w_m)$, where, w_{in} represents input cell, and w represents an input multiset. When the system is running, transitions can be generated by applying rules. For such system, transitions are finite. Hence, in the end, a recognizer *THTP system* has to halt after certain RS-steps, and generates a computing result in the output area. At that time, if \mathcal{YES} appears in the output area, we call it *accepting computation*; if \mathcal{NO} appears in the output area, we call it *rejecting computation*.

Definition 3: Suppose $X = (I_X, \theta_X)$ is a decision problem, I_X represents instances of the decision problem, and θ_X represents a predicate over these instances. In the time-free mode, a recognizer *THTP system* Π can solve a decision problem in polynomial RS-steps and obtain uniform solution of this problem, if the following holds:

(1) the family Π is polynomially uniform by Turing machines; that is, for a system $\Pi(n)$, a deterministic Turing machine can work in polynomial time;

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

- u is an instance, $u \in I_X$, $s(u)$ is a natural number and $cod(u)$ is an input multiset of the system $\Pi(s(n))$.
- the family Π is time-free sound with regard to (X, cod, s) . For $u \in I_X$ of a problem, the system has an accepting computation with $cod(u)$.
- the family Π is time-free complete with regard to (X, cod, s) . For $u \in I_X$ of a problem, in the time-free mode, computations of $\Pi(s(u), e)$ with $cod(u)$ can be an accepting one.
- the family Π is time-free polynomially bounded with regard to (X, cod, s) , that is, for any time-mapping e , there is a polynomial function $p(n)$ such that for each $u \in I_X$, when the system stops running, it takes at most $p(|u|)$ RS-steps.

Definition 4: $PMC_{THTP(k)}^f$ represents that a recognizer *THTP system* can solve a decision problem and obtain the uniform solution in polynomial time in time-free manner (the maximal length rules is k); $NOP_{m,n}^f(r)$ represents that natural numbers can be generated by the P system with cells no more than m , rule length no more than n , where, f represents time-free manner, and r represents rule type.

Turing machines have outstanding computational performance and can solve any problems computed by algorithms. When we research membrane computing models, after a type

of model is defined, the general research approach is to investigate its computational power by comparing it with Turing machines or other classic computing models. In this way, we can know how powerful the computational power of such models is. In addition, an important approach is to research the computational efficiency of such models. Specifically, the general research approach is to solve NP-hard problems using finite computing resources in feasible time. Therefore, for the theoretical research of membrane computing, the main approach is to investigate the computational power and computational efficiency of a system.

IV. UNIVERSALITY OF THTP SYSTEMS

Theorem 1: $NOP_{1,4}^f((a), (b)) = NRE$.

Proof: $M = (m, H, l_0, l_h, I)$ is a register machine with m registers. Next, we construct a system to simulate the register machine.

$$\Pi = (V, w_0, w_1, R, e, i_{out}),$$

where,

- $V = \{a_r | 1 \leq r \leq m\} \cup \{l, l', l^{(2)}, l^{(3)}, l^{(4)}, l^{(5)}, l^{(6)} | l \in H\}$;
- $w_0 = \{l^{(3)} | l \in H\}$;
- $w_1 = \lambda$;
- e is rule execution time;
- $i_{out} = 1$.

(1) Rules of the ADD instruction $l_i : (ADD(r), l_j, l_k)$:

$$R_1 : []_0 [l_i]_1 \rightarrow [l'_i]_0 []_1.$$

$$R_2 : [l'_i]_0 []_1 \rightarrow []_0 [a_r l_j]_1.$$

$$R_3 : [l'_i]_0 []_1 \rightarrow []_0 [a_r l_k]_1.$$

First, the system applies rule R_1 to simulate ADD instruction l_i . Next step, the value of register r will increase by 1 with using rule R_2 or R_3 . In addition, the generated l_j or l_k will simulate next instruction. Obviously, the results of ADD instruction l_i are correct in the time-free mode.

(2) Rules of the SUB instruction $l_i : (SUB(r), l_j, l_k)$:

$$R_4 : [l'_i]_0 [l_i]_1 \rightarrow [l'_i]_0 [l'_i]_1.$$

$$R_5 : [l_i^{(2)}]_0 [a_r]_1 \rightarrow []_0 [l_i^{(2)}]_1.$$

$$R_6 : [l_i^{(3)}]_0 [l'_i]_1 \rightarrow [l_i^{(4)}]_0 [l_i^{(3)}]_1.$$

$$R_7 : [l_i^{(4)}]_0 [l_i^{(2)}]_1 \rightarrow [l_i^{(5)}]_0 []_1.$$

$$R_8 : [l_i^{(2)}]_0 [l_i^{(3)}]_1 \rightarrow [l_i^{(6)}]_0 [l_i^{(3)}]_1.$$

$$R_9 : [l_i^{(5)}]_0 [l_i^{(3)}]_1 \rightarrow [l'_i]_0 [l_j]_1.$$

$$R_{10} : [l_i^{(6)}]_0 []_1 \rightarrow []_0 [l_i^{(6)}]_1.$$

$$R_{11} : [l_i^{(4)}]_0 [l_i^{(6)}]_1 \rightarrow [l'_i]_0 [l_k]_1.$$

The rules constructed above are correct even when running in the time-free mode. First, the system apply rule R_4 to simulate SUB instruction l_i . Next step, there may be two cases in cell 1:

