Dipartimento di Informatica, Bioingegneria,
Robotica ed Ingegneria dei Sistemi

Topological Data Analysis through Homology and Discrete Morse Theory

by

Ulderico Fugacci

Theses Series **DIBRIS-TH-2017-04**

DIBRIS, Università di Genova
Via Opera Pia, 13 16145 Genova, Italy http://www.dibris.unige.it/

Università degli Studi di Genova

Dipartimento di Informatica, Bioingegneria,
Robotica ed Ingegneria dei Sistemi

Dottorato di Ricerca in Informatica

Ph.D. Thesis in Computer Science

Topological Data Analysis through Homology and Discrete Morse Theory

by

Ulderico Fugacci

December, 2017

Dottorato in Informatica
Dipartimento di Informatica, Bioingegneria, Robotica ed Ingegneria dei Sistemi
Università degli Studi di Genova

DIBRIS, Univ. di Genova
Via Opera Pia, 13
I-16145 Genova, Italy
http://www.dibris.unige.it/

Ph.D. Thesis in Computer Science (S.S.D. INF/01)

Submitted by Ulderico Fugacci
DIBRIS, Univ. di Genova
ulderico.fugacci@dibris.unige.it

Date of submission: 19 February 2016

Title: Topological Data Analysis through Homology and Discrete Morse Theory

Advisors:
Leila De Floriani
Dipartimento di Informatica, Bioingegneria, Robotica ed Ingegneria dei Sistemi
Università degli studi di Genova
deflo@disi.unige.it

Maria Evelina Rossi
Dipartimento di Matematica
Università degli studi di Genova
rossim@dima.unige.it

Ext. Reviewers:
Claudia Landi
Dipartimento di Scienze e Metodi dell'Ingegneria
Università di Modena e Reggio Emilia
claudia.landi@unimore.it

Heike Leitte
Interdisciplinary Center for Scientific Computing
Heidelberg University
heike.leitte@iwr.uni-heidelberg.de

Abstract

Two of the most relevant tools in topological data analysis are homology and discrete Morse theory. Homology and its more recent development, persistent homology, provide topological information on an object including connectivity and the classification of loops, handles and voids within the space. Discrete Morse theory, on the other hand, is a powerful analysis tool providing a morphology- and homology-consistent model of the space to be analyzed. The major objectives of this thesis are to investigate homology and discrete Morse theory, to understand and reveal relations between them and with other analysis tools, massively exploit them to efficiently retrieve relevant topological information from large-size and high-dimensional data. Since we are dealing with complexes defined by very large point clouds, our work has been developed considering and introducing compact and efficient data structures.

A first contribution of the thesis is an in-depth investigation and a complete classification of the techniques described in the literature to algorithmically build a (discrete) Morse complex and of the algorithms for computing homology and persistent homology. Another contribution of our work concerns the relationships between the construction of a discrete Morse gradient and the efficient computation of standard and persistent homology of a simplicial complex. A theoretical comparison of different methods based on homology-preserving operators to build a discrete Morse gradient and the use of compact data structures for representing the complex and the discrete Morse gradient lead us to develop an algorithm to efficiently compute standard and persistent homology of a simplicial complex.

Another way to efficiently compute homology is by considering a hierarchical representation of the complex. To this aim, we introduce a general and theoretically-consistent definition of a hierarchical, multi-resolution model. We develop homology-preserving versions of this model for cell and simplicial complexes and we exploit their expressive power to efficiently compute homology and homology generators at various degrees of resolution.

We have also considered discrete Morse theory as a tool for studying the morphology of an object through its segmentation as a Morse-Smale complex. Working with real data, the presence of noise often requires a morphological simplification of the dataset in order to capture only the relevant topological information. Unfortunately, the simplification process can generate topolog-

ically-inconsistent representations of the object. In this work, we introduce a new simplification process that prevents inconsistencies and we define a compact data structure for representing a discrete Morse complex improving the efficiency of such a process.

Relationships between homology and discrete Morse theory have also been investigated from a combinatorial and algebraic point of view. By deeply exploiting a correspondence between simplicial complexes and ideals in the polynomial ring, we establish a first relation between the existence of a perfect Morse function for a simplicial complex and the algebraic property of admitting a splitting for the corresponding polynomial ideal.

Acknowledgments

I wish to thank some people who played a fundamental role during my Ph.D. studies.

Maria Evelina Rossi for her wisdom in introducing me to the computer science world.
Leila De Floriani for making me fall in love with that world.
Emanuela De Negri for her constant willingness to listen.
Paola Magillo and Lidija Čomić for their patience.
Federico Ciccio Iuricich because he has been "simplicially" essential.
Davide Bolognini for being a theoretically-consistent eclectic.
Sara Scaramuccia because we are homologically equivalent.
Riccardo Fellegara for his "stellar" help.

This work has been partially supported by the National Science Foundation under Grant Number IIS-1116747.

Contents

Introduction

The aim of this thesis is the study and the development of structural methods in topological data analysis. Nowadays, one of the most exciting and pressing challenge in several scientific disciplines is to face data of *high dimensions*, often *unorganized* and of *large size*. In order to overcome the limitations affecting the geometrical methods in the analysis of high-dimensional datasets, in our work we have chosen to consider *topological tools*. These methods allow to describe data independently from geometrical coordinates and to retrieve just the crucial information about their shape. Point-cloud information leads to the construction of irregularly distributed and unorganized data. In order to treat a large class of data which could include also the last cited one, we have elected *simplicial complexes* as the privileged mathematical structure to represent their connectivity and shape. Further, high dimensionality and large size require to develop *new theoretical approaches* to retrieve the topological information and the use of *efficient and compact data structures*.

The two main tools used in our work to reach these goals are *homology* and *discrete Morse theory*. In the thesis, we have deeply exploited them to efficiently retrieve topological information of a shape and we have investigated the connections between these two notions. *Homology* is a topological invariant which gives a basic description of the shape of an object by counting and geometrically retrieving its "holes", allowing to extract information such as the number of the connected components, the independent non-bounding cycles and the generalizations in higher dimensions of these notions. Recently, a generalization of homology, called *persistent homology*, has been proposed [EH08]. Persistent homology is able to manage a "sequence" of complexes representing an object which evolves with respect to a parameter. In a nutshell, persistent homology describes the changes in homology that occur during the evolution of the considered object. This property makes persistent homology a very flexible tool which plays a crucial role in several application domains.
Discrete Morse theory is a combinatorial version of the more general and continuous Morse theory [For98]. It enables to analyze the topology of an object by studying functions defined on it. Discrete Morse theory is relevant for different applications, roughly subdivided in two classes: applications devoted to *homological analysis* and *segmentation of shapes* into regions of influence of the critical points of a function defined on them.

The first contribution in our investigation is a complete and systematic study of the current *state of the art* (Chapter 2). We classify and compare also experimentally several

topological data structures for representing a simplicial complex revealing that, in order to manage objects in high dimensions, a data structure explicitly encoding only a subset of the entities of a complex is required [FID14, FIDon]. Our work investigates several techniques to speed up the computation of homology and persistent homology with respect to the standard but time-consuming method to retrieve them. According to the strategy they adopt, we classify these approaches in: direct optimizations, coarsening and pruning approaches, distributed approaches and methods based on annotations. Our analysis of the state of the art finally involves Morse theory. We discuss various discretizations of this theory and we describe different methods to extract a Morse complex indicating them as boundary-based, region-growing, watershed and Forman-based approaches. Further, we discuss about the encoding of a Morse complex and we focus our attention on the Forman's discrete Morse theory describing constrained and unconstrained algorithms to retrieve a Forman gradient [DFIM15, DFI15].

Two key contributions of this thesis are related to the development of techniques to *efficiently compute homological information* of a simplicial complex. One is based on discrete Morse theory, the other on a multi-resolution model.

The first strategy adopted is based on the efficient construction of a discrete Morse gradient in order to build a Morse complex from which quickly retrieve homology and persistent homology of a simplicial complex (Chapter 3). We discuss about the use of two homology-preserving operators (reduction and coreduction) and about the possibility of combine them in order to build a valid gradient on a simplicial complex. Our comparison leads to prove a theoretical equivalence between the capabilities of these different methods. Based on these considerations, we develop an algorithm to efficiently compute homological information of a simplicial complex based on the compact IA^* data structure. The equivalence between the use of various operators gives us the possibility to choose coreductions as privileged operators since they can be better performed by the chosen data structure. At the same time, the use of this data structure implies the need of the development of a new encoding of the gradient that reveals to be very compact. The developed algorithm, based on a partition of the input simplicial complex, produces satisfying experimental results compared with similar approaches proposed in the literature [FID14, FIDon].

A new developed strategy to efficiently retrieve homology is based on the use of a *multi-resolution model* (Chapter 4). In order to reach this goal, we work on an in-depth study of the topological operators for modifying cell and simplicial complexes and we develop the definition of a general geometry-based multi-resolution model. We specialize this general model in the context of cellular and simplicial complexes. Further, focusing our attention on homology-preserving modifications, we propose, for each specialization, an efficient encoding of the model and algorithms for updating homology and homology generators. The homology-preserving multi-resolution model for cell complexes has been implemented and experimental results reveal its efficiency in the homology computation and in the retrieval of homology generators at various levels of detail [ČomićDIF14].

Driven by the desire of further improvements in persistent homology computation and of handle real datasets describing scalar fields, we study some problems affecting the *simplification of a discrete Morse complex* (Chapter 5). Open issues concerning this topic include the lack of a compact data structure able to efficiently perform simplification process and the presence of topological inconsistencies for simplifications involving complexes of dimension 3 or greater [GRSW13]. In our work, we solve both these problems by proposing a new data structure for Morse complexes able to combine compactness and efficiency during the simplification step and a new simplification algorithm that has been proven to be topologically consistent. The algorithm is based on the use of the $remove$ operator ensuring the topological consistence and on a preprocessing step in which all the gradient configurations leading to topological ambiguities are eliminated [IFD15].

Finally, connections between homology and discrete Morse theory are investigated from an *algebraic point of view* (Chapter 6). The theoretical bridge allowing to relate topology and algebra is given by the Stanley-Reisner correspondence [Rei76, Sta75]. This tool allows us to "translate", in both directions, geometrical and topological properties of a simplicial complex into combinatorial and algebraic properties of a monomial squarefree ideal of the polynomial ring. Exploiting this tool, we prove that, under suitable assumptions, admitting for a simplicial complex a perfect Morse function ensures the existence of particular homology-consistent decompositions, called homological and Betti splittings, for the correspondent polynomial ideal [BFRDon].

Chapter 1

Background Notions

The main objective of this thesis is the analysis of discretized shapes through topological tools. In this chapter, we introduce some background notions useful to handle these tools. Simplicial and cell complexes, defined in Section 1.1, are the mathematical structures we use to represent discretized shapes. In Section 1.2, we introduce the notions of homology and of persistent homology which are the main topological descriptors considered in our work. Finally, in Section 1.3, we describe Morse and discrete Morse theory which represent powerful tools to efficiently manage the topological information of a shape.

1.1 Simplicial and cell complexes

In the literature, several notions have been introduced to formally describe and to computationally work with geometrical objects. Simplicial complexes represent the mathematical tool we use for discretizing shapes. This choice is mainly due to their capability in representing also shapes originated from unorganized data and their combinatorial properties that allow to efficiently encode them.

In this section, we introduce the notion of simplicial complex [Mun84]. Other classes of complexes we are going to present in this section are regular grids [ČDMI14, KMM04], frequently used in the literature for representing regular data, and cell complexes [LW69, Mas91, Hat02] that, thanks to their generality, are mainly used to prove theoretical results for a wide class of complexes.

1.1.1 Simplicial complexes

Simplicial complexes are built upon the concept of *simplex*. Our presentation is based on [Mun84].

Definition 1.1. A set $\{v_0, v_1, ..., v_k\}$ of points in the n-dimensional real space $\mathbb{R}^n$ is said

to be *geometrically independent*, if for any real values t_i, equations

$$\sum_{i=0}^{k} t_i = 0 \qquad \text{and} \qquad \sum_{i=0}^{k} t_i v_i = 0$$

imply that $t_0 = t_1 = \cdots = t_k = 0$.

The condition of geometric independence for the set $\{v_0, v_1, ..., v_k\}$ is equivalent to requiring that vectors $v_1 - v_0, ..., v_k - v_0$ are linearly independent over $\mathbb{R}$. This condition is therefore also equivalent to affine independence, i.e., to the request that the smallest affine subspace of $\mathbb{R}^n$ containing the points $v_0, ..., v_k$ has dimension k. Two distinct points in $\mathbb{R}^n$ form a geometrically independent set, as do three non-collinear points, four non-coplanar points, etc.

Definition 1.2. Let $V = \{v_0, v_1, ..., v_k\}$ be a geometrically independent set in $\mathbb{R}^n$. We define the *k-simplex* σ spanned by $v_0, v_1, ..., v_k$ to be the convex hull of V, i.e., the set of all points $x \in \mathbb{R}^n$ such that

$$x = \sum_{i=0}^{k} t_i v_i \qquad \text{where} \qquad \sum_{i=0}^{k} t_i = 1$$

and $t_i \geq 0$ for all i.

The numbers t_i are uniquely determined by x; they are called *barycentric coordinates* of x in σ with respect to $v_0, v_1, ..., v_n$.

For example a 0-simplex is a vertex, a 1-simplex is an edge, a 2-simplex is a triangle, a 3-simplex is a tetrahedron, as shown in Figure 1.1.

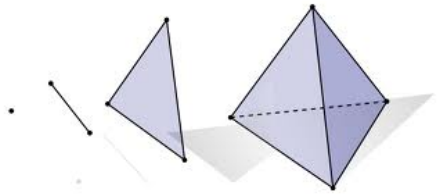
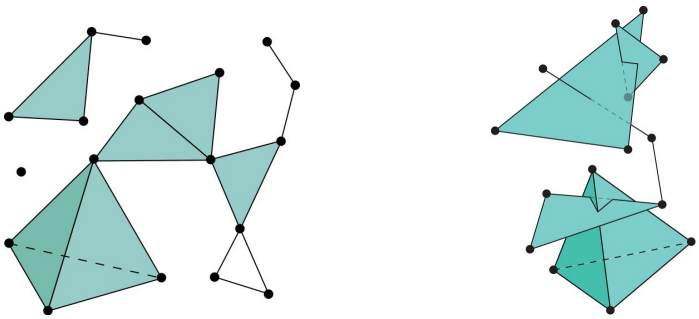


Figure 1.1: Examples of simplices of dimension 0, 1, 2, 3.

The points $v_0, v_1, ..., v_k$ spanning a simplex σ are called the *vertices* of σ; k is said to be the *dimension* of σ, and it is denoted as $\dim \sigma$. Any simplex σ' spanned by a subset of V is called a *face* of σ. Conversely, σ is called a *coface* of σ'.

Definition 1.3. A *simplicial complex* Σ in $\mathbb{R}^n$ is a collection of simplices in $\mathbb{R}^n$ such that

1. every face of a simplex of Σ is in Σ;
2. the intersection of any two simplices of Σ is a face of each of them.



(a) (b)

Figure 1.2: A simplicial complex (a) and a collection of simplices that is not a simplicial complex (b).

Remark 1.4. It is easy to show that condition 2. is equivalent to

2'. every pair of distinct simplices of Σ have disjoint interiors.

We define the *dimension* of a simplicial complex Σ in $\mathbb{R}^n$, denoted as $\dim \Sigma$, to be the supremum of the dimensions of the simplices of Σ. We call *maximal simplices* the elements σ of Σ such that $\dim \sigma = \dim \Sigma$. We say that a simplex of Σ is a *top simplex* if it is not a proper face of any simplex of Σ. Clearly, a maximal simplex is also a top simplex but the converse, in general, does not hold. A *simplicial d-complex* is a simplicial complex in which all the top simplices are maximal of dimension d. A subcollection of Σ that is itself a simplicial complex is called a *subcomplex* of Σ.

Simplicial complexes are a natural choice to represent geometric objects computationally since these latter can be represented by an enumeration of the top simplices. The knowledge of the top simplices of a complex uniquely determines the whole complex.

Now, we endow a simplicial complex Σ with a topological structure.

Definition 1.5. Let $|\Sigma|$ be the subset of $\mathbb{R}^n$ defined as the union of the simplices of Σ. Let us give to each simplex σ the subspace topology τ_σ with respect to the Euclidean space $\mathbb{E}^n$. Then, we define a topology τ_Σ on $|\Sigma|$ by saying that a subset A of $|\Sigma|$ is a closed set of $(|\Sigma|, \tau_\Sigma)$ if and only if $A \cap \sigma$ is a closed set of (σ, τ_σ), for each σ in Σ. It is easy to see that τ_Σ is a topology on $|\Sigma|$, since this collection of closed sets is closed under finite unions and arbitrary intersections. The space $|\Sigma|$ is called the *underlying space* of Σ, or the *polytope* of Σ.

Remark 1.6. The topology τ_Σ of $|\Sigma|$ is finer than the topology $\tau_{|\Sigma|}$ that $|\Sigma|$ inherits as a subspace of $\mathbb{R}^n$. If A is a closed set of $|\Sigma|$ with respect to the subspace topology $\tau_{|\Sigma|}$, then $A = B \cap |\Sigma|$ for some closed set B in $\mathbb{E}^n$. This implies that $B \cap \sigma$ is a closed set of (σ, τ_σ) for each σ and, so, $B \cap |\Sigma| = A$ is a closed set of $(|\Sigma|, \tau_\Sigma)$.

The two topologies τ_Σ and $\tau_{|\Sigma|}$ are different in general.

Example 1.7. Let Σ be the collection of all 1-simplices in $\mathbb{R}$ of the form $[m, m+1]$, where m is an integer different from 0, along with all the simplices of the form $[1/(n+1), 1/n]$ for n a positive integer, along with all the faces of these simplices. Then, Σ is a complex whose underlying space is equal to $\mathbb{R}$ as a set, but not as a topological space. For instance, the set of the points of the form $1/n$ is a closed set of $(|\Sigma|, \tau_\Sigma)$, but not of $\mathbb{E} = (|\Sigma|, \tau_{|\Sigma|})$.

