Solving the Bin-Packing Problem by Means of Tissue P System with 2-Division

Hepzibah A. Christinal[1], Rose Rani John[1], D. Abraham Chandy[1], and Miguel A. Gutiérrez-Naranjo[2]

[1] Karunya University, Coimbatore, Tamilnadu, India
hepzia@yahoo.com, roseranijohn@karunya.edu, abrahamdchandy@gmail.com
[2] Department of Computer Science and Artificial Intelligence, University of Seville, 41012 Seville, Spain
magutier@us.es

Abstract. The ability of tissue P systems with 2-division for solving **NP** problems in polynomial time is well-known and many solutions can be found in the literature to several of such problems. Nonetheless, there are very few papers devoted to the Bin-packing problem. The reason may be the difficulties for dealing with different number of bins, capacity and number of objects by using exclusively division rules that produce two offsprings in each application. In this paper we present the design of a family of tissue P systems with 2 division which solves the Bin-packing problem in polynomial time by combining design techniques which can be useful for further research.

1 Introduction

Membrane computing was born from the assumption that the processes taking place within the compartmental structure of a living cell can be interpreted as computations [26]. From the beginning, the computational complexity of the membrane algorithms has been a vivid research area [25]. In particular, there is a large amount of papers dealing with the **P** versus **NP** problem in the framework of membrane computing [22]. The **P** versus **NP** problem is one of the most important unsolved problem in computer science and it was chosen as one of the seven Millennium Prize Problems [11]. The precise statement of the problem was introduced in 1971 by Stephen Cook [1], although it was essentially mentioned in a personal communication between Gödel and von Neumann [10].

The problem of deciding whether **P** and **NP** complexity classes are same or not is not yet solved but the efforts for solving it have contributed to the development of new research areas full of interesting open questions. One of them is the research of *frontiers of tractability*, i.e., to identify some features of the computational models such that the corresponding device is able to solve **NP** problems or not depending on the endowment of such features.

In membrane computing there exists an extensive literature devoted to this type of problem (see [22] and the references therein) and the present paper is a novel contribution in such research line. We consider here a variant of one of the

most popular P systems architectures: tissue P systems. Such model was firstly presented in [16,17] by placing the cells in a general graph instead of a tree-like graph as in the cell-like model. Under the hypothesis **P**≠**NP**, Zandron *et al.* [33] established the limitations of P systems that do not use membrane division concerning the efficient solution of **NP**-complete problems.

From this premise, Păun *et al.* presented in [28] the model of tissue P systems endowed with cell division, which can solve **NP**-problems. Since then, many other variants have been presented, e.g., [6,12,13,18,19]. In this way, all solutions of **NP**-complete problems in membrane computing rely on the possibility of P systems to obtain *exponential space* in polynomial time[1]. Besides creating an exponential amount of cells in polynomial time, to solve **NP**-complete problems, we need to be able to effectively use that workspace, by making objects interact. For instance, it is known that, even with membrane division, without polarizations and without dissolution only problems in **P** may be solved [9].

The cell division proposed in [28] was inspired in the *mitosis* of alive cells and consists on the division of one cell into two offsprings. The most extended use of this type of rules is obtaining an exponential amount of cells in linear time, since 2^n cells can be obtained from an initial one after n steps. The ability of tissue P systems with 2-division for solving **NP** problems is well known [19], nonetheless, the study of new problems requires the effort of solving specific designs which, on the one hand, makes stronger the theory and on the other hand, can provide new ideas for unsolved problems in membrane computing.

In particular, the design of a solution for the Bin-packing problem by using 2-division presents important subproblems whose solution can be useful for further development of the theory. The intrinsic difficulty of this problem explains that there are very few papers [23,24] devoted to this problem in the vast literature of solutions for **NP** problems in membrane computing.

The problem can be stated as follows: *Given a set* $A = \{s_1, \ldots, s_n\}$, *a weight function* $\omega : A \to \mathbb{N}$ *and two constants* $b \in \mathbb{N}$, $c \in \mathbb{N}$ *decide whether or not there exists a partition of* A *into* b *subsets such that their weights do not exceed* c. The traditional strategy for designing a solution of an **NP**-problem has a first stage where an exponential amount of cells is created and each feasible candidate to be a solution is placed into one of the exponential amount of available cells. In the Bin-packing problem, this generation of feasible candidates consists on considering all the possibilities for distributing the n objects among the b bins. This means that b^n feasible candidates should be encoded in the corresponding multisets and placed into b^n different cells.

