Math. Struct. in Comp. Science (2017), *vol.* 27, *pp.* 17–32. © Cambridge University Press 2015
doi:10.1017/S0960129515000018 First published online 27 February 2015

A time-free uniform solution to subset sum problem by tissue P systems with cell division

BOSHENG SONG[†,‡], TAO SONG[†] and LINQIANG PAN[†]

[†]*Key Laboratory of Image Information Processing and Intelligent Control, School of Automation, Huazhong University of Science and Technology, Wuhan 430074, Hubei, China*
Email: lqpan@mail.hust.edu.cn
[‡]*Department of Computer Science and Artificial Intelligence, University of Sevilla, Avda. Reina Mercedes s/n, 41012 Sevilla, Spain*

Received 14 May 2014

Tissue P systems are a class of bio-inspired computing models motivated by biochemical interactions between cells in a tissue-like arrangement. Tissue P systems with cell division offer a theoretical device to generate an exponentially growing structure in order to solve computationally hard problems efficiently with the assumption that there exists a global clock to mark the time for the system, the execution of each rule is completed in exactly one time unit. Actually, the execution time of different biochemical reactions in cells depends on many uncertain factors. In this work, with this biological inspiration, we remove the restriction on the execution time of each rule, and the computational efficiency of tissue P systems with cell division is investigated. Specifically, we solve subset sum problem by tissue P systems with cell division in a time-free manner in the sense that the correctness of the solution to the problem does not depend on the execution time of the involved rules.

1. Introduction

Membrane computing is a branch of natural computing introduced by Păun (2000). Computation models (usually called *P systems*) in membrane computing are inspired by the structure and functioning of cells as living organisms able to process and generate information. Until now, there are three main types of P systems investigated: cell-like P systems, inspired by the living cell with its compartments arranged in a hierarchical structure, each compartment containing other compartments and bio-chemical elements inside (Păun 2000); tissue-like P systems, inspired by cell inter-communication in tissues and in certain environment, which is used to exchange information or chemical elements (Martín-Vide *et al.* 2003); neural-like P systems, inspired by the way the neurons communicate by means of electrical impulses of identical shape, called *spikes* (Ionescu *et al.* 2006). An introduction of membrane computing can be found in Păun (2002), and a recent coverage of this area can be found in Păun *et al.* (2010), while the most up-to-date source of information for membrane computing, one may refer to the P systems website: http://ppage.psystems.eu.

The present work focuses on *tissue-like P systems.* Tissue-like membrane structures are described by graphs, where membranes (also called *cells*) are placed in nodes of a graph. An edge between two nodes corresponds to a communication channel between

https://doi.org/10.1017/S0960129515000018 Published online by Cambridge University Press

cells placed in these nodes. Certain cells can communicate directly through channels and all cells can communicate through the environment. The communication among cells is based on symport/antiport rules (Păun and Păun 2002). Symport rules move objects across a membrane together in one direction, whereas in the case of antiport rules, objects residing at both sides of the membrane cross it simultaneously but in opposite directions.

There are several variants of tissue-like P systems (see Alhazov *et al.* (2005, 2006); Bernardini and Gheorghe (2005); Freund *et al.* (2005)). An interesting variant of tissue P systems, called *tissue P systems with cell division*, was proposed in Păun *et al.* (2008), where the biological inspiration of such computing model is obvious: alive tissues are not static network of cells, since cells are duplicated via mitosis in a natural way. An exponential amount of cells can be generated by cell division, which provides a theoretical basis for solving the computationally hard problems. Therefore, it is natural to investigate the computational efficiency of tissue P systems with cell division. The first attempt in this topic was done in Păun *et al.* (2008), where SAT problem was solved by using tissue P systems with cell division in a polynomial time. In a series of subsequent works, tissue P systems with cell division were also considered to solve other **NP**-complete problems, such as vertex cover problem (Díaz-Pernil *et al.* 2008a), subset sum problem (Díaz-Pernil *et al.* 2007a), 3-colouring problem (Díaz-Pernil *et al.* 2007b, 2008b), independent set problem (Díaz-Pernil *et al.* 2009).

In fact, in all the above mentioned works of tissue P systems with cell division the following feature is used: a global clock is assumed and used to mark time units for the system, in each time unit, all the applicable rules are applied simultaneously, and the execution of each rule completes in exactly one time unit. Actually, in cell biology, the execution time of various biological reactions might be difficult to know precisely, and it depends on many factors, such as environmental conditions, catalyst, medium temperature and biological signals. With this biological fact, *timed P systems* were proposed in Cavaliere and Sburlan (2005), where to each rule a natural number is associated representing the execution time of the rule. In particular, an interesting class of timed P systems, called *time-free P systems*, were investigated and proved to be universal, where such P systems generate (or accept) the same family of vectors of natural numbers, independently from the execution time of each rule (Cavaliere and Sburlan 2005).

Time-free solutions to computationally hard problems as open problems were formulated in Gheorghe *et al.* (2013). In Song *et al.* (2014a), a time-free solution to SAT problem using P systems with active membranes was given in the sense that the correctness of the solution to the problem does not depend on the execution time of the involved rules. Time-free solution to subset sum problem by spiking neural P systems was presented (Song *et al.* 2014b) (the universality result of time-free spiking neural P systems was already obtained in Pan *et al.* (2011)).

In this work, we investigate the computational efficiency of tissue P systems with cell division in the context of time-freeness. Specifically, we solve subset sum problem by tissue P systems with cell division in a time-free manner, and furthermore, the P systems are constructed in a uniform way in the sense that there exists a deterministic Turing machine working in polynomial time which constructs the systems from the sizes of instances.

https://doi.org/10.1017/S0960129515000018 Published online by Cambridge University Press

2. Tissue P systems with cell division

2.1. *Preliminaries*

We assume the readers to be familiar with basic notions and notations from formal language theory, and here we only introduce some necessary notions and notations, one can refer to Salomaa and Rozenberg (1997) for more details.

An *alphabet* Σ, is a non-empty set whose elements are called *symbols*. An ordered finite sequence of symbols is a *string* or *word*. If u and v are strings over Σ, then so is their *concatenation* uv, obtained by juxtaposition, that is, writing u and v after one another. The number of symbols in a string u is the *length* of the string, and it is denoted by $|u|$. As usual, the empty string (with length 0) will be denoted by λ. The set of all strings over an alphabet Σ is denoted by Σ^* and by $\Sigma^+ = \Sigma^* \setminus \{\lambda\}$ we denote the set of non-empty strings. A *language* over Σ is a subset of Σ^*.

A *multiset* m over a set A is a pair (A, f) where $f : A \to \mathbb{N}$ is a mapping. If $m = (A, f)$ is a multiset then its *support* is defined as $supp(m) = \{x \in A \mid f(x) > 0\}$. A multiset is finite if its support is a finite set. If $m = (A, f)$ is a finite multiset over A, and $supp(m) = \{a_1, \ldots, a_k\}$ then it will be denoted as $m = \{a_1^{f(a_1)}, \ldots, a_k^{f(a_k)}\}$.