- the object a_r appears

Because there are objects $l_i^{(2)}$ and l'_i , rules R_5 and R_6 will start executing simultaneously. Then, rule R_7 is executed. Hence, there will be in the environment and in cell 1. After these objects are generated, R_9 can be applied, and the value of register r will decrease by 1, thereby simulating the SUB instruction correctly. In this case, the application of rules is listed in Table.1,

TABLE 1. Process when there is an object a_r .

RS-steps	the application of rules	objects in the environment	objects in cell 1
1	$R_4 : [l'_i]_0[l_i]_1 \rightarrow [l_i^{(2)}]_0[l'_i]_1$	$l_i^{(2)} l_i^{(3)}$	l'_i
2	$R_5 : [l_i^{(2)}]_0[a_r]_1 \rightarrow []_0[l_i^{(2)}]_1$ $R_6 : [l_i^{(3)}]_0[l'_i]_1 \rightarrow [l_i^{(4)}]_0[l_i^{(3)}]_1$	$l_i^{(4)}$	$l_i^{(2)} l_i^{(3)}$
3	$R_7 : [l_i^{(4)}]_0[l_i^{(2)}]_1 \rightarrow [l_i^{(5)}]_0[]_1$	$l_i^{(5)}$	$l_i^{(3)}$
4	$R_9 : [l_i^{(5)}]_0[l_i^{(3)}]_1 \rightarrow [l'_i]_0[l_j]_1$	l'_i	l_j

TABLE 2. Process when no object a_r appears.

RS-steps	applying rules	objects in the environment	objects in cell 1
1	$R_4 : [l'_i]_0[l_i]_1 \rightarrow [l_i^{(2)}]_0[l'_i]_1$	$l_i^{(2)} l_i^{(3)}$	l'_i
2	$R_6 : [l_i^{(3)}]_0[l'_i]_1 \rightarrow [l_i^{(4)}]_0[l_i^{(3)}]_1$	$l_i^{(2)} l_i^{(4)}$	$l_i^{(3)}$
3	$R_8 : [l_i^{(2)}]_0[l_i^{(3)}]_1 \rightarrow [l_i^{(6)}]_0[l_i^{(3)}]_1$	$l_i^{(4)} l_i^{(6)}$	$l_i^{(3)}$
4	$R_{10} : [l_i^{(6)}]_0[]_1 \rightarrow []_0[l_i^{(6)}]_1$	$l_i^{(4)}$	$l_i^{(3)} l_i^{(6)}$
5	$R_{11} : [l_i^{(4)}]_0[l_i^{(6)}]_1 \rightarrow [l'_i]_0[l_k]_1$	l'_i	$l_i^{(3)} l_k$

which can be applied in turn. Main objects in the environment and objects in cell 1 are listed in the table, but some objects have been omitted.

- no object a_r exists

In this case, rule R_5 do not be applied, and rule R_6 can be executed. Next, rule R_8 , R_{10} and R_{11} will be applied in turn. Finally, the object l_k appears, indicating *THTP system* is ready to simulate instruction l_j . In this case, the application of rules is listed in Table.2, which can be applied in turn.

In the end, the system halts, and object l_h will appear in cell 1. It can be verified that the final results of the system have nothing to do with rules execution time.

V. A UNIFORM SOLUTION TO SAT BY THTP SYSTEMS

Applying cell division rules can generate exponential computing space in feasible time, so time can be obtained by space. In addition, a parallel computing system is designed based on *THTP systems*, so that maximal parallelism can be used, thereby greatly improving the computational efficiency of *THTP systems*. Therefore, theoretically, *THTP systems* can solve NP-hard problems in polynomial time (or even linear time).

A. THEOREM PROVING

Theorem 2: The SAT problem can be solved by a uniform family of *THTP systems* in polynomial time.

Proof: For a SAT formula with n Boolean variables and m clauses, the formula consists of m clauses:

$$C_j = y_{1,j} \vee \dots \vee y_{p_j,j},$$

where $y_{i,j} \in \{x_l, \neg x_l | 1 \leq l \leq n\}$, $1 \leq i \leq p_j$, $1 \leq j \leq m$; $\neg x_l$ is the negation of a propositional variable x_l .

