Time-free solution to SAT problem using P systems with active membranes

Tao Song [a], Luis F. Macías-Ramos [b], Linqiang Pan [a,*], Mario J. Pérez-Jiménez [b]

[a] Key Laboratory of Image Processing and Intelligent Control, School of Automation, Huazhong University of Science and Technology, Wuhan 430074, Hubei, China
[b] Department of Computer Science and Artificial Intelligence, University of Sevilla, Avda. Reina Mercedes s/n, 41012 Sevilla, Spain

Keywords:

Membrane computing
P system
Time-free solution
Semi-uniform solution
NP-complete problem

A B S T R A C T

P systems are a class of distributed and parallel computation models inspired by the structure and the functioning of living cells. P systems have been used to solve computation hard problems, where the execution of each rule is completed in unit time (a global clock is assumed for timing and synchronizing the execution of rules). The assumption that the execution of each rule takes exactly one time unit plays an vital role to make a system working synchronously, and it has also been used to characterize the computational efficiency and time complexity of a system. In this work, we investigate the computation power of P systems without such time assumption. Specifically, we give a time-free solution to SAT problem using P systems with active membranes in the sense that the correctness of the solution does not depend on the precise timing of the involved rules.

1. Introduction

Membrane computing is a branch of natural computing, first introduced in [17]. The research aim of membrane computing is focused on abstracting computing concepts (i.e. models, data structures, data manipulation operations, operation control modes, etc.) from the structure and the functioning of living cells, considered both individually and as part of complexes, such as tissues and organs. An introduction to the area of membrane computing can be found in [19], while an overview of the "state-of-the-art" in 2010 can be found in [20], with up-to-date information available at the membrane computing website [27].

The computation models obtained in the framework of membrane computing are usually called *P systems*, which are distributed and parallel computation models. There are three main classes of P systems investigated: cell-like P systems [17], tissue-like P systems [11], neural-like P systems [9]. The present paper deals with a class of cell-like P systems, called *P systems with active membranes*, introduced in [18].

Briefly, P systems with active membranes consist of membranes that are organized in a hierarchical structure, where membranes can have an electrical charge (positive $+$, negative $-$ or neutral 0). Each membrane contains a multiset of objects. The charge of membranes, the whole membrane structure and the objects contained in membranes evolve by the specified evolution rules. When the evolution of the system stops, we obtain a computation result, where the computation

☆ This work was supported by National Natural Science Foundation of China (61033003, 91130034, 61100145, 61272071 and 61320106005), Ph.D. Programs Foundation of Ministry of Education of China (20100142110072 and 2012014213008), and Natural Science Foundation of Hubei Province (2011CDA027).

* Corresponding author. Tel.: +86 27 87556070.
E-mail address: lqpan@mail.hust.edu.cn (L. Pan).

result can be defined in several ways such as the number objects inside a specified membrane or the number objects in each membrane (i.e., a vector).

P systems with active membranes are proved to be universal [15], and have been used to solve computation hard problems [1–3,10,12–14,21,22,24]. All the above-mentioned P systems with active membranes work in a synchronized and parallel way (a global clock is assumed to mark the time for the system), in each tick of the global clock, all the applicable rules are applied simultaneously, and the execution of rules takes exactly one time unit. The assumption that the execution of rules takes exactly one time unit plays an important role to make the computation in each membrane happening synchronously, and it has been used to character the computational efficiency and time complexity of a system. It is of interest to investigate the computation power of P systems without such timing assumption [7].

In this work, we present a "time-free" solution to SAT problem using P systems with active membranes in the sense that the correctness of the solution does not depend on the precise timing of the involved rules.

2. P systems with active membranes

2.1. P systems with active membranes

In this subsection, we first introduce some necessary notion and notation from formal language theory (please refer to [23] for more detail), then recall the definition of P systems with active membranes [19].

For an alphabet V, V^* denotes the set of all finite strings of symbols from V, while the empty string is denoted by λ, and the set of all non-empty strings over V is denoted by V^+.

By $\mathbb{N}$ we denote the set of non-positive integers. Let U be an arbitrary set. A multiset (over U) is a mapping $M : U \to \mathbb{N}$. The multiplicity of a in the multiset M is denoted by $M(a)$ with $a \in U$. This can be expressed by the pair $(a, M(a))$. If the set $U = \{a_1, a_2, \ldots, a_n\}$ is finite, a multiset M over U, represented by the set of mappings $\{(a_1, M(a_1)), (a_2, M(a_2)), \ldots, (a_n, M(a_n))\}$ can also be represented by a string $w = a_1^{M(a_1)} a_2^{M(a_2)} \ldots a_n^{M(a_n)}$ or by any of its permutations. In what follows, we will not distinguish between the representation of multiset in mapping form or string form.