If $m_1 = (A, f_1)$, $m_2 = (A, f_2)$ are multisets over A, then we define the union of m_1 and m_2 as $m_1 + m_2 = (A, g)$, where $g = f_1 + f_2$.

For any sets A and B, we define the *relative complement* $A \setminus B$ of B in A as follows: $A \setminus B = \{x \in A \mid x \notin B\}$.

2.2. *Tissue P systems with cell division*

The model of tissue P systems was first present in Martín-Vide *et al.* (2002), where the membranes are placed in the nodes of a graph. Based on the basic feature of cell-like P systems with active membranes, cell division, a new model of tissue P systems with cell division was proposed in Păun *et al.* (2008).

Definition 1. A tissue P system with cell division of degree $q \geqslant 1$ is a construct

$$\Pi = (\Gamma, \mathcal{E}, \mathcal{M}_1, \ldots, \mathcal{M}_q, \mathcal{R}, i_{out}),$$

where

1. Γ is a finite alphabet, whose symbols are called *objects*;
2. $\mathcal{E} \subseteq \Gamma$ is the set of objects initially placed in the environment of the system, all of them available in an arbitrary number of copies;
3. $\mathcal{M}_i, 1 \leqslant i \leqslant q$, are multisets over Γ, representing the finite multisets of objects initially placed in the q cells of the system;
4. $\mathcal{R}$ is a finite set of rules of the following forms:
   a. Communication rules: $(i, u/v, j)$, where $i, j \in \{0, 1, 2, \ldots, q\}$, $i \neq j$, $u, v \in \Gamma^*$, $|uv| > 0$, we use region i $(0 \leqslant i \leqslant q)$ to refer to membrane i in the case $1 \leqslant i \leqslant q$ and to refer to the environment in the case $i = 0$.
   When applying a rule $(i, u/v, j)$, the objects of the multiset represented by u are sent from region i to region j and simultaneously the objects of the multiset v are

https://doi.org/10.1017/S0960129515000018 Published online by Cambridge University Press

sent from region j to region i (the maximum of the lengths of u and v is the weight of the rule).

b. Division rules: $[\ a\ ]_i \rightarrow [\ b\ ]_i[\ c\ ]_i$, where $i \in \{1, 2, \ldots, q\}$, $a, b, c \in \Gamma$, and $i \neq i_{out}$. A division rule $[\ a\ ]_i \rightarrow [\ b\ ]_i[\ c\ ]_i$ is inspired under the influence of object a in the cell with label i, when applying such a rule, membrane i is divided in two cells with the same label; in the first copy the object a is replaced by b, in the second copy the object a is replaced by c; all other objects are replicated and copies of them are placed in the two new cells.

5. $i_{out} \in \{0, 1, 2, \ldots, q\}$ is the output region (it can be a region inside a cell or the environment).

The rules of a tissue P system with cell division are used in the non-deterministic maximally parallel manner as customary in membrane computing. At each step, all cells which can evolve must evolve in a maximally parallel way (at each step we apply a multiset of rules which is maximal, no further rule can be added being applicable). This way of applying rules has only one restriction: when a cell is divided, the division rule is the only one which is applied for that cell at that step; thus, the objects inside that cell do not evolve by means of communication rules. The new cells resulting from division could participate in the interaction with other cells or the environment by means of communication rules at the next step – providing that they are not divided once again. The label of a cell precisely identify the rules which can be applied to it.

A *configuration* at any instant of a tissue P system with cell division is described by all multisets of objects over Γ associated with all the cells present in the system, and the multiset of objects over $\Gamma \setminus \mathcal{E}$ associated with the environment at that moment. Note that the objects from $\mathcal{E}$ have infinite copies in the environment, they are not properly changed along the computation. The initial configuration is described by $(\mathcal{M}_1, \ldots, \mathcal{M}_q, \varnothing)$. The system passes from one configuration to another by a maximally parallel application of rules as described above. A *computation* is a (finite or infinite) sequence of configurations such that: (a) the first term of the sequence is the initial configuration of the system; (b) each non-initial configuration of the sequence is obtained from the previous configuration by applying rules of the system in a maximally parallel manner with the restrictions previously mentioned; and (c) if the sequence is finite (called *halting computation*) then the last term of the sequence is a halting configuration, that is, a configuration is a halting configuration if no rule of the system is applicable to it.

All the computations start from an initial configuration and proceed as stated above; only halting computations give a result, which is encoded by the objects present in the output region i_{out} in the halting configuration.

2.3. *Recognizer tissue P systems with cell division*

Let us recall that a *decision problem*, X, is a pair (I_X, θ_X) such that I_X is a language over a finite alphabet (whose elements are called *instances*) and θ_X is a total Boolean function (that is, predicate) over I_X.

https://doi.org/10.1017/S0960129515000018 Published online by Cambridge University Press

In order to solve decision problems, we usually use recognizer P systems, which was first introduced in Pérez-Jiménez *et al.* (2006). In this work, we give the definition of recognizer tissue P systems with cell division, which is slightly different from Păun *et al.* (2008).

Definition 2. A recognizer tissue P system with cell division of degree $q \geqslant 1$ is a tuple $\Pi = (\Gamma, \Sigma, \mathcal{E}, \mathcal{M}_1, \ldots, \mathcal{M}_q, \mathcal{R}, i_{out}, i_{in})$ such that

— The tuple $(\Gamma, \mathcal{E}, \mathcal{M}_1, \ldots, \mathcal{M}_q, \mathcal{R}, i_{out})$ is a tissue P system with cell division of degree $q \geqslant 1$.
— Σ is an (input) alphabet strictly contained in Γ, and $\mathcal{E} \subseteq \Gamma \setminus \Sigma$.
— $\mathcal{M}_i$, $1 \leqslant i \leqslant q$, are multisets over $\Gamma \setminus \Sigma$.
— $i_{in} \in \{1, \ldots, q\}$ is the input cell.
— The output region i_{out} is the environment.
— The working alphabet contains two distinguished elements `yes` and `no`.
— All the computations halt.
— 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.

For recognizer tissue P systems with cell division, 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.

For each $w \in \Sigma^*$, the computation of the system Π with input w starts from the configuration of the form $(\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_{i_{in}} + w, \ldots, \mathcal{M}_q, \varnothing)$, that is, the input multiset w has been added to the contents of the input cell i_{in}. Therefore, we have an initial configuration associated with each input multiset w (over the input alphabet Σ) in this kind of systems.

2.4. *Recognizer timed tissue P systems with cell division*

In this subsection, we first introduce the definition of timed tissue P systems with cell division, then the notion of recognizer timed tissue P systems with cell division is given.

Definition 3. A timed tissue P system with cell division of degree $q \geqslant 1$ is a pair (Π, e), where Π is a tissue P system with cell division of degree m, and e is a time-mapping of Π, that is, e is a mapping from the finite set of rules $\mathcal{R}$ into the set of natural numbers $\mathbb{N}$, which represent the execution times for the rules. We denote by $\Pi(e)$ the timed tissue P system with cell division.