Next, to solve the SAT, we will construct a system denoted by $\Pi_{SAT(m,n)}(e)$. Algorithm 1 gives the computing process of $\Pi_{SAT(m,n)}(e)$.

For a given instance γ , it can be encoded as follows:

$$cod(\gamma) = \alpha_{1,1} \dots \alpha_{n,1} \alpha_{1,2} \dots \alpha_{n,2} \dots \alpha_{1,m} \dots \alpha_{n,m},$$

where,

$$\alpha_{i,j} = \begin{cases} Y_{i,j} & \text{if } x_i \text{ appears in } C_j; \\ N_{i,j} & \text{if } \neg x_i \text{ appears in } C_j; \\ B_{i,j} & \text{both } x_i \text{ and } \neg x_i \text{ do not appear in } C_j. \end{cases}$$

Next, we construct the system

$$\Pi_{SAT(m,n)}(e) = (V, \Sigma, w_0, w_1, w_2, R, e, i_{in}, i_{out}),$$

where,

- The set of objects can be denoted as follows:

$$V = \Sigma \cup \{a_i | 1 \leq i \leq n + 1\} \cup \{t_{i,j}, f_{i,j} | 1 \leq i \leq n, 1 \leq j \leq m + 1\} \cup \{e_i | 2 \leq i \leq n + 1\} \\ \cup \{G_{i,j}, T_i, F_i, r_j | 1 \leq i \leq n, 1 \leq j \leq m\} \\ \cup \{s_j | 2 \leq j \leq m + 1\} \cup \{d, p, q, \text{yes}, \text{no}\};$$

- $\Sigma = \{Y_{i,j}, N_{i,j}, B_{i,j} | 1 \leq i \leq n, 1 \leq j \leq m\}$
- w_0 is objects in the environment, $w_i (1 \leq i \leq m)$ is multisets in cell i over V ,

$$w_0 = \{\mathcal{NO}, q^2\}, w_1 = p^2, w_2 = \{a_1, d\};$$

- $i_{in} = 1, i_{out} = 0$;
- e is rule execution time;
- R represents the rules.

The computing process consists of the following phases:

Algorithm 1 The Computing Process of $\Pi_{SAT(m,n)}(e)$ Input: an instance of the SAT Output: \mathcal{YES} , or \mathcal{NO} **Begin****Initialization:** input the instance of in cell 1 as an input multiset, and input the initialization object in the corresponding region;**for** ($i = 1, i \leq n, i++$)

{

By applying division rules, each cell 2 can be divided into two new cells. /* cell 2 is used for division rules */

for ($j = 1, j \leq m+1, j++$)

{

(1) Check whether each clause of current variable assignment (*true* and *false*) is satisfied in each cell 2;

(2) Synchronize rules in cell 1 so that all the previous rules have been executed;

(3) An object is generated in each cell 2 for applying the next cell division. In addition, objects for the evolutionary antiport rules of the next iteration are generated.

}

}

if all the elements in the set $\{r_1, r_2, \dots, r_m\}$ appear in a cell with label 2 **then**the object \mathcal{YES} appears in the environment;**else**the object \mathcal{NO} appears in the environment;**End**

1) GENERATION PHASE

$$R_{1,i} : [a_i]_2 \rightarrow [t_{i,1}]_2 [f_{i,1}]_2, \quad 1 \leq i \leq n.$$

$$R_{2,i,j} : [p]_1 [t_{i,j} Y_{i,j}]_2 \rightarrow [p]_1 [G_{i,j}]_2, \\ 1 \leq i \leq n, 1 \leq j \leq m.$$

$$R_{3,i,j} : [p]_1 [G_{i,j}]_2 \rightarrow [p]_1 [r_j t_{i,j+1}]_2, \\ 1 \leq i \leq n, 1 \leq j \leq m.$$

$$R_{4,i,j} : [p]_1 [t_{i,j} N_{i,j}]_2 \rightarrow [p]_1 [t_{i,j+1}]_2, \\ 1 \leq i \leq n, 1 \leq j \leq m.$$

$$R_{5,i,j} : [p]_1 [t_{i,j} B_{i,j}]_2 \rightarrow [p]_1 [t_{i,j+1}]_2, \\ 1 \leq i \leq n, 1 \leq j \leq m.$$

$$R_{6,i,j} : [p]_1 [f_{i,j} N_{i,j}]_2 \rightarrow [p]_1 [G_{i,j}]_2, \\ 1 \leq i \leq n, 1 \leq j \leq m.$$