The 2-division proposed in [28] fits perfectly when the exponential growth of the number of candidates is on basis 2, as in the SAT problem (e.g., [2,7]) or the Partition problem [5,8] or even on basis 3 by performing a new division on

[1] Such solutions are technically correct, but, of course, the exponential generation of new working space has evident limits from a practical point of view. Any practical implementation of P systems solving **NP**-problems with physical support only could solve small instances of the problem.

one of the offsprings, as in the 3-COL problem [4]. In our case, the problem is different, since the number of necessary membranes depends on the parameter b. In this way, the designed family of tissue P systems must be able to build b^n membranes in polynomial time on b and n for each b and n by using 2-division.

This leads us to solve two subproblems: The first one is place each of the n objects in one of the b bins (if the distribution condition of objects is satisfied). This means that we need to represent each of the symbols in $\{0,\ldots,b-1\}$ with a 2-division process for each object s_i with $i \in \{1,\ldots,n\}$. And the second problem is to implement such distribution of n objects, one for each object in the set A. In the next sections, we explain the details of the corresponding solutions.

The paper is organized as follows. In Sect. 2, the formal description of the used model of P systems is recalled. Section 3 is devoted to the main ideas related to recognizer P systems, the framework for solving decision problems. Our solution is presented in Sect. 4 and the paper ends with some conclusions and hints for future work.

2 Tissue P Systems with Cell Division

In the first definition of the model of tissue P systems [16,17] the membrane structure did not change along with the computation. Based on the biological *mitosis*, Păun *et al.* presented in [28] a tissue P system model endowed with 2-division is one of the most used and well-known P system models nowadays. We briefly recall their syntax and semantics.

Formally[2], a *tissue P system with cell division* of degree $q \geq 1$ is a tuple of the form

$$\Pi = (\Gamma, \mathcal{E}, \Sigma, w_1, \ldots, w_q, \mathcal{R}, i_{in}, i_{out}),$$

where:

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

A tissue P system with cell division of degree $q \geq 1$ can be seen as a set of q cells (each one consisting of an elementary membrane) labelled by $1,2,\ldots,q$. We shall use 0 as the label of the environment, and i_{in} and i_{out} denote, respectively, the input and output regions (which can be a region inside a membrane or the environment).

[2] The reader is supposed to be familiar with the basic concepts of membrane computing. See [29] for details.

The communication rules determine an implicit net of channels, where the nodes are the cells and the edges indicate if it is possible for pairs of cells to communicate directly. This is a dynamic graph, in which new nodes can be produced by the application of division rules. Note also that the connections only depend on the label of the cell, and thus when a cell is divided, the two new cells will have identical connections. Nevertheless, this graph is just an intuition, we shall not handle it explicitly along the computations.

The strings $w_1, \ldots, w_q$ describe the multisets of objects placed 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 arbitrary large amount of copies.

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. The application of this rule means that the objects of the multisets represented by u and v are interchanged between the two cells.

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.

Rules are used as usual in the framework of membrane computing, that is, in a maximally parallel way. In one step, each object in a membrane 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 evolve in that step.

3 Recognizer P Systems

The notion of recognizer P system is general enough to cover many P system variants. Such P systems are a well-known model of P systems which are basic for the study of complexity aspects in membrane computing. Roughly speaking, a recognizer P system is a P system which takes some information as *input* and outputs a distinguished object which can be considered as a *decision* on the input. Of course, some other conditions are imposed, but the general framework does not depend on the type of rules or the membrane structure of the P system.

Next, we briefly recall some basic ideas related to them. A detailed description, is given in [21,22]. In recognizer P systems all computations halt; there are two distinguished objects traditionally called `yes` and `no` (used to signal the result of the computation), and exactly one of these objects is sent out to the environment (only) in the last computation step.

Let us recall that a decision problem X is a pair (I_X, θ_X) where I_X is a language over a finite alphabet (the elements are called *instances*) and θ_X is a predicate (a total Boolean function) over I_X. Let $X = (I_X, \theta_X)$ be a decision problem. A *polynomial encoding* of X is a pair (cod, s) of polynomial time

computable functions over I_X such that for each instance $w \in I_X$, $s(w)$ is a natural number representing the *size* of the instance and $cod(w)$ is a multiset representing an encoding of the instance. Polynomial encodings are stable under polynomial time reductions.