A timed tissue P system with cell division $\Pi(e)$ 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 rule r from $\mathcal{R}$ selected to be executed, then execution of such rule takes $e(r)$ time units to complete. Therefore, if the execution is started at instant j, the rule is completed at instant $j + e(r)$ and the resulting objects and membranes generated by the division rule become available only at the beginning of step

https://doi.org/10.1017/S0960129515000018 Published online by Cambridge University Press

$j + e(r) + 1$. If a communication rule is started, then the occurrences of objects subject to this rule cannot be subject to other rules until the implementation of the rule completes; if a division rule is started, then the occurrences of object and membrane subject to this rule cannot be subject to other rules until the implementation of the rule completes.

Definition 4. A recognizer timed tissue P system with cell division of degree $q \geqslant 1$ is a tuple (Π, e), where Π is a recognizer tissue P system with cell division of degree q and e is a time-mapping of Π.

2.5. *Time-free uniform solutions to decision problems by tissue P systems with cell division*

In a timed tissue P system with cell division, each rule is associated with a natural number representing the execution time of the rule, and it is possible there exist rules whose execution times are inherently exponential. So, we use the notion of RS-step (see Song *et al.* (2014a)) to characterize how 'fast' the constructed tissue P system with cell division solves a decision problem in the context of time-freeness.

Definition 5. In timed tissue P systems with cell division, a computation step is called a RS-step if at this step at least one rule starts its execution, that is, steps in which some objects 'start' to change their places or cells 'start' to divide.

Definition 6. We say that a decision problem $X = (I_X, \theta_X)$ is solvable in polynomial RS-steps by a family $\mathbf{\Pi} = \{\Pi_n \mid n \in \mathbb{N}\}$ of recognizer tissue P systems with cell division in a time-free manner, if the following holds:

— The family $\mathbf{\Pi}$ is polynomially uniform by Turing machines, that is, there exists a deterministic Turing machine working in polynomial time which constructs the system Π_n from $n \in \mathbb{N}$.
— There exists a pair (cod, s) of polynomial-time computable functions over I_X such that:
  – for each instance $u \in I_X$, $s(u)$ is a natural number and $cod(u)$ is an input multiset of the system $\Pi_{s(u)}$;
  – the family $\mathbf{\Pi}$ is time-free sound with respect to (X, cod, s); 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 $\Pi_{s(u)}(e)$ with input $cod(u)$, we have $\Theta_X(u) = 1$;
  – the family $\mathbf{\Pi}$ is time-free complete with respect to (X, cod, s); 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 $\Pi_{s(u)}(e)$ with input $cod(u)$ is an accepting computation;
  – the family $\mathbf{\Pi}$ is time-free polynomially bounded with respect to (X, cod, s); that is, there exists a polynomial function $p(n)$ such that for any time-mapping e and for each $u \in I_X$, all the computations in $\Pi_{s(u)}(e)$ with input $cod(u)$ halt in, at most, $p(|u|)$ RS-steps.

We also say that the family $\mathbf{\Pi}$ is a *time-free uniform solution* to the decision problem X.

https://doi.org/10.1017/S0960129515000018 Published online by Cambridge University Press

From the time-free soundness and time-free completeness conditions in Definition 6, we can deduce that every P system Π_n is *time-free confluent* in the following sense: every computation of a system with the *same* input multiset must always give the *same* answer.

3. A time-free uniform solution to subset sum problem using tissue P systems with cell division

The subset sum problem is described as follows: *given a finite set A, a weight function, $w : A \to \mathbb{N}$, and a constant $k \in \mathbb{N}$, determine whether or not there exists a subset $B \subseteq A$ such that $w(B) = k$. If A has n elements with weights $w_1, \ldots, w_n$, one instance of the problem can be encoded as $(n, (w_1, \ldots, w_n), k)$.*

Theorem 3.1. Subset sum problem can be solved in polynomial RS-steps by a uniform family of tissue P systems with cell division in a time-free manner.

Proof. Let $A = \{s_1, s_2, \ldots, s_n\}$ be a finite set, $w : A \to \mathbb{N}$ a weight function with $n = |A|$ and $k \in \mathbb{N}$. Let $g : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ be a function defined by $g\langle n, k\rangle = ((n+k)(n+k+1)/2)+n$. It is a primitive recursive and bijective function from $\mathbb{N}^2$ onto $\mathbb{N}$. Let us denote by $u = (n, (w_1, \ldots, w_n), k)$, where $w_i = w(a_i), 1 \leqslant i \leqslant n$, the given instance of the problem. We define the polynomially computable function $s(u) = g\langle n, k\rangle$.

In what follows, we will provide a uniform family of tissue P systems, where each P system solves all the instances of the subset sum problem with the same size. The weight function w of the concrete instance will be provided via an input multiset determined via the function $cod(u) = \{v_i^j : w(a_i) = j \wedge 1 \leqslant i \leqslant n\} \cup \{q^k\}$, where v_i^j (j copies of object v_i) represents that j is the weight of the element a_i.

For each $\langle n, k\rangle \in \mathbb{N}^2$, we consider the tissue P system

$$\Pi_{(\langle n,k\rangle)} = (\Gamma, \Sigma, \mathcal{M}_1, \mathcal{M}_2, \mathcal{R}, \mathcal{E}, i_{in}, i_{out}),$$

where

— $\Gamma = \Sigma \cup \{a_i, a_i^{(1)}, a_i^{(2)}, a_i^{(3)}, a_i^{(4)}, B_i, c_i, d_i, e_i, d_{i,1} \mid 1 \leqslant i \leqslant n\} \cup \{b_i \mid 2 \leqslant i \leqslant \log n + 1\} \cup \{d_i' \mid 2 \leqslant i \leqslant n+1\} \cup \{B_{i,j} \mid 1 \leqslant i \leqslant n, 1 \leqslant j \leqslant \log(k+1)+1\} \cup \{f_{1,i} \mid 1 \leqslant i \leqslant \log(k+1)+1\} \cup \{f_{i,j} \mid 2 \leqslant i \leqslant n+1, 1 \leqslant j \leqslant n + \log(k+1)+1\} \cup \{t_i \mid 1 \leqslant i \leqslant 6\} \cup \{\mathtt{yes}, \mathtt{no}, a_0, a_{n+1}, g, h, l, m, n, p, t\}$,
— $\Sigma = \{q\} \cup \{v_i : 1 \leqslant i \leqslant n\}$,
— $\mathcal{M}_1 = l$ yes no,
— $\mathcal{M}_2 = a_1\ g\ m\ n$,
— $\mathcal{R}$ is the following set of rules:

$r_{1,i} \equiv [\ a_i\ ]_2 \to [\ a_i^{(1)}\ ]_2[\ a_i^{(2)}\ ]_2,\ 1 \leqslant i \leqslant n.$
$r_{2,i} \equiv (2, a_i^{(1)}/a_0 a_i^{(3)} B_i, 0),\ 1 \leqslant i \leqslant n.$
$r_{3,i} \equiv (2, a_i^{(2)}/a_0 a_i^{(4)}, 0),\ 1 \leqslant i \leqslant n.$
$r_{4,i} \equiv (2, a_i^{(3)}/\lambda, 1),\ 1 \leqslant i \leqslant n.$
$r_{5,i} \equiv (2, a_i^{(4)}/\lambda, 1),\ 1 \leqslant i \leqslant n.$
$r_{6,i} \equiv (1, a_i^{(3)} a_i^{(4)}/a_{i+1}^2, 0),\ 1 \leqslant i \leqslant n.$
$r_{7,i} \equiv (2, a_0/a_{i+1}, 1),\ 1 \leqslant i \leqslant n.$

https://doi.org/10.1017/S0960129515000018 Published online by Cambridge University Press

$r_8 \equiv (2, a_{n+1}/b_2^2, 0)$.
$r_{9,i} \equiv (2, b_i/b_{i+1}^2, 0),\ 2 \leqslant i \leqslant \log n$.
$r_{10,i} \equiv (2, b_{\log n+1} B_i/c_i d_{i,1} B_{i,1}, 0),\ 1 \leqslant i \leqslant n$.
$r_{11} \equiv (2, c_1/\lambda, 1)$.
$r_{12,i} \equiv (2, c_{i+1}/c_i, 1),\ 1 \leqslant i \leqslant n-1$.
$r_{13} \equiv (1, c_n/d_1, 0)$.
$r_{14,i} \equiv (2, d_{i,1}/d_i, 1),\ 1 \leqslant i \leqslant n$.
$r_{15,i} \equiv (2, B_{i,1} d_i/B_{i,2}^2 e_i, 0),\ 1 \leqslant i \leqslant n$.
$r_{16,i,j} \equiv (2, B_{i,j}/B_{i,j+1}^2, 0),\ 1 \leqslant i \leqslant n, 2 \leqslant j \leqslant \log(k+1) - 1$.
$r_{17,i} \equiv (2, B_{i,\log(k+1)}/B_{i,\log(k+1)+1}^2 d'_{i+1}, 0),\ 1 \leqslant i \leqslant n$.
$r_{18,i} \equiv (2, d'_i/\lambda, 1),\ 2 \leqslant i \leqslant n+1$.
$r_{19,i} \equiv (1, d'_i/d_i, 0),\ 2 \leqslant i \leqslant n$.
$r_{20} \equiv (1, d'_{n+1}/f_{1,1}, 0)$.
$r_{21} \equiv (2, e_1/f_{1,1}, 1)$.
$r_{22,i} \equiv (2, f_{1,i}/f_{1,i+1}^2, 0),\ 1 \leqslant i \leqslant \log(k+1)$.
$r_{23} \equiv (2, B_{1,\log(k+1)+1} v_1 f_{1,\log(k+1)+1}/f_{2,1} p, 0)$.
$r_{24,i} \equiv (2, f_{i,1}/\lambda, 1),\ 2 \leqslant i \leqslant n+1$.
$r_{25,i,j} \equiv (1, f_{i,j}/f_{i,j+1}^2, 0),\ 2 \leqslant i \leqslant n+1, 1 \leqslant j \leqslant n$.
$r_{26,i} \equiv (2, e_i/f_{i,n+1}, 1),\ 2 \leqslant i \leqslant n$.
$r_{27,i,j} \equiv (2, f_{i,n+j}/f_{i,n+j+1}^2, 0),\ 2 \leqslant i \leqslant n+1, 1 \leqslant j \leqslant \log(k+1)$.
$r_{28,i} \equiv (2, B_{i,\log(k+1)+1} v_i f_{i,n+\log(k+1)+1}/f_{i+1,1} p, 0),\ 2 \leqslant i \leqslant n$.
$r_{29} \equiv (2, g/f_{n+1,n+1}, 1)$.
$r_{30} \equiv (2, f_{n+1,n+\log(k+1)+1} pq/ht_1, 0)$.
$r_{31} \equiv (2, hmp/\lambda, 0)$.
$r_{32} \equiv (2, hmq/\lambda, 0)$.
$r_{33} \equiv (2, nt_1/t_2 t_3, 0)$.
$r_{34} \equiv (2, mt_2/t_4, 0)$.
$r_{35} \equiv (2, t_3/\lambda, 1)$.
$r_{36} \equiv (2, t_4/\lambda, 1)$.
$r_{37} \equiv (1, lt_4/t, 0)$.
$r_{38} \equiv (1, t_3 \mathtt{no}/t_5, 0)$.
$r_{39} \equiv (1, t t_5 \mathtt{yes}/t_6, 0)$.
$r_{40} \equiv (1, t_6/\mathtt{no}, 0)$.

— $\mathcal{E} = \Gamma - \{\mathtt{yes}, \mathtt{no}\}$.
— $i_{in} = 2$ is the input cell.
— $i_{out} = 0$, the output cell is the environment.

The design is structured in the following phases:

— **Generation phase:** The cell with label 2 is divided for n times generating 2^n cells with label 2. Note that the division rule can be applied only when the object a_i $(2 \leqslant i \leqslant n)$ appears in cell with label 2, and the object a_i will be present in each cell with label 2 at the same time. The 2^n cells with label 2 represent the 2^n subsets of set A.
— **Pre-checking phase:** When the generation phase completes, we have 2^n copies of cell with label 2, each of them encoding a subset of set A. In each cell with label 2, as

https://doi.org/10.1017/S0960129515000018 Published online by Cambridge University Press

many copies of objects p that represent the weight the subset (corresponding the cell 2) will be generated.

— **Checking phase:** In each cell with label 2, the number of copies of objects p and q are compared. All possible pairs (p, q) are removed from the cell in one RS-step, after that, if some objects p or q remain in the cell, then the cell was not encoding a solution of the problem; otherwise, the weight of the subset of A encoded in the cell equals to k, hence it encodes a solution to the problem.

— **Output phase:** The system sends the answer of the problem to the environment.

1. After the checking phase, if there exists a cell with label 2 having neither object p nor q, the object `yes` will be sent to the environment. It means the answer of the problem is positive.
2. After the checking phase, if there exists object p or q in each cell with label 2, object `no` will be sent to the environment, which means the answer of the problem is negative.

We now check the number of RS-steps taken by system $\Pi_{(\langle n,k\rangle)}$ in each phase of solving the subset sum problem.

— **Generation phase:** Initially, cell 1 contains objects $l, \texttt{yes}, \texttt{no}$, and cell 2 has objects a_1, g, m, n. The object a_1 in cell 2 corresponds to variable s_1. At step 1, rule $r_{1,1}$ is applied. By using the rule, cell 2 is divided into two cells with label 2, and object a_1 evolves to objects $a_1^{(1)}$ and $a_1^{(2)}$, which are placed in the two generated cells 2, respectively. The application of rule $r_{1,1}$ takes $e(r_{1,1})$ steps, for any time-mapping e. During the execution of rule $r_{1,1}$, there is no rule starting, so the number of RS-steps is one.

