Pattern Recognition Letters 32 (2011) 2206–2212

Contents lists available at ScienceDirect

Pattern Recognition Letters

journal homepage: www.elsevier.com/locate/patrec

Region-based segmentation of 2D and 3D images with tissue-like P systems

Hepzibah A. Christinal [a,b], Daniel Díaz-Pernil [b,*], Pedro Real [b]

[a] Karunya University, Coimbatore 641 114, Tamilnadu, India
[b] Research Group on Computational Topology and Applied Mathematics, Universidad de Sevilla, Avda. Reina Mercedes s/n, 41012 Sevilla, Spain

ARTICLE INFO

Article history:
Available online 12 May 2011

Keywords:
P systems
Membrane Computing
Digital image
Region-based segmentation
Digital topology

ABSTRACT

Membrane Computing is a biologically inspired computational model. Its devices are called P systems and they perform computations by applying a finite set of rules in a synchronous, maximally parallel way. In this paper, we develop a variant of P-system, called *tissue-like P system* in order to design in this computational setting, a region-based segmentation algorithm of 2D pixel-based and 3D voxel-based digital images. Concretely, we use 4-adjacency neighborhood relation between pixels in 2D and 6-adjacency neighborhood relation between voxel in 3D for segmenting digital images in a constant number of steps. Finally, specific software is used to check the validity of these systems with some simple examples.

1. Introduction

Natural Computing studies new computational paradigms inspired on Nature. It abstracts the way in which Nature "computes", conceiving new computing models. There are several fields in Natural Computing that are now well established, such as Genetic Algorithms (Holland, 1992), Neural Networks (McCulloch and Pitts, 1988), DNA-based molecular computing (Adleman, 1994).

Membrane Computing is a theoretical model of computation inspired by the structure and functioning of cells as living organisms able to process and generate information. The computational devices in Membrane Computing are called *P systems* (Păun, 2000). Roughly speaking, a P system consists of a membrane structure, in whose compartments one places multisets of objects which evolve according to given rules. In the most common model, the rules are applied in a synchronous non-deterministic maximally parallel way, but some other semantics are being explored.

According to their architecture, these models can be split into two sets: cell-like P systems (Păun, 2000) and tissue-like P systems (Mira and Álvarez, 2007; Díaz-Pernil et al., 2008, 2009). In the first systems, membranes are hierarchically arranged in a tree-like structure. The inspiration for such architecture is the set of vesicles inside the cell. All of them perform their biological processes in parallel and life is the consequence of the harmonious conjunction of such processes. This paper is devoted to the second approach: tissue-like P systems.

Segmentation in computer vision (Stockman and Shapiro, 2001) refers to the process of partitioning a digital image into multiple segments (sets of pixels). The goal of segmentation is to simplify and/or change the representation of an image into something that is more meaningful and easier to analyze. Image segmentation is typically used to locate objects and boundaries (lines, curves, etc.) in images. More precisely, image segmentation is the process of assigning a label to every pixel in an image in such a way those pixels with the same label share certain visual characteristics. There exist different techniques to segment an image. Some of them are Clustering methods (Li et al., 2008), Histogram-based methods (Tobias and Seara, 2002), Watershed transformation methods (Tarabalka et al., 2010), Graph partitioning methods (Yuan et al., 2009). Some of the practical applications of image segmentation are medical Imaging (Campadelli et al., 2009), study of anatomical structures, location of objects in satellite images (roads, forests, etc.) (Gamanya et al., 2007), and face recognition (Zhao et al., 2003).

Previous work putting into relation Natural Computing with Digital Imagery are (Ceterchi et al., 2003a,b), among others. Here, we develop massively parallel algorithms for segmentation of digital images using a recent scheme in Natural Computing: tissue-like P systems.

The paper is structured as follows. In Section 2, we introduce the definition of basic tissue-like P systems and show an example to understand how these systems work. In Section 3, we design a family of systems for region-based segmentation in 2D image ($n \times m$). Afterwards, we check our model using a program called Tissue Simulator with very easy images. At the end of this section, we introduce a family of tissue-like P systems to obtain a region-based segmentation of 3D images. Finally, some conclusions and future work are given.

* Corresponding author.
E-mail addresses: hepzi@us.es (H.A. Christinal), sbdani@us.es (D. Díaz-Pernil), real@us.es (P. Real).

0167-8655/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.patrec.2011.05.004

2. Description of a model of membranes

We begin this section by briefly recalling 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 alphabet Σ is denoted by Σ^n and $\Sigma^* = \cup_{n\geqslant 0}\Sigma^n$. A *language* over Σ is a subset from Σ^*.

A *multiset* over a set A is a pair (A,f) where $f : A \to \mathbb{N}$ is a mapping. The set of all multisets on A will be denoted by $\mathcal{M}(A)$. 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). If $m = (A,f)$ is a finite multiset over A, then it will be denoted as $m = a_1^{f(a_1)}a_2^{f(a_2)}\ldots a_k^{f(a_k)}$, where $supp(m) = \{a_1,\ldots,a_k\}$, and for each element a_i, $f(a_i)$ is called the multiplicity of a_i. If $f(a_i) = 1$, we will write a_i instead of a_i^1. In what follows, we assume the reader is already familiar with the basic notions and the terminology underlying P systems.[1]

Martín-Vide et al. introduced in (Martín-Vide et al., 2003) a new variant of P systems where the cells are ordered in tissues. It has two biological inspirations: intercellular communication and cooperation between neurons. In this paper, we work with a new tissue-like model presented in (Mira and Álvarez, 2007; Díaz-Pernil et al., 2009) closer to the cell-like systems (classical P systems). The common mathematical model of these two mechanisms is a network of processors dealing with symbols and communicating these symbols along channels specified in advance.