Remark 1.8. If Σ is a finite collection of simplices, $\tau_\Sigma = \tau_{|\Sigma|}$. Let us suppose Σ is finite and A is a closed set of $(|\Sigma|, \tau_\Sigma)$. Then, $A \cap \sigma$ is a closed set of (σ, τ_σ) and, hence, it is a closed set of $\mathbb{E}^n$. Since A is the union of finitely many sets $A \cap \sigma$, the set A is also a closed set of $\mathbb{E}^n$ and, so, $\tau_{|\Sigma|}$ is finer than τ_Σ.
The above argument does not work if Σ is an infinite collection of simplices.

One of the reasons to give to a polytope the topological structure just defined is the following characterization of continuous functions. For a polytope provided with the subspace topology the following proposition does not hold in general.

Proposition 1.9. *Let Σ be a simplicial complex and let (X, τ) be a topological space. A function f from $(|\Sigma|, \tau_\Sigma)$ to (X, τ) is continuous if and only if $f|_\sigma$ is continuous for each $\sigma \in \Sigma$.*

Proof. Since σ is a subspace of $|\Sigma|$, if f is continuous, so is $f|_\sigma$. Conversely, assume that each map $f|_\sigma$ is continuous. If A is a closed set of (X, τ), then, for each $\sigma \in \Sigma$, $f^{-1}(A) \cap \sigma = (f|_\sigma)^{-1}(A)$, which is a closed set of (σ, τ_σ) by continuity of $f|_\sigma$. Thus, $f^{-1}(A)$ is a closed set of $(|\Sigma|, \tau_\Sigma)$. □

Definition 1.10. Let Σ, Σ' be two simplicial complexes. A function $f : \Sigma \to \Sigma'$ is called a *simplicial map* if for every simplex $\sigma = v_0 v_1 \cdots v_k$ in Σ, $f(\sigma) = f(v_0) f(v_1) \cdots f(v_k)$ is a simplex in Σ'. The restriction f_V of f to the set of vertices of Σ is called a *vertex map*.

From now on, we will consider only *finite* simplicial complexes simply referring to them as simplicial complexes. The motivation is to endow simplicial complexes with the topological structure better reflecting our intuition and to be able to computationally handle a simplicial complex.

Topological manifolds represent a relevant class of topological spaces. In the following, we denote a simplicial complex whose polytope is a manifold as a *(simplicial) mesh*. Simplicial meshes of dimension 2 and 3 will be called triangle and tetrahedral meshes, respectively. Please refer to Section 6.1 for a discussion about topological manifolds and simplicial complexes.

In some application domains such as combinatorics, it is useful to work with simplicial complexes without considering their geometric realization. The notion of abstract simplicial complex meets this need.

Definition 1.11. An *abstract simplicial complex* Σ on a set V is a collection of finite subsets of V, called simplices, such that if $\tau \in \Sigma$, $\sigma \subseteq \tau$, then $\sigma \in \Sigma$.

According to the standard definition of simplicial complex, the elements of V are called *vertices* of Σ, the dimension of a simplex τ is one less than the number of its elements and the largest dimension of the simplices in Σ is called *dimension* of Σ. Each nonempty subset σ of a simplex $\tau \in \Sigma$ is called a *face* of τ and, conversely, τ is a *coface* of σ.

The two notions of simplicial complex and abstract simplicial complex are equivalent. It always possible, given an abstract simplicial complex Σ, to endow Σ with a geometric realization or, given a simplicial complex, to forget its geometry thus obtaining an abstract simplicial complex [Mun84].

1.1.1.1 Classes of simplicial complexes

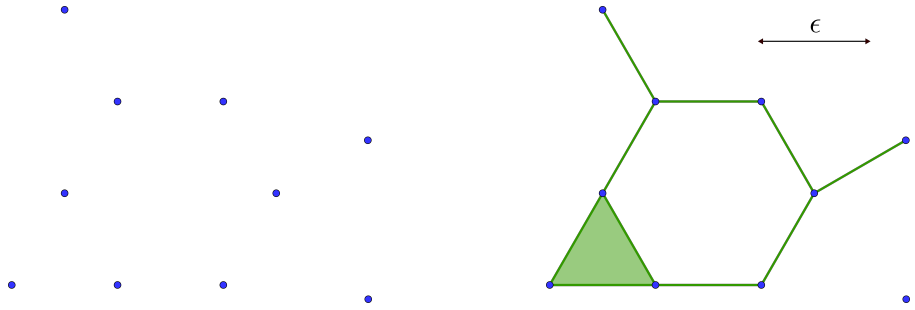
Scientific data sets are usually in the form of *point cloud* data, characterized by a set of points in an ambient space, for instance datasets from particle physics, electoral votes or data for weather forecast [RL14]. Many types of simplicial complexes can be used to connect the data points. Other datasets are networks, which can be sensor networks [DG07b] or collaboration or social networks [MSS06, HMR09, KSSM13, CH13]. In such cases, each node of the network represents a vertex of the simplicial complex and the k-simplices are created by semantic relations inferred by the application: a joint work among k actors in collaborative networks, in social networks an k-simplex would represent a set of k actors mutually connected. Sensor networks provide parameters for building a simplicial complex: knowing the working radius of each sensor a simplex is defined among all the sensors mutually included in the circle generated by the working radius of each other [DG07b].

Much work has been done for building simplicial complexes from point clouds embedded in high-dimensional metric spaces. *α-shapes* [EKS83], *combinatorial Delaunay triangulations* [CDEG10], *Čech complexes* [Hat02], *Vietoris-Rips complexes* [Zom10a], *witness complexes* [DC04, De 03, GO08] and graph induced complexes [DFW13] are all examples of those complexes. Given a finite set of points P in a metric space (such as the Euclidean space) and a positive real number ϵ, a subset of $k+1$ points in P determines a k-simplex in a Čech complex if and only if such points lie in a ball of radius $\epsilon/2$. A Čech complex is one of the most classical ways to build a simplicial complex starting from a point cloud, but its construction is infeasible in practice. Vietoris-Rips (VR) complexes, approximation of the Čech complexes, have been widely used [Zom10a].

Definition 1.12. Given a finite set of points P in a metric space (X, d) and a positive parameter ϵ, the *Vietoris-Rips (VR) complex* associated with P and ϵ is the abstract simplicial complex on P defined as

$$V_\epsilon(P) := \{\sigma \subseteq P \,|\, d(u,v) \leq \epsilon,\, \forall u \neq v \in \sigma\}$$

An example of VR complex is depicted in Figure 1.3. Informally, chosen a set P and a positive number ϵ, the associated VR complex consists of a simplex for every subset of points of P that has diameter at most ϵ.
Also VR complexes are often too large to handle even in dimension as low as three. The



(a) (b)

Figure 1.3: (a) A finite set of points P. (b) The Vietoris-Rips complex associated with P choosing as ϵ the depicted distance.

witness complex [DC04, De 03, GO08] and the graph induced complex [DFW13] try to overcome this problem through a subsampling strategy.

1.1.2 Cell complexes and regular grids

Simplicial complexes are not the unique class of mathematical objects useful to discretize a shape. In this subsection, we briefly introduce the notions of cell complexes and regular grids.

1.1.2.1 Cell and CW complexes

Cell and CW complexes are two different ways to generalize the class of the simplicial complexes. Even if their relevance in applications is more limited, their generality is very useful for proving theoretical results for a large class of complexes. For a more detailed presentation of cell and CW complexes, see [LW69, Mas91, Hat02].

Let $\mathbb{R}^k$ be the k-dimensional Euclidean space. We denote as $D^k := \{x \in \mathbb{R}^k \,:\, |x| \leq 1\}$ the *k-dimensional closed disk* and as $\mathrm{int}(D^k)$ and $\mathrm{fr}(D^k)$ its interior and its boundary (or frontier), respectively.

A *k-cell* is a set p homeomorphic to $\mathrm{int}(D^k)$. Moreover, k is said to be the *dimension* of p and denoted $\dim p$.

Definition 1.13. A Hausdorff space $\Gamma \subseteq \mathbb{R}^n$ is called a *cell complex* if Γ is a disjoint union of cells such that:

- for each k-cell p of Γ, there exists a continuous function $\Phi_p : D^k \to \Gamma$, such that its restriction $\Phi_p|_{\mathrm{int}(D^k)} : \mathrm{int}(D^k) \to p$ to $\mathrm{int}(D^k)$ is a homeomorphism;
- $\Phi_p(\mathrm{fr}(D^k)) \subseteq \Gamma^{(k-1)}$, where $\Gamma^{(i)}$, called the *i-skeleton* of Γ, is the union of the cells of Γ with dimension less than or equal to i.

Intuitively, a cell complex is just a collection of cells homeomorphic to a open disks suitably "glued" together. We define the *dimension* of the cell complex Γ to be the supremum of the dimensions of its cells.

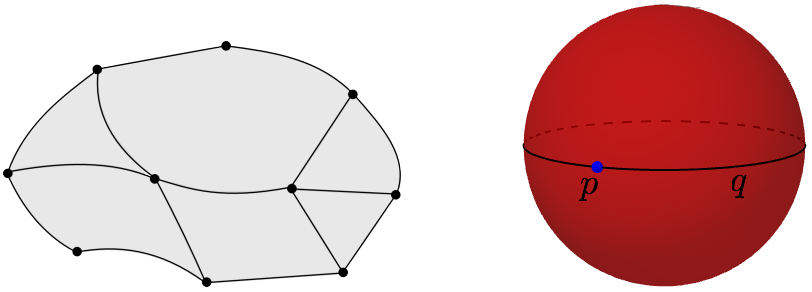
The notion of CW complex allows endowing a cell complex with nice topological features. The letter C stands for closure finiteness and W stands for weak topology.

Definition 1.14. A *CW complex* Γ is a cell complex satisfying the following two properties:

- *Closure finiteness*: for each k-cell p of Γ, $\Phi_p(\mathrm{fr}(D^k))$ is contained in the union of a finite number of cells of $\Gamma^{(k-1)}$;
- *Weak topology*: Γ is endowed with the topology for which $A \subseteq \Gamma$ is a closed set of Γ if and only if, for all $p \in \Gamma$, $A \cap p$ is a closed set of p.

Properties in Definition 1.14 trivially hold if Γ is a finite cell complex. Since in our work we only consider finite cell complexes, in the following we do not distinguish anymore between the two notions and we use only the term cell complex.

Definition 1.15. A cell complex is called *regular* if, for each $p \in \Gamma$, Φ_p is bijective (or, equivalently, a homeomorphism) on its image.



(a) (b)

Figure 1.4: (a) A regular cell complex of dimension 2. (b) A non regular cell complex of dimension 2; 0-cell p represents a irregular face for 1-cell q.

As depicted in Figure 1.4, regularity ensures that no identification occurs on the boundary of each cell. Given a cell complex Γ, a cell $p \in \Gamma$ is called *(proper) face* of a cell $q \in \Gamma$ if p is contained in $\mathrm{fr}(q)$. Conversely, q is called *(proper) coface* of the cell p. A k-cell p of Γ which is face of a $(k+1)$-cell q of Γ is called *regular* if the closure of $\Phi_q^{-1}(p)$ is homeomorphic to the k-dimensional closed disk D^k. Otherwise, p is called an *irregular* face of q.
For $k > 0$, a k-cell p is said to be *adjacent* to a k-cell p' if p and p' share a $(k-1)$-face. Two 0-cells u and v are called *adjacent* if they are both faces of a same 1-cell.
The star and the link of a cell p are relevant subsets of a cell complex Γ useful to describe the neighborhood of p. The *star* of a cell p of a cell complex Γ is the set of cells $q \in \Gamma$ which are cofaces of p and is denoted as $\mathrm{star}_\Gamma\, p$. The *link* of a cell $p \in \Gamma$, denoted as

$\mathrm{link}_\Gamma\, p$, is the set of cells $q \in \Gamma$ such that q is a face of a coface of p, and is not a coface of p. Further, we define the *closed star* of a cell $p \in \Gamma$ as the cell complex $\overline{\mathrm{star}}_\Gamma\, p$ obtained by the union of $\mathrm{star}_\Gamma\, p$ and $\mathrm{link}_\Gamma\, p$.

Queries on a cell complex are often expressed in terms of the *topological relations* defined by the adjacencies and incidences of its cells. Let Γ be a cell complex of dimension d and let p be a i-cell of Γ.
We denote as

- *boundary relation* $R_{i,j}(p)$, with $0 \leq j < i$, the set of j-cells in Γ that are faces of p;
- *coboundary relation* $R_{i,j}(p)$, with $i < j \leq d$, the set of j-cells in Γ that are cofaces of p;
- *adjacency relation* $R_{i,i}(p)$ the set of i-cells in Γ that are adjacent to p.

In the following, we will often denote as *immediate* the boundary relation $R_{i,i-1}(p)$ and the coboundary relation $R_{i,i+1}(p)$.

1.1.2.2 Regular grids

Regular grids represent a specific subclass of cell complexes widely used for regularly distributed data, such as digital images [ČDMI14, KMM04].

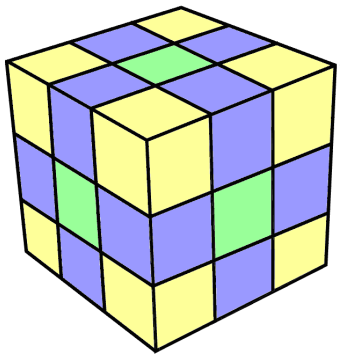


Figure 1.5: An example of 3-dimensional regular grid.

Definition 1.16. An *axis-parallel k-dimensional hyper-cube* η in $\mathbb{R}^n$ is the Cartesian product of n closed intervals, where exactly k of them are non-degenerate with equal length, i.e., $\eta = \{(x_1, \ldots, x_n) \in \mathbb{R}^n \,|\, x_i \in [a_i, b_i]\}$, where $\#\{i \,|\, a_i < b_i\} = k$ and, for such i, $b_i - a_i$ is constant. We say that hyper-cube η is generated by intervals $[a_i, b_i]$.

Usually, intervals have integer endpoints and unit length, i.e., $a_i \in \mathbb{Z}$, and $b_i = a_i$ or $b_i = a_i + 1$. Given a hyper-cube η, generated by intervals $[a_i, b_i]$, $i = 1, \ldots n$, any hyper-cube η' generated by intervals $[a'_i, b'_i]$, with either $a'_i = a_i$ and $b'_i = b_i$, or $a'_i = b'_i = a_i$, or $a'_i = b'_i = b_i$, is called a *face* of η. Hyper-cube η' is a *proper face* of η if $\eta' \neq \eta$.

Definition 1.17. A *regular (hyper-cubic) grid* in $\mathbb{R}^n$ is a finite collection H of hyper-cubes of different dimensions, such that:

- for any hyper-cube $\eta \in H$, all hyper-cubes that are proper faces of η are in H;
- for any pair of hyper-cubes $\eta_1, \eta_2 \in H$, either $\eta_1 \cap \eta_2 = \emptyset$, or $\eta_1 \cap \eta_2$ is a hyper-cube of H;

and the domain of H is a hyper-cube in $\mathbb{R}^n$.

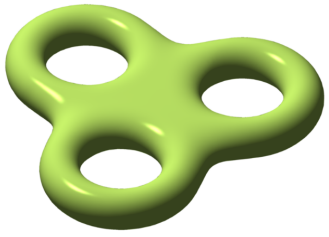
A 2D regular grid is also called a *square grid*, and a 3D regular grid a *cubic grid*.

1.2 Simplicial and persistent homology

For an object discretized through a complex, our interest is in defining a topological invariant which can describe the shape of the complex itself. Homology and persistent homology are powerful tools in shape analysis, since they provide invariants for shape description and characterization. These notions can be defined for any discretization of a geometric shape such as cell complexes, simplicial complexes or regular grids. For the sake of simplicity, in Subsection 1.2.1 and Subsection 1.2.2, we will introduce the notions of homology and persistent homology just for a simplicial complex. We refer to [Mas91, Hat02] for the cellular case.

1.2.1 Simplicial homology

Homology is a fundamental tool in algebraic topology. As shown in Figure 1.6, intuitively homology provides global quantitative and qualitative information about a shape, such as the number of its connected components, the number of holes and tunnels.



$$H_k(\Sigma) = \begin{cases} \mathbb{Z} & \text{if } k = 0 \\ \mathbb{Z}^6 & \text{if } k = 1 \\ \mathbb{Z} & \text{if } k = 2 \end{cases}$$

Figure 1.6: A triple torus and its homology groups. The exponent of each group has a geometrical meaning. The exponent 1 in the 0^{th} homology groups implies that the shape is path-connected, the first homology group $\mathbb{Z}^6$ describes that the triple torus has six independent non-bounding cycles, finally, the exponent 1 of the second homology group tells us that the surface surrounds a void.

In order to formally define the notion of simplicial homology, we need to introduce *chain complexes* [Rot70], which are mathematical structures able to encode in an algebraic structure the topological nature of a shape.

1.2.1.1 Chain complexes and homology

Definition 1.18. A *chain complex* C_* is a pair $(C_k, d_k)_{k\in\mathbb{Z}}$

$$\cdots \longrightarrow C_{k+1} \xrightarrow{d_{k+1}} C_k \xrightarrow{d_k} C_{k-1} \longrightarrow \cdots$$

where C_k are Abelian groups and d_k group homomorphisms such that $d_k\, d_{k+1} = 0$ for all k.
This latter condition is equivalent to requiring $\operatorname{Im} d_{k+1} \subseteq \ker d_k$. We will call C_k the *chain group* of degree k and d_k the *differential map*.

A chain map is a function which relates two chain complexes and preserves their algebraic structure.

Definition 1.19. Let $A_* = (A_k, d_k^A)_{k\in\mathbb{Z}}$ and $B_* = (B_k, d_k^B)_{k\in\mathbb{Z}}$ be two chain complexes. A *chain map* $f : A_* \to B_*$ is a collection of homomorphisms of groups $f = (f_k : A_k \to B_k)_{k\in\mathbb{Z}}$ satisfying for every k the differential condition $f_{k-1}d_k^A = d_k^B f_k$. In other words, it is required that the following diagrams commutes.

$$\begin{array}{ccc} A_k & \xrightarrow{d_k^A} & A_{k-1} \\ {\scriptstyle f_k}\downarrow & & \downarrow{\scriptstyle f_{k-1}} \\ B_k & \xrightarrow[d_k^B]{} & B_{k-1} \end{array}$$

Now, we introduce the notion of homology for a chain complex.

Definition 1.20. Let C_* be a chain complex.
We denote:

- $Z_k(C_*) := \ker d_k \subseteq C_k$ the group of *k-cycles* of C_*;
- $B_k(C_*) := \operatorname{Im} d_{k+1} \subseteq C_k$ the group of *k-boundaries* of C_*.

