A Linear Solution for Subset Sum Problem with Tissue P Systems with Cell Division

Daniel Díaz-Pernil, Miguel A. Gutiérrez-Naranjo,
Mario J. Pérez-Jiménez, Agustín Riscos-Núñez

Research Group on Natural Computing
University of Sevilla
Avda Reina Mercedes s/n, 41012 Sevilla, Spain
{sbdani,magutier,marper,ariscosn}@us.es

Summary. Tissue P systems are a computing model in the framework of Membrane Computing where the tree-like membrane structure is replaced by a general graph. Recently, it has been shown that endowing these P systems with cell division, **NP**-complete problems can be solved in polynomial time. In this paper we present a solution to the Subset Sum problem via a family of such devices, and we also include the formal verification of such solution. This is the first solution to a numerical **NP**-complete problem by using tissue P systems with cell division.

1 Introduction

Membrane Computing is a bio-inspired computing model based on the assumption that the processes taking place in the compartmental structure of a living cell can be interpreted as computations. The devices of this model are generically called *P Systems*.

In the initial definition of the cell-like model of P systems [6], membranes are hierarchically arranged in a tree-like structure. Its biological inspiration comes from the morphology of cells, where small vesicles are surrounded by larger ones. This biological structure can be abstracted into a tree-like graph, where the root represents the skin of the cell (i.e. the outermost membrane) and the leaves represent membranes that do not contain any other membrane (elementary membranes). Besides, two nodes in the graph are connected if they represent two membranes such that one of them contains the other one.

Recently, new models of P systems have been explored. One of them is the model of *tissue P systems* where the tree-like membrane structure is replaced by a general graph. This model has two biological inspirations (see [3, 4]): intercellular communication and cooperation between neurons. The common mathematical model of these two mechanisms is a net of processors dealing with symbols and

communicating these symbols along channels specified in advance. The communication among cells is based on symport/antiport rules, which were introduced as communication rules for P systems in [5]. In symport rules, objects cooperate to traverse a membrane together in the same direction, whereas in the case of antiport rules, objects residing at both sides of the membrane cross it simultaneously but in opposite directions.

This paper is devoted to the study of the computational efficiency of tissue P systems with cell division. In literature, different models of cell-like P systems have been successfully used in order to design efficient solutions to **NP**-complete problems (see, for example, [2] and the references therein). These solutions are obtained by generating an exponential amount of workspace in polynomial time and using parallelism to check simultaneously all the candidate solutions.

From the seminal definition of tissue P systems [3, 4], several research lines have been developed and other variants have arisen (see [1] and references therein). One of the most interesting variants of tissue P systems was presented in [8], where the definition of tissue P systems is combined with the one of P systems with active membranes, yielding *tissue P systems with cell division*. The biological inspiration is clear: alive tissues are not *static* networks of cells, since cells are duplicated via mitosis in a natural way. One of the main features of such tissue P systems with cell division is related to their computational efficiency. In [8], a polynomial-time solution to the **NP**-complete problem SAT is shown, and in [1] a linear-time solution for the 3-COL problem was presented. In this paper we go on with the research in this model and present a linear-time solution to another well-known numerical **NP**-complete problem: the Subset Sum problem.

The paper is organized as follows: first we recall some preliminary concepts and the definition of tissue P systems with cell division. Next, recognizing tissue P systems are briefly described. A linear–time solution to the Subset Sum problem is presented in the following section, including a short overview of the computation and the formal verification of the solution. Finally, some conclusions and lines for future research are presented.

2 Preliminaries

In this section we briefly recall some of the concepts used later on in the paper.

An *alphabet*, Σ, is a non empty set, whose elements are called *symbols*. An ordered sequence of symbols is a *string*. 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 strings of length n built with symbols from the alphabet Σ is denoted by Σ^n and $\Sigma^* = \cup_{n\geq 0}\Sigma^n$. A *language* over Σ is a subset from 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 $supp(m) = \{x \in A \,|\, f(x) > 0\}$ and its *size* is defined as $\sum_{x\in A} f(x)$. A multiset is empty (resp. finite) if its support is the empty set (resp. finite).

A finite multiset $m = (A, f)$ will be denoted as $m = \{\{x_1^{f(x_1)}, \ldots, x_k^{f(x_k)}\}\}$, where $supp(m) = \{x_1, \ldots, x_k\}$, or alternatively as the string $x_1^{f(x_1)} \cdots x_k^{f(x_k)}$. The union of multisets will be denoted as concatenation when using the string notation.

In what follows we assume the reader is already familiar with the basic notions and the terminology underlying P systems. For details, see the handbook [7].

3 Tissue P Systems with Cell Division

In the first definition of the model of tissue P systems [3, 4] the membrane structure did not change along the computation. The main features of tissue P systems with cell division, from the computational point of view, are that cells obtained by division have the same labels as the original cell, and if a cell is divided, then its interaction with other cells or with the environment is blocked during the mitosis process. In some sense, this means that while a cell is dividing it closes all its communication channels. This features imply that the underlying graph is dynamic, as nodes can be added during the computation by division and the edges can be deleted/re-established for dividing cells.

Actually, the underlying graph of connections between cells will not be handled explicitly: the initial structure is implicitly given by the number of initial cells (nodes) and the communication rules (marking edges that connect nodes); the opening/closing edges will be controlled by the semantics.

Formally, a *tissue P system with cell division* of initial degree $q \geq 1$ is a tuple of the form $\Pi = (\Gamma, w_1, \ldots, w_q, \mathcal{E}, \mathcal{R}, i_0)$, where:

1. Γ is a finite *alphabet*, whose symbols will be called *objects*.
2. $w_1, \ldots, w_q$ are strings over Γ.
3. $\mathcal{E} \subseteq \Gamma$.
4. $\mathcal{R}$ is a finite set of rules of the following form:
   (a) *Communication rules*: $(i, u/v, j)$, for $i, j \in \{0, 1, \ldots, q\}, i \neq j$, $u, v \in \Gamma^*$.
   (b) *Division rules*: $[a]_i \rightarrow [b]_i[c]_i$, where $i \in \{1, 2, \ldots, q\}$ and $a, b, c \in \Gamma$.
5. $i_0 \in \{0, 1, 2, \ldots, q\}$.

A tissue P system with cell division of degree $q \geq 1$ can be seen as a set of q cells labelled by $1, 2, \ldots, q$. We shall use 0 as the label of the environment, and i_0 for the output region (which can be the region inside a cell or the environment). As we said before, the underlying graph expressing connections between cells is implicit, being determined by the communication rules: the nodes are the cells and the edges indicate if it is possible for pairs of cells to communicate directly. The communication rule $(i, u/v, j)$ can be applied over two cells i and j such that u is contained in cell i and v is contained in cell j, and neither i nor j are being divided. The application of this rule means that the objects of the multisets represented by u and v are interchanged between the two cells.