Let $\mathcal{R}$ be a class of recognizer P systems with input membrane. A decision problem $X = (I_X, \theta_X)$ is solvable in a uniform way and polynomial time by a family $\mathbf{\Pi} = (\Pi(n))_{n \in \mathbf{N}}$ of P systems from $\mathcal{R}$ – we denote this by $X \in \mathbf{PMC}_{\mathcal{R}}$ – if the family $\mathbf{\Pi}$ is polynomially uniform by Turing machines, i.e., there exists a polynomial encoding (cod, s) from I_X to $\mathbf{\Pi}$ such that the family $\mathbf{\Pi}$ is polynomially bounded with regard to (X, cod, s); this means that 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 and complete with regard to (X, cod, s).

4 The Solution to Bin Packing Problem

In this section we provide a family of tissue P systems with 2-division which solves the Bin-packing problem in polynomial time. Before giving the formal description of the P system, we provide some hints about how the problem has been solved.

As pointed out in the Introduction, each of the n objects can potentially be placed on one of the b bins. In such way, b^n candidates (and hence b^n cells) should be built in polynomial time by using 2-division. The solution needs the preprocess of the parameter m, where m is the least integer which satisfies $b \leq 2^m$, i.e., $m = \lceil log_2 b \rceil$. In the same way, by generating 2^m cells with m 2-division steps we can assure that at least b cells are generated. The idea of considering a number of logarthmic steps is not new in the design of membrane computing solutions (see, e.g., [3]). Nevertheless, the mere generation of b (or more) cells are not enough. Each object s_i of the set $A = \{s_1, \ldots, s_n\}$ must be placed in one of the bins $\{0, \ldots, b-1\}$, where each membrane encoding a different bin has been generated by 2-division. In this way, the natural encoding of each number in $\{0, \ldots, b-1\}$ is performed by a binary representation $C_{m-1}C_{m-2}\ldots C_0$ where[3] $C_i \in \{T_i, F_i\}$ for $i \in \{0, \ldots, b-1\}$. In this way, each 2-division will produce an object T_i or F_i in each offspring and the value i will control the number of the division in the iterative process of building 2^m cells. This process will be controlled by the set of rules **R1** (see below).

From the 2^m generated cells, only b of them are necessary in our solution. The remaining $2^m - b$ cells (if any) will remain inactive. This means that we only need rules for control b different values $\{0, \ldots, b-1\}$. Such values are originally represented via a binary representation by using m objects. Nonetheless, for the sake of readability, in our solution we propose to take a decimal representation by a unique symbol object b_j with $j \in \{0, \ldots, b-1\}$. The trick for recovering such notation is controlled by the set of rules **R2** (see below).

[3] T_i and F_i stand for $True$ and $False$ in the $i-th$ position and, as usual, represents 1 and 0 in a binary representation.

The key idea of such set of rules consists of keeping in each cell of label 1 an object b_k which acts as an accumulator in the index i (with b_0 at the beginning). The different values of the binary representation $C_{m-1}C_{m-2}\dots C_0$ with $C_i \in \{T_i, F_i\}$ arrive sequentially to the cell:

- If the object T_i arrives and $k + 2^i$ does not exceed the value $b - 1$ then, the objects T_i and b_k are traded against a new object b_{k+2^i}. If $k + 2^i \geq b$, then the corresponding membrane does not encode one of the possible bins (enumerated from 0 to $b - 1$) and it stays inactive during the remaining computation steps.
- If the object F_i arrives, nothing is added to the accumulator and the object b_k does not change.

Bearing in mind these considerations, the proposed solution can be split in to the following stages:

- *Generation and Calculation stage:* In this stage the b^n candidates for solution are generated (each of the n objects in $A = \{s_1, \dots, s_n\}$ can go to one of the b bins). For each s_i, $2m + 3$ steps are performed and the number of steps in this stage is $O(m \cdot n)$.
  During the generation stage, the process of assigning an object s_i to a bin is performed n times. After each assignment, the free capacity of the corresponding bin (represented by objects p_j) is decreased in an amount equal to the weight $w(s_i)$ if it is possible. This decrement is performed by the set of rules **R4** which send pairs of objects representing unit of weights and units of free capacity to the environment. If there is no free capacity enough for such assignment, some objects representing weights will not be sent to the environment.
- *Checking stage:* This stage is performed in parallel in the all b^n cells which represent candidates (let us recall that the $2^m - b$ cells extra generated in each construction process keep inactive). This stage consists on checking if all the objects representing weights have been sent to the environment. If there is no such objects in the corresponding cell, then the candidate placed in such membrane represents a solution to the Bin-packing problem. Otherwise, if there is one or more object in the cell, then the assignment of objects to cells does not satisfy the restriction and it is not a solution to the problem.
- *Output stage:* Finally, the output stage controls that only one object *yes* or only one object *no* is sent to the output cell in the last step of computation.

