Programa de Doctorado "Matemáticas"

PhD Dissertation

Parallelization strategies for extraction of (co)homological information of digital objects

Author
Raúl Reina Molina

Supervisor
Prof. Dr. *Pedro Real Jurado*

Co-supervisor
Dr. *Daniel Díaz Pernil*

July 5, 2017

Acknowledgments

The work and dedication that requires a work like the one presented in this dissertation cannot be faced alone. From these lines I want to thank all the people who have helped me in some way or another. Not only their technical or theoretical help is important, although I have received it from my supervisors Pedro Real and Daniel Díaz, from other fellow doctoral students such as Aldo González Lorenzo or from other doctors like Paweł Pilarczyk without whose help it would have been impossible to adapt the algorithm suitable for the calculation of Smith's Normal Form; but it is equally inestimable the emotional help, understanding and effort made by all those who have understood my absences and dedication to this work. In particular, I want to thank the understanding received by my wife without whose help I would not have been able to face this thesis. Last but not least, I can not miss the opportunity to apologize to all those who have received less attention from the accustomed and especially my son born in the course of the development of this dissertation.

To all of them, thank you very much.

Contents

List of Figures

Abstract

Digital objects are finite subsets of n-xels within a n-dimensional digital image. The study of the connectivity of these objects, interpreted from a discrete, subdivided or continuous way, has been a priority issue from the very beginning of Digital Imagery. The topological tools that have been exhaustively used in this setting are the notions of connected component, simple point and Euler characteristic. Others topological invariants with a recent increasing popularity are (co)homology groups of digital objects treated as cell complexes. In this thesis, we propose parallelization strategies based on extraction methods of (co)homological information, like Discrete Morse Theory, Effective Homology or AT-model.

The first approach is related with Natural Computing, which is a fruitful research area that provides interesting approaches to computational problems inspiring in the way that Nature "computes". Concretely, we use Membrane Computing, which summarizes with computational rules, the manner living cells work. This area has provided interesting results in theoretical and applied works. The application of these ideas to the process of calculating (co)homology groups of digital objects allows us to develop better algorithms as it brings a natural parallelization of the algorithms implemented in Membrane Computing.

Nowadays, there is no current device capable of executing Membrane Computing algorithms, hence the previous processes need to be adapted to be executed by ordinary computing devices. Therefore, in this thesis we present a set of algorithms that provides a compact representation, optimal in some way, of a digital object along with a bidirectional transformation that allows us to compute not only the (co)homology groups but compute some algebraic invariants or operation involving (co)homology classes which can be used as intrinsic information of the digital object.

The work presented in this thesis focuses in two main contributions. The first of all is related with Natural Computing. We present a Membrane Computing framework used to make easier the development of Membrane Computing algorithms in Computational Algebraic Topology. This framework is strongly connected with Discrete Morse Theory.

The second main contribution is the application of the framework mentioned above for developing a parallel algorithm used to compute a reduction from a cubical cell complex to a CW complex with a

minimal amount of cells. This reduction makes the extraction of (co)homological information simpler. This algorithm focus on n-dimensional cubical complexes and uses $\mathbb{Z}$ as the ground ring, which makes it useful for computing torsion.

Resumen

Un objeto digital es un conjunto de n-xels en de una imagen digital n-dimensional. El estudio de la conectividad de estos objetos, interpretados de manera discreta, subdividida o continua, ha sido una custión prioritaria desde los inicios del tratamiento de Imagen Digital. Las herramientas topológicas que se han usado exhaustivamente en esta tarea son las nociones de componente conexa, punto simple o la característica de Euler. Otros invariantes topológicos con una creciente popularidad son los grupos de (co)homología de objetos digitales tratados como complejos celulares. En esta tesis proponemos estrategias de paralelización basadas en métodos de extracción de información (co)homológica como la Teoría Discreta de Morse, la homología efectiva o los modelos AM.

La primera aproximación que realizamos está relacionada con la Computación Natural, que es un área de investigación fructífera que proporciona formas interesantes de abordar problemas computacionales inspirándose en la forma en la que la Naturaleza "computa". Concretamente, usamos Computación con Membranas, que resume mediante reglas de computación, la manera en la que trabajan las células de los seres vivos. Este área ha proporcionado resultados interesantes en trabajos teóricos y aplicados. La aplicación de estas ideas al proceso de cálculo de los grupos de (co)homología de objetos digitales nos permite desarrollar mejores algoritmos, ya que los algoritmos de Computación con Membranas son naturalmente paralelos.

En la actualidad no existe ningún dispositivo capaz de ejecutar (no emular) algoritmos de Computación con Membranas, por lo que los algoritmos citados anteriormente necesitan ser adaptados para su ejecución en dispositivos de computación ordinarios. De esta manera, en esta tesis presentamos una serie de algoritmos que proporcionan una representación compacta, de alguna manera óptima, de un objeto digital junto con una transformación bidireccional que nos permite no solo calcular los grupos de (co)homología sino computar algunos invariantes algebraicos mediante operaciones que involucren clases de (co)homología que puedan usarse como información intrínseca del objeto digital.

El trabajo presentado en esta tesis se centra fundamentalmente en dos contribuciones. La primera de ellas está relacionada con la Computación Natural. Presentamos un marco de trabajo en Computación con Membranas usado para hacer más fácil el desarrollo de algoritmos en Computación con Membranas sobre Topología Algebraica Computacional. Este marco de trabajo está fuertemente relacionado con la Teoría Discreta de Morse.

La segunda contribución principal es la aplicación del marco de trabajo anterior para desarrollar un algoritmo paralelo que compute una reducción de un complejo celular cúbico en un CW complejo con una cantidad minimal de celdas, esta reducción hace que la extracción de información (co)homológica sea más simple. Este algoritmo se centra en complejos cúbicos n-dimensionales y usa $\mathbb{Z}$ como anillo base, lo que lo convierte en una herramienta muy útil para el cálculo de la torsión.

CHAPTER 1

Introduction

The understanding of a digital object is not only determined by the knowledge of its geometry, given by its volume, its curvature or some other geometrical characteristic, but a deeper knowledge that remains unaltered by continuous deformations is preferable. In this case the topology presents a fundamental framework. In this case the key aspect is not how close are two points (geometric information given by a metric) but it is the relation of proximity in itself what matters. The connectivity information appears in this case.

Some connectivity information is given by connected components, tunnels, voids and in general, by any part that encloses a void like the one enclosed by a sphere of some dimension. This topo-geometric idea appears unclear, since it refers to something that does not exist in the given space and is perceived via confrontation to an ambient space. However, a digital object is conceived as an abstract entity in an aseptical and independent manner of the ambiance in which, actually, it lives on. Therefore it is necessary to define holes, which do not exist in space, through information provided by the space. This information is detailed by the homology groups.

On the other hand, cohomology generators in some sense represent cutting paths by which a digital object can be cut so that a hole disappears. There must be somehow a cohomology generator for each homology generator in order to make them disappear[1].

(Co)homology groups are commonly computed by algebraic means. Its

[1]This idea could be thought as an intuitive version of the Universal Coefficient Theorem on cohomology.

computation involves a large number of calculations that, in digital objects coming from real applications, require working with a huge amount of data.

This work shows a solution to the problem of extracting (co)homological information in a way that allows an improvement in its performance by exploiting the massive parallelism present in the GPUs of today's computers. At the same time it focuses on applications of these ideas to n-dimensional complexes. The usual (co)homological study is centered on objects in $\mathbb{R}^3$ or in the use of $\mathbb{Z}_2$ as coefficients ring. In both cases it is achieved that the (co)homology groups are free, i.e. the information of the torsion subgroups is lost.

The solution presented in this dissertation is based on the key concept of reduction. This algebraic object represents a transformation of the chain complex associated to a digital object in another chain complex with the minimum information to represent (co)homology. These reductions also make visible the "geometric" process that allows to obtain, from a homotopy point of view, the reduced complex from the original one. This process can be reversed to allow a reconstruction of the original object from the reduced information.

This dissertation is organized as follows. Chapter 2 reviews fundamental topological concepts. In chapter 3 a bioinspired strategy is presented to parallelize the extraction of homological information from digital objects. Next is shown how to extract advanced (co)homological information from digital objects from a mimimal reduction of the chain complex associated with the digital object. Finally, several practical cases of application are presented and the implementation of the proposed algorithms in a real prototype is discussed.

CHAPTER 2

Overview of algebraic topology

In this chapter, some required concepts of Topology and Algebraic Topology will be reviewed.

2.1 General Topology

Topology is the branch of mathematics devoted to the study of the properties of spaces which are preserved under continuous deformations. In this section we recall some concepts in this area.

Definition 2.1 (Topological Space)**.**
A topological space is a pair $(X, \mathcal{T})$ where X is a set and $\mathcal{T} \subset 2^X$ is a family of subsets of X, called open sets, which satisfies the following conditions:

(a) $\{\emptyset, X\} \subset \mathcal{T}$: both empty set and the whole space X are open sets.

(b) For any family $\{A_i : i \in \Lambda\}$ of open sets, its union $\bigcup_{i\in\Lambda} A_i$ is open.

(c) For any finite family $\{A_i : 1 \leq i \leq n\}$ of open sets, its intersection $\bigcap_{i=1}^{n} A_i$ is open.

As an example, the Euclidean space $\mathbb{R}^n$ is a topological space with the Euclidean topology $\mathcal{E} = \{A \subset \mathbb{R}^n | A = \bigcup_{i\in\Lambda} \mathbf{B}(x_i, r_i)\}$ where $\mathbf{B}(x, r) = \left\{ y \in \mathbb{R}^n \,\middle|\, \sqrt{\sum_{i=1}^{n}(y_i - x_i^2)} < r \right\}$ is the open ball centred at x with radius r. Basically, open Euclidean sets are those who can be expressed as the union of a family of open balls.

Any subset of a topological space $(X, \mathcal{T})$ is, itself, a topological space using the so called relative topology.

Definition 2.2 (Relative topology)**.**
Let $(X, \mathcal{T})$ be a topological space and $U \subset X$. The relative topology $\mathcal{T}_U$ is defined as:

$$\mathcal{T}_U = \{A \cap U | A \in \mathcal{T}\}$$

Definition 2.3 (Continuous map)**.**
A map $f : X \to Y$ between two topological spaces $(X, \mathcal{T}_X)$ and $(Y, \mathcal{T}_Y)$ is continuous if for all $A \in \mathcal{T}_Y$, $f^{-1}(A) \in \mathcal{T}_X$.

The usual way of working in Mathematics involves the definition of objects, relation (maps) between them and a relation of equality among spaces. In Topology, the equality relation is given by homeomorphisms.

Definition 2.4 (Homeomorphism)**.**
An homemorphism $f : X \to Y$ between two topological spaces is a bijective continuous map with continuous inverse. In this case, X and Y are called homeomorphic spaces.

Topological properties are those properties that remain unchanged by homeomorphisms.

Definition 2.5 (Homotopic maps)**.**
Two continuous map $f : X \to Y$ and $g : X \to Y$ are homotopic, $f \simeq g$, if there is a continuous map $F : X \times [0, 1] \to Y$ such that $F(x, 0) = f(x)$ and $F(x, 1) = g(x)$. The map F is called a homotopy. Two topological spaces X and Y are homotopy equivalent if there exist two continuous map $f : X \to Y$ and $g : Y \to X$ such that $f \circ g \simeq 1_Y$ and $g \circ f \simeq 1_X$.

Finding whether two spaces are homeomorphic is, in general, a very hard problem. On the other hand, the relation of "equality" induced by homotopy equivalence relaxes the exigences of homeomorphism at the same time it remain unchanged many interested properties of topological spaces. For example, $\mathbb{R} \times \{0\}$ and $\mathbb{R} \times \{0\} \cup \{0\} \times \mathbb{R}$ are two subspaces of the Euclidean plane which are homotopy equivalent but not homeomorphic. The homotopy equivalence is intuitively viewed as one copy of $\mathbb{R}$ retracting to a point. However, the "cross"-like set is not homeomorphic to the line as the former has all of it points as cut point of degree two, i.e. removing any of its points leave two connected components, while the later has one point of degree four, and this property is conserved under homeomorphisms.

2.2 Algebraic Topology

To find out whether two topological spaces are homeomorphic is sometimes an impossible task. It is preferable to find some properties that remain unchanged under suitable maps and such that they allow to differentiate both spaces. In other words, if both spaces have different values in some property then it can be said that those spaces cannot be the same. Algebraic Topology studies how an algebraic object (called algebraic invariant) can be assigned to any topological space such that homotopy equivalent spaces have isomorphic objects. Therefore, this algebraic property of spaces allows to differentiate them.

One of very first algebraic invariant is the fundamental group of a topological space.

The fundamental group

The fundamental group of a space X will be defined so that its elements are loops in X starting and ending at a fixed basepoint $x_0 \in X$, but two such loops are regarded as being the same if one loop can be continuously deformed to the other within the space X . (All loops that occur during deformations must also start and end at x_0.)

Definition 2.6 (Path)**.**
A path in a space X is a continuous map $\gamma : I = [0, 1] \subset \mathbb{R} \to X$.

Two paths will be considered the same if they are homotopic as continuous maps between topological spaces.

As show in Proposition 1.2 in [16], the relation of homotopy on paths with fixed endpoints in any space is an equivalence relation.

The equivalence class of a path γ under the equivalence relation of homotopy will be denoted $[\gamma]$ and called the homotopy class of γ.

Given two paths $\gamma_1, \gamma_2 : I \to X$ such that $\gamma_1(1) = \gamma_2(0)$, there is a composition or product path $\gamma_1 \cdot \gamma_2$ that traverses first γ_1 and then γ_2 , defined by the formula

$$(\gamma_1 \cdot \gamma_2)(s) = \begin{cases} \gamma_1(2s) & , 0 \leq s \leq 1/2 \\ \gamma_2(2s-1) & , 1/2 \leq s \leq 1 \end{cases} \tag{2.1}$$

This product operation respects homotopy classes (see [16], for example).

In particular, we restrict attention to paths $\gamma : I \to X$ with the same starting and ending point $\gamma(0) = \gamma(1) = x_0 \in X$. Such paths are called loops, and the common starting and ending point x_0 is referred to as the basepoint. The set of all homotopy classes $[\gamma]$ of loops $\gamma : I \to X$ at the basepoint x_0 is denoted $\pi_1(X, x_0)$. As is proved in Proposition 1.3 in [16], $\pi_1(X, x_0)$ is a group with respect to the product $[\gamma_1] \cdot [\gamma_2] = [\gamma_1 \cdot \gamma_2]$.

This group is called the fundamental group of X at the basepoint x_0. $\pi_1(X, x_0)$ is the first in a sequence of groups $\pi_n(X, x_0)$, called homotopy groups, which are defined in an entirely analogous fashion using the n dimensional cube I^n in place of I.

Calculation of fundamental group and, in general, higher homotopy groups is a very difficult task even in simple spaces. For example, calculation of higher homotopy groups of spheres involves a non-negligent number of sophisticated algebraic tools. This difficulty made mathematicians opt for other algebraic invariants with simpler computation.

Homology groups

Homology groups are algebraic invariants with a purely geometric origin. They are defined to capture the geometric idea of hole. A hole is easy to imagine but has a non negligent definition. It can be though as a void in the space, but this idea requires the definition of an ambient space. In order to associate homology groups to a topological space, some decomposition has to be done. This decomposition is known as combinatorialization of the space.

We first introduce the required algebraic machinery.

Definition 2.7 (Chain complex)**.**
Let $\mathcal{R}$ be a commutative ring with unit. A chain complex $\mathcal{C}$ is a sequence

$$\cdots \xrightarrow{\mathrm{d}_{n+2}} \mathsf{C}_{n+1} \xrightarrow{\mathrm{d}_{n+1}} \mathsf{C}_n \xrightarrow{\mathrm{d}_n} \mathsf{C}_{n-1} \xrightarrow{\mathrm{d}_{n-1}} \cdots$$

where C_n are free $\mathcal{R}$-modules, and d_n are $\mathcal{R}$-module homomorphisms, which are called differentials, satisfying the identity $d_n \circ d_{n+1} = 0$, for all n.

The subgroup of cycles is defined as $\mathsf{Z}_n(\mathcal{C}; \mathcal{R}) := \ker(\mathrm{d}_n)$ and $\mathsf{B}_n(\mathcal{C}; \mathcal{R}) := \mathrm{Im}(\mathrm{d}_{n+1})$ is the subgroup of boundaries. The property $\mathrm{d}_n \circ \mathrm{d}_{n+1} = 0$ implies that $\mathsf{Z}_n(\mathcal{C}; \mathcal{R}) \supseteq \mathsf{B}_n(\mathcal{C}; \mathcal{R})$, for all n, making the following definition possible.

Definition 2.8 (nth homology group)**.**
For a chain complex $\mathcal{C}$, the nth homology groups is defined by

$$\mathsf{H}_n(\mathcal{C}; \mathcal{R}) := \mathsf{Z}_n(\mathcal{C}; \mathcal{R}) / \mathsf{B}_n(\mathcal{C}; \mathcal{R})$$

.

Note that any object in an homology group represent a class of cycles which are not boundaries so it represents a "hole" in some dimension.

Definition 2.9 (Chain map)**.**
Let $\mathcal{C} = (\mathsf{C}_\bullet, \mathrm{d}_\bullet)$ and $\mathcal{C}' = (\mathsf{C}'_\bullet, \mathrm{d}'_\bullet)$ be two chain complexes. A collection of $\mathcal{R}$-module homomorphisms $f = \{f_\bullet : \mathsf{C}_\bullet \to \mathsf{C}'_\bullet$, is called a chain map, written $f : \mathcal{C} \to \mathcal{C}'$, if for all n:

$$f_{n-1} \circ \mathrm{d}_n = \mathrm{d}'_n \circ f_n \tag{2.2}$$

Proposition 2.10 (3.30 in [21])**.**
Let $\mathcal{C}$ and $\mathcal{C}'$ be arbitrary chain complexes. Then any chain map $f : \mathcal{C} \to \mathcal{C}'$ induces homomorphisms on the homology groups $f_* : \mathsf{H}_n(\mathcal{C}; \mathcal{R}) \to \mathsf{H}_n(\mathcal{C}'; \mathcal{R})$, for all n.

In the proof of previous proposition, $f_*([c]) := [f(c)]$

An important part of the theory of chain complexes is the observation that the chain maps can be deformed algebraically.

Definition 2.11.
Let $\mathcal{C} = (\mathsf{C}_\bullet, \mathrm{d}_\bullet)$ and $\mathcal{C}' = (\mathsf{C}_\bullet, \mathrm{d}'_\bullet)$ be two chain complexes, and let $f = \{f_\bullet\}$ and $g = \{g_\bullet\}$ be two chain maps $f, g : \mathcal{C} \to \mathcal{C}'$. A sequence of homomorphisms $h = \{h_\bullet\}$, where $h_n : \mathsf{C}_n \to \mathsf{C}_{n+1}$ is a chain homotopy between f and g if for all n we have

$$h_{n-1} \circ \mathrm{d}_n + \mathrm{d}_{n+1} \circ h_n = f_n - g_n \tag{2.3}$$

Clearly, the existence of a chain homotopy between two maps is an equivalence relation: just replace h with its negative to show the symmetry, and add two chain homotopy maps to show the transitivity.

Without a doubt, the most important use of chain homotopies is to show that two chain maps induce the same homology homomorphisms.

Proposition 2.12 (3.32 in [21])**.**
Let $\mathcal{C}^1 = (\mathsf{C}^1_*, d^1_*)$ and $\mathcal{C}^2 = (\mathsf{C}^2_*, d^2_*)$ be two chain complexes, and let $f, g : \mathcal{C}^1 \to \mathcal{C}^2$ be two chain maps such that there exists a chain homotopy h between f and g. Then the induced maps $f_*, g_* : \mathsf{H}_\bullet(\mathcal{C}^1) \to \mathsf{H}_\bullet(\mathcal{C}^2)$ are equal.

Once the algebraic construction of homology groups has been introduced, it is mandatory to assign a chain complex to every topological space. In order to do that, the space is decomposed into smaller pices called cells in general. Depending on the building blocks used to make that partition we can speak of different types of combinatorialization of topological spaces. The following subsection is devoted to review the most important ways of decomposing a space into smaller pieces.

Combinatorialization of topological spaces

Each of the decompositions shown below will play a crucial role in the calculation of homology groups of a topological space.

For each cellular structure associated to the topological space there is one homology theory. We review here cubical homology, simplicial homology, homology of CW complexes (CW homology) and, finally, singular homology. The later does not become from a combinatorialization of the topological space and, hence, does not require any additional structure. However, is much more difficult to be calculated.

Simplicial complexes

Definition 2.13 (Simplex)**.**
Given $p+1$ points $\{v_0,\ldots,v_p\}\subset\mathbb{R}^n$ such that the vectors $\{\overrightarrow{v_0v_1},\ldots,\overrightarrow{v_0v_p}\}$ are linearly independent, the p-simplex $\langle v_0,\ldots,v_p\rangle$ is the convex hull of the set $\{v_0,\ldots,v_p\}$, i.e.:

$$\langle v_0,\ldots,v_p\rangle=\left\{\sum_{i=0}^{p}\lambda_i v_i\,\middle|\,\sum_{i=1}^{p}\lambda_i=1\wedge\lambda_i\geq 0\right\}$$

The faces of the simplex $\langle v_0,\ldots,v_p\rangle$ are the simplices $\langle v_{i_0},\ldots,v_{i_q}\rangle$ where $\{i_0,\ldots,i_q\}\subseteq\{0,\ldots,p\}$ and $0\leq q\leq p$.

Definition 2.14 (Simplicial complex)**.**
A simplicial complex K in $\mathbb{R}^n$ is a set of simplices such that:

1. *The faces of all the simplices in K are also in K.*
2. *The intersection of any two simplices in K is empty or a simplex in K.*

The set of p-simplices in K is denoted by K_p.

A simplicial complex is a collection of sets in $\mathbb{R}^n$, therefore the set $|K| = \bigcup_{\sigma \in K} \sigma \subset \mathbb{R}^n$ can be endowed with the relative topology and is, of course, a topological space. Hence the simplicial complex K can be viewed as simplicial structure of $|K|$. A space X is said to be triangulable if there is a simplicial complex K such that there exists a homemorphism $f : |K| \to X$. Hence, a simplicial structure in X is given by the set $\{f(\sigma) \,|\sigma \in K\}$.

For any triangulable space X, the corresponding simplicial structure K enables the calculation of the simplicial homology groups of X as follows.

Definition 2.15 (Simplicial chain complex)**.**
Let K be a simplicial complex and $\mathcal{R}$ a commutative ring with unit. The $\mathcal{R}$-module of p-chains, $\mathsf{C}_p(K;\mathcal{R})$ is the free $\mathcal{R}$-module spanned by K_p. The differential map d_p is defined by linear extension of the differential of a simplex, given by the following formula:

$$d_p(\langle v_0, \ldots, v_p \rangle) = \sum_{i=0}^{p} (-1)^i \langle v_0, \ldots, \hat{v_i}, \ldots, v_p \rangle \tag{2.4}$$

where the hat over the vertex v_i means the elimination of the vertex v_i.

The chain complex $\mathcal{C}(K) = (\mathsf{C}_\bullet(K;\mathcal{R}), \mathrm{d})$ is the chain complex associated to the simplicial complex K.