The strings $w_1, \ldots, w_q$ describe the multisets of objects placed initially in the q cells of the system. We interpret that $\mathcal{E} \subseteq \Gamma$ is the set of objects placed in the environment, each one of them in an arbitrarily large amount of copies.

The division rule $[a]_i \rightarrow [b]_i[c]_i$ can be applied over a cell i containing object a. The application of this rule divides this cell into two new cells with the same label. All the objects in the original cell are replicated and copied in each of the new cells, with the exception of the object a, which is replaced by the object b in the first new cell and by c in the second one. Since both new cells keep the same label as their father cell, they keep the same connections too. There is no connection between both new cells.

Rules are used as usual in the framework of membrane computing, that is, in a maximally parallel way (a universal clock is considered). In one step, each object in a cell can only be used for one rule (non-deterministically chosen when there are several possibilities), but any object which can participate in a rule of any form must do it, i.e, in each step we apply a maximal set of rules. 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 in that step; the objects inside that cell do not move in that step.

4 Recognizing Tissue P Systems with Cell Division

NP-completeness has been usually studied in the framework of *decision problems*. Let us recall that a decision problem is a pair (I_X, θ_X) where I_X is a language over a finite alphabet (whose elements are called *instances*) and θ_X is a total boolean function over I_X.

In order to study the computational efficiency, a special class of tissue P systems is introduced in [8]: *recognizing*[1] *tissue P systems*.

A recognizing tissue P system with cell division of degree $q \geq 1$ is a tuple $\Pi = (\Gamma, \Sigma, w_1, \ldots, w_q, \mathcal{E}, \mathcal{R}, i_{in}, i_0)$, where

- $(\Gamma, w_1, \ldots, w_q, \mathcal{E}, \mathcal{R}, i_0)$ is a tissue P system with cell division of degree $q \geq 1$ (as defined in the previous section).
- The working alphabet Γ has two distinguished objects `yes` and `no`, present in at least one copy in an initial multiset $w_1, \ldots, w_q$, but not present in $\mathcal{E}$.
- Σ is an (input) alphabet strictly contained in Γ.
- $i_{in} \in \{1, \ldots, q\}$ is the input cell.
- The output region i_0 is the environment.
- All computations halt.
- If $\mathcal{C}$ is a computation of Π, then either the object `yes` or the object `no` (but not both) must have been released into the environment, and only in the last step of the computation.

The computations of the system Π with input $w \in \Gamma^*$ start from a configuration of the form $(w_1, w_2, \ldots, w_{i_{in}} w, \ldots, w_q; \mathcal{E})$, that is, after adding the multiset w to the contents of the input cell i_{in}. We say that the multiset w is *recognized* by Π if and only if the object `yes` is sent to the environment, in the last step

[1] In [8] they were called *recognizer* tissue P systems.

of all its associated computations. We say that $\mathcal{C}$ is an accepting (resp. rejecting) computation if the object `yes` (resp. `no`) appears in the environment associated with the corresponding halting configuration of $\mathcal{C}$.

Definition 1. *We say that a decision problem $X = (I_X, \theta_X)$ is solvable in polynomial time by a family $\mathbf{\Pi} = \{\Pi(n) : n \in \mathbb{N}\}$ of recognizing tissue P systems with cell division 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 $\Pi(n)$ 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* polynomially bounded *with regard to (X, cod, s), that is, there exists a polynomial function p, such that for each $u \in I_X$ every computation of $\Pi(s(u))$ with input $cod(u)$ is halting and, moreover, it performs at most $p(|u|)$ steps;*
  - *the family $\mathbf{\Pi}$ is* sound *with regard to (X, cod, s), that is, for each $u \in I_X$, if there exists an accepting computation of $\Pi(s(u))$ with input $cod(u)$, then $\theta_X(u) = 1$;*
  - *the family $\mathbf{\Pi}$ is* complete *with regard to (X, cod, s), that is, for each $u \in I_X$, if $\theta_X(u) = 1$, then every computation of $\Pi(s(u))$ with input $cod(u)$ is an accepting one.*

In the above definition we have imposed to every tissue P system $\Pi(n)$ a *confluent* condition, in the following sense: every computation of a system with the *same* input multiset must always give the *same* answer. The pair of functions (cod, s) are called a *polynomial encoding* of the problem in the family of P systems.

We denote by $\mathbf{PMC}_{TD}$ the set of all decision problems which can be solved by means of recognizing tissue P systems with cell division in polynomial time.

5 A Solution for the Subset Sum Problem

The Subset Sum problem is the following one: *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 $w(B) = k$.*

Next, we shall prove that the Subset Sum problem can be solved in a linear time by a family of recognizing tissue P systems with cell division. We shall address the resolution via a brute force algorithm.

We shall use a tuple $(n, (w_1, \ldots, w_n), k)$ to represent an instance of the problem, where n stands for the size of $A = \{a_1, \ldots, a_n\}$, $w_i = w(a_i)$, and k is the constant given as input for the problem.

Theorem 1. `Subset Sum` $\in \mathbf{PMC}_{TD}$.

Proof. Let $A = \{a_1, \ldots, a_n\}$ be a finite set, $w : A \to \mathbb{N}$ a weight function, and $k \in \mathbb{N}$. Let $g : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ be a function defined by

$$g(n,k) = \frac{(n+k)(n+k+1)}{2} + n$$

This function is primitive recursive and bijective between $\mathbb{N} \times \mathbb{N}$ and $\mathbb{N}$ and computable in polynomial time. Let us denote by $u = (n, (w_1, \ldots, w_n), k)$, where $w_i = w(a_i)$, $1 \leq i \leq n$, the given instance of the problem. We define the polynomially computable function $s(u) = g(n,k)$.

We shall provide a 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 $cod(u) = \{\{v_i^{w_i} : 1 \leq i \leq n\}\} \cup \{\{q^k\}\}$.

