AAECC (2012) 23:233–249
DOI 10.1007/s00200-012-0176-6

ORIGINAL PAPER

Using Membrane Computing for Effective Homology

Daniel Díaz-Pernil · Hepzibah A. Christinal · Miguel A. Gutiérrez-Naranjo · Pedro Real

Received: 7 February 2011 / Revised: 23 April 2012 / Accepted: 25 April 2012 /
Published online: 4 November 2012

Abstract *Effective Homology* is an algebraic-topological method based on the computational concept of chain homotopy equivalence on a cell complex. Using this algebraic data structure, Effective Homology gives answers to some important computability problems in Algebraic Topology. In a discrete context, Effective Homology can be seen as a combinatorial layer given by a forest graph structure *spanning* every cell of the complex. In this paper, by taking as input a pixel-based 2D binary object, we present a logarithmic-time uniform solution for describing a chain homotopy operator ϕ for its adjacency graph. This solution is based on Membrane Computing techniques applied to the spanning forest problem and it can be easily extended to higher dimensions.

Keywords Tissue-like P systems · Membrane Computing · Effective Homology · Computational Algebraic Topology · Digital Topology

D. Díaz-Pernil (✉) · P. Real
Research Group on Computational Topology and Applied Mathematics,
Universidad de Sevilla, Sevilla, Spain
e-mail: sbdani@us.es

P. Real
e-mail: real@us.es

H. A. Christinal
Karunya University, Coimbatore, Tamilnadu, India
e-mail: hepzi@us.es

M. A. Gutiérrez-Naranjo
Research Group on Natural Computing, Universidad de Sevilla, Sevilla, Spain
e-mail: magutier@us.es

1 Introduction

Homology theory is a branch of Algebraic Topology that attempts to distinguish between spaces by constructing algebraic invariants that reflect the connectivity properties of the space. The field has its origins in the work of the French mathematician Henri Poincaré. Homology groups (related to the *different* n-dimensional holes, connected components, tunnels, cavities, etc.) are invariants from Algebraic Topology which are frequently used in Digital Image Analysis and Structural Pattern Recognition. In some sense, they reflect the topological nature of the object in terms of the number and features of their holes.

Effective Homology [23,26], is a algebraic-topological theory mainly based on the computational notion of chain homotopy equivalence, a concept which algebraically connects a cell complex or subdivided object with its homology groups. Roughly speaking, a chain homotopy equivalence can be specified by an operator, called *chain homotopy operator*, working at level of linear combinations of cells which represents an efficient and non-redundant way of connecting cells. For instance, a chain homotopy operator at level of cells of dimension 0 of a cell complex K can be completely described by a directed spanning forest (as many trees as connected components the object has) of the graph subcomplex formed by all the cells of K of dimension 0 and 1. Effective Homology uses chain homotopy operators for capturing homology information and for representing the object in an algebraic-topological way. In fact, this idea is underlying the Eilenberg-MacLane work [9,10] for computing the homology of *prime* spaces in homotopy theory, and it has been recently used in discrete image context. In [13], a method for computing homology aspects (with coefficients in the finite field $\mathbb{Z}/2\mathbb{Z} = \{0, 1\}$) of a three dimensional digital binary-valued volume V considered over a body-centered-cubic grid is described. The representation used in that paper for a digital image is an algebraic-topological model (AT-model) consisting in two parts: (a) (*geometric modeling level*) A cell complex $K(V)$ topologically equivalent to the original volume is constructed. A 3D-cell complex consists of vertices (0-cells), edges (1-cells), faces (2-cells) and polyhedra (3-cells). In particular, each edge connects two vertices, each face is enclosed by a loop of edges, and each 3-cell is enclosed by an envelope of faces; (b) (*homology analysis level*) Homology information about $K(V)$ is exclusively codified in terms of a chain homotopy operator [11,12].

This method has recently evolved to a technique for generating a $\mathbb{Z}/2\mathbb{Z}$-coefficients. It takes an AT-model for a 26-adjacency voxel-based digital binary volume V using a polyhedral cell complex at geometric modeling level [14,15,17] and a chain homotopy operator described by a combinatorial vector field (a set of semidirected forests or a discrete differential form) at homology analysis level [24,25]. For instance, a chain homotopy operator at level of cells of dimension 0 (vertices) of a cell complex $K(V)$ can be completely described by a semidirected spanning forest of the graph subcomplex formed by all the cells of $K(V)$ of dimension 0 and 1.

In Fig. 1, a pixel-based digital object O (first picture from the left) is analyzed as a cell complex in which the square pixels are the 0-cells. The 1-cells are edges joining 8-neighbor pixels and these 2-cells are triangles or squares formed by three or four mutually (and in a maximal way) 8-adjacent pixels. Figure 1b describes this cell complex (in dark grey) in which the barycenters of the different cells are drawn (solid

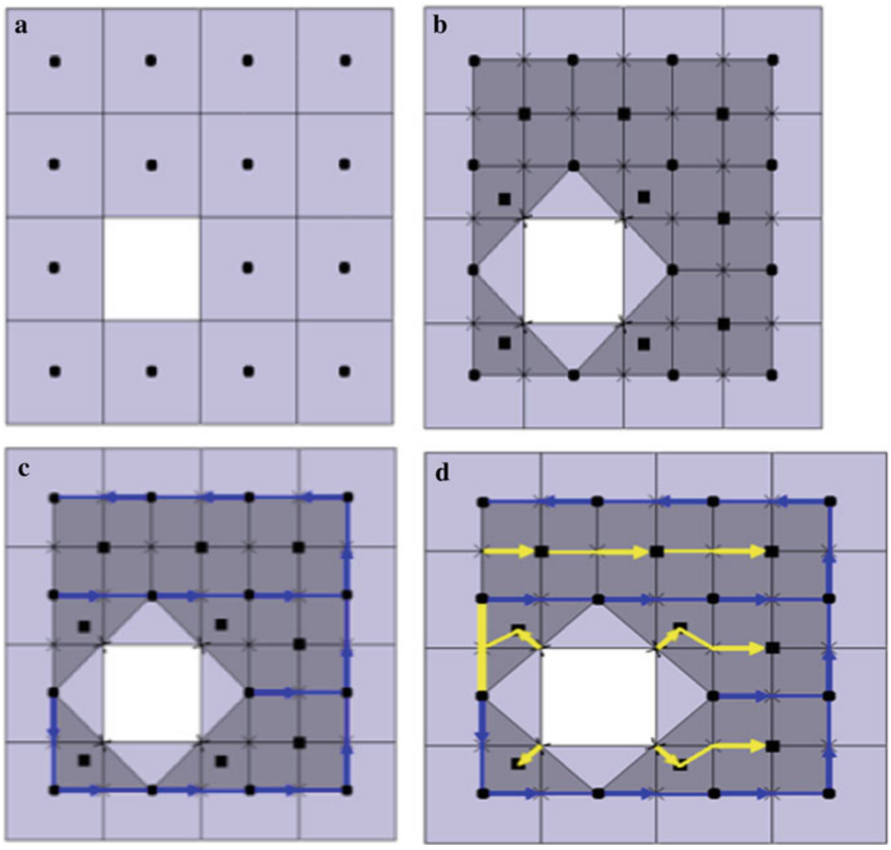


Fig. 1 Example. **a** Left up, **b** right up, **c** left down and **d** right down

circles for the 0-cells, crosses for the 1-cells and solid squares for the 2-cells). The subcomplex formed by the 0 and 1-cells can be seen as a subgraph of the 8-adjacency graph of O. In Fig. 1c, a spanning tree covering all the vertices of the cell complex is specified (in blue). In fact, we consider a subdivision of this tree, having as 0-cells the vertices of the cell complex and the barycenters of the 1-cells belonging to the tree. An arrow in the tree determines the *pairing* of the source (0-cell) and sink (1-cell) cells and, consequently, indicating in this way that both are killed in homology group computation. Let us emphasize that only the top left 0-cell of the complex is not paired. It is a representative cycle (critical 0-cell of the homological process determined by the tree) of the unique connected component that the object has. Finally, in Fig. 1d we also draw the trees (in yellow) *covering* the rest of cells. They are semidirected, with arrows from the barycenters of 1-cells to the barycenter of the 2-cells. In terms of a process for computing homology groups, an arrow also means here that its source and sink cells are both *killed*. There is a edge marked in yellow which is not paired with an arrow. This 1-cell is a representative *critical* cell of the one-dimensional homology generator that the object has.