Each instance of the Bin-packing Problem is stated by a set of n objects, $A = \{s_1, \dots, s_n\}$, a set of weights $\omega(A) = \{\omega(s_1), \dots, \omega(s_n)\}$ and two constants $b \in \mathbb{N}$ (the number of bins) and $c \in \mathbb{N}$ (the capacity). We propose a family of tissue P systems with 2-division which solves the Bin-packing Problem where each tissue P system of the family depends on the parameters n, b and c. For the sake of simplicity, we will consider a pre-computed extra parameter $m = \lceil log_2 b \rceil$ which only depends on b. Let us notice that the set of objects $A = \{s_1, \dots, s_n\}$ is encoded as the multiset $s_1^{w(s_1)}, \dots, s_n^{w_{s_n}}$, i.e., the weight of the object s_i is

represented in its multiplicity. Such multiset is placed in the input cell in the initial configuration and the computation starts.

Each tissue P system of the family is of the following form[4]:

$$\Pi(n,b,c,m) = (\Gamma, \mathcal{E}, \Sigma, w_1, w_2, R, i_{in}, in_{out})$$

where:

– Γ is the alphabet of objects used in the computation

$$\Gamma = \mathcal{E} \cup \Sigma \cup \{A_0, b_0, d_1, z_1, x_0\} \cup \{p_i \,:\, i \in \{0,\ldots,b-1\}\}$$

– $\mathcal{E}$ is the set of objects initially placed in the environment.

$$\begin{aligned}\mathcal{E} = {} & \{yes, no, g, f_0, f_1, y_0, y_1, \#\} \\ & \cup \{T_i, k_i\, k_i, r_i \,:\, i \in \{0,\ldots,m-1\}\} \\ & \cup \{A_i \,:\, i \in \{0,\ldots,m\}\} \\ & \cup \{b_i \,:\, i \in \{0,\ldots,b-1\}\} \\ & \cup \{q_i, h_i \,:\, i \in \{0,\ldots,b-1\}\} \\ & \cup \{z_i, d_i \,:\, i \in \{2,\ldots,n\}\} \\ & \cup \{x_i \,:\, i \in \{1,\ldots,n*(2m+3)+1\} \\ & \cup \{k_{i,j} \,:\, i \in \{0,\ldots,b-1\}\, j \in \{0,1\}\}\end{aligned}$$

– $\Sigma = \{s_i \,:\, i \in \{1,\ldots,n\}\}$ is the input alphabet.
– Initially, the initial configuration has only two cells. The initial multisets are

$$w_1 = A_0\, b_0\, d_1\, z_1^c\, p_0^c \ldots p_{b-1}^c \quad \text{and} \quad w_2 = x_{0,1}$$

– The following are the set of rules R :

R1. $[A_i]_1 \rightarrow [T_i]_1[F_i]_1 : 0 \leq i \leq m-1$

These rules control the generation of new membranes by division. Since $b \leq 2^m$, it is guaranteed that after m applications of 2-division rules, the number of membranes is enough for encoding b bins. Each of the offsprings has an object T_i or F_i of the corresponding number in binary encoding. $C_{m-1}C_{m-2}\ldots C_0$ with $C_i \in \{T_i, F_i\}$.