Next, we shall provide a family $\mathbf{\Pi} = \{\Pi(g(n,k)) : n, k \in \mathbb{N}\}$ of recognizing tissue P systems with cell division which solve the Subset Sum problem in a linear time. For each $(n,k) \in \mathbb{N} \times \mathbb{N}$ we shall consider the system $\Pi(g(n,k)) = (\Gamma, \Sigma, \omega_1, \omega_2, \mathcal{R}, \mathcal{E}, i_{in}, i_0)$, where

- $\Gamma = \Sigma(n) \cup \{A_i, B_i : 1 \leq i \leq n\}$
  $\cup \{z_i : 1 \leq i \leq n + \lceil \log n \rceil + \lceil \log(k+1) \rceil + 11\}$
  $\cup \{c_i : 1 \leq i \leq n+1\}$
  $\cup \{d_i : 1 \leq i \leq \lceil \log n \rceil + \lceil \log(k+1) \rceil + 4\}$
  $\cup \{e_i : 1 \leq i \leq \lceil \log n \rceil + 1\}$
  $\cup \{B_{ij} : 1 \leq i \leq n \wedge 1 \leq j \leq \lceil \log(k+1) \rceil + 1\}$
  $\cup \{b, f_1, g_1, g_2, p, D, T, S, N, \mathtt{yes}, \mathtt{no}\}$
- $\Sigma = \{q\} \cup \{v_i : 1 \leq i \leq n\}$
- $\omega_1 = z_1\, b\, c_1\, \mathtt{yes}\, \mathtt{no}$
- $\omega_2 = DA_1 \cdots A_n$
- $\mathcal{R}$ is the following set of rules:
  1. *Division rules:*
     $r_{1,i} \equiv [A_i]_2 \to [B_i]_2[\lambda]_2$ for $i = 1, \ldots, n$
  2. *Communication rules:*
     $r_{2,i} \equiv (1, z_i / z_{i+1}, 0)$ for $i = 1, \ldots, n + \lceil \log n \rceil + \lceil \log(k+1) \rceil + 10$
     $r_{3,i} \equiv (1, c_i / c_{i+1}^2, 0)$ for $i = 1, \ldots, n$
     $r_4 \equiv (1, c_{n+1} / D, 2)$
     $r_5 \equiv (2, c_{n+1} / d_1 e_1, 0)$
     $r_{6,i} \equiv (2, e_i / e_{i+1}^2, 0)$ for $i = 1, \ldots, \lceil \log n \rceil$
     $r_{7,i} \equiv (2, d_i / d_{i+1}, 0)$ for $i = 1, \ldots, \lceil \log n \rceil + \lceil \log(k+1) \rceil + 3$
     $r_{8,i} \equiv (2, e_{\lceil \log n \rceil + 1} B_i / B_{i1}, 0)$ for $i = 1, \ldots, n$
     $r_{9,i,j} \equiv (2, B_{ij} / B_{ij+1}^2, 0)$ for $i = 1, \ldots, n$, $j = 1, \ldots, \lceil \log(k+1) \rceil$
     $r_{10,i} \equiv (2, B_{i \lceil \log(k+1) \rceil + 1} v_i / p, 0)$ for $i = 1, \ldots, n$
     $r_{11} \equiv (2, pq / \lambda, 0)$
     $r_{12} \equiv (2, d_{\lceil \log n \rceil + \lceil \log(k+1) \rceil + 4} / g_1 f_1, 0)$

$r_{13} \equiv (2, f_1 p/\lambda, 0)$
$r_{14} \equiv (2, f_1 q/\lambda, 0)$
$r_{15} \equiv (2, g_1/g_2, 0)$
$r_{16} \equiv (2, g_2 f_1/T, 0)$
$r_{17} \equiv (2, T/\lambda, 1)$
$r_{18} \equiv (1, bT/S, 0)$
$r_{19} \equiv (1, S\texttt{yes}/\lambda, 0)$
$r_{20} \equiv (1, z_{n+\lceil \log n \rceil + \lceil \log(k+1) \rceil + 11} b/N, 0)$
$r_{21} \equiv (1, N\texttt{no}/\lambda, 0)$

- $\mathcal{E} = \Gamma - \{\texttt{yes}, \texttt{no}\}$
- $i_{in} = 2$, is the input cell
- $i_0 = env$, is the output cell

The design is structured in the following stages:

- *Generation Stage*: The initial cell labelled by 2 is divided into two new cells; and the divisions are iterated n times until a cell has been produced for each possible candidate solution. Simultaneously to this process, two counters (c_i and z_i) evolve in the cell labelled by 1: the first one controls the step in which the communication between cells 2 and cell 1 starts and the second one will be useful in the output stage.
- *Pre–checking Stage*: When this stage starts, we have 2^n cells labelled by 2, each of them encoding a subset of the set A. In each such a cell, as many objects p as the weight of the corresponding subset will be produced. Recall that there are k copies of the object q in every cell labelled by 2 (since they were introduced as part of the input multiset).
- *Checking Stage*: In each cell labelled by 2, the number of copies of objects p and q are compared. The way to do that is removing from the cell in one step all possible pairs (p, q). After doing so, 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 on the cell equals to k and hence it encodes a solution to the problem.
- *Output Stage*: The system sends to the environment the right answer according to the results of the previous stage:
  - *Answer yes:* After the checking stage, there is a cell labelled by 2 without objects p nor q. In this case, such a cell sends an object T to the cell 1. This object T causes the cell 1 to expel an object `yes` to the environment (see rules $r_{17} - r_{19}$).
  - *Answer no:* Every cell labelled by 2 contains some objects p or q. In this case, no object T arrives to the cell labelled by 1 and an object `no` is sent to the environment.

The proof will be concluded in Subsection 5.2. Before going on, let us informally present an overview of the computation.

5.1 An overview of the computation

First of all, we recall the polynomial encoding of the Subset Sum problem in the family $\mathbf{\Pi}$ constructed above. Let $u = (n, (w_1, \ldots, w_n), k)$ be an instance of the problem, $s(u) = g(n, k)$ and $cod(u) = \{\{v_i^{w_i} : 1 \leq i \leq n\}\} \cup \{\{q^k\}\}$.

Next, we describe informally how the recognizing tissue P system with cell division $\Pi(s(u))$ with input $cod(u)$ works. Let us start with the *generation stage*. Recall that if a division rule is triggered in a cell, then communication rules cannot be simultaneously applied to the contents of such cell. In this stage we have two parallel processes:

- On the one hand, in the cell labelled by 1 we have two counters: z_i, which will be used in the answer stage, and c_i, which will be multiplied until getting 2^n copies in exactly n steps.
- On the other hand, in the cells labelled by 2, the division rules are applied. For each object A_i (which codifies a member of the set A) we obtain two cells labelled by 2: one of them has an element B_i and the other does not.

When all divisions have been done, after n steps, we shall have 2^n cells with label 2 and each of them will contain the encoding of a subset of A. At this moment, the generation stage ends and the pre-checking stage begins.