In this paper, the well-known *spanning tree problem* is studied here in the framework of *Membrane Computing*. This computational paradigm was introduced by Păun in [19,20] and it is based on the assumption that the processes taking place within the compartmental structure of a living cell can be interpreted as computations. The computational devices in Membrane Computing are called *P systems*. Roughly speaking, a P system consists of a membrane structure, in whose compartments one places multisets of objects which evolve according to given rules. These multisets encode the information and the rules deal with them performing the computation. Following a biological inspiration, the multisets of objects represent the chemicals placed in a vesicle of a living cell. Such chemicals are sent to other vesicles or transformed according to biochemical reactions, represented here by computational rules. These rules are usually applied in a synchronous non-deterministic maximally parallel manner, but some other semantics are being explored.[1]

In this paper, we use one of the tools defined in Membrane Computing for the flow of information: the antiport transport of chemicals across biological membranes. In biology, some molecules use the energy stored in the electrochemical gradient of Na^+ or H^+ ions to power the movement of another substance through a membrane in a process called cotransport. When the protein and the ion move in the same direction, the process is called symport; when they move in opposite directions, the process is called antiport. Symport/antiport rules were introduced in Membrane Computing in [18] and nowadays are widely used in the flow of information in the *tissue-like* paradigm. In this paper, we encode the information stored in an image as a set of chemicals placed in a virtual compartment (in a *cell*) and we use symport/antiport rules to deal with this information. We use Membrane Computing *data structures* for representing partial chain homotopy equivalences under a combinatorial graph layer and also Membrane Computing *techniques* to calculate them. In the setting of pixel-based digital 2D binary images, the black connected components can be codified by a spanning forest (one for each component) of the (four or eight) adjacency graph of the set of black pixels.

This is not the first time where life-based methods are applied to Algebraic Topology. In 1996, Chao and Nakayama [2] connected Natural Computing and Algebraic Topology using Neural Networks (extended Kohonen mapping). Some years after, Subramanian et al. presented in [1] two works where Digital Image and Membrane Computing were linked. In 2009, Christinal et al. started a new way where algebraic-topologic process have been parallelized (see [3–5,8]).

The approach presented in this paper can be seen as a new step towards the solution of the problem of obtaining representatives of the 3D homology groups.

The paper is organized as follows: Firstly, we recall the formal framework used in the paper. Next section shows the P systems family that solves the problem of the spanning trees. The paper ends with some conclusions showed in the fourth section.

[1] We refer to [21] for basic information in this area, to [22] for a comprehensive presentation and the web site http://ppage.psystems.eu for the up-to-date information.

2 Formal framework

In the basic definition of cell-like P systems [20], membranes are hierarchically arranged in a tree-like structure. The biological inspiration comes from the morphology of cells, where small vesicles are surrounded by larger ones. In *tissue-like P systems*, the tree-like membrane structure is replaced by a general graph. This model has two biological inspirations (see [16]): 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. Tissue-like P systems have been used to solve computational problems in other areas (see e.g. [6,7]). In this paper we use tissue-like P systems (cells are nodes in a general graph) and the application of the rules are regulated by *promoters*. These promoters have a clear biological inspiration. The rule is applied if the reactants are present, but it is also necessary the presence of the promoter in the corresponding cell. The promoter is not *consumed* nor *produced* by the application of the rule, but if it is not in the cell, the rule cannot be applied. In one step, each object in a membrane can only be used for one rule. In the general case, if there are several possibilities, the rule is non-deterministically chosen, but sometimes we will consider a priority relation between rules, so we need the concept of *priority* in our systems. Next, we recall the formal definition of these P systems.

Definition 1 A *tissue-like P system with promoters and priorities* of degree $q \geq 1$ is a tuple of the form

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

where

1. Γ is a finite alphabet, whose symbols will be called objects;
2. $\Sigma \subseteq \Gamma$ is the input alphabet;
3. $\mathcal{E} \subseteq \Gamma$ is a finite alphabet representing the set of the objects in the environment available in an arbitrary large amount of copies;
4. $w_1, \ldots, w_q$ are strings over Γ representing the multisets of objects associated with the cells in the initial configuration.
5. $\mathcal{R}$ is a finite set of rules of the following form (where 0 is the label for the environment):

   $$(pro \,|\, i, u/v, j), \quad \text{for} \quad 0 \leq i \neq j \leq q, \;\; pro, u, v \in \Gamma^*$$

   The length of a rule $(pro \,|\, i, u/v, j)$ is given by the sum of the number of objects in pro, u and v.
6. Pri is a partial order relation over $\mathcal{R}$. If $R_i > R_j$ and R_i and R_j can be applied, then the application of R_i has *priority* on R_j.
7. $i_{in} \in \{1, 2, \ldots, q\}$ denotes the input region;
8. $i_{out} \in \{0, 1, 2, \ldots, q\}$ denotes the output region.

The rule $(pro \,|\, i, u/v, j)$ can be applied over two cells (or a cell and the environment) i and j such that u (contained in cell i) is traded against v (contained in cell j).

The rule is applied if in i the objects of the promoter pro are present. The promoter is not modified by the application of the rule. If the promoter is empty, we will write $(i, u/v, j)$ instead of $(\emptyset \,|\, i, u/v, j)$.

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 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 multiset of rules. A *configuration* is an instantaneous description of the contents of the cells and it is represented as a tuple $(w_1, \dots, w_q)$. Given a configuration,[2] we can perform a computation step and obtain a new configuration by applying the rules in a parallel manner as it is shown above. A sequence of computation steps is called a *computation*. A configuration is *halting* when no rules can be applied to it.

3 Obtaining a forest of spanning trees

In this section, we present a family of tissue-like P systems with promoters which solves the Spanning Tree problem. It can be stated as follows.

Spanning Tree (STree) Problem: *Given a binary 2D digital image I, generate a spanning tree of each black connected component of I.*

Given a binary image I, firstly, we must identify the number of black connected components of it. One pixel of each component will be root of the future spanning tree of the component.

In order to provide a logarithmic-time uniform solution to the STree problem, we design a family of tissue-like P systems with promoters, $\{\Pi(n)\}_{n\in\mathbb{N}}$. Given a binary image I of size n^2 we codify each pixel by an object a_{ij}, where $a = b \wedge w$ and i, j are odd numbers. As usual, b and w stands for $black$ and $white$. Images of size n^2 will be dealt with the P system $\Pi(n)$ from the family. Next, we construct, in a parallel way, one spanning tree by considering each connected component as a connected graph.

If $a = b$, we say there is a vertex of a black connected component. If two pixels a_{ij}, a'_{kl} are adjacent, then there is an edge (b_{rs}) connecting them, where one of r or s is an odd number and the other is an even one. Moreover, ($r = i + 1$, $k = i + 2$, and $j = l$) or ($s = j + 1$, $l = j + 2$ and $i = k$). If the considered two neighbor vertices are not adjacent, then we write w_{rs}. The input data (image I) is codified by a set of objects A_{ij} with $A = B \vee W$, $1 \le i, j \le 2n - 1$ and i, j are odd. We can see an example in Fig. 2.

The family of P systems is defined as follows:

$$\Pi(n) = (\Gamma, \Sigma, \mathcal{E}, \omega_1, \omega_2, \omega_3, \mathcal{R}_1 \cup \cdots \cup \mathcal{R}_{21}, Pri, i_{in}, i_{out})$$

where:

1. $\Gamma = \{z_i : 1 \le i \le 2n + 5\} \cup$
   $\{B_{ij}, b_{ij}, b'_{ij}, \bar{b}_{ij}, W_{ij}, w_{ij}, w'_{ij}, (b_{ij}, (k, l)), (w_{ij}, (k, l)) :$

[2] The objects from $\Gamma - \mathcal{E}$ placed in the environment along the computation are not explicitely showed in the configuration since they are not relevant in this approach.