A *P system with active membranes* of degree m is a construct

$$\Pi = (O, H, \mu, w_1, \ldots, w_m, R),$$

where:

(i) $m \geqslant 1$ is the initial degree of the system;
(ii) O is the alphabet of *objects*;
(iii) H is a finite set of *labels* for membranes;
(iv) μ is the initial *membrane structure*, consisting of m membranes; membranes are labelled (not necessarily in an injective way) with elements of $H \times C$, where C is the set of electrical charges $\{+, -, 0\}$ (the elements in C denote positive, negative, neutral, respectively);
(v) $w_1, \ldots, w_m$ are strings over O, describing the *initial multisets of objects* placed in the m regions of μ;
(vi) R is a finite set of *development rules*, of the following types:
  (a) $[\ a \to v\ ]_h^\alpha$, $h \in H, \alpha \in C, a \in O, v \in O^*$ (object evolution rules, associated with membranes and depending on the label and the charge of the membranes);
  (b) $a[\ \ ]_h^{\alpha_1} \to [\ b\ ]_h^{\alpha_2}$, $h \in H, \alpha_1, \alpha_2 \in C, a, b \in O$ (communication rules; an object is sent into the membrane, possibly modified during this process; also the polarization of the membrane can be modified, but not its label);
  (c) $[\ a\ ]_h^{\alpha_1} \to [\ \ ]_h^{\alpha_2} b$, $h \in H, \alpha_1, \alpha_2 \in C, a, b \in O$ (communication rules; an object is sent out of the membrane, possibly modified during this process; also the polarization of the membrane can be modified, but not its label);
  (d) $[\ a\ ]_h^\alpha \to b$, $h \in H, \alpha \in C, a, b \in O$ (dissolving rules; in reaction with an object, a membrane can be dissolved, while the object specified in the rule can be modified);
  (e) $[\ a\ ]_h^{\alpha_1} \to [\ b\ ]_h^{\alpha_2} [\ c\ ]_h^{\alpha_3}$, $h \in H, \alpha_1, \alpha_2, \alpha_3 \in C, a, b, c \in O$ (division rules for elementary membranes; in reaction with an object, the membrane is divided into two membranes with the same label, possibly of different polarizations; the object specified in the rule is replaced in the two new membranes by possibly new objects;
  (f) $[[\ \ ]_{h_1}^{\alpha_1} \ldots [\ \ \ ]_{h_k}^{\alpha_1}\ [\ \ ]_{h_{k+1}}^{\alpha_2} \ldots [\ \ ]_{h_n}^{\alpha_2}\ ]_{h_0}^{\alpha_0} \to [[\ \ ]_{h_1}^{\alpha_3} \ldots [\ \ ]_{h_k}^{\alpha_3}\ ]_{h_0}^{\alpha_5}\ [[\ \ ]_{h_{k+1}}^{\alpha_4} \ldots [\ \ ]_{h_n}^{\alpha_4}]_{h_0}^{\alpha_6}$, $k \geqslant 1, n > k, h_i \in H, 0 \leqslant i \leqslant n$, and $\alpha_0, \ldots, \alpha_6 \in C$ with $\{\alpha_1, \alpha_2\} = \{+, -\}$ (if the membrane with label h_0 contains other membranes than those with the labels $h_1, \ldots, h_n$ specified above, then they must have neutral charges; these membranes are duplicated and then are part of the contents of both new copies of the membrane h_0).

The previous rules can be considered as "standard" rules of P systems with active membranes; the following rule can be considered as the extension of rule (e).

(e′) $[\ a\ ]_{h_1}^{\alpha_1} \to [\ b\ ]_{h_2}^{\alpha_2} [\ c\ ]_{h_3}^{\alpha_3}$, $h_1, h_2, h_3 \in H, \alpha_1, \alpha_2, \alpha_3 \in C, a, b, c \in O$ (h_1 is an elementary membrane; in reaction with an object, the membrane is divided into two membranes not necessarily with the same label; also the polarizations of the

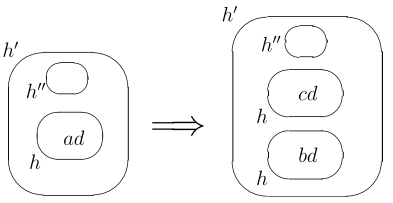

Fig. 1. Division of elementary membranes.

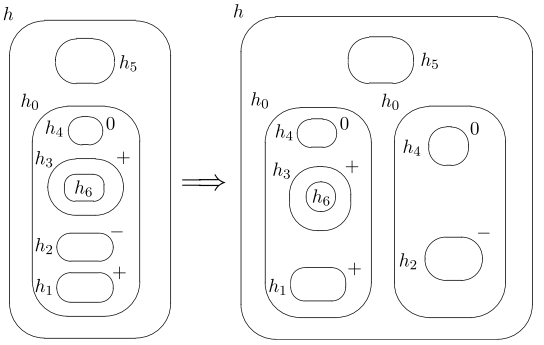

Fig. 2. Division of non-elementary membranes.

new membranes can be different from the polarization of the initial one; the object specified in the rule is replaced in the two new membranes by possibly new objects).