When the execution of rule $r_{1,1}$ completes, both of the rules $r_{2,1}$ and $r_{3,1}$ can be applied. Although the applications of the rules $r_{2,1}$ and $r_{3,1}$ start at the same step, they may finish at different steps due to the fact that the rules can be associated with different execution time for any time mapping e. By the application of the rule $r_{2,1}$ (resp. $r_{3,1}$) in the cell with label 2 having object $a_1^{(1)}$ (resp. $a_1^{(2)}$), cell 2 sends out object $a_1^{(1)}$ (resp. $a_1^{(2)}$) to the environment and receives objects $a_0 a_1^{(3)} B_1$ (resp. objects $a_0 a_1^{(4)}$) from the environment. After the execution of rule $r_{2,1}$ completes, object $a_1^{(3)}$ appears in cell 2. At that moment, rule $r_{4,1}$ is enabled and applied, by which object $a_1^{(3)}$ can be sent to cell 1. When the application of rule $r_{3,1}$ completes, cell 2 contains object $a_1^{(4)}$. At that moment, rule $r_{5,1}$ is applied, sending object $a_1^{(4)}$ to cell 1. The execution time of the rules $r_{2,1}$ and $r_{3,1}$ can be different for any time mapping e, so the applications of the rules may complete at different steps such that rules $r_{4,1}$ and $r_{5,1}$ are applied at different steps.

Rule $r_{6,1}$ can be applied only when the executions of both rules $r_{4,1}$ and $r_{5,1}$ have completed. At that moment, cell 1 contains both of the objects $a_1^{(3)}, a_1^{(4)}$. By using the rule $r_{6,1}$, the objects $a_1^{(3)}, a_1^{(4)}$ in cell 1 are sent out to the environment, and two copies of object a_2 enters in cell 1. The rule $r_{6,1}$ has a synchronization functioning, because $e(r_{2,1}) + e(r_{4,1})$ may not equal to $e(r_{3,1}) + e(r_{5,1})$ for any time mapping e. Anyway, when the application of rule $r_{6,1}$ completes, it takes five RS-steps, which is independent on any time mapping e.

https://doi.org/10.1017/S0960129515000018 Published online by Cambridge University Press

When the application of rule $r_{6,1}$ completes, cell 1 has two copies of object a_2 and rule $r_{7,1}$ is applied. Note that there are two cells with label 2, and each of them contains one copy of object a_0. When the execution of rule $r_{6,1}$ completes, cell 1 has two copies of object a_2. Each cell 2 will obtain one copy of object a_2 due to the fact that the rule $r_{7,1}$ is applied in a maximally parallel manner to all the cells with label 2. The application of rule $r_{7,1}$ in the cells with label 2 starts at the same step and completes at the same time, since the execution time is $e(r_{7,1})$ for any time mapping e. In this way, object a_2 appears in each cell with label 2 at the same moment. The process of using rules $r_{j,1}$ $(j = 1, 2, \ldots, 7)$ takes at most six RS-steps.

The two cells with label 2 represent the two subsets of $A = \{s_1\}$. In the two cells with label 2, one has object B_1 corresponding to the subset $\{s_1\}$; the other one has no object B_1 corresponding the empty set.

When the cells with label 2 contain object a_2, they apply the rule $r_{1,2}$ at the same moment, by which four cells with label 2 can be generated at the same time. When the application of rule $r_{1,2}$ completes, the rules $r_{2,2}$ and $r_{3,2}$ are enabled and applied. The two cells with label 2 having object $a_2^{(1)}$ (resp. $a_2^{(2)}$) use the rule $r_{2,2}$ (resp. $r_{3,2}$) and complete at the same moment. But the execution time of rules $r_{2,2}$ and $r_{3,2}$ may be different for any time mapping e. When the execution of rule $r_{2,2}$ (resp. $r_{3,2}$) completes, rule $r_{4,2}$ (resp. $r_{5,2}$) is enabled and applied in the cells with label 2. Note that the cells 2 having object $a_2^{(3)}$ (resp. $a_2^{(4)}$) apply rule $r_{4,2}$ (resp. $r_{5,2}$) at the same moment, but the applications of rules $r_{4,2}$ and $r_{5,2}$ may complete at different steps. Rule $r_{6,2}$ is applied when the executions of rules $r_{4,2}$ and $r_{5,2}$ have completed. At that moment, cell 1 contains two copies of object $a_2^{(3)}$ and object $a_2^{(4)}$, respectively. By the application of rule $r_{6,2}$, cell 1 can receive four copies of object a_3 at the same time due to the fact that the system works in a maximally parallel manner. After the execution of rule $r_{6,2}$ completes, rule $r_{7,2}$ is applied, by which each cell 2 receives one copy of object a_3. The process of the executions of rules $r_{j,2}$ $(j = 1, 2, \ldots, 7)$ takes at most six RS-steps, which is independent on any time-mapping e.

The four cells with label 2 represent the four subsets of A where $A = \{s_1, s_2\}$. In the four cells with label 2, we can check that one has both objects B_1, B_2 (corresponding the subset $\{s_1, s_2\}$ of A); one has object B_1 (corresponding the subset $\{s_1\}$); one has object B_2 (corresponding the subset $\{s_2\}$); the other one has no object corresponding the empty set.

In general, after at most $6n$ RS-steps, 2^n copies of cell 2 are generated, and each cell with label 2 has some objects of B_i. The 2^n cells with label 2 represent 2^n subsets of $A = \{s_1, s_2, \ldots, s_n\}$.

— **Pre-checking phase:** When the generation phase completes, each cell 2 has a copy of object a_{n+1}. The cells with label 2 will use rule r_8 at the same moment. Each of the cells 2 sends out object a_{n+1} to the environment and receives two copies of object b_2. When the application of rule r_8 completes, rule $r_{9,2}$ is applied for all the cells 2 at the same moment. By using the rule, in each cell 2 one copy of object b_2 is sent out to the environment and two copies of b_3 enter each cell 2. After the execution of rule r_8 completes, the application of rule $r_{9,3}$ starts, by which the cell can introduce two copies of object b_4 from the environment by sending a copy of object b_3. The rules

https://doi.org/10.1017/S0960129515000018 Published online by Cambridge University Press

$r_{9,i}$ $(2 \leqslant i \leqslant \log n)$ can be applied one by one until the object $b_{\log n+1}$ is generated. Note that there are n copies of object $b_{\log n+1}$ presented in each cell with label 2 at the same time. So, the applications of rules $r_{9,i}$ start and complete at the same moment to all the cells 2. It takes $\log n$ RS-steps during the applications of rules r_8 and $r_{9,i}$ with $2 \leqslant i \leqslant \log n$, which is independent on the time mapping e.