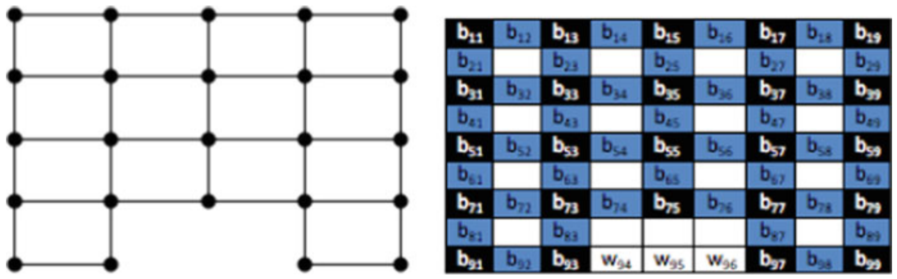

b11 b12 b13 b14 b15 b16 b17 b18 b19
b21 b23 b25 b27 b29
b31 b32 b33 b34 b35 b36 b37 b38 b39
b41 b43 b45 b47 b49
b51 b52 b53 b54 b55 b56 b57 b58 b59
b61 b63 b65 b67 b69
b71 b72 b73 b74 b75 b76 b77 b78 b79
b81 b83 b87 b89
b91 b92 b93 w94 w95 w96 b97 b98 b99

Fig. 2 Example: input graph (image) and codified input for membrane system

$1 \leq i, j, k, l \leq 2n - 1\} \cup$
$\{A_{ijkl} : (1, 1) \leq (i, j) < (k, l) \leq (2n - 1, 2n - 1)\}$,

2. $\Sigma = \{A_{ij} : A = B \wedge W,\ 1 \leq i, j \leq 2n - 1\}$,
3. $\mathcal{E} = \Gamma - \Sigma$,
4. $\omega_1 = \emptyset, \omega_2 = z_1, \omega_3 = \emptyset$,
5. The sets of rules are $\mathcal{R}_1 \cup \cdots \cup \mathcal{R}_{21}$. It will be described below.
6. The set Pri has the following relations:
   - $\mathcal{R}_8 > \{\mathcal{R}_6, \mathcal{R}_7\}$, which denote that any rule from the set $\mathcal{R}_8$ has priority on any rule from $\mathcal{R}_6 \cup \mathcal{R}_7$.
   - $R_{12}, R_{13} > \{R_{14}, \ldots, R_{19}\}$
7. $i_{in} = 1$ is the input cell.
8. $i_{out} = 3$ is the output cell.

The rules are the following:

- Rules associated to the input:
  - $R_1 \equiv (1, A_{ij}/a_{ij}a'_{ij}, 0)$ for $1 \leq i, j \leq 2n - 1$.

  Initially, the image is encoded as a set of pixels A_{ij} placed on cell 1. The P system uses this set of rules to generate two copies of our pixels in the cell 1.

  - $R_2 \equiv (1, a_{ij}/\lambda, 2)$ for $1 \leq i, j \leq 2n - 1$.
  - $R_3 \equiv (1, a'_{ij}/\lambda, 3)$ for $1 \leq i, j \leq 2n - 1$.

  The sets of rules 2 and 3 send one copy of the input image to the cell 2 and another to the cell 3.

- Rules associated to the process of computing homology groups (H_0 process):
  - $R_4 \equiv (2, z_i/z_{i+1}, 0)$ for $1 \leq i \leq n + 1$.

The set of objects z_i can be considered as a counter. The object z_1 is placed in cell 2 in the initial configuration. This set of rules deals with this counter. The object z_i is traded against z_{i+1} in each computation step.

  - $R_5 \equiv (2, b_{ij}/(b_{ij}, (i, j)), 0)$ for $1 \leq i, j \leq 2n - 1$.

  The set of rules R_5 deals with black pixels b_{ij}. These objects are substituted by more complex ones. These new objects have two parts: the first one keeps the information b_{ij}. The second one can be considered as a label associated to the object. This label has initially the value (i, j).

∘ $R_6 \equiv (2, (b_{ij}, (k,l))(b_{i'j'}, (k',l'))/(b_{ij}, (k,l))(b_{i'j'}, (k,l))A_{klk'l'}, 0)$ for $(1,1) \le (k,l) < (k',l') \le (2n-1, 2n-1)$, $1 \le i, j, i', j' \le 2n-1$ and $(i, j), (i', j')$ adjacent vertices.

∘ $R_7 \equiv (2, (b_{ij}, (k,l))(b_{i'j'}, (k',l'))/(b_{ij}, (k',l'))(b_{i'j'}, (k',l'))A_{k'l'kl}, 0)$ for $(1,1) \le (k',l') < (k,l) \le (2n-1, 2n-1)$, $1 \le i, j, i', j' \le 2n-1$ and $(i, j), (i', j')$ adjacent vertices.

The sets of rules R_6 and R_7 interchange two objects from cell 2 with 3 objects from the environment. If the objects b_{ij} and $b_{i'j'}$ represent adjacent pixels in the image, then, the objects $(b_{ij}, (k,l))$ and $(b_{i'j'}, (k',l'))$ from the cell 2 are replaced by $(b_{ij}, (k,l))$ and $(b_{i'j'}, (k,l))$ together with $A_{klk'l'}$.

∘ $R_8 \equiv (A_{ijkl}|2, (b_{i'j'}, (k,l))/(b_{i'j'}, (i,j)), 0)$ for $1 \le i, j, k, l, i', j' \le 2n-1$.

This is the first set of rules where promoters are used. In a close analogy with biological enzymes, promoters make the process faster. The promoter is created when the pixel labeled by (k,l) changes its label for (i, j). So, (i, j) and (k,l) are labels of adjacent pixels and other pixels with these labels can change their label. Let us recall that $\mathcal{R}_8 > \{\mathcal{R}_6, \mathcal{R}_7\}$, so the P system applies rules from $\mathcal{R}_8$ before than from $\mathcal{R}_6 \cup \mathcal{R}_7$. So, the P system firstly uses the rules with promoter and after the rules without them.

∘ $R_9 \equiv (z_{n+2}|2, (b_{ij}, (i,j))/0_{ij}z_{n+3}, 0)$ for $1 \le i, j \le n$.

The P system uses this rules to bring the objects 0_{ij}, codifying the roots of the spanning trees, and the objects z_{n+3} to the cell 2.
Notice that z_{n+2} is used as a promoter.

∘ $R_{10} \equiv (2, 0_{ij}z_{n+3}/b'_{ij}, 3)$ for $1 \le i, j \le n$.

One pixel from each connected component is sent to the cell 3. In this way one pixel is marked as the root for each spanning tree which will be constructed in the next steps.

– Rules associated to the construction stage:

∘ $R_{11} \equiv (3, z_i/z_{i+1}, 0)$ for $n+3 \le i \le 2n+9$.

The set of rules R_{10} sends the counter z_i to cell 3. This new set of rules controls the counter in this new cell.

Note 1 In the description of the next sets of rules we will consider a pictorial notation (see [1]). These rules can be technically described by adjacency terms, but we adopt this representation for a clearer exposition. We denote objects codifying black pixels (b'_{ij}) as objects b, objects codifying white pixels (w'_{ij}) as objects w and objects k_{ij} (for example 0_{11} or 3_{57}) as object k (0 or 3).