Formally, a *tissue-like P system* of degree $q \geqslant 1$ is a tuple

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

where

1. Γ is a finite *alphabet*, whose symbols will be called *objects*,
2. $\Sigma(\subset\Gamma)$ is the input alphabet,
3. $\mathcal{E} \subseteq \Gamma$ (the objects in the environment),
4. $w_1,\ldots,w_q$ are strings over Γ representing the multisets of objects associated with the cells at the initial configuration,
5. $\mathcal{R}$ is a finite set of communication rules of the following form: $(i,u/v,j)$, for $i,j \in \{0,1,2,\ldots,q\}$, $i \neq j$, $u,\ v \in \Gamma^*$,
6. $i_\Pi \in \{1,2,\ldots,q\}$,
7. $o_\Pi \in \{0,1,2,\ldots,q\}$.

A tissue-like P system of degree $q \geqslant 1$ can be seen as a set of q cells (each one consisting of an elementary membrane) labelled 1, 2, ..., q. We will use 0 to refer to the label of the environment, i_Π denotes the input region and o_Π denotes the output region (which can be the region inside a cell or the environment).

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

The communication rule $(i,u/v,j)$ can be applied over two cells labelled 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. Note that, if either $i = 0$ or $j = 0$, then the objects are interchanged between a cell and the environment. Therefore, some objects not belonging to $\mathcal{E}$ can go to the environment. So, in a configuration (not initial) we could find two types of objects in the environment: First, those which belong to the environment that appear in an arbitrary large number of copies. So, system could take many copies of them as it needs in each computation step. Second, those which not belong to the environment. For them, the environment works like a cell, i.e., if three copies of an object, for example $a \notin \mathcal{E}$, arrive to the environment during a computation step. There was not copies of a in the environment before. Then, system has only three copies of element a to work, no more.

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 so, i.e., in each step we apply a maximal set of rules.

A *configuration* is an instantaneous description of the system Π, and it is represented as a tuple $(w_0,w_1,\ldots,w_q)$. Given a configuration, 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. Then, a computation halts when the system reaches a halting configuration. In the literature, the output of a computation is collected from its halting configuration by reading the objects contained in the output cell.

Let us now show a simple example:

Consider a tissue-like P system

$$\Pi' = (\Gamma, \Sigma, \mathcal{E}, w_1, w_2, w_3, \mathcal{R}, i_\Pi, o_\Pi),$$

where

1. $\Gamma = \{a,b,e,f\}$ is the working alphabet,
2. $\Sigma = \emptyset$ is the input alphabet. In this case we consider empty,
3. $\mathcal{E} = \{a,f\}$ is the environment alphabet,
4. $w_1 = ab^4e$, $w_2 = a^2bf$, $w_3 = e^4f$, are the multiset of cells 1, 2 and 3, respectively,
5. $\mathcal{R} = \{r_1 \equiv (1,b/f,3),\ r_2 \equiv (2,a/e,3),\ r_3 \equiv (1,a/b,2),\ r_4 \equiv (3,e/f^2,0),\ r_5 \equiv (1,f/\lambda,0)\}$ is the set of communication rules,
6. $i_\Pi = 1$ is the label of input cell,
7. $o_\Pi = 0$ is the label of output cell.

We can see the initial configuration of this system in first image of Fig. 1. There are five rules to use, r_1 is applied once and system exchanges an object b in cell 1 by an object f in cell 3. r_3 is applied once too and an object a in cell 1 is traded against an object b. r_5

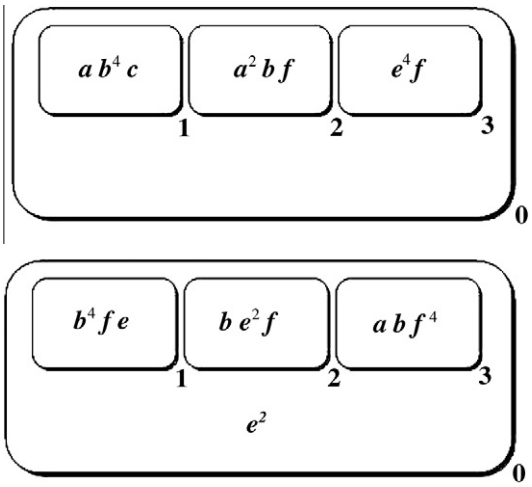


Fig. 1. Initial configuration of system Π' and the next configuration.