When the execution of rules $r_{9,i}$ have completed, each cell 2 has a copy of object $b_{\log n+1}$. For the cells 2 having object B_i, they can use rules $r_{10,i}$ $(1 \leqslant i \leqslant n)$. By the application of the rule $r_{10,i}$, objects $b_{\log n+1}B_i$ are sent out of cells 2 and objects $c_i d_{i,1} B_{i,1}$ enter in the cell from the environment for all cells 2 at the same moment (there exist n copies of object $b_{\log n+1}$ in each cell 2, which ensures that all the objects B_i in cells with label 2 can be sent out in a maximal parallel manner). The applications of rules $r_{9,i}$ to all cells with label 2 can complete at the same moment, hence rules $r_{10,i}$ are applied at the same time when cells with label 2 have object $b_{\log n+1}$. By using rule $r_{10,1}$, object B_1 can be sent to the environment and object c_1 enters in cell 2 from the environment. When the application of rule $r_{10,1}$ completes, rule r_{11} is enabled and applied, by which object c_1 is sent to cell 1. Rule $r_{12,1}$ has a synchronization functioning, which can be used only when both rules $r_{10,2}$ and r_{11} have completed. By using rule $r_{12,1}$, one copy of object c_2 in cell 2 is exchanged with one copy of object c_1 in cell 1. Rules $r_{12,i}$ $(1 \leqslant i \leqslant n-1)$ can be used one by one. The rules r_{11} and $r_{12,i}$ are used to check whether the applications of rules $r_{10,i}$ complete. If the object c_n appears in cell 1, it means that the applications of all the rules $r_{10,i}$ complete. The applications of rules $r_{11}, r_{12,i}$ take $e(r_{11}) + \Sigma_{1 \leqslant i \leqslant n-1} e(r_{12,i})$ steps for any time mapping, and it takes n RS-steps.

When cell 1 contains object c_n, the execution of rule r_{13} starts, by which each object c_n in cell 1 can involve one copy of object d_1 from the environment. For any $1 \leqslant i \leqslant n-1$, the numbers of objects c_i are equal with each other, hence the number of object d_1 is equal to the number of c_i. Cell 1 has object d_1 inside, rule $r_{14,1}$ is enabled and applied, each object $d_{1,1}$ in cell 2 is exchanged with one copy of d_1 in cell 1. If the object d_1 appears in cell 2, the application of rule $r_{15,1}$ starts, where the objects $B_{1,1}d_1$ in each cell 2 are sent to cell 1 and one copy of object e_1 and two copies of object $B_{1,2}$ are emitted to cell 2. When cells 2 have object $B_{1,2}$, rule $r_{16,1,2}$ is enabled and applied, one copy of object $B_{1,2}$ is sent out cell 2 while two copies of object $B_{1,3}$ will enter the cell. When cell 2 has object $B_{1,3}$, rule $r_{16,1,3}$ can be used. After that, rules $r_{16,1,j}$ $(2 \leqslant j \leqslant \log(k+1)-1)$ can be used in order. It takes $\log(k+1)$ RS-steps during the applications of rules $r_{14,1}$, $r_{15,1}$ and $r_{16,1,j}$ with $2 \leqslant j \leqslant \log(k+1)-1$.

If cell 2 has object $B_{1,\log(k+1)}$, rule $r_{17,1}$ is applied. In the cell, object $B_{1,\log(k+1)}$ is sent out by receiving one copy of d'_2 and two copies of $B_{1,\log(k+1)+1}$ from the environment. There are $k+1$ copies of object $B_{1,\log(k+1)+1}$ and $(k+1)/2$ copies of object d'_2 in the cell 2, which receives the object $B_{1,1}$ at some moment before. If the object d'_2 appears in cell 2, the execution of rule $r_{18,2}$ starts and object d'_2 is sent to cell 1. When object d'_2 presents in cell 1, the execution of rule $r_{19,2}$ starts. Each copy of object d'_2 in cell 1 is exchanged with one copy of object d_2 from the environment. We can check that during the applications of rules $r_{14,1}, r_{15,1}, r_{16,1,j}, r_{17,1}, r_{18,2}, r_{19,2}$, there are $\log(k+1)+3$ RS-steps.

https://doi.org/10.1017/S0960129515000018 Published online by Cambridge University Press

When object d_2 presents in cell 1, rule $r_{14,2}$ is enabled and applied. One copy of object $d_{2,1}$ in cell 2 is exchanged with one copy of object d_2 in cell 1. In the following, the applications of rules $r_{15,2}, r_{16,2,j}, r_{17,2}, r_{18,3}, r_{19,3}$ can start one by one. When the execution of rule $r_{19,3}$ completes, cell 1 has object d_3 and the application of rule $r_{14,3}$ starts. In this way, rules $r_{14,i}, r_{15,i}, r_{16,i,j}, r_{17,i}, r_{18,i}, r_{19,i}$ can be applied as many times as possible due to the fact the rules are applied in a maximal parallel manner. When the execution of rule $r_{18,n+1}$ finishes, there are $k+1$ copies of object $B_{i,\log(k+1)+1}, 1 \leqslant i \leqslant n$ in each cell 2. This process takes at most $n\log(k+1)+3n-1$ RS-steps, which is independent on time mapping e.

If the object d'_{n+1} appears in cell 1, rule r_{20} can be applied. Each object d'_{n+1} is exchanged with one copy of object $f_{1,1}$ from the environment. When cell 1 receives object $f_{1,1}$, rule r_{21} is enabled and applied, each object e_1 in cell 2 is exchanged with one copy of object $f_{1,1}$. When the application of rule r_{21} completes, the object $f_{1,1}$ appears in cell 2 and rule $r_{22,1}$ is applied. By the application of the rule, two copies of object $f_{1,2}$ enter cell 2 by sending out one copy of object $f_{1,1}$. When the execution of rule $r_{22,1}$ completes, rule $r_{22,2}$ is applied. After that, rules $r_{22,i}$ $(1 \leqslant i \leqslant \log(k+1))$ can be applied one by one, and the executions of rules $r_{22,i}$ take $\log(k+1)$ RS-steps.

If the object $f_{1,\log(k+1)+1}$ appears in cell 2, the application of rule r_{23} starts, where objects $B_{1,\log(k+1)+1}v_1 f_{1,\log(k+1)+1}$ in cell 2 are exchanged with the objects $f_{2,1}p$ from the environment. When the object $f_{2,1}$ presents in cell 2, the execution of rule $r_{24,2}$ starts, object $f_{2,1}$ is sent into cell with label 1. When cell 1 has object $f_{2,1}$ inside, rule $r_{25,2,1}$ is enabled and applied, one copy of object $f_{2,1}$ in cell 1 is exchanged with two copies of object $f_{2,2}$ from the environment. Rules $r_{25,2,j}$ $(1 \leqslant j \leqslant n)$ can be used one by one, which takes n RS-steps.