Note 2 We denote by **b** the objects codifying vertices, by b the objects codifying edges, by k ($k \in \mathbb{N}$) the objects codifying edges of a spanning tree (i.e., the P system uses natural numbers to mark edges belonging to a spanning tree), and by $\overline{b}$ the edges not belonging to spanning trees, i.e., edges eliminate as candidate to be part of a spanning tree.

$$\circ\ R_{12} \equiv \left(3, \begin{array}{ccc} \mathbf{b} & b & \mathbf{b} \\ b & & b \\ \mathbf{b} & b & \mathbf{b} \\ b & & b \end{array} \Big/ \begin{array}{ccc} \mathbf{b} & b & \mathbf{b} \\ b & & b \\ \mathbf{b} & \bar{b} & \mathbf{b} \\ b & & b \end{array}, 0\right), \quad R_{13} \equiv \left(3, \begin{array}{ccc} \mathbf{b} & \bar{b} & \mathbf{b} \\ b & & b \\ \mathbf{b} & b & \mathbf{b} \\ b & & b \end{array} \Big/ \begin{array}{ccc} \mathbf{b} & \bar{b} & \mathbf{b} \\ b & & b \\ \mathbf{b} & \bar{b} & \mathbf{b} \\ b & & b \end{array}, 0\right),$$

There are 4 rules of each one of the last two sets of rules, depending on where objects $\bar{b}$ appear. Defining these rules, we eliminate the minimal rules and our algorithm is more efficient. We will also consider the priorities $R_{12}, R_{13} > \{R_{14}, \ldots, R_{19}\}$, i.e., the P system eliminates a lot of edges that cannot belong to the spanning tree, before it begins to add edges to the spanning trees and mark edges like $\bar{b}$. In fact, the P system eliminates all the minimal cycles before the objects 0_{ij} arrive to cell 3.

$$\circ\ R_{14} \equiv \left(3, \begin{array}{ccc} 0 & \mathbf{b} & b \\ & b & \end{array} \Big/ \begin{array}{ccc} 0 & \mathbf{b} & 1 \\ & 1 & \end{array}, 0\right), \quad R_{15} \equiv \left(3, \begin{array}{ccc} 0 & \mathbf{b} & b \\ & t & \end{array} \Big/ \begin{array}{ccc} 0 & \mathbf{b} & 1 \\ & t & \end{array}, 0\right),$$

where $t = w$ or $\bar{b}$. There are two sets of rules depending on the position of t. These two types of rules identify the first edges of our spanning trees. The P system uses rules of types from R_{16} to R_{19} to look for specific patterns in the image. So, it is possible to add new edges to the spanning trees.

$$\circ\ R_{16} \equiv \left(3, \begin{array}{cccc} & & b & \\ \mathbf{t} & k & \mathbf{b} & b \\ & & b & \end{array} \Big/ \begin{array}{cccc} & & (k+1) & \\ \mathbf{b} & k & \mathbf{b} & (k+1) \\ & & (k+1) & \end{array}, 0\right),$$

where $\mathbf{t} = \mathbf{b}$ or 0. There are 4 sets of rules of this type. These types of rules are used to add new edges to our spanning trees. Given an edge codified by b_{ij}, the P system changes b by a natural number, $k+1$, obtaining $(k+1)_{ij}$.

$$\circ\ R_{17} \equiv \left(3, \begin{array}{cccc} & & t & \\ \mathbf{b} & k & \mathbf{b} & b \\ & & b & \end{array} \Big/ \begin{array}{cccc} & & t & \\ \mathbf{b} & k & \mathbf{b} & (k+1) \\ & & (k+1) & \end{array}, 0\right),$$

where $t = w$ or $\bar{b}$ or empty. There are 12 sets of rules of this type. These types of rules are used to add new edges to our spanning trees. Given an edge codified by b_{ij}, the P system changes b by a natural number, $k+1$, obtaining $(k+1)_{ij}$.

$$\circ\ R_{18} \equiv \left(3, \begin{array}{cccc} & & t & \\ \mathbf{b} & k & \mathbf{b} & t \\ & & b & \end{array} \Big/ \begin{array}{cccc} & & t & \\ \mathbf{b} & k & \mathbf{b} & t \\ & & (k+1) & \end{array}, 0\right),$$

where $t = w$ or $\bar{b}$ or empty. There are 14 sets of rules of this type. These types of rules are also used to add new edges to our spanning trees. Given an edge codified by b_{ij}, system changes b by a natural number, $k+1$, obtaining $(k+1)_{ij}$.

$$\circ\ R_{19} \equiv \left(3, \begin{array}{ccc} & k_1 & \\ t & \mathbf{b} & t \\ & b & \\ k_2 & \mathbf{b} & t \\ & t & \end{array} \Big/ \begin{array}{ccc} & k_1 & \\ t & \mathbf{b} & t \\ & \bar{b} & \\ k_2 & \mathbf{b} & t \\ & t & \end{array}, 0\right),$$

where $t = b$ or $\bar{b}$ or w or a natural number or it does not exist. It is not important distinguish one of these cases. We use $\bar{b}$ for edges than they cannot belong to any spanning tree. There are 9 rules of this type, depending the position of the numbers k_1 and k_2, and k_1 and k_2 cannot codify adjacent edges. These types of rules are used to avoid cycles when we are finding spanning trees. $R_{19} > \{R_{16}, R_{17}, R_{18}\}$, i.e., the P system firstly applies the rules to eliminate cycles and after apply the rules to add new edges to the spanning trees.

– Rules associated to the output:

  ∘ $R_{20} \equiv (z_{2n+10}|3, w_{ij}/\lambda, 0)$, $R_{21} \equiv (z_{2n+10}|3, \overline{b}_{ij}/\lambda, 0)$

for $1 \leq i, j \leq 2n - 1$. The P system sends to the environment white pixels and edges which have not been taken to construct the spanning forest.

The STree problem will be solved by $\Pi(n)$ according with the following stages:

1. *Input Stage*: When the objects A_{ij} (with $A = B \vee W$) arrive to cell 1 the first type of rules are applied. These objects are changed by objects a_{ij} and a'_{ij} (with $a = b \vee w$). In the next step, objects a_{ij} are sent to cell 2 and objects a'_{ij} to cell 3. The process occurs in cell 2, and it is dedicated to obtain the number of black connected components (H_0).
2. H_0 *Process*: It begins when objects a_{ij} arrive to cell 2.
   (a) *Label Allocation Stage*: Cell 2 trades objects b_{ij} against others with the form $(b_{ij}, (i, j))$ from the environment. White objects are not transformed.
   (b) *Label Conversion Stage*: We can compare the black adjacent pixels by using promoter, and we trade the label of the greatest pixel against the label of the other pixel, as we can see in the following rules:

$$(i, (b_{ij}, (i', j'))(b_{kl}, (k', l'))/(b_{ij}, (i', j'))(b_{kl}, (i', j'))A_{i'j'k'l'}, j)$$

   where (i, j) and (k, l) are adjacent pixels. Moreover, we can see a new object ($A_{i'j'k'l'}$ in this case) arriving to cell i. It is a promoter and it is used to codify two labels have been compared and, they are connected. So, one of them is changed by the other one.
   (c) *Identifying Roots (of connected component) Stage*: In the step $n+2$, the object z_{n+2} arrives to the cell 1 due to the counter. It is used by the system as a promoter, and the objects with the form $(b_{ij}, (i, j))$ are marked as 0_{ij} representing the root of a black connected component. The P system has used $n + 2$ steps to obtain the roots of black connected components of an n^2 image.
3. *Construction (of Spanning tree) Stage*: In this step, for each root (which the P system identified in the previous stage), new edges are added to each spanning tree. The cycles inside of the black connected components are eliminated and order the nodes of the generated trees. To identify the edges b_{ij} of our spanning trees we change the object b by a natural number. So, we will obtain an order in our trees.
4. *Answer Stage*: Using a counter, the P system can bring a promoter to the work cell to send to the output cell the objects codifying the nodes of the spanning trees generated.

3.1 Example

In this section we illustrate with a simple example our technique to obtain homology groups H_0, and a spanning tree from H_0 of a $2D$ image by using Membrane Computing. Let us consider the right image that appear in Fig. 2 which has 9×9 pixels.

Initially, we have three cells whose multisets are empty excepting $w_2 = z_1$ and, we introduce the input of the system in the cell with label 2. So, two types of rules are applied in the first step of computation. The rules R_1 generate two copies of the pixels (a_{ij}, a'_{ij}) and the rules R_4 activate a counter to generate the object z_{10} in the eleventh step.

When the objects a_{ij}, a'_{ij} arrive to cell 1 rules R_2 and R_3 are applied, and the first type of objects are sent to the cell 2 and the second type are sent to the cell 3 in the second step of the computation.

In this point, the stage to calculate the homology groups of type H_0 begins. We need to know how many black connected components appear in our image and label one representative object of each component.