[1] We refer to (Păun, 2002) for basic information in this ares, to (Păun et al., 2010) for a comprehensive presentation and the web site (The P Systems Webpage, 2008) for the up-to-date information.

cannot be applied by the system in this computation step, because there is not an object f in cell 1. The question is the application of the rules r_2 and r_4. System has two possibilities: it can use rules r_2 and r_4 twice each one or, r_4 could works four times (r_2 no one). System chooses one or another possibility in a non deterministic manner. In this example we have chosen by the system the first one. So, two copies of e go to cell 1 and two other copies go to the environment. We must remember that system can apply rule r_2 twice because f is an element of the alphabet (of the environment) $\mathcal{E}$. Moreover, we do not show the objects of $\mathcal{E}$ in the environment in Fig. 1, but system can take them of the environment when it needs and many copies as it wants. We only show in the environment those objects which not belong to $\mathcal{E}$ (as we can see in the second image in Fig. 1 two copies of the object e). System does not stop in this computation step because it can apply rules r_1 four times and r_5 once. Next, system applies four times rule r_5 again and then, system halts.

3. Segmenting digital images in a constant time

A 2-*D digital image* $\mathcal{I}$ *with size* $n \times m$ $(n, m \in \mathbb{N})$ is a rectangular net of objects (i,j) called *pixels* (*voxels* in 3-D images), with $1 \leqslant i \leqslant n$ and $1 \leqslant j \leqslant m$. Let *the alphabet of colors of* $\mathcal{I}$ $(\mathcal{C} \subseteq \mathbb{N})$ be the ordered set of all colors in $\mathcal{I}$. We define the size of $\mathcal{C}$, $|\mathcal{C}|$, as the number of colors of this alphabet. Moreover, we will assume that each pixel of $\mathcal{I}$ is associated to a color of $\mathcal{C}$. So, we encode the pixel (i,j) with associated color $a \in \mathcal{C}$ as the object a_{ij}. Therefore, we codify image $\mathcal{I}$ as the set $\{a_{ij} : a \in \mathcal{C} \wedge 1 \leqslant i \leqslant n \wedge 1 \leqslant j \leqslant m\}$.

Given two pixels $p_1 = (i,j)$ and $p_2 = (k,l)$, we say they are *adjacent* when the distance $d(p_1,p_2) = \sqrt{(i-k)^2 + (j-l)^2}$ is 1. Then, we consider a 4-adjacency (Rosenfeld, 1979) neighborhood relation between pixels. In an analogous way, we can design a segmentation algorithm for the 8-adjacency relation, but with first adjacency we need to define more types of rules with respect to the second one, i.e., we consider a more complex and more interesting problem from a membrane computing point of view.

Following this way, we can divide an image in *regions*, where each region is a set of two to two adjacent pixels and everyone has the same associated color. We define a *boundary* of a region as the set of pixels of this region with the property to be adjacent to other pixel with a different associated color.

In this section, we segment pixel-based digital images using a region-based segmentation algorithm. This type of segmentation finds regions with different color from each other present in an image. We define two family of tissue-like P systems to realize this type of segmentation. $\boldsymbol{\Pi_1}$ segments 2D digital images and $\boldsymbol{\Pi_2}$ is the adaptation of the first system to segment 3D voxel-based digital images.

Region-based segmentation of n–D digital images (n–D–RS) problem: Given a *n–D* (with $n = 2 \vee 3$) digital image with pixels (or voxels) of (possibly) different colors, eliminate the boundaries between regions in that image.

3.1. Region-based segmentation of a 2D digital image

Given a digital image with $n \times m$ pixels $(n, m \in \mathbb{N})$ we define a tissue-like P system whose input is given by the set $\{a_{ij} : a \in \mathcal{C} \wedge 1 \leqslant i \leqslant n \wedge 1 \leqslant j \leqslant m\}$.

Next, we shall give some outlines of how to prove that the *2D-ES problem* can be solved in a constant amount of time with respect to the input data using a family of basic P systems.

First, the system marks the boundary pixels. Next, it marks the necessary pixels to connect all the boundary pixels of the same color. The system needs 8 computation steps for this and uses a counter, z_i, to send the marked objects to the cell 2. But as the system only uses 8 steps to do the segmentation we need initially $\left\lceil r_1^{1/2^7} \right\rceil$ copies of object z_1, with $r_1 = max(n,m)$, to generate enough copies of z_1 for the system output. It is given by the objects that appear in the cell 2 when it stops. So, the system is ready to send the objects codifying the complete image to the cell 2 in the last step of computation.

3.1.1. A family of tissue-like P systems

So, we define a family of tissue-like P systems to do the region-based segmentation of a 2D image. For each $n, m \in \mathbb{N}$ we consider the tissue-like P system of degree 2

$$\boldsymbol{\Pi_1}(n,m) = (\Gamma, \Sigma, \mathcal{E}, w_1, w_2, \mathcal{R}, i_\Pi, o_\Pi),$$

which is defined by the following conditions:

(a) The working alphabet is $\Gamma = \Sigma \cup \mathcal{E}$,
(b) the input alphabet is $\Sigma = \{a_{ij} : a \in \mathcal{C},\ 1 \leqslant i \leqslant n,\ 1 \leqslant j \leqslant m\}$,
(c) the environment alphabet is $\mathcal{E} = \{\bar{a}_{ij} : 1 \leqslant i \leqslant n,\ 1 \leqslant j \leqslant m\} \cup \{z_i : 1 \leqslant i \leqslant 9\}$,
(d) the multisets of cells 1 and 2 are $w_1 = z_1^{\lceil r_1^{(1/2^7)} \rceil}$, $w_2 = \emptyset$, respectively,
(e) R is the following set of communication rules:
  1. $(1, z_i / z_{i+1}^2, 0)$, for $i = 1, \ldots, 8$.
     These rules are used to update the counter z_i duplicating the number of copies in each step.
  2. $(1, a_{ij} b_{kl} / \bar{a}_{ij} b_{kl}, 0)$, for $a, b \in \mathcal{C},\ a < b,\ 1 \leqslant i, k \leqslant n$ and $1 \leqslant j, l \leqslant m$ (see Fig. 2, center image).
     These rules are used when the image has two adjacent

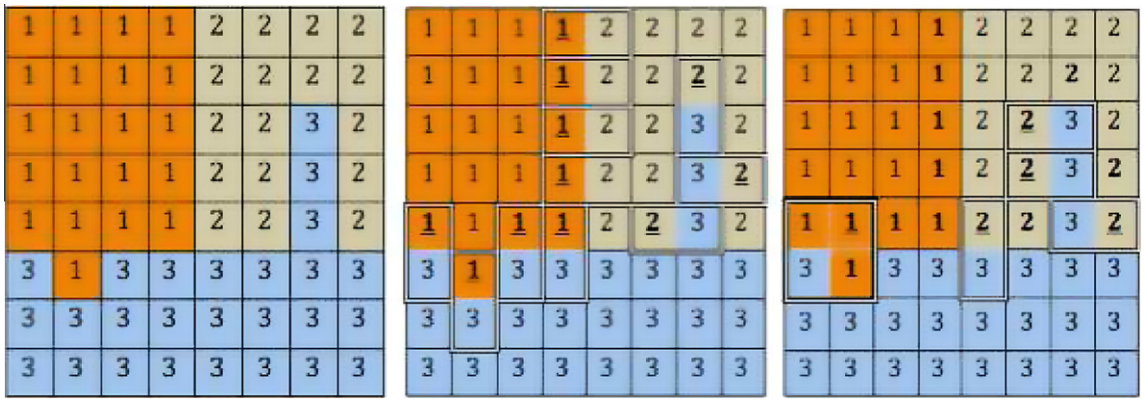
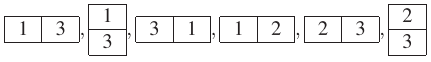


Fig. 2. $\boldsymbol{C_0}$ (left): Original Image. $\boldsymbol{C_1}$ (center): Some edge pixels have been marked. Only rules of type 1 have been used. $\boldsymbol{C_2}$ (right): System has applied more rules of type 1 and one rule of type 2.

pixels with different associated colors (border pixels). For example, in Fig. 2 we can find pair of adjacent pixels of the form:



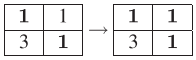
Then, the pixel with smallest associated color is marked and system brings an object representing this marked pixel (edge pixel) from the environment. For example, in the center of Fig. 2 we use bold fonts to highlight the edge pixels. So, if we take the pair of colored adjacent pixels of the form $\boxed{1}\boxed{3}$ the edge pixel would be $\boxed{1}$.

3. $(1, \bar{a}_{ij}a_{ij+1}\bar{a}_{i+1j+1}b_{i+1j}/\bar{a}_{ij}\bar{a}_{ij+1}\bar{a}_{i+1j+1}b_{i+1j}, 0)$, for $a, b \in \mathcal{C},\ a < b,\ 1 \leqslant i \leqslant n-1,\ 1 \leqslant j \leqslant m-1$,
$(1, \bar{a}_{ij}a_{i-1j}\bar{a}_{i-1j+1}b_{ij+1}/\bar{a}_{ij}\bar{a}_{i-1j}\bar{a}_{i-1j+1}b_{ij+1}, 0)$, for $a, b \in \mathcal{C},\ a < b,\ 2 \leqslant i \leqslant n,\ 1 \leqslant j \leqslant m-1$,

$(1, \bar{a}_{ij}a_{ij+1}\bar{a}_{i-1j+1}b_{i-1j}/\bar{a}_{ij}\bar{a}_{ij+1}\bar{a}_{i-1j+1}b_{i-1j}, 0)$, for $a, b \in \mathcal{C},\ a < b,\ 2 \leqslant i \leqslant n,\ 1 \leqslant j \leqslant m-1$,

$(1, \bar{a}_{ij}a_{i+1j}\bar{a}_{i+1j+1}b_{ij+1}/\bar{a}_{ij}\bar{a}_{i+1j}\bar{a}_{i+1j+1}b_{ij+1}, 0)$, for $a, b \in \mathcal{C},\ a < b,\ 1 \leqslant i \leqslant n-1,\ 1 \leqslant j \leqslant m-1$.

The rules mark with a bar the pixels which are adjacent to two edge pixels and they are adjacent to other pixel with a different color with respect to them. If we see the right image on Fig. 2, we can find four adjacent colored pixels, and apply this type of rules of the following form:



adding a new edge pixel. We use bold font to the edge pixels, i.e. objects of the form $\bar{a}$ (in this case we use **1** for $\bar{1}$).