$$R_{7,i,j} : [p]_1 [G_{i,j}]_2 \rightarrow [p]_1 [r_j t_{i,j+1}]_2, \\ 1 \leq i \leq n, 1 \leq j \leq m.$$

$$R_{8,i,j} : [p]_1 [f_{i,j} Y_{i,j}]_2 \rightarrow [p]_1 [t_{i,j+1}]_2, \\ 1 \leq i \leq n, 1 \leq j \leq m.$$

$$R_{9,i,j} : [p]_1 [f_{i,j} B_{i,j}]_2 \rightarrow [p]_1 [t_{i,j+1}]_2, \\ 1 \leq i \leq n, 1 \leq j \leq m.$$

$$R_{10,i} : [p]_1 [t_{i,m+1}]_2 \rightarrow [p^2 T_i]_1 []_2, \quad 1 \leq i \leq n.$$

$$R_{11,i} : [p]_1 [f_{i,m+1}]_2 \rightarrow [p^2 F_i]_1 []_2, \quad 1 \leq i \leq n.$$

$$R_{12,i} : [T_i F_i]_1 []_0 \rightarrow []_1 [q a_{i+1}^2]_0, \quad 1 \leq i \leq n.$$

$$R_{13} : [q]_0 []_1 \rightarrow []_0 [q^4]_1.$$

$$R_{14,i} : [a_i]_0 [d]_2 \rightarrow []_0 [d a_i]_2, \quad 2 \leq i \leq n+1.$$

For the object a_i , it corresponds to variable x_i in the SAT . At the first RS-step, rule $R_{1,1}$ will start to run. By applying this division rule, $t_{1,1}$ and $f_{1,1}$ can be generated in the two new cells, which represent *true* of variable x_1 and *false* of variable x_1 , respectively. Under the time-free mode, the computational efficiency of *THTP systems* is characterized by using RS-steps. Here, it is obvious that this computing process only takes one RS-step. At the next RS-step, rules from $R_{2,1,j}$ to $R_{9,1,j}$ can be applied selectively. If a multiset $t_{1,1} Y_{1,1}$ (resp., $f_{1,1} N_{1,1}$) appears in a cell 2, the evolutionary antiport rule $R_{2,1,1}$ (resp., $R_{6,1,1}$) is executed. Next RS-step, $R_{3,1,j}$ (resp., $R_{7,1,j}$) will be applied, and object r_1 is generated in cell 2, at this moment, we can conclude that current clause is satisfied by x_1 . In other cases, by applying the corresponding rules, the second subscript of $t_{1,j}$ (resp., $f_{1,j}$) continues to increase. Finally, its value will reach $m+1$. However, because the system runs in a time-free manner, the execution time corresponding to the assignments $t_{1,j}$ and $f_{1,j}$ cannot be predicted. Here, we design the following rules to synchronize the previous two computing processes, so that the computing process of previous assignments by variables $t_{1,j}$ and $f_{1,j}$ have finished. When the second subscript of $t_{1,j}$ (resp., $f_{1,j}$) reach $m+1$, rules from $R_{10,1}$ to $R_{14,2}$ will be executed in turn. These rules have two important functions. First, they can synchronize the previous rules. Since *THTP systems* run under the time-free mode, we cannot determine the execution time of each rule. Here, we adopt the approach of rule synchronization. Therefore, we need to complete all the previous rules, where, rule $R_{12,i}$ plays the synchronization function. In addition, relevant objects need to be generated for subsequent computing processes, such as object a_2 (for the next division rule), p and q .

At the end of the computing process of assignment corresponding to the variable x_1 , the system will generate a multiset da_2 in each cell 2, where, object d corresponds to the rule $R_{12,2}$ used in the next computing process; and the object a_2 can be used for rule $R_{1,2}$, which will be applied in the next computing process. The rule $R_{1,2}$ is a division rule, which corresponds to the following computing process of generation phase.

The subsequent computing process is similar to the variable x_1 , that is, the above computing process is executed in an iteration manner. When the computing process corresponding to n variables finishes, the generation phase ends. On the whole, after $3mn + 7n$ RS-steps, 2^n copies of cell 2 are generated.

2) CHECKING PHASE

$$R_{15} : []_1 [a_{n+1}]_2 \rightarrow [a_{n+1}]_1 []_2.$$

$$R_{16} : [a_{n+1}]_1 [r_1]_2 \rightarrow [s_2]_1 []_2.$$

$$R_{17,j} : [s_j]_1 [r_j]_2 \rightarrow [s_{j+1}]_1 []_2, \quad 2 \leq j \leq m.$$

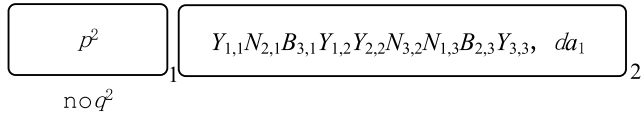


FIGURE 8. The initial configuration.