The condition of chain complex allows us to define

$$H_k(C_*) := Z_k(C_*)/B_k(C_*)$$

called the k^{th} *homology group* of C_*.

Definition 1.21. A chain complex $C_* = (C_k, d_k)_{k\in\mathbb{Z}}$ is called *exact* if, for each $k \in \mathbb{Z}$, $\operatorname{Im} d_{k+1} = \ker d_k$.

From a mathematical point of view, the homology groups of C_* therefore measure "how far" the chain complex C_* is from being exact.

1.2.1.2 Homology groups of simplicial complexes

The goal of this subsection is to associate with each simplicial complex a non-negative chain complex (i.e., all its Abelian groups of degree $k < 0$ are null) in order to compute its homology.

Definition 1.22. Let σ be a simplex. Two orderings of its vertex set are *equivalent* if they differ by an even permutation. If $\dim \sigma > 0$, the orderings of the vertices of σ fall into two equivalence classes. Each of these classes is called an *orientation of* σ.[1] An *oriented simplex* is a simplex σ together with an orientation of σ.

If points $v_0, ..., v_k$ are geometrically independent, we shall use the symbol $v_0 \cdots v_k$ to denote the simplex they span, and the symbol $[v_0, ..., v_k]$ to denote the oriented simplex consisting of simplex $v_0 \cdots v_k$ and the equivalence class of the specific ordering $(v_0, ..., v_k)$. When clear from the context, we will use a single symbol such as σ to denote either a simplex or an oriented simplex. By abuse of notation, if σ and σ' are opposite orientation of the same simplex, we often write $\sigma' = -\sigma$.

Definition 1.23. Let Σ be a simplicial complex. Having selected an orientation for each k-simplex σ of Σ, we denote as $C_k(\Sigma)$ the free Abelian group with basis the set of oriented k-simplices of Σ and is called the *group of (oriented) k-chains of* Σ. If $k < 0$ or $k > \dim \Sigma$, $C_k(\Sigma)$ denotes the trivial group.

In other words, each element c of $C_k(\Sigma)$ is called a *k-chain* of Σ, and it is a finite linear combination with integer coefficients of oriented k-simplices, i.e.,

$$c = \sum_i n_i \sigma_i$$

where $n_i \in \mathbb{Z}$ and σ_i is an oriented k-simplex of Σ.

Group $C_0(\Sigma)$ differs from the others, since it has a natural basis (because a 0-simplex has only one orientation). Group $C_k(\Sigma)$, with $k > 0$, has no natural basis; one must orient the k-simplices of Σ in some arbitrary fashion in order to obtain a basis.

Definition 1.24. If $\sigma = [v_0, ..., v_k]$ is an oriented simplex with $k > 0$, a homomorphism

$$\partial_k : C_k(\Sigma) \to C_{k-1}(\Sigma)$$

called the *boundary operator (or, boundary map)* is defined as

$$\partial_k(\sigma) = \partial_k[v_0, ..., v_k] = \sum_{i=0}^{k} (-1)^i \, [v_0, ..., \hat{v_i}, ..., v_k], \tag{1.24.1}$$

where $\hat{v_i}$ means that vertex v_i is not present. Since $C_k(\Sigma)$ is the trivial group for $k < 0$, operator ∂_k is the trivial homomorphism for $k \leq 0$.

It is easy to prove the following proposition.

[1] If σ is a 0-simplex, then there is only one class and hence only one orientation.

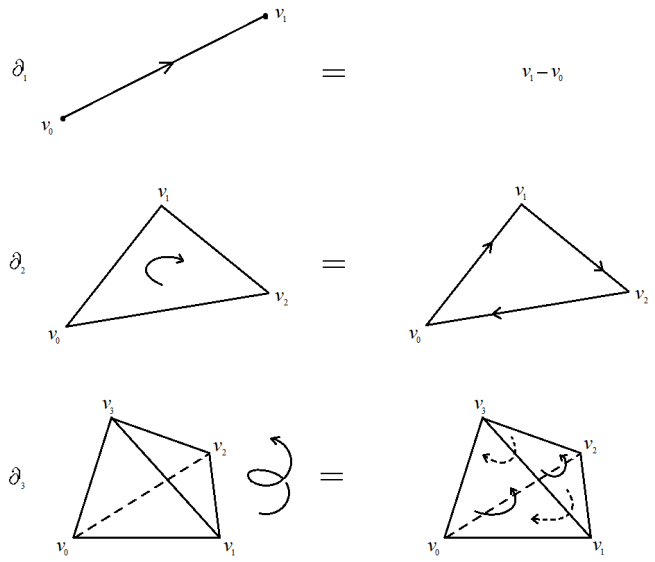


Figure 1.7: Three examples that intuitively show how the boundary operator acts.

Proposition 1.25. *The chain map ∂_* is well-defined and, for each $k > 0$, $\partial_k \, \partial_{k+1} = 0$.*

In accordance with the definitions given in Subsection 1.2.1.1, the construction of $C_k(\Sigma)$ and Proposition 1.25 allow us to associate chain complex $C_*(\Sigma) = (C_k(\Sigma), \partial_k)_{k \in \mathbb{N}}$ with simplicial complex Σ. By attaching a chain complex to a simplicial complex, we can define the homology of a simplicial complex.

Definition 1.26. Let Σ be a simplicial complex and let $C_*(\Sigma) = (C_k(\Sigma), \partial_k)_{k \in \mathbb{N}}$ be the chain complex associated with Σ. Denote by:

- $Z_k(\Sigma) := Z_k(C_*(\Sigma)) = \ker \partial_k \subseteq C_k(\Sigma)$ the group of *k-cycles* of Σ;
- $B_k(\Sigma) := B_k(C_*(\Sigma)) = \operatorname{Im} \partial_{k+1} \subseteq C_k(\Sigma)$ the group of *k-boundaries* of Σ;
- $H_k(\Sigma) := H_k(C_*(\Sigma)) = Z_k(\Sigma)/B_k(\Sigma)$ the *k^{th} simplicial homology group* of Σ.

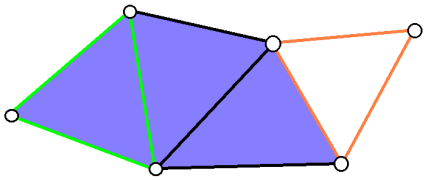


Figure 1.8: A simplicial complex with two highlighted 1-cycles. The green one is also a boundary and so it is null in homology. The orange one instead is not a boundary and provides a non-null contribution in the first homology. This is in accord with the intuitive idea that homology detects holes.

The following theorem allows us to better characterize the homology groups of a finite simplicial complex.

Theorem 1.27. (Structure for finitely generated Abelian groups, [Art91]). *For G a finitely generated Abelian group,*

$$G \cong \mathbb{Z}^s \oplus \bigoplus_{i=1}^{p} \mathbb{Z}_{\lambda_i}$$

where $s \geq 0$, λ_i integers greater than 1 s.t., for $i = 1, ..., p-1$, $\lambda_{i+1} \mid \lambda_i$.
Furthermore the numbers $s, \lambda_1, \ldots, \lambda_p$ are uniquely determined by G.

If Σ is finite, then, by Theorem 1.27, the homology groups can be expressed as

$$H_k(\Sigma) \cong \mathbb{Z}^{\beta_k} < c_1, \cdots, c_{\beta_k} > \oplus \mathbb{Z}_{\lambda_1} < c'_1 > \oplus \cdots \oplus \mathbb{Z}_{\lambda_{p_k}} < c'_{p_k} >$$

with $\lambda_{i+1} \mid \lambda_i$. We call β_k the k^{th} *Betti number of* Σ, $\bigoplus_{i=1}^{p_k} \mathbb{Z}_{\lambda_i}$ the *torsion part of* $H_k(\Sigma)$ and $c_1, \cdots, c_{\beta_k}, c'_1, \cdots, c'_{p_k}$ the *generators of* $H_k(\Sigma)$.

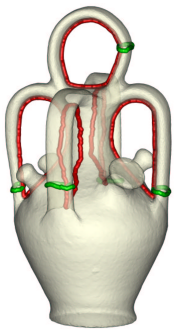


Figure 1.9: A simplicial complex Σ in which a set of generators of $H_1(\Sigma)$ has been highlighted. The number of generators of $H_1(\Sigma)$ coincides with $\beta_1 = 10$.

The homology of a simplicial complex Σ has also a geometric meaning. It is one of the most important tools to obtain topological invariants. Homeomorphic topological spaces have the same homology, while the converse is false in general. For instance, there is a topological space, called *Poincaré homology sphere*, which is not homeomorphic to the sphere but it has the same homology. In spite of this limit, homology provides many global quantitative and qualitative information about a shape. For each k, the k^{th} Betti number β_k measures the number of independent, non-bounding, k-cycles in Σ. In dimension 0, the Betti number counts the number connected components of the complex, in dimension 1, its tunnels and its holes, in dimension 2, the shells surrounding voids or cavities, and so on (see, for instance, Figure 1.6).

The torsion part of a simplicial complex Σ has not such a clear geometric meaning. Intuitively, the presence of torsion reveals that the shape has a "twisted" behavior (see Figure 1.10 for an example). In spite of this, the torsion part is relevant only for shapes embedded in a high-dimensional space [MTCW10]. It can be proven (see [AH35], Chapter X) that, for simplicial complexes embeddable in $\mathbb{R}^3$, each homology group is free and, thus, the torsion part is null.

Given a simplicial complex Σ and an arbitrary Abelian group G, we can consider the k^{th} *homology group with coefficients in* G of Σ as $H_k(\Sigma; G) := H_k(C_*(\Sigma) \otimes_{\mathbb{Z}} G)$, where

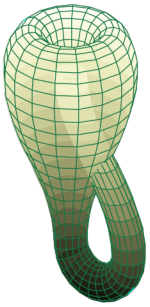


Figure 1.10: A Klein bottle Σ forcedly embedded in $\mathbb{R}^3$. Its homology groups are $H_0(\Sigma) = \mathbb{Z}$, $H_1(\Sigma) = \mathbb{Z} \oplus \mathbb{Z}_2$ and $H_2(\Sigma) = \mathbb{Z}$.

$\otimes_{\mathbb{Z}}$ denotes the tensor product of Abelian groups [Hat02]. If we take as G a field $\mathbb{F}$, $C_*(\Sigma) \otimes_{\mathbb{Z}} \mathbb{F}$ is the chain complex whose chain groups $C_k(\Sigma) \otimes_{\mathbb{Z}} \mathbb{F}$ are just the vector spaces over $\mathbb{F}$ generated by the k-simplices of Σ and so, for each k, $H_k(\Sigma; \mathbb{F}) \cong \mathbb{F}^{\beta_k}$ and, in particular, it has no torsion part. Moreover, taking $\mathbb{F} = \mathbb{Z}_2$, the homomorphisms $\partial_k \otimes_{\mathbb{Z}} \mathbb{Z}_2$ are the boundary maps ∂_k of Σ considered modulo 2. As mentioned above, since up to embedding dimension 3 homology groups with coefficients in $\mathbb{Z}$ are free of the torsion part, homology with integer coefficient and homology with coefficients in $\mathbb{Z}_2$ are equivalent for simplicial complexes embeddable in $\mathbb{R}^3$.

Simplicial homology is not the unique homology theory. The notion of homology can be defined for any topological space through the singular homology theory. A topological space $|\Sigma|$ can be implicitly associated with a simplicial complex Σ. So, for a simplicial complex we can define both simplicial homology and singular homology. Fortunately, it can be proven (see [Hat02]) that the two homological theories are equivalent for simplicial complexes. This result also proves that simplicial homology is well-defined and it does not depend on the orientation given to the simplices of a simplicial complex. Furthermore, this ensures that simplicial homology gives topological invariants, i.e., homeomorphic simplicial complexes have the same (up to isomorphism) homology groups. In spite of this, homology does not completely characterize the shape of a simplicial complex. It is not true, in general, that simplicial complexes with isomorphic homology groups are homeomorphic, or just homotopy equivalent.

Another topological invariant for simplicial complexes is the *Euler characteristic*. Although it represents a weaker topological invariant with respect to the homology, the Euler characteristic of a simplicial complex can be immediately retrieved.

Definition 1.28. Let Σ be a d-dimensional simplicial complex and let n_k be the number of k-simplices in Σ. We define as *Euler characteristic* of Σ the number

$$\chi(\Sigma) := \sum_{k=0}^{d} (-1)^k \, n_k$$

The following result shows that $\chi(\Sigma)$ can be defined purely in terms of homology.

Proposition 1.29. (Euler-Poincaré formula, [Hat02]). *Let Σ be a d-dimensional simplicial complex and let β_k be the k^{th} Betti number of Σ. We have*

$$\chi(\Sigma) = \sum_{k=0}^{d} (-1)^k \beta_k$$

Since homology is a topological invariant, the Euler-Poincaré formula ensures that the Euler characteristic is a topological invariant too. Analogously to the homology, the notion of Euler characteristic and the Euler-Poincaré formula can be immediately generalized to the framework of the cell complexes.

Reduced and relative homology

In the remainder of the work, we will make use of the notion of *reduced* and *relative homology*.

In order to define reduced homology of a given simplicial complex Σ, we have to introduce the *augmented chain complex* of Σ denoted as $\tilde{C}_*(\Sigma)$. It consists of the following chain complex

$$\cdots \xrightarrow{\partial_{k+2}} C_{k+1}(\Sigma) \xrightarrow{\partial_{k+1}} C_k(\Sigma) \xrightarrow{\partial_k} \cdots \xrightarrow{\partial_2} C_1(\Sigma) \xrightarrow{\partial_1} C_0(\Sigma) \xrightarrow{\epsilon} \mathbb{Z} \longrightarrow 0$$

where $\epsilon\big(\sum_i n_i v_i\big) = \sum_i n_i$, for each $n_i \in \mathbb{Z}$ and each v_i vertex of Σ.

Definition 1.30. We define the k^{th} *reduced homology group* of Σ as the k^{th} homology group $\tilde{H}_k(\Sigma) := H_k(\tilde{C}_*(\Sigma))$ of the augmented chain complex $\tilde{C}_*(\Sigma)$.

The knowledge of the reduced homology of a simplicial complex Σ allows to easily retrieve standard homology of Σ, and the converse is still true. It is immediate to show that

$$H_k(\Sigma) \cong \begin{cases} \tilde{H}_k(\Sigma) \oplus \mathbb{Z} & \text{if } k = 0 \\ \tilde{H}_k(\Sigma) & \text{otherwise} \end{cases}$$

Intuitively, the reduced homology of a simplicial complex Σ can be considered as the homology of Σ by regarding the empty set $\emptyset$ as a simplex in Σ of dimension -1.

Let Σ be a simplicial complex, Σ' be a subcomplex of Σ. For each k, we denote with $C_k(\Sigma, \Sigma')$ the quotient group $C_k(\Sigma)/C_k(\Sigma')$. Thus, k-chains of Σ' are trivial in $C_k(\Sigma, \Sigma')$. Since the boundary map $\partial_k : C_k(\Sigma) \to C_{k-1}(\Sigma)$ takes $C_k(\Sigma')$ to $C_{k-1}(\Sigma'))$, it induces a quotient boundary map $\partial_k : C_k(\Sigma, \Sigma') \to C_{k-1}(\Sigma, \Sigma')$. Relation $\partial_{k-1}\partial_k = 0$ holds for these boundary maps, since it holds before passing to quotient groups. So, $C_k(\Sigma, \Sigma')$ is a chain complex.

Definition 1.31. According to the above notation, the k^{th} homology group of the chain complex $C_k(\Sigma, \Sigma')$ is called the k^{th} *relative homology group* $H_k(\Sigma, \Sigma')$ of the pair (Σ, Σ').

Roughly speaking, relative homology groups of the pair (Σ, Σ') represent the homology groups of the complex obtained considering Σ and collapsing to a single point its subcomplex Σ'.

1.2.1.3 Simplicial homology computation through Smith Normal Form reduction

For an algorithmic computation of simplicial homology, we have to consider only *finite* simplicial complexes. In this context, we can describe the boundary maps ∂_k through matrices, called *boundary matrices*. The classical way to compute homology is to reduce the boundary matrices by using an algorithm similar to the Gauss reduction, called *Smith Normal Form (SNF) reduction* [EH10, Mun84, Ago05]. This reduction algorithm allows computing homology with coefficients in any principal ideal domain (PID) [Art91]. For the sake of brevity, in the following, we just consider the case of integer coefficients.

The goal of the Smith Normal Form reduction is to compute directly the quotient between the subgroups $\ker \partial_k$ and $\operatorname{Im} \partial_{k+1}$ that defines the homology:

$$H_k(\Sigma) = \ker \partial_k / \operatorname{Im} \partial_{k+1}$$

In order to compute $H_k(\Sigma)$, we need to be able to determine the two subgroups and also to perform the quotient between them.

Smith Normal Form reduction
Smith Normal Form reduction is an algorithm which allows to reduce in a canonical form any matrix, not necessary a square one, with entries in $\mathbb{Z}$ or in any other PID. Given such a matrix, it is always possible by a sort of similarity transformation to reduce it in the following form:

$$N = \begin{pmatrix} \mathrm{Id} & 0 & 0 \\ 0 & \boldsymbol{\lambda} & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

where

$$\boldsymbol{\lambda} = \begin{pmatrix} \lambda_1 & 0 & 0 & 0 \\ 0 & \lambda_2 & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & \lambda_p \end{pmatrix}$$

is a diagonal matrix with $\lambda_1, ..., \lambda_p$ in the diagonal s.t. $\lambda_i > 1$ and $\lambda_{i+1} | \lambda_i$. N is said to be in *Smith Normal Form*.

A similar form can be obtained by modifying rows and columns of the matrix. However, attention must be paid to perform "legal" modifications, i.e., changes that can be undone in our coefficient space $\mathbb{Z}$. For example, a multiplication of the elements of a row by 2 is illegal, since to undo it the row should be divided by 2 but 1/2$\notin \mathbb{Z}$. Hence, the elementary rows operations are:

- exchange row i and j,
- multiply a row i by -1,

- replace a row i by row i + q row j, where $q \in \mathbb{Z}$ and $i \neq j$.

The same elementary operations are defined for the columns of the matrix.