If the object $f_{2,n+1}$ appears in cell 1, the application of rule $r_{26,2}$ starts. Object e_2 in cell 2 is sent to cell 1 and object $f_{2,n+1}$ is sent to cell 2. After the application of rule $r_{26,2}$, rule $r_{27,2,1}$ can be applied. By using $r_{27,2,1}$, one copy of object $f_{2,n+1}$ in cell 2 is exchanged with two copies of object $f_{2,n+2}$ from the environment. When the execution of rule $r_{27,2,1}$ completes, the application of rule $r_{27,2,2}$ starts. The rules $r_{27,2,j}$ $(1 \leqslant j \leqslant \log(k+1))$ can be used in order, and the applications of rules $r_{27,2,j}$ take $\log(k+1)$ RS-steps, for any time mapping. When the object $f_{2,n+\log(k+1)+1}$ appears in cell 2, the application of rule $r_{28,2}$ starts, objects $B_{2,\log(k+1)+1}v_2 f_{2,n+\log(k+1)+1}$ in cell 2 are exchanged with the objects $f_{3,1}p$ from the environment. We can check that the applications of rules $r_{24,2}, r_{25,2,j}, r_{26,2}, r_{27,2,j}, r_{28,2}$ take $n+\log(k+1)+3$ RS-steps.

If the object $f_{3,1}$ appears in cell 2, rule $r_{24,3}$ is enabled and applied, where object $f_{3,1}$ is sent into cell 1. When the execution of rule $r_{24,3}$ completes, rules $r_{25,3,j}, r_{26,3}, r_{27,3,j}, r_{28,3}$ are used in order. Rules $r_{24,i}, r_{25,i,j}, r_{26,i}, r_{27,i,j}, r_{28,i}$ can be applied as many times as possible. When the execution of rule $r_{28,n}$ completes, in each cell 2, as many copies of object p as the weight of the corresponding subset will be generated. During the applications of rules $r_{24,i}, r_{25,i,j}, r_{26,i}, r_{27,i,j}, r_{28,i}$ take $(n-1)(n+\log(k+1)+3)$ RS-steps for any time mapping e.

If the object $f_{n+1,1}$ appears in cell 2, the application of rule $r_{24,n+1}$ starts, object $f_{n+1,1}$ is sent into cell 1. When object $f_{n+1,1}$ appears in cell 1, rules $r_{25,n+1,j}$ $(1 \leqslant j \leqslant n)$ will be used in order, and the applications of rules $r_{24,n+1}, r_{25,n+1,j}$ take $n+1$ RS-steps.

https://doi.org/10.1017/S0960129515000018 Published online by Cambridge University Press

If the object $f_{n+1,n+1}$ presents in cell 1, rule r_{29} is enabled and used. One copy of object g in cell 2 is exchanged with one copy of object $f_{n+1,n+1}$ in cell 1. When the object $f_{n+1,n+1}$ presents in cell 2, rules $r_{27,n+1,j}$ $(1 \leqslant j \leqslant \log(k+1))$ will be applied in order. The process takes $\log(k+1)+1$ RS-steps during the applications of rules $r_{29}, r_{27,n+1,j}$.

In general, we can check that the pre-checking phase takes $n^2+7n+\log n+(2n+1)\log(k+1)+3$ RS-steps.

— **Checking phase:** When the object $f_{n+1,n+\log(k+1)+1}$ appears in cell 2, the application of rule r_{30} starts, the objects $f_{n+1,n+\log(k+1)+1}pq$ in cell 2 are exchanged with objects ht_1 from the environment. For a given time-mapping e, it takes one RS-step during the execution of rule r_{30}.

— **Output phase:** When the execution of rule r_{30} completes, we have the following two cases: if the weight of the subset associated with a cell is equal to k, then no object p or q remains in this cell; if the weight of the subset associated with a cell is not equal to k, then at least one object p or q will remain in the cell.

- There is a cell with label 2 without object p nor q. When the execution of rule r_{30} finishes, the application of rule r_{33} starts, the objects n and t_1 in cell 2 are exchanged with objects t_2 and t_3 from the environment. When the objects t_2 and t_3 appear in cell 2, the applications of rule r_{34} and r_{35} start at the same step. By using rule r_{34}, the objects mt_2 in cell 2 are exchanged with the object t_4 from the environment, and by applying the rule r_{35}, the object t_3 in cell 2 is sent into cell 1. If the object t_3 appears in cell 1, rule r_{38} is enabled and used, the objects t_3`no` in cell 1 are exchanged with the object t_5 from the environment. If the object t_4 presents in cell 2, the application of rule r_{36} starts, object t_4 is sent into cell 1. When the execution of rule r_{36} completes, the execution of rule r_{37} starts, where the objects lt_4 in cell 1 are exchanged with the object t from the environment (only one copy of object t is sent into cell 1, because there is only one copy of object l in cell 1). Rule r_{39} can be applied only when the executions of rules r_{37}, r_{38} finish. By the application of rule r_{39}, object `yes` is sent to the environment, and object t_6 is sent into cell 1. When the execution of rule r_{39} finishes, rule r_{40} is enabled and applied, object `no` is sent back to cell 1. In this case, when the computation halts, one copy of object `yes` appears in the environment. In total, if the weight of the subset associated with a cell is equal to k, the output phase takes seven RS-steps.

- There exists an object p or q in each cell with label 2. In this case, when the execution of rule r_{30} finishes, the applications of rule r_{31} (or r_{32}) and rule r_{33} start at the same step. By using the rule r_{31} or r_{32}, the object m in each cell 2 is sent to the environment, then rules $r_{34}, r_{36}, r_{37}, r_{39}, r_{40}$ cannot be applied. By applying the rule r_{33}, objects t_2 and t_3 will be sent into cell 2. When the execution of rule r_{33} finishes, rule r_{35} is enabled and used, object t_3 is sent into cell 1. If object t_3 appears in cell 1, the application of rule r_{38} starts, object `no` is sent to the environment. In this case, when the computation halts, one copy of `no` appears in the environment. In total, if the weight of the subset in any cell 2 is not equal to k, the output phase takes three RS-steps.

https://doi.org/10.1017/S0960129515000018 Published online by Cambridge University Press

According to the analysis above, we can check that for any time-mapping $e : \mathcal{R} \to \mathbb{N}$, the object yes appears in the environment when the computation halts if and only if there exists a subset $B \subseteq A$ such that $w(B) = k$; and the object no appears in the environment when the computation halts if and only if there does not exist a subset $B \subseteq A$ such that $w(B) = k$. So, the system $\Pi_{(\langle n,k\rangle)}$ is time-free sound and time-free complete.

For any time-mapping $e : \mathcal{R} \to \mathbb{N}$, if there exists a subset $B \subseteq A$ such that $w(B) = k$, the computation takes at most $n^2 + 13n + (2n + 1)\log(k + 1) + \log n + 11$ RS-steps, and the system halts; if there does not exist a subset $B \subseteq A$ such that $w(B) = k$, the computation takes at most $n^2 + 13n + (2n + 1)\log(k + 1) + \log n + 7$ RS-steps, and the system halts.

Therefore, the family of tissue P systems with cell division is time-free polynomially bounded.

The family $\mathbf{\Pi} = \{\Pi_{(g\langle n,k\rangle)} \mid n, k \in \mathbb{N}\}$ is polynomially uniform by Turing machines, because the rules of a system $\Pi_{(g\langle n,k\rangle)}$ of the family are recursively from the values n and k. Besides, the necessary resources for defining each system are of polynomial order.