Is straightforward to prove that the chain complex defined below is, in fact, a chain complex, i.e, the composition of any two consecutive differentials is 0.

The p-th homology group $\mathsf{H}_p(K)$ can be, then, defined as the p-th homology group of the chain complex $\mathcal{C}(K)$.

Recall that we have introduced a way of calculating the homology of a simplicial structure K on a topological space X. For considering the homology groups of X as topological invariant, it must be assured that they do not depend on the triangulation of the space.

In the following paragraphs singular homology is presented. This homology theory provides the same results as simplicial homology when boths can be applied. Moreover, singular homology does not depend on any combinatorial structure in the space to be calculated. On the contrary it is not suitable to be used for real calculations.

Singular complexes

The standard n-simplex, for a non negative integer n, is the set

$$\Delta^n = \left\{ (\lambda_0, \ldots, \lambda_n) \in \mathbb{R}^{n+1} \left| \sum_{i=0}^{n+1} \lambda_i = 1 \wedge \lambda_i \geq 0 \right. \right\}$$

It is clear that the standard n-simplex is the simplex spanned by the $n+1$ points in $\mathbb{R}^{n+1}$ $(1, 0, \ldots, 0), \ldots, (0, 0, \ldots, 1)$.

There are some interesting maps from the standard $(n-1)$-simplex to the standard n-simplex, called the i-th degeneracy operator and defined by the formula above:

$$\epsilon_n^i : \Delta^{n-1} \to \Delta^n, \epsilon_n^i(x_0, \ldots, x_{n-1}) = (x_0, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_{n-1}) \quad (2.5)$$

Definition 2.16 (Singular simplex)**.**
Let X be a topological space. A singular p-simplex is a continuous map $\sigma : \Delta^p \to X$. The faces of σ are given by the singular p-simplices $\sigma \circ \epsilon_p^i : \Delta^{p-1} \to X$ for $0 \leq i \leq p$.

Once defined the singular simplices, we can define the singular chain complex similarly as the simplicial chain complex.

Definition 2.17 (Singular chain complex)**.**
Let X be a topological space and $\mathcal{R}$ a commutative ring with unit. The $\mathcal{R}$-module of singular p-chains is the free $\mathcal{R}$-module spanned by the singular p-simplices in X. It is denoted as $\mathsf{C}_p(X; \mathcal{R})$. Again, the differential map d_p is defined as the linear extension of the differential of a singular p-simplex, given in the formula above:

$$d_p(\sigma) = \sum_{i=0}^{p} (-1)^i \sigma \circ \epsilon_p^i \quad (2.6)$$

It is, again, straightforward to check that $d_p \circ d_{p+1} = 0$ and the singular chain complex $\mathcal{C}(X) = (\mathsf{C}_*(X; \mathcal{R}), d_*)$ is, in fact, a chain complex. The singular homology groups of X are then defined as the homology groups of the singular chain complex $\mathcal{C}(X)$.

Note that the singular chain complex does not depend on any choice and any two homotopy equivalents spaces have isomorphic homology groups (see [16, Corollary 2.11]). Moreover, if K is a simplicial structure of X, then both

simplicial and singular homology groups are isomorphic (see [16, Theorem 2.27]).

Recall that, even though simplicial homology does not provide a practical tool for calculating homology groups of spaces, when a space has some combinatorial structure, e.g. it is a simplicial complex, both calculation agrees as the resulting homology groups are isomorphic. In this sense, singular homology is the reference for all the other homology theories on topological spaces.

Cubical complexes

We mainly follow the process introduced in [19] with some minor changes. Specifically, Kaczyński *et al.* uses closed cubes as main combinatorial objects while we use open cubes.

A *cubical cell* σ is a finite product of intervals:

$$\sigma = I_1 \times \cdots \times I_k \subset \mathbb{R}^k$$

where I_j is an interval $(m_j, m_j + 1)$ with integer extremes and length one or the singleton $\{m_j\}$, denoted as (m_j), for each $j \in \{1, \ldots, k\}$. The interval I_j is referred to as the jth component of σ and denoted by $I_j(\sigma)$. The set of all cubical cells in $\mathbb{R}^k$ is denoted by $\mathcal{K}^k$. The set of all cubical cells is

$$\mathcal{K} = \bigcup_{k=1}^{\infty} \mathcal{K}^k \tag{2.7}$$

We usually require the cubical cells to be bounded. Hence, we define

$$\mathcal{K}^k_{*,n} = \{\sigma \in \mathcal{K}^k : 0 \leq \inf I_p(\sigma), \sup I_p(\sigma) < n, 1 \leq p \leq k\} \tag{2.8}$$

where k and n are nonnegative integer numbers.

Given a cubical cell σ in $\mathbb{R}^k$, its *embedding number* k is denoted by $\mathrm{emb}(\sigma)$. The dimension of σ is defined as the number of unitary intervals in its expression as product of intervals and is denoted by $\dim(\sigma)$. The set of all elementary cubes with dimension p is denoted by $\mathcal{K}_p$. The set of all elementary cubes in $\mathbb{R}^k$ with dimension p is denoted by $\mathcal{K}^k_p$. Whenever the dimension of a cubical cell require to be made explicit, it is denoted as a superscript between parenthesis. Therefore, $\dim\left(\sigma^{(p)}\right) = p$. We also explicitly indicate the dimension of a cell $\sigma^{(p)}$ writing that σ is a p-cell.

The closure[1] of a cubical cell can be decomposed into the union of lower-dimensional cubical cells. If δ and σ are two cubical cells in $\mathbb{R}^k$ of any dimension and $\delta \subset \overline{\sigma}$, then δ is a *face* of σ and is denoted as $\delta \leq \sigma$. If δ is a face of σ and $\delta \neq \sigma$, then δ is a *proper face* of σ, denoted as $\delta < \sigma$. If δ is a face of σ and $\dim(\delta) = \dim(\sigma) - 1$ then δ is a *primary face* of σ, denoted by $\delta \in \partial(\sigma)$. Therefore, the set of primary faces of a cell σ will be denoted as $\partial(\sigma)$

As an example, let σ be the open square $(0,1) \times (3,4)$, a 2-dimensional cubical cell in $\mathbb{R}^2$. Its closure is the square $\overline{\sigma} = [0,1] \times [3,4]$ and is decomposed as follows

$$\begin{aligned}[0,1] \times [3,4] =& (0,1) \times (3,4) \cup \\ & (0,1) \times (3) \cup (0,1) \times (4) \cup \\ & (0) \times (3,4) \cup (1) \times (3,4) \cup \\ & (0) \times (3) \cup (0) \times (4) \cup \\ & (1) \times (3) \cup (1) \times (4)\end{aligned}$$

Namely, the interior of the square, its four edges and its four vertices. The set of primary faces is given by

$$\partial\left((0,1) \times (3,4)\right) = \{(0,1) \times (3), (0,1) \times (4), (0) \times (3,4), (1) \times (3,4)\}$$

A *cubical complex* is a collection K of cubical cells with the same embedding number and such that, for every cubical cell $\sigma \in K$, all of its primary faces are in the complex. We denote the set of p-cells in K as K_p. In figure 2.1 a cubical complex in $\mathbb{R}^2$ is showed.

As usual, using cubical cells as algebraic objects allows us to define the corresponding chain complex whose homology will be called cubical homology of the given cubical complex. Again, as expected, cubical homology agrees with singular homology (see [19] for more details).

However, the definition of the differential of a cubical cell is not as simple as the simplicial or singular cases. We will proceed in a similar manner as in the singular case, following the work in [31, 38].

Let $\sigma = I_1 \times \cdots \times I_n$ be a p-cubical cell. Denote $I_q = (a_q^0, a_q^1)$ with a_q^i two integers such that $0 \leq a_q^1 - a_q^0 \leq 1$. Let $k_1, \ldots, k_p$ denote the indexes of non-degenerated intervals. For any set $J \subset \{1, \ldots, p\}$ define $k(J) = \{k_i \,|i \in J\}$

[1]Given a set $A \subset \mathbb{R}^k$, its closure is $\overline{A} = \{p \in \mathbb{R}^k : \forall \epsilon > 0 \, \exists q \in A | d(p,q) > \epsilon\}$. Informally, the closure of a set is the set itself together with the points that are "infinitely" near to the set.

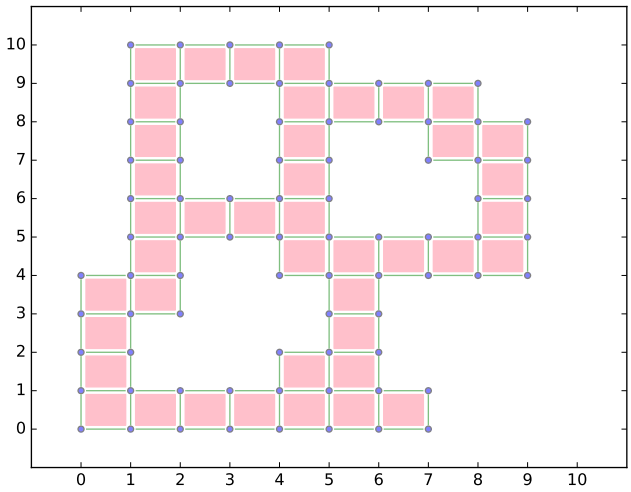


Figure 2.1: Example of cubical complex K_0 in $\mathbb{R}^2$.

and for $i \in \{0, 1\}$ the cubical cell $\lambda^i_J(\sigma) = I'_1 \times \cdots, I'_q$, where $I'_j = \{a^i_j\}$ if $j \in k(J)$ or $I'_j = I_j$ otherwise. Roughly speaking, for any set of non degenerated intervals of σ given by indices in J, λ^i_J represents the face given by degenerating those given intervals to the "lower" value (given for $i = 0$) or the "upper" one (given by $i = 1$). We denote $\lambda^i_{\{j\}}$ as λ^i_j.

With the above notation in mind, the differential of a p-cubical cell σ is given by:

$$d_p(\sigma) = \sum_{j=1}^{p} (-1)^j (\lambda^1_j(\sigma) - \lambda^0_j(\sigma)) \tag{2.9}$$

It is a simple exercise to check that, in fact, $d_p \circ d_{p+1} = 0$ and, hence, the chain complex given by cubical chains and the differential defined by linear extension of the differential of a cubical cell can be used to define the homology groups of a cubical complex.

If we do proceed in a similar manner as we done in the simplicial case, and we call cubical spaces to those homeomorphic to the support of any cubical complex (recall that the support of a cubical complex K is $|K| = \bigcup_{\sigma \in K} \sigma \subset \mathbb{R}^{\text{emb}(K)}$). Again as expected, cubical homology of K and singular homology of $|K|$ are isomorphic (see [19]).

CW Complexes

As shown before, computation of homology groups of a space X requires some partition of the space into smaller pieces (simplices or cubical cells). CW complexes are a generalization of this procedure that moves away from the simple geometry of cubes or simplices to more abstract shapes.

An m-cell is any topological space homeomorphic to an m-dimensional closed unit ball $\mathbf{B}^m$. An open m-cell is a topological space homeomorphic to the interior of a ball $\mathbf{B}^m$. To build a CW complex from its constituent cells, we glue some cells with other cells. The following definition concretes this gluing procedure.

Definition 2.18 (Attaching spaces along maps)**.**
Let X and Y be topological spaces, let $A \subseteq X$ be a closed subspace, and let $f : A \to Y$ be a continuous map. $Y \cup_f X$ denotes the quotient space $X \coprod Y/\sim$, where the equivalence relation $\sim$ is generated by $a \sim f(a)$, for all $a \in A$. We say that the space $Y \cup_f X$ is obtained from Y by attaching X along f.

Observe that the space $Y \cup_f X$ is equipped with the quotient topology, which means that $S \subseteq Y \cup_f X$ is open if and only if $q^{-1}(S)$ is open in $X \coprod Y$, where $q : X \coprod Y \to Y \cup_f X$ is the quotient map.

Definition 2.19 (Constructive definition of CW complex)**.**
A CW complex X is obtained by the following inductive construction of the skeletons:

(1) The 0-skeleton $X^{(0)}$ is a discrete set.

(2) Construct the n-skeleton $X^{(n)}$ by the simultaneous attachment of the n-cells to $X^{(n-1)}$ along their boundaries. In particular, $X^{(n)}$ gets the quotient topology as described above.

(3) Equip the space $X = \bigcup_{n=0}^{\infty} X^{(n)}$ with the weak topology: $A \subset X$ is open if and only if $A \cap X^{(n)}$ is open for any n.

The definition above asserts that a set $A \subseteq X$ is open if and only if $f_\alpha^{-1}(A)$ is open for any cell α, where $f_\alpha : \mathbf{B}_\alpha^n \to X$ is the attachment map (also called the characteristic map).

Definition 2.20 (Subcomplex)**.**
Let X be a CW complex; $A \subseteq X$ is called a subcomplex if A is a union of open cells such that if $e \subseteq A$, then the closure $\overline{e} \subseteq A$.

An important property of CW complexes is expressed by the following proposition.

Proposition 2.21.
A compact subspace of a CW complex is contained in a finite subcomplex.

Corollary 2.22.
For each open cell e in a CW complex X, its closure $\overline{e}$ is contained in finitely many open cells.

Definition 2.23 (Definition of CW complex by J.H.C. Whitehead)**.**
Let X be a Hausdorff topological space, and assume that it is represented as a disjoint union of open cells e_α. Then, the pair $(X, \{e_\alpha\}_\alpha)$ is called a CW complex if the following two conditions are satisfied:

(1) For any α, there exists a continuous map $f_\alpha : \mathbf{B}^m \to X$ (m is the dimension of e_α) such that

- *the restriction of f_α to B^m is a homeomorphism onto e_α;*
- *$f_\alpha(\partial \mathbf{B}^m)$ is a subset of a union of finitely many cells of dimension lessthan m.*

(2) The subset $A \subseteq X$ is closed in X if and only if $A \cap \overline{e}_\alpha$ is closed for any α.

CW complexes enjoy several properties:

- CW complexes are normal (meaning that disjoint closed subspaces can be encapsulated in disjoint open subspaces);
- a CW complex is connected if and only if it is path connected;
- all CW complexes are locally contractible.

A nicer class of complexes is obtained by imposing an extra condition.

Definition 2.24 (Regular CW complex)**.**
A CW complex X is called regular if for each cell α, the restriction of the characteristic map $f_\alpha : \partial \mathbf{B}_\alpha \to f_\alpha(\partial \mathbf{B}_\alpha)$ is a homeomorphism.

In order to define the differential for the chain complex associated to a CW complex, we need to recall the concept of degree of a continuous map between spheres.

Definition 2.25 (Degree of continuous map between spheres)**.**
For an arbitrary continuous map $f : S^n \to S^n$, we have an induced map $f_* : \mathsf{H}_n(S^n;\mathbb{Z}) \to \mathsf{H}_n(S^n;\mathbb{Z})$. As $\mathsf{H}_n(S^n;\mathbb{Z}) = \mathbb{Z}$ and the only homomorphisms in $\mathbb{Z}$ are $g_n(x) = nx$ where $n = g(1)$, the value $f_*(1)$ is called the degree of f, and is denoted by $\deg(f)$.

The differential of a n-cell σ in a CW complex structure K is given by

$$\mathrm{d}_n(\sigma) = \sum_{\tau \in K_{n-1}} [\tau : \sigma]\tau \tag{2.10}$$

where the numbers $[\tau : \sigma]$ are the incidence numbers. These are defined by

$$[\tau : \sigma] := \deg(p_\tau \circ f_{\partial\sigma})$$

where $f_{\partial\sigma} : \partial\mathbf{B}^n \to K_{(n-1)}$ is the attachment map of the cell σ, and p_τ is the composition $p_\tau : K_{(n-1)} \to K_{(n-1)}/K_{(n-2)} \to S^{n-1}$, where the first map is the quotient map shrinking the $(n-2)$-skeleton to a point, and the second map is the projection onto the sphere corresponding to τ.

As can be check in related literature (see e.g. [16]), the chain complex of cells with the differential above is in fact a chain complex whose homology groups (called cellular homology groups) are isomorphic to the singular homology groups.

Discrete Morse Theory

Discrete Morse Theory (DMT for short) is a discretization of Morse Theory. The latter is used to build a CW complex structure on manifolds and to obtain substantial information about their (co)homology.

A real-valued smooth map defined over a compact k-manifold is a *Morse function* if all its critical points have non-singular Hessian matrix and no two critical points have the same function value ([15]). Morse functions allow to endow the manifold with a cellular structure.

Discrete Morse Theory introduced by Forman ([11]) adapts Morse Theory to CW complexes instead of smooth manifolds.

Definition 2.26 (Discrete Morse function)**.**
Let K be a cell complex and $f : K \to \mathbb{R}$ a function that assigns scalar values to every cell of K. f is a discrete Morse function *if, for every cell $\sigma^{(p)} \in K$ the following conditions hold:*

- $\#\{\tau^{(p+1)} \in K | \tau > \sigma \wedge f(\tau) \leq f(\sigma)\} \leq 1$. *I.e., there is at most one facet*[2] *of* σ *where* f *takes in it a lower value than it does on* σ.
- $\#\{\mu^{(p-1)} \in K | \mu < \sigma \wedge f(\tau) \geq f(\sigma)\} \leq 1$. *I.e., there is at most one face of* σ *where* f *takes in it a greater value than it does on* σ.

where $\#A$ *denotes de number of elements of the (finite) set* A.

A discrete Morse function f can be thought as a discrete function that is increasing with respect to cell dimension except for, at most, two cells for each cell. Concretely, for each cell σ there is at most one face μ ($\mu \in \partial\sigma$) with $f(\mu) \geq f(\sigma)$ and there is at most one facet τ ($\sigma \in \partial\tau$) with $f(\tau) \leq f(\sigma)$.

A cell $\sigma^{(p)}$ is *critical* if one of the following conditions hold:

- $\#\{\tau^{(p+1)} \in K | \tau > \sigma \wedge f(\tau) \leq f(\sigma)\} = 0$
- $\#\{\mu^{(p-1)} \in K | \mu < \sigma \wedge f(\tau) \geq f(\sigma)\} = 0$

We define a *discrete vector* as a pair of incident cells $\{\sigma^{(p)} < \tau^{(p+1)}\}$. Vectors are represented as arrows from the cell of lower dimension to the higher dimension cell[3]. A *discrete vector field* V on K is a collection of vectors in K such that each cell in the vector belongs to, at most, one pair of V.

Definition 2.27 (Discrete vector field)**.**
A discrete vector field is a map $V : K \to K \cup \{0\}$ *such that:*

1. *for each* $\sigma \in K$, *if* $V(\sigma) \neq 0$ *then* $\dim V(\sigma) = \dim \sigma + 1$
2. *for each* $\sigma \in K_p$, *either* $V(\sigma) = 0$ *or* $\sigma \in \partial V(\sigma)$
3. *if* $\sigma \in \mathrm{Im}(V)$ *then* $V(\sigma) = 0$
4. *for each* $\sigma \in K_p$
$$\#\{\mu^{(p-1)} \in K | V(\mu) = \sigma\} \leq 1$$

In figure 2.2, a vector field (red arrows) in the cubical complex in figure 2.1 is showed.

[2] A cell with σ in its boundary.

[3] In simplicial or cubical complexes, a discrete vector can be represented as an arrow from the barycenter of the lower dimension cell to barycenter of the higher dimension cell

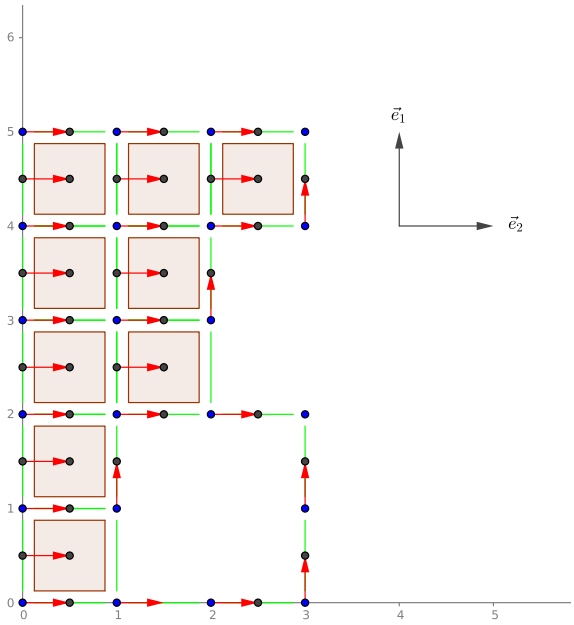


Figure 2.2: Discrete vector field V_0 defined in K_0 (see figure 2.1).

Definition 2.28 (V-path)**.**
Given a discrete vector field V on K, a V-path *of dimension p, γ, is a sequence of cubical p-cells $\sigma_0, \sigma_1, \sigma_2, \ldots, \sigma_r$ such that*

1. *If $V(\sigma_i) = 0$, then $\sigma_{i+1} = \sigma_i$.*

2. *If $V(\sigma_i) \neq 0$, then $\sigma_{i+1} < V(\sigma_i)$ with $\sigma_{i+1} \neq \sigma_i$.*

The set of V-paths is denoted as $\Gamma(V)$.

For example, $\gamma = (0) \times (0), (0) \times (1), (0) \times (2), (0) \times (3)$ is a V-path of dimension 0, where V is the vector field in figure 2.2.

A V-path $\gamma = \sigma_0, \sigma_1, \ldots, \sigma_r$ is called *closed* is $\sigma_r = \sigma_0$ and is called *non-stationary* if $\sigma_1 \neq \sigma_0$. A vector field V is called *acyclic* if there is no non-stationary closed V-paths. As an example, the vector field in figure 2.2 is, indeed, an acyclic vector field.

Given a discrete Morse function, a special discrete vector field called *discrete gradient vector field* is defined so that $f(V(\sigma)) \leq f(\sigma)$. As shown in [11], a cell complex can be transformed into another homotopically

equivalent following a series of simplicial collapses[4], where each of them collapses both cells in each vector in the corresponding discrete gradient vector field.

Recall that vectors give some kind of dynamic to the cells. Namely, one can (imaginary) move from one cell σ to another one μ if $\mu \in \partial V(\sigma)$. In the previous example, one can move from cell $(0) \times (0)$ to $(0) \times (1)$ as $V((0) \times (0)) = (0) \times (0,1)$ and $(0) \times (1)$ is in the boundary of the later.

We recall here one of the main results of Discrete Morse Theory.

Theorem 2.29 ([11] 9.3)**.**
A discrete vector field V is the gradient vector field of some discrete Morse function if and only if there are no non-stationary closed V-paths.

A vector field V is extended to a graded group homomorphism $V : C_p(K) \to C_{p+1}(K)$ such that

$$V(\sigma^{(p)}) = \begin{cases} \langle \sigma, \partial V(\sigma) \rangle \tau^{(p+1)} & \text{if } \{\sigma < \tau\} \text{ is a vector in } V \\ 0 & \text{otherwise} \end{cases} \tag{2.11}$$

The *reduced (discrete time) flow map*, denoted as φ is defined by

$$\varphi = \mathrm{id} - \partial \circ V \tag{2.12}$$

The reduced flow map associates a $(p+1)$-chain $\varphi(\sigma)$ to any p-cell σ, such that $\partial V(\sigma) = \sigma - \varphi(\sigma)$. Hence, $\langle \sigma, \varphi(\sigma) \rangle = 0$. The map φ establishes a way of *following* the flow determined by the vector field. Concretely, if there is a V-path of length r from a p-cell σ to another p-cell σ', then $\langle \varphi^r(\sigma), \sigma' \rangle \neq 0$. This fact can be proved by induction in the length of the V-path.