In the third step, the P system permutes objects b_{ij} by objects $(b_{ij}, (i, j))$ using rules of type R_5 (see third configuration in Fig. 3). So, we associate to each object a new label. Initially, this label is the pixel (i, j) associated to the element and later, system will can change this label to each object. In fact, in the next step rule of type R_6 and R_7 are applied. System change the labels associated to a lot of objects as we can see in the Fig. 3. Moreover, we generate the following objects: $A_{1113}, A_{1517}, A_{1939}, A_{3133}, A_{3537}, \quad A_{5153}, A_{5557}, A_{5979}, A_{7173}, A_{7577}, A_{9193}, A_{9799}$ that the P system can use as promoters in following steps of computation. For example, we can see how the following rule works:

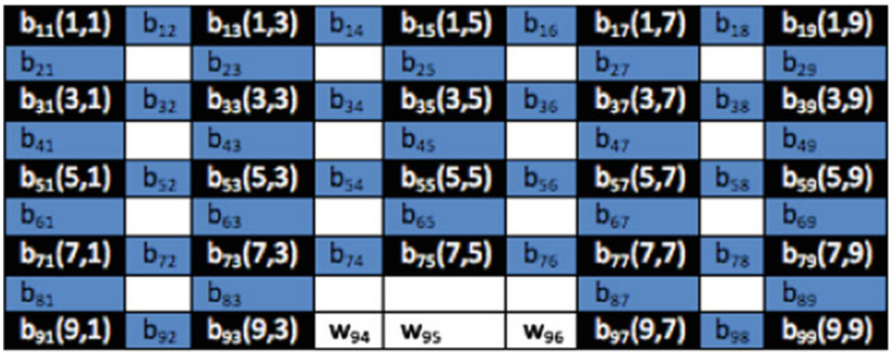
$$(2, (b_{11}, (1, 1))(b_{13}, (1, 3))/(b_{11}, (1, 1))(b_{13}, (1, 1))A_{1113}, 0)$$

the P system trades the object $(b_{13}, (1, 3))$ against to the objects $(b_{13}, (1, 1))A_{1113}, 0)$ as we can see in Fig. 3.

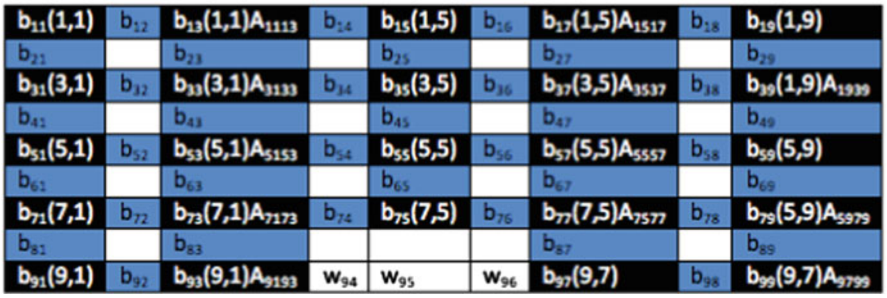
From the moment when the promoters appear in the cell 2, the P system can use three types of rules R_6, R_7 and R_8. The last type of rules needs specific promoters to become active. We must remember the last type has priority with respect to the first two types. So, when the P system changes the label of an object, it must use a promoter. But, if we examine to the Fig. 3, we can see that no rule of type R_8 can be applied in this step, because we have an only object associate to each label. We have to wait to the following steps to see how the promoters are used. We can see how the following rule of type R_8 is applied in the fifth step of computation:

$$(A_{1519}|2, (b_{ij}, (1, 9))/(b_{ij}, (1, 5)), 0), \quad \text{para} \quad 1 \le i, j \le 9$$

In cell 2, we have objects $(b_{37}, (1, 9))$, $(b_{39}, (1, 9))$, $(b_{59}, (1, 9))$ and others (as we can see in first image of Fig. 4). Then, when system applies this rule, our objects are changed by $(b_{37}, (1, 5))$, $(b_{39}, (1, 5))$, $(b_{59}, (1, 5))$.

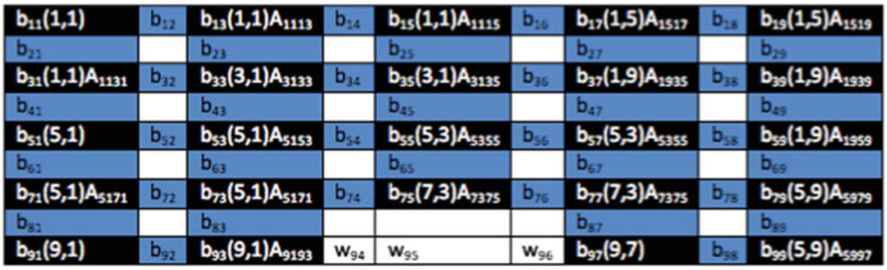


$b_{11}(1,1)$	b_{12}	$b_{13}(1,3)$	b_{14}	$b_{15}(1,5)$	b_{16}	$b_{17}(1,7)$	b_{18}	$b_{19}(1,9)$
b_{21}		b_{23}		b_{25}		b_{27}		b_{29}
$b_{31}(3,1)$	b_{32}	$b_{33}(3,3)$	b_{34}	$b_{35}(3,5)$	b_{36}	$b_{37}(3,7)$	b_{38}	$b_{39}(3,9)$
b_{41}		b_{43}		b_{45}		b_{47}		b_{49}
$b_{51}(5,1)$	b_{52}	$b_{53}(5,3)$	b_{54}	$b_{55}(5,5)$	b_{56}	$b_{57}(5,7)$	b_{58}	$b_{59}(5,9)$
b_{61}		b_{63}		b_{65}		b_{67}		b_{69}
$b_{71}(7,1)$	b_{72}	$b_{73}(7,3)$	b_{74}	$b_{75}(7,5)$	b_{76}	$b_{77}(7,7)$	b_{78}	$b_{79}(7,9)$
b_{81}		b_{83}				b_{87}		b_{89}
$b_{91}(9,1)$	b_{92}	$b_{93}(9,3)$	w_{94}	w_{95}	w_{96}	$b_{97}(9,7)$	b_{98}	$b_{99}(9,9)$

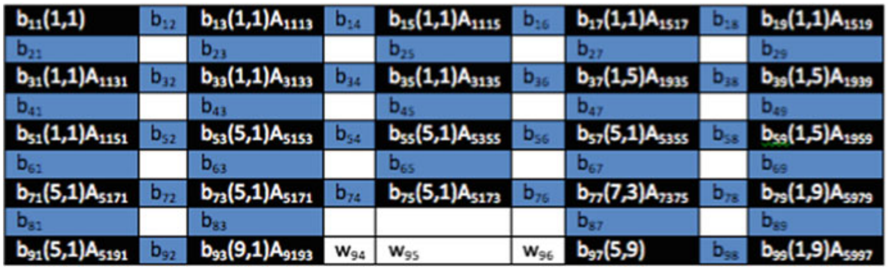


$b_{11}(1,1)$	b_{12}	$b_{13}(1,1)A_{1113}$	b_{14}	$b_{15}(1,5)$	b_{16}	$b_{17}(1,5)A_{1517}$	b_{18}	$b_{19}(1,9)$
b_{21}		b_{23}		b_{25}		b_{27}		b_{29}
$b_{31}(3,1)$	b_{32}	$b_{33}(3,1)A_{3133}$	b_{34}	$b_{35}(3,5)$	b_{36}	$b_{37}(3,5)A_{3537}$	b_{38}	$b_{39}(1,9)A_{1939}$
b_{41}		b_{43}		b_{45}		b_{47}		b_{49}
$b_{51}(5,1)$	b_{52}	$b_{53}(5,1)A_{5153}$	b_{54}	$b_{55}(5,5)$	b_{56}	$b_{57}(5,5)A_{5557}$	b_{58}	$b_{59}(5,9)$
b_{61}		b_{63}		b_{65}		b_{67}		b_{69}
$b_{71}(7,1)$	b_{72}	$b_{73}(7,1)A_{7173}$	b_{74}	$b_{75}(7,5)$	b_{76}	$b_{77}(7,5)A_{7577}$	b_{78}	$b_{79}(5,9)A_{5979}$
b_{81}		b_{83}				b_{87}		b_{89}
$b_{91}(9,1)$	b_{92}	$b_{93}(9,1)A_{9193}$	w_{94}	w_{95}	w_{96}	$b_{97}(9,7)$	b_{98}	$b_{99}(9,7)A_{9799}$