4. $(1, z_9\bar{a}_{ij}/\lambda, 2)$, for $1 \leqslant i \leqslant n$, $1 \leqslant j \leqslant m$. These rules send the marked pixels to the environment (see Fig. 3):

(f) the input cell is $i_\Pi = 1$,

(g) the output cell is $o_\Pi = 0$.

3.1.2. An overview of the computation

When the input objects a_{ij} encoding the colored pixels from a 2D digital image and the counter z_i appear in the input cell, the system begins to work. Rules of type 1 identify the border pixels and bring the edge pixels from the environment in a parallel manner. At the same time, rules of type 3 duplicate the counter z_i during eight steps to obtain the needed copies of the object z_9. The rules of type 1 need four steps to mark all the border pixels similar to the edge-based segmentation. From the second step the rules of type 2 can be used in parallel with the first rules. So, in another four steps, we can bring from the environment the edge pixels adjacent to two border pixels. The system can apply the first two types of rules simultaneously in some configurations. The system always applies the same number of these two types of rules because this number is given by the edge pixels (we consider 4-adjacency). Finally, the fourth type of rules are applied in the following step, using the objects z_9 the edge pixels are sent to the environment leaving behind different color regions. We need only 9 steps to obtain a region-based segmentation for a $n \times m$ image.

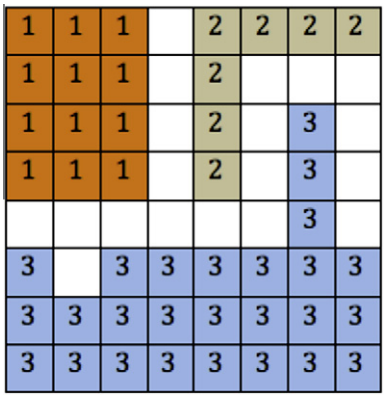


Fig. 3. Output image.

3.1.3. Complexity and resources needed

Taking into account that the size of the input data is $O(n \cdot m)$ and the number of colors of the image ($|\mathcal{C}|$) is h, the amount of resources needed to define the systems of our family and the complexity of our problem are determined in the following table:

2D-RS Problem	
Complexity	
Number of steps of a computation	9
Resources needed	
Size of the alphabet	$n \cdot m \cdot h$
Initial number of cells	2
Initial number of objects	$\lceil r_1^{(1/2^7)} \rceil$
Number of rules	$O(n \cdot m \cdot h^2)$
Upper bound for the length of the rules	8

3.1.4. Checking this family of systems with Tissue Simulator software

We have used a program called *Tissue Simulator* (see Fig. 4) introduced by Borrego et al. (2007). We have simulated our family of systems to segment 2D digital images with this software. Finally, we have introduced as instances of our system the examples appearing in Fig. 5 and, in a constant number of steps, we have obtained an encoding of a region-segmentation (showed in Fig. 6) for each one of the examples introduced before.

3.2. Segmenting 3D digital images

The next step consists of extending our models to work with 3D images. Now, the input data are voxels ($(i,j,k) \in \mathbb{N}^3$) encoded by the objects a_{ijk}, with $a \in \mathcal{C}$. We consider the 26-adjacency relation for determining the neighbors of one voxel.

3.2.1. A family of tissue-like P systems

Given a digital image with $n \times m \times l$ voxels ($n, m, l \in \mathbb{N}$) we define a tissue-like P system whose input is given by the voxels of the image encoded by the objects a_{ijk}, where $1 \leqslant i \leqslant n$, $1 \leqslant j \leqslant m$ and $1 \leqslant k \leqslant l$.

Next, we shall give an outline of how to prove that the *3D-ES problem* can be solved in a constant amount of time with respect to the input data using a family of basic P systems.

A system of this family works in the following way: First, the system marks the boundary pixels. Next, the system uses a counter (z_i whose number of initial copies in the system is $\lceil r_2^{1/2^7} \rceil$, where $r_2 = max(n,m,l)$) to send the marked objects to the environment.