The reduced flow map encodes the dynamic idea presented before. If a cell μ is present in the chain $\varphi(\sigma)$, then σ, μ is a valid V-path.

Definition 2.30 (Morse chains)**.**
Let K be a cell complex and f be a Morse discrete function on K. Let $\mathsf{C}_p(K)$ denote the p-chains on K and $\mathsf{M}_p \subseteq \mathsf{C}_p(K)$ the span of the critical p-cells[5]. The graded group $\mathcal{M} = \{\mathsf{M}_p\}_{p \in \mathbb{Z}}$ is called the space of Morse chains.

[4] In [11], Forman works with simplicial complexes, however the mathematical scaffolding provided by DMT can be settled with no change to finite CW complexes.

[5] Linear combinations of the critical cells.

In [11], Forman makes use of the set of V-paths to build the boundary map of the Morse complex associated to a given complex and an acyclic vector field. It is shown in the following result.

Theorem 2.31 ([11] 7.1)**.**
There are boundary maps $\mathrm{d}_M : \mathsf{M}_\bullet \to \mathsf{M}_{\bullet-1}$ *so that*

$$\mathrm{d}_M \circ \mathrm{d}_M = 0$$

and such that the differential complex

$$0 \xrightarrow{(\mathrm{d}_M)_{k+1}} \mathsf{M}_k \xrightarrow{(\mathrm{d}_M)_k} \mathsf{M}_{k-1} \xrightarrow{(\mathrm{d}_M)_{k-1}} \cdots \xrightarrow{(\mathrm{d}_M)_2} \mathsf{M}_1 \xrightarrow{(\mathrm{d}_M)_1} \mathsf{M}_0 \xrightarrow{(\mathrm{d}_M)_0} 0$$

calculates de homology of K*. That is, if we define*

$$\mathsf{H}_p(\mathcal{M}) = \frac{\ker (\mathrm{d}_M)_p}{\mathrm{Im}\, (\mathrm{d}_M)_{p+1}}$$

then, for each p

$$\mathsf{H}_p(\mathcal{M}) \equiv \mathsf{H}_p(K).$$

AM-models

As shown in previous sections, given a topological space X and some cellularization K (simplicial, cubical or CW complex), we can compute the homology groups as the corresponding quotient group of cycles module boundaries. This computation is usually done by application of the Smith Normal Form. This algorithm is not suitable for large complexes so it is needed some procedure to reduce the size of the complexes without loosing homological information. In this section we recall the notion of AM-model, which represents a way of simplify the calculation of homology groups and represents our main tool for extracting homological information of digital objects. In this section we mainly follows [31].

Definition 2.32 (Reduction)**.**
A reduction from $\mathcal{C} = (\mathsf{C}_\bullet, \mathrm{d}_\bullet)$ *to* $\mathcal{C}' = (\mathsf{C}'_\bullet, \mathrm{d}'_\bullet)$ *is a triple* (f, g, h) *of chain maps* $f : \mathcal{C} \to \mathcal{C}'$ *(projection),* $g : \mathcal{C}' \to \mathcal{C}$ *(inclusion) and* $h_\bullet : C_\bullet \to C_{\bullet+1}$ *(chain homotopy or integral operator) that satisfy the following conditions:*

(a) $\mathrm{id}_\mathcal{C} - g \circ f = \mathrm{d} \circ h + h \circ \mathrm{d}$

(b) $f \circ g = \mathrm{id}_{\mathcal{C}'}$

(c) $f \circ h = 0$

(d) $h \circ g = 0$

(e) $h \circ h = 0$

This is a classical notion in homological algebra and algebraic topology; see e.g. [10, §12] and comments on the terminology and applications in [14, p. 86]. Note that because of the condition (a), if there exists a chain contraction from $\mathcal{C}$ to $\mathcal{C}'$ then their homology modules are isomorphic.

Definition 2.33 (AM model)**.**
An algebraic minimal model (introduced in [14]), or an AM model for short, of a cell complex K is a chain contraction from $\mathcal{C}(K)$ to a chain complex $\mathcal{M} = (\mathsf{M}_*, \mathrm{d}_{\mathcal{M}})$ such that each M_p is a free $\mathcal{R}$-module and all the non-zero elements in the SNF of each $(\mathrm{d}_{\mathcal{M}})_p$ are non-invertible in $\mathcal{R}$.

An AM model exists for every cell complex, and any two AM models for the same complex are isomorphic.

Note that we are not distinguishing here the cellular structure of K, as it can be any of simplicial, cubical or CW complex structure.

Cohomology

Cohomology is the result of a process of dualization in homology. However, only the chain complex is dualized, instead of homology groups. This dualization operation enriches the homological information of a topological space not only with cohomology groups but with a larger structure called cohomology ring. In this section we review the construction of cohomology groups of any cell complex and how to define some "products" of chains to induce a "product" operation in cohomology suitable to define the mentioned ring structure.

As usual, given a $\mathcal{R}$-module C, $\mathrm{Hom}(\mathsf{C}, \mathcal{R})$ denotes the homomorphisms from C to $\mathcal{R}$.

Definition 2.34 (Dual chain complex)**.**
Let $\mathcal{C} = (\mathsf{C}_\bullet, \mathrm{d}_\bullet)$ be a chain complex. Its dual complex $\mathcal{C}^* = (\mathsf{C}^\bullet, \delta^\bullet)$ is defined as follows:

1. *The module of cochains is* $\mathsf{C}^p = \mathrm{Hom}(\mathsf{C}_p, \mathcal{R})$

2. *The codifferential* $\delta^\bullet : \mathsf{C}^\bullet \to \mathsf{C}^{\bullet+1}$ *is defined by* $\delta^p(\varphi^{(p)}) = \varphi^{(p)} \circ \mathrm{d}_{p+1}$ *where* $\varphi^{(p)}$ *is a* p*-cochain.*

Basically, a p-cochain is an homomorphism that assigns an element in $\mathcal{R}$ to every p-cell. As $\mathcal{R}$-modules, C_p is isomorphic to C^p, hence the real difference between a chain complex and a cochain complex is in the connection morphism. In a chain complex, the differential map decreases the dimension while the codifferential increases the dimension. From a matricial point of view, the matrix of δ_p is the transpose of the matrix of d_{p+1}. This easily proves that $\delta^{p+1} \circ \delta^p = 0$.

If we define the submodule of cocycles as $\mathsf{Z}^p = \ker \delta^p$ and the submodule of coboundaries as $\mathsf{B}^p = \operatorname{Im} \delta^{p-1}$, the p-th cohomology group of the cochain complex is defined as $\mathsf{H}^p = \mathsf{Z}^p / \mathsf{B}^p$.

Again, this definition does not depend on the choice of the cellularization of the given topological space.

If p-th homology generators can be understood as p-dimensional holes (subspaces homotopy equivalent to S^p), p-th cohomology generators can be viewed as "paths" to cut the complex along them in order to remove a hole.

Most of the time, homoloy and cohomology groups are not enough to distinguish two topological spaces. In some of these cases, establishing some relations between (co)homology generators open the possibility of detecting differences between the spaces. The following section is devoted to one of this operations: the cup product.

Cup product

Sometimes, (co)homology groups are not enough to distinguish two topological spaces. For example, the torus $T^2 = S^1 \times S^1$ and the wedge $S^2 \vee S^1 \vee S^1$ have the same (co)homology groups but are not homotopy equivalents (both spaces have non isomorphic fundamental groups, so they cannot be homotopy equivalents as fundamental group is homotopy invariant). Therefore, (co)homology groups need to be enriched with more elaborated algebraic structure. Is in this place where cup product arises.

Cup product is an operation on cohomology classes. Given two cohomology classes $[\varphi^{(p)}]$ and $[\psi^{(q)}]$, its cup product is a $(p+q)$ cohomology class. Of course, cup product is closely associated with the relationship between the (co)homology of the Cartesian product of two spaces and the (co)homology of the factors.

Cup product on singular or simplicial cohomology is defined as follows:

$$(\varphi^{(p)} \smile \psi^{(q)})(\sigma) = \varphi^{(p)}(\sigma_0^p) \cdot \psi^{(q)}(\sigma_p^{p+q}) \tag{2.13}$$

where $\sigma = \langle v_0, \ldots, v_{p+q} \rangle$ and $\sigma_i^j = \langle v_i, v_{i+1}, \ldots, v_j \rangle$.

This simple definition is not suitable to be directly translated to cubical or cellular cohomology. Instead, it is required the intervention of the chain map **diag** induced by the diagonal map $\mathrm{diag} : x \in X \mapsto (x, x) \in X \times X$. Concretely, the cup product is given by the following composition

$$\smile : \mathcal{C}^*(X) \times \mathcal{C}^*(X) \to \mathcal{C}^*(X \times X) \to \mathcal{C}^*(X) \tag{2.14}$$

where the first map is the Küneth map and the second is the map $\mathbf{diag}^*$. Recall that cup product is defined in cohomology as in homology the map diag goes in the opposite direction of the one required.

In the previous example, if $H^1(T^2) = \mathbb{Z}[a] \oplus \mathbb{Z}[b]$ and $H^1(S^2 \vee S^1 \vee S^1) = \mathbb{Z}[a'] \oplus \mathbb{Z}[b']$, then $a \smile b \neq a' \smile b'$, hence both spaces cannot be homotopy equivalent.

In order to compute the cup product, it is helpful the calculation of the approximation of the diagonal map. This approximation map is computed in many different ways depending on the cellularization of the space. The worst case is the CW decomposition, where the diagonal has to be constructed inductively in the dimension, as states the Cellular Approximation Theorem. However, we will not require the application of that result.

The approximation of the diagonal map presents a very simple aspect in the case of simplicial homology. In this case, its definition is given below.

$$\mathbf{diag}^{\mathrm{Simp}}(\langle v_0, \ldots, v_n \rangle) = \sum_{p=0}^{n} \langle v_0, \ldots, v_p \rangle \otimes \langle v_p, \ldots, v_n \rangle \tag{2.15}$$

However, the cubical case is not as simple as the simplicial one. For example see [20] where an important proportion of the paper is devoted to the construction of a cubical approximation to the diagonal map. We adopt the ideas in [31, 38], giving as a result the following definition for $\mathbf{diag}^{\mathrm{Cub}}$

$$\mathbf{diag}^{\mathrm{Cub}}\left(\sigma^{(n)}\right) = \sum_{J \subset \{1,\ldots,n\}} \rho_{J,J'} \lambda^0_{J'}(\sigma) \otimes \lambda^1_J(\sigma) \tag{2.16}$$

where $\rho_{J,J'} = (-1)^\nu$, $\nu = \#\{(i,j) \in J \times J' \,|\, j < i\}$ and $J' = \{1, \ldots, n\} \setminus J$.

CHAPTER 3

Parallelization strategy for homological calculation based on membrane computing

Most applications of topological data analysis (TDA for short) are characterized by using a very large data set from which the internal structure is revealed through TDA. This structure is extracted in the form of connected components, holes and voids of higher dimensions. This characterization is formally described by the notion of homology. Calculation of homology requires a combinatorial representation of space as a cellular complex (usually simplicial or cubical).

These applications require calculating real-time homology over a growing data set. This fact makes fundamental the development of parallel algorithms for calculating homology. One parallelization strategy based on membrane computing is shown below.

Natural Computing studies new computational paradigms inspired by Nature. It abstracts the way in which Nature computes, giving rise to new models of computation. There are several research fields currently well established in Natural Computing. Among them, Genetic Algorithms introduced by Holland [18] that are inspired by evolution and natural selection in order to find an optimal solution among a large set of feasible solutions; Neural Networks, introduced by McCulloch and Pitts [28], that are based on the interconnection of neurons in brain; or DNA-based molecular computing, initiated by Head [17].

Membrane Computing is a research area in Natural Computing based on the interpretation of the processes that take place inside the cells of living beings as computations.

Computational devices in Membrane Computing are called P systems. In general, a P system consists of a membrane structure within which there are multisets of objects, which evolve according to given rules, in an asynchronous non-deterministic maximally parallel manner. From the foundational article [32], different models of P systems have been studied. Based on their architecture, these models can be classified into two broad categories: cell-like P systems and tissue-like P systems. In cell-like P systems, membranes are organized hierarchically into a tree-like structure. The inspiration of this architecture is the set of organs inside the cell. Each and every one of them carries out its biological processes in parallel.

Tissue-like P systems were introduced in [26, 27]. These types of P systems are inspired by intercellular communication and cooperation between neurons. The mathematical model of these devices is a network of processors that manipulate symbols and communicate them through pre-established channels. Communication between cells is based on symport / antiport rules. Symport rules move objects across a membrane in a single direction, while antiport rules move objects in opposite directions.

In tissue-like P systems the membrane structure is a general undirected graph. The edges of this graph are not given explicitly, but are deduced from the set of rules. Since the initial definition of the tissue-like P systems, several lines of research have been developed and other variants have appeared (see, for example, [1, 2, 13, 23, 30]).

Catalyst P systems were introduced by Păun in [32]. The fundamental characteristic of these P systems is the presence of objects in the membranes that are not consumed by the application of the rules, but their presence is the membrane is necessary for the rule to be chosen as applicable. Catalysts have been studied extensively in Membrane Computing (see, e.g., [12, 22, 24]).

This is not the first relation of Membrane Computing and Computational Algebraic Topology. For instance, in [34] Reina-Molina et al present a Membrane Computing algorithm to solve the problem of thinning 2D and 3D images using cell complexes. In [4, 33], tissue-like P systems are used to segment an image, in [5, 6, 7, 8, 9] Membrane Computing is used for calculating homology groups of 2D digital images.

We present here an implementation of a Membrane Computing

framework to adapt the ideas behind Discrete Morse Theory in order to develop algorithms in order to solve the problem of computing homology groups of 2D and 3D digital binary images. This work is based in [35].

3.1 Membrane computing

In this section we present various definitions concerned with Membrane Computing.

We will use a variant of tissue-like P systems where the application of the rules are regulated by *promoters* and *inhibitors*. These promoters have a clear biological inspiration. The rule is applied if the reactants are present, but it is also necessary the presence of all the promoters and none of the inhibitors in the corresponding cell. The promoters are not *consumed* nor *produced* by the application of the rule, but if they are not in the cell, the rule cannot be applied. In one step, each reactant in a membrane can be used only for one rule, but if several rules need the presence of the same promoter, then the presence of one unique copy of the promoter suffices for the application of all the rules.

In the general case, if there are several possibilities, the rule is non-deterministically chosen, but sometimes a priority relation between rules is considered, so the concept of *priority* in our P systems is required.

Next, the formal definition of these P systems is recalled.

Definition 3.1 (Tissue-like P system with catalysts and priorities)**.** *A* tissue-like P system with promoters, inhibitors, priorities and input *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 q *is the number of cells (or membranes) of the P system and*

1. Γ *is a finite alphabet, whose symbols are called objects. These objects can be placed in the cells or in the surrounding space (called the* environment*).*

2. $\Sigma \subseteq \Gamma$ *is the input alphabet. The* input *of the computation performed by the P system is encoded by using this alphabet.*

3. $\mathcal{E} \subseteq \Gamma$ *is a finite alphabet representing the set of the objects in the environment. Following a biological inspiration, the objects in the environment are available in an arbitrary large amount of copies;*

4. *$w_1, \ldots, w_q$ are strings over Γ representing the multisets of objects placed inside the cells at the beginning of the computation;*

5. *$\mathcal{R}$ is a finite set of rules of the following form:*

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

6. *Pri is a finite set of relations $R_i > R_j$, where R_i and R_j are rules from $\mathcal{R}$. It means that if R_i and R_j can be applied, then the application of R_i has* priority *on the application of R_j.*

7. *$i_{in} \in \{1, 2, \ldots, q\}$ denotes the input cell, i.e., the cell where the* input *of the computation will be placed.*

8. *$i_{out} \in \{1, 2, \ldots, q\}$ denotes the output cell, i.e., the cell where the* output *of the computation will be placed.*

Informally, a tissue-like P system with promoters, inhibitors and priorities of degree $q \geq 1$ can be seen as a set of q cells labeled by $1, 2, \ldots, q$. The cells are the nodes of a virtual graph, where the edges connecting the cells are determined by the communication rules of the P system, i.e., as usual in tissue-like P systems, the edges linking cells are not provided explicitly: if a rule $(pro\,\neg inh \,|\, i, u/v, j)$ is given, then cells i and j are considered linked. The application of a rule $(pro\,\neg inh \,|\, i, u/v, j)$ consists of trading the multiset u (initially in the cell i) against the multiset v (initially in j). After the application of the rule, the multiset u disappears from the cell i and it appears in the cell j. Analogously, the multiset v disappears from the cell j and it appears in the cell i. The trade can also be between one cell and the environment, labeled by 0. The rule is applied if in the cell with label i the objects of pro are present in the cell i (*promoters*), while any of the objects in inh do not appear in the cell (*inhibitors*). The promoters or the inhibitors are not modified by the application of the rule. If the promoters and inhibitors are empty, we write $(i, u/v, j)$ instead of $(\emptyset\,\neg\emptyset|\, i, u/v, j)$. Finally, we write $(pro\,|i, u/v, j)$ or $(\neg inh\,|i, u/v, j)$ when only promoters or inhibitors appear, respectively.

As usual, some objects not belonging to $\mathcal{E}$ can arrive to the environment during a computation. So, in a configuration (not initial) it could be found two types of objects in the environment: firstly, those which belong to the environment and appear in an arbitrary large number of copies. Secondly, those which do not belong to the environment but have been sent to the environment by the application of a rule.

Rules are used as usual in the framework of membrane computing, that is, in a maximally parallel way (a universal clock is considered). A *configuration* is an instantaneous description of the P system and it is represented as a tuple $(w_0, w_1, \ldots, w_q)$, where w_0 is the multiset of objects from $\Gamma - \mathcal{E}$ placed in the environment (initially, $w_0 = \emptyset$). 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 configuration is *halting* when no rules can be applied to it. A *computation* is a sequence of computation steps such that either it is infinite or it is finite and the last step yields a halting configuration (i.e., no rules can be applied to it). Then, a computation halts when the P system reaches a halting configuration. The output of a computation is collected from its halting configuration by reading the objects contained in the output cell.

Definition 3.2 (Distributed scheme)**.**
A dP scheme (of degree $s \geq 1$) is a construct

$$\Delta = (\Gamma, \Sigma, \Pi_1, \ldots, \Pi_s, R, i_{in}, i_{out})$$

where:

1. Γ *is an alphabet of (communicated) objects.*
2. $\Sigma \subset \Gamma$ *is the input alphabet, used as input for the module* $\Pi_{i_{in}}$*.*
3. $\Pi_1, \ldots, \Pi_s$ *are tissue like P systems with promoters, inhibitors and priorities whose alphabets contain* Γ*. Each* Π_i *is called a component or module of* Δ*.*
4. *R is a set of distribution rules of the form* $(pro\neg inh|i, u \to v, j)$ *where pro, inh, u and v are multisets on* Γ*,* $0 \leq i \neq j \leq s$ *and* $i \neq 0$*. The objects in pro are called promoters of the rule and the objects in inh are called inhibitors, as usual in tissue like P systems with promoters and inhibitors.*
5. i_{in} *and* i_{out} *represents the modules used as input and output, respectively.*

Informally, the evolution of a dP scheme can be summarized as follows. A universal common clock is used for all the modules in the scheme. Initially all the modules start evolving together, each of them with their respective initial configuration except the input module, which also uses the input alphabet Σ as input. On each computation step, every time a

module reaches its halting configuration, a maximal multiset of distribution rules is selected. Then, these rules are applied and some objects are sent from the output membrane of some modules to the input membrane of another. Then, the evolution starts again until all the modules have reached their halting configuration and no distribution rule can be selected. We use an special destination module, indexed by 0. Every time a rule $(pro\neg inh|i, u \rightarrow v, 0)$ is executed, the multiset u is removed from the output membrane of the ith module. This is used as some kind of garbage collection.

3.2 Encoding images as cubical complexes and cubes as tuples

In this section, we work with a cubical cell version of a digital image. A common problem appearing in the development and implementation of algorithms with cubical complexes is the requirement of a huge amount of memory resources. Below, an efficient encoding of cubical cells is described.

Given two non-negative integers k and $n \geq 2$, the following auxiliary sets are defined

$$T_n^k := \{0, 1, 2, \ldots, n-1\}^k, \tag{3.1}$$

$$\overline{T}_n^k := T_{2n-1}^k, \tag{3.2}$$

$$I_{n,k} := T_{n^k} = \{0, 1, 2, \ldots, n^k - 1\}, \tag{3.3}$$

$$\overline{I}_{n,k} := T_{(2n-1)^k} = \{0, 1, 2, \ldots, (2n-1)^k - 1\}. \tag{3.4}$$

The set of points[1] for the source images is the set $T_n^k \subset \mathbb{Z}^k$ equipped with a *cubic* neighborhood function, described as follows: two points $\mathbf{i} = (i_1, \ldots, i_k)$ and $\mathbf{j} = (j_1, \ldots, j_k)$ are said $(2k)$-*adjacent* if $\sum_{p=1}^k |i_p - j_p| = 1$. The neighborhood function is given by

$$N(i_1, \ldots, i_k) = \left\{ (j_1, \ldots, j_k) \in T_n^k : \sum_{p=1}^k |i_p - j_p| = 1 \right\}$$

This function restricted to $k = 2$ defines 4-adjacency and to $k = 3$ defines 6-adjacency.

[1]The reader is supposed to be familiar with concepts of Image Algebra. For a detailed text, see [36].

Let $\mathcal{I} : T_n^k \to \{0,1\}$ be a k-D binary image of size n^k, where the set of points in the object (or black points) is given by $\mathcal{I}^{-1}(1)$. Let $K = K(\mathcal{I})$ be the cubical cell complex associated to $\mathcal{I}$. In K, the 0-cells are points in the object, the 1-cells represent pairs of $(2k)$-adjacent points, the 2-cells unit squares where its edges are pairs of $(2k)$-adjacent points, and so on. In general, each p-cell is a p-dimensional unit hypercube whose edges are determined by pairs of $(2k)$-adjacent points.

A cubical cell is encoded as a tuple through the function $\mathbb{T} : \mathcal{K}^k \to \mathbb{Z}^k$ defined as follows

$$\mathbb{T}(\sigma) = (\sup I_1(\sigma) + \inf I_1(\sigma), \ldots, \sup I_k(\sigma) + \inf I_k(\sigma)) \tag{3.5}$$

The main property of $\mathbb{T}$ is given in the following lemma.

Lemma 3.3.
The function $\mathbb{T}$ is a bijection and $\mathbb{T}(\mathcal{K}_{*,n}^k) = \overline{T}_n^k$.