The difference of the rules of type (e) and type (e′) is that the resulting membranes obtained by applying the rules of type (e) have the same label with their parent membrane, and the resulting membranes obtained by applying the rules of type (e′) can have different labels with their parent membrane.

These rules are applied according to the following principles [18,19]:

(1) The rules of type (a) are applied to all objects to which they can be applied, and all other rules are applied to all membranes to which they can be applied; an object can be used by only one rule, non-deterministically chosen, but any object which can evolve by a rule of any form, should evolve.
(2) If a membrane is dissolved, then all the objects from its region are left free in the surrounding region. The skin membrane is never dissolved.
(3) All objects and membranes not specified in a rule and which do not evolve are passed unchanged to the next step. For instance, if a membrane with the label h is divided by a rule of type (e) which involves an object a, then all other objects from membrane h which do not evolve are introduced in each of the two resulting membranes h (this is the case of object d in Fig. 1). Similarly, when dividing a membrane h_0 by means of a rule of type (f), the neutral membranes are reproduced in each of the two new membranes with the label h_0, unchanged if no rule is applied to them (in particular, the contents of these neutral membranes are reproduced unchanged in these copies, providing that no rule is applied to their objects) – this is the case of membrane h_4 in Fig. 2.
(4) If at the same time a membrane h is divided by a rule of type (e) and there are objects in this membrane which evolve by means of rules of type (a), then in the new copies of the membrane we introduce the result of the evolution; that is, we suppose that first the evolution rules of type (a) are used, changing the objects, and then the division is produced, so that in the two new membranes with label h we introduce copies of the changed objects. Of course, this process takes only one step. The same assertions apply to the division by means of a rule of type (f): we always assume that the rules are applied in the bottom-up manner in one step, but first the rules of the innermost region and then level by level until the region of the skin membrane.
(5) The rules associated with a membrane h are used for all copies of this membrane, irrespective of whether or not the membrane is an initial one or it is obtained by division. At one step, a membrane h can be the subject of only one rule of types (b)–(f).
(6) The skin membrane can never divide.

The membrane structure of the system at a given time, together with all multisets of objects associated with the regions of this membrane structure, is the *configuration* of the system at that time. The $(m+1)$-tuple $(\mu, w_1, \ldots, w_m)$ is the *initial configuration*. We can pass from one configuration to another one by using the rules from R according to the principles given above. A sequence of transitions which starts from the initial configuration is called a *computation* with respect to Π.

A computation is *halting* if it cannot be continued: there is no rule which can be applied to objects and membranes in the last configuration. During a computation, objects can leave the skin membrane (by means of rules of type (c)). The result of a halting computation is the number of objects which are sent out of the system during the computation.

2.2. Timed P systems with active membranes

We recall the notion of timed P system from [4] (in our case, P systems with active membranes).

A *timed P system with active membranes* $\Pi(e) = (O, H, \mu, w_1, \ldots, w_m, R, e)$ is obtained by adding a time-mapping $e : R \to \mathbb{N}$ to a P system with active membranes $\Pi = (O, H, \mu, w_1, \ldots, w_m, R)$, where $\mathbb{N}$ is the set of natural numbers and the time-mapping e specifies the execution times for the rules.

A timed P system with active membranes $\Pi(e)$ works in the following way. An external clock is assumed, which marks time-units of equal length, starting from instant 0. According to this clock, the step t of computation is defined by the period of time that goes from instant $t-1$ to instant t. If a membrane i contains some rule r from types (a)–(f) and (e') selected to be executed, then the execution of such rule takes $e(r)$ time units to complete. Therefore, if the execution is started at instant j, the rule is completed at instant $j+e(r)$ and the resulting objects and membranes become available only at the beginning of step $j+e(r)+1$. When a rule r is started, then the occurrences of symbol-objects and the membrane subject to this rule cannot be subject to other rules until the implementation of the rule completes.

In a timed P system with active membranes, the application of rules also follows the bottom-up manner. For instance, at time instant j a membrane with label h is divided by a rule r_1: $[\ a\ ]_h^{\alpha_1} \to [\ b\ ]_h^{\alpha_2}[\ c\ ]_h^{\alpha_3}$, an object d in this membrane evolves by a rule r_2: $[\ d\ ]_h^{\alpha_1} \to [\ v\ ]_h^{\alpha_1}$, and suppose that $e(r_2) > e(r_1)$, then at time instant $j+e(r_1)$ the implementation of rule r_2 is still in process; by the bottom-up manner, at time instant $j+e(r_1)$, the implementation of rule r_1 does not complete; until time instant $j+e(r_2)$ (that is, the implementation of rule r_2 completes), the evolution result v is introduced in the new two copies of membrane with label h, and the implementation of rule r_1 completes (that is, the implementation of rule r_1 actually takes $e(r_2)$ steps).