R2. $(1, T_i\, b_{k-2^i}/b_k\, A_{i+1}, 0) : 0 \leq i \leq m-1, \quad 0 \leq k \leq min\{b-2^i, 2^i\} - 1$
$(1, F_i/A_{i+1}, 0) : 0 \leq i \leq m-1$

As pointed out above, the binary number $C_{m-1}C_{m-2}\ldots C_0$ with $C_i \in \{T_i, F_i\}$ is not stored in such form in the cells. Each cell with label 1 has an object b_k (initially b_0) which can be considered as an accumulator for the decimal representation of the number $C_{m-1}C_{m-2}\ldots C_0$ with $C_i \in \{T_i, F_i\}$. If a new object T_i arrives to the cell by the application of a division rule, then T_i and b_k are traded against one object b_{k+2^i} if b_{k+2^i} is the index of one of the bins,

[4] As usual, we omit the parameters in the description for the sake of readability.

i.e., if it belongs to $\{0,\ldots,b-1\}$. Let us remark that the application of the rule also brings an object A_{i+1} and the cell is prepared for a new division. If $k+2^i \geq b$, then the rule is not applied, the object A_{i+1} is not brought into the cell and the cell remains inactive. If an object F_i arrives, the accumulator b_k is not modified.

R3. $(1, A_m\, b_r / q_r^c\, k_{r,0}) : r \in \{0,\ldots,b-1\}$

When the object A_m arrives to a cell 1, the calculus of b_r has finished. This means that the corresponding object s_i from $A = \{s_1,\ldots,s_n\}$ is placed on the $r-th$ bin. In order to help in the subtraction of the weight of the object s_i from the free capacity of the bin r, c objects q_r are brought into the cell. An object $k_{r,0}$ is also brought for technical reasons.

R4. $(1, z_i\, s_i\, q_r\, p_r / \#) : i \in \{1,\ldots,n\} \quad r \in \{0,\ldots,b-1\}$

This set of rules is the key in the calculus of the total weigh associated to each bin. Its usage exploits the massive parallelism of P systems and the maximal application of the rules. Let us notice that in each application of the rule only one object of the four kind z_i, s_i, q_r and p_r is sent out of the cell, so the number of applications correspond to the minimum of amounts of these objects. Since there are exactly c objects of type z_i and q_r, the number of applications corresponds to the minimum between the multiplicities of s_i and p_r. Let us remark that if the weight of the object s_i (encoded in their multiplicity) is greater than the free capacity of the r-th bin (encoded in the multiplicity of objects p_r) then at least one object s_i will remain in the cell after the application of these rules.

R5. $\left.\begin{array}{l}(1, k_{r,0}/h_r^c, k_{r,1}, 0)\\ (1, h_r\, q_r/\#, 0)\end{array}\right\}$ for $r \in \{0,\ldots,b-1\}$.

These rules can be considered as *cleaning* rules. After the subtraction performed by the set of rules **R4**, the leftovers objects q_r (if any) must be sent out of the cells with label 1 in order to avoid undesired interactions. This is the technical reason of the objects $k_{r,0}$ and $k_{r,1}$.

R6. $(1, k_{r,1}\, d_i / A_0\, b_0\, z_{i+1}^c\, d_{i+1}, 0) : i \in \{1,\ldots,n-1\} \quad r \in \{0,\ldots,b-1\}$
$(1, k_{r,1}\, d_n / f_0\, g, 0) : r \in \{0,\ldots,b-1\}$

When the object $k_{r,1}$ appears in a cell with label 1, the cleaning process performed by the set of rules **R5** has finished and the cell is prepared for starting the process of a new object s_{i+1} This set of rules **R6** brings the objects A_0 and b_0 for starting newly the division together with the objects z_{i+1} and d_{i+1}. When all the objects s_i, $i \in \{1,\ldots,n\}$ are processed, the objects f_0 and g are brought into the cells 1 which remain active.

R7. $(2, x_i / x_{i+1}, 0) : 0 \leq i \leq n*(2m+3)-1$
$(2, x_{n*(2m+3)} / x_{n*(2m+3)+1}, y_0, 0)$

The objects x_i in the cell 2 act as a counter. When it reaches $x_{n*(2m+3)+1}$, a new object y_0 is also brought into the cell 2 for controlling the output process.

R10. $(1, g\, s_i/\#, 0) : i \in \{1, \ldots, n\}$
$(1, f_0/f_1, 0)$
$(1, f_1\, g/x_{n*(2m+3)+1}, 2)$
$(2, y_0/y_1, 0)$
$(2, g, y_1/yes, 0)$
$(2, x_{n*(2m+3)+1}\, no, 0)$

These rules control the output process. Two cases must be considered:

- **Case 1:** In all the active cells with label 1, there is at least one object s_i. This means that all the possible assignment of objects to bins exceed the capacity. In this case, a rule $(1, g\, s_i/\#, 0)$ is applied in such membranes and g is sent out to the environment, the rule $(1, f_1\, g/x_{n*(2m+3)+1}, 2)$ is not applied and finally, the rule $(2, x_{n*(2m+3)+1}\, no, 0)$ sends an object no to the cell with label 2.
- **Case 2:** There is at least one of the cells with label 1, where there are no objects s_i. This means that all the assignment of objects encoded in such membrane does not exceed the capacity and it represents a solution for the problem. In this case, a rule $(1, g\, s_i/\#, 0)$ is not applied in such membrane and, in the next step of computation, the rule $(1, f_1\, g/x_{n*(2m+3)+1}, 2)$ is applied, the object g is sent to the cell with label 2 and finally, the application of $(2, g, y_1/yes, 0)$ sends an object yes to the output cell.

– Finally, the input cell has label 1, $i_{in} = 1$ and the output cell has label 2, $i_{out} = 2$.

4.1 A Short Overview

Next, we provide some hints on the computation. The initial configuration $\mathbb{C}_0$ contains an object A_0 which starts the division process, an object b_0 which works as an accumulator initially set to 0 and objects d_1 and z_1, where the index a denotes that the first object in the set $A = \{s_1, \ldots, s_n\}$ will be process at the beginning. According to the process described above, the cells with label 1 are divided and, in parallel, the counted b_r is increased. The configuration $\mathbb{C}_{2m}$ contains an object A_m and an object b_r encoding that the first object of the set A is placed into the r-th bin. In the next three steps, rules from the sets **R3**, **R4** and **R5** perform the subtraction between the weight of the processed object and the free capacity of the r-th bin (represented by the multiplicity of the object p_r). The first set of rules from **R6** is also applied and in $\mathbb{C}_{2m+3}$ the corresponding cells 1 contain new objects A_0 and b_0 and objects d_2 and z_2, denoting that the second object of the set A will be processed.

This sequence of $2m + 3$ steps is repeated n times (as many times as objects in A). In the configuration $\mathbb{C}_{n*(2m+3)}$ all the objects in A have been processed and the objects f_0 and g appear. Only three steps later, in output stage with two possible cases described above, the halting configuration $\mathbb{C}_{n*(2m+3)+3}$ is reached.

4.2 Computational Resources

Each tissue P system of the family described above depends on three parameters: n, the number of object to distribute among the bins; c, the capacity of the bins;

and b the number of the bins. For the sake of simplicity, the description also includes a forth parameter m, which only depends on the parameter b, $m = \lceil log_2 b \rceil$. Each P system $\Pi(n, b, c, m)$ processes all the input set $A = \{s_1, \ldots, s_n\}$ regardless the weight function $\omega : A \rightarrow \mathbb{N}$.

Initially, the P system has only two membranes and the number of objects of the alphabet is $O(nm + b)$. The set of rules is $O(nb)$ and the number of steps of the computation is $O(nm)$. Let us remark the special case of rules of the set **R2**. In spite of a potential exponential number of cells can be built, not all of them encode an object b_r with $r \in \{0, \ldots, 2^m - 1\}$, which would lead to an exponential amount of initial resources. In our solution, we only deal with objects b_i with $i \in \{0, \ldots, b - 1\}$, so the initial amount of resources are polynomial function on the input parameters.

5 Conclusions

The complexity theory of membrane computing is full of interesting open problems. The most important is the *Păun conjecture* [14,20,27] which ask if a concrete P system model is able to solve or not **NP**-problems in polynomial time. The question has been open for more than ten years and even today nobody knows the answer. If the answer is that **NP**-problems can be solved in the such model, a simple way to prove is by providing the design of a family which effectively solves an **NP**-complete problem. In this way, or it is impossible to find such design or we need to make new efforts and finding new techniques for designing solutions to **NP**-problems which allow us to get the skills for solving the conjecture. In this research line, the design presented in this paper can help to make the theory stronger and provide new ideas for dealing with open questions.

Finally, let us remark the important role of the definition for recognizer P systems we have used in this paper. On the one hand, this definition is quite restrictive, since only one object *yes* or *no* is sent to the environment in any computation. In the literature one can find other definitions of recognizer P systems and therefore other definitions of what it means *to solve* a problem in the framework of Membrane Computing. On the second hand, the synchronization of the processes in the different membranes plays a key role in the design presented in this paper, but in the literature one can find some solutions to **NP**-problems in time free membrane computing models (e.g., [15,30–32]). The study of the complexity classes in membrane computing deserves a deep revision under these new definitions.