Proof. First at all, let us prove that $\mathbb{T}$ is injective. Let σ and μ be two distinct cells. Then, let $j_0 \in \{1, 2, \ldots k\}$ be the first index such that $I_{j_0}(\sigma) \neq I_{j_0}(\mu)$. The following cases may come up:

- $I_{j_0}(\sigma) = (a, a+1)$ and $I_{j_0}(\mu) = (b, b+1)$ with $a \neq b$. Then
$$\inf I_{j_0}(\sigma) = a \neq b = \inf I_{j_0}(\mu)$$
- $I_{j_0}(\sigma) = (a, a+1)$ and $I_{j_0}(\mu) = (b)$. Then, either $a \neq b$ or $a+1 \neq b$. In the first case,
$$\inf I_{j_0}(\sigma) = a \neq b = \inf I_{j_0}(\mu)$$
and in the second case
$$\sup I_{j_0}(\sigma) = a+1 \neq b = \sup I_{j_0}(\mu)$$

In all the cases showed above $\inf I_{j_0}(\sigma) + \sup I_{j_0}(\sigma) \neq \inf I_{j_0}(\mu) + \sup I_{j_0}(\mu)$ and, hence, $\mathbb{T}(\sigma) \neq \mathbb{T}(\mu)$.

To prove that $\mathbb{T}$ is surjective, it is enough to find a cell $\sigma = I_1 \times \cdots \times I_k$ such that, given a tuple $(c_1, \ldots, c_k) \in \mathbb{Z}^k$, $\mathbb{T}(\sigma) = (c_1, \ldots, c_k)$. It is easily verified that such cell σ si given by setting each interval I_p as follows

$$I_p = \begin{cases} \left(\frac{c_p-1}{2}, \frac{c_p+1}{2}\right) & \text{if } c_p \equiv 1 \pmod 2 \\ \left(\frac{c_p}{2}\right) & \text{if } c_p \equiv 0 \pmod 2 \end{cases} \tag{3.6}$$

for $1 \leq p \leq k$.

Finally notice that, if $I = (m, m+1)$ with $0 \leq m, m+1 \leq n-1$, then $\sup I + \inf I = 2m + 1 \leq 2n - 2$. In consequence, $\sup I + \inf I \in T_{2n-1}$, proving the last statement of the Lemma. □

As an example, the cell $\sigma = (0,1) \times (3,4) \times (2)$ is encoded as $\mathbb{T}(\sigma) = (1, 7, 4)$.

A cubical cell can also be encoded as an integer. Concretely, a cubical cell[2] is encoded as a number $\mathbb{I}_{n,k}(\sigma)$ in $\overline{I}_{n,k}$ as follows:

$$\mathbb{I}_{n,k}(\sigma) = \mathbb{I}_{n,k}(c_1, \ldots, c_k) = \sum_{p=1}^{k} c_p \cdot (2n-1)^{k-p} \tag{3.7}$$

Lemma 3.4.
The function $\mathbb{I}_{n,k} : \overline{T}^k_n \to \overline{I}_{n,k}$ defined above is a bijection.

Proof. To prove that the function $\mathbb{I}_{n,k}$ is surjective is enough to find a cell σ_l for any $l \in \overline{I}_{n,k}$ such that $\mathbb{I}_{n,k}(\sigma_l) = l$. The cell $\sigma_l = (c_1, \ldots, c_k)$ such that

$$c_p = \lfloor \frac{l}{(2n-1)^{(k-p)}} \rfloor \pmod{(2n-1)} \text{ for } 1 \leq p \leq k \tag{3.8}$$

satisfies that $\mathbb{I}_{n,k}(\sigma_l) = l$.

To prove that $\mathbb{I}_{n,k}$ is injective, let $\sigma, \sigma' \in \overline{T}^k_n$ be two cells such that $\mathbb{I}_{n,k}(\sigma) = \mathbb{I}_{n,k}(\sigma')$. Then, if $\sigma = (c_1, \ldots, c_k)$ and $\sigma' = (c'_1, \ldots, c'_k)$,

$$0 = \mathbb{I}_{n,k}(\sigma') - \mathbb{I}_{n,k}(\sigma) = \sum_{p=1}^{k} (c'_p - c_p)(2n-1)^{(k-p)}$$

hence, taking iteratively remainders modulo $(2n-1)$, it turns out that $c'_p \equiv c_p \pmod{(2n-1)}$ and, as $0 \leq c'_p, c_p < 2n-1$, it is proved that $c'_p = c_p$. □

3.3 Membrane computing implementation of homological calculation

In previous section, the formal requirements for a Membrane Computing algorithm are presented, i.e. all the elements required to formally describe

[2]Lemma 3.3 grants the identification of cubical cells in a cubical complex in $\mathbb{R}^k$ with k-tuples.

a P system. Hence, a language must be defined in order to characterize all the elements present in each membrane. The main goal in this paper is the introduction of a membrane computing framework flexible enough to be useful in many others applications than those presented below. Hence, we define now those essential notions in order to represent the objects and relations already outlined in Section 2.2.

Let $K \subset \overline{T}_n^k$ be a cubical complex. Let d represents the boundary map in the associated chain complex. The cubical cells in K are represented as σ_i, for some $i \in \overline{I}_{n,k}$. The cubical geometry of K is achieved using objects $U_{ij,d}^{\pm}$ for $i, j \in \overline{I}_{n,k}$ and $1 \leq d \leq k$. The presence of such an object is interpreted as the cell σ_i is in the boundary of σ_j and the vector joining their barycenters (from lower dimension to higher dimension cell) is $\pm\frac{1}{2}\vec{u}_d$, where $\vec{u}_d$ is the d-th vector in the canonical base in $\mathbb{R}^k$.

As the cubical complex K is usually built from a binary k-D image[3] $\mathcal{I} : T_n^k \to \{0, 1\}$, black pixels in $\mathcal{I}^{-1}(1)$ are denoted as $B_{\mathbf{i}}$ for $\mathbf{i} \in T_n^k$.

The dimension of a cubical p-cell σ_i is represented by the presence of the object d_{ip}, for $i \in \overline{I}_{n,k}$ and $0 \leq p \leq k$.

The boundary map is symbolized by objects $\delta_{ij}^{\pm}$ whenever $\langle \sigma_i, \mathrm{d}\sigma_j \rangle = \pm 1$, respectively. Recall that cubical complexes are regular, so it is ensured that $\langle \sigma, \mathrm{d}\mu \rangle \in \{0, \pm 1\}$ for any pair of cells $\sigma^{(p)}$ and $\mu^{(p+1)}$. Sometimes the knowledge of how many cells share a given cell in their border is required. The presence of objects $\mathbf{D}_{if}$ means that the set $\{\sigma_j \in K : \langle \sigma_i, \mathrm{d}\sigma_j \rangle \neq 0\}$ has f elements.

Vector fields are represented by the presence of objects $V_{ij}^{\pm}$ where the sign is given by the sign of $\langle \sigma_i, \mathrm{d}\sigma_j \rangle$ with $i, j \in \overline{I}_{n,k}$. Objects V_i depict that the cubical cell σ_i is used as part of a vector. If a cubical cell is critical for a given vector field V, it is denoted by the presence of some object $\mathbf{C}_i$ for $0 \leq i \leq N$.

In order to stand for the operator φ defined in section 2.2, objects $\phi_{ij}^{\pm}$, for $0 \leq i, j \leq N$, are used.

Hence, the common starting language for all the algorithms designed in

[3] In case of 2D images, $B_{\mathbf{i}} = B_{i_1 i_2}$.

the proposed framework is defined below:

$$\begin{aligned}\Gamma_0 =&\{B_{\mathbf{i}} : \mathbf{i} \in T_n^k\} \cup \left\{\sigma_i : i \in \overline{I}_{n,k}\right\} \cup \{\mathbf{C}_i : i \in \overline{I}_{n,k}\} \cup \\ &\cup \left\{\mathrm{d}_{ip} : i \in \overline{I}_{n,k} \wedge 0 \leq p \leq k\right\} \cup \left\{\mathbf{D}_{if} : i \in \overline{I}_{n,k} \wedge 0 \leq f \leq 2^k\right\} \cup \\ &\cup \left\{\delta_{ij}^{\pm} : i, j \in \overline{I}_{n,k}\right\} \cup \left\{V_{ij}^{\pm} : i, j \in \overline{I}_{n,k}\right\} \cup \left\{V_i : i \in \overline{I}_{n,k}\right\} \cup \{V\} \cup \\ &\cup \left\{U_{ij,p}^{\pm} : i, j \in \overline{I}_{n,k} \wedge 0 \leq p \leq k\right\} \cup \left\{\phi_{ij}^{\pm} : i, j \in \overline{I}_{n,k}\right\}\end{aligned}$$

The framework presented here does not only consist of a language, but also some dP modules and distribution rules are defined. Firstly, the dP modules have to be such that allows performing only simple tasks in each module. Hence, the implementation of the Membrane Computing algorithm in current parallel devices is simplified. Secondly, the development of algorithms in Computational Algebraic Topology usually reduces to simplify some cell complexes by using Discrete Morse Theory. We specify here some modules related to the process of removal of cells which are the head and tails of discrete gradient vectors.

Previously mentioned common tasks are introduced as modules of a dP scheme. Namely, two main tasks are to be defined:

1. Build a cubical complex from a binary k-D image.
2. Specify the Morse complex from a given cubical complex and a gradient vector field.

For any $i \in \overline{I}_{n,k}$, the cubical cell $\sigma(i) \in \mathcal{K}_{*,n}^k$ is defined as

$$\sigma(i) = \mathbb{T}^{-1}(\mathbb{I}_{n,k}^{-1}(i))$$

To determine the vertices of a cubical cell σ the function $\mathrm{gen} : \mathcal{K}_n^k \to 2^{\mathbb{Z}^k}$ defined as

$$\mathrm{gen}(I_1 \times \ldots \times I_k) = \{\inf I_1, \sup I_1\} \times \cdots \times \{\inf I_k, \sup I_k\} \tag{3.9}$$

is used. For example,

$$\mathrm{gen}\left((1,2) \times (2) \times (0,1)\right) = \{1,2\} \times \{2\} \times \{0,1\} = \{(1,2,0), (1,2,1), (2,2,0), (2,2,1)\}$$

are the vertices of a square in $\mathbb{R}^3$.

Let σ and μ be cubical cells such that $\sigma \in \mathrm{d}\mu$. Let $b(\sigma)$ and $b(\mu)$ represent the barycenter of cells σ and μ, respectively. Finally, denote

the vector $\overrightarrow{b(\sigma)b(\mu)}$ as $\vec{v}_{\sigma\mu}$. For example, let $\sigma = (0,1) \times (2) \times (2,3)$ and $\mu = (0,1) \times (2,3) \times (2,3)$. Then, $\vec{v}_{\sigma\mu} = \left(0, \frac{1}{2}, 0\right)$ as the barycenters are, respectively, $b(\sigma) = \left(\frac{1}{2}, 2, \frac{5}{2}\right)$ and $b(\mu) = \left(\frac{1}{2}, \frac{5}{2}, \frac{5}{2}\right)$.

Notice that the barycenter of a cubical cell is defined as

$$b(I_1 \times \cdots \times I_k) = \left(\frac{\inf I_1 + \sup I_1}{2}, \ldots, \frac{\inf I_k + \sup I_k}{2}\right) \tag{3.10}$$

Definition 3.5 (P system to build cubical complex)**.**
Let $\mathcal{I} : T_n^k \to \{0,1\}$ be a binary k-D image. The tissue like P system with promoters, inhibitors and priorities Π_{Cub} is defined as

$$\Pi_{Cub} = \{\Gamma, \Sigma, \mathcal{E}, w_1, \mathcal{R}, Pri, i_{in}, i_{out}\}$$

where

- $\Gamma = \Gamma_0$
- $\Sigma = \{B_{\mathbf{i}} : \mathbf{i} \in \mathcal{I}^{-1}(1)\}$
- $\mathcal{E} = \Gamma_0$
- $w_1 = \emptyset$
- $\mathcal{R}$ *is the following set of rules:*
  - $R_1 \equiv (\{B_{\mathbf{i}} : \mathbf{i} \in \mathrm{gen}(\sigma(l))\}\, \neg\{\sigma_l\}|1, \lambda/\sigma_l \mathbf{C}_l, 0)$ *for* $l \in \overline{I}_{n,k}$
  - $R_2 \equiv \left(\{\sigma_i, \sigma_j\}\, \neg\{\delta_{ij}^{\pm}\}|1, \lambda/\delta_{ij}^{\pm}, 0\right)$ *for* $i, j \in \overline{I}_{n,k}$ *and* $\langle \sigma(i), \mathrm{d}\sigma(j)\rangle = \pm 1$
  - $R_3 \equiv \left(\{\sigma_i, \sigma_j\}\, \neg\{U_{ij,d}^{+}\}|1, \lambda/U_{ij,d}^{+}, 0\right)$ *for* $i, j \in \overline{I}_{n,k}$, $1 \leq d \leq k$, $\langle \sigma(i), \mathrm{d}\sigma(j)\rangle \neq 0$ *and* $\vec{v}_{\sigma(i)\sigma(j)} \cdot \vec{u}_d = \frac{1}{2}$

  Note that $\vec{u}_d$ is the d-th vector of the canonical base in $\mathbb{R}^k$.
- $Pri = \emptyset$
- $i_{in} = i_{out} = 1$

Theorem 3.6.
Given a k-D binary image $\mathcal{I}$, the corresponding P system Π_{Cub} builds the cubical complex for $\mathcal{I}$ in two computation steps.

Remark. Let $\mathcal{I} : T_n^k \to \{0,1\}$ be a binary k-D image. Let K be the cubical complex built from $\mathcal{I}$. In the initial configuration, the first membrane consists on objects $B_{\mathbf{i}}$ for $\mathbf{i} \in \mathcal{I}^{-1}(1)$. In this situation, only some rules in R_1 can be selected, because the other rules require the presence of objects σ_i. The rules in R_1 that can be selected are those for $l \in \overline{I}_{n,k}$ such that all the points in $\mathrm{gen}(\sigma(l))$ are in $\mathcal{I}^{-1}(1)$. Therefore, the first computation step builds the objects σ_l that represents cubical cells in the complex and, also, introduce the mark for critical cells ($\mathbf{C}_l$).

In the second configuration, no rule in R_1 can be selected since the proper objects representing each cubical cell act as inhibitors for this rules. However, the rules in R_2 and R_3 are susceptible to be selected. In fact, the selected rules in R_2 introduce into the first membrane the objects that symbolize the boundary map ($\delta_{ij}^{\pm}$). The selected rules in R_3 introduce the objects that mark the (positive) direction of the boundary barycentric vector $\vec{v}_{\sigma\mu}$ for every pair of cells $\sigma, \mu \in K$ such that $\sigma \in \mathrm{d}\mu$ and $\vec{v}_{\sigma\mu} \cdot \vec{u}_d = \frac{1}{2}$.

The application of rules in R_2 and R_3 prevent the selection of any other rule in R_2 or R_3 any time, because the inserted objects act as inhibitors. Hence, the next computation step (the third) makes the system reach the halting condition, inasmuch as no rule can be selected.

Remark 3.7.
If σ is in the boundary of μ, then both cells share all its intervals except one, degenerated in σ and non-degenerated in μ. We suppose here that this interval is the d-th. Then, all the coordinates of the vector $\vec{v}_{\sigma\mu}$ are 0, except the d-th one, whose value is $\frac{1}{2}$ and, hence, $\vec{v}_{\sigma\mu} = \frac{1}{2}\vec{u}_d$.

□

Definition 3.8 (P system to compute the Morse complex)**.**
Let K be a cubical complex and d *the associated boundary map. Let V be a gradient vector field in K. The P system Π_{Flow} is defined as*

$$\Pi_{Flow} = (\Gamma, \Sigma, \mathcal{E}, w_1, \mathcal{R}_{Flow}, Pri, i_{in}, i_{out})$$

where

- *The alphabet of objects is given by $\Gamma = \Gamma_0 \cup \{E_{ij} : i, j \in \overline{I}_{n,k}, i \neq j\}$.*

- *The input alphabet is given by*

$$\begin{aligned}\Sigma =&\{\sigma_i : i \in \mathbb{I}_{n,k}(\mathbb{T}(K))\} \cup \{\delta_{ij}^{\pm} : i,j \in \mathbb{I}_{n,k}(\mathbb{T}(K)) \wedge \langle \sigma(i), \mathrm{d}\sigma(j)\rangle = \pm 1\} \cup \\ &\cup \{V_{ij}^{\pm}, V_i, V_j : i,j \in \mathbb{I}_{n,k}(\mathbb{T}(K)) \wedge V(\sigma(i)) = \pm\sigma(j)\} \cup \\ &\cup \{\mathbf{C}_i : i \in \mathbb{I}_{n,k}(\mathbb{T}(K)) \wedge V(\sigma(i)) = 0 \wedge V^{-1}(\sigma(i)) = \emptyset\} \cup \\ &\cup \{E_{ij} : i,j \in \mathbb{I}_{n,k}(\mathbb{T}(K)) \wedge \langle \sigma(j), \mathrm{d}V(\sigma(i))\rangle \neq 0 \wedge i \neq j\}\end{aligned}$$

- *The environment alphabet is given by* $\mathcal{E} = \Gamma_0$.
- *The set of rules* $\mathcal{R}$ *is given by*
  - $R_1 \equiv \left(\{\sigma_i\sigma_j\sigma_l\delta_{il}^{s_1}\delta_{jl}^{s_2}V_{il}^{-s_1}\}\, \neg\{\phi_i\phi_j\}|1, \lambda/\phi_{ij}^{-s_1s_2}\phi_i\phi_j, 0\right)$ *for* $i,j,l \in \overline{I}_{n,k}$, $i \neq j \neq l \neq i$ *and* $s_1, s_2 \in \{+,-\}$
  - $R_2 \equiv \left(\{\sigma_i\mathbf{C}_i\}\, \neg\{\phi_{ii}^{+}\}|1, \lambda/\phi_{ii}^{+}, 0\right)$ *for* $i \in \overline{I}_{n,k}$
  - $R_3 \equiv \left(\{\phi_{i_1i_2}^{s_1}V_{i_2j_1}^{s_2}\phi_{j_1j_2}^{s_3}\}|1, E_{i_2j_1}/\phi_{ij}^{-s_1s_2s_3}, 0\right)$ *for* $i,j,l \in \overline{I}_{n,k}$, $s_1,s_2,s_3 \in \{+,-\}$ *and* $i \neq j \neq l \neq i$.
  - $R_4 \equiv \left(\{\delta_{li}^{s_1}\mathbf{C}_i\mathbf{C}_j\}\, \neg\{\tilde{\delta}_{ji}^{s_1s_2}\}|1, \phi_{lj}^{s_2}/\tilde{\delta}_{ji}^{s_1s_2}\tilde{\mathbf{C}}_i\tilde{\mathbf{C}}_j\tilde{\sigma}_i\tilde{\sigma}_j, 0\right)$ *for* $i,j,l \in \overline{I}_{n,k}$ *with* $i \neq j \neq l \neq i$.
  - $R_5 \equiv \left(1, \tilde{\delta}_{ij}^{+}\tilde{\delta}_{ij}^{-}/\lambda, 0\right)$ *for* $i,j \in \overline{I}_{n,k}$, $i \neq j$.
- $Pri = \emptyset$
- $i_{in} = i_{out} = 1$.

Theorem 3.9.
Let $K \subset \mathcal{K}_{*,n}^k$ *be a cubical complex and* V *a gradient vector field in* K. *The module* Π_{Flow} *computes the Morse complex for* K *associated to* V *in, at most,* $4 + \lfloor \log_2 \nu \rfloor$ *computation steps, where* $\nu \leq n^k$ *is the length of the greatest* V*-path on* K.

Remark. The initial configuration for Π_{Flow} consists in all the elements that codify the cubical complex K (objects σ_i for cubical cells and $\mathbf{C}_i$ marking critical cells), its boundary map (objects $\delta_{ij}^{\pm}$) and the gradient vector field (objects $V_{ij}^{\pm}$). There also are objects E_{ij} depicting that σ_i and σ_j are distinct maximal faces in $V(\sigma_i)$. In this situation, only rules in R_1 and R_2 can be selected, as the other rules require the existence of objects $\phi_{ij}^{\pm}$ introduced in the first membrane by these rules. The application of rules R_1 creates one object $\phi_{ij}^{\pm}$ for each pair of cubical cells $\sigma_i, \sigma_j \in K$ such that $\langle \varphi(\sigma_i), \sigma_j\rangle = \pm 1$. Recall that $\varphi : \mathcal{C} \to \mathcal{C}$ given by $\varphi = \mathrm{id} - \mathrm{d} \circ V$ is the reduced (discrete time) flow of the gradient vector field ([11]).

The application of rules R_2 previously selected introduce objects ϕ_{ii}^{+} for each critical cubical cell σ_i, representing that, for each critical cell μ is $\varphi(\mu) = \mu$. In the current situation, the second configuration, only rules in R_3, R_4 or R_5 are available to be selected.

Recall that objects $\phi_{ij}^{\pm}$ encodes the presence of a V-path from σ_i to σ_j with multiplicity equals to ± 1. Rules in R_3 concatenate two V-paths γ and γ' if there is a cell σ such that the ending cell of γ', named $\sigma(i)$, the starting cell of γ lay on its boundary and $V(\sigma(i)) = \sigma$. Hence, rules in R_3 are applied until a maximal path is constructed. This process needs, at most, $\log_2 \nu$ steps, where ν is the length of the longest maximal V-path.

Rules in R_4 get some objects from the environment, encoding the cells of the Morse complex. Indeed, for each critical path from σ_i to σ_j with multiplicity ± 1, there is an object $\phi_{ij}^{\pm}$, hence for each critical cubical cell σ_l with $\langle \sigma_i, \mathrm{d}\sigma_l \rangle \neq 0$ there will be as many objects $\phi_{ij}^{\pm}$ as the number of critical V-paths from σ_i to σ_j with multiplicity ± 1, respectively.

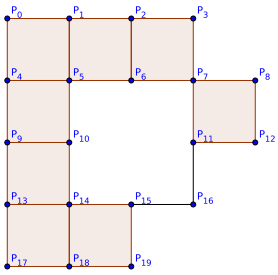
Finally, rules in R_5 cancel pairs of objects $\tilde{\delta}_{ij}^{+}$, $\tilde{\delta}_{ij}^{-}$. The halting configuration of Π_{Flow} has as many objects $\tilde{\delta}_{ij}^{\pm}$ as the incidence of σ_i in the boundary of σ_j.

Notice that the entire process needs, at most, $4 + \lfloor \log_2 \nu \rfloor$ steps, where ν is the length of the longest critical V-path. □

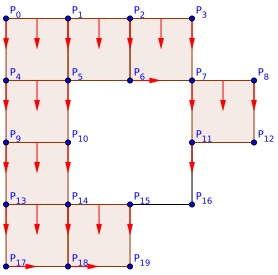
We deal here with a membrane algorithm for computing an optimal gradient vector field for a 3D digital object $\mathcal{I}$. Working with a cubical "cellularization" $K(\mathcal{I})$ of $\mathcal{I}$, we show that the critical cells of dimension p ($p = 0, 1, 2, 3$) of this particular gradient vector field coincide with the pth Betti number of $K(\mathcal{I})$.

Discrete Morse Theory establishes a tool to simplify a cell complex[4] while the homology groups are kept. This can be used to calculate the homology groups of appropriate cell complexes. Concretely, the technique explained below require the complex to be torsion free.