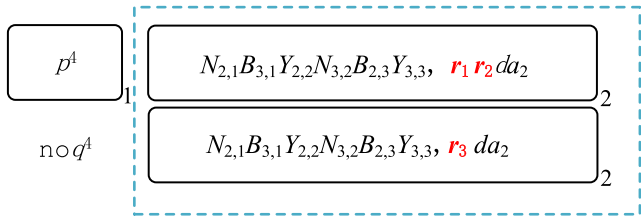


FIGURE 9. After the execution of the first iteration.

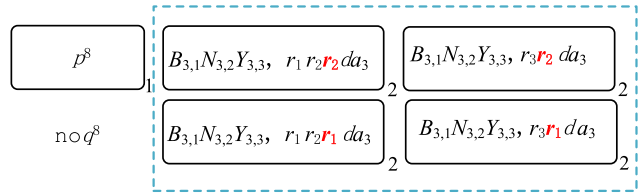


FIGURE 10. After the execution of the second iteration.

Once an object a_{n+1} appears in a cell with label 2, the generation phase has been executed. If all the elements in the set $\{r_1, r_2, \dots, r_m\}$ appear in a cell 2, we can conclude that the SAT is satisfiable. By applying the rules in the checking phase, the purpose is to check whether all the elements in the set $\{r_1, r_2, \dots, r_m\}$ exist in this cell. If all the elements exist, s_{m+1} will appear in the cell 1, corresponding to a satisfiable solution.

3) OUTPUT PHASE

$$R_{18} : [\mathcal{NO}]_0[s_{m+1}]_1 \rightarrow [\mathcal{YES}]_0[]_1.$$

Once an object s_{m+1} appears in cell 1, rule R_{18} can be applied. Finally, the system will stop running, and \mathcal{YES} generates in the output area, which shows γ is satisfiable. On the contrary, if object s_{m+1} does not appear, \mathcal{NO} remains in the output area in the end, which shows γ is not satisfiable.

B. COMPUTATIONAL EFFICIENCY

The computing resources of $\Pi_{SAT(m,n)}(e)$:

- size of the set O : $5mn + m + 3n + 6$;
- initial number of objects: 5;
- initial number of cells: 2;
- the total number of rules: $8mn + m + 4n + 5$;
- the maximal length of rules: 5.

In the time-free mode, the computational efficiency of $\Pi_{SAT(m,n)}(e)$ is characterized by RS-steps. The following is the detailed RS-steps of the system at each computing phase: the generation phase takes at most $3mn + 7n$ RS-steps. If the formula γ is satisfiable, the checking phase and the output phase will take at most m RS-steps and 2 RS-steps, respectively; otherwise checking phase and output phase will take at most $m - 1$ RS-steps and no RS-step, respectively.

Theorem 3: $SAT \in PMC_{THTP(5)}^f$.

The SAT is a classic computational intractable problem. With our constructed recognizer $THTP$ systems, it has proved that the maximal length of P systems is 5 in the time-free mode. According to the previous definition and proof process, this result can be obtained, and $\Pi_{SAT(m,n)}(e)$ is complete and polynomially bounded.

Corollary 1: $NP \cup co - NP \subseteq PMC_{THTP(5)}^f$

C. AN INSTANCE

By using of $\Pi_{SAT(m,n)}(e)$ constructed above, we will consider a specific instance as follows.

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

For the formula, $m = n = 3$. The initial configuration can be shown in Fig.8. At RS-step 1, by applying division rule, the number of cell 2 will increase. Next, by applying $R_{2,1,j}$ to $R_{9,1,j}$, $t_{1,4}$ (resp., $f_{1,4}$) will appear in the end. When $R_{14,1}$ is completed, object a_2 will appear in a cell with label 2, indicating that the first iteration of variable x_i has completed. At this moment, we can obtain the configuration (see Fig.9).

At the end of the execution of the first iteration of variable x_i , because there are objects a_2 in the environment and object d in cell 2, by applying evolutionary antiport rule $R_{14,2}$, each object a_2 in environment can enter each cell 2 at the same time. In this way, the appearance of the object a_2 will start to apply division rule of the next variable x_2 , thus starting to execute the computing process of the following iteration. Next, the variable x_2 and x_1 have the same computing process. When the computing process of x_2 is completed, Fig.10 shows the current configuration at that moment. In this way, when the computing process of x_3 is completed, Fig.11 shows the current configuration. At that moment, generation phase finishes.

Once an object a_4 appears in a cell 2, the system starts to execute checking phase. By applying rules of this phase, s_4 will be generated, and Fig.12 shows the current configuration at that moment.