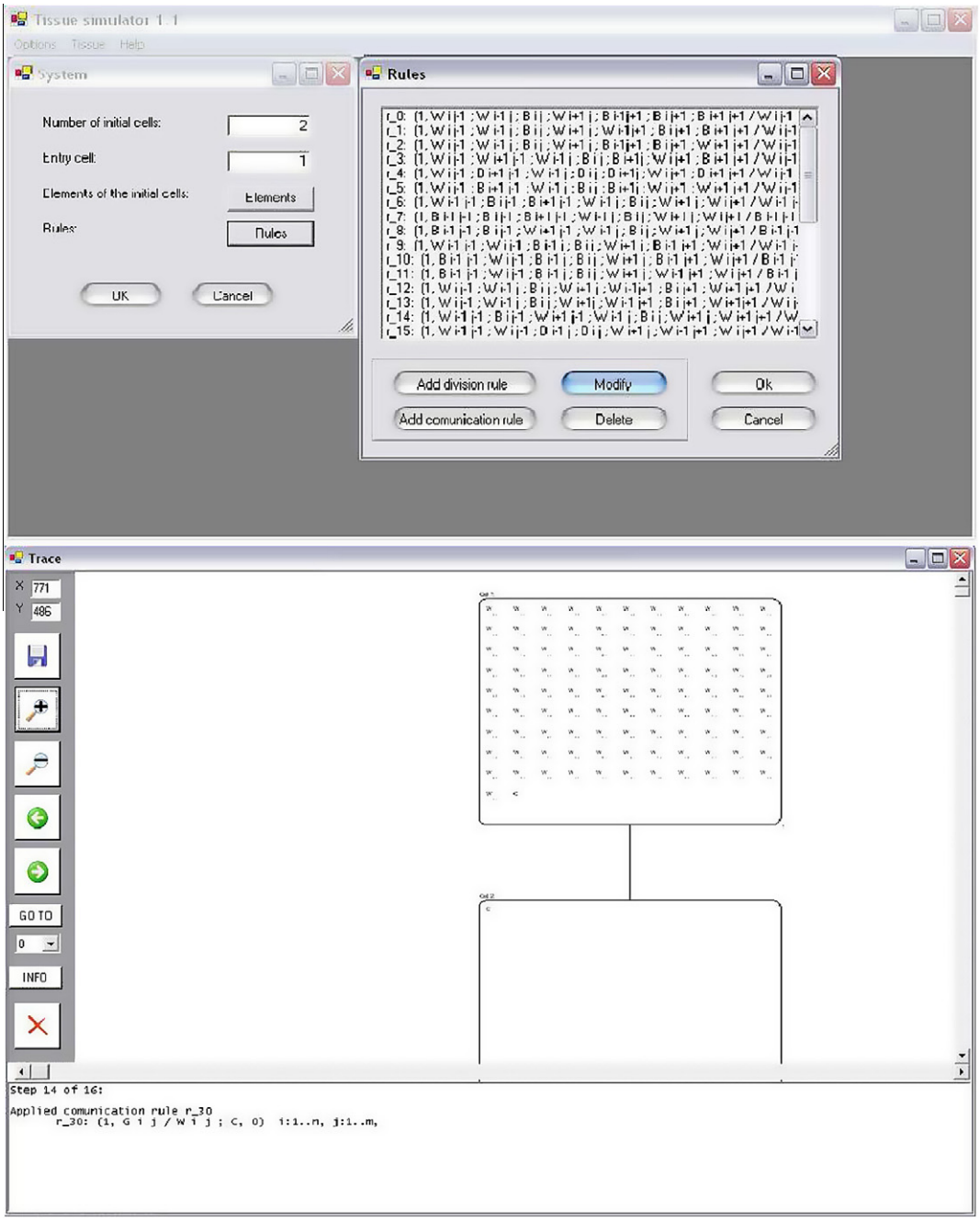


Fig. 4. Two images of Tissue Simulator.

The output is given by the objects appearing in the output cell when the system stops.

$\boldsymbol{\Pi_2}(n,m,l) = (\Gamma, \Sigma, \mathcal{E}, w_1, w_2\mathcal{R}, i_\Pi, o_\Pi)$

defined as follows:

(a) The working alphabet is $\Gamma = \Sigma \cup \mathcal{E}$,
(b) the input alphabet is $\Sigma = \{a_{ijk} : a \in \mathcal{C},\ 1 \leqslant i \leqslant n,\ 1 \leqslant j \leqslant m,\ 1 \leqslant k \leqslant l\}$,
(c) the environment alphabet is $\mathcal{E} = \{\bar{a}_{ijk} : 1 \leqslant i \leqslant n,\ 1 \leqslant j \leqslant m,\ 1 \leqslant k \leqslant l\} \cup \{y, z_i : 1 \leqslant i \leqslant 27\}$,
(d) the multisets of cells 1 and 2 are $w_1 = z_1^{\lceil r_2^{(1/2(25))} \rceil}$, $w_2 = \emptyset$, respectively,
(e) R is the following set of communication rules:

1. $(1, z_i/z_{i+1}^2, 0)$, for $i = 1, \ldots, 26$. These rules are used to update the counter z_i duplicating the number of copies in each step.
2. $(1, a_{i_1j_1k_1} b_{i_2j_2k_2}/\bar{a}_{i_1j_1k_1} b_{i_2j_2k_2}, 0)$, for $1 \leqslant i_1, i_2 \leqslant n$, $1 \leqslant j_1, j_2 \leqslant m$, $1 \leqslant k_1,\ k_1 \leqslant l$, with (i_1,j_1,k_1) and (i_2,j_2,k_2) adjacent voxels, and finally $a, b \in \mathcal{C}, a < b$.
These rules are used when the image has two adjacent voxels with different associated colors (border voxels). Then, the voxel with smallest associated color is marked and the system brings from the environment an object representing this marked voxel (edge voxel).
3. $(1, z_{27}\bar{a}_{ijk}/\lambda, 2)$, for $1 \leqslant i \leqslant n$, $1 \leqslant j \leqslant m$ and $1 \leqslant k \leqslant l$.
These rules send the marked pixels to the environment.