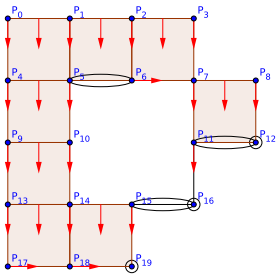
An acyclic vector field over a cubical complex K is built, removing as many cubical cells as possible. In case that the resulting Morse complex has non-null differential, another acyclic vector field is constructed to accomplish the so called *critical cell cancellation* (see section 11 in [11]). In this way, a sequence of Morse reductions is built until a complex with null-differential is "reached". To ensure the finiteness of this process, the



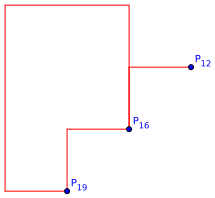
(a) Example of cubical complex.



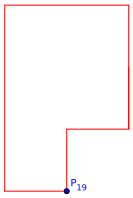
(b) Acyclic vector field.



(c) Critical cells where two kinds of critical cells are shown: ordinary critical cells representing a hole at any dimension and other critical cells showing a "step" configuration which will be canceled later



(d) Critical complex.



(e) Final homological complex with null differential.

Figure 3.1: Example of homology group calculation in a cubical complex.

cubical complex has to be torsion free.

Figure 3.1 presents an example of the calculation of homology groups for a 2D cubical complex. In this example the strategy presented above is shown. First, an acyclic vector field is built such that almost every cubical p-cell σ is paired to one of its co-faces μ (cubical $(p+1)$-cell with $\sigma \in \mathrm{d}\mu$). The algorithm used to determine the acyclic vector field returns two kind of critical cubical cells: on one hand, cubical p-cells that represent a p-dimensional hole and, on the other hand, 1-cells that represents an "step" configuration (as $\langle P_{11}, P_{12}\rangle$ or $\langle P_{15}, P_{16}\rangle$ in figure 3.1). Therefore, another acyclic vector field is required to cancel these spurious critical cells. Notice that the first Morse complex is not a cubical complex but a CW complex. However, it is usually much more simpler than the original cubical complex.

For binary 3D images, it suffices to build up to two acyclic vector fields to calculate an homotopically equivalent cell complex with null differential, which makes homology group calculation trivial. In the paragraphs below, two P modules are defined to perform this task: Π_{Vector} and Π_{Vector_2}.

Definition 3.10 (P system to create an acyclic vector field)**.**
Let $\mathcal{I} : T_n^3 \to \{0,1\}$ be a k-D binary image. The module Π_{Vector} is defined as

$$\Pi_{Vector} = \{\Gamma, \Sigma, \mathcal{E}, w_1, \mathcal{R}, Pri, i_{in}, i_{out}\}$$

where

- $\Gamma = \Gamma_0$
- $\Sigma = \{\sigma_i : i \in \mathbb{I}_{n,k}(\mathbb{T}(K(\mathcal{I})))\}$
- $\mathcal{E} = \Gamma_0$
- $w_1 = \emptyset$
- $\mathcal{R}$ *is the following set of rules:*
  - $R_1 \equiv \left(\neg\{V_iV_j\}|1, \sigma_i\sigma_j\delta_{ij}^{\pm}\mathrm{d}_{ip}U_{ij,d}^{+}C_iC_j/\sigma_i\sigma_j\delta_{ij}^{\pm}U_{ij,d}^{+}V_{ij}^{\mp}V_iV_j, 0\right)$ *for* $i,j \in \overline{I}_{n,3}$, $i \neq j$, $0 \leq p \leq 3$ *and* $1 \leq d \leq 3$
    These rules take a critical cubical p-cell (σ_i), a critical cubical $(p+1)$-cell (σ_j) with $\sigma_i \in \mathrm{d}\sigma_j$ and such that the barycentric vector $\vec{v}_{\sigma_i\sigma_j}$ is $\frac{1}{2}\vec{u}_d$. Then, objects $V_{ij}^{\pm}$ are brought from the environment (this represents that $V(\sigma_i) = \pm\sigma_j$) as long as objects V_i and V_j (used to prevent that a cubical cell is used in two different vectors in the vector field).

[4]CW complex in [11] and cubical complex in this paper.

- *The priorities are given by*

$$\begin{aligned} Pri = &\{R_1[i,j,p,d] > R_1[i,j',p,d'] : i,j,j' \in \overline{I}_{n,3} \wedge 0 \leq p \leq 3 \wedge 1 \leq d < d' \leq 3\} \\ &\cup \{R_1[i,j,p,d] > R_1[i',j,p,d'] : i,i',j \in \overline{I}_{n,3} \wedge 0 \leq p \leq 3 \wedge 1 \leq d < d' \leq 3\} \\ &\cup \{R_1[i,j,p,d] > R_1[j,j',p+1,d'] : i,j,j' \in \overline{I}_{n,3} \wedge 0 \leq p \leq 2 \wedge 1 \leq d \leq 3\} \end{aligned}$$

  where $R_1[i,j,p,d]$ represents the corresponding instance of rule R_1 for the given values of the parameters.

  The priorities in the first set are used to assign to a cubical cell σ_i a cubical cell σ_j with the corresponding barycentric vector parallel to the vector $\vec{u}_d$ with the lowest d as possible.

  The priorities of the second kind are used to choose between two cubical cells σ_i and $\sigma_{i'}$ when another cubical cell σ_j can be assigned to be the image by V. The criteria is again to select the cubical cell with the lowest barycentric vector.

  The last priority type is used to choose whether a cubical cell is chosen as the source cell of a vector or the end cell of another. In this case it is used as the end cell.

- $i_{in} = i_{out} = 1$

Informally speaking, the vector field tries to "go" in the direction of $\vec{u}_1$. When it is not possible to go in that direction, it is used $\vec{u}_2$ and, if it cannot be performed, $\vec{u}_3$ is used. Hence, the flow of V can be thought as a waterfall going downside until it goes from left to right and, finally, rear to back.

Definition 3.11 (P system to reduce a vector field)**.**
Let $\mathcal{I} : T_n^3 \to \{0,1\}$ be a k-D binary image. The module Π_{Vector_2} is defined as

$$\Pi_{Vector_2} = \{\Gamma, \Sigma, \mathcal{E}, w_1, \mathcal{R}, Pri, i_{in}, i_{out}\}$$

where

- $\Gamma = \Gamma_0$
- $\Sigma = \{\sigma_i : i \in \mathbb{I}_{n,k}(\mathbb{T}(\mathcal{M}))\}$ *where $\mathcal{M}$ is the Morse complex built from the acyclic vector field created by Π_{Vector}.*
- $\mathcal{E} = \Gamma_0$
- $w_1 = \emptyset$

- $\mathcal{R}$ *is the following set of rules:*
  - $R_1 \equiv \left(\neg\{V_iV_j\}|1, \sigma_i\sigma_j\delta_{ij}^{\pm}C_iC_j/\sigma_i\sigma_j\delta_{ij}^{\pm}V_{ij}^{\mp}V_iV_jV, 0\right)$ *for* $i, j \in \overline{I}_{n,3}$, $i \neq j$.

    These rules take two cells that can be used to build a vector and create that vector. Notice that, once a cell is used in a vector, it will not be used in another.

- $Pri = \emptyset$.

- $i_{in} = i_{out} = 1$

This P module cancels pairs of critical cells of Π_{Vector} *due to the previously mentioned "step" configuration.*

The dP scheme

$$\Delta_{\text{Bin3DHom}} = (\Gamma_0, \Sigma, \Pi_{\text{Cub}}, \Pi_{\text{Vector}}, \Pi_{\text{Flow}}, \Pi_{\text{Vector}_2}, \mathcal{R}, i_{\text{in}}, i_{\text{out}})$$

with four modules defined below, computes the homology groups for the cubical complex generated from $\mathcal{I}$.

- $\Sigma = \{B_{\mathbf{i}} : \mathbf{i} \in \mathcal{I}^{-1}(1)\}$

- The modules are given by

  $\Pi_{\mathbf{Cub}}$ This module builds a cubical complex from the 3D binary image $\mathcal{I}$.

  $\Pi_{\mathbf{Vector}}$ This module creates a maximal acyclic vector field.

  $\Pi_{\mathbf{Flow}}$ This module determines the Morse complex associated to the vector field created above.

  $\Pi_{\mathbf{Vector_2}}$ This module creates a maximal acyclic vector field in the Morse complex.

- The distribution rules are given by
  - $\mathbf{R}_1 \equiv (\text{Cub}, \sigma_i\mathbf{C}_i \to \sigma_i\mathbf{C}_i, \text{Vector})$ for $i \in \overline{I}_{n,3}$
  - $\mathbf{R}_2 \equiv \left(\text{Cub}, \delta_{ij}^{\pm} \to \delta_{ij}^{\pm}, \text{Vector}\right)$ for $i, j \in \overline{I}_{n,3}$
  - $\mathbf{R}_3 \equiv \left(\text{Cub}, U_{ij,d}^{+} \to U_{ij,d}^{+}, \text{Vector}\right)$ for $i, j \in \overline{I}_{n,3}$ and $1 \leq d \leq 3$
  - $\mathbf{R}_4 \equiv (\text{Cub}, \text{d}_{ip} \to \text{d}_{ip}, \text{Vector})$ for $i \in \overline{I}_{n,3}$ and $0 \leq p \leq 3$
  - $\mathbf{R}_5 \equiv (\text{Vector}, \sigma_i \to \sigma_i, \text{Flow})$ for $i \in \overline{I}_{n,3}$

- $\mathbf{R}_6 \equiv (\text{Vector}, \mathbf{C}_i \to \mathbf{C}_i, \text{Flow})$ for $i \in \overline{I}_{n,3}$
- $\mathbf{R}_7 \equiv \left(\text{Vector}, \delta_{ij}^{\pm} \to \delta_{ij}^{\pm}, \text{Flow}\right)$ for $i, j \in \overline{I}_{n,3}, i \neq j$
- $\mathbf{R}_8 \equiv (\text{Vector}, \mathrm{d}_{ip} \to \mathrm{d}_{ip}, \text{Flow})$ for $i \in \overline{I}_{n,3}$ and $0 \leq p \leq 3$
- $\mathbf{R}_9 \equiv \left(\text{Vector}, V_{ij}^{\pm} \to V_{ij}^{\pm}, \text{Flow}\right)$ for $i, j \in \overline{I}_{n,3}, i \neq j$
- $\mathbf{R}_{10} \equiv \left(\text{Flow}, \tilde{\sigma}_i \tilde{\mathbf{C}}_i \to \sigma_i \mathbf{C}_i, \text{Vector}_2\right)$ for $i \in \overline{I}_{n,3}$
- $\mathbf{R}_{11} \equiv \left(\text{Flow}, \tilde{\delta}_{ij}^{\pm} \to \delta_{ij}^{\pm}, \text{Vector}_2\right)$ for $i, j \in \overline{I}_{n,3}, i \neq j$
- $\mathbf{R}_{12} \equiv (\{V\}|\text{Vector}_2, \sigma_i \to \sigma_i, \text{Flow})$ for $i \in \overline{I}_{n,3}$
- $\mathbf{R}_{13} \equiv (\{V\}|\text{Vector}_2, \mathbf{C}_i \to \mathbf{C}_i, \text{Flow})$ for $i \in \overline{I}_{n,3}$
- $\mathbf{R}_{14} \equiv \left(\{V\}|\text{Vector}_2, \delta_{ij}^{\pm} \to \delta_{ij}^{\pm}, \text{Flow}\right)$ for $i, j \in \overline{I}_{n,3}, i \neq j$
- $\mathbf{R}_{15} \equiv \left(\{V\}|\text{Vector}_2, V_{ij}^{\pm} \to V_{ij}^{\pm}, \text{Flow}\right)$ for $i, j \in \overline{I}_{n,3}, i \neq j$

- $i_{\text{in}} = \text{Cub}$, $i_{\text{out}} = \text{Vector}_2$

The modules Π_{Vector} and Π_{Vector_2} are the only modules that do not rely on the framework. In other words, the user of the framework only has to deal, most of the times, with writing modules for calculating acyclic vector fields, as the tasks for building the cubical complex associated to the binary image and the building of the Morse complex associated to it, are left to the framework.

Theorem 3.12.
The dP scheme $\Delta_{Bin3DHom}$ defined above calculates the homology groups for a given binary 3D image $\mathcal{I} : T_n^3 \to \{0, 1\}$ in $O(log_2 n)$ computation steps.

Remark. The result is proved by following the behavior of the dP scheme Δ_{Bin3DHom} for an input image $\mathcal{I}$. First of all, the input alphabet for Π_{Cub} is set up from $\mathcal{I}$ as Π_{Cub} is the input module. No distribution rules can be selected until one module reaches its halting stage. By theorem 3.6, Π_{Cub} halts in two steps. Then, the only distribution rules that can be selected are $\mathbf{R}_1$, $\mathbf{R}_2$, $\mathbf{R}_3$ and $\mathbf{R}_4$. After it application, the module Π_{Vector} starts calculating a vector field.

Rules R_1 in Π_{Vector} build a vector from a cell σ_i to a co-face σ_j. The priorities ensures that all the p-cells are mapped by V, the vector field, to a $p+1$-cell, prioritizing that the vector joining its barycenter has as lowest index as possible. Also, the rules ensure that the vector field is acyclic, as all the vectors are "parallel" to the vectors in the canonical base in $\mathbb{R}^3$ with positive direction. This work is done in only one computation step.

This way of building a vector field creates two kind of critical cells. First of all, there are critical cells due to homology generators. Second, there are critical cells due to "steps" in the cubical complex. In figure 3.1a, the cells $\langle P_{11}, P_{16}\rangle$ and $\langle P_{11}, P_{12}\rangle$ are in such position. It looks somewhat an "step" in a stair in a bottom view. This configuration of critical cell is due to the priority relation among rules in Π_{Vector}. Hence, we have two kinds of critical cells, those representing an homology generator and those cells that must have to be canceled in later computation steps.

Once Π_{Vector} halts, the only distribution rules that can be selected are $\mathbf{R}_5$ to $\mathbf{R}_9$, sending objects to Π_{Flow}. Concretely, those rules send cells (σ_i), boundary map $(\delta_{ij}^{\pm})$, critical cells $(\mathbf{C}_i)$ and vector field $V_{ij}^{\pm}$.

Module Π_{Flow} calculates the Morse complex associated to the previous vector field in, at most, $4+3\log_2(n-1)$. To prove this note that the longest V-path in a 3D cubical complex with, at most, $(n-1)^3$ 1-cells have length, at most, $(n-1)^3$ and, then, use theorem 3.9.

Once the Morse complex is built, it is sent to the module Π_{Vector_2} by distribution rules $\mathbf{R}_{10}$ and $\mathbf{R}_{11}$. This module builds an acyclic vector field that "removes" all the cells that do not represent an homology generator. This computation is completed in only one step. If some vector field is built, which means that there are cells that can be cancelled, at least one object V comes from the environment that ensures the selection and application of distribution rules $\mathbf{R}_{12}$ to $\mathbf{R}_{15}$, sending the Morse complex and the gradient vector field to the module Π_{Flow}, where another Morse complex is built and sent to Π_{Vector_2}.

Once the second Morse complex has came to Π_{Vector_2} the dP scheme $\Delta_{Bin3DHom}$ reaches the halting state, as no other distribution rule can be selected. This takes place as the only critical cells that are not homology generators are 1-cells that are in a "step" configuration, and those cells are removed by the second vector field. Formally, we have reduced the initial cubical complex to another complex with null differential, which means that all the cells are generators of the homology groups.

Recall that the sum of the computation steps required by each module is, in the worst case, at most $12 + 6log_2(n-1)$, which is $O(\log_2 n)$. □

The full calculation process of homology groups can be followed in Figure 3.1. The module Π_{Cub} builds the cubical complex in Figure 3.1a in two computation steps. Then, the module Π_{Vector} builds the acyclic vector field in Figure 3.1b in one computation step. Following, the module Π_{Flow} constructs the (first) Morse complex in 7 steps (because the longest V-path

in the complex has length 6). Figure 3.1c shows the two kinds of critical cells. On one hand, P_{19} is a critical 0-cell due to an homology generator. On the other hand, P_{12} and P_{16} are 0-cells due to "step" configuration. $\langle P_5, P_6 \rangle$ is a critical 1-cell due to "hole" while $\langle P_{11}, P_{12} \rangle$ and $\langle P_{15}, P_{16} \rangle$ are critical 1-cells originated by a "step" configuration. Next, the module Π_{Vector_2} specify, in one computation step, another acyclic vector field in the complex in Figure 3.1d which leads to the computation in 5 computation steps of the homological complex in Figure 3.1e, that has null differential, so that all the cells are homology generators.

3.4 Conclusions of the chapter

In this chapter we have presented a brief introduction to Membrane Computing and how it can be used to efficiently compute the homology generators of 3D digital objects. The presented algorithms are executed sequentially but each of them is massively parallel in the way that all the rules that can be applied in parallel.

However, for the best of our knowledge, there is no computing device capable of execute "membrane code" as if it was a living cell and some adaptation have to be performed in those algorithms.

On the other hand, Membrane Computing algorithms do not count with algebraic operations in its operations, but precisely Algebra is one of our most useful tool as we are working with Computational Algebraic Topology. Hence we must restrict our research in the development of Membrane computing algorithms in Computational Algebraic Topology to those situations where the algebra can be eluded. This is the case of homology groups of 3D digital objects, which are torsion free.

In the following chapters the ideas presented above are used to develop some massively parallel algorithms that provides an efficient tool to extract (co)homological information of digital objects.

CHAPTER 4

Parallel calculation of an AM-model for a nD digital object

In previous chapter we present a combinatorial approximation to calculation of homology groups of a 3D digital object. In this chapter we show how these ideas can be used to design an algorithm that allows us to calculate an AM-model for a digital object nD. In a later chapter we demonstrate how the AM-model will effectively provide us with great homological information of the digital object.

The core of this algorithm is also based on DMT. We see that by applying the DMT it is possible to construct a reduction of the cubical cell complex associated with a digital object to a much simpler CW complex.

4.1 Parallel calculation of GVF in cubical complexes

The higher organizational structure of a cubical complex can be exploded to develop a parallel algorithm to create an acyclic vector field on the cubical complex. This vector field is not, generally, optimal in the sense that there are (critical) cells that do not represent an homology class. The problem of finding an optimal GFV is known to be NP hard [25], so the development of simplification techniques is required. Forman used the so called critical cells cancellation, where a condition to delete two critical cells is presented.

Concretely, two critical cells can be removed if there is a unique V-path from the boundary of the higher dimension cell to the lower dimension cell. This technique is also known as arrow reversing. However, the computational cost of counting V-paths between two cells is, sometimes, too expensive.

We do prefer another equivalent way of facing the simplification of GVF. Concretely, building another GVF in the critical complex can be proved equivalent to arrow reversing, however building a GVF in a general cell complex can be performed using a spanning tree based technique.

Let K be a cubical complex. We suppose, without loose of generality, that there exists a non negative integer n, such that for every cell σ in K

$$0 \leq \inf I_p(\sigma) \leq \sup I_p(\sigma) < n \quad (1 \leq p \leq \operatorname{emb} K) \tag{4.1}$$

Let $\vec{e}_p$ denote the p-th vector in the orthonormal canonical base of $\mathbb{R}^{\operatorname{emb} K}$. We define the *right shift* of a cell $\sigma = I_1 \times I_2 \times \cdots \times I_{\operatorname{emb} K}$ by $\sigma \oplus \vec{e}_p = I'_1 \times I'_2 \times \cdots \times I'_{\operatorname{emb} K}$ where

$$I'_k = \begin{cases} (\inf I_p, 1 + \inf I_p) & \text{if } k = p \wedge \inf I_p = \sup I_p \\ I_k & \text{otherwise} \end{cases}$$

For example, $(1,2) \times (2,3) \times (2) \oplus \vec{e}_2 = (1,2) \times (2,3) \times (2)$ as the second interval is non degenerated, and $(1,2) \times (2,3) \times (2) \oplus \vec{e}_3 = (1,2) \times (2,3) \times (2,3)$. Roughly speaking, shifting a cell in a (principal) direction consists on extending the cell along this direction.

Recall that $\mathbb{Z}[A]$ is the free $\mathbb{Z}$-module with basis the elements of the set A and $V[\mu, \sigma]$ is the coefficient of the chain $V(\sigma)$ corresponding to cell μ. Note that, essentially, a morphism between two free modules can be represented as a matrix, hence the later notation gives direct access to the elements of that matrix.

Algorithm 1 show how to compute an acyclic vector field in parallel for a given cubical complex. It works as follows. In line 2 a null chain map of degree +1 is created. The "for" loop in line 3 sequentially iterates over the vectors belonging to the canonical orthonormal base for the embedding space of the cubical complex. The parallel "for" in line 4 iterates in parallel over all the cells in the cubical complex. In this situation, at most at theoretical level, the algorithm will treat every cell at the same time. In line 5 the right shift of every cell along the current vector in the canonical base is computed. The condition at line 6 ensures that the shift is well defined (i.e., it returns a facet inside the cubical complex). In lines 8 through 11,

Algorithm 1 Acyclic cubical vector field for a cubical complex.

1: **function** AcyclicCubicalVectorField(K)
2: $V \leftarrow (0 : \mathbb{Z}[K_*] \to \mathbb{Z}[K_{*+1}])$
3: **for** $p \leftarrow n, 1$ **do**
4: **for all** $\sigma \in K$ **parallel do**
5: $\tau \leftarrow \sigma \oplus \vec{e}_p$
6: **if** $\tau \neq \sigma \wedge \tau \in K$ **then**
7: $q \leftarrow \dim \sigma$
8: $dom_\sigma \leftarrow \sum_{\mu \in K_{q+1}} |V_q[\mu, \sigma]|$
9: $dom_\tau \leftarrow \sum_{\mu \in K_{q+2}} |V_{q+1}[\mu, \tau]|$
10: $img_\sigma \leftarrow \sum_{\mu \in K_q} |V_q[\sigma, \mu]|$
11: $img_\tau \leftarrow \sum_{\mu \in K_{q+1}} |V_{q+1}[\tau, \mu]|$
12: **if** $dom_\sigma + dom_\tau + img_\sigma + img_\tau = 0$ **then**
13: $V_q(\sigma) \leftarrow \langle \sigma, \mathrm{d}(\tau) \rangle \, \tau$
14: **end if**
15: **end if**
16: **end for**
17: **end for**
18: **return** V
19: **end function**

we calculate if σ or τ belongs to the domain or the image of the (previously calculated) vector field. The condition at line 12 ensures that the vector field is well defined, i.e. every cell belongs, at most, to only one vector in the vector field. Finally, line 13 creates the corresponding vector with the correct sign. Recall that the incidence of σ in $V(\sigma)$ is the same as the incidence of σ in τ.

Recall that all the vectors in the vector field follow one of the positive directions in the canonical orthonormal base in $\mathbb{R}^{\mathrm{emb}\, K}$. This ensures that the vector field is acyclic as we avoid the possibility of turning back and, hence, preventing the generation of cycles.

The vector field showed in figure 2.2 is computed using algorithm 1.