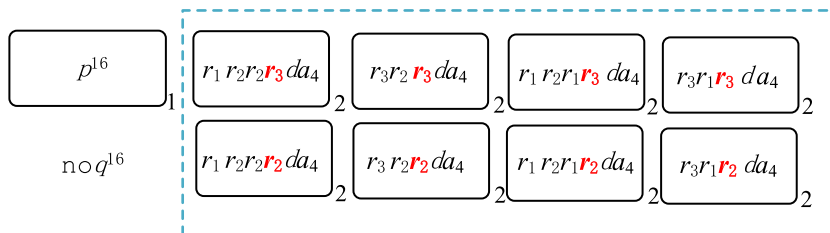


FIGURE 11. After the execution of the third iteration.

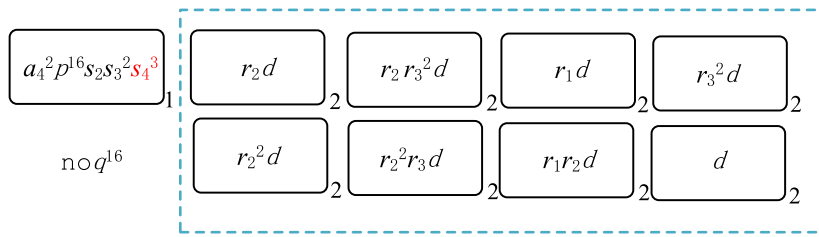


FIGURE 12. After the execution of the checking phase.

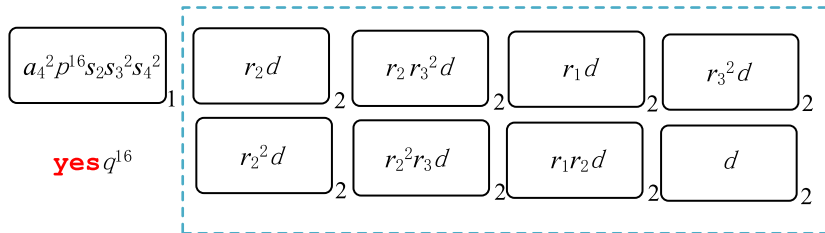


FIGURE 13. The final configuration.

In the end, $\Pi_{SAT(m,n)}(e)$ has to halt after certain RS-steps, and generates a computing result in the output area. Once an object s_4 appears in cell 1, by applying the rule of output phase, \mathcal{YES} will be generated in the environment, so we can conclude that it is an accepting computation and the instance is satisfiable (see Fig. 13).

VI. CONCLUSIONS

In our work, inspired by the biological reality of homeostasis, we introduce “homeostasis” into tissue-like P systems with evolutional symport/antiport rules, so the environment no longer provides powerful energy for cells. We restricted the model to be too powerful, and we think that this model will better reflect “homeostasis”. In addition, we have introduced the time-free mode into such a model, thereby constructing a more robust computing system, so our work expand the scope of applying this type of model. Compared with the standard mode, the time-free mode is a more complicated manner. However, we still obtain some interesting results: although running in a time-free manner, this system is not only Turing universal, but also can solve NP-complete problem.

Finally, we look forward to future work. The readers can use the P systems constructed in this paper to solve other NP-complete problems, such as the 3-coloring problem, subset sum problem, etc. In particular, it is worthy of further study that the computational efficiency of solving the PSPACE-complete problems.

In this paper, we apply division rules. From a biological point of view, it deserves to introduce cell separation [20] to *THTP systems* and investigate the computational efficiency of such a system. Moreover, it deserves study whether the concept of homeostasis can be extended to other membrane systems, such as spiking neural P systems.

In addition, under the model framework of this paper, the readers can investigate the computational power and computational efficiency running in various modes, such as the minimal parallel mode [41], the flat maximal parallelism mode [42], the local synchronization mode [43], etc.