References

1. Cook, S.A.: The complexity of theorem-proving procedures. In: Proceedings of the Third Annual ACM Symposium on Theory of Computing, STOC 1971, NY, USA, pp. 151–158. ACM, New York (1971)
2. Cordón-Franco, A., Gutiérrez-Naranjo, M.A., Pérez-Jiménez, M.J., Sancho-Caparrini, F.: A prolog simulator for deterministic P systems with active membranes. New Gener. Comput. **22**(4), 349–363 (2004)

3. Díaz-Pernil, D., Gutiérrez-Naranjo, M.A., Pérez-Jiménez, M.J., Riscos-Núñez, A.: A logarithmic bound for solving subset sum with P systems. In: Eleftherakis, G., Kefalas, P., Păun, G., Rozenberg, G., Salomaa, A. (eds.) WMC 2007. LNCS, vol. 4860, pp. 257–270. Springer, Heidelberg (2007). doi:10.1007/978-3-540-77312-2_16
4. Díaz-Pernil, D., Gutiérrez-Naranjo, M.A., Pérez-Jiménez, M.J., Riscos-Núñez, A.: A uniform family of tissue P systems with cell division solving 3-COL in a linear time. Theoret. Comput. Sci. **404**(1–2), 76–87 (2008)
5. Díaz-Pernil, D., Gutiérrez-Naranjo, M.A., Pérez-Jiménez, M.J., Riscos-Núñez, A.: A linear time solution to the partition problem in a cellular tissue-like model. J. Comput. Theor. Nanosci. **7**(5), 884–889 (2010)
6. Freund, R., Păun, G., Pérez-Jiménez, M.J.: Tissue P systems with channel states. Theoret. Comput. Sci. **330**(1), 101–116 (2005)
7. Gazdag, Z., Kolonits, G.: A new approach for solving SAT by P systems with active membranes. In: Csuhaj-Varjú, E., Gheorghe, M., Rozenberg, G., Salomaa, A., Vaszil, G. (eds.) CMC 2012. LNCS, vol. 7762, pp. 195–207. Springer, Heidelberg (2013). doi:10.1007/978-3-642-36751-9_14
8. Gutiérrez-Naranjo, M.A., Pérez-Jiménez, M.J., Riscos-Núñez, A.: A fast P system for finding a balanced 2-partition. Soft. Comput. **9**(9), 673–678 (2005)
9. Gutiérrez–Naranjo, M.A., Pérez–Jiménez, M.J., Riscos–Núñez, A., Romero–Campero, F.J.: On the power of dissolution in P systems with active membranes. In: Freund, R., Păun, G., Rozenberg, G., Salomaa, A. (eds.) WMC 2005. LNCS, vol. 3850, pp. 224–240. Springer, Heidelberg (2006). doi:10.1007/11603047_16
10. Hartmanis, J.: Gödel, von Neumann and the P =? NP problem. In: Rozenberg, G., Salomaa, A. (eds.) Current Trends in Theoretical Computer Science - Essays and Tutorials, World Scientific Series in Computer Science, vol. 40, pp. 445–450. World Scientific, Singapore (1993)
11. Jaffe, A.M.: The millennium grand challenge in mathematics. Not. Am. Math. Soc. **53**(6), 652–660 (2006)
12. Krishna, S.N., Lakshmanan, K., Rama, R.: Tissue P systems with contextual and rewriting rules. In: PĂun, G., Rozenberg, G., Salomaa, A., Zandron, C. (eds.) WMC 2002. LNCS, vol. 2597, pp. 339–351. Springer, Heidelberg (2003). doi:10.1007/3-540-36490-0_22
13. Lakshmanan, K., Rama, R.: On the power of tissue P systems with insertion and deletion rules. In: Alhazov, A., Martín-Vide, C., Păun, G. (eds.) Preproceedings of the Workshop on Membrane Computing, pp. 304–318, Tarragona, 17–22 July 2003
14. Leporati, A., Manzoni, L., Mauri, G., Porreca, A.E., Zandron, C.: Simulating elementary active membranes - with an application to the P conjecture. In: Membrane Computing - 15th International Conference, CMC 2014, Prague, Czech Republic, 20–22 August 2014, Revised Selected Papers, pp. 284–299 (2014)
15. Liu, X., Suo, J., Leung, S.C.H., Liu, J., Zeng, X.: The power of time-free tissue P systems: attacking NP-complete problems. Neurocomputing **159**, 151–156 (2015)
16. Martín-Vide, C., Pazos, J., Păun, G., Rodríguez-Patón, A.: A new class of symbolic abstract neural nets: tissue P systems. In: Ibarra, Oscar H., Zhang, Louxin (eds.) COCOON 2002. LNCS, vol. 2387, pp. 290–299. Springer, Heidelberg (2002). doi:10.1007/3-540-45655-4_32
17. Martín-Vide, C., Pazos, J., Rodríguez-Patón, A.: Tissue P systems. Theoret. Comput. Sci. **296**(2), 295–326 (2003)
18. Pakash, V.: On the power of tissue P systems working in the maximal-one mode. In: Alhazov, A., Martín-Vide, C., Păun, G. (eds.) Preproceedings of the Workshop on Membrane Computing, pp. 356–364, Tarragona, 17–22 July 2003