Fig. 3 Image in configurations 3 and 4

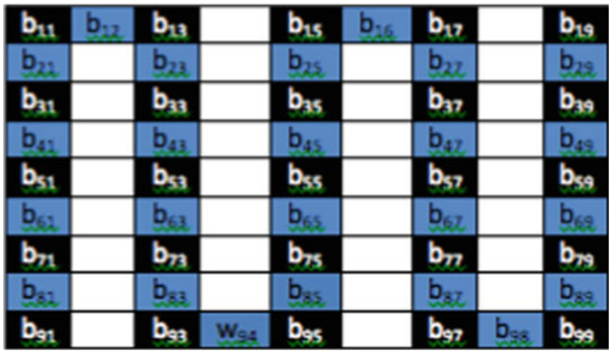


$b_{11}(1,1)$	b_{12}	$b_{13}(1,1)A_{1113}$	b_{14}	$b_{15}(1,1)A_{1115}$	b_{16}	$b_{17}(1,5)A_{1517}$	b_{18}	$b_{19}(1,5)A_{1519}$
b_{21}		b_{23}		b_{25}		b_{27}		b_{29}
$b_{31}(1,1)A_{1131}$	b_{32}	$b_{33}(3,1)A_{3133}$	b_{34}	$b_{35}(3,1)A_{3135}$	b_{36}	$b_{37}(1,9)A_{1935}$	b_{38}	$b_{39}(1,9)A_{1939}$
b_{41}		b_{43}		b_{45}		b_{47}		b_{49}
$b_{51}(5,1)$	b_{52}	$b_{53}(5,1)A_{5153}$	b_{54}	$b_{55}(5,3)A_{5355}$	b_{56}	$b_{57}(5,3)A_{5355}$	b_{58}	$b_{59}(1,9)A_{1959}$
b_{61}		b_{63}		b_{65}		b_{67}		b_{69}
$b_{71}(5,1)A_{5171}$	b_{72}	$b_{73}(5,1)A_{5171}$	b_{74}	$b_{75}(7,3)A_{7375}$	b_{76}	$b_{77}(7,3)A_{7375}$	b_{78}	$b_{79}(5,9)A_{5979}$
b_{81}		b_{83}				b_{87}		b_{89}
$b_{91}(9,1)$	b_{92}	$b_{93}(9,1)A_{9193}$	w_{94}	w_{95}	w_{96}	$b_{97}(9,7)$	b_{98}	$b_{99}(5,9)A_{5997}$



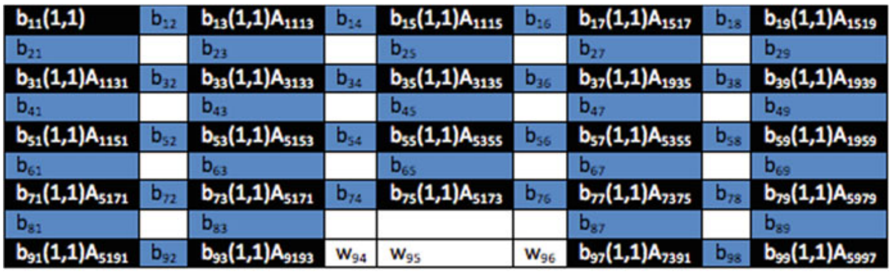
$b_{11}(1,1)$	b_{12}	$b_{13}(1,1)A_{1113}$	b_{14}	$b_{15}(1,1)A_{1115}$	b_{16}	$b_{17}(1,1)A_{1517}$	b_{18}	$b_{19}(1,1)A_{1519}$
b_{21}		b_{23}		b_{25}		b_{27}		b_{29}
$b_{31}(1,1)A_{1131}$	b_{32}	$b_{33}(1,1)A_{3133}$	b_{34}	$b_{35}(1,1)A_{3135}$	b_{36}	$b_{37}(1,5)A_{1935}$	b_{38}	$b_{39}(1,5)A_{1939}$
b_{41}		b_{43}		b_{45}		b_{47}		b_{49}
$b_{51}(1,1)A_{1151}$	b_{52}	$b_{53}(5,1)A_{5153}$	b_{54}	$b_{55}(5,1)A_{5355}$	b_{56}	$b_{57}(5,1)A_{5355}$	b_{58}	$b_{59}(1,5)A_{1959}$
b_{61}		b_{63}		b_{65}		b_{67}		b_{69}
$b_{71}(5,1)A_{5171}$	b_{72}	$b_{73}(5,1)A_{5171}$	b_{74}	$b_{75}(5,1)A_{5173}$	b_{76}	$b_{77}(7,3)A_{7375}$	b_{78}	$b_{79}(1,9)A_{5979}$
b_{81}		b_{83}				b_{87}		b_{89}
$b_{91}(5,1)A_{5191}$	b_{92}	$b_{93}(9,1)A_{9193}$	w_{94}	w_{95}	w_{96}	$b_{97}(5,9)$	b_{98}	$b_{99}(1,9)A_{5997}$

Fig. 4 Image in configurations 5 and 6

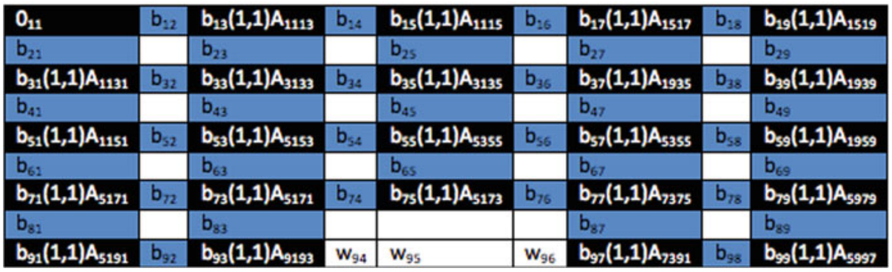


b_{11}	b_{12}	b_{13}		b_{15}	b_{16}	b_{17}		b_{19}
b_{21}		b_{23}		b_{25}		b_{27}		b_{29}
b_{31}		b_{33}		b_{35}		b_{37}		b_{39}
b_{41}		b_{43}		b_{45}		b_{47}		b_{49}
b_{51}		b_{53}		b_{55}		b_{57}		b_{59}
b_{61}		b_{63}		b_{65}		b_{67}		b_{69}
b_{71}		b_{73}		b_{75}		b_{77}		b_{79}
b_{81}		b_{83}		b_{85}		b_{87}		b_{89}
b_{91}		b_{93}	W_{94}	b_{95}		b_{97}	b_{98}	b_{99}

Fig. 5 Worst case



$b_{11}(1,1)$	b_{12}	$b_{13}(1,1)A_{1113}$	b_{14}	$b_{15}(1,1)A_{1115}$	b_{16}	$b_{17}(1,1)A_{1517}$	b_{18}	$b_{19}(1,1)A_{1519}$
b_{21}		b_{23}		b_{25}		b_{27}		b_{29}
$b_{31}(1,1)A_{1131}$	b_{32}	$b_{33}(1,1)A_{3133}$	b_{34}	$b_{35}(1,1)A_{3135}$	b_{36}	$b_{37}(1,1)A_{1935}$	b_{38}	$b_{39}(1,1)A_{1939}$
b_{41}		b_{43}		b_{45}		b_{47}		b_{49}
$b_{51}(1,1)A_{1151}$	b_{52}	$b_{53}(1,1)A_{5153}$	b_{54}	$b_{55}(1,1)A_{5355}$	b_{56}	$b_{57}(1,1)A_{5355}$	b_{58}	$b_{59}(1,1)A_{1959}$
b_{61}		b_{63}		b_{65}		b_{67}		b_{69}
$b_{71}(1,1)A_{5171}$	b_{72}	$b_{73}(1,1)A_{5171}$	b_{74}	$b_{75}(1,1)A_{5173}$	b_{76}	$b_{77}(1,1)A_{7375}$	b_{78}	$b_{79}(1,1)A_{5979}$
b_{81}		b_{83}				b_{87}		b_{89}
$b_{91}(1,1)A_{5191}$	b_{92}	$b_{93}(1,1)A_{9193}$	W_{94}	W_{95}	W_{96}	$b_{97}(1,1)A_{7391}$	b_{98}	$b_{99}(1,1)A_{5997}$