4.2 Parallel calculation of an AM-model for a cubical complex

Given a cubical complex K, we show in this section how to compute an AM-model for K using as much parallel operations as possible. Recall that

an AM-model is represented by a reduction

$$(f, g, h) : \mathcal{C}(K) = (\mathsf{C}_p, \mathrm{d}) \to \mathcal{M} = (\mathsf{M}_p, \mathrm{d}_M)$$

where the matrices of d_M are in SNF. We proceed following the steps below:

Stage 1 Create an acyclic vector field V within the cubical complex K. This stage finishes in $\operatorname{emb} K$ parallel steps. See algorithm 1. This stage is the only one requiring that K is a cubical complex. If other kind of cell complex is given, e.g. simplicial or CW, the construction of the acyclic vector field is the only step to be changed. The rest of the stages remain unchanged as they do not depend on the geometry of the cell complex but on its associated chain complex, which is an algebraic object.

Stage 2 Calculate the reduction $(f_V, g_V, h_V) : \mathcal{C}(K) \to \mathcal{M}_V(K)$ associated to the vector field V.

Stage 3 Create another acyclic vector field in the Morse complex of the previous vector field.

Stage 4 Repeat the two previous stages until the differential of the Morse complex associated to the simplified vector field has no invertible elements in its matrix.

Stage 5 If the Morse complex of the last reduced vector field has some non zero element in its matrix representation, calculate the SNF-reduction for it.

Stage 6 Compose all the reductions associated to the vector fields previously calculated. This composition represents, in fact, an AM-model for the cubical complex K.

From vector fields to reductions

In the seminal paper for DMT [11], Forman did not mention the term reduction[1] explicitly, however in the main theorem of equivalence of the homology of the original cell complex and the critical complex, a chain homotopy is used. From a simple combination of this chain homotopy with the stabilization map of the flow and the inclusion, the required reduction can be calculated. However, this method of calculation is too expensive when the complexes are large.

[1]The term reduction is also known as chain contraction or Eilenberg-Zilber data.

Fortunately, Romero et al calculated in [37] the corresponding reduction in a more efficient way. In the following paragraphs is shown how to compute a reduction $\rho_V : \mathcal{C}(K) \to \mathcal{M}_V(K)$ for a given cell complex K and acyclic vector field V.

The definition of vector field induces a partition in the set of cells. Let us represent the set of source p-cells as $\mathbf{S}_p$, the set of target p-cells as $\mathbf{T}_p$ and the set of critical p-cells as $\mathbf{C}_p$. As V is a (degree $+1$) homomorphism of modules, the previously defined partition can be interpreted algebraically expressing each chain module as a direct sum, concretely,

$$\mathsf{C}_p = \mathbf{T}_p \oplus \mathbf{S}_p \oplus \mathbf{C}_p \tag{4.2}$$

This decomposition induces a series of homomorphisms from the differential map. Let us define the following projections from the full sum to each operand and the inclusion of each operand into the sum. To emphasize the decomposition above, each element $a \in \mathsf{C}_p$ will be represented as a "vector" $\begin{bmatrix} t_a & s_a & c_a \end{bmatrix}^t$ where each coordinate is a unique chain in each operand.

$$\pi_{p,t} : \mathsf{C}_p \to \mathbf{T}_p \text{ where } \pi_{p,t}(\begin{bmatrix} t_a & s_a & c_a \end{bmatrix}^t) := t_a \tag{4.3}$$
$$\pi_{p,s} : \mathsf{C}_p \to \mathbf{S}_p \text{ where } \pi_{p,s}(\begin{bmatrix} t_a & s_a & c_a \end{bmatrix}^t) := t_s \tag{4.4}$$
$$\pi_{p,c} : \mathsf{C}_p \to \mathbf{C}_p \text{ where } \pi_{p,c}(\begin{bmatrix} t_a & s_a & c_a \end{bmatrix}^t) := t_c \tag{4.5}$$
$$\iota_{p,t} : \mathbf{T}_p \to \mathsf{C}_p \text{ where } \iota_{p,t}(c) := \begin{bmatrix} c & 0 & 0 \end{bmatrix}^t \tag{4.6}$$
$$\iota_{p,s} : \mathbf{S}_p \to \mathsf{C}_p \text{ where } \iota_{p,s}(c) := \begin{bmatrix} 0 & c & 0 \end{bmatrix}^t \tag{4.7}$$
$$\iota_{p,c} : \mathbf{C}_p \to \mathsf{C}_p \text{ where } \iota_{p,c}(c) := \begin{bmatrix} 0 & 0 & c \end{bmatrix}^t \tag{4.8}$$
$$\mathrm{d}_{p,ss} = \pi_{p-1,s} \circ \mathrm{d}_p \circ \iota_{p,s} \tag{4.9}$$
$$\mathrm{d}_{p,st} = \pi_{p-1,s} \circ \mathrm{d}_p \circ \iota_{p,t} \tag{4.10}$$
$$\mathrm{d}_{p,sc} = \pi_{p-1,s} \circ \mathrm{d}_p \circ \iota_{p,c} \tag{4.11}$$
$$\mathrm{d}_{p,ts} = \pi_{p-1,t} \circ \mathrm{d}_p \circ \iota_{p,s} \tag{4.12}$$
$$\mathrm{d}_{p,tt} = \pi_{p-1,t} \circ \mathrm{d}_p \circ \iota_{p,t} \tag{4.13}$$
$$\mathrm{d}_{p,tc} = \pi_{p-1,t} \circ \mathrm{d}_p \circ \iota_{p,c} \tag{4.14}$$
$$\mathrm{d}_{p,cs} = \pi_{p-1,c} \circ \mathrm{d}_p \circ \iota_{p,s} \tag{4.15}$$
$$\mathrm{d}_{p,ct} = \pi_{p-1,c} \circ \mathrm{d}_p \circ \iota_{p,t} \tag{4.16}$$
$$\mathrm{d}_{p,cc} = \pi_{p-1,c} \circ \mathrm{d}_p \circ \iota_{p,c} \tag{4.17}$$

Using previous decomposition, in [37] an explicit formula for the

reduction associated to V is constructed. More explicitly:

$$\mathrm{d}'_p = \mathrm{d}_{p,cc} - \mathrm{d}_{p,ct} \circ d^{-1}_{p,st} \circ \mathrm{d}_{p,sc} \tag{4.18}$$

$$f_p = \begin{bmatrix} 0 & -\mathrm{d}_{p+1,ct} \circ d^{-1}_{p+1,st} & 1 \end{bmatrix} \tag{4.19}$$

$$g_p = \begin{bmatrix} -\mathrm{d}^{-1}_{p,st} \circ \mathrm{d}_{p,sc} \\ 0 \\ 1 \end{bmatrix} \tag{4.20}$$

$$h_p = \begin{bmatrix} 0 & \mathrm{d}^{-1}_{p+1,st} & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{4.21}$$

The existence of $\mathrm{d}^{-1}_{p,st}$ is granted by the aciclicity of V (see [37] for details). Is straightforward to check that

$$(f_p, g_p, h_p) : (\mathsf{C}_p, \mathrm{d}_p) \Rightarrow\!\!\!\Rightarrow (\mathbf{C}_p, \mathrm{d}'_p)$$

from the chain complex $\mathcal{C}$ to the critical complex.

Simplification of vector fields

Now, the simplification of each vector acyclic field is performed by creating a new vector field in the reduced complex.

To simplify a vector field we proceed by creating a new vector field in the reduced complex. This complex, call it M for further reference,has a combinatorial structure which is no longer cubic or even simplicial, but can only be expected to be a CW complex.

The generation of an acyclic vector field in M is based on the creation of a directed acyclic graph G whose nodes are cells satisfying the following properties:

1. If $\sigma^{(p)}\tau^{(p+1)}$ is an edge of G then there is no other edge $\sigma^{(p)}\eta^{(p+1)}$ with $\eta^{(p+1)} \in M \setminus \{\tau\}$.

2. If $\sigma^{(p)}\tau^{(p+1)}$ is an edge of G then there is no other edge $\nu^{(p)}\tau^{(p+1)}$ with $\nu^{(p)} \in M \setminus \{\sigma\}$

3. If $\sigma^{(p)}\tau^{(p+1)}$ is an edge of G, there is an edge $\tau^{(p+1)}\mu^{(p)}$ for each cell $\mu \in \partial(\sigma) \setminus \{\sigma\}$. With this property we deal with the vector field flow concept. This step is fundamental because in this extension of the vector field is where it should be ensured that there are no cyclic V-paths.

The algorithm 2 details the construction of an acyclic vector field in the reduced complex.

Algorithm 2 Acyclic vector field for a generic CW complex.

```
1: function AcyclicCWVectorField(K)
2:     V ← (0 : R[K_•] → R[K_{•+1}])
3:     G ← DirectedGraph()
4:     for σ ∈ K do
5:         G.add_node(σ)
6:         if σ is critical then
7:             for τ ∈ δ(σ) do
8:                 if V(τ) = 0 ∧ τ ∉ Im(V) ∧ ⟨σ, d(τ)⟩ ∈ R_* then
9:                     G.add_node(τ)
10:                    G.add_edge(σ, τ)
11:                    for μ ∈ ∂(τ) \ {σ} do
12:                        if not G.has_node (μ) then
13:                            G.add_node(μ)
14:                        end if
15:                        G.add_edge(τ, μ)
16:                    end for
17:                    if G.has_cycles() then
18:                        G.remove_node(τ)
19:                    else
20:                        V(σ) ← ⟨σ, d(τ)⟩τ
21:                        break
22:                    end if
23:                end if
24:            end for
25:        end if
26:    end for
27:    return V
28: end function
```

Recall that a cell σ is critical if it is not either the source or the target of any vector in V. In other words,a critical cell σ verifies that $V(\sigma) = 0$ and σ is not in the support of any chain $c \in \mathrm{Im}(V)$. This can be easily checked in the matrices of V. Let us denote by $[V_p]$ th matrix of $V_p : \mathsf{C}_p \to \mathsf{C}_{p+1}$ in the standard basis of chains. Then $\sigma^{(p)}$ is critical if both the p-th column in $[V_p]$ and the p-th row in $[V_{p-1}]$ are zero.

Theorem 4.1.
The function V returned by algorithm 2 is an acyclic vector field.

Proof. To prove that the map V returned by algorithm 2 is an acyclic vector field it is mandatory to prove

(a) $\#\operatorname{supp}(V(\sigma)) \leq 1$ where (c) denotes the support of a chain c, defined by $\operatorname{supp}\left(\sum_{i\in I} c_i\sigma_i\right) = \{\sigma_i \,|i \in I\,\}$;

(b) $V^2 = 0$ and

(c) there is no closed non-trivial V-paths.

(a) This statement is equivalent to prove that any $(p+1)$-cell τ is the target of, at most, one vector in V. This fact is guaranteed by line 8 in algorithm 2.

(b) To prove that $V^2 = 0$ is enough to show that for any cell σ. First of all, suppose that $V(\sigma) = \langle \sigma, \mathrm{d}(\tau)\rangle\tau$. Let us prove that $V(\tau) = 0$. In fact, line 6 in algorithm 2 only consider assigns a value to $V(\tau)$ if $\tau \notin \operatorname{Im}(V)$. Hence, if $\tau \in \operatorname{supp}(\operatorname{Im}(V))$ then $V(\tau) = 0$, so $V^2(\sigma) = 0$.

(c) The acyclicity of V is a direct consequence of the acyclicity of the graph G in algorithm 2.

□

The SNF reduction

Pilarczyk implemented in [31] a SNF algorithm such that the change of bases matrices of two consecutive dimension are mutually inverses, more concretely:

$$[\mathrm{d}_p] = A_{p-1} \cdot D_p \cdot G_p \tag{4.22}$$

where $[\mathrm{d}_p]$ is the matrix of the differential map at dimension p, D_p is in SNF and $A_p = G_p^{-1}$. This algorithm allows us to construct a reduction of a chain complex to another in which the differential matrices are in SNF, where the projection is given by matrices A_p and inclusion is given by matrices G_p.

Finally, the composition of all the reductions computed above gives a reduction from the original cubical complex to a reduced complex in Smith Normal Form.

4.3 Conclusions

In this chapter we have seen a theoretical development to calculate an AM-model for a cell complex. This model is based on iterative calculation of Morse reductions until the differential of the final complex does not have invertible inputs in the ground ring. A Morse reduction is given univocally by an acyclic vector field. It is in this step where massive parallelism is applied. Once the cubical cell complex is drastically reduced, the resulting chain complex is reduced again in order to remove all the pairs $\sigma^{(p)}, \tau^{(p+1)}$ such that $\langle \sigma, \mathrm{d}(\tau) \rangle$ is invertible in the ground ring. This cancellations are selected sequentially and performed in parallel. Recall that matricial operations involved in composition of reductions are implemented using the massive parallelism provided by current GPU.

In chapter 6 some implementation details are to be discussed.

CHAPTER 5

Advanced (co)homological information extraction from digital objects

5.1 Introduction

An AM-model not only provides information on the (co)homology groups of a cell complex, but also supplies a relationship between the corresponding (co)chain complexes. This relation, given by the projection and inclusion chain maps, allows an analysis of several characteristics in (co)homology. For example, the inclusion of a reduced complex generator is always a (co)cycle.

In addition, the homotopy operator of an AM-model can be interpreted as a "geometric" homotopy that allows to reduce the original complex into the reduced complex.

In this chapter we will how to obtain a large amount of (co)homology information from an AM-model. We also show how to compute cohomology operations (cup and cap products) in the reduced complex so we can get more information in order to distinguish two cell complexes.

From here to the end of the chapter, we assume that K is a cell complex, $\mathcal{C} = \{\mathsf{C}_p, \mathrm{d}\}$ is its associated chain complex, and

$$(f, g, h) : \mathcal{C} \Rightarrow\!\!\!\!\Rightarrow \mathcal{M} = \{\mathsf{M}_p, \mathrm{d}_M\}$$

is an AM-model. For simplifying the presentation, we will assume that the ground ring is $\mathbb{Z}$.

5.2 (Co)homology information from AM-model

The first information of relevance at homology level is, of course, the homology and cohomology groups. The AM-model directly provides the (co)homology groups using the two following results.

Proposition 5.1.
A generator σ in M_p is a free homology generator if and only if $\mathrm{d}_M(\sigma) = 0$ and $\mathrm{d}^*_M(\sigma) = 0$. Moreover, if $\mathrm{d}_M(\sigma) = 0$ and $\mathrm{d}^*_M(\sigma) = k\tau$ then σ is a homology generator of the torsion subgroup with coefficient k.

From now on and depending on the context, a chain σ can also means the corresponding cochain whose value is 1 only in σ.

Proof. First of all, if $\mathrm{d}_M(\sigma) = 0$ it is clear that $\sigma \in \ker(\mathrm{d}_{Mp})$ for some p. If $\mathrm{d}^*_M(\sigma) = 0$ then σ is not in $\mathrm{Im}(\mathrm{d}_{Mp+1})$. So σ is an homology generator of the free part of the p-th homology group.

On the other hand, if $\mathrm{d}_M(\sigma) = 0$ and $\mathrm{d}^*_M(\sigma) = k\tau$, then σ contributes with a factor $\mathbb{Z}$ to $\ker(\mathrm{d}_{Mp})$. Also, as $\mathrm{d}^*_M(\sigma) = k\tau$, there is a factor $k\mathbb{Z}$ in $\mathrm{Im}(\mathrm{d}_{Mp+1})$. Therefore, there is a factor $\mathbb{Z}/k\mathbb{Z} = \mathbb{Z}_k$ in the quotient that defines homology. □

By dualization of the proposition below, we can state the following result concerning cohomology groups.

Proposition 5.2.
A generator σ in M_p is a free cohomology generator if and only if $\mathrm{d}^*_M(\sigma) = 0$ and $\mathrm{d}_M(\sigma) = 0$. Moreover, if $\mathrm{d}^*_M(\sigma) = 0$ and $\mathrm{d}_M(\sigma) = k\tau$ then σ is a cohomology generator of the torsion subgroup with coefficient k.

Let us bear in mind that $\mathsf{M}_p \cong \mathsf{M}^*_p$ as any chain can be uniquely viewed as a cochain taking the value 1 on itself.

By direct combination of previous results, we can state the following theorem.

Theorem 5.3.
Let $(f, g, h) : (\mathsf{C}_p, \mathrm{d}) \Rightarrow\!\!\!\!\Rightarrow (\mathsf{M}_p, \mathrm{d}_M)$ be an AM-model. Then:

$$(a)\ \mathsf{H}_p = \left(\bigoplus_{\sigma\in\mathsf{M}_p\wedge \mathrm{d}_M(\sigma)=0\wedge \mathrm{d}^*_M(\sigma)=0} \mathbb{Z} \right) \oplus \left(\bigoplus_{\sigma\in\mathsf{M}_p\wedge \mathrm{d}_M(\sigma)=0\wedge \mathrm{d}^*_M(\sigma)=k\tau} \mathbb{Z}_k \right)$$

$$(b)\ \mathsf{H}^p = \left(\bigoplus_{\sigma\in\mathsf{M}_p\wedge \mathrm{d}^*_M(\sigma)=0\wedge \mathrm{d}_M(\sigma)=0} \mathbb{Z} \right) \oplus \left(\bigoplus_{\sigma\in\mathsf{M}_p\wedge \mathrm{d}^*_M(\sigma)=0\wedge \mathrm{d}_M(\sigma)=k\tau} \mathbb{Z}_k \right)$$

Note that the second formula (b) is another way of expressing a consequence of the Universal Coefficient Theorem on Cohomology in the finite dimensional case. In other words, $\mathsf{H}^p \cong \mathbb{Z}^{\beta_p} \oplus \mathsf{Tor}_{p-1}$ where β_p is the rank of H_p, the p-th Betti number, and Tor_{p-1} is the $(p-1)$-th torsion subgroup.

Another homological property is the (co)boundary relationship. Let us consider two generators $\sigma \in \mathsf{M}_p$ and $\tau \in \mathsf{M}_{p+1}$. We say that σ is a face of τ in M if there is a face μ of τ in K such that $f(\mu) = \sigma$. We can see here how the boundary relationship in K is propagated to M. Forman proved in [11, Corollary 3.5] that, given a Morse function[1] f, K is homotopy equivalent to a CW complex with exactly one cell per critical cell. This CW complex is, concretely, the complex M. The boundary relationship defined above is another step in the determination of the minimal CW structure for a given cell complex.

The invariant factor decomposition for finitely generated Abelian groups states that any finitely generated Abelian group G is isomorphic to the direct sum $\mathbf{Z}^n \oplus \mathbf{Z}_{k_1} \oplus \cdots \oplus \mathbf{Z}_{k_r}$ where $k_i \mid k_{i+1}$. This result, combined with the last theorem, allows us to proof the following theorem.

Theorem 5.4.
The chain complex $\mathcal{M}$ is minimal in the number of cells.

Proof. The construction algorithm for an AM-model (see section 4.2) ensures that the matrices of d_M are in SNF and has no ± 1 entries. Each non invertible coefficient represents a torsion coefficient. Of course, the matrix representation of the invariant factor decomposition is diagonal $\mathrm{diag}(1, \ldots, 1, k_1, \ldots, k_r, 0, \ldots, 0)$ where the 1 is repeated n times. Now, as the differentials d_M has no invertible entry, the matrices of d_M has the form $\mathrm{diag}(k_1, \ldots, k_r, 0, \ldots, 0)$. And those k_i are precisely the torsion coefficients calculated as in Proposition 5.1. Hence, the chain complex $\mathcal{M}$ is built bearing in mind that any two cells with incidence invertible in the ground

[1] It is equivalent to give a Morse function or to give an acyclic vector field (see [11]).

ring are cancelled. So the remaining cells are those essential for homology. The using of SNF ensures that the differentials are either zero or a non invertible multiple of a cell. □

5.3 Cohomology operations from AM-model

Cup product (see Section 2.2) defines a ring structure in the graded cohomology group. This operation allows to topologically discriminate, for example, a torus from a wedge of an sphere with two rings. In this case, (co)homology groups cannot distinguish these two spaces. Cup product is easily defined in simplicial homology.In cubical homology it is required a cubical approximation of the diagonal map. We mainly follows the approach in [31].

The definition of cup product on the cohomology ring $\mathsf{H}^*(K)$ of a digital object[2] K is easy starting from an AM-model (f, g, h) and a cubical approximation of the diagonal map $\mathbf{diag} : \mathcal{C}(K) \to \mathcal{C}(K) \oplus \mathcal{C}(K)$ (see equation 2.16). Concretely, given two cochains $\varphi^{(p)}$ and $\psi^{(q)}$,

$$(\varphi^{(p)} \smile \psi^{(q)})(\sigma^{(p+q)}) = (\mu \circ (\varphi \otimes \psi) \circ \Delta)(\sigma) \tag{5.1}$$

where Δ is the approximation of the diagonal map on the reduced complex $\mathcal{M}$ given by

$$\Delta = (f \otimes f) \circ \mathbf{diag} \circ g \tag{5.2}$$

Cap product is another operation involving cohomology classes. Its motivation and definition can be found in, for example, [16, Chapter 3]. It can be thought as a map from $\mathcal{C}(K) \otimes \mathcal{C}^*$ to $\mathcal{C}(K)$ such that assigns a $(p-q)$-chain to the tensor product of a p-chain and a q-cochain. The following equation express this relation in terms of the Künneth formula

$$\frown : \mathcal{C} \otimes \mathcal{C}^* \xrightarrow{\Delta \otimes \mathrm{id}} \mathcal{C} \otimes \mathcal{C} \otimes \mathcal{C}^* \xrightarrow{\mathrm{id} \otimes \epsilon} \mathcal{C} \tag{5.3}$$

where $\epsilon : \mathcal{C} \otimes \mathcal{C}^* \to \mathcal{R}$ is the evaluation map defined below

$$\epsilon\left(\sigma^{(p)} \otimes \varphi^{(q)}\right) = \begin{cases} \varphi(\sigma) & \text{if } p = q \\ 0 & \text{otherwise} \end{cases} \tag{5.4}$$

and the first id is the identity map on cochains while the second one is the identity map on chains.

[2]Note that digital objects and cubical complexes can be exchanged without problems.

If we define the projections onto the factors as $\pi_i : \mathcal{C}_1 \otimes \mathcal{C}_2 \to \mathcal{C}_i$ for $i = 1, 2$ with $\pi_i(\sigma_1 \otimes \sigma_2) = \sigma_i$, then the formula for cap product is given by

$$\sigma \frown \varphi = \varphi\left(\pi_2\left(\Delta(\sigma)\right)\right) \pi_1\left(\Delta(\sigma)\right) \tag{5.5}$$

Note how the equation above can be easily computed in the reduced complex as the aproximation of the diagonal map has been previously calculated.

5.4 Conclusions to the chapter

In this chapter, we show the versatility of an AM-model representation in order to get (co)homological information of digital objects. This information extraction begins with the construction of the cubical complex associated to the digital object. This construction is computed in parallel using a quasi literal implementation of the P system Π_{Cub} in definition 3.5. This adaptation basically check all the available cubical cells within the cubes that cover all the points in the digital object and the cubical cells in the complex are those such that all their vertices are in the object.

Once the cubical complex associated to the digital object has been calculated, we need to create an AM-model and this is easily performed following the steps in section 4.2. Recall that the construction of an acyclic vector field in the cubical complex is performed in parallel. The design patterns used in the parallelization of the construction of the vector field in the cubical complex guarantee that the vector field is acyclic, however some critical cells that, in fact, do not represent any homology class of the object (called *false critical cells*), remain. This spurious cells are removed in the next steps.