A *recognizer timed P system with active membranes* is a *timed P system with active membranes* such that: (i) the working alphabet contains two distinguished elements `yes` and `no`; (ii) all computations halt; and (iii) if $\mathcal{C}$ is a computation of the system, then either object `yes` or object `no` (but not both) must appear in the environment when the system halts. In recognizer timed P systems with active membranes, we say that a computation is an *accepting computation* (resp., *rejecting computation*) if the object `yes` (resp., `no`) appears in the environment associated with the corresponding halting configuration.

2.3. Time-free solutions to decision problems by P systems with active membranes

In this subsection, we give the definition of time-free solutions to decision problems by P systems with active membranes, which is actually obtained by combining and adapting the notion of semi-uniform solution [8] and the notion of time-freeness [4].

In timed P systems with active membranes, a computation step is called a *rule starting step* (RS-step, for short) if at this step at least one rule starts its execution. In the following definition of time-free solutions to decision problems by P systems with active membranes, we will only count RS-steps (i.e., steps in which some object "starts" to evolve or some membrane "starts" to change). In timed P systems with active membranes, the execution time of rules is determined by the time mapping e, and a possible existence of rules with inherently exponential execution time. Therefore, the number of RS-steps in a computation characterize how "fast" the constructed P system with active membranes solves a decision problem in the context of time-freeness.

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

Let $X = (I_X, \Theta_X)$ be a decision problem. We say that X is solvable in a *polynomial time* by a family of time-free recognizer P systems with active membranes $\Pi = \Pi_u, u \in I_X$ (we also say that the family Π is a *time-free solution* to the decision problem X) if the following items are true:

- the family Π is polynomially uniform by a Turing machines; that is, there exists a deterministic Turing machine working in polynomial time which constructs the system Π_u from the instance $u \in I_X$;
- the family Π is *time-free sound* (with respect to X); that is, for any time-mapping e, the following property holds: if for each instance of the problem $u \in I_X$ such that there exists an accepting computation of $\Pi_u(e)$, we have $\Theta_X(u) = 1$;
- the family Π is *time-free complete* (with respect to X); that is, for any time-mapping e, the following property holds: if for each instance of the problem $u \in I_X$ such that $\Theta_X(u) = 1$, every computation of $\Pi_u(e)$ is an accepting computation;
- the family Π is *time-free polynomially bounded*; that is, there exists a polynomial function $p(n)$ such that for any time-mapping e and for each $u \in I_X$, all computations in $\Pi_u(e)$ halt in, at most, $p(|u|)$ RS-steps.

3. A time-free solution to SAT problem by P systems with active membranes

The SAT problem (satisfiability of propositional formulae in the conjunctive normal form) is probably the best known NP-complete problem [6], which asks whether or not for a given formula in the conjunctive normal form there is a truth-assignment of variables such that the formula assumes the value *true*.

The following theorem shows that SAT problem can be solved in a linear time by a family of time-free P systems with active membranes.

Theorem 3.1. SAT *problem can be solved by a family of time-free P systems with active membranes with rules of types* (a), (b), (c), (e′), (f) *in a linear time with respect to the number of variables and the number of clauses.*

Proof. Let us consider a propositional formula $C = C_1 \wedge C_2 \wedge \cdots \wedge C_m$, with $C_i = y_{i,1} \vee \cdots \vee y_{i,p_i}$, for some $m \geqslant 1$, $p_i \geqslant 1$, and $y_{i,j} \in \{x_k, \neg x_k \mid 1 \leqslant k \leqslant n\}$, for each $1 \leqslant i \leqslant m$, $1 \leqslant j \leqslant p_i$, where $\neg x_k$ is the negation of a propositional variable x_k, the two connections $\vee, \wedge$ are *or*, *and*, respectively.

For the given propositional formula C, we construct the P system with active membranes