The algorithm starts finding a non null entry of the matrix, moving it on the upper left corner and improving it until it divides all entries of its row and its column. Thus, it is possible to zero out the rest of the first row and first column and recursively apply the same procedure to the submatrix obtained by removing the first column and first row.

Computing homology through Smith Normal Form reduction
Smith Normal reduction can be used to retrieve homology groups of an arbitrary simplicial complex Σ. The first step for computing the homology groups of Σ is to express each boundary map ∂_k through the matrix D_k. Suppose we have chosen, for each k, a basis of $C_k(\Sigma)$ given by $\{\sigma_1^k, ..., \sigma_{n_k}^k\}$. Since the boundary operator is completely determined by the image elements of the canonical basis, map ∂_k can be encoded as the matrix:

$$D_k = \begin{array}{c} \\ \sigma_1^{k-1} \\ \vdots \\ \sigma_{n_{k-1}}^{k-1} \end{array} \begin{array}{c} \begin{array}{ccc} \sigma_1^k & \cdots & \sigma_{n_k}^k \end{array} \\ \begin{pmatrix} & \vdots & \\ \cdots & \eta_{j,i}^k & \cdots \\ & \vdots & \end{pmatrix} \end{array}$$

where $\eta_{j,i}^k$ represents the coefficient of σ_j^{k-1} in $\partial_k(s_i^k)$.
Then, in order to obtain the k^{th} homology group $H_k(\Sigma)$ of Σ, we have to compute N_{k+1} and N_k representing the Smith Normal form of ∂_{k+1} and of ∂_k, respectively. After these reductions, the homology group $H_k(\Sigma)$ can be easily retrieved. Let q be the number of null columns in N_k, t the rank of the matrix N_{k+1} and p the number of the elements greater than 1 in the diagonal of N_{k+1}. With this notation, the k^{th} homology group of Σ is

$$H_k(\Sigma) \cong \mathbb{Z}^{q-t} \oplus \mathbb{Z}_{\lambda_1} \oplus \cdots \oplus \mathbb{Z}_{\lambda_p}$$

In order to retrieve also the generators of the k^{th} homology group of Σ, we need to modify the bases in which D_k and D_{k+1} are expressed, according to the changes performed during the execution of the Smith Normal Form reduction. The modifications are resumed in Table 1.1.

To easily perform the quotient between $\ker \partial_k$ and $\operatorname{Im} \partial_{k+1}$, we need that the elements representing a basis of $\operatorname{Im} \partial_{k+1}$ are a subcollection of the elements of the basis obtained for $\ker \partial_k$. In order to reach this purpose, we can adopt the following strategy.
First of all, matrix D_{k+1}, expressed in terms of the canonical bases, is reduced to N_{k+1}. Then, we represent boundary map ∂_k through the matrix expressed by choosing as basis of $C_k(\Sigma)$ the one obtained in N_{k+1} and as basis of $C_{k-1}(\Sigma)$ the canonical one. Finally, we reduce this matrix in its Smith normal form and we are able to easily perform the quotient $\ker \partial_k / \operatorname{Im} \partial_{k+1}$ obtaining its generators.

Example 1.32. Let Σ be the simplicial complex of dimension 2 represented in Figure 1.11. We want to compute its homology groups.

Operation on the matrix	**Operation on the $(k-1)$-basis**	**Operation on the k-basis**
$row_i \leftrightarrow row_j$	$\sigma_i^{k-1} \leftrightarrow \sigma_j^{k-1}$	-
$row_i \leftarrow -row_i$	$\sigma_i^{k-1} \leftarrow -\sigma_i^{k-1}$	-
$row_i \leftarrow row_i + q\, row_j$	$\sigma_j^{k-1} \leftarrow \sigma_j^{k-1} - q\,\sigma_i^{k-1}$	-
$col_i \leftrightarrow col_j$	-	$\sigma_i^k \leftrightarrow \sigma_j^k$
$col_i \leftarrow -col_i$	-	$\sigma_i^k \leftarrow -\sigma_i^k$
$col_i \leftarrow col_i + q\, col_j$	-	$\sigma_i^k \leftarrow \sigma_i^k + q\,\sigma_j^k$

Table 1.1: Modifications of $(k-1)$- and k-bases in which matrix D_k is expressed required by the operations performed during the SNF reduction.

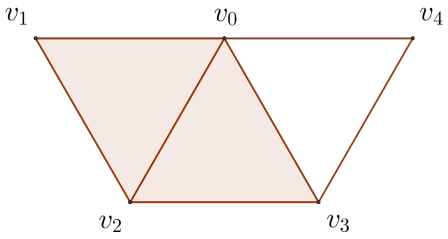


Figure 1.11: The simplicial complex Σ used in Example 1.32.

Let us start with $H_2(\Sigma)$. Since ∂_3 is the null map, we have that $\mathrm{Im}\,\partial_3$ is equal to the trivial group 0. Let us consider the matrix D_2 encoding the boundary map ∂_2 in terms of the 2-simplices of Σ which represent the canonical basis of $C_2(\Sigma)$.

$$D_2 = \begin{array}{c} \\ [v_0,v_1] \\ [v_0,v_2] \\ [v_0,v_3] \\ [v_0,v_4] \\ [v_1,v_2] \\ [v_2,v_3] \\ [v_3,v_4] \end{array} \begin{array}{c} \begin{array}{cc} [v_0,v_1,v_2] & [v_0,v_2,v_3] \end{array} \\ \begin{pmatrix} 1 & 0 \\ -1 & 1 \\ 0 & -1 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix} \end{array}$$

Then, we reduce D_2 to its Smith Normal form N_2, keeping track of the changes performed on the bases.

$$N_2 = \begin{array}{c} \\ [v_1,v_2]-[v_0,v_2]+[v_0,v_1] \\ [v_2,v_3]-[v_0,v_3]+[v_0,v_2] \\ [v_0,v_3] \\ [v_0,v_4] \\ [v_1,v_2] \\ [v_2,v_3] \\ [v_3,v_4] \end{array} \begin{array}{c} \begin{array}{cc} [v_0,v_1,v_2] & [v_0,v_2,v_3] \end{array} \\ \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix} \end{array}$$

Since N_2 has no null columns, $\ker \partial_2 = 0$. So, $H_2(\Sigma) = \ker \partial_2 / \operatorname{Im} \partial_3 = 0$.

To compute the first homology group of Σ, we have to find a basis for $\ker \partial_1$ which includes the elements of the basis of $\operatorname{Im} \partial_2$. To reach this goal, we express the boundary maps ∂_1 in terms of the just obtained basis of $C_1(\Sigma)$ and of the canonical basis of $C_0(\Sigma)$.

$$D_1' = \begin{array}{c} \\ v_0 \\ v_1 \\ v_2 \\ v_3 \\ v_4 \end{array} \begin{array}{c} \begin{array}{ccccccc} [v_0,v_3] & [v_0,v_4] & [v_1,v_2] & [v_2,v_3] & [v_3,v_4] & {\scriptstyle [v_1,v_2]-[v_0,v_2]+[v_0,v_1]} & {\scriptstyle [v_2,v_3]-[v_0,v_3]+[v_0,v_2]} \end{array} \\ \begin{pmatrix} -1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \end{array}$$

Performing the Smith Normal Form reduction of D_1', we obtain N_1.

$$N_1 = \begin{array}{c} \\ v_0 - v_3 \\ v_1 - v_2 \\ v_2 - v_3 \\ v_3 - v_4 \\ v_4 \end{array} \begin{array}{c} \begin{array}{ccccccc} {\scriptstyle -[v_0,v_3]} & {\scriptstyle -[v_1,v_2]} & {\scriptstyle -[v_2,v_3]} & {\scriptstyle [v_0,v_3]-[v_0,v_4]} & {\scriptstyle [v_3,v_4]-[v_0,v_4]+[v_0,v_3]} & {\scriptstyle [v_1,v_2]-[v_0,v_2]+[v_0,v_1]} & {\scriptstyle [v_2,v_3]-[v_0,v_3]+[v_0,v_2]} \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

Now, we are ready to retrieve $H_1(\Sigma)$.
We can easily read from N_1 and N_2 that $\{[v_3,v_4] - [v_0,v_4] + [v_0,v_3], [v_1,v_2] - [v_0,v_2] + [v_0,v_1], [v_2,v_3] - [v_0,v_3] + [v_0,v_2]\}$ and $\{[v_1,v_2] - [v_0,v_2] + [v_0,v_1], [v_2,v_3] - [v_0,v_3] + [v_0,v_2]\}$ represent a basis for $\ker \partial_1$ and for $\operatorname{Im} \partial_2$, respectively.
So, $H_1(\Sigma) = \ker \partial_1 / \operatorname{Im} \partial_2 = \mathbb{Z}$ and it is generated by $[v_3,v_4] - [v_0,v_4] + [v_0,v_3]$.

Let us conclude with $H_0(\Sigma)$.
As showed by N_1, $\{v_0 - v_3, v_1 - v_2, v_2 - v_3, v_3 - v_4\}$ represents a basis for $\operatorname{Im} \partial_1$. Furthermore, since the boundary map ∂_0 is the null map, the basis $\{v_0 - v_3, v_1 - v_2, v_2 - v_3, v_3 - v_4, v_4\}$ of $C_0(\Sigma)$ can be considered as a basis for $\ker \partial_0$.
So, $H_0(\Sigma) = \ker \partial_0 / \operatorname{Im} \partial_1 = \mathbb{Z}$ and it is generated by v_4.

Notice that the retrieved homology groups actually reflect the shape of the simplicial complex Σ allowing to understand that Σ is connected and to reveal the presence of a 1-dimensional hole bounded by the simplices $[v_3,v_4]$, $[v_0,v_4]$, $[v_0,v_3]$.

Smith Normal Form reduction provides the homology groups of any finite simplicial complex but, the time complexity of the SNF algorithm is super-cubical in the number of the simplices of the simplicial complex [Sto96]. Another well-known problem is the appearance of large integers during reduction [HM91]. These limitations lead to consider other methods to compute homology in an efficient way.

1.2.2 Persistent homology

Persistent homology [EH08, Zom05, Ghr08] is an important tool in topological shape analysis, which aims at overcoming intrinsic limitations of classical homology by allowing for a multi-scale approach to shape description. The possibility to retrieve essential topological features of a shape has led to an increasing development of persistent homology in various application domains, such as, for instance, biology and chemistry [CBK09, DAE+08, WAB+05, MTCW10], astrophysics [vdWVE+11], automatic classification of images [BEK10, CIDZ08, CFG06], sensor networks [DG07b, DG07a] and social networks [HMR09, CH13].

Similarly to standard homology, persistent homology can be defined for chain and cell complexes. In spite of this, it is typically introduced for simplicial complexes.

Definition 1.33. Let Σ be a simplicial complex. A *filtration* F of Σ is a finite sequence of subcomplexes $\{\Sigma^m \mid 0 \leq m \leq M\}$ of Σ such that $\emptyset = \Sigma^0 \subseteq \Sigma^1 \subseteq \cdots \subseteq \Sigma^M = \Sigma$.

An example of filtration is depicted in Figure 1.12.

The *associated chain filtration* $F(\Sigma)$ is defined as the following sequence of chain complexes

$$C_*(\Sigma^1) \xrightarrow{i_1} C_*(\Sigma^2) \xrightarrow{i_2} \ldots \xrightarrow{i_{M-1}} C_*(\Sigma^M)$$

where maps i_m arise from inclusion of groups.

For $p \in \mathbb{N}$, we denote as $i_{m,p} : C_*(\Sigma^m) \to C_*(\Sigma^{m+p})$ the composition $i_{m+p-1} \cdot \ldots \cdot i_m$ when it makes sense.

Definition 1.34. The *p-persistent k^{th} homology group of* Σ^m is defined to be

$$H_k^p(\Sigma^m) := \frac{i_{m,p}(Z_k(\Sigma^m))}{i_{m,p}(Z_k(\Sigma^m)) \cap B_k(\Sigma^{m+p})}$$

Informally, $H_k^p(\Sigma^m)$ consists of the k-cycles included from $C_k(\Sigma^m)$ into $C_k(\Sigma^{m+p})$ modulo boundaries.

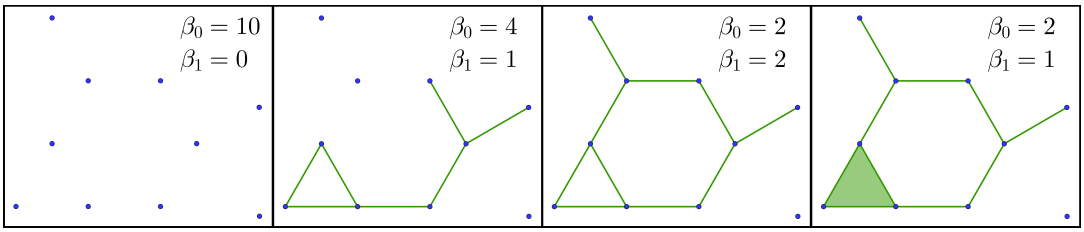


Figure 1.12: An example of filtration F of a simplicial complex Σ. Betti numbers of each simplicial complex of the filtration are reported. Persistent homology captures the changes in the Betti numbers during the filtration.

Persistent homology provides more information about a shape than standard homology. While homology captures cycles in a shape by factoring out the boundary cycles, persistent homology allows the retrieval of cycles that are non-boundary elements in a certain

step of the filtration and that will turn into boundaries in some subsequent step. The persistence of a cycle during the filtration gives quantitative information about the relevance of the cycle itself for the shape. As much as standard homology, persistent homology can be defined with coefficients in any Abelian group. Moreover, persistent homology with coefficients in a PID can be computed using Smith Normal Form reduction [Zom05]. In spite of this, as we will see in Section 2.3, it is usually computed by choosing a field as coefficient group.

Multi-dimensional persistent homology

Recently, a generalization of classical persistent homology to more than one filtering variable, called *multi-dimensional persistent homology*, has been introduced [CZ09, CSZ09]. While persistent homology captures the topology of a one-parameter family of increasing shapes, multi-dimensional persistent homology allows to do it for a family of shapes parameterized along multiple scalar functions.

Multi-dimensional persistent homology is an extension of persistent homology motivated by the fact that data analysis and comparison often involve the examination of properties that are naturally described by multiple scalar values. In the multi-dimensional setting, we have more directions along which the filtration varies, each induced by a different scalar value. We consider a family of subcomplexes $\{\Sigma^m\}$, where now all indexes m are given by several parameters, one for each filtering function. Indexes for the filtration are then vectors rather than scalars. Inclusions of complexes in the multi-dimensional filtration are now required only if two indexes are comparable according to a partial order among the indexes, which is defined componentwise. Any two indexes are comparable if they simultaneously are in all components. By defining the sum of indexes componentwise, we can say that, for an index p whose components are greater than or equal to 0, the *multi-dimensional p-persistent k^{th} homology group $H_k^p(\Sigma^m)$ of* Σ^m consists of the k-cycles included from $C_k(\Sigma^m)$ into $C_k(\Sigma^{m+p})$ modulo boundaries.

For multi-filtrations, the theory of multi-dimensional persistence homology shows that no complete discrete invariant exists, and thus an incomplete invariant, called rank invariant, has been proposed [CZ09]. This latter keeps track of the number of persistence classes surviving from one step of the multi-filtration to another, although it does not determine completely the homology persistence module. For each homology degree k and any pair of comparable indexes $(m, m+p)$, the inclusion of complexes between Σ^m and Σ^{m+p} induces a linear map $i_{m,p}$ between homology modules $H_k(\Sigma^m)$ and $H_k(\Sigma^{m+p})$ the *rank invariant* is the function ρ_k, which associates with any a pair of comparable indexes $(m, m+p)$, the rank of the image of $H_k(\Sigma^m)$ through map $i_{m,p}$ as a submodule of $H_k(\Sigma^{m+p})$.

Another notion to be mentioned is the so-called *localized homology* [CF08]. Similarly to persistent homology, localized homology aims to overcome limitations of the standard homology by distinguish between noise and relevant cycles generating homology. Specifically, its purpose is actually to determine an optimal location of the homology generators allowing to geometrically locate the topological features of a complex and measure them.

1.3 Morse and discrete Morse theory

Morse theory [Mat02, Mil63] studies the relationships between the topology of a shape and the critical points of a real-valued smooth function defined on it. It has been recognized as an important tool for shape analysis and understanding in many applications, including physics, chemistry, medicine and geography. Morse theory is defined for smooth functions, but recently a discrete counterpart has been proposed in an entirely combinatorial setting by Robin Forman for cell complexes [For98, For02]. Discrete Morse theory is the basis for computing Morse decompositions of discretized geometric shapes, and it provides a tool for retrieving topological invariants, like homology and persistent homology groups.

In this section, we introduce Morse theory in the smooth case, we review theoretical tools at the basis of several algorithms in the literature for computing Morse decompositions of shapes, namely piecewise linear Morse theory and watershed transform and, then, we present discrete Morse theory.

1.3.1 Morse theory

As mentioned above, Morse theory [Mat02, Mil63] studies the relationships between the topology of a smooth d-manifold $M \subseteq \mathbb{R}^n$ and the critical points of a C^2-differentiable real-valued function f defined on it.

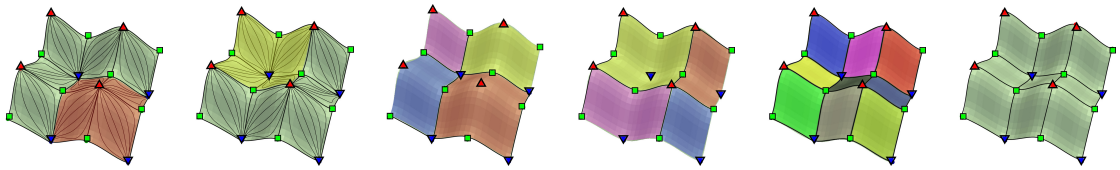
Definition 1.35. Let $f : M \to \mathbb{R}$ be a C^2-differentiable function. A point $p \in M$ is called *critical point* of f if and only if the gradient ∇f of f vanishes on p, i.e., $\nabla f(p) = 0$. Moreover, the critical point p is denoted as *non-degenerate critical point* if the determinant of the Hessian matrix $Hess_p(f)$ of the second-order partial derivatives of f, evaluated in p, is not null.

The number of negative eigenvalues of $Hess_p(f)$ is called the *index* k of a critical point p and p is called a *k-saddle*, with $0 \leq k \leq n$. A 0-saddle is called a *minimum* and a d-saddle a *maximum*. The corresponding eigenvectors define the directions in which function f is decreasing.