The spanning tree strategy used in the construction of an acyclic vector field can be parallelized using the framework in [40]. We have not implemented this parallelization in the software presented in chapter6 and it is delayed to a future work.

The iterative application of the algorithm 2 results in a cell complex, namely a CW complex, whose differential map is either zero or has all the coefficients in the units set of the ground ring. This complex is transformed in another with the same cells but with an equivalent differential in Smith Normal Form. This last step allows to easily calculate all the (co)homology groups of the digital object as the are the same of the last reduced complex. Note that the condition on the differential ensures the minimality of the

AM-model. As for each unit coefficient in the differential a reduction can be performed so a pair of cells would disappear.

The AM-model brings in two chain maps, projection and inclusion, that allow us to map cells from the reduced complex to (co)cycles in the original complex and cells in the original complex to homology generators in the reduced complex.

The approximation of the diagonal given in 5.2 is crucial in the computation of cup and cap products. Both represents operations in (co)homology that leads to homotopy invariants that provides more interesting information of a digital object and relates holes ((co)homology generator) at different dimensions. For instance, the cup product of the two 1-dimensional holes of the torus is precisely its 2-dimensional homology generator.

CHAPTER 6

Implementation and experimentation

6.1 Implementation

Associated to the theoretical study of this dissertation, a prototype of the proposed algorithms is implemented. The prototype is developed in Python ([39], `https://www.python.org/`). This language is used because it is multi-paradigm, since it allows imperative programming, object oriented or functional programming; it also offers a large battery of modules that simplify the implementation of many of the tasks that we need to implement.

The developed code is basically supported in two large modules: `numpy` (`www.numpy.org`, [41]) and `numba` (`numba.pydata.org`). The first provides the necessary machinery for matricial calculation and serves as support to implement massive parallelism, which is carried out through the interface that `numba` implements for CUDA.

CUDA ([29]) offers the computing power of massive parallelism present in GPUs for use by any software, regardless of whether its purpose is graphic or not. This is what is known as GPGPU.

The paradigm or programming model that is used is Object Oriented Programming. The following is a brief description of the implemented objects, indicating in which source file they are and which dependencies they have.

- `Reduction (reduction.py)` This class represents a reduction between two chain complexes. Requires the use of `numpy`, the class `Chain (chain.py)`, the class `ChainMap (chain_map.py)`, and the function `AW (alexander_whitney.py)`. This class is not only a container of the three functions that define a reduction, i.e. inclusion, projection and homotopy operator, but also implements other additional methods of special utility:

  - `vector_field()`: this method returns a vector field whose associated reduction is the instance of the `Reduction` class.
  - `diagonal()`: this method calculates the cellular approximation of the diagonal associated with the reduced complex.
  - `cup_product()`: This method returns a chain morphism

$$\begin{array}{rccc} \smile: & \mathcal{M}^* \otimes \mathcal{M}^* & \to & \mathcal{M}^* \\ & \varphi \otimes \psi & \mapsto & \varphi \smile \psi \end{array}$$

    Which represents the cup product of two cochains.

  - `cap_product()`: This method returns a chain morphism

$$\begin{array}{rccc} \frown: & \mathcal{M} \otimes \mathcal{M}^* & \to & \mathcal{M} \\ & \tau \otimes \varphi & \mapsto & \tau \frown \varphi \end{array}$$

    Which represents the cup product of two cochains.

  In addition, through operator overloading, the composition of reductions is implemented, which is a key operation in the development of the calculation algorithm of an AM-model.

- `ChainMap (chain_map.py)` This class implements a graded morphism of modules. An example of graded morphism of modules are chain maps. This class, instead, do not satisfy the relationship with differential that chain maps obey. A `ChainMap` instance can be regarded as a sequence of linear maps between modules. `ChainMap` instances can be added, substracted, multiplied by an integer or composed. The class for vector fields inherits from this class.

- `Chain (cells.py)` This class implements a chain, which is a dictionary[1] whose keys are cells and whose values are integers. This class implements all the common operations on chains.

[1] A dicionary in python is a data structure that associate values with keys. It is also known as associative array.

- `AbstractCell, Simplex, CubicalCell (cells.py)` These classes implement several types of cells. The first one is used as base class and is not instantiated. The others represent simplex and cubical cell, respectively.

- `TensorCell (cells.py)` A tensor cell is the cell resulting from the tensor product of two cells. This class and the previous two classes mentioned above are, from the programmer point of view, indistinguishable as all of them inherits from `AbstractCell` and all of them behaves as expected. Concretely, all classes that represent a cell implement the same interface.

- `VectorField (vector_field.py)` This class implements a vector field. It inherits from `ChainMap` as a vector field is a degree +1 graded morphism of modules. However, the class `VectorField` implements other interesting methods that are described below:

  - `is_critical(), is_source(), is_target()` These methods return whether a cell is critical, source or target.
  - `decomposition()` This method returns a triple of three exact chain complexes generated, respectively, by target cells, source cells and critical cells. It is used in the calculation of the reduction associated to the vector field.
  - `_projections_inclusions()` This private method[2] returns the projections from the source chain complex to any of the decomposition complexes and the inclusions from the decomposition complexes to the source complex. It is used in the calculation of the reduction associated to the vector field.
  - `reduction()` This method calculate de reduction associated to the vector field.
  - `am_model()` This is one of the main goals in the development. This method compute the AM-model from an acyclic vector field using the stages in section 4.2.

  Objects instantiating the class `VectorField` are not created by hand, but they are created using two functions. The first of them is `create_vector_field()` in the source file `vector_field.py`. This function implements the algorithm 2. The second function is

[2]Python does not include private methods as C++. This is a naming convention commonly adopted that informs to other developers that this method are not thought to be used outside of the class.

`create_cubical_vector_field()` in the source file `cubical_homology.py`. This function calculates an acyclic vector field in a cubical complex using massive parallelism by the CUDA interface in `numba`, implementing algorithm 1.

- `ChainComplex (chain_complex.py)` This class implements a chain complex over the ring of integers. Its more important methods and properties are listed below.
  - `d` This property is the differential of the chain complex. It is an instance of `ChainMap` of degree -1.
  - `D, A, G` These properties[3] represents, for each integer p, matrices D_p, A_p and G_p such that D_p is the SNF of the differential d_p and A_p and G_p are invertible matrices in $\mathbb{Z}$ such that $D_p = A_{p-1} \cdot [d_p] \cdot G_p$ and $A_p = G_p^{-1}$. These properties are used to compute a reduction from a chain complex to another chain complex whose differentials are in SNF. This reduction is not really a reduction as the basis of the chain complex are of the same size, however they are in a more simpler form.
  - `computeSNF()`: This method calculates de SNF of the differentials using the algorithm in [31]
  - `is_minimal()`: This method evaluates whether a chain complex is in minimal. Recall that a chain complex is minimal if there is no invertible elements of the ground ring (± 1 in $\mathbb{Z}$) in the matrices of the differentials. A minimal chain complex is the result of iteratively apply vector field reductions.
  - `id()`: This method returns the identity map on the source complex.
  - `SNF_reduction()`: This method returns the reduction from the the source complex to other isomorphic complex whose differentials are in SNF.
  - `__matmul __`: Operators in Python are over-loadable using special methods. Concretely, `__matmul __` overloads the operator `@` which is initially used to represent ordinary product of matrices, to distinguish it from `*` which is the element wise product. This operator is used in our prototype to represent the tensor product of chains, cells or chain complexes. However the tensor product

[3] More concretely, they must be called item properties as they behaves as lists indexed by integers.

of chain complexes does not actually implement the differential, as it is currently a huge time consumer and it is not required by any other element of the implemented software.

- `CellComplex (cell_complexes.py)`: This class represents a cell complex which is mainly thought as a dictionary whose cells are the dimensions of the cells and whose values are the cells at that dimension. It is used in development stage to represent simplicial complexes or the result of the reduction of any other cell complex. Its most important method is `chain_complex()` which is used to calculate the chain complex associated to the cell complex. The most time consuming stage of this method is the creation of the differential map.

- `CubicalComplex (cell_complexes.py)`: This class inherits from `CellComplex` and represents a cubical complex. It has two main differences from its ancestor: first of all is the data structure used to represent a cubical complex. It is a n-dimensional `numpy` array, which can be viewed as an n-dimensional matrix. The element at position $(i_1, \ldots, i_n)$ represents the presence (if the value is not zero) of a cubical cell $I_1 \times \cdots \times I_n$ where

$$I_p = \begin{cases} \left(\frac{i_p}{2}\right) & \text{if} \quad i_p \equiv 0 \mod 2 \\ \left(\frac{i_p-1}{2}, \frac{i_p+1}{2}\right) & \text{if} \quad i_p \equiv 1 \mod 2 \end{cases}$$

  This structure is ideal to represent a cubical complex in a compact way at the time it is in the appropriate form to be used in CUDA. The most important method in this class are the following:

  - `chain_complex()`: This method calculates the chain complex associated to the cubical complex, as expected. However, the calculation of the differential operator is performed using CUDA's massive parallelism. It can be thought as calculating te differential of all the cubical cells in te complex at the same time.

  - `from_file(), to_file()`: These methods are used to read cubical complexes and write cubical complexes to files. The file format is the one used in the software proposed in [31].

  - `random()`: This class method[4] is designed to create random cubical complexes.
  - `__mul__`: This method overloads the `*` product operator in Python. It is used to represent the Cartesian product of two cubical complexes.

- `HomologyGroups, CohomologyGroups (am_homology.py)`: These classes represents graded groups of homology and cohomology, respectively. Basically they are dictionaries with integer keys representing dimensions and a list of pairs of generator and torsion coefficient. The absence of torsion represents a generator of the free part of (co)homology. The main method is the class method `from_am_model`, which calculates the (co)homology groups from an AM-model.

6.2 Experimentation

Simplicial Klein bottle

Consider the triangulation of the Klein bottle in figure 6.1. In this figure blue big dots represents 0-simplices, black lines represents 1-simplices and triangles represents 2-simplices.

The full process of calculating homology information on a triangulation of the Klein bottle involves building acyclic vector fields iteratively and, finally, calculating the SNF of some matrices. In figure 6.1 the triangulation is shown along with the barycenters of each cell. Figure 6.2 shows the acyclic vector field V^1 generated by algorithm 2, where the barycenters of critical cells are drawn in green. The reduction ρ_1 associated to V^1, the first vector field, has as reduced complex the following

$$\mathcal{M}^1 : 0 \xleftarrow{\mathrm{d}_0^1} \mathsf{M}_0^1 \xleftarrow{\mathrm{d}_1^1} \mathsf{M}_1^1 \xleftarrow{\mathrm{d}_2^1} \mathsf{M}_2^1 \xleftarrow{\mathrm{d}_3^1} 0 \tag{6.1}$$

where

$$\mathsf{M}_0^1 = \mathbb{Z}[\langle 9 \rangle] \tag{6.2}$$

$$\mathsf{M}_1^1 = \mathbb{Z}[\langle 0, 8 \rangle] \oplus \mathbb{Z}[\langle 3, 5 \rangle] \oplus \mathbb{Z}[\langle 5, 9 \rangle] \oplus \mathbb{Z}[\langle 6, 9 \rangle] \oplus \mathbb{Z}[\langle 6, 9 \rangle] \tag{6.3}$$

$$\mathsf{M}_2^1 = \mathbb{Z}[\langle 0, 1, 5 \rangle] \oplus \mathbb{Z}[\langle 2, 5, 6 \rangle] \oplus \mathbb{Z}[\langle 0, 4, 7 \rangle] \oplus \mathbb{Z}[\langle 1, 7, 8 \rangle] \tag{6.4}$$

[4] A class method is a special kind of method in Python classes. They are designed to be called from the class itself instead of from a class instance. They can be thought as specialized constructors.

and the differential d^1 is given by

$$\mathrm{d}_0^1 = 0 \tag{6.5}$$
$$\mathrm{d}_1^1 = 0 \tag{6.6}$$
$$\mathrm{d}_2^1(\langle 0,1,5\rangle) = -\langle 0,8\rangle - \langle 3,5\rangle \tag{6.7}$$
$$\mathrm{d}_2^1(\langle 2,5,6\rangle) = \langle 5,9\rangle - \langle 6,9\rangle + \langle 0,8\rangle \tag{6.8}$$
$$\mathrm{d}_2^2(\langle 0,4,7\rangle) = \langle 5,9\rangle + \langle 6,9\rangle - \langle 7,9\rangle + \langle 3,5\rangle \tag{6.9}$$
$$\mathrm{d}_2^1(\langle 1,7,8\rangle) = \langle 7,9\rangle \tag{6.10}$$

As there are invertible coefficients in the matrices of d^1, another acyclic vector field V^2 can be calculated on $\mathcal{M}^1$. The reduced complex is given in the following equations.

$$\mathsf{M}_0^2 = \mathbb{Z}[\langle 9\rangle]; d_0^2 = 0 \tag{6.11}$$
$$\mathsf{M}_1^2 = \mathbb{Z}[\langle 6,9\rangle] \oplus \mathbb{Z}[\langle 0,8\rangle]; d_1^2 = 0 \tag{6.12}$$
$$\mathsf{M}_2^2 = \mathbb{Z}[\langle 2,5,6\rangle]; \mathrm{d}_2^2(\langle 2,5,6\rangle) = -2\langle 6,9\rangle + 2\langle 0,8\rangle \tag{6.13}$$

Here, we can see that this reduced complex cannot be reduced using a vector field, so let us compute the SNF reduction.This gives as result the final reduced complex $\mathcal{M}$ shown below

$$\mathsf{M}_0 = \mathbb{Z}[\langle 9\rangle]; \mathrm{d}_{M0} = 0 \tag{6.14}$$
$$\mathsf{M}_1 = \mathbb{Z}[\langle 6,9\rangle] \oplus \mathbb{Z}[\langle 0,8\rangle]; \mathrm{d}_{M1} = 0 \tag{6.15}$$
$$\mathsf{M}_2 = \mathbb{Z}[\langle 2,5,6\rangle]; \mathrm{d}_{M2}(\langle 2,5,6\rangle) = 2\langle 6,9\rangle \tag{6.16}$$

This final complex is modified in order to have all the coefficients non negative. The composition of all the reductions calculated above results in the reduction $\rho_{AM} = (f,g,h) : \mathcal{C} \Rightarrow\!\!\!\!\!\rightarrow \mathcal{M}$ defining the AM-model for the Klein bottle. In figure 6.3 the vector field generated by ρ_{AM} is shown. Please note that the reduction generated by this vector field is not ρ_{AM} as this reduction has been calculated using the SNF reduction.

From ρ_{AM} several homological information can be extracted. First, the homology groups of the Klein bottle are given by

$$\mathsf{H}_0(K) = \mathbb{Z}[\langle 9\rangle] \tag{6.17}$$
$$\mathsf{H}_1(K) = \mathbb{Z}[\langle 0,8\rangle] \oplus \mathbb{Z}_2[\langle 6,9\rangle] \tag{6.18}$$

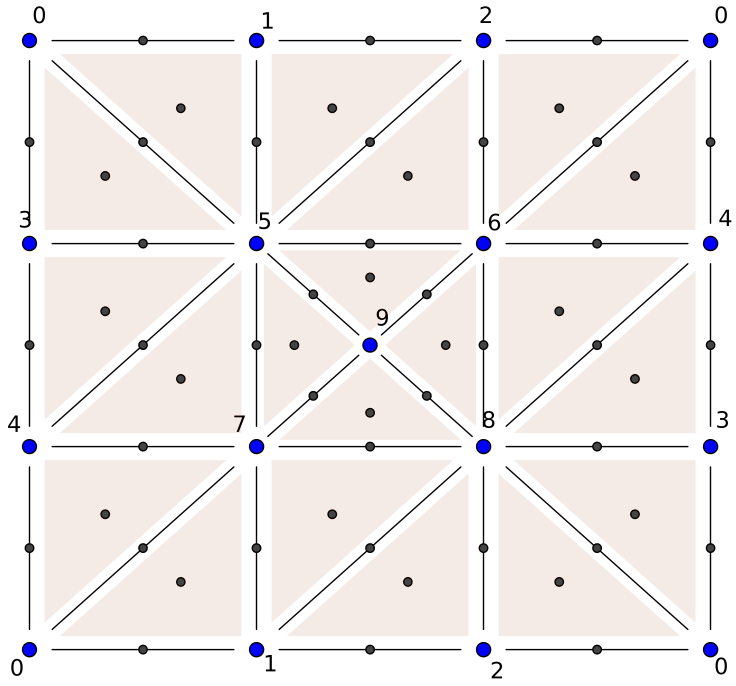


Figure 6.1: Simplicial structure on the Klein bottle. The barycenters of each cell are also drawn.

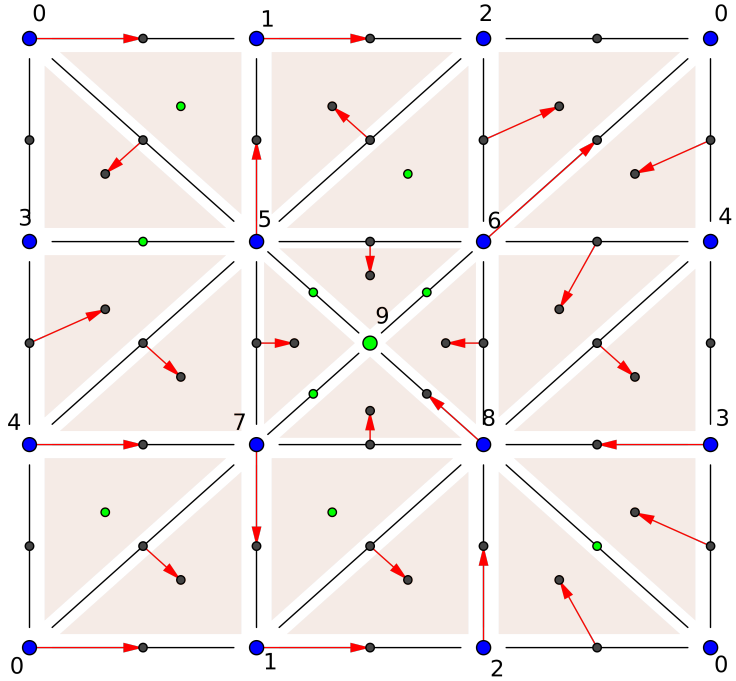


Figure 6.2: Acyclic vector field on the Klein bottle. Critical cells have its barycenters drawn in green.

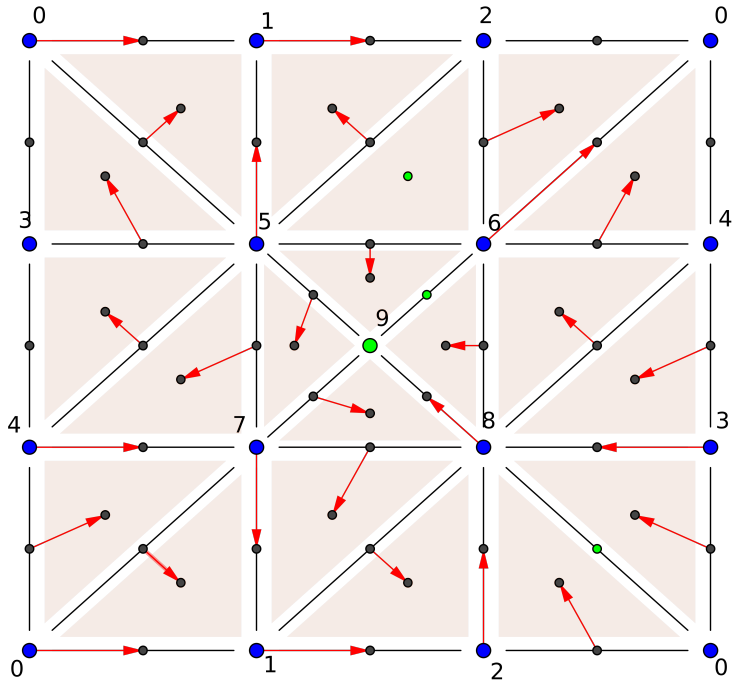


Figure 6.3: Final acyclic vector field on the Klein bottle from the AM-model. Critical cells have its barycenters drawn in green.

Note that both homology generators in dimension 1 are not really cycles. Properly speaking, the homology generators are given below

$$g(\langle 0,8\rangle) = -\langle 0,1\rangle - \langle 1,2\rangle + \langle 0,8\rangle - \langle 2,8\rangle \tag{6.19}$$

$$g(\langle 6,9\rangle) = \langle 8,9\rangle - \langle 0,6\rangle - \langle 6,9\rangle + \langle 0,8\rangle \tag{6.20}$$

However, for the sake of simplicity and bearing in mind that g takes homology generators in $\mathcal{M}$ to cycles generating homology classes in $\mathcal{C}$, we use cells[5] in $\mathcal{M}$ as homology classes.

From the AM-model, the cohomology groups can also be easily calculated. This gives as result

$$\mathsf{H}^0 = \mathbb{Z}[\langle 9\rangle^*] \tag{6.21}$$

$$\mathsf{H}^1 = \mathbb{Z}[\langle 0,8\rangle^*] \tag{6.22}$$

$$\mathsf{H}^2 = \mathbb{Z}_2[\langle 2,5,6\rangle^*] \tag{6.23}$$

From the contravariance of cohomology functor, the dual reduction ρ^*_{AM} is given by

$$\rho^*_{AM} = (g^*, f^*, h^*) \tag{6.24}$$

[5]As every chain complex is a a sequence of free modules, we call cells to the elements in the corresponding basis.

Please note that the dual of the inclusion is the projection and viceversa. Hence, cohomology generators in the original complex are given below

$$f^*(\langle 9\rangle^*) = \langle 5\rangle^* + \langle 0\rangle^* + \langle 6\rangle^* + \langle 1\rangle^* + \langle 7\rangle^* + \langle 2\rangle^* + \\ + \langle 8\rangle^* + \langle 3\rangle^* + \langle 9\rangle^* + \langle 4\rangle^* \quad (6.25)$$

$$f^*(\langle 0,8\rangle^*) = \langle 6,9\rangle^* - \langle 2,6\rangle^* + \langle 6,8\rangle^* + \langle 4,8\rangle^* - \langle 5,6\rangle^* + \langle 0,3\rangle^* + \\ + \langle 0,8\rangle^* - \langle 3,4\rangle^* + \langle 0,2\rangle^* - \langle 3,5\rangle^* \quad (6.26)$$

$$f^*(\langle 2,5,6\rangle^*) = \langle 2,5,6\rangle^* \quad (6.27)$$

Cup product can also be easily calculated using the aproximation of the diagonal given by equation 5.2.