For each cell 2, an object D is changed by a copy of the counter c_i. In this way, 2^n copies of D will appear in the cell 1, and in each cell labelled by 2 there will be an object c_{n+1}. The occurrence of such object c_{n+1} in the cells 2 will produce the apparition of two counters:

(a) The counter d_i lets the checking stage start, since it produces the apparition of the objects g_1 and f_1 after $\lceil \log n \rceil + \lceil \log(k+1) \rceil + 4$ steps.
(b) The counter e_i will be multiplied until obtaining $2^{\lceil \log n \rceil}$ copies, ensuring that at least n copies of $e_{\lceil \log n \rceil + 1}$ will be available in the step $n + \lceil \log n \rceil + 2$. Then, we trade objects $e_{\lceil \log n \rceil + 1}$ and B_i against B_{i1} for each element A_i in the subset associated with the cell.
After that, for each $1 \leq i \leq n$ we get $2^{\lceil \log(k+1) \rceil}$ copies of $B_{i\lceil \log(k+1) \rceil + 1}$, ensuring that at least $k+1$ copies will be available. Then for each element A_i in the subset associated with the cell we get $\min\{2^{\lceil \log(k+1) \rceil}, w(a_i)\}$ copies of object p, in the step $n + \lceil \log n \rceil + \lceil \log(k+1) \rceil + 5$.

The checking takes place in the step $n + \lceil \log n \rceil + \lceil \log(k+1) \rceil + 6$, when all pairs of objects p and q present in any cell labelled by 2 are sent to the environment. In this way, if the weight of the subset associated with a cell is equal to k, then no object p or q remains in this cell in the next step. Otherwise, if the encoding is not exactly of weight k, then at least one object p or q will remain in the cell. In the next step the answer stage starts. Two cases must be considered for each cell:

- If no object p or q remain in the cell, the object f_1 does not evolve, g_1 evolves to g_2, and in the step $n + \lceil \log n \rceil + \lceil \log(k+1) \rceil + 8$ the objects f_1 and g_2 are traded against T from the environment. In the next step T is sent to the cell 1,

and in the step $n + \lceil \log n \rceil + \lceil \log(k+1) \rceil + 10$, the objects T and b are sent to the environment traded by S. Finally, in the step $n + \lceil \log n \rceil + \lceil \log(k+1) \rceil + 11$ the objects S and `yes` are sent to the environment.

- If any object p or q remains in the cell, such object is sent to the environment together with the object f_1. This causes that the object b still remains in the cell 2 after the step $n + \lceil \log n \rceil + \lceil \log(k+1) \rceil + 10$. In this way, the objects b and $z_{n+\lceil \log n \rceil + \lceil \log(k+1) \rceil + 11}$ are traded by the object N with the environment, and in the step $n + \lceil \log n \rceil + \lceil \log(k+1) \rceil + 12$ the objects N and `no` are sent to the environment.

5.2 Verification

Next, we prove that the family of recognizing tissue P systems with cell division described above solves the Subset Sum problem in a linear time, according to Definition 1.

Before going on, let us remark that the defined family is *consistent*, i.e., all systems of the family are recognizing tissue P systems with cell division. By construction (type of rules and working alphabet) we can check that it is a family of tissue P systems with cell division. Moreover, we shall prove next that all computations of all systems in the family always halt and in the last step of computations either an object `yes` or `no` is sent to the environment.

Polynomial uniformity of the family

Next, we show that the family $\mathbf{\Pi} = \{\Pi(g(n,k)) \ : \ n,k \in \mathbb{N}\}$ defined in Theorem 1 is polynomially uniform by Turing machines. To this aim we are going to show that it is possible to build $\Pi(g(n,k))$ in polynomial time with respect to the size of u.

It is easy to check that the rules of a system $\Pi(g(n,k))$, with $n,k \in \mathbb{N}$ of the family are defined recursively from the values n and k. Besides, the necessary resources to build an element of the family are of polynomial order with respect to the same:

- Size of the alphabet: $(n+2) \cdot \lceil \log(k+1) \rceil + 6n + 3\lceil \log n \rceil + 28 \in O(n \cdot \log k)$
- Initial number of cells: $2 \in \theta(1)$.
- Initial number of objects: $n + 6 \in \theta(n)$.
- Number of rules: $(n+2) \cdot \lceil \log(k+1) \rceil + 5n + 3\lceil \log n \rceil + 26 \in O(n \cdot \log k)$
- Maximal length of a rule: $3 \in \theta(1)$.

Therefore, a deterministic Turing machine can build $\Pi(g(n,k))$ in a polynomial time with respect to n and k.

Notice that every instance $u = (n, (w_1, \ldots, w_n), k)$ is introduced in the initial configuration of its associated cellular system via an input multiset (i.e. an 1-ary representation) and hence, $|u| \in O(n+k)$ holds.

We would like to recall that the functions cod and s have been defined above for an instance $u = (n, (w_1, \ldots, w_n), k)$ of the Subset Sum problem as follows: $cod(u) = \{\{v_i^{w_i} : 1 \leq i \leq n\}\} \cup \{\{q^k\}\}$, and $s(u) = g(n, k)$, respectively. Both functions are computable in polynomial time and the pair (cod, s) is a polynomial encoding of $I_{\texttt{SubsetSum}}$ in $\mathbf{\Pi}$, since for each instance u of the Subset Sum problem we have that $cod(u)$ is an input multiset of the system $\Pi(s(u))$.

In order to settle the formal verification of the family of tissue P systems we shall prove that all the systems of the family are polynomially bounded, and also that they are sound and complete with respect to $(\texttt{SubsetSum}, cod, s)$.

Polynomial boundness of the family

In order to ensure that the system $\Pi(s(u))$ with input $cod(u)$ is polynomially (indeed, linearly) bounded, it suffices to find the moment in which the computation halts, or at least, an upper bound for it. As we shall show, the number of steps of the computations of any system of the family can always be bounded by a linear function. Nonetheless, we would like to stress that the amount of pre-computed resources for each instance u is polynomial in the size of the instance, since $cod(u)$ needs to be computed and $\Pi(s(u))$ needs to be built.

Proposition 1. *The family* $\mathbf{\Pi} = \{\Pi(g(n, k)) : n, k \in \mathbb{N}\}$ *is polynomially bounded with respect to* $(\texttt{SubsetSum}, cod, s)$.

Proof. (Sketch). We shall informally go through the stages of the computation in order to estimate a bound for the number of steps. The computation will be studied in more detail when addressing the soundness and completeness proof.

Let $u = (n, (w_1, \ldots, w_n), k)$ be an instance of the Subset Sum problem. We shall study what happens during the computation of the system $\Pi(s(u))$ with input $cod(u)$ which processes such instance in order to find the halting step, or at least, an upper bound for it.

First, the generation stage lasts exactly n steps, where all the divisions of the cells of the system are performed.