Definition 1.36. Under the above assumptions, a function $f : M \to \mathbb{R}$ is called a *Morse function* if all its critical points are not degenerate.

As a consequence of the *Morse Lemma* [Mil63], each non-degenerate critical point is isolated and, therefore, it has a neighborhood which does not contain other critical points.

An *integral line* of a function f is a maximal path everywhere tangent to the gradient of f. An integral line follows the direction in which the function has the maximum growth. An integral line, which connects a critical point p of index k to a critical point q of index $k + 1$, is called a *separatrix line*. The integral lines cover the entire domain of f and they form cells, each corresponding to a critical point.



(a) (b) (c) (d) (e) (f)

Figure 1.13: Red and blue triangles indicate maxima and minima. Green squares indicate saddles. (a) The set of integral lines converging to a maximum and forming the (red) descending cell. (b) The set of integral lines originating from a minimum and forming the (yellow) ascending cell. The set of all the descending and ascending cells forming (c) the descending Morse complex Γ_D and (d) the ascending Morse complex Γ_A. (e) Resulting Morse-Smale complex and (f) 1-skeleton of the Morse-Smale complex.

Integral lines that converge to a critical point p of index k form a k-cell, called the *descending manifold* of p. Integral lines that originate from a critical point p of index k form a $(d-k)$-cell, called the *ascending manifold* of p. The descending [ascending] manifolds are pairwise disjoint and partition the domain of M into cells, which form a cell complex since the boundary of each cell is the union of lower-dimensional cells. The collection of all the descending [ascending] manifolds form the *descending Morse complex* Γ_D [*ascending Morse complex* Γ_A]. The two Morse complexes are mutually dual.

In the 2D case (see Figure 1.13), the integral lines converging to a maximum/saddle/minimum form a 2-cell/1-cell/0-cell of the descending Morse complex, respectively (see Figures 1.13 (a) and (c)). Dually, the integral lines originating from a minimum/saddle/maximum form a 2-cell/1-cell/0-cell of the ascending Morse complex, respectively (see Figures 1.13 (b) and (d)). In 3D, the set of integral lines converging to a maximum/2-saddle/1-saddle/minimum form a 3-cell/ 2-cell/1-cell/0-cell of the descending Morse complex, respectively. The dual situation happens in the ascending Morse complex.

A Morse function f is called a *Morse-Smale function* if and only if the descending and ascending Morse complexes intersect transversally. The connected components of the intersection of the descending and ascending cells decompose M into a *Morse-Smale (MS) complex*, denoted Γ_{MS} . If f is a Morse-Smale function, then there is no integral line connecting two different critical points of f of the same index. Note that, in a d-dimensional MS complex, each 1-saddle is connected to exactly two minima and each $(d-1)$-saddle is connected to exactly two maxima, not necessarily distinct. The *1-skeleton* of the Morse-Smale complex, i.e., is the subcomplex composed of its 0-cells and 1-cells, is often called the *critical net*.
The intersection of the descending and ascending Morse complexes of Figures 1.13 (c) and (d) forms the Morse-Smale complex illustrated in Figure 1.13 (e). For each critical point p, the descending and ascending cells of p intersect only at p. Thus, the 0-cells of the Morse-Smale complex are the critical points. The 1-cells are the separatrix lines connecting pairs of critical points. The distinctive feature of Morse-Smale cells is the uniform gradient flow inside each of them.

1.3.2 Piecewise linear Morse theory and watershed transform

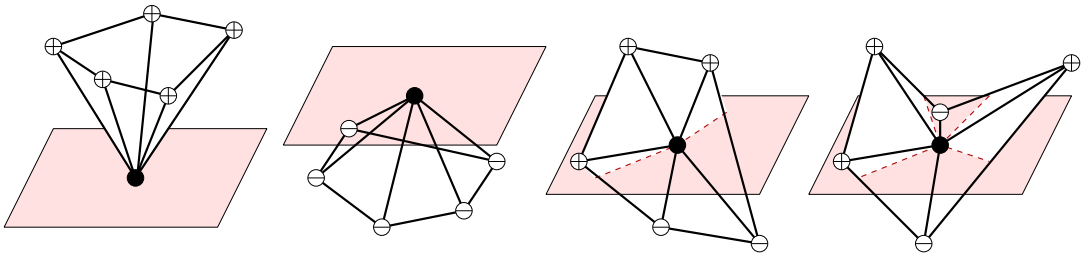
In this section, we discuss two theoretical tool for the actual computation of Morse and Morse-Smale complexes for segmenting shapes discretized as simplicial complexes or regular grids and endowed with a scalar field. Examples are intensity images, terrains, volume data, etc. The approaches discussed here are based on piecewise linear Morse theory and on the watershed transform. It has been shown in [VCY12] that the Morse-Smale complex built on the discretization of a smooth function has not, in general, the same structure as the "true" Morse-Smale complex of the original function.

1.3.2.1 Piecewise linear Morse theory

Piecewise linear Morse theory has been introduced by Banchoff [Ban67, Ban70] to extend the results of smooth Morse theory to polyhedral surfaces, and, thus, triangulated ones (see [BDF$^+$08] for more details).

In this subsection, we denote as triangulated surface the graph of a real-valued function f defined at the vertices of a triangle mesh Σ on the plane, interpolated through linear interpolation. In order to ensure that critical points are isolated, function f is assumed to have different values at every pair of vertices connected by an edge in Σ, i.e., no *flat edges* are allowed [TIKU95].

Let us consider a vertex v, its star $\mathrm{star}_\Sigma\, v$, and the horizontal plane passing through v , as illustrated in Figure 1.14. Vertex v is a local *maximum* or a local *minimum* if the plane does not intersect $\mathrm{star}_\Sigma\, v$; it is *regular* if the plane cuts $\mathrm{star}_\Sigma\, v)$ into two parts; it is a *saddle* if the plane cuts $\mathrm{star}_\Sigma\, v$ into k parts, with $k \geq 4$. The saddle is *simple* if $k = 4$, and *multiple* otherwise. The *multiplicity* of a saddle is equal to $k/2-1$. Note the piecewise linear case includes isolated degenerate critical points, such as multiple saddles, that do not occur in smooth Morse theory.



(a) (b) (c) (d)

Figure 1.14: Classification of a vertex according to Banchoff [Ban67]: (a) minimum, (b) maximum, (c) regular, (d) simple saddle.

The characterization provided by Banchoff [Ban67] correctly detects critical points in dimensions two and three, while a more complete characterization for higher-dimensional spaces is based on Betti numbers (see [EHNP03]).

In [EHZ01, EHNP03], the notion of *Quasi Morse-Smale* (QMS) complex is introduced as a piecewise linear counterpart of the Morse-Smale complex for triangle and tetrahedral

meshes. The idea behind a QMS, called *simulation of differentiability*, is that of extending the smooth notions to the piecewise linear case so as to guarantee that the complex has the same structure of its smooth counterpart. Numerical accuracy is achieved via local transformations that preserve the structure of the complex. In the 2D case, a QMS complex is a quadrangulation of the triangle mesh M. In the QMS complex 0-cells are the critical points of f, the 1-cells connect minima to saddles and maxima to saddles [EHNP03]. In 3D, the 3-cells of the QMS complex are *crystals*, the faces of such crystals are quadrangles with vertices at a minimum, two 1- saddles and a 2-saddle, or at a maximum, two 2-saddles and a 1-saddle. Note that 1-cells and 2-cells of a QMS complex are not necessarily those of maximal ascent/descent, as they are in a Morse-Smale complex.

1.3.2.2 Watershed transform

The watershed transform is an alternative framework to Morse theory. It has been first defined for grey-scale images, and several definitions exist in the discrete case [Mey94, VS91]. The watershed transform has also been defined for a C^2-differentiable function f over a connected domain D, having the property that the gradient ∇f is non-null everywhere except possibly at some isolated points. This includes Morse functions. Basic notions in the watershed transform are *catchment basins* and *watershed lines*, both defined in terms of topographic distance [Mey94, RM00].

Definition 1.37. The *topographic distance* $T_D(p, q)$ between two points p and q belonging to the domain D of f is

$$T_D(p, q) = \inf_P \int_P ||\nabla f(P(s))||ds$$

where P is a smooth path inside D such that $P(0) = p$, $P(1) = q$.

The above definition ensures that the path which minimizes the topographic distance between p and q is the path of steepest slope, if it exists.

Definition 1.38. Let f be a C^2-differentiable functions over a connected domain D and let m_i be a minimum of function f. We define:

- the *catchment basin* $CB(m_i)$ of m_i as the set of points which are closer in terms of topographic distance to m_i than to any other minimum;
- the *watershed* (or *watershed lines*) $WS(f)$ of f as the set of points in D which do not belong to any catchment basin.

When f is a C^2-differentiable Morse function, then the closure of the catchment basins of the minima of f are the closure of the 2-cells of the ascending Morse complex of f, and watershed lines form a subset of separatrix lines that connect saddles to maxima. Symmetrically, the closure of the catchment basins of function $-f$ provides the closure of the 2-cells of the descending Morse complex of f.

The watershed transform in the discrete case uses a discrete version of the topographic distance [Mey94] defined for an undirected labeled graph, $H = (N_H, A_H, f)$, where the nodes in N_H are labeled through function f. The nodes of graph H can be the pixels or the voxels in an image, or the vertices of a cell complex. The arcs can represent the adjacencies of the top cells in a grid or of the 1-cells of a cell complex. The *lower slope* at a node p is the maximal slope linking p to any of its neighbors of lower function value. A cost is associated with the arcs of H defined in terms of the lower slope. Given a path π between two nodes p and q in H, the *π-topographic distance* is given by the sum of costs for traversing all directed arcs composing π. The *topographic distance* $T(p, q)$ between p and q is the minimum of the π-topographic distances along all paths π between p and q.

The topographic distance is actually not a true distance function because it is equal to zero on distinct nodes of H, if they belong to the same plateau. A *plateau* is a connected set of nodes in H having the same function values. The definition of a catchment basin in graph H is similar to the smooth case.
Given a minimum m_i of function f, where m_i can be a single node or a plateau, the *catchment basin* of m_i is defined as

$$CB(m_i) := \{p \in D : f(m_i) + T(p, m_i) < f(m_j) + T(p, m_j),\ \forall j \neq i\}$$

1.3.3 Discrete Morse theory

Discrete Morse theory [For98, For02] is a discrete counterpart of Morse theory, with the main purpose of transposing the results of Morse theory from a smooth to a combinatorial setting. It has been introduced for cell complexes but, in the following, we will just describe it for simplicial complexes for simplicity.

Discrete Morse theory is a powerful tool for handling a scalar field defined on a shape and for providing a more compact complex while it preserves the homological features. This goal is achieved by considering a function defined over all the simplices of a simplicial complex Σ. Given two simplices σ, τ of Σ, we write $\sigma \prec \tau$ if σ is a face of τ and $\dim \tau = \dim \sigma + 1$.

Definition 1.39. A function $f : \Sigma \to \mathbb{R}$ is called a *discrete Morse function* if, for every simplex σ in Σ,

- $c^+(\sigma) := \#\{\tau \succ \sigma \mid f(\tau) \leq f(\sigma)\} \leq 1$,
- $c^-(\sigma) := \#\{\rho \prec \sigma \mid f(\rho) \geq f(\sigma)\} \leq 1$.

Remark 1.40. For a discrete Morse function defined on a cell complex Γ, the further condition is required:

- if p and q are cells of Γ such that p is an irregular face of q, then $f(q) > f(p)$.

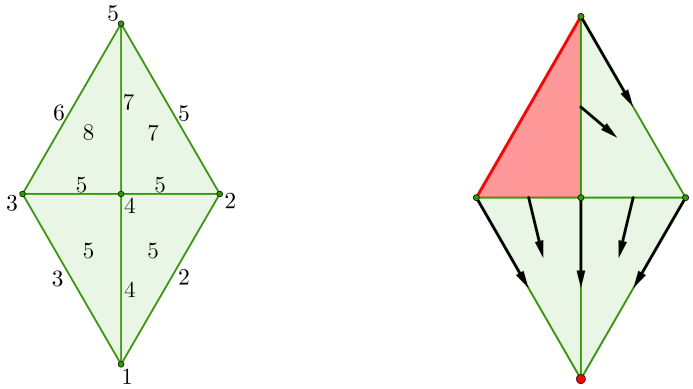
In other words, a discrete Morse function is just a function increasing with respect to the dimension of the simplices which, for each k-simplex, admits at most one exception to the

rule above among the simplices of dimension $k-1$, or among the simplices of dimension $k+1$. It is easy to show (see [For98], Lemma 2.5) that, for a discrete Morse function, $c^+(\sigma)$ and $c^-(\sigma)$ cannot be simultaneously equal to 1.

Definition 1.41. A k-simplex σ in Σ is called *critical simplex of index* k (or, k*-saddle*) if $c^+(\sigma) = c^-(\sigma) = 0$.

A critical simplex of index 0 is called a *minimum* and a critical simplex of index $d = \dim \Sigma$ a *maximum*.

Note that, for every simplicial complex Σ, there exists at least a discrete Morse function on it. Given an k-simplex σ of Σ, we can simply define $f(\sigma)$ as k. However, in this case, each cell is critical. Figure 1.15 (a) shows a discrete Morse function f defined on a simplicial complex. Each simplex is labeled by the value of function f. Vertex 1 is critical (minimum), since f has a higher value on all edges incident to it. Triangle 8 is critical (maximum), since f has a lower value on all edges incident to it. Edge 6 is critical (saddle), since f has a higher value on the incident triangle 8, and lower values on its extreme vertices.



(a) (b)

Figure 1.15: (a) A discrete Morse function on a simplicial complex and (b) the corresponding gradient vector field (red simplices represent critical simplices).

Definition 1.42. A *discrete vector field* V on a simplicial complex Σ is a collection of pairs of simplices $(\sigma, \tau) \in \Sigma \times \Sigma$ such that $\sigma \prec \tau$ and each simplex of Σ is in at most one pair in V.

A discrete Morse function $f : \Sigma \to \mathbb{R}$ induces a discrete vector field $V = \{(\sigma, \tau) \in \Sigma \times \Sigma \,|\, \sigma \prec \tau \text{ and } f(\sigma) \geq f(\tau)\}$ called the *gradient vector field* (or *Forman gradient*) *of* f *on* Σ. Each pair $(\sigma, \tau) \in V$ can be visualized as an arrow from σ to τ.

Definition 1.43. Given a discrete vector field V, a V*-path* (or *gradient path*) is a sequence

$$[(\sigma_1, \tau_1), (\sigma_2, \tau_2), \ldots, (\sigma_r, \tau_r)]$$

of pairs of k-simplices σ_i and $(k+1)$-simplices τ_i, such that $(\sigma_i, \tau_i) \in V$, σ_{i+1} is a face of τ_i, and $\sigma_i \neq \sigma_{i+1}$.

A V-path is a *closed path* if σ_1 is a face of τ_r different from σ_r.

Theorem 1.44. ([For02], Thm. 3.5). *A discrete vector field V is the gradient vector field of a discrete Morse function if and only if there are no closed paths.*

Now, our goal is to build, starting from a discrete Morse function, a chain complex $\mathcal{M}_* = (\mathcal{M}_k, \tilde{\partial}_k)_{k\in\mathbb{N}}$, whose groups $\mathcal{M}_k$ are generated by the critical k-simplices and the boundary maps $\tilde{\partial}_k$ are obtained by following the gradient paths of V.

Definition 1.45. Let σ and τ be two critical simplices of dimension k and $k+1$, respectively. We call *separatrix V-path* between τ and σ each sequence of $(k+1)$- and k-simplices $[\tau, (\sigma_1, \tau_1), (\sigma_2, \tau_2), \ldots, (\sigma_r, \tau_r), \sigma]$ such that all the pairs form a V-path from a face σ_1 of τ to a coface τ_r of σ, or, in case of such a V-path is empty, σ is a face of τ.

Given two critical simplices τ, σ, we define the *multiplicity of the incidences* between τ and σ to be the number $\mu(\tau, \sigma)$ of separatrix V-paths between τ and σ.

In Figure 1.15(b), we have two separatrix V-paths between critical edge 6 and the critical vertex 1. They are identified by following the sequence of (vertex, edge) pairs starting from the boundary of edge 6 and ending on vertex 1. The two separatrices are $[6, (3, 3), 1]$ and $[6, (5, 5), (2, 2), 1]$. The multiplicity of the incidences between edge 6 and vertex 1 is two.

Given a separatrix V-path $\pi = [\tau, (\sigma_1, \tau_1), (\sigma_2, \tau_2), \ldots, (\sigma_r, \tau_r), \sigma]$, we call *multiplicity of* π the number

$$m(\pi) := (-1)^r \frac{\prod_{i=0}^{r} \langle \partial\tau_i, \sigma_{i+1} \rangle}{\prod_{i=1}^{r} \langle \partial\tau_i, \sigma_i \rangle}$$

where $\langle \quad , \quad \rangle$ denotes the scalar product between two chains, τ_0 and σ_{r+1} represent τ and σ, respectively.

Definition 1.46. Given a gradient vector field V on a simplicial complex Σ, the *discrete Morse complex* associated with Σ is the chain complex $\mathcal{M}_* := (\mathcal{M}_k, \tilde{\partial}_k)_{k\in\mathbb{Z}}$ defined by

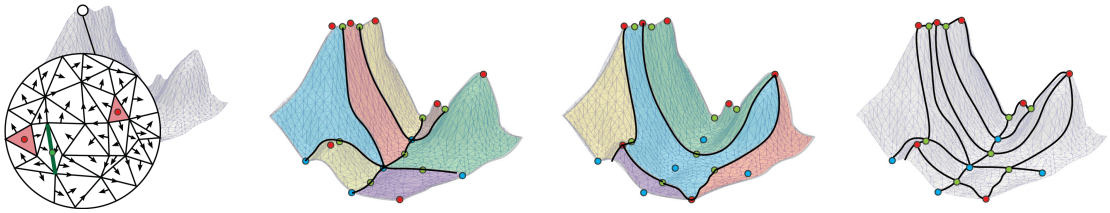
- $\mathcal{M}_k$ is the free Abelian group generated by the critical k-simplices of Σ,
- $\langle \tilde{\partial}\tau, \sigma \rangle := \sum_{\pi\in\Pi(\tau,\sigma)} m(\pi)$, where $\Pi(\tau, \sigma)$ is the set of all separatrix V-paths between τ and σ.