$$\Pi_C = (O, H, \mu, w_0, w_1, w_{n+m+2}, R),$$

where

- $O = \{a_i, t_i, f_i \mid 1 \leqslant i \leqslant n\} \cup \{r_i \mid 1 \leqslant i \leqslant m\} \cup \{b_i \mid 1 \leqslant i \leqslant n\} \cup \{\mathtt{yes}, \mathtt{no}, b\}$ is the alphabet;
- $H = \{-1, 0, 1, 2, \ldots, n+m+1, n+m+2\}$ is the set of labels of the membranes;
- $\mu = [\,[\,[\,]_1^0\,]_0^0\,]_{n+m+2}^0$ is initial membrane structure;
- $w_0 = \lambda$ (that is, membrane 0 contains no object in the initial configuration);
- $w_1 = a_1 a_2 \ldots a_n \mathtt{yes}$ is the initial multiset contained in membrane 1;
- $w_{n+m+2} = \mathtt{no}$ is the initial multiset contained in membrane $n+m+2$;
- R is the set of rules of the following forms:
  - $G_{1i} : [\, a_i\,]_i^0 \to [\, t_i\,]_{i+1}^+ [\, f_i\,]_{i+1}^+$, $1 \leqslant i \leqslant n$,
  - $G_{2i} : [\, t_i \to r_{h_{i,1}} \ldots r_{h_{i,j_i}}\,]_{i+1}^+$, $1 \leqslant i \leqslant n$, and the clauses $C_{h_{i,1}}, \ldots, C_{h_{i,j_i}}$ contain the literal x_i,
  - $G_{3i} : [\, f_i \to r_{h_{i,1}} \ldots r_{h_{i,j_i}} b_i\,]_{i+1}^+$, $1 \leqslant i \leqslant n$, and the clauses $C_{h_{i,1}}, \ldots, C_{h_{i,j_i}}$ contain the literal $\neg x_i$,
  - $G_{4i} : [\, b_i\,]_{i+1}^+ \to [\,\,]_{i+1}^- b$, $1 \leqslant i \leqslant n$,
  - $G_{5i} : [\,[\,\,]_{i+1}^+ [\,\,]_{i+1}^-]_0^0 \to [\,[\,\,]_{i+1}^0]_0^0 [\,[\,\,]_{i+1}^0]_0^0$, $1 \leqslant i \leqslant n$,
  - $C_{1j} : [\, r_j\,]_{n+j}^0 \to [\,\,]_{n+j+1}^0 [\,\,]_{-1}^0$, $1 \leqslant j \leqslant m$,
  - $O_1 : [\, \mathtt{no}\,]_{n+m+2}^0 \to [\,\,]_{n+m+2}^+ \mathtt{no}$,
  - $O_2 : \mathtt{no} [\,\,]_{n+m+2}^- \to [\, \mathtt{no}\,]_{n+m+2}^-$,
  - $O_3 : [\, \mathtt{yes}\,]_{n+m+1}^0 \to [\,\,]_{n+m+1}^0 \mathtt{yes}$,
  - $O_4 : [\, \mathtt{yes}\,]_0^0 \to [\,\,]_0^0 \mathtt{yes}$,
  - $O_5 : [\, \mathtt{yes}\,]_{n+m+2}^+ \to [\,\,]_{n+m+2}^- \mathtt{yes}$.

In what follows, we show how the above constructed system Π_C give a solution to the propositional formula C. Generally, the computation process can be separated into three phases: generating phase, checking phase and output phase.

Generating phase. In the initial configuration of the system, membrane 1 contains objects $w_1 = a_1 a_2 \ldots a_n \mathtt{yes}$, and membrane 0 contains no object. The objects a_i in membrane 1 correspond to variable x_i, $1 \leqslant i \leqslant n$. At step 1, the rule G_{11} is applied, producing the truth values *true* (represented by t_1) and *false* (represented by f_1) assigned to variable x_1, placed in two separate copies of membrane 2. Note that when the membrane with label 1 is divided by the rule G_{11}, the obtained two membranes with label 2 instead of label 1, and their charges change from neutral to positive. For any given time-mapping e, the execution of rule G_{11} completes in $e(G_{11})$ steps. As we will see below, at step 1, exception for the application of rule G_{11}, the rule O_1 also starts; and from step 2 to step $e(G_{11})$, there is no rule starting. So, during the execution of rule G_{11} (i.e., from step 1 to step $e(G_{11})$), there is one RS-step. Note that the number of RS-steps during the execution of rule G_{11} is independent on the time-mapping e.

After the execution of rule G_{11} completes, the applications of rules G_{21} and G_{31} start (note that the application starts at the same step, but it may complete at different steps), which is actually a process looking for the clauses satisfied by the truth-assignment of variable x_1. After the execution of rule G_{31} completes, the application of rules G_{41} starts, where object b_1 evolves to a "dummy" object b (it will not evolve anymore), and object b exits the membrane changing its polarization from positive to negative.

After the execution of rule G_{41} completes, the membrane with label 0 contains membranes with polarization positive and negative. We have the following two cases:

- when the execution of rule G_{41} completes, the execution of rule G_{2i} already completes. In this case, the rule G_{51} is enabled, and its application starts;
- when the execution of rule G_{41} completes, the execution of rule G_{21} does not complete. In this case, the system will continue the execution of rule G_{21}. Only when the execution of rule G_{21} completes, the rule G_{51} will be enabled and applied.

So, the rule G_{51} has a synchronization functioning because $e(G_{31}) + e(G_{41})$ may not equal to $e(G_{21})$ for a time-mapping e. Anyway, when the execution of rule G_{51} completes, the computation takes four RS-steps, which is independent on any time-mapping e.