After that, the pre-checking stage starts with the rule r_4. In the following step the object d_1 arrives to all cells 2 and the counter d_i starts. At the step $n + \lceil \log n \rceil + \lceil \log(k+1) \rceil + 5$, the last element of the counter d_i is reached and the checking stage ends. In this way, in the $(n + \lceil \log n \rceil + \lceil \log(k+1) \rceil + 6)$-th step of the computation the checking takes place. Recall that only one step is needed for the checking, as rule r_{11} takes out in parallel all pairs (p, q) from all cells 2.

The last one is the answer stage. The longest case is obtained when the answer is negative. In this case there is one step where only the counter z_i is working since no element T has reached the cell 1. In the next step an object N is brought from the environment and, finally, in the $(n + \lceil \log n \rceil + \lceil \log(k+1) \rceil + 12)$-th step, the object `no` is sent to the environment.

Therefore, there exists a linear bound with respect to $(n + \log k)$ on the number of steps of the computation.

Soundness and Completeness of the family

In order to prove the soundness and completeness of the family $\mathbf{\Pi}$ with respect to $(\texttt{SubsetSum}, cod, s)$, we shall prove that given an instance u of the Subset Sum problem, the system $\Pi(s(u))$ with input $cod(u)$ sends out an object `yes` if and only if the answer to the problem for the considered instance u is affirmative, and the object `no` is sent out otherwise.

Proposition 2. *The family* $\mathbf{\Pi} = \{\Pi(g(n,k)) : n,k \in \mathbb{N}\}$ *is sound and complete with respect to* $(\texttt{SubsetSum}, cod, s)$.

Proof. In order to complete the proof we shall proceed through a number of auxiliary results.

Remark 1. Given a computation $\mathcal{C}$ we denote the configuration at the i-th step as $\mathcal{C}_i$. Moreover, $\mathcal{C}_i(1)$ will denote the multiset associated to cell 1 in such configuration.

We start with the generation stage (i.e., the n first steps of the computation). It consists of two parallel processes, each of them in one cell.

Lemma 1. *If* $\mathcal{C}$ *is an arbitrary computation of the system, then for all* j *such that* $0 \leq j \leq n$, $\mathcal{C}_j(1) = \{\{z_{j+1}, c_{j+1}^{2^j}, b, \texttt{yes}, \texttt{no}\}\}$ *holds.*

Proof. We shall reason by induction on j.

Base Case. We have $\mathcal{C}_0(1) = \{z_1, c_1, b, \texttt{yes}, \texttt{no}\}$, and thus the lemma holds for $j = 0$.

Case $j < n \rightarrow j+1$**.** Let j be such that $1 \leq j < n$ and we have, by inductive hypothesis, $\mathcal{C}_j(1) = \{\{z_{j+1}, c_{j+1}^{2^j}, b, \texttt{yes}, \texttt{no}\}\}$. In this configuration, only the rules $r_{3,j+1}$ and $r_{4,j+1}$ can be applied to cell 1, and therefore $\mathcal{C}_{j+1}(1) = \{\{z_{j+2}, c_{j+2}^{2^{(j+1)}}, b, \texttt{yes}, \texttt{no}\}\}$.

Lemma 2. *If* $\mathcal{C}$ *is an arbitrary computation of the system, then:*

1. *For each subset* $V \subseteq \{1, \ldots, n\}$ *there exists only one cell* 2 *in* $\mathcal{C}_n$ *whose multiset is* $cod(u) \cup \{\{D\}\} \cup \{\{B_i : i \in V\}\}$
2. *There exist exactly* 2^n *cells labelled by* 2 *in configuration* $\mathcal{C}_n$

Proof. It is clear that division rules cannot be applied in parallel over the same cell. At this point there is an intrinsic non-determinism of the system. We have to apply all the division rules, but the order is non-deterministically chosen.

At time 0, there are n division rules that can be applied. When we apply one of them to the cell labelled by 2, for example, $r_{1,i}$ $(1 \leq i \leq n)$ we eliminate the object A_i and will obtain two new cells, in the first one an object B_i will appear, but not in the second one. The remaining contents of the original cell will be in the two new cells as well.

In the following step, another division rule can be applied to the two cells labelled by 2. As described above, the object A_j that triggers the rule disappears,

and a new object B_j appear in one of the new cells (note that the two cells may chose different objects A_j). This process is repeated in all cells 2 for each division rule.

It is clear that by triggering a division rule $r_{1,j}$ we get two different cells. Only one of them containing B_j. On the other hand, each object A_i appears exactly once in their initial configuration. Therefore, after applying n division rules we obtain 2^n cells labelled by 2.

Moreover, let V be a subset of $\{1,\ldots,n\}$, if for each division rule $[A_i]_2 \rightarrow [B_i]_2[\lambda]_2$ we focus on the cell containing B_i only for $i \in V$ (we select the other cell otherwise), then after n division steps we will have a cell labelled by 2 such that B_j belongs to the cell if and only if $j \in V$, irrespectively of the order in which the division rules are applied.

Lemma 3. *If $\mathcal{C}$ is an arbitrary computation of the system, then for all i such that $1 \leq i \leq \lceil \log n \rceil + \lceil \log(k+1) \rceil + 7$, $\mathcal{C}_{n+i}(1) = \{\{z_{n+i+1}, D^{2^n}, b, \texttt{yes}, \texttt{no}\}\}$ holds*

Proof. In order to prove the lemma it suffices to observe the following:

- $\mathcal{C}_n(1) = \{\{z_{n+1}, c_{n+1}^{2^n}, b, \texttt{yes}, \texttt{no}\}\}$ holds from Lemma 1.
- There exist exactly 2^n cells labelled by 2 in configuration $\mathcal{C}_n$, each of them containing an object D (this follows from Lemma 2).
- In the next step of the computation, only rules r_4 and $r_{2,n+1}$ are applicable on cell 1, yielding $\mathcal{C}_{n+1}(1) = \{\{z_{n+1+1}, D^{2^n}, b, \texttt{yes}, \texttt{no}\}\}$
- During the rest of the checking stage, only rules of type $r_{2,i}$ are applicable on cell 1, and the result follows.

Lemma 4. *Let $\mathcal{C}$ be an arbitrary computation of the system. Then:*

- *For each subset $V \subseteq \{1,\ldots,n\}$ there exists only one cell 2 in $\mathcal{C}_{n+1}$ whose associated multiset is*

$$cod(u) \cup \{\{c_{n+1}\}\} \cup \{\{B_i : i \in V\}\}$$

- *There exist exactly 2^n cells labelled by 2 in configurations $\mathcal{C}_{n+i}$, for $1 \leq i \leq \lceil \log n \rceil + \lceil \log(k+1) \rceil + 12$*

Proof. $\mathcal{C}_{n+1}$ is obtained from $\mathcal{C}_n$ by the application of the rules r_4 and $r_{2,n+1}$ and hence, 2^n objects c_{n+1} in the cell 1 are traded against 2^n objects D from the cells 2 (one from each cell). Then $\mathcal{C}_{n+1}(1) = \{\{z_{n+2}, D^{2^n}, b, \texttt{yes}, \texttt{no}\}\}$ and for every $V \subseteq \{1,\ldots,n\}$ there exists only one cell 2 whose associated multiset is $cod(u) \cup \{\{c_{n+1}\}\} \cup \{\{B_i : i \in V\}\}$

Since no division rule has been applied in this step (actually, they will not be applied anymore along the computation), the number of cells 2 remains the same as in the previous configuration.