Remark 1.47. By considering $\mathbb{Z}_2$ as coefficient group, the number $\langle \tilde{\partial}\tau, \sigma \rangle$ coincides with the multiplicity of the incidences $\mu(\tau, \sigma)$ between τ and σ.

As already mentioned, the discrete Morse complex $\mathcal{M}_*$ is actually a chain complex and it has the same homological information of the original simplicial complex Σ.

Theorem 1.48. ([For98], Thm. 8.2). *Let G be a Abelian group. $\mathcal{M}_*$ is a chain complex and, for each $k \in \mathbb{N}$,*

$$H_k(\mathcal{M}_*; G) \cong H_k(\Sigma; G)$$



(a) (b) (c) (d)

Figure 1.16: (a) A gradient vector field defined on a simplicial complex. For a terrain dataset, discrete Morse function f corresponds to the height function and its critical points are peaks (red dots), saddles (green dots) and pits (blue dots). (b) Descending Morse complex decomposes the terrain in a collection of 2-cells in one to one correspondence with the peaks while (c) the 2-cells forming the ascending Morse complex are in one-to-one correspondence with the pits. (d) The 1-skeleton of the Morse-Smale complex. The separatrix V-paths for a terrain dataset always connect a saddle with a maximum or a saddle with a minimum.

Analogously to the definitions of the Morse theory (see Subsection 1.3.2), given a gradient vector field V on a simplicial complex Σ (see Figure 1.16(a)), the notions of *descending* and *ascending Morse complexes* are defined based on the behavior of the gradient arrows of V on Σ. Given a critical k-simplex τ, its corresponding descending k-cell is the collection of the k-simplices of Σ belonging to a V-path starting at τ. Dually, its corresponding ascending $(d-k)$-cell is the collection of all the k-simplices belonging to a V-path that converges to τ. The collection of all the descending [ascending] cells form the *descending Morse complex* Σ_D [*ascending Morse complex* Σ_A] (see Figure 1.16(b-c)). The *Morse-Smale (MS) complex* Σ_{MS} consists of the connected components of the intersection of descending and ascending Morse cells. The *1-skeleton* of the Morse-Smale complex Σ_{MS} is the subcomplex of Σ_{MS} composed only of its 0-cells and 1-cells (see Figure 1.16(d)).
Descending, ascending and Morse-Smale complexes can be considered as different representations of the original simplicial complex Σ on which the gradient V is defined. So, since they share the same geometrical realization, they also have the same homology groups. Actually, despite in different context of the literature they assume different names, the notion of descending Morse complex coincides with the notion of discrete Morse complex.

Acyclic matchings
Given a simplicial complex Σ, a gradient vector field V can be built without defining a discrete Morse function on Σ. It is enough to consider an acyclic matching of Σ.

Definition 1.49. Let Σ be a simplicial complex. A *matching of* Σ consists of a partition of Σ into three sets $\mathcal{A}$, $\mathcal{K}$, and $\mathcal{Q}$ along with a bijection $w : \mathcal{Q} \to \mathcal{K}$, such that, for each $\sigma \in \mathcal{Q}$,

$$\langle \partial w(\sigma), \sigma \rangle = \pm 1.$$

We denote such a decomposition as $(\mathcal{A}, w : \mathcal{Q} \to \mathcal{K})$.

Given a matching of Σ, we define by transitive closure a relation $\leq$ on $\mathcal{Q}$ as follows. Let σ, σ' be two distinct elements in $\mathcal{Q}$,

$$\text{if } \langle \partial w(\sigma), \sigma' \rangle \neq 0, \text{ then } \sigma' < \sigma.$$

Definition 1.50. A matching of Σ is *acyclic* if $\leq$ is antisymmetric, and, thus, it defines a partial order on $\mathcal{Q}$.

In order to formalize the relationship among these notions and discrete Morse theory, it is sufficient to note that, if we define $V := \{(\sigma, w(\sigma)) | \sigma \in \mathcal{Q}\}$, the condition on the acyclicity of the matching is equivalent to the requirement that V does not have closed paths. Then, by Theorem 1.44, we are able to conclude that a matching is acyclic if and only if it generates a gradient vector field.

Chapter 2

State of the Art

This chapter is mainly devoted to the description of the context in which our work has been developed.

First, we focus our attention on the data structures used in the literature for encoding simplicial and cell complexes. In Section 2.1, we describe the most relevant data structures representing a simplicial complex of arbitrary dimension by suitably storing a subset of its entities and of their topological relationships. We present an analysis and comparison of such data structures [FID14, FIDon]. In Section 2.2, we give an overview on the literature concerning a specific class of representations, called multi-resolution models, providing an efficient way to encode a cell or simplicial complex and to easily retrieve it at different levels of detail.

Sections 2.3 and 2.4 represent two of the most relevant contributions of the thesis. In these sections, we focus on algorithms for the computation of standard and persistent homology and for the retrieval and simplification of Morse and discrete Morse complexes. An in-depth study and analysis of the literature allows us to provide a formal and complete classifications of these algorithms. Some of the content of Section 2.4 has been published in two survey papers on Morse theory and its applications [DFIM15] and on discrete Morse theory for homology computation [DFI15].

2.1 Data structures for simplicial complexes

Topological data structures allow representing a cell complex by partially encoding its cells and the boundary, coboundary and adjacency relations between them. The main purpose of a topological data structure is to encode a complex balancing the storage cost for representing a subset of the cells and topological relations between them, and the efficiency in retrieving those which are not explicitly represented. The analysis developed in this section takes into account only topological data structures for encoding simplicial complexes of arbitrary dimension. We refer to [DH05, DH07] for a discussion about data structures encoding simplicial complexes of specific dimensions.

In the literature, some topological data structures for encoding a simplicial complex of arbitrary dimension have been proposed. The most common one is the Incidence Graph. An *Incidence Graph (IG)* [Ede87] is a topological graph-based representation of a cell complex which encodes all the cells in the nodes of the graph and their immediate incidence relations in its arcs explicitly. When dealing with data characterized by a huge number of points, or when working in a high-dimensional space, using a data structure which stores all the simplices is unfeasible in practice.
Data structures based on the encoding of only top simplices and vertices have been shown to be particularly effective in two and three dimensions being, in practice, also perfect candidates for extension to higher dimensions. An example is provided by the dimension-independent adjacency-based data structure for encoding arbitrary simplicial complexes called *Generalized Indexed data structure with Adjacencies (IA^*)* [CDW11].
Unlike the IG and IA^* data structures which allow the efficient extraction of all boundary and coboundary relations, the *Simplex Tree* [BM12, BDM13] and the *Skeleton Blocker* [ALS11] have been developed with the purpose of performing a specific operation efficiently.
Recently, a data structure, called *Stellar Tree*, based on a different approach has been proposed [Fel15]. Stellar Tree requires an ambient space in which vertices of the complex are embedded. It can be considered as a spatio-topological data structure for performing efficient topological queries on simplicial and a selected set of non-simplicial complexes.

In our work, we have mainly focused our attention on three topological data structures: the Incidence Graph, the Generalized Indexed data structure with Adjacencies and the Simplex Tree. These three topological data structures are depicted in Figure 2.1 for a simplicial complex Σ.
Further, we briefly describe also the Skeleton Blocker representation and the spatio-topological data structure called Stellar Tree. Finally, we present a comparison among all the described data structures.

The Incidence Graph
Given a simplicial complex Σ, the *Incidence Graph (IG)* is a data structure describing its Hasse diagram [PS03], i.e., the graphical representation of the partially ordered set generated by all the simplices of Σ and their immediate incidence relations. More precisely, the IG is a graph $G = (N, A)$, where

- the nodes in N correspond to the simplices of Σ;
- an arc $a \in A$ connects two nodes of consecutive dimension, if and only if the corresponding simplices σ and τ are mutually incident, i.e., if $\sigma \prec \tau$ or $\tau \prec \sigma$.

Figure 2.1 (b) depicts the Incidence Graph encoding the simplicial complex represented in Figure 2.1 (a).
The *Simplified Incidence Graph (SIG)* [DGH04] and the *Incidence Simplicial (IS)* data structure [DHPC10] are simplified representations of the IG specific for simplicial complexes. Both the SIG and the IS encode all the simplices of a simplicial complex and, for each k-simplex σ, its $(k-1)$-faces and a subset of the simplices in the coboundary

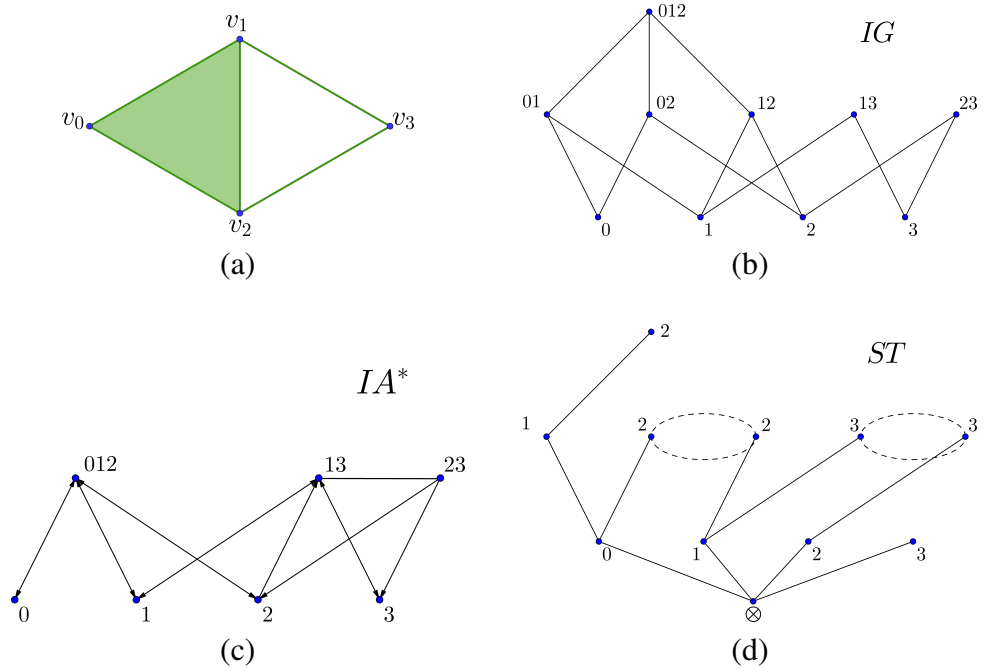


Figure 2.1: A simplicial complex (a) and its representation as the Incidence Graph (b), as the IA^* data structure (c) and as the extended version of Simplex Tree (d).

of σ. The IG, SIG, IS data structures have been all implemented in a public domain library called Mangrove Topological Data Structure (TDS) Library [Can12a, CD13] which is a dimension-independent and extensible framework, targeted to the fast prototyping of topological data structures. In [Can12b], experimental comparisons of these structures have been performed in 2D, 3D and higher dimensions. The IS data structure is as compact as the SIG data structure for arbitrary simplicial 2-complexes, and more compact than the SIG representation for arbitrary simplicial 3-complexes. Both these structures have shown a behavior similar to the IG when working in high dimensions.

The Generalized Indexed data structure with Adjacencies

The *Generalized Indexed data structure with Adjacencies (IA^*)* [CDW11] only encodes the vertices and the top simplices of a simplicial complex Σ and a subset of its adjacent and boundary relations. The IA^* data structure stores the set of vertices and top simplices of Σ plus the following relations:

- for each top k-simplex σ, its vertices;
- for each top k-simplex σ, all top k-simplices sharing with σ a $(k-1)$-face;
- for each $(k-1)$-simplex σ on the boundary of at least two top k-simplices, all top k-simplices on the coboundary of σ;
- for each vertex v, all top 1-simplices in the coboundary of v;
- for each vertex v, one arbitrary selected top k-simplex for each $(k-1)$-connected component of the set of the simplices incident in v.

A subcomplex Σ' of a complex Σ is called $(k-1)$-*connected* if, for any two simplices σ, σ' of Σ', there exists a sequence $(\sigma_0, \sigma_1, \ldots, \sigma_m)$ of k-simplices of Σ' such that any two consecutive simplices σ_i, σ_{i+1} are adjacent, σ is a face of σ_0 and σ' is a face of σ_m.
For a better comparison with the IG, IA^* data structure can be described in graph terminology.
The graph representation of the IA^* data structure is defined as follows. A simplicial complex Σ is encoded by using a graph $G_{IA^*} = (N_{IA^*}, A_{IA^*})$:

- the set N_{IA^*} of the nodes is partitioned in two classes N_{top} and N_0 corresponding to the top simplices and the vertices of Σ, respectively;
- the nodes in N_{top} corresponding to two k-simplices are connected by an arc if they share a common face of dimension $k-1$;
- for each node in N_{top} representing a k-simplex σ, $k+1$ oriented arcs connect it to the nodes in N_0 corresponding to its vertices;
- given a node v in N_0, an oriented arc from it to a node in N_{top} is encoded for each connected component C of the graph consisting of the nodes in N_{top} and the arcs between them and such that there exists at least one arc starting from a node of C and reaching v.

See Figure 2.1 (c) for an example.
A first implementation of the IA^* data structure is available in the public domain software Mangrove TDS Library[Can12a, CD13]. Currently, an optimized implementation is in development.

The Simplex Tree
While the IG and the IA^* representations are complete data structures allowing the efficient retrieval of all the boundary and coboundary relations and the navigation of the encoded simplicial complex Σ, the *Simplex Tree* [BM12] has been originally designed with the task of quickly performing only boundary queries. The Simplex Tree performs this task by storing explicitly all the simplices of Σ through a *trie*.
Chosen a total order among the vertices of Σ, the Simplex Tree encodes the arcs of the Incidence Graph of Σ respecting the lexicographic order of the vertices. More precisely, Simplex Tree stores the simplicial complex Σ in a tree satisfying the following properties:

- the nodes of the Simplex Tree are in bijection with the simplices of the complex and the root is associated with the empty simplex;
- each node of the tree, except for the root, stores the greatest label between the ones associated with the vertices of the corresponding simplex;
- each path from the root to a node corresponds to the simplex σ generated by the vertices corresponding to the labels of the traversed nodes;
- labels are sorted by increasing order along such a path and each label appears exactly once.

Similarly to the IG, the Simplex Tree encodes all the simplices of the represented simplicial complex. The represented relations consist of a specific spanning tree of the Hasse diagram. In order to be able to perform also queries different to the boundary retrieval, an extended version of the Simplex Tree has been proposed [BM12]. This extended version

- encodes a circular list linking all the nodes at the same depth of the simplices whose last vertex, with respect to the lexicographic order, is the same, in order to locate all the instances of a given label in the tree;
- attachs to each set of sibling nodes a pointer to their parent, in order to access a parent in constant time.

In Figure 2.1 (d), the extended version of the Simplex Tree encoding the simplicial complex represented in Figure 2.1 (a) is depicted.

The Skeleton Blocker

The *Skeleton Blocker* data structure [ALS11] has been developed for simplicial complexes close to *flag complexes*, where the flag complex for an undirected graph G is the largest simplicial complex having G as its 1-skeleton. It encodes the 1-skeleton of the complex plus a set of blockers, where a blocker is a simplex which is not contained in the complex but its faces are.
The Skeleton Blocker data structure has been defined specifically to perform a specific operator, called *edge contraction*, on flag complexes, since flag complexes do not have blockers. On complexes of different types the blocker computation is in general difficult, as mentioned in [BM12]. These considerations and our preliminary experiments have led us to do not consider this data structure as a good tool for encoding an arbitrary simplicial complex.

The Stellar Tree

A different method for encoding data embedded in a metric space (the ambient space) is through a spatial index. Recently, a new spatio-topological data structure for triangle and tetrahedral simplicial complexes, based on a point-region quadtree and octree [Sam05] has been proposed [WDFV11]. Differently from other hierarchical space-based spatial indices, the hierarchy is not a spatial index on the complex (i.e., to support efficient spatial queries such as point location), but it is a tool to support efficient retrieval of topological connectivity (i.e., for topological queries), thus encoding a minimum topology and trading space for connectivity. The benefits of such a representation for triangle and tetrahedral simplicial complexes have been shown in the computation of a discrete Morse complexes on triangulated terrains and on unstructured volume datasets [FlDFW14, WIFD13].
Recently, these ideas have been extended to define a new topological data structure called *Stellar Tree* for simplicial complexes and for certain classes of cell complexes based on a hierarchical spatial index [Fel15]. The obtained data structure is a compact representation for such complexes which scales well with both the size and the dimension of the complex, more compact than current simplicial data structures and suitable for performing topological queries efficiently. The spatial index also provides a natural way for organizing the data points based on a metric space, guaranteeing thus a way to distribute the

Dataset	d	$\lvert\Sigma_0\rvert$	$\lvert\Sigma_T\rvert$	$\lvert\Sigma\rvert$	Cost (GB)		
					IA^*	IG	ST
DTI-SCAN	3	0.9M	5.5M	24M	0.97	11.9	2.4
VISMALE	3	4.6M	26M	118M	4.7	-	9.7
ACKLEY4	4	1.5M	32M	204M	6.8	-	12.8
AMAZON01	6	0.2M	0.4M	2.2M	0.12	1.6	0.3
AMAZON02	7	0.4M	1.0M	18.4M	0.28	9.8	1.5
ROADNET	3	1.9M	2.5M	4.8M	0.8	3.3	1.0
SPHERE-1.0	16	100	224	0.6M	0.003	0.9	0.04
SPHERE-1.2	21	100	285	26M	0.0032	-	1.5
SPHERE-1.3	23	100	382	197M	0.0034	-	11.01

Table 2.1: Dataset used and storage cost for encoding the corresponding simplicial complex through IA^*, IG and Simplex Tree (ST) data structures.

computation in chunks of approximatively homogeneous size. The idea is to have a decomposition of the embedding space of the complex Σ according to a hierarchical spatial index built on the vertices of the complex. Thus, a block b of the space subdivision induced by the spatial index contains a subset V_b of the vertices of the complex plus all top simplices having at least one vertex in V_b.
The Stellar Tree allows for a local reconstruction of the optimal application-dependent topological data structure to solve the task at hand using a fraction of the memory required by the corresponding global topological data structure. For this reason, Stellar Tree is well suited for applications where we efficiently extract local topological relations.