REFERENCES

- [1] G. Păun, “Computing with membranes,” *J. Comput. Syst. Sci.*, vol. 61, no. 1, pp. 108–143, Aug. 2000.
- [2] G. Păun, “A quick introduction to membrane computing,” *J. Log. Algebr. Program.*, vol. 79, no. 6, pp. 291–294, Aug. 2010.
- [3] D. Díaz-Pernil, M. J. Pérez-Jiménez, and Á. Romero-Jiménez, “Efficient simulation of tissue-like P systems by transition cell-like P systems,” *Natural Comput.*, vol. 8, no. 4, pp. 797–806, Dec. 2009.
- [4] C. Martín-Vide, G. Păun, J. Pazos, and A. Rodríguez-Patón, “Tissue P systems,” *Theor. Comput. Sci.*, vol. 296, no. 2, pp. 295–326, 2003.
- [5] M. Ionescu, G. Păun, and T. Yokomori, “Spiking neural P systems,” *Fundam. Inf.*, vol. 71, no. 2, pp. 279–308, 2006.
- [6] Y. Luo, H. Tan, Y. Zhang, and Y. Jiang, “The computational power of timed P systems with active membranes using promoters,” *Math. Struct. Comput. Sci.*, vol. 29, no. 5, pp. 663–680, May 2019.
- [7] H. Peng and J. Wang, “Coupled neural P systems,” *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 6, pp. 1672–1682, Jun. 2019.
- [8] M. García-Quismondo, M. Levin, and D. Lobo, “Modeling regenerative processes with membrane computing,” *Inf. Sci.*, vol. 381, pp. 229–249, Mar. 2017.
- [9] A. Leporati, L. Manzoni, G. Mauri, A. E. Porreca, and C. Zandron, “Monodirectional P systems,” *Natural Comput.*, vol. 15, no. 4, pp. 551–564, Dec. 2016.
- [10] B. Song, L. Pan, and M. J. Pérez-Jiménez, “Cell-like P systems with channel states and symport/antiport rules,” *IEEE Trans. Nanobiosci.*, vol. 15, no. 6, pp. 555–566, 2016.
- [11] X. Zhang, L. Pan, and A. Paun, “On the universality of axon P systems,” *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 11, pp. 2816–2829, Nov. 2015.
- [12] Z. Gazdag and G. Kolonits, “A new method to simulate restricted variants of polarizationless P systems with active membranes,” *J. Membrane Comput.*, vol. 1, no. 4, pp. 251–261, Dec. 2019, doi: [10.1007/s41965-019-00024-z](https://doi.org/10.1007/s41965-019-00024-z).
- [13] P. Guo, Y. Dai, and H. Chen, “A P system for Hamiltonian cycle problem,” *Optik*, vol. 127, no. 20, pp. 8461–8468, Oct. 2016.
- [14] D. Díaz-Pernil, H. A. Christinal, and M. A. Gutiérrez-Naranjo, “Solving the 3-COL problem by using tissue P systems without environment and proteins on cells,” *Inf. Sci.*, vols. 430–431, pp. 240–246, Mar. 2018.
- [15] J. Cooper and R. Nicolescu, “Alternative representations of P systems solutions to the graph colouring problem,” *J. Membrane Comput.*, vol. 1, no. 2, pp. 112–126, Jun. 2019, doi: [10.1007/s41965-019-00013-2](https://doi.org/10.1007/s41965-019-00013-2).
- [16] P. Guo, C. Quan, and H. Chen, “MEAMVC: A membrane evolutionary algorithm for solving minimum vertex cover problem,” *IEEE Access*, vol. 7, pp. 60774–60784, 2019.
- [17] Y. Luo, Z. Xiong, Y. Chen, and H. Tan, “Efficient solutions to all-SAT by P systems with active membranes,” *J. Comput. Theor. Nanosci.*, vol. 13, no. 6, pp. 3887–3894, Jun. 2016.