- Size of the alphabet: $n^2 + 13n + (2n + 1)\log(k + 1) + \log n + 18$.
- Initial number of cells: 2.
- Initial number of objects: 7.
- Number of rules: $n^2 + 15n + (2n + 1)\log(k + 1) + \log n + 13$.
- Maximal length of a rule: 5.

Therefore, subset sum problem can be solved in polynomial RS-steps by a family of recognizer tissue P systems with cell division in a time-free uniform manner, this concludes the proof. □

4. Conclusions and future work

In this work, we investigate the computational efficiency of tissue P systems with cell division in the context of time-freeness. Specifically, we give a time-free solution to subset sum problem by a uniform family of tissue P systems with cell division, where the correctness of the solution to subset sum problem does not depend on the execution time of the involved rules.

The tissue P systems with cell division constructed in this work have arbitrarily large amount of objects in the environment. It remains open whether subset sum problem or other **NP**-complete problems can be solved by tissue P systems with cell division without an arbitrarily large amount of objects in the environment in a time-free manner.

The tissue P systems constructed in the proof of Theorem 3.1 have the maximum length of communication rules 5. It is of interest to optimize the maximum length of communication rules without loss of the computational efficiency of tissue P systems with cell division.

Tissue P systems with cell separation are a new variant of tissue P systems (Pan and Pérez-Jiménez 2010). In such P systems, cells do not have the duplication function; that

https://doi.org/10.1017/S0960129515000018 Published online by Cambridge University Press

is, when a cell is separated, the objects in the cell are divided and placed in the newly generated cells instead of copying the objects and then placing one copy in each of the newly generated cells. It remains open how we can construct tissue P systems with cell separation to time-freely solve **NP**-complete problems.

Acknowledgements

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

References

Alhazov, A., Freund, R. and Oswald, M. (2005) Tissue P systems with antiport rules and small numbers of symbols and cells. In: Developments in Language Theory. *Lecture Notes in Computer Science* **3572** 100–111.

Alhazov, A., Freund, R., Leporati, A., Oswald, M. and Zandron C. (2006) (Tissue) P systems with unit rules and energy assigned to membranes. *Fundamenta Informaticae* **74** (4) 391–408.

Bernardini, F. and Gheorghe, M. (2005) Cell communication in tissue P systems: Universality results. *Soft Computing* **9** (9) 640–649.

Cavaliere, M. and Sburlan, D. (2005) Time–independent P systems. In: Membrane Computing. *Lecture Notes in Computer Science* **3365** 239–258.

Díaz-Pernil, D., Pérez-Jiménez, M. J., Riscos-Núñez, A. and Romero-Jiménez, Á. (2008a) Computational efficiency of cellular division in tissue-like membrane systems. *Romanian Journal of Information Science and Technology* **11** (3) 229–241.

Díaz-Pernil, D., Gutiérrez-Naranjo, M. A., Pérez-Jiménez, M. J. and Riscos-Núñez, A. (2007a) Solving subset sum in linear time by using tissue P systems with cell division. In: Bio-inspired Modeling of Cognitive Tasks. *Lecture Notes in Computer Science* **4527** 170–179.

Díaz-Pernil, D., Gutiérrez-Naranjo, M. A., Pérez-Jiménez, M. J. and Riscos-Núñez, A. (2007b) A linear–time tissue P system based solution for the 3-coloring problem. *Electronic Notes in Theoretical Computer Science* **171** (2) 81–93.

Díaz-Pernil, D., Gutiérrez-Naranjo, M. A., Pérez-Jiménez, M. J. and Riscos-Núñez, A. (2008b) A uniform family of tissue P systems with cell division solving 3-COL in a linear time. *Theoretical Computer Science* **404** (1) 76–87.

Díaz-Pernil, D., Gutiérrez-Naranjo, M. A., Pérez-Jiménez, M. J. and Riscos-Núñez, A. (2009) Solving the independent set problem by using tissue-like P systems with cell division. In: Methods and Models in Artificial and Natural Computation. A Homage to Professor Mira's Scientific Legacy. *Lecture Notes in Computer Science* **5601** 213–222.

Freund, R., Păun, G. and Pérez-Jiménez, M. J. (2005) Tissue P systems with channel states. *Theoretical Computer Science* **330** (1) 101–116.

Gheorghe, M., Păun, G., Pérez-Jiménez, M. J. and Rozenberg, G. (2013) Research frontiers of membrane computing: Open problems and research topics. *International Journal of Foundations of Computer Science* **24** (5) 547–623.

Ionescu, M., Păun, G. and Yokomori, T. (2006) Spiking neural P systems. *Fundamenta informaticae* **71** (2) 279–308.

https://doi.org/10.1017/S0960129515000018 Published online by Cambridge University Press

Martín-Vide, C., Pazos, J., Păun, G. and Rodríguez-Patón, A. (2002) A new class of symbolic abstract neural nets: Tissue P systems. In: Computing and Combinatorics. *Springer-Verlag Lecture Notes in Computer Science* **2387** 290–299.

Martín-Vide, C., Pazos, J., Păun, G. and Rodríguez-Patón, A. (2003) Tissue P systems. *Theoretical Computer Science* **296** (2) 295–326.

Pan, L. and Pérez-Jiménez, M. J. (2010) Computational complexity of tissue-like P systems. *Journal of Complexity* **26** (3) 296–315.

Pan, L., Zeng, X. and Zhang, X. (2011) Time-free spiking neural P systems. *Neural Computation* **23** (5) 1320–1342.

Păun, A. and Păun, G. (2002) The power of communication: P systems with symport/antiport. *New Generation Computing* **20** (3) 295–305.

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

Păun, G. (2002) *Membrane Computing: An Introduction*, Springer-Verlag.

Păun, G., Pérez-Jiménez, M. J. and Riscos-Núñez, A. (2008) Tissue P systems with cell division. *International Journal of Computers, Communications and Control* **3** (3) 295–303.

Păun, G., Rozenberg, G. and Salomaa, A. (eds.) (2010) *Handbook of Membrane Computing*, Oxford University Press.

Pérez-Jiménez, M. J., Romero-Jiménez, A. and Sancho-Caparrini, F. (2006) A polynomial complexity class in P systems using membrane division. *Journal of Automata, Languages and Combinatorics* **11** (4) 423–434.

Salomaa A. and Rozenberg, G. (eds.) (1997) *Handbook of Formal Languages*, Springer-Verlag, Berlin.

Song, T., Macías-Ramos, L. F., Pan, L. and Pérez-Jiménez, M. J. (2014a) Time-free solution to sat problem using P systems with active membranes. *Theoretical Computer Science* **529** 61–68.

Song, T., Luo, L., He, J., Chen, Z. and Zhang, K. (2014b) Solving subset sum problems by time-free spiking neural P systems. *Applied Mathematics and Information Sciences* **8** (1) 327–332.

https://doi.org/10.1017/S0960129515000018 Published online by Cambridge University Press