Comparisons
Several comparisons between implementations of the presented data structures have been performed.
For low-dimensional simplicial complexes, comparisons between IA^*, IG and IS data structures have been presented in [CDW11, CD13]. They have shown that:

- for 2-complexes, the IA^* data structure requires on average 60% of the space of the IS data structure and 20% of that of the IG;
- for 3-complexes, it requires on average 40% of the space of the IS data structure and 30% of that of the IG.

A contribution of our work has been to extend such experimental comparisons to simplicial complexes of arbitrary dimensions. In [FID14, FIDon], we have performed such a comparison between the IA^* data structure, the IG and the Simplex Tree. Table 2.1 summarizes the characteristics of the datasets and the storage costs with the three data structures used. For each dataset we are indicating the dimension of the resulting simplicial complex (column d), the number of vertices (column $\lvert\Sigma_0\rvert$) and top simplices (column $\lvert\Sigma_T\rvert$), the size of the complex (column $\lvert\Sigma\rvert$) and the storage cost required by the three different data structures in Gigabytes (GB).

Observing the storage costs of the three data structures we can notice that this latter increases based on the total number of simplices only when we are using the IG or the Simplex Tree. The IG runs pretty fast out of memory (indicated with -) while the Simplex Tree has a much higher limits since it encodes only a subset of the boundary relations between simplices. The IA^* is not depending on the total number of simplices but only on the top simplices. This means that simplicial complexes in low dimensions (like ROADNET or the volumetric datasets) may require much more memory than, for example, SPHERE-1.3 (being a 23-simplicial complex composed by less than 400 top simplices). In general, we notice that the IA^* behaves better than the Simplex Tree. The ratio between the two data structures roughly depends on the ratio between the number of top simplices and the size of the complex. Thus, in the worst case we have a IA^* that is 1.2 times more compact than the Simplex Tree (ROADNET dataset), up to the case of SPHERE-1.3 where the storage cost for the IA^* is negligible with respect to the 11 GB required by the Simplex Tree.
Another comparison has involved the Stellar Tree, IA^* data structure and the Simplex Tree [Fel15]. First of all, a comparison between the expressive power of these data structures can be considered. From this point of view, the Stellar Tree requires that the complex is embedded in a metric space, while the IA^* and the Simplex Tree can handle abstract complexes.
Experimental evaluations have showed that the Stellar Tree is always more compact than the IA^* and of the Simplex Tree, requiring on average:

- 20% of the storage of the IA^* data structure;
- 2% to 5% of the storage used by the Simplex Tree.

To investigate the feasibility of the Stellar Tree as a general purpose data structures for simplicial complexes, the timings required for retrieving basic topological relations (boundary, coboundary, adjacency), which are crucial building blocks for navigating the simplicial complex, are being evaluated. Our interest is in data structures which can support several basic modeling operations and then can be the basis for a tool for homology computation and topological segmentation. First results show that extracting massively the coboundary relations of the vertices, the Stellar Tree requires from 80% to 95% less time than the same massive algorithm executed on the IA^* data structure.
Moreover, working separately on each chunk of the spatial index, the Stellar Tree provides an implicit mechanism to maintain the memory consumption at runtime low.

2.2 Multi-resolution models

Due to the complexity of real data sets and shapes, such as terrain models or volumetric scalar fields, the investigation of hierarchical methods to control and adjust the *level of detail (LOD)* of a given dataset is an active research area. A *multi-resolution model (or LOD model)* permits to obtain different representations of an object at different levels of detail.

Progressive models [SG98] encode a coarse complex plus a linear sequence of updates refining it, and a shape at an intermediate resolution is obtained by truncating the sequence of refinements at some point. Unlike progressive models, in a multi-resolution model the level of detail can be uniform or vary other the object.
The resolution of a cell complex can be considered as an accuracy parameter in the representation of a given spatial object and is related to the density of its cells. According to our intuition, in order to produce accurate object descriptions a high resolution, i.e., a high number of cells of small size, is required. On the other hand, there is not always the need of the highest possible accuracy in each part of the shape. A sufficiently high accuracy for the specific application task can be achieved by locally adapting the resolution of the considered data set in different parts, thus reducing processing costs and memory space consumption. A multi-resolution model allows to face this problem and it basically consists of a structure which organizes a collection of alternative representations of the shape at full resolution. This structure is built off-line in a preprocessing step and can be efficiently queried according to parameters specified by an application task.

Several application domains have taken advantage from the use of a multi-resolution model. So, in the literature a lot of different LOD models have been developed. A complete classification of the proposed multi-resolution models is out of the scope of this thesis. The main aim of this section is to give a brief overview of such LOD models more related to our work.
According to the kind of information they allow to handle, multi-resolution models can be subdivided in two main classes:

- *geometry-based* models,
- *morphology-based* models.

We denote as geometry-based the multi-resolution models based on simplification and refinement modifications only affecting the cellular structure and the geometry of the cell complex on which they are performed. A multi-resolution model reveals to be useful tool especially to the description of scalar fields defined on a shape, e.g., terrain models or volumetric scalar fields. In order to manage and model these data, a morphology-based multi-resolution model is preferred. These models use the scalar field to guide the simplification process and are based on refinement and simplification operators acting on the regions of influences of the critical points of the field leaving the geometry and the cellular structure of the underling shape unchanged.

Geometry-based models can be further classified into nested and non-nested model. Nested models are based on recursive modifying operations acting on top cells, such as bisections, and exploit spatial data structures such as quadtrees and octrees. Non-nested models are based on modifications that can affect more than one top cell and can be classified according to the kind of operators used, e.g., vertex-based and edge-based multi-resolution models. For detailed discussions about geometry-based models please refer to [DM02, DDMP03, DDM$^+$06].

Morphology-based models can be defined when the shape to be studied is endowed with a scalar field. In this context, the data represented at different levels of detail is the decomposition of the shape induced by the field, e.g., the Morse complexes induced by a Morse function, instead of the structure and the geometry of the shape itself. A survey of such models can be found in [ČDMI14]. To be mentioned is the multi-resolution model for Morse complexes introduced in [ČDI12, CDI13, Iur14] which is based on a graph-based structure compactly encoding and quickly extracting the topology of the two Morse complexes as well as the 1-skeleton of the Morse-Smale complex at different levels of detail.

Recently, a new class of multi-resolution models combining both geometrical and morphological modifications is under investigation. A first example of such a model has been proposed and developed for scalar fields on triangulated terrain models [ID14].

2.3 Computing simplicial homology

The retrieval of homological information of a shape is a time-consuming task. To efficiently compute standard and persistent homology, a bunch of methods has been proposed. In this section, we provide a classification of these techniques. Then, we describe each class of methods focusing on the most relevant algorithms from our point of view. Finally, we give an overview of the software tools to actually compute standard and persistent homology.

2.3.1 Classification

In the literature, several algorithms have been developed to compute homology and persistent homology. Most of the techniques introduced for persistent homology computation can be suitably adapted for retrieving homology just by choosing a trivial filtration.
The classical way to retrieve homological information is based on a matrix reduction, the *Smith Normal Form (SNF) reduction* (see Subsection 1.2.1.3), applied to the boundary matrices. Similarly, an algorithm based on SNF reductions [Zom05] can be used to compute persistent homology. Although SNF-based methods are theoretically valid in any dimension, their complexity is super-cubical in the number of the simplices of the complex. This has led to develop other techniques to compute standard and persistent homology.
Based on the strategy they adopt, we can subdivide the methods introduced in the literature to retrieve homological information into the following classes:

- direct optimizations,
- coarsening and pruning approaches,
- distributed approaches,
- methods based on annotations.

We refer as *direct optimizations* all those methods that improve the efficiency of the SNF computation or that introduce new techniques, based on a matrix reduction, to compute the homological information. Stochastic [Gie96] and deterministic [KB79, Sto96] methods represent a first attempt of optimization of the SNF reduction algorithm. Another strategy, only available for simplicial complexes embeddable in a 3-dimensional sphere, to retrieve homological information efficiently is the incremental algorithm proposed in [DE95, ELZ02]. In [ZC05], persistent homology with coefficients in a field is obtained with a linear complexity in practical applications by studying a graded module over a polynomial ring. Several sequential optimizations of this algorithm have been proposed: in [MMS11] exploiting zigzag persistent homology, in [CK13] by using an output-sensitive algorithm, in [CK11] improving the running time thanks to a suitable change in the order of column reduction, in [DMVJ11] investigating relations between persistent homology and cohomology, in [BM14] introducing an algorithm computing persistent homology with respect various coefficient fields in a single matrix reduction.

We classify as *coarsening and pruning approaches* the methods which reduce the size of the input complex without changing its homology, by applying iterative simplifications, and by computing the homology when no more simplifications are possible. Some of these approaches are based on reductions and coreductions [MB09, MW10, DKMW11], others simplify the simplicial complex via acyclic subspaces [MPZ08, BDMZ12] or by using edge contractions [ALS11]. A similar approach is based on the notion of tidy set [Zom10b], a minimal simplicial set obtained trimming and thinning the original simplicial complex. This method is particularly effective for computing the homology of clique complexes, such as Vietoris-Rips complexes, since it prevents to first construct them. Another class of reduction approaches [RWS11, HMMN14, HMM$^+$10] is based on the construction of the discrete Morse complex, which has the same homology as the input complex [For98, For02], or of its iterated version [DW12].

Distributed approaches efficiently retrieve the homological information of a complex through parallel and distributed computations. Some of such approaches are based on a decomposition of the simplicial complex [BCA$^+$11, LZ14]. Some others act directly on the boundary matrices. For instance, in [EH08, LSVJ11], persistent homology is locally computed and then merged through spectral sequences; in [MNV13], homology groups are retrieved by parallelizing a coreduction-based algorithm; in [BKR14a], persistent homology is obtained thanks to a parallelizable algorithm performing matrix reductions only after having decomposed the matrices in chunks; in [BKR14b], a scalable algorithm for computing persistent homology in parallel in a distributed memory environment is proposed.

Recently, a new strategy of computation is revealing very efficient. We call these approaches *methods based on annotations*. In [DFW12, BDM13], persistent homology is efficiently retrieved by the use of annotations which are vectors associated with each simplex of the complex encoding in a compact way the homological class of the simplex itself. The annotation-based approach is not far from being considered in the class of the direct optimizations. We prefer to describe this approach in a separated subsection

because of the use and the encoding of a matrix specific for annotations improving this method from the point of view both of the performances and of the storage cost.

2.3.2 Direct optimizations

Direct optimizations represent a first attempt in reducing the time complexity of the algorithm retrieving the groups of standard and persistent homology of a simplicial complex. These methods can be used independently or can be combined with a coarsening, a pruning or a distributed approach in order to further improve the efficiency.

In this subsection, we focus on the two most relevant algorithms in the class of the direct optimizations. Specifically, we briefly describe the algorithm proposed in [ZC05] and the dual approach introduced in [DMVJ11]. Both methods aims to efficiently compute persistent homology.

2.3.2.1 The standard algorithm

The algorithm introduced in [ZC05], that from now on we will call *standard algorithm*, is an approach generalizing and extending to any dimension the incremental method proposed in [DE95, ELZ02].
This method is only valid for persistent homology with coefficients in a field. Under such assumption, time complexity is linear in practical applications. This positive result, in combination with the fact that in low dimensions the homological information with respect to coefficients in a field completely characterizes the homology groups of a simplicial complex, is one of the reasons why in many applications standard and persistent homology are considered with respect to coefficients in a field.
The approach is based on the fact that the persistent homology of a filtered simplicial complex is simply the standard homology of a specific graded module over a polynomial ring. Choosing as coefficient group a field $\mathbb{F}$, typically $\mathbb{F} = \mathbb{Z}_2$, the considered polynomial ring $\mathbb{F}[t]$ becomes a PID. In this context, by executing an SNF reduction, the graded module over $\mathbb{F}[t]$ can be decomposed and the persistent homology of the filtered simplicial complex retrieved.

2.3.2.2 The dual algorithm

In [DMVJ11], a *dual algorithm* with respect to the standard one is presented. The work establishes algebraic relationships between homology and cohomology groups and shows that they contain equivalent information if considered with coefficients in a field.
The dual algorithm proposed in [DMVJ11] is based on a matrix reduction similar to the one described in [ZC05] but, by exploiting the theoretical duality between homology and cohomology, it acts on the row of the matrix instead of on the columns. In most of the cases, this strategy, according to a further optimization, allows to compute persistent

homology of a filtered simplicial complex in a more efficient way with respect to the standard algorithm.

2.3.3 Coarsening and pruning approaches

The methods introduced in this subsection are mainly based on preprocessing the complex in order to reduce the size of the complex and thus the complexity of the homological computation that will be usually performed by a matrix reduction method.

The methods described here have been originally developed in order to speed up the computation of standard homology. In spite of this, by requiring additional compatibility conditions between the simplification operator and the filtration of the input complex, most of the approaches we present in this subsection can be easily exploited to efficiently retrieve persistent homology [DW+14].

2.3.3.1 Approaches based on acyclic subcomplexes

The first class of coarsening and pruning approaches we describe is based on the removal of subcomplexes, called *acyclic*, with null homology. This idea has been introduced and developed in [MPZ08, BDMZ12]. Here, we focus mainly on [BDMZ12] which deals with simplicial complexes, but this approach can be generalized in the context of cell complexes. We refer to the above mentioned papers for a more general and detailed discussion.

As already claimed, the main idea of this coarsening approach consists of determining an acyclic subcomplex $\mathcal{A}$ of a simplicial complex Σ as large as possible and computing the relative homology of Σ with respect to $\mathcal{A}$, which is much easier to compute than the homology of Σ.

Definition 2.1. A simplicial complex $\mathcal{A}$ is called *acyclic* if its homology is isomorphic to the homology of the one point complex P, i.e.,

$$H_k(\mathcal{A}) \cong H_k(P) \cong \begin{cases} \mathbb{Z} & \text{for} \quad k = 0 \\ 0 & \text{for} \quad k \geq 1 \end{cases}$$

The main theoretical result on which the work in [BDMZ12] is based is given by the following theorem.

Theorem 2.2. *Let Σ be a path-connected simplicial complex, $\mathcal{A}$ be an acyclic subcomplex of Σ, then*

$$H_k(\Sigma) \cong \begin{cases} \mathbb{Z} & \textit{for} \quad k = 0 \\ H_k(\Sigma, \mathcal{A}) & \textit{for} \quad k \geq 1 \end{cases}$$

Relative homology groups of a pair (Σ, Σ') represent the homology groups of the complex obtained considering Σ and collapsing to a single point the subcomplex Σ'. So, considering as Σ' an acyclic subcomplex $\mathcal{A}$, it is intuitively true that the homology of the obtained

complex is the same of the original one. Theorem 2.2 allows us to reduce the computation of the homology of Σ to the calculation of the relative homology of Σ with respect to $\mathcal{A}$, that is easier to compute since chain complex $C_k(\Sigma, \mathcal{A})$ is smaller than $C_k(\Sigma)$.

The main aim is now to determine an acyclic subcomplex of Σ which is as large as possible. In [BDMZ12], the way to obtain this large acyclic subcomplex $\mathcal{A}$ is not uniquely established.
One can proceed in the following way:

Step 1 initialize $\mathcal{A} := \tau$, where τ is an arbitrary simplex of Σ;

Step 2 take σ a simplex of Σ; if $\mathcal{A} \cup \sigma$[1] is acyclic, then set $\mathcal{A} := \mathcal{A} \cup \sigma$ and repeat **Step 2**, otherwise return $\mathcal{A}$ (see Figure 2.2 for an example).

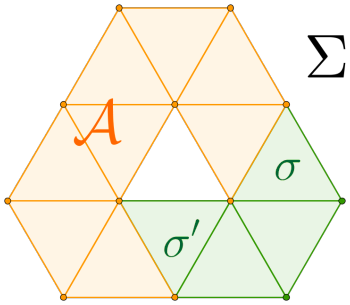


Figure 2.2: An acyclic subcomplex $\mathcal{A}$ of a simplicial complex Σ. By considering the simplex σ, $\mathcal{A} \cup \sigma$ is still acyclic and $\mathcal{A}$ can be updated to $\mathcal{A} \cup \sigma$. Instead, the simplex σ' cannot be added to $\mathcal{A}$, since $\mathcal{A} \cup \sigma'$ is no more acyclic.

By using this approach, it is therefore essential to have a procedure to determine the acyclicity of simplicial complex $\mathcal{A} \cup \sigma$. We want an efficient algorithm, called *AcyclicityTest*, that, taking $\mathcal{A}$ and σ, is able to return `true` if and only if $\mathcal{A} \cup \sigma$ is acyclic.

Theorem 2.3. *Let Σ and Σ' be acyclic simplicial complexes. Then, $\Sigma \cup \Sigma'$ is acyclic if and only if $\Sigma \cap \Sigma'$ is acyclic.*

In our case, we choose $\mathcal{A}$ and simplex σ as the acyclic simplicial complexes of the Theorem 2.3, thus, in order to determine if $\mathcal{A} \cup \sigma$ is acyclic is sufficient (and necessary) to determine if $\mathcal{A} \cap \sigma$ is acyclic. This result gives us a great advantage in computational terms, because $\mathcal{A} \cap \sigma$ is a subcomplex of a simplex and so its size is very small. The most intuitive way to determine the acyclicity of $\mathcal{A} \cap \sigma$ is the direct computation. Unfortunately, if the dimension of the input complex Σ is greater than 4 this method is computationally very expensive. The strategy is therefore to use a partial test, i.e., a test such that if it returns `true`, then $\mathcal{A} \cap \sigma$ is acyclic but, if it returns `false` as output denotes a failure to prove that $\mathcal{A} \cap \sigma$ is acyclic.

[1] In this subsection, when we write $\mathcal{A} \cup \sigma$ or $\mathcal{A} \cap \sigma$, we are committing a little abuse of notation, In fact, we are using σ to denote the closure by inclusion of the simplex σ, i.e., the collection of simplices composed by σ and all its faces.