0_{11}	b_{12}	$b_{13}(1,1)A_{1113}$	b_{14}	$b_{15}(1,1)A_{1115}$	b_{16}	$b_{17}(1,1)A_{1517}$	b_{18}	$b_{19}(1,1)A_{1519}$
b_{21}		b_{23}		b_{25}		b_{27}		b_{29}
$b_{31}(1,1)A_{1131}$	b_{32}	$b_{33}(1,1)A_{3133}$	b_{34}	$b_{35}(1,1)A_{3135}$	b_{36}	$b_{37}(1,1)A_{1935}$	b_{38}	$b_{39}(1,1)A_{1939}$
b_{41}		b_{43}		b_{45}		b_{47}		b_{49}
$b_{51}(1,1)A_{1151}$	b_{52}	$b_{53}(1,1)A_{5153}$	b_{54}	$b_{55}(1,1)A_{5355}$	b_{56}	$b_{57}(1,1)A_{5355}$	b_{58}	$b_{59}(1,1)A_{1959}$
b_{61}		b_{63}		b_{65}		b_{67}		b_{69}
$b_{71}(1,1)A_{5171}$	b_{72}	$b_{73}(1,1)A_{5171}$	b_{74}	$b_{75}(1,1)A_{5173}$	b_{76}	$b_{77}(1,1)A_{7375}$	b_{78}	$b_{79}(1,1)A_{5979}$
b_{81}		b_{83}				b_{87}		b_{89}
$b_{91}(1,1)A_{5191}$	b_{92}	$b_{93}(1,1)A_{9193}$	W_{94}	W_{95}	W_{96}	$b_{97}(1,1)A_{7391}$	b_{98}	$b_{99}(1,1)A_{5997}$

Fig. 6 Image in configurations 9 and 11

In fact, until ninth step of computation only these three types of rules are applied. We can see in Fig. 6 how only one label appear in objects of the cell 2 codifying black vertices in configuration 9.

In the next step, the P system only applies the last rule of type R_4, bringing the object z_{11} from the environment to cell 2. The P system executes this step because it can be necessary in the worst case, which we can see in Fig. 5.

When the object z_{11} appears in cell 2, rule $(z_{11}|2, (b_{11}, (1,1))/0_{11}z_{12}, 0)$ is applied. Then, we can find objects $0_{11}, z_{12}$ in cell 2, and the P system applies rule $(2, 0_{11}z_{12}/b'_{11}, 3)$. On this way, the root of the unique spanning tree of our image arrives to cell 3 in configuration 12, and the construction (of spanning tree) stage begins.

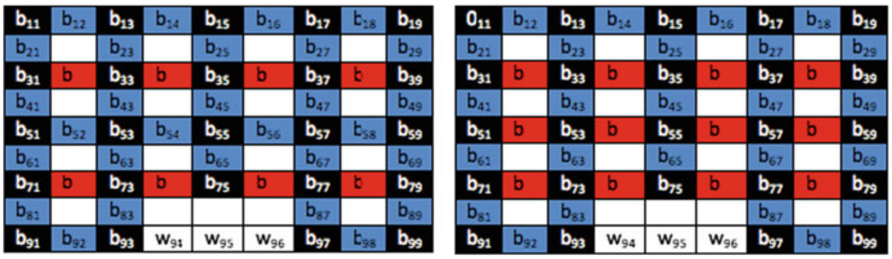


Fig. 7 Image in configurations 13 and 14

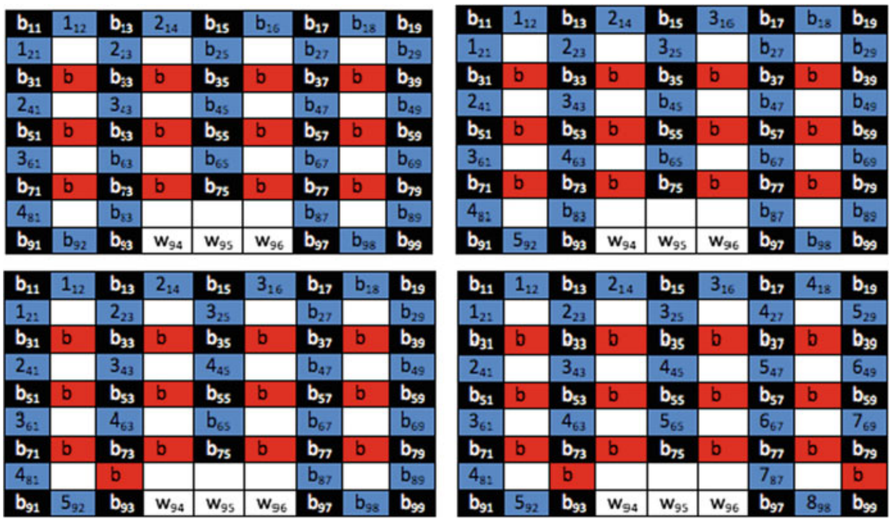


Fig. 8 Image in configurations 19 (*left up*), 20 (*right up*), 21 (*left down*) and 27 (*right down*)

From here, the P system divides the processes in two threads. In the first one, it uses the counter z_i to generate the object z_{28}, that we use to generate the output of the P system. In the other, the P system constructs a spanning tree of our binary image.

To the second thread, the P system applies rules of type R_{12}, as we can see in Fig. 7. In the next step, the system uses rules of type R_{13}, so we arrive to the configuration 14 (see Fig. 7).

In the step 15, the P system takes a rule of type R_{14} to add first two edges to our spanning tree. In the following steps, the P system applies rules of types R_{16}, R_{17}, R_{18}. For example, when system arrives to the configuration 19, the following rules of type R_{18} can be applied between others (see Fig. 8):

$$\left(3,\ \begin{array}{cc} \mathbf{b} & \\ 4 & \\ \mathbf{b} & b \end{array} \Big/ \begin{array}{cc} \mathbf{b} & \\ 4 & \\ \mathbf{b} & 5 \end{array},0\right), \left(3,\ \begin{array}{c} \mathbf{b} \\ 3 \\ \mathbf{b} \\ b \end{array} \Big/ \begin{array}{c} \mathbf{b} \\ 3 \\ \mathbf{b} \\ 4 \end{array},0\right)$$

So, we can obtain configuration 20 (see Fig. 8). Here, a new type of rules can be used, rules R_{19}. These rules are defined to avoid cycles. In fact the following rule

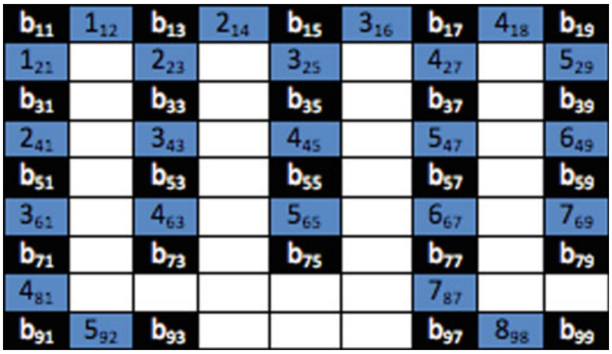


Fig. 9 Output of the system

$$\left(3,\ \begin{array}{cc} \mathbf{b} & 4 \\ & \mathbf{b} \\ & b \\ 5 & \mathbf{b} \end{array} \Big/ \begin{array}{cc} \mathbf{b} & 4 \\ & \mathbf{b} \\ & \overline{b} \\ 5 & \mathbf{b} \end{array}, 0\right)$$

is applied to arrive configuration 21. Then, the P system, using four types of rules, generates a spanning tree in step 27 (see right down image in Fig. 8).

The output cell is cell with label 3, so with help of the rules of type R_{20}, R_{21} system sends the objects not necessary to the environment, i.e. the P system sends the objects $w_{ij}, \overline{b}_{ij}$ to the environment. To this aim, the P system uses promoter z_{28}. The the output of the system is given in Fig. 9.