- [18] P. Guo, J. Zhu, H. Chen, and R. Yang, "A linear-time solution for all-SAT problem based on P system," *Chin. J. Electron.*, vol. 27, no. 2, pp. 367–373, Mar. 2018.
- [19] S. Jiang, Y. Wang, and Y. Su, "A uniform solution to SAT problem by symport/antiport P systems with channel states and membrane division," *Soft Comput.*, vol. 23, no. 12, pp. 3903–3911, Jun. 2019.
- [20] M. J. Pérez-Jiménez and P. Sosík, "An optimal frontier of the efficiency of tissue P systems with cell separation," *Fundamenta Informaticae*, vol. 138, nos. 1–2, pp. 45–60, 2015.
- [21] G. Singh and K. Deep, "Effectiveness of new multiple-PSO based membrane optimization algorithms on CEC 2014 benchmarks and iris classification," *Natural Comput.*, vol. 16, no. 3, pp. 473–496, Sep. 2017.
- [22] G. Zhang, J. Cheng, and M. Gheorghe, "Dynamic behavior analysis of membrane-inspired evolutionary algorithms," *Int. J. Comput. Commun. Control*, vol. 9, no. 2, pp. 227–242, Apr. 2014.
- [23] P. Guo, C. Quan, and L. Ye, "UPSimulator: A general P system simulator," *Knowl.-Based Syst.*, vol. 170, pp. 20–25, Apr. 2019.
- [24] G. Zhang, M. J. Pérez-Jiménez, G. Mauri, A. E. Porreca, and C. Zandron, "Characterising the complexity of tissue P systems with fission rules," *J. Comput. Syst. Sci.*, vol. 90, pp. 115–128, Dec. 2017.
- [25] 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," *Theor. Comput. Sci.*, vol. 404, nos. 1–2, pp. 76–87, Sep. 2008.
- [26] R. Freund, G. Păun, and M. J. Pérez-Jiménez, "Tissue P systems with channel states," *Theor. Comput. Sci.*, vol. 330, no. 1, pp. 101–116, Jan. 2005.
- [27] B. Song, L. Pan, and M. J. Pérez-Jiménez, "Tissue P systems with protein on cells," *Fundamenta Informaticae*, vol. 144, no. 1, pp. 77–107, Mar. 2016.
- [28] B. Song and L. Pan, "The computational power of tissue-like P systems with promoters," *Theor. Comput. Sci.*, vol. 641, pp. 43–52, Aug. 2016.
- [29] M. Cavaliere and D. Sburlan, "Time-independent P systems," in *Membrane Computing (Lecture Notes in Computer Science)*. Berlin, Germany: Springer-Verlag, 2005, pp. 239–258.
- [30] T. Song, L. F. Macías-Ramos, L. Pan, and M. J. Pérez-Jiménez, "Time-free solution to SAT problem using P systems with active membranes," *Theor. Comput. Sci.*, vol. 529, pp. 61–68, Apr. 2014.
- [31] Y. Luo, Z. Xiong, and G. Zhang, "Time-free solution to SAT problem by tissue P systems," *Math. Problems Eng.*, vol. 2017, pp. 1–8, Feb. 2017.
- [32] X. Liu, J. Suo, S. C. H. Leung, J. Liu, and X. Zeng, "The power of time-free tissue P systems: Attacking NP-complete problems," *Neurocomputing*, vol. 159, pp. 151–156, Jul. 2015.
- [33] B. Song, M. J. Pérez-Jiménez, and L. Pan, "An efficient time-free solution to SAT problem by P systems with proteins on membranes," *J. Comput. Syst. Sci.*, vol. 82, no. 6, pp. 1090–1099, 2016.
- [34] B. Song, C. Zhang, and L. Pan, "Tissue-like P systems with evolutionary symport/antiport rules," *Inf. Sci.*, vol. 378, pp. 177–193, Feb. 2017.
- [35] Y. Tamori and W.-M. Deng, "Compensatory cellular hypertrophy: The other strategy for tissue homeostasis," *Trends Cell Biol.*, vol. 24, no. 4, pp. 230–237, Apr. 2014.
- [36] G. Rozenberg and A. Salomaa, *Handbook of Formal Languages*. Berlin, Germany: Springer, 1997.
- [37] M. Minsky, *Computation: Finite and Infinite Machines*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1967.
- [38] G. Ciobanu, L. Pan, G. Păun, and M. J. Pérez-Jiménez, "P systems with minimal parallelism," *Theor. Comput. Sci.*, vol. 378, no. 1, pp. 117–130, Jun. 2007.
- [39] L. Pan, Y. Wang, S. Jiang, and B. Song, "Flat maximal parallelism in tissue P systems with promoters," *Romanian J. Inf. Sci. Tech.*, vol. 20, no. 1, pp. 42–56, 2017.
- [40] L. Pan, A. Alhazov, H. Su, and B. Song, "Local synchronization on asynchronous tissue P systems with symport/antiport rules," *IEEE Trans. Nanobiosci.*, vol. 19, no. 2, pp. 315–320, Apr. 2020.



YUEGUO LUO was born in Gulin, Sichuan, China, in 1979. He received the Ph.D. degree from Chongqing University, China, in 2017. In 2004, he began to work as a Teacher at Yangtze Normal University, Chongqing, where he is currently an Associate Professor. His research interests include membrane computing and computational intelligence.



PING GUO (Member, IEEE) was born in Meishan, Sichuan, China, in 1963. He received the Ph.D. degree in computer software and theory from Chongqing University, China, in 2004. He is currently a Professor at the Chongqing Key Laboratory of Software Theory and Technology, College of Computer Science, Chongqing University, Chongqing, China. His research interests include different aspects of artificial intelligence and biological computing. He has authored/coauthored more than 150 refereed publications.



YUN JIANG was born in Hongan, Hubei, China, in 1983. She received the Ph.D. degree in system analysis and assembling from the Huazhong University of Science and Technology, China, in 2011. She is currently an Associate Professor at the School of Artificial Intelligence, Chongqing Technology and Business University, Chongqing, China. Her research interests include different aspects of artificial intelligence and biological computing.



YING ZHANG was born in Xianning, Hubei, China, in 1981. She received the master's degree in software engineering from Chongqing University, in 2006. She currently works with the College of Big Data and Intelligent Engineering, Yangtze Normal University. Her current research interests include image segmentation and computational intelligence.

...