Lemma 5. *Let $\mathcal{C}$ be an arbitrary computation of the system. For each i ($1 \leq i \leq \lceil \log n \rceil + 1$) and for each $V \subseteq \{1,\ldots,n\}$ there exists only one cell 2 in $\mathcal{C}_{n+i+1}$ whose associated multiset is*

$$cod(u) \cup \{\{d_i, e_i^{2^{(i-1)}}\}\} \cup \{\{B_j : j \in V\}\}$$

Proof. We shall reason by induction on i.

Case $i = 1$. It suffices to note that r_5 is the only rule that can be applied on cells 2 in configuration $\mathcal{C}_{n+1}$, and the result follows from the previous Lemma.

Case $1 \leq i < \lceil \log n \rceil + 1 \rightarrow i + 1$.

Let i be such that $1 \leq i < \lceil \log n \rceil + 1$ and let us suppose that the result holds for i. Let V be an arbitrary subset of $\{1, \ldots, n\}$, then let us consider its associated cell whose contents are indicated by inductive hypothesis. The only rules applicable to this cell are $r_{6,i}$ and $r_{7,i}$, and therefore the multiset of such cell 2 in $\mathcal{C}_{n+i+2}$ will be

$$cod(u) \cup \{\{d_{i+1}, e_{i+1}^{2^i}\}\} \cup \{\{B_j \,:\, j \in V\}\},$$

as we wanted to prove.

Lemma 6. *Let $\mathcal{C}$ be an arbitrary computation of the system. For each l $(1 \leq l \leq \lceil \log(k+1) \rceil + 2)$ and for each $V \subseteq \{1, \ldots, n\}$ there exists only one cell 2 in $\mathcal{C}_{n+\lceil \log n \rceil + l + 2}$ whose associated multiset is*

$$cod(u) \cup \{\{d_{\lceil \log n \rceil + l + 1}, e_{\lceil \log n \rceil + 1}^{(2^{\lceil \log n \rceil} - |V|)}\}\} \cup \{\{B_{jl}^{2^{(l-1)}} \,:\, j \in V\}\}$$

Proof. We shall reason by induction on l.

Case $l = 1$. Let V be an arbitrary subset of $\{1, \ldots, n\}$, then from the previous Lemma it follows that there exists a cell whose associated multiset in $\mathcal{C}_{n+\lceil \log n \rceil + 2}$ is $cod(u) \cup \{\{d_{\lceil \log n \rceil + 1}, e_{\lceil \log n \rceil + 1}^{2^{\lceil \log n \rceil}}\}\} \cup \{\{B_j \,:\, j \in V\}\}$. The next configuration for this cell is obtained by applying rules $r_{7,\lceil \log n \rceil + 1}$ and $r_{8,j}$ for every $j \in V$:

- $r_{7,\lceil \log n \rceil + 1}$ allows the evolution of the counter d to $d_{\lceil \log n \rceil + 2}$ in each cell 2.
- Each $r_{8,j}$ allows the replacement of the objects B_j (together with a copy of $e_{\lceil \log n \rceil + 1}$) by B_{j1}.

and thus the result holds for $l = 1$.

Case $1 \leq l < \lceil \log(k+1) \rceil \rightarrow l + 1$.

Let l be such that $1 \leq l < \lceil \log n \rceil + 1$ and let us suppose that the result holds for l. That is, there exists only one cell 2 in $\mathcal{C}_{n+\lceil \log n \rceil + l + 3}$ with the multiset

$$cod(u) \cup \{\{d_{\lceil \log n \rceil + l + 1}, e_{\lceil \log n \rceil + 1}^{2^{\lceil \log n \rceil} - |V|}\}\} \cup \{\{B_{jl}^{2^{(l-1)}} \,:\, j \in V\}\}$$

In the next step, rules $r_{7,n+\lceil \log n \rceil + l + 3}$ and $r_{9,j,l}$, for every $j \in V$, will be applied on this cell, and thus the multiset of such cell 2 in $\mathcal{C}_{n+\lceil \log n \rceil + l + 1 + 3}$ will be

$$cod(u) \cup \{\{d_{\lceil \log n \rceil + l + 2}, e_{\lceil \log n \rceil + 1}^{2^{\lceil \log n \rceil} - |V|}\}\} \cup \{\{B_{j(l+1)}^{2^l} \,:\, j \in V\}\}$$

Lemma 7. *Let $\mathcal{C}$ be an arbitrary computation of the system. For each $V \subseteq \{1, \ldots, n\}$ there exists only one cell 2 in $\mathcal{C}_{n+\lceil \log n \rceil + \lceil \log(k+1) \rceil + 5}$ whose associated multiset is*

$$\{\{p^{\sum_{j \in V} w_j}, q^k\}\} \cup \{\{d_{\lceil \log n \rceil + \lceil \log(k+1) \rceil + 4}, e_{\lceil \log n \rceil + 1}^{2^{\lceil \log n \rceil} - |V|}\}\} \cup \{\{B_{j(\lceil \log(k+1) \rceil + 1)}^{2^{\lceil \log(k+1) \rceil} - w_j} \,:\, j \in V\}\}$$

Proof. Let V be an arbitrary subset of $\{1,\ldots,n\}$. From the previous Lemma (taking $l = \lceil \log(k+1) \rceil + 2$), it follows that in $\mathcal{C}_{n+\lceil \log n \rceil + \lceil \log(k+1) \rceil + 4}$ there is a cell 2 associated with this subset whose content is indicated above. In the next step, the following rules are applied:

- $r_{7,\lceil \log n \rceil + \lceil \log(k+1) \rceil + 3}$ allows the evolution of the counter $d_{\lceil \log n \rceil + \lceil \log(k+1) \rceil + 4}$.
- $r_{10,j}$, for every $j \in V$, allow the replacement of all the existing objects v_j (recall that there are $w(a_j)$ copies of v_j in $cod(u)$), together with objects $B_{j(\lceil \log(k+1) \rceil + 1)}$ against $\sum_{j \in V} w_j$ objects p.