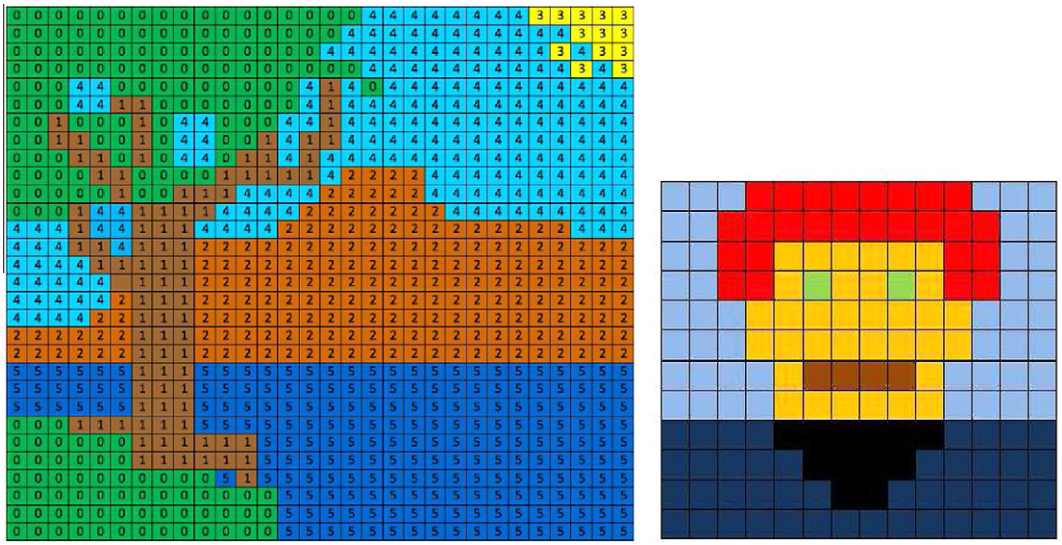


Fig. 5. Input.

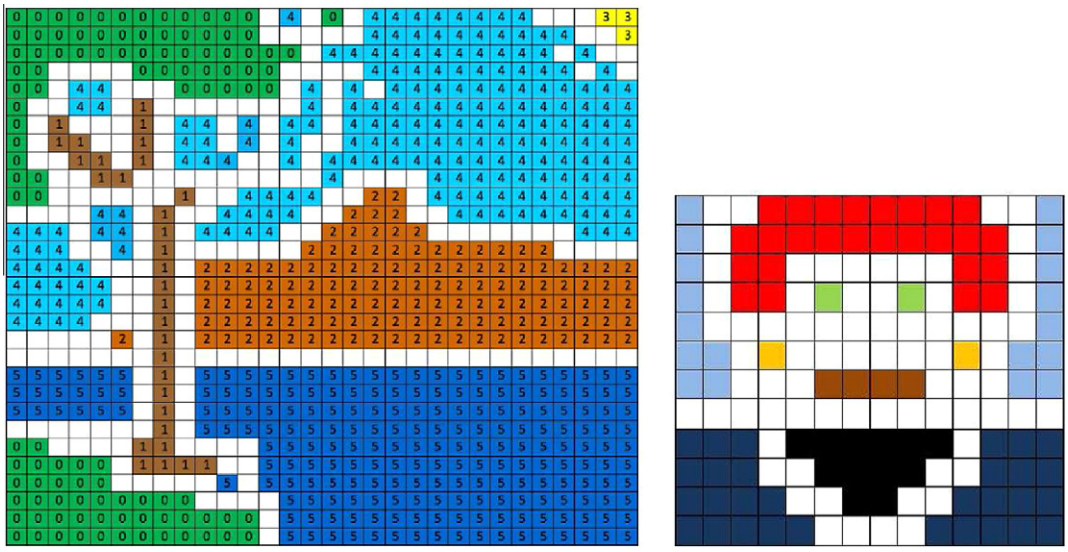


Fig. 6. Output.

(f) $i_{\Pi} = 1$,
(g) $o_{\Pi} = 0$.

3.2.2. An overview of the computation

This computation would be very similar if we consider an 8-adjacency in 2D. Rules of type 1 identify the border voxels and bring the edge voxels from the environment. These rules need as much as 26 steps for this. Finally, the second type of rules are applied in the following step sending the edge voxels to the environment. So we need again a constant number of steps in this 3D region-segmentation massively parallel algorithm.

3.2.3. Complexity and resources needed

Taking into account the size of the input data is $O(n \cdot m \cdot l)$ and the number of colors of the image $|C|$ is h, the amount of necessary resources for defining the systems of our family and the complexity of our problem can be observed in the following table:

3D-RS problem	
Complexity	
Number of steps of a computation	9
Resources needed	
Size of the alphabet	$n \cdot m \cdot l \cdot h$
Initial number of cells	2
Initial number of objects	$\lceil r_2^{(1/2^7)} \rceil$
Number of rules	$O(n \cdot m \cdot l \cdot h^2)$
Upper bound for the length of the rules	4

4. Conclusions and future work

In this paper, tissue-like P-systems (Díaz-Pernil et al., 2008) are used for theoretically solving a region-based segmentation prob-

lem for digital images. This work opens new promising research lines: (a) developing efficient sequential software mimicking these Natural Computing techniques; (b) developing efficient parallel software working with a cluster or a graphic card (GPUs); (c) and also designing topology-based processes (topological noise removal, topological simplification, snakes and active contours, topological filling, image compression,...) in Digital Imagery using Membrane Computing. Segmentation can be seen as a topology-based process in Digital Imagery, attending to the topological characteristics (connected components and holes, mainly) of each object defined by one unique label. To have at hand representation models for digital images trying to capture full topological and geometrical information of them (González-Díaz and Real, 2005; González-Díaz et al., 2009a,b) could be of help for getting this last goal.