By the application of rule G_{51}, the polarization of the membranes with label 2 changes to neutral. In this way, the rule G_{12} is enabled and applied, which means that the system starts to assign truth values *true* and *false* to variable x_2. Similar to the case of variable x_1, the process of true value assignment of variable x_2 takes four RS-steps, and four membranes with label 0 are generated, each membrane with label 0 contains a membrane with label 3. In general, after $4n$ RS-steps, 2^n separate copies of membrane with label 0 are generated, all of which are placed in the membrane with label $n+m+2$; each membrane with label 0 contains a membrane with label $n+1$.

In the computation of generating phase, by the fact that all membranes with the same labels applied the same division rules, and the bottom-up manner of the application of rules, we have that the 2^n separate copies of membrane with label 0 are generated at the same time instant, which ensures that the following checking phase can start at the same time.

Checking phase. Each membrane with label $n+1$ contains some of objects $r_1, r_2, \ldots, r_m$ that correspond to the clauses satisfied by the truth assignment from that membrane. If there is at least one membrane with label $n+1$ that contains all objects $r_1, r_2, \ldots, r_m$, this means that the truth assignment from that membrane satisfies all clauses, hence it satisfies formula C. Otherwise, (if no membrane with label $n+1$ contains all objects $r_1, r_2, \ldots, r_m$), the formula C is not satisfiable.

By applying the rule C_{11}, at the same time for all 2^n membranes with label $n+1$, the system checks whether object r_1 is present in each membrane. If this is the case, then the membranes containing object r_1 are divided into two membranes with label $n+2$ and -1, respectively, where the membrane with label -1 is a "dummy" membrane that will not evolve anymore. The membranes that do not contain object r_1 cannot divide and they will no longer evolve, as no further rule can be applied to them. For a given time-mapping e, the checking of object r_1 takes $e(C_{11})$ steps, where the number of RS-steps is one.

In membranes with label $n+2$ (that is, the membranes where the first clause are already satisfied), by applying the rule C_{12}, the system checks whether object r_2 is present in each membrane. Only the membranes containing object r_2 will divide and membranes with label $n+3$ are obtained. As the above iteration, the rules of type C_{1j} are applied as many times as possible. Clearly, if a membrane with label $n+1$ does not contain an object r_i, then that membrane will stop evolving at the time when r_i is supposed to trigger a division. In this way, for a given time-mapping e, after at most $\sum_{1 \leqslant j \leqslant m} e(C_{1j})$ steps (where there are at most m RS-steps), we can find whether there is a membrane with label $n+1$ that contains all objects $r_1, r_2, \ldots, r_m$. The membranes with label $n+1$ having this property, and only they, will have offspring membranes with label $n+m+1$.

Output phase. At step 1, the rule O_1 is applied, object `no` exits the skin membrane $n+m+2$ changing its polarization from neutral to positive.

When the process of checking whether all clauses are satisfied completes, if no membrane with label $n+m+1$ is generated in any membrane with label 0, then the rules O_3, O_4 and O_5 cannot be applied. In this case, when the computation halts, object `no` remains in the environment, telling us that the formula is not satisfiable.

When the process of checking whether all clauses are satisfied completes, if there are membranes with label $n+m+1$ generated in membranes with label 0, then the rules O_3 and O_4 will be applied one by one. For a given time-mapping, after $e(O_3) + e(O_4)$ steps (where there are two RS-steps), object `yes` reaches membrane $n+m+2$. At this moment, if the execution of rule O_1 is not yet completed, then no rule can be started in the system before the execution of rule O_1 completes (note that the system will take computation steps to complete the execution of rule O_1, but there is no RS-step from this moment to the end of the execution of rule O_1). Only when the execution of rule O_1 completes, the polarization of membrane $n+m+2$ changes to positive, and the rule O_5 is enabled and applied. By the application of rule O_5, object `yes` exits membrane $n+m+2$ changing its polarization from positive to negative; in this way, the objects `yes` remaining in membrane $n+m+2$ are not able to continue exiting into the environment. After the execution of rule O_5 completes, the rule O_2 is enabled and applied, object `no` enters membrane $n+m+2$; in this way, when the computation halts, one copy of `yes` appears in the environment, telling us that the formula is satisfiable.

In what follows, we show that the system Π_c is time-free sound, time-free complete, and time-free linear bounded. Furthermore, the construction of system Π_C can be done in polynomial time by a Turing machine.

By the computation process to solve the formula C, we can find that for any time-mapping $e : R \to \mathbb{N}$, the following property holds: the object `yes` appears in the environment when the computation halts if and only if the formula C is satisfiable; and the object `no` appears in the environment when the computation halts if and only if the formula C is not satisfiable. So, the system Π_C is time-free sound and time-free complete.