On simplicial Klein bottle, the cup product in integer cohomology is given by

$$\langle 9\rangle^* \smile \langle 9\rangle^* = \langle 9\rangle^* \quad (6.28)$$

$$\langle 9\rangle^* \smile \langle 0,8\rangle^* = \langle 0,8\rangle^* \quad (6.29)$$

$$\langle 9\rangle^* \smile \langle 2,5,6\rangle^* = \langle 2,5,6\rangle^* \quad (6.30)$$

$$\langle 0,8\rangle^* \smile \langle 9\rangle^* = \langle 0,8\rangle^* \quad (6.31)$$

$$\langle 0,8\rangle^* \smile \langle 0,8\rangle^* = 0 \quad (6.32)$$

$$\langle 0,8\rangle^* \smile \langle 2,5,6\rangle^* = 0 \quad (6.33)$$

$$\langle 2,5,6\rangle^* \smile \langle 9\rangle^* = \langle 2,5,6\rangle^* \quad (6.34)$$

$$\langle 2,5,6\rangle^* \smile \langle 0,8\rangle^* = 0 \quad (6.35)$$

$$\langle 2,5,6\rangle^* \smile \langle 2,5,6\rangle^* = 0 \quad (6.36)$$

Cubical Klein bottle

Now we will repeat the same calculation above using a cubical version of Klein bottle found in the source code associated to [31]. This cubical complex has 42 0-cells, 84 1-cells and 42 2-cells in $\mathbb{R}^6$.

Homology and cohomology groups are given by:

$$\mathsf{H}_0 = \mathbb{Z}[(0)\times(1)\times(0)\times(1)\times(1)\times(1)] \quad (6.37)$$

$$\mathsf{H}_1 = \mathbb{Z}[(1)\times(1)\times(1)\times(0)\times(0)\times(0,1)] \oplus \\ \oplus \mathbb{Z}_2[(1)\times(1)\times(0)\times(0)\times(0,1)\times(1)] \quad (6.38)$$

$$\mathsf{H}^0 = \mathbb{Z}[(0)\times(1)\times(0)\times(1)\times(1)\times(1)] \quad (6.39)$$

$$\mathsf{H}^1 = \mathbb{Z}[(1)\times(1)\times(1)\times(0)\times(0)\times(0,1)] \quad (6.40)$$

$$\mathsf{H}^2 = \mathbb{Z}_2[(0)\times(0,1)\times(0,1)\times(0)\times(0)\times(0)] \quad (6.41)$$

$\smile$	$\nu^{(0)}$	$\sigma^{(1)}$	$\tau^{(2)}$
$\nu^{(0)}$	ν	σ	τ
$\sigma^{(1)}$	σ	0	0
$\tau^{(2)}$	τ	0	0

Table 6.1: Product table on the cohomology ring of the Klein bottle.

as expected by previous calculations.

However, the (co)homology generators inside the original cubical complex are not presented here as they are too long and do not contribute with any interesting information.

Cup product is given by table 6.1, whose results are congruent with the simplicial case. Note that ν, σ and τ are the generator of cohomology at dimensions 0, 1 and 2, respectively.

Bing's house with two rooms

Bing's house, also known as house with two rooms, is a contractible[6] 2-complex that is not collapsible (see [3]). In figure 6.4 a cubical decomposition of Bing's house is shown.

The (co)homology groups are given below

$$\mathsf{H}_0 = \mathbb{Z}[(5) \times (5) \times (3)] \tag{6.42}$$
$$\mathsf{H}^0 = \mathbb{Z}[((5) \times (5) \times (3))^*] \tag{6.43}$$

The inclusion of the cohomology generator is a chain supported by 90 0-cells. Both results are consistent with the fact that Bing's house is null homotopic.

Of course, the cup product is trivial.

[6] A cell complex is contractible if it is homotopy equivalent to a point.

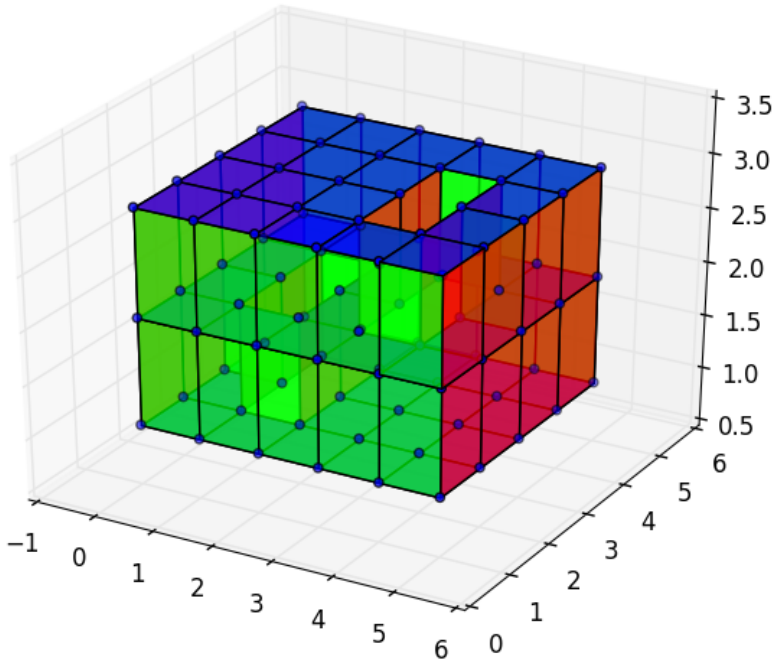


Figure 6.4: Cubical complex decomposition of Bing's house. Faces (cubical 2-cells) are coloured using its normal vector.

Torus and two handled sphere

Consider the torus $T = S^1 \times S^1$ with cubical decomposition given in figure 6.5. The (co)homology groups are:

$$\mathsf{H}_0 = \mathbb{Z}[(3) \times (3) \times (1)] \tag{6.44}$$
$$\mathsf{H}_1 = \mathbb{Z}[(2) \times (2,3) \times (0), (1,2) \times (1) \times (1)] \tag{6.45}$$
$$\mathsf{H}_2 = \mathbb{Z}[(0,1) \times (0,1) \times (0)] \tag{6.46}$$
$$\mathsf{H}^0 = \mathbb{Z}[(3) \times (3) \times (1)] \tag{6.47}$$
$$\mathsf{H}^1 = \mathbb{Z}[(2) \times (2,3) \times (0), (1,2) \times (1) \times (1)] \tag{6.48}$$
$$\mathsf{H}^2 = \mathbb{Z}[(0,1) \times (0,1) \times (0)] \tag{6.49}$$

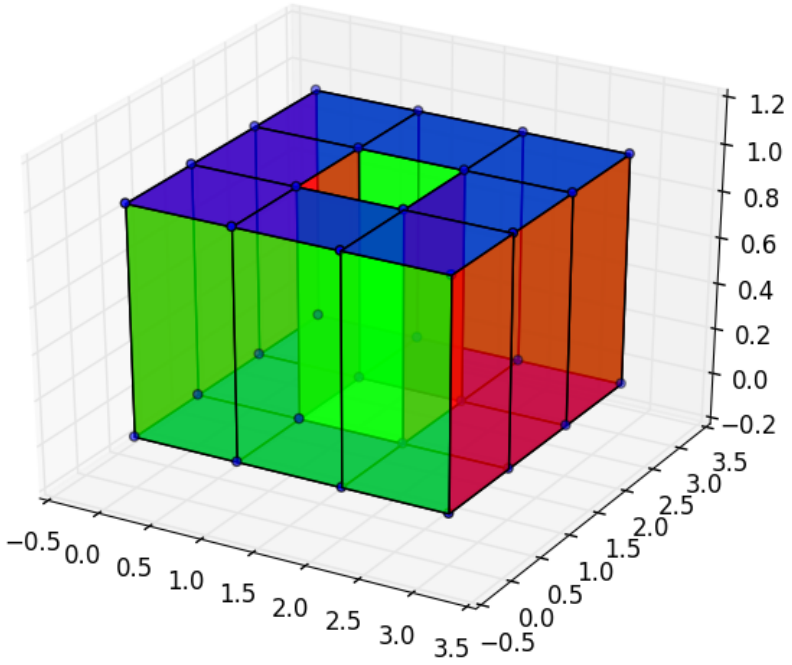


Figure 6.5: Cubical decomposition of torus.

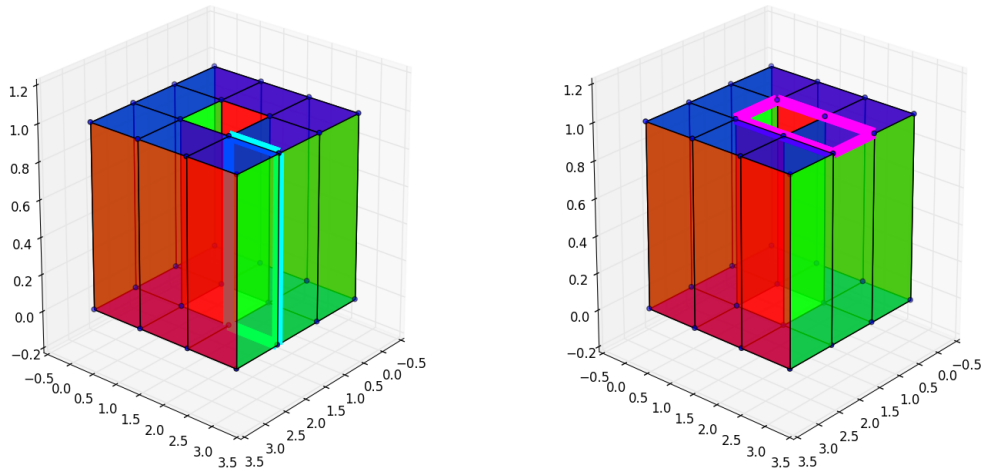


Figure 6.6: Homology generators at dimension 1

The two 1-cycles generating homology group at dimension 1, are

$$c_1^1 = -(2)\times(2,3)\times(1) + (2)\times(2,3)\times(0) - (2)\times(2)\times(0,1) + \\ +(2)\times(3)\times(0,1) \quad (6.50)$$

$$c_2^1 = (2)\times(2,3)\times(1) + (1,2)\times(1)\times(1) - (1)\times(1,2)\times(1) + \\ +(2)\times(1,2)\times(1) - (1)\times(2,3)\times(1) - (1,2)\times(3)\times(1) \quad (6.51)$$

The non trivial cohomology cup product is the one given by $\smile : \mathsf{H}^1 \times \mathsf{H}^1 \to \mathsf{H}^2$ where the non zero products are given by

$$(2)\times(2,3)\times(0) \smile (1,2)\times(1)\times(1) = (0,1)\times(0,1)\times(0) \quad (6.52)$$

Two handled sphere is the space $S^1 \vee S^2 \vee S^1$. Its homology is trivially isomorphic to the one of the torus. Below are the generators:

$$\mathsf{H}_0 = \mathbb{Z}[(0)\times(3)\times(1)] \quad (6.53)$$

$$\mathsf{H}_1 = \mathbb{Z}[(0)\times(0,1)\times(0), (0)\times(2,3)\times(0)] \quad (6.54)$$

$$\mathsf{H}_2 = \mathbb{Z}[(0,1)\times(1,2)\times(0)] \quad (6.55)$$

$$\mathsf{H}^0 = \mathbb{Z}[(0)\times(3)\times(1)] \quad (6.56)$$

$$\mathsf{H}^1 = \mathbb{Z}[(0)\times(0,1)\times(0), (0)\times(2,3)\times(0)] \quad (6.57)$$

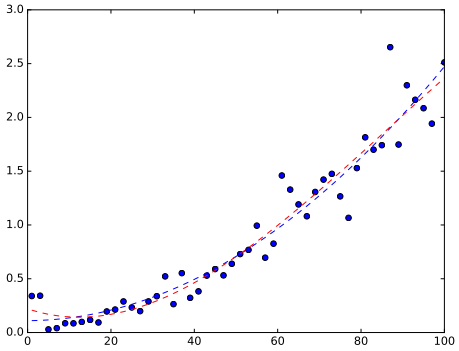
$$\mathsf{H}^2 = \mathbb{Z}[(0,1)\times(1,2)\times(0)] \quad (6.58)$$

Therefore, torus and two handled sphere are homologically indistinguishable. However, as previously shown, both spaces are not homotopy equivalent. The cup product is, in two handled sphere, trivial, as can be computed used the equation 5.1.

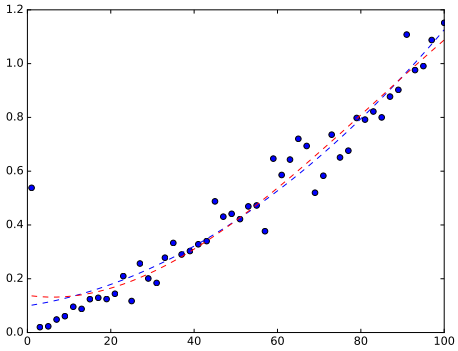
6.3 Empirical complexity analysis

We approach the complexity analysis of the proposed algorithms from an empirical point of view. We choose this approach as the algorithm complexity finally depends on the number of spurious critical cells, that are those critical cells in one step that are removed in the next stage.

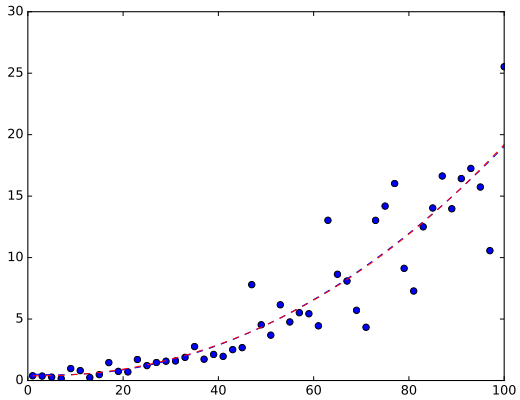
We perform this analysis as follows. First, a battery of random cubical complexes are created. Then, an AM-model is constructed for each complex. Finally, the point cloud formed by the variables *size* of the complexes, given by its number of cells, and *time* used in its computation in the development computer, is approximated using the best fitting curve. This curve represents, precisely, the empirical complexity of the proposed



(a) 2D cubical complexes



(b) 3D cubical complexes



(c) 4D cubical complexes

Figure 6.7: Empirical complexity graph for 2, 3 and 4 dimensional complexes. Blue dots represents pairs of (size, time) data. Blue and red lines represents the best fitting polynomial with degree 2 and 3 respectively.

algorithm. The curve election has been performed by experimentation with different types of curves, resulting the better results with polynomials of degree two and three. In figure 6.7 it can be observed how the empirical complexity is quadratic (blue line), with practically no difference with the cubical case (red line).

6.4 Conclusion

Along this chapter a software prototype is presented. This prototype is developed using Python as it is easy to read a write, is multiparadigm and has a lot of implemented modules that help to the developer in many different tasks. However, it is an interpreted language that experience sometimes less speed than other compiled language. Notwithstanding the foregoing, this implementation can be used as an "assembly map" to build another compiled version much faster.

Another improvement that can be done to the implemented prototype is related with the parallelization of the vector field simplification stage. As mentioned above, a parallel implementation of the spanning tree algorithm in [40] leads to an important speed up.

It also has to be improved one of the most used matrix operation, which is the product of matrices. In cases where the dimension of the matrices are greater than two thousand rows or columns, the multiplication performs so slow. Nonetheless, this can be improved using CUDA libraries.

As can be seen from previous paragraphs, many improvements can be done to the software in order to be used in real applications. However, this improvements are left to future work. However, the starting point determined by the work presented here is promising enough, as it shows a good complexity as can be observed in figure 6.7.

As final conclusion to this dissertation, two different approaches to the parallelization of (co)homological information extraction are presented. Both approaches lives in two different universes, however they are closely linked to each other. It can be thought as one being the concretion of other to the real world, as it can be easily implemented in current computers.

At this point it is beyond doubt the utility of Computational Algebraic Topology in many practical applications. Hence, the work presented here provides a tool to extract many interesting (co)homological information from digital objects regardless the dimension of the object or the ground ring. In many of the literature consulted, the tools and research are relegated to be only in 3D or the ground ring is supposed to be a field. In both cases the homology groups are torsion free. In this dissertation we focus, precisely, in ensuring that the torsion coefficients are calculated. This presents the main obstacle to a full Membrane Computing implementation of the information extraction procedure, as those coefficients have a purely algebraic nature, not combinatorial, which is better to be boarded from Membrane Computing.

Bibliography

[1] A. Alhazov, R. Freund, and M. Oswald. Tissue p systems with antiport rules and small numbers of symbols and cells. In *International Conference on Developments in Language Theory*, pages 100–111. Springer, 2005.

[2] F. Bernardini and M. Gheorghe. Cell communication in tissue p systems: universality results. *Soft Computing*, 9(9):640–649, 2005.

[3] R. Bing. Some aspects of the topology of 3-manifolds related to the poincaré conjecture. *Lectures on modern mathematics*, 2:93–128, 1964.

[4] J. Carnero, H. A. Christinal, D. Daniel, R. Reina-Molina, and M. Subathra. Improved parallelization of an image segmentation bio-inspired algorithm. In *Proceedings of the Second International Conference on Soft Computing for Problem Solving (SocProS 2012), December 28-30, 2012*, pages 75–82. Springer, 2014.

[5] H. Christinal, D. Díaz-Pernil, and P. Jurado. Using Membrane Computing for Obtaining Homology Groups of Binary 2D Digital Images. In *Combinatorial Image Analysis*, volume 5852 of *Lecture Notes in Computer Science*, pages 383–396. Springer Berlin / Heidelberg, 2009.

[6] H. Christinal, D. Díaz-Pernil, and P. Real. P systems and computational algebraic topology. *Mathematical and Computer Modelling*, 52(11-12):1982–1996, 2010.

[7] D. Díaz-Pernil, H. A. Christinal, M. A. Gutiérrez-Naranjo, and P. Real. Using membrane computing for effective homology. *Applicable Algebra in Engineering, Communication and Computing*, 23(5-6):233–249, 2012.

[8] D. Díaz Pernil, M. Á. Gutiérrez Naranjo, P. Real Jurado, V. Sánchez Canales, et al. A cellular way to obtain homology groups

in binary 2d images. *Proceedings of the Eighth Brainstorming Week on Membrane Computing, 89-99. Sevilla, ETS de Ingeniería Informática, 1-5 de Febrero, 2010*, 2010.

[9] D. Díaz-Pernil, M. Á. G. Naranjo, P. R. Jurado, and V. Sánchez-Canales. A new way to obtain homology groups in binary 2d images using membrane computing. In *EACA 2010: XII Encuentro de Álgebra Computacional y Aplicaciones= 12th Meeting on Computer Algebra and Applications: Libro de resúmenes= Book of abstracts: Santiago de Compostela, 19-21 de julio de 2010*, pages 107–112. Servicio de Publicaciones, 2010.

[10] S. Eilenberg and S. MacLane. On the groups h (π, n), ii: Methods of computation. *Annals of Mathematics*, pages 49–139, 1954.

[11] R. Forman. Morse theory for cell complexes. *Advances in mathematics*, 134(1):90–145, 1998.

[12] R. Freund, L. Kari, M. Oswald, and P. Sosík. Computationally universal p systems without priorities: two catalysts are sufficient. *Theoretical Computer Science*, 330(2):251–266, 2005.

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

[14] R. Gonzalez-Diaz, P. Real, et al. Computation of cohomology operations of finite simplicial complexes. *Homology, Homotopy and Applications*, 5(2):83–93, 2003.

[15] A. Gyulassy, P.-T. Bremer, B. Hamann, and V. Pascucci. A practical approach to morse-smale complex computation: Scalability and generality. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1619–1626, 2008.

[16] A. Hatcher. *Algebraic Topology*. Cambridge Univ. Press, 2001.

[17] T. Head. Formal language theory and dna: an analysis of the generative capacity of specific recombinant behaviors. *Bulletin of mathematical biology*, 49(6):737–759, 1987.

[18] J. H. Holland. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press, 1992.

[19] T. Kaczynski, K. Mischaikow, and M. Mrozek. *Computational homology*, volume 157. Springer Science & Business Media, 2006.

[20] T. Kaczynski and M. Mrozek. The cubical cohomology ring: An algorithmic approach. *Foundations of Computational Mathematics*, 13(5):789–818, 2013.

[21] D. Kozlov. *Combinatorial algebraic topology*, volume 21. Springer Science & Business Media, 2007.

[22] S. N. Krishna. On pure catalytic p systems. In *International Conference on Unconventional Computation*, pages 152–165. Springer, 2006.

[23] S. N. Krishna, K. Lakshmanan, and R. Rama. Tissue p systems with contextual and rewriting rules. In *Workshop on Membrane Computing*, pages 339–351. Springer, 2002.

[24] S. N. Krishna and A. Pâun. Results on catalytic and evolution-communication p systems. *New Generation Computing*, 22(4):377–394, 2004.

[25] T. Lewiner. Constructing discrete morse functions. *Master's thesis, PUC-Rio*, 2002.

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

[27] C. Martín-Vide, J. Pazos, G. Păun, and A. Rodríguez-Patón. A new class of symbolic abstract neural nets: Tissue p systems. In *Computing and Combinatorics*, volume 2387 of *Lecture Notes in Computer Science*, pages 573–679. Springer Berlin / Heidelberg, 2002.

[28] W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[29] J. Nickolls, I. Buck, M. Garland, and K. Skadron. Scalable parallel programming with cuda. *Queue*, 6(2):40–53, 2008.

[30] G. Păun, M. J. Pérez-Jiménez, and A. Riscos-Nunez. Tissue p systems with cell division. *International Journal of Computers Communications & Control*, 3(3):295–303, 2008.

[31] P. Pilarczyk and P. Real. Computation of cubical homology, cohomology, and (co) homological operations via chain contraction. *Advances in Computational Mathematics*, 41(1):253–275, 2015.

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

[33] R. Reina Molina, J. Carnero Iglesias, and D. Díaz Pernil. Image segmentation using tissue-like p systems with multiple auxiliary cells. *Image-A: Applicable Mathematics in Image Engineering, 2 (4), 25-28*, 2011.

[34] R. Reina Molina, D. Díaz Pernil, and M. Á. Gutiérrez Naranjo. Cell complexes and membrane computing for thinning 2d and 3d images. *Proceedings of the Tenth Brainstorming Week on Membrane Computing,(2) 167-186. Sevilla, ETS de Ingeniería Informática, 30 de Enero-3 de Febrero, 2012,*, 2012.

[35] R. Reina-Molina, D. Díaz-Pernil, P. Real, and A. Berciano. Membrane parallelism for discrete morse theory applied to digital images. *Applicable Algebra in Engineering, Communication and Computing*, 26(1-2):49–71, 2015.

[36] G. X. Ritter, J. N. Wilson, and J. L. Davidson. Image algebra: An overview. *Computer Vision, Graphics, and Image Processing*, 49(3):297–331, 1990.

[37] A. Romero and F. Sergeraert. Discrete vector fields and fundamental algebraic topology. *arXiv preprint arXiv:1005.5685*, 2010.

[38] J.-P. Serre. Homologie singuliere des espaces fibres. *Annals of Mathematics*, pages 425–505, 1951.

[39] G. Van Rossum and F. L. Drake Jr. *Python reference manual*. Centrum voor Wiskunde en Informatica Amsterdam, 1995.

[40] V. Vineet, P. Harish, S. Patidar, and P. Narayanan. Fast minimum spanning tree for large graphs on the gpu. In *Proceedings of the Conference on High Performance Graphics 2009*, pages 167–171. ACM, 2009.

[41] S. v. d. Walt, S. C. Colbert, and G. Varoquaux. The numpy array: a structure for efficient numerical computation. *Computing in Science & Engineering*, 13(2):22–30, 2011.