Remark 2. From this moment on, for the sake of simplicity, given $V \subseteq \{1,\ldots,n\}$ we shall note by $w_V = \sum_{j \in V} w_j$, and we shall also note by $trash_V$ the following multiset:

$$\{\{e_{\lceil \log n \rceil + 1}^{2^{\lceil \log n \rceil} - |V|}\}\} \cup \{\{B_{j(\lceil \log(k+1) \rceil + 1)}^{2^{\lceil \log(k+1) \rceil} - w_j} : j \in V\}\}$$

Lemma 8. *Let $\mathcal{C}$ be an arbitrary computation of the system and let $V \subseteq \{1,\ldots,n\}$. We have the following:*

- *If $k = w_V$ then there exists only one cell 2 in $\mathcal{C}_{n+\lceil \log n \rceil + \lceil \log(k+1) \rceil + 6}$ whose associated multiset is*
$$\{\{f_1, g_1\}\} \cup trash_V$$
- *If $k < w_V$ then there exists only one cell 2 in $\mathcal{C}_{n+\lceil \log n \rceil + \lceil \log(k+1) \rceil + 6}$ whose associated multiset is*
$$\{\{p^{w_V - k}, f_1, g_1\}\} \cup trash_V$$
- *If $k > w_V$ then there exists only one cell 2 in $\mathcal{C}_{n+\lceil \log n \rceil + \lceil \log(k+1) \rceil + 6}$ whose associated multiset is*
$$\{\{q^{k - w_V}, f_1, g_1\}\} \cup trash_V$$

Proof. Let V be an arbitrary subset of $\{1,\ldots,n\}$. From the previous Lemma, it follows that in $\mathcal{C}_{n+\lceil \log n \rceil + \lceil \log(k+1) \rceil + 5}$ there is a cell 2 associated with this subset whose content is indicated above. In the next step, the following rules are applied:

- r_{11} sends all existing pairs (p, q) to the environment. So, if $k = w_V$ then all objects p and q will disappear of the cell. However, if $k < w_V$ then $w_V - k$ objects p will remain in the cell, and conversely if $k > w_V$ then $k - w_V$ objects q will remain in the cell.
- r_{12} trades the object $d_{\lceil \log n \rceil + \lceil \log(k+1) \rceil + 4}$ against the objects f_1, g_1.

Lemma 9. *Let $\mathcal{C}$ be an arbitrary computation of the system, and let V be a subset of $\{1,\ldots,n\}$.*

1. *For each V such that $w_V = k$, there exists one cell 2 such that:*
   a) its associated multiset in $\mathcal{C}_{n+\lceil \log n \rceil + \lceil \log(k+1) \rceil + 7}$ is $\{\{f_1, g_2\}\} \cup trash_V$
   b) its associated multiset in $\mathcal{C}_{n+\lceil \log n \rceil + \lceil \log(k+1) \rceil + 8}$ is $\{\{T\}\} \cup trash_V$

c) For each $i\,(1 \leq i \leq 4)$*, its associated multiset in* $\mathcal{C}_{n+\lceil \log n\rceil+\lceil \log(k+1)\rceil+i+8}$ *is* $trash_V$

2. *For each* V *such that* $w_V \neq k$*, there exists one cell* 2 *such that for each* $i\,(1 \leq i \leq 6)$ *its associated multiset in* $\mathcal{C}_{n+\lceil \log n\rceil+\lceil \log(k+1)\rceil+i+6}$ *is* $\{\{g_2, p^{w_V-k-1}\}\} \cup trash_V$ *or* $\{\{g_2, q^{k-w_V-1}\}\} \cup trash_V$

Proof. 1. From the previous Lemma, if $k = w_V$ then there exists only one cell 2 in $\mathcal{C}_{n+\lceil \log n\rceil+\lceil \log(k+1)\rceil+6}$ whose associated multiset is

$$\{\{f_1, g_1\}\} \cup trash_V$$

Then, by applying the rule r_{15} (no other rule can be applied in this cell) we obtain in the next step $\{\{f_1, g_2\}\} \cup trash_V$. After that, by applying the rule r_{16} we obtain $\{\{T\}\} \cup trash_V$. Finally, by applying the rule r_{17} the element T is sent to the cell labelled by 1. No more rules can be applied after this moment in the cells labelled by 2. Therefore, for $1 \leq i \leq 4$, the contents of the cell in $\mathcal{C}_{n+\lceil \log n\rceil+\lceil \log(k+1)\rceil+i+8}$ is the multiset $trash_V$.

2. From the previous Lemma, if $k < w_V$ then there exists only one cell 2 in $\mathcal{C}_{n+\lceil \log n\rceil+\lceil \log(k+1)\rceil+6}$ whose associated multiset is

$$\{\{p^{w_V-k}, f_1, g_1\}\} \cup trash_V$$

Then, by applying the rule r_{13} an object p is sent to the environment together with object f_1. In parallel, rule r_{15} trades g_1 against g_2. No more rules can be applied in the cell after this moment. Therefore, for $1 \leq i \leq 5$, the contents of the cell in $\mathcal{C}_{n+\lceil \log n\rceil+\lceil \log(k+1)\rceil+i+7}$ is the multiset $\{\{g_2, p^{w_V-k-1}\}\} \cup trash_V)$. Analogously for the case when $k > w_V$ (applying the rule r_{14} instead of r_{13}).

Lemma 10. *Let* $\mathcal{C}$ *be an arbitrary computation of the system. Let us suppose that there exists* $V \subseteq \{1, \ldots, n\}$ *such that* $w_V = k$*. Then*

(a) $\mathcal{C}_{n+\lceil \log n\rceil+\lceil \log(k+1)\rceil+9}(1) = \{\{z_{n+\lceil \log n\rceil+\lceil \log(k+1)\rceil+10}, D^{2^n}, b, \texttt{yes}, \texttt{no}, T\}\}$
(b) $\mathcal{C}_{n+\lceil \log n\rceil+\lceil \log(k+1)\rceil+10}(1) = \{\{z_{n+\lceil \log n\rceil+\lceil \log(k+1)\rceil+11}, D^{2^n}, S, \texttt{yes}, \texttt{no}\}\}$
(c) $\mathcal{C}_{n+\lceil \log n\rceil+\lceil \log(k+1)\rceil+11}(1) = \{\{z_{n+\lceil \log n\rceil+\lceil \log(k+1)\rceil+11}, D^{2^n}, \texttt{no}\}\}$

Proof. The configuration of item (a) is obtained by the application of rules r_{17} and $r_{2,n+\lceil \log n\rceil+\lceil \log(k+1)\rceil+9}$ to the previous configuration. Analogously, the configurations of items (b) and (c) are obtained by the application of rules r_{18} and r_{19} respectively.