Acknowledgement

The first author acknowledges the support of the project "Computational Topology and Applied Mathematics" PAICYT research project FQM-296. The second author acknowledge the support of the project TIN2009-13192 of the Ministerio de Educación y Ciencia of Spain, cofinanced by FEDER funds, and the support of the Project of Excellence P08-TIC-04200 of the Junta de Andalucía. The third author 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

Adleman, L.M., 1994. Molecular computation of solutions to combinatorial problems. Science 266 (5187), 1021–1024.

Borrego, R., Díaz-Pernil, D., de J. Pérez-Jiménez, M., 2007. Tissue simulator: A graphical tool for tissue p systems. In: Proceedings of the International Workshop Automata for Cellular and Molecular Computing, satellite of the 16th International Symposium on Fundamentals of Computational Theory.

Campadelli, P., Casiraghi, E., Esposito, A., 2009. Liver segmentation from computed tomography scans: A survey and a new algorithm. Artif. Intell. Med. 45 (2–3), 185–196.

Ceterchi, R., Gramatovici, R., Jonoska, N., Subramanian, K.G., 2003a. Tissue-like p systems with active membranes for picture generation. Fundam. Inform. 56 (4), 311–328.

Ceterchi, R., Mutyam, M., Păun, G., Subramanian, K.G., 2003b. Array-rewriting p systems. Nat. Comput. an Internat. J. 2 (3), 229–249.

Díaz-Pernil, D., Gutiérrez-Naranjo, M.A., Pérez-Jiménez, M.J., Riscos-Núñez, A., 2008. A uniform family of tissue p systems with cell division solving 3-col in a linear time. Theor. Comput. Sci. 404 (1–2), 76–87.

Díaz-Pernil, D., Pérez-Jiménez, M. de J., Romero, A., 2009. Efficient simulation of tissue-like p systems by transition cell-like p systems. Nat. Comput. 8, 797–806.

Gamanya, R., Maeyer, P.D., Dapper, M.D., 2007. An automated satellite image classification design using object-oriented segmentation algorithms: A move towards standardization. Expert Systems Appl. 32 (2), 616–624.

González-Díaz, R., Real, P., 2005. On the cohomology of 3d digital images. Discrete Appl. Math. 147 (2–3), 245–263.

González-Díaz, R., Jiménez, M.J., Medrano, B., Real, P., 2009a. Chain homotopies for object topological representations. Discrete Appl. Math. 157 (3), 490–499.

Gonzalez-Diaz, R., Jimenez, M.J., Medrano, B., Real, P., 2009b. A tool for integer homology computation: λ-at-model. Image Vision Comput. 27 (7), 837–845.

Holland, J.H., 1992. Adaptation in Natural and Artificial Systems. MIT Press, Cambridge, MA, USA.

Li, X., Zhang, T., Qu, Z., 2008. Image segmentation using fuzzy clustering with spatial constraints based on markov random field via bayesian theory. IEICE Trans. Fundam. Electron. Comm. Comput. Sci. E91-A (3), 723–729.

Martín-Vide, C., Paun, G., Pazos, J., Rodríguez-Patón, A., 2003. Tissue p systems. Theor. Comput. Sci. 296 (2), 295–326.

McCulloch, W.S., Pitts, W., 1988. A logical calculus of the ideas immanent in nervous activity. Neurocomput. Found. Res., 15–27.

Mira, J., Álvarez, J.R. (Eds.), 2007. Bio-inspired Modeling of Cognitive Tasks, Proceedings of the Second International Work-Conference on the Interplay Between Natural and Artificial Computation, IWINAC 2007, La Manga del Mar Menor, Spain, June 18–21, 2007, Part 1. Lecture Notes in Computer Science, vol. 4527. Springer.

Păun, G., 2000. Computing with membranes. J. Comput. Systems Sci. 61 (1), 108–143.

Păun, G., 2002. Membrane Computing. An Introduction. Springer-Verlag, Berlin Germany.

Păun, G., Rozenberg, G., Salomaa, A., 2010. The Oxford Handbook of Membrane Computing. Oxford University Press.

Rosenfeld, A., 1979. Digital topology. American Mathematical Monthly 86, 621–630.

Stockman, G., Shapiro, L.G., 2001. Computer Vision. Prentice Hall PTR, Upper Saddle River, NJ, USA.

Tarabalka, Y., Chanussot, J., Benediktsson, J.A., 2010. Segmentation and classification of hyperspectral images using watershed transformation. Pattern Recognition 43 (7), 2367–2379.

The P Systems Webpage, 2008. Electronic references. Available from: http://ppage.psystems.eu/.

Tobias, O.J., Seara, R., 2002. Image segmentation by histogram thresholding using fuzzy sets. IEEE Trans. Image Process. 11 (12), 1457–1465.

Yuan, X., Situ, N., Zouridakis, G., 2009. A narrow band graph partitioning method for skin lesion segmentation. Pattern Recognition 42 (6), 1017–1028.

Zhao, W., Chellappa, R., Phillips, P.J., Rosenfeld, A., 2003. Face recognition: A literature survey. ACM Comput. Surv. 35 (4), 399–458.