3.2 Complexity and necessary resources

Bearing in mind the size of the input data is $O(n^2)$, the amount of necessary resources for defining the P systems of our family and the complexity of our problems can be observed in the following table

Number of steps of a computation	$2n + 11$
Necessary resources	
Size of the alphabet	$O(n^4)$
Initial number of cells	3
Initial number of objects	$O(n^2)$
Number of rules	$O(n^6)$
Upper bound for the length of rules of the P systems	20

4 Conclusions

Using Effective Homology Theory as main tool for designing algorithms for computing complexes topological invariants (cohomology ring, (co)homology operations, homotopy groups,...), the problem of decomposing the objects into combinatorial graph-like pieces appears in a natural way. A possible solution to solve the high complexity costs of these processes is provided here by Membrane Computing. Within the Digital Imagery setting, we determine here an Membrane Computing strategy for partially specifying a chain homotopy operator at level of pixels for a pixel-based digital 2D binary object O. This fundamental data structure in Effective Homology is obtained in terms of a forest spanning every vertex of its associated adjacency graph. Every tree of this forest determine and *localize* the corresponding connected component of O. The possibility to extend this results and others to higher dimensions and to more general cell complexes is a goal of our research in a near future.

Acknowledgments DDP and MAGN acknowledge the support of the projects TIN2008-04487-E and TIN-2009-13192 of the Ministerio de Ciencia e Innovación of Spain and the support of the Project of Excellence with *Investigador de Reconocida Valía* of the Junta de Andalucía, grant P08-TIC-04200. PR and HAC acknowledge the support of the project MTM2006-03722 of the Ministerio español de Educación y Ciencia and the project PO6-TIC-02268 of Excellence of Junta de Andalucía.

References

1. Ceterchi, R., Mutyam, M., Păun, G., Subramanian, K.G.: Array-rewriting P systems. Nat. Comput. **2**(3), 229–249 (2003)
2. Chao, J., Nakayama, J.: Cubical singular simplex model for 3D objects and fast computation of homology groups. In: 13th International Conference on Pattern Recognition (ICPR'96), vol. IV, pp. 190–194. IEEE Computer Society, IEEE Computer Society, Los Alamitos, CA, USA (1996)
3. Christinal, H.A., Díaz-Pernil, D., Real, P.: Segmentation in 2D and 3D image using tissue-like P system. In: Bayro-Corrochano, E., Eklundh, J.O. (eds.) Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications 14th Iberoamerican Conference on Pattern Recognition, CIARP 2009, Guadalajara, Jalisco, Mexico, November 15–18, 2009. Proceedings, Lecture Notes in Computer Science, vol. 5856, pp. 169–176. Springer, Berlin Heidelberg (2009)
4. Christinal, H.A., Díaz-Pernil, D., Real, P.: Using membrane computing for obtaining homology groups of binary 2D digital images. In: Wiederhold, P., Barneva, R.P. (eds.) Combinatorial Image Analysis 13th International Workshop, IWCIA 2009, Playa del Carmen, Mexico, November 24–27, 2009. Proceedings, Lecture Notes in Computer Science, vol. 5852, pp. 383–396. Springer, Berlin Heidelberg (2009)
5. Christinal, H.A., Díaz-Pernil, D., Real, P.: P systems and computational algebraic topology. Math. Comput. Model. 52(11–12), 1982–1996 (2010) (The BIC-TA 2009 Special Issue. International Conference on Bio-Inspired Computing, Theory and Applications)
6. Díaz-Pernil, D., Gutiérrez-Naranjo, M.A., Pérez-Jiménez, M.J., Riscos-Núñez, A.: A linear-time tissue P system based solution for the 3-coloring problem. Electron. Notes Theor. Comput. Sci. **171**(2), 81–93 (2007)
7. Díaz-Pernil, D., Gutiérrez-Naranjo, M.A., Pérez-Jiménez, M.J.: Riscos-Núñez A Solving subset sum in linear time by using tissue P systems with cell division. In: Mira, J., Álvarez, J.R. (eds.) IWINAC (1), Lecture Notes in Computer Science, vol. 4527, pp. 170–179. Springer, Berlin Heidelberg (2007)
8. Díaz-Pernil, D., Gutiérrez-Naranjo, M.A., Real, P., Sánchez-Canales, V.: Computing homology groups in binary 2D imagery by tissue-like P systems. Romanian J. Inf. Sci. Technol. **13**(2), 141–152 (2010)
9. Eilenberg, S., MacLane, S.: Relations between homology and homotopy groups of spaces. Ann. Math. 46(3), pp. 480–509 (1945). http://www.jstor.org/stable/1969165
10. Eilenberg, S., MacLane, S.: Relations between homology and homotopy groups of spaces. ii. Ann. Math. 51(3), pp. 514–533 (1950). http://www.jstor.org/stable/1969365
11. González-Díaz, R., Jiménez, M.J., Medrano, B., Molina-Abril, H., Real, P.: Integral operators for computing homology generators at any dimension. In: Ruiz-Shulcloper, J., Kropatsch, W.G. (eds.) CIARP, Lecture Notes in Computer Science, vol. 5197, pp. 356–363. Springer, Berlin Heidelberg (2008)

12. González-Díaz, R., Jiménez, M.J., Medrano, B., Real, P.: Chain homotopies for object topological representations. Discret. Appl. Math. **157**(3), 490–499 (2009)
13. González-Díaz, R., Real, P.: On the cohomology of 3D digital images. Discret. Appl. Math. **147**(2–3), 245–263 (2005). http://dx.doi.org/10.1016/j.dam.2004.09.014
14. Kenmochi, Y., Imiya, A., Ichikawa, A.: Discrete combinatorial geometry. Pattern Recogn. **30**(10), 1719–1728 (1997)
15. Kenmochi, Y., Imiya, A., Ichikawa, A.: Boundary extraction of discrete objects. Comput. Vis. Image Underst. **71**(3), 281–293 (1998)
16. Martín-Vide, C., Păun, G., Pazos, J., Rodríguez-Patón, A.: Tissue P systems. Theor. Comput. Sci. **296**(2), 295–326 (2003)
17. Molina-Abril, H., Real, P.: Advanced homology computation of digital volumes via cell complexes. In: da Vitoria Lobo, N., Kasparis, T., Roli, F., Kwok, J.T.Y., Georgiopoulos, M., Anagnostopoulos, G.C., Loog, M. (eds.) SSPR/SPR, Lecture Notes in Computer Science, vol. 5342, pp. 361–371. Springer, Berlin Heidelberg (2008)
18. Păun, A., Păun, G.: The power of communication: P systems with symport/antiport. New Gener. Comput. **20**(3), 295–306 (2002)
19. Păun, G.: Computing with membranes. Technical Report 208, Turku Centre for Computer Science, Turku, Finland (1998)
20. Păun, G.: Computing with membranes. J. Comput. Syst. Sci. **61**(1), 108–143 (2000). http://dx.doi.org/10.1006/jcss.1999.1693. See also [19]
21. Păun, G.: Membrane Computing. An Introduction. Springer, Berlin, Germany (2002)
22. Păun, G., Rozenberg, G., Salomaa, A. (eds.): The Oxford Handbook of Membrane Computing. Oxford University Press, Oxford, England (2010)
23. Real, P.: Homological perturbation theory and associativity. Homol. Homotopy Appl. **2**(5), 51–88 (2000)
24. Real, P., Molina-Abril, H.: Cell at-models for digital volumes. In: Torsello, A., Escolano, F., Brun, L. (eds.) GbRPR, Lecture Notes in Computer Science, vol. 5534, pp. 314–323. Springer, Berlin Heidelberg (2009)
25. Real, P., Molina-Abril, H., Kropatsch, W.G.: Homological tree-based strategies for image analysis. In: Jiang, X., Petkov, N. (eds.) CAIP, Lecture Notes in Computer Science, vol. 5702, pp. 326–333. Springer, Berlin Heidelberg (2009)
26. Sergeraert, F.: The computability problem in algebraic topology. Adv. Math. **104**, 1–29 (1994)