Lemma 11. *Let* $\mathcal{C}$ *be an arbitrary computation of the system. Let us suppose that there does not exist any subset* $V \subseteq \{1, \ldots, n\}$ *such that* $w_V = k$*. Then*

(a) $\mathcal{C}_{n+\lceil \log n\rceil+\lceil \log(k+1)\rceil+9}(1) = \{\{z_{n+\lceil \log n\rceil+\lceil \log(k+1)\rceil+10}, D^{2^n}, b, \texttt{yes}, \texttt{no}\}\}$
(b) $\mathcal{C}_{n+\lceil \log n\rceil+\lceil \log(k+1)\rceil+10}(1) = \{\{z_{n+\lceil \log n\rceil+\lceil \log(k+1)\rceil+11}, D^{2^n}, b, \texttt{yes}, \texttt{no}\}\}$
(c) $\mathcal{C}_{n+\lceil \log n\rceil+\lceil \log(k+1)\rceil+11}(1) = \{\{D^{2^n}, N, \texttt{yes}, \texttt{no}\}\}$
(d) $\mathcal{C}_{n+\lceil \log n\rceil+\lceil \log(k+1)\rceil+12}(1) = \{\{D^{2^n}, \texttt{yes}\}\}$

Proof. The configuration of item (a) and (b) are obtained by the application of rules $r_{2,n+\lceil \log n \rceil + \lceil \log(k+1) \rceil + 9}$ and $r_{2,n+\lceil \log n \rceil + \lceil \log(k+1) \rceil + 10}$ to the previous configuration. Analogously, the configurations of items (c) and (d) are obtained by the application of rules r_{20} and r_{21} respectively.

5.3 Main Results

From the discussion in the previous sections and according to the definition of solvability given in Definition 1, we deduce the following result:

Theorem 1. `Subset Sum` $\in \mathbf{PMC}_{TD}$.

As a consequence of this result we have:

Theorem 2. $\mathbf{NP} \cup \mathbf{co} - \mathbf{NP} \subseteq \mathbf{PMC}_{TD}$.

Proof It suffices to make the following observations: the Subset Sum problem is **NP**-complete, `SubsetSum` $\in \mathbf{PMC}_{TD}$ and the class $\mathbf{PMC}_{TD}$ is stable under polynomial-time reduction, and also closed under complement.

6 Conclusions and Future Work

The physical limitations of current silicon-based hardware have been one of the triggers for the development of alternative models of computation (also known as *unconventional*). In particular, the scientific community is getting increasingly interested on computing models inspired by Nature. These new models abstract features of living entities and use them as inspiration for designing algorithms within new computing paradigms.

Membrane Computing is a new cross-disciplinary field of Natural Computing which has reached an important success in its short life. In these years many results have been presented related to the computational power of membrane devices, but up to now no implementation *in vivo* or *in vitro* has been carried out.

As the classical complexity classes **P** and **NP** are very likely to be different, the design of efficient solutions (in time) to **NP**-complete problems consequently needs to handle an exponential amount of resources. Cellular Computing with Membranes provides a framework where this trade-off between time and space is formalized in a natural way, getting inspiration from the way new cells are created (are *born*) from existing ones. Indeed, the mitosis process (cell division) is the motivation for the model of tissue P systems with cell division used in this paper. As the model allows all existing cells to be divided in parallel at every step, it follows directly that one can produce 2^n membranes in n steps. Using this ability, we have presented in this paper a solution to a numerical **NP**-complete problem using a family of recognizing tissue P systems with cell division.

More precisely, this paper deals with the design and formal verification of an *algorithm* to solve a well-known problem in an efficient and uniform way, and in

this sense it is a theoretical result, mainly related to computational complexity classes. We would like to stress that the result presented in this paper improves previous designs (for other problems) in two senses. On the one hand, the size of the rules is bounded by 3 and, on the other hand, the number of steps and the initial resources are of $O(\log k)$ order instead of being linearly dependent on k.

This is the first design of a solution to a numerical problem in this framework (up to our knowledge), and thus it may be useful as a template or guidance when addressing other numerical problems. Besides, the strategies that have been applied in the design presented in this paper can be also used when working on similar models. For instance, there is a promising new paradigm within Membrane Computing, namely Spiking Neural P systems, that is based on communication between neurons (recall that the inspiration of tissue P systems comes from communication and cooperation between cells in a tissue). Efficient resolution of hard problems has not yet been addressed in this new model, but it may in a near future. We would also like to mention as future work the development of software tools to simulate such computational processes, as the existing simulators for other membrane computing models have proved to be very useful as assistants for designing P systems and for understanding the way they work.

Acknowledgment

The authors acknowledge the support of the project TIN2006-13425 of the Ministerio de Educación y Ciencia of Spain, cofinanced by FEDER funds, and the support of the project of excellence TIC-581 of the Junta de Andalucía.

References

1. D. Díaz-Pernil, M.A. Gutiérrez-Naranjo, M.J. Pérez-Jiménez, A. Riscos-Núñez: A linear–time tissue P system based solution for the 3–coloring problem. *Theoretical Computer Science*, to appear.
2. M.A. Gutiérrez-Naranjo, M.J. Pérez-Jiménez, F.J. Romero-Campero: A linear solution for QSAT with Membrane Creation. In *Membrane Computing. International Workshop WMC6, Vienna, Austria, 2005*, LNCS 3850, Springer, 2006, 241–252.
3. C. Martín Vide, J. Pazos, Gh. Păun, A. Rodríguez Patón: A New Class of Symbolic Abstract Neural Nets: Tissue P Systems. In *Computing and Combinatorics: 8th Annual International Conference, COCOON 2002, Singapore, 2002*, LNCS 2387, Springer, 2002, 290–299.
4. C. Martín Vide, J. Pazos, Gh. Păun, A. Rodríguez Patón: Tissue P systems. *Theoretical Computer Science*, 296 (2003), 295–326.
5. A. Păun, Gh. Păun: The power of communication: P systems with symport/antiport. *New Generation Computing*, 20, 3 (2002), 295–305.
6. Gh. Păun: Computing with membranes. *Journal of Computer and System Sciences*, 61, 1 (2000), 108–143.
7. Gh. Păun: *Membrane Computing – An Introduction*. Springer, Berlin, 2002.

8. Gh. Păun, M.J. Pérez-Jiménez, A. Riscos-Núñez: Tissue P System with cell division. In *Second Brainstorming Week on Membrane Computing*, Sevilla, Report RGNC 01/2004, (2004), 380–386.
9. M.J. Pérez-Jiménez, A. Romero-Jiménez, F. Sancho-Caparrini: A polynomial complexity class in P systems using membrane division. In *Proceedings of the 5th Workshop on Descriptional Complexity of Formal Systems, DCFS 2003*, (2003), 284–294.