19. Pan, L., Pérez-Jiménez, M.J.: Computational complexity of tissue-like P systems. J. Complex. **26**(3), 296–315 (2010)
20. Pérez-Hurtado, I., Pérez-Jiménez, M.J., Riscos-Núñez, A., Gutiérrez-Naranjo, M.A., Rius-Font, M.: On a partial affirmative answer for a Păun's conjecture. Int. J. Found. Comput. Sci. **22**(1), 55–64 (2011)
21. Pérez-Jiménez, M.J.: An approach to computational complexity in membrane computing. In: Mauri, G., Păun, G., Pérez-Jiménez, M.J., Rozenberg, G., Salomaa, A. (eds.) WMC 2004. LNCS, vol. 3365, pp. 85–109. Springer, Heidelberg (2005). doi:10.1007/978-3-540-31837-8_5
22. Pérez-Jiménez, M.J., Riscos-Núñez, A., Romero-Jiménez, A., Woods, D.: Complexity - membrane division, membrane creation. In: Păun et al. [29], pp. 302–336
23. Pérez-Jiménez, M.J., Romero-Campero, F.J.: An efficient family of P systems for packing items into bins. J. Univ. Comput. Sci. **10**(5), 650–670 (2004)
24. Pérez-Jiménez, M.J., Romero-Campero, F.J.: Solving the binpacking problem by recognizer P systems with active membranes. In: Păun, G., Riscos-Núñez, A., Romero-Jiménez, Á., Sancho-Caparrini, F. (eds.) Second Brainstorming Week on Membrane Computing, pp. 414–430. Fénix Editora, Sevilla (2004)
25. Pérez-Jiménez, M.J., Romero-Jiménez, Á., Sancho-Caparrini, F.: Complexity classes in models of cellular computing with membranes. Nat. Comput. **2**(3), 265–285 (2003)
26. Păun, G.: Membrane Computing: An Introduction. Springer, Berlin (2002)
27. Păun, G.: Further twenty six open problems in membrane computing. In: Third Brainstorming Week on Membrane Computing, pp. 249–262. Fénix Editora, Sevilla, Spain (2005)
28. Păun, G., Pérez-Jiménez, M.J., Riscos-Núñez, A.: Tissue P systems with cell division. Int. J. Comput. Commun. Control **3**(3), 295–303 (2008)
29. Păun, G., Rozenberg, G., Salomaa, A. (eds.): The Oxford Handbook of Membrane Computing. Oxford University Press, Oxford (2010)
30. Song, B., Song, T., Pan, L.: A time-free uniform solution to subset sum problem by tissue P systems with cell division. Math. Struct. Comput. Sci. **27**(1), 17–32 (2017)
31. Song, T., Luo, L., He, J., Chen, Z., Zhang, K.: Solving subset sum problems by time-free spiking neural P systems. Appl. Math. Inf. Sci. **8**(1), 327–332 (2014)
32. Song, T., Macías-Ramos, L.F., Pan, L., Pérez-Jiménez, M.J.: Time-free solution to SAT problem using P systems with active membranes. Theoret. Comput. Sci. **529**, 61–68 (2014)
33. Zandron, C., Ferretti, C., Mauri, G.: Solving NP-complete problems using P systems with active membranes. In: Antoniou, I., Calude, C.S., Dinneen, M.J. (eds.) UMC 2000. DMTCS, pp. 289–301. Springer, London (2000). doi:10.1007/978-1-4471-0313-4_21