If formula C is satisfiable, then object yes appears in the environment at RS-step $4n+m+3$: in $4n$ RS-steps the system generated 2^n membranes with $n+1$ (as well as the 2^n different truth-assignments); it takes m RS-steps to check whether all clauses are satisfied by an assignment; 3 RS-steps are necessary to output the computing result yes. Furthermore, it takes one more RS-step such that object no enters into the skin membrane $n+m+2$, and the system halts. So, the computation takes $4n+m+4$ RS-steps. If formula C is not satisfiable, then at step 1 the system sends object no into the environment (exactly, it takes one RS-step), and in $4n+m$ RS-steps the system halts. Therefore, the family of membrane systems we have constructed is linear time efficient in the context of time-freeness.

The family $\boldsymbol{\Pi} = \{\Pi_C \mid C \text{ is an instance of SAT problem}\}$ is polynomially uniform because the construction of P systems described in the proof can be done in polynomial time by a Turing machine:

- the total number of objects is $4n+m+3$;
- the number of initial membranes is 3;
- the cardinality of the initial multisets is 3;
- the total number of evolution rules is $5n+m+5$;
- the maximal length of a rule (the number of symbols necessary to write a rule, both its left and right sides, the membranes, and the polarizations of membranes involved in the rule) is $m+4$.

We omit the detailed construction due to the fact that it is straightforward but cumbersome as explained in the proof of Theorem 7.2.3 in [19]. Therefore, SAT problem can be decided in linear RS-steps with respect to the number of variables and the number of clauses by time-free recognizer P systems with active membranes and this concludes the proof. □

4. Conclusions and remarks

In this work, we give a time-free solution to SAT problem using P systems with active membranes in the sense that the correctness of the solution does not depend on the precise timing of the involved rules.

In a timed P system with active membranes, the application of rules follows the bottom-up manner. As explained in Section 2, with this manner of rule application, the implementation time of rules at a bottom level can influence the implementation time of rules at an up level. Such influence can be removed in the following manner of rule application. For instance, at time instant j a membrane with label h is divided by a rule r_1: $[\ a\ ]_h^{\alpha_1} \to [\ b\ ]_h^{\alpha_2}[\ c\ ]_h^{\alpha_3}$, an object d in this membrane evolves by a rule r_2: $[\ d\ ]_h^{\alpha_1} \to [\ v\ ]_h^{\alpha_1}$, and suppose that $e(r_2) > e(r_1)$, then at time instant $j+e(r_1)$ the implementation of rule r_1 completes, which produces two new membranes with label h; the implementation of rule r_2 is still in process; these two new membranes inherit the implementation of rule r_2, which will complete at time instant $j+e(r_2)$; these two new membranes and their objects are subject to further rules except for both copies of the object d that are still processed by rule r_2 until the instant $j+e(r_2)$. It remains open with the new manner of rule application, how we can solve SAT problem by P systems with active membranes in the context of time-freeness.

The definition of "time-free solutions to decision problems by P systems with active membranes" was given in Section 2, where we call a family of P systems is time-free polynomially bounded if there exists a polynomial function $p(n)$ such that for any time-mapping e and for each $u \in I_X$, all computations in $\Pi_u(e)$ halt in, at most, $p(|u|)$ RS-steps. By this definition, at each computation step, an observer has to know whether or not there is any rule that starts at this step, in order to count the number of RS-steps in the computation. Particularly, it is possible that the time of execution of a rule is inherently exponential with respect to the size of an instance; that is, during the execution of the rule, an observer has to check exponential steps in order to know the number of RS-steps. For a more reasonable definition, we can consider to require that for a given system, the numbers of RS-steps in all computations associated with different time-mapping are same. For any time-mapping, the number of RS-steps in a computation equals to the number of RS-steps when the time-mapping is as follows: $e(r) = 1$, $r \in R$, R is the set of rules. So, in this case, in order to know the number of RS-steps in a computation, it is enough to know the number of computation steps when the associated time-mapping $e = 1$. Actually, the proof given in Section 3 also works by the more "reasonable" definition.

The P systems constructed in the proof of Theorem 3.1 have the rules of types (a), (b), (c), (e′) and (f), where the rules of type (e′) are an extension of the rules of type (e), the new membranes obtained by the application of the rules of type (e′) can have different labels. It remains open how we construct P systems with rules of types (a), (b), (c), (e) and (f) to time-freely solve SAT problem.

The solution to SAT problem given in Section 3 is semi-uniform in the sense that P systems are constructed from the instances of the problem. It remains open how we can give a uniform time-free solution to SAT problem in the sense that P systems are constructed from the size of instances of the problem (that is, a P system can solve a family of instances with the same size).

Small universal P systems have been studied in the framework of membrane computing [5,16,25,26]. It is interesting whether we can construct small universal P systems in the context of time-freeness.

In this work, we already show that an NP-complete problem, the SAT problem, can be solved by P systems in a time-free manner. It remains open whether PSPACE problems can be solved by P systems in a time-free manner.

We can find many solutions to computationally hard problems by P systems in the area of membrane computing. It is of interest to investigate whether we can find a way to transform the solutions into equivalent time-free versions.

References

[1] A. Alhazov, R. Freund, On the efficiency of P systems with active membranes and two polarizations, in: Pre-proceedings of the Fifth Workshop on Membrane Computing, Milano, Italy, 2004, pp. 81–94.
[2] A. Alhazov, C. Martín-Vide, L. Pan, Solving a PSPACE-complete problem by recognizing P systems with restricted active membranes, Fundam. Inform. 58 (2) (2003) 67–77.
[3] A. Alhazov, L. Pan, Gh. Păun, Trading polarizations for labels in P systems with active membranes, Acta Inform. 41 (2–3) (2004) 111–144.
[4] M. Cavaliere, D. Sburlan, Time-independent P systems, in: Lecture Notes on Computer Science, vol. 3365, Springer-Verlag, Berlin, 2005, pp. 239–258.
[5] C. Erzsébet, M. Maurice, V. György, V. Sergey, On small universal antiport P systems, Theor. Comput. Sci. 372 (2) (2007) 152–164.
[6] M.R. Garey, D.J. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W.H. Freeman, San Francisco, 1979.
[7] M. Cavaliere, Time-free solution to hard computational problems, Section 12, in: M. Gheorghe, Gh. Păun, M.J. Pérez-Jiménez (Eds.), Frontiers of Membrane Computing: Open Problems and Research Topics, Proceedings of Tenth Brainstorming Week on Membrane Computing, Sevilla, vol. 1, 2012, pp. 204–210.
[8] M.A. Gutiérrez-Naranjo, M.J. Pérez-Jiménez, A. Riscos-Núñez, F.J. Romero-Campero, Computational efficiency of dissolution rules in membrane systems, Int. J. Comput. Math. 83 (2006) 593–611.
[9] M. Ionescu, Gh. Păun, T. Yokomori, Spiking neural P systems, Fundam. Inform. 71 (2–3) (2006) 279–308.
[10] S.N. Krishna, R. Rama, A variant of P systems with active membranes: solving NP-complete problems, Roman. J. Inf. Sci. Technol. 2 (4) (1999) 357–367.
[11] C. Martin-Vide, J. Pazos, Gh. Păun, A. Rodriguez-Patón, Tissue P systems, Theor. Comput. Sci. 296 (2) (2003) 295–326.
[12] L. Pan, D. Díaz-Pernil, M.J. Pérez-Jiménez, Computation of Ramsey numbers by P systems with active membranes, Int. J. Found. Comput. Sci. 22 (1) (2011) 29–38.
[13] L. Pan, C. Martín-Vide, Solving multidimensional 0–1 knapsack problem by P systems with input and active membranes, J. Parallel Distrib. Comput. 65 (2005) 1578–1584.
[14] L. Pan, C. Martín-Vide, Further remark on P systems with active membranes and two polarizations, J. Parallel Distrib. Comput. 66 (2006) 867–872.
[15] A. Păun, On P systems with membrane division, in: Proceedings of the Second International Conference on Unconventional Models of Computation, London, UK, 2000, pp. 187–201.
[16] A. Păun, G. Păun, Small universal spiking neural P systems, Biosystems 90 (1) (2007) 48–60.
[17] Gh. Păun, Computing with membranes, J. Comput. Syst. Sci. 61 (1) (2000) 108–143.
[18] Gh. Păun, P systems with active membranes: attacking NP-complete problems, J. Autom. Lang. Comb. 6 (1) (2001) 75–90.
[19] Gh. Păun, Membrane Computing. An Introduction, Springer-Verlag, Berlin, 2002.
[20] Gh. Păun, G. Rozenberg, A. Salomaa (Eds.), Handbook of Membrane Computing, Oxford University Press, 2010.
[21] Gh. Păun, Y. Suzuki, H. Tanaka, T. Yokomori, On the power of membrane division in P systems, Theor. Comput. Sci. 324 (1) (2004) 61–85.
[22] M.J. Pérez-Jiménez, Agustín Riscos-Núñez, A linear-time solution to the knapsack problem using P systems with active membranes, in: Pre-proceedings of the Fourth Workshop on Membrane Computing, Tarragona, Spain, 2003, pp. 17–22.
[23] G. Rozenberg, A. Salomaa (Eds.), Handbook of Formal Languages, vol. 3, Springer-Verlag, Berlin, 1997.
[24] P. Sosík, The computational power of cell division in P systems: Beating down parallel computers?, Nat. Comput. 2 (3) (2003) 287–298.
[25] T. Song, Y. Jiang, X. Shi, X. Zeng, Small universal spiking neural P systems with anti-spikes, J. Comput. Theor. Nanosci. 10 (4) (2013) 999–1006.
[26] X. Zhang, X. Zeng, L. Pan, Smaller universal spiking neural P systems, Fundam. Inform. 87 (1) (2008) 117–136.
[27] The P systems website, http://ppage.psystems.eu.