In [BDMZ12], two different acyclicity tests are proposed. The first one is based on the fact that it is impossible, in a simplicial complex, to create a $(k-1)$-cycle with less than k simplices of dimension $k-1$. So, if we add to the already built acyclic subcomplex $\mathcal{A}$ a k-simplex σ such that the set of the top simplices of $\mathcal{A} \cap \sigma$ is not empty, consists only of $(k-1)$-simplices and has cardinality strictly less than k, then $\mathcal{A} \cap \sigma$ is acyclic.
The second test only requires to compute the geometrical intersection of the top simplices of $\mathcal{A} \cap \sigma$. If this intersection is not empty, then $\mathcal{A} \cap \sigma$ is acyclic and so $\mathcal{A} \cup \sigma$.
The two acyclicity tests proposed in [BDMZ12] for a simplicial complex provide a fast and memory efficient preprocessing of the given complex and an effective way for the computation of its homology. With this method, the Betti numbers and the torsion coefficients of the original simplicial complex can be retrieved. In spite of it could be possible, none of the approaches based on acyclic subcomplexes has been adapted to the computation of the homology generators of a simplicial complex .

Persistent homology computation through acyclic subcomplexes. The approach presented in [BDMZ12, MPZ08] has been adapted for computing persistent homology [DW$^+$14]. Let Σ be a simplicial complex endowed with a filtration $F = \{\Sigma^m \,|\, 0 \leq m \leq M\}$. By adding a further feasibility condition to the acyclic subcomplex $\mathcal{A}$ of Σ ensuring a compatibility with the filtration F, it can be proven that the persistent homology groups of the filtered complex Σ are isomorphic to the ones of $C_*(\Sigma, \mathcal{A})$ with the filtration naturally induced by F. Specifically, the compatibility condition with respect to the filtration F for an acyclic subcomplex $\mathcal{A}$ of Σ is the request that, for each m, the subcomplex $\mathcal{A}^m := \mathcal{A} \cap \Sigma^m$ is an acyclic subcomplex in Σ^m.

2.3.3.2 Approaches based on reductions and coreductions

Another class of coarsening and pruning approaches improves standard and persistent homology computation by iteratively applying two homology-preserving simplifications called *reduction and coreduction operators*. This method is described in [MB09, MW10, DKMW11] for (regular) CW complexes. In our presentation, we will mainly refer to [MB09] and we will focus our attention on simplicial complexes.

In order to formally describe this method, we need to introduce the notions of S-complex and describe reduction and coreduction operators.

S-complexes
An S-complex can be considered as a chain complex in which each chain group is generated by a fixed and finite basis. The notion of S-complex is useful since it allows to theoretically handle cell complexes in which some faces are missing.

Definition 2.4. Consider a finite graded set $S = \bigsqcup_{k \in \mathbb{N}} S_k$ along with a function $\kappa : S \times S \to \mathbb{Z}$. An element $\sigma \in S_k$ is called a *cell* of dimension k (and we write $\dim \sigma = k$). The pair (S, κ) is called *S-complex* if the following properties are satisfied:

1. $\forall \sigma, \tau \in S, \kappa(\tau, \sigma) \neq 0 \implies \dim \tau = \dim \sigma + 1$;

2. $\forall \rho, \tau \in S, \sum_{\sigma \in S} \kappa(\tau, \sigma)\kappa(\sigma, \rho) = 0$.

Under these conditions, κ is called the *incidence function* of the S-complex (S, κ).

Definition 2.5. Let (S, κ) be an S-complex. The *associated chain complex* is the pair $(C_k(S), \partial_k)_{k \in \mathbb{N}}$ where

- $C_k(S)$ the free Abelian group generated by the elements of S_k,
- the boundary map $\partial_k : C_k(S) \to C_{k-1}(S)$, $\forall \tau \in S_k$, is determined by

$$\partial_k(\tau) = \sum_{\tau \in S} \kappa(\tau, \sigma)\sigma$$

Thanks to Definition 2.4, it is easy to prove the following result.

Proposition 2.6. *Let (S, κ) be an S-complex. Then, $(C_k(S), \partial_k)_{k \in \mathbb{N}}$ is a chain complex.*

Then, we can define the homology groups of an S-complex (S, κ) as the homology groups of the chain complex $(C_k(S), \partial_k)_{k \in \mathbb{N}}$ and we can denote them $H_k(S)$.

The concept of S-complex can be related to the notion of simplicial complex, regular grid and cell complex. An S-complex is only a reformulation of a (free) chain complex, because the map κ may be viewed as the matrix of the boundary map. For example, given a simplicial complex Σ, we can see it as S-complex by setting:

- S_k the set of k-simplices of Σ,
- $\forall \sigma, \tau \in \Sigma$, $\tau = [v_0, \cdots, v_k]$,

$$\kappa(\tau, \sigma) = \begin{cases} (-1)^i & \text{if } \sigma = [v_0, \cdots, \hat{v}_i, \cdots, v_k] \\ 0 & \text{otherwise} \end{cases}$$

Consider Σ as a simplicial complex or as an S-complex leads in both cases to the same chain complex and then to the same homology groups.
For our purposes, in the following, we can intuitively consider an S-complex as a simplicial complex in which some simplices may be not present even if their cofaces are in the complex.

Reductions and coreductions
The simplification operators used in [MB09] to reduce the size of the input simplicial complex are reductions and coreductions.
At the topological level, the case of a reduction corresponds to a deformation retraction of a free face onto the complex. The problem is that in most situations free face reductions are quickly exhausted. In order to overcome this problem, coreductions have been introduced. A coreduction is the dual operation with respect to reduction, but is not possible in a standard simplicial complex, while it is available in the context of the S-complexes.
In order to formally introduce the operators of reduction and coreduction, we need to define the notion of boundary and coboundary of a cell.

Definition 2.7. Let (S, κ) be an S-complex and let σ be a cell of S. We call the following sets of cells respectively *(immediate) boundary* and *(immediate) coboundary* of σ with respect to S.

$$\mathrm{bd}_S\,\sigma := \{\rho \in S \,|\, \kappa(\sigma, \rho) \neq 0\}$$
$$\mathrm{cbd}_S\,\sigma := \{\tau \in S \,|\, \kappa(\tau, \sigma) \neq 0\}$$

Definition 2.8. Let (S, κ) be an S-complex and let (σ, τ) be a pair of elements of S such that $\kappa(\tau, \sigma) = \pm 1$. We define (σ, τ)

a *reduction pair* if $\mathrm{cbd}_S\,\sigma = \{\tau\}$,

a *coreduction pair* if $\mathrm{bd}_S\,\tau = \{\sigma\}$.

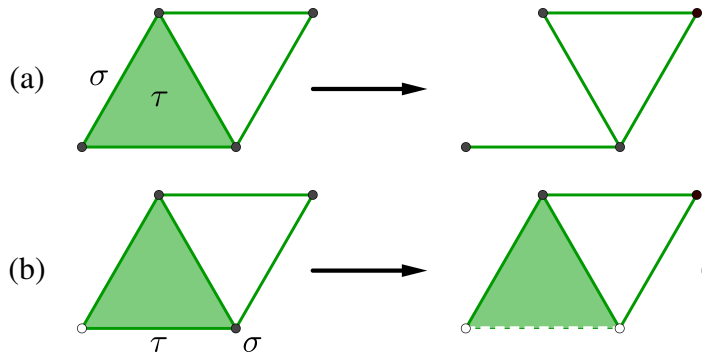


Figure 2.3: On the left, the pairs denoted as (σ, τ) represent for the corresponding S-complexes a reduction pair (a) and a coreduction pair (b), respectively. On the right, the removals of the reduction pair (a) and of the coreduction pair (b) have been performed. Empty vertices and hashed edges represent missing cells.

Even if in the context of the S-complexes our geometrical intuition of homology gets lost, the removal of a reduction or a coreduction pair always preserves homological information.

Theorem 2.9. ([MB09], Thm. 4.1; [MW10], Thm. 2.8). *Let (S, κ) be an S-complex and let (σ, τ) be a pair of elements of S. If (σ, τ) is a reduction or coreduction pair in S, then $(S' := S \setminus \{\sigma, \tau\}, \kappa' := \kappa|_{S' \times S'})_{k \in \mathbb{N}}$ is an S-complex. Furthermore, for each $k \in \mathbb{N}$, $H_k(S)$ and $H_k(S')$ are isomorphic.*

In Figure 2.3, an example of removal of a reduction and of a coreduction pair is depicted.

Coreduction homology algorithm

The simplest reduction algorithm consists in considering all the cells of an S-complex and performing reductions whenever a reduction pair is found. In Figure 2.4, we illustrate this approach starting from a simplicial complex. However, there are at least two problems with this algorithm: the depth of the reduction crucially depends on the order in which the cells are analyzed and the output of the algorithm could produce an S-complex not much smaller than the initial one. Consider now the outcome of the reduction algorithm applied

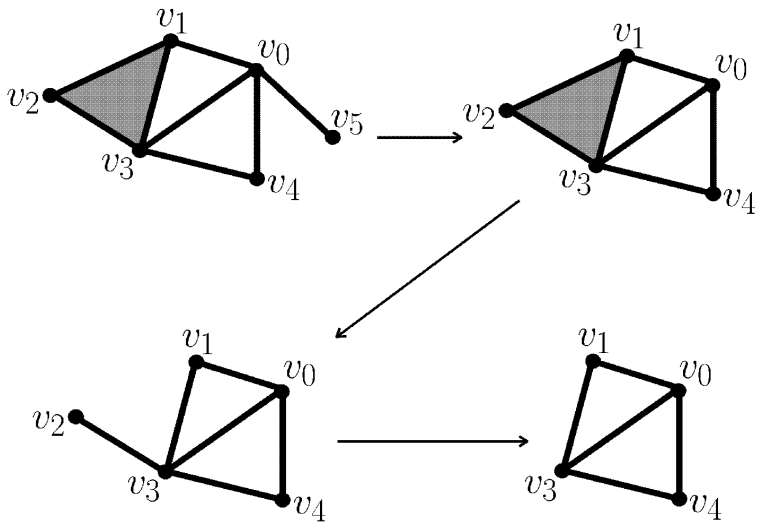


Figure 2.4: An example of reduction operations. In the last complex, no more reduction is available.

to the considered example from the point of view of possible coreductions. First recall that a simplicial complex Σ cannot admit a coreduction pair, because the cardinality of the boundary of any k-simplex is $k+1$ if $k > 0$ or 0 for a vertex. However, if we consider the augmented chain complex associated with the simplicial complex Σ, i.e., if we assume that the empty set is an additional simplex of dimension -1 which is in the boundary of all vertices, then the situation becomes different. Every vertex becomes a simplex with the empty set as the unique element in the boundary and then at least a coreduction is admitted. Having replaced the original chain complex with its augmented version, we will obtain an S-complex whose homology corresponds to the reduced homology $\tilde{H}_k(\Sigma)$ of Σ.

In our example, the resulting sequence of coreductions is presented in Figure 2.5. The outcome is an S-complex consisting of exactly two cells of dimension 1, whose boundary is null. Therefore, these cells are also homology generators and obviously there are no more homology generators.

Note that, if the starting simplicial complex has more than one path-connected component, then the above procedure works only for the connected component of the vertex which has been paired with the empty set. A straightforward remedy is to add one extra cell in dimension -1 for each connected component with its coboundary consisting of all vertices in this component.

Homology generators. Being interested in computing the generators of each homology group, explicit formulas for the isomorphism established in Theorem 2.9 are required. For this, let (S, κ) be an S-complex, (σ, τ) a reduction or coreduction pair in S and $S' = S \setminus \{\sigma, \tau\}$.

Let $\psi_*^{(\sigma,\tau)} : C_*(S) \to C_*(S')$, $\iota_*^{(\sigma,\tau)} : C_*(S') \to C_*(S)$ be two chain maps defined by

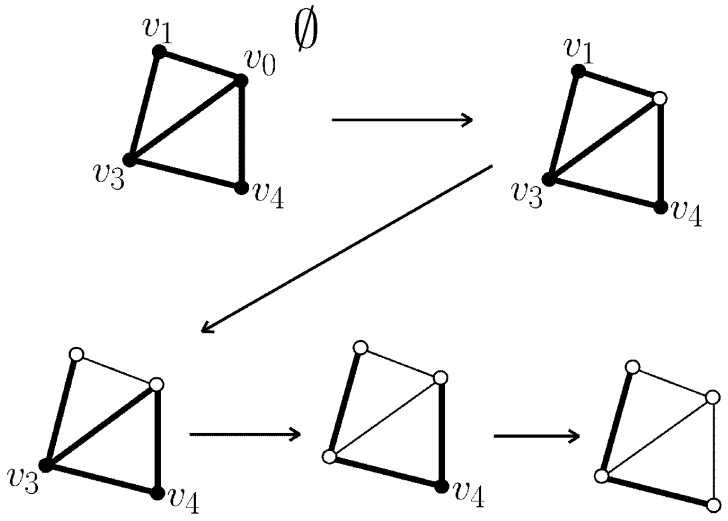


Figure 2.5: An example of coreduction operations. The missing vertices are marked with empty circles and the missing edges with thin lines.

$$\psi_k^{(\sigma,\tau)}(c) = \begin{cases} c - \frac{\langle c,\sigma\rangle}{\langle \partial\tau,\sigma\rangle}\partial\tau & \text{if } k = \dim\tau - 1 \\ c - \langle c,\tau\rangle\tau & \text{if } k = \dim\tau \\ c & \text{otherwise} \end{cases}$$

$$\iota_k^{(\sigma,\tau)}(c) = \begin{cases} c - \frac{\langle \partial c,\sigma\rangle}{\langle \partial\tau,\sigma\rangle}\tau & \text{if } k = \dim\tau \\ c & \text{otherwise} \end{cases}$$

where $\langle \quad , \quad \rangle$ represents the scalar product between two chains.

The chain maps $\psi_*^{(\sigma,\tau)}$ and $\iota_*^{(\sigma,\tau)}$ establish a homology equivalence between the chain complexes $C_*(S)$ and $C_*(S')$ stated in Theorem 2.9. Furthermore, since $\psi_*^{(\sigma,\tau)} \cdot \iota_*^{(\sigma,\tau)} = \mathrm{id}_{C_*(S')}$, the homology generators of S can be retrieved by computing the generators of S' and applying to them the map $\iota_*^{(\sigma,\tau)}$.

Previous results allow to define an algorithm for efficient computation of the homology, called *coreduction homology algorithm* (see [MB09]).

In order to compute the homology of an S-complex S, the strategy is to reduce S as much as possible obtaining a smaller S-complex T and to compute its homology groups through the Smith reduction algorithm. So, since the complexity of Smith's algorithm is super-cubical in the number of cell of the complex (see [Sto96]), the complexity of the presented algorithm will be $O(|T|^\alpha)$ where $\alpha \simeq 3,376$.

From a theoretical point of view, it is possible to exhibit examples where only the trivial coreduction which involves the empty set is available and so $|T|$ is equal to $|S| - 1$. But, from an application point of view, the coreduction homology algorithm allows us to obtain an S-complex T which cannot be reduced further and whose size is significantly

smaller than the size of the starting S-complex S. For example, in the case of regular grids, numerical experiments have shown that one can assume

$$|T| \simeq |S|^{\frac{d-1}{d}}$$

where d is the dimension of the original complex S. By considering this assumption, the complexity of finding homology is $O(|S|^{\alpha\frac{d-1}{d}})$ (see [MB09], Thm. 8.3) and this means that the algorithm is particularly efficient for low-dimensional complexes.

If we are also interested in the retrieval of the homology generators of S, providing of the homology generators of T, we have to compute their images under the maps ι_*. It can proven (see [MW10], Thm. 3.1) that the cost of this procedure is bounded by $O(|S|w(S))$ where $w(S)$ is the maximum number between the faces and the cofaces of a cell of S. Hence, in many practical contexts, we can consider constant this number and assume that the retrieval of the homology generators of the input S-complex S runs in linear time.

Persistent homology computation through reductions and coreductions. The above approach can be easily adapted to the context of the persistent homology computation [DW$^+$14]. Analogously to the approaches based on acyclic subcomplex, a compatibility condition between the simplification operators and the filtration F of the input simplicial complex Σ can be introduced. Both for a reduction and a coreduction pair (σ, τ), the compatibility condition with the filtration is the request that the simplices σ and τ appear at the same step of the filtration. More formally, there must exist $m \in \{1, \ldots, M\}$, such that $\sigma, \tau \in \Sigma^m$ but $\sigma, \tau \notin \Sigma^{m-1}$. Under such assumption, it has been proven that the persistent homology groups of the filtered complex Σ and the ones of the reduced complex with respect to the filtration naturally induced by F are the same up to isomorphisms. So, persistent homology computation of a filtered simplicial complex can be easily redirected to the one of the fully reduced and coreduced complex.

2.3.3.3 Approaches based on edge contractions

Another strategy to preprocess a simplicial complex in order to apply the Smith Normal Form reduction for a smaller input is based on *edge contractions*. As in the previous case, we define a topological operator, called *edge contraction*, which preserves the homology of the original simplicial complex. Unlike coreductions, this operator brings a simplicial complex into an other without losing the simplicity of the resulting complex. This subsection is mainly inspired by [ALS11], and [DEGN99].

Let Σ be a simplicial complex and let Σ_0 be the set of its vertices. Consider $u, v \in \Sigma_0$ and $w \notin \Sigma_0$. In order to describe the edge contraction of uv into w, we define the vertex map f_V that takes vertices u and v to w and takes all other vertices to themselves. We consider the simplicial map f naturally induced by the vertex map f_V. Given a simplex $\sigma = v_0 \cdots v_k$ in Σ, we have that $f(\sigma) = f_V(v_0) \cdots f_V(v_k)$.

Definition 2.10. According to the above notation, the *edge contraction of uv into w* is the operation that given a simplicial complex Σ returns its image Σ' under the simplicial map f, i.e., $\Sigma' = \{f(\sigma) \,|\, \sigma \in \Sigma\}$.

By construction f is surjective and Σ' is a simplicial complex. Note that the edge contraction of uv into w is well defined even when uv is not an edge of Σ.

The main result about edge contraction states the required hypotheses ensuring this operator is homology-preserving. Recall that, given a simplex σ of a simplicial complex Σ, we denote as $\operatorname{link}_\Sigma \sigma$ the simplicial complex $\{\sigma' \in \Sigma \,|\, \sigma' \cup \sigma \in \Sigma,\ \sigma' \cap \sigma = \emptyset\}$ consisting of all the simplices not incident in σ and face of a coface of σ.

Theorem 2.11. (Link condition, [ALS11], Thm. 1). *Let Σ be a simplicial complex. The contraction of the edge $uv \in \Sigma$ preserves the homology whenever*

$$\operatorname{link}_\Sigma uv = \operatorname{link}_\Sigma u \cap \operatorname{link}_\Sigma v$$

Theorem 2.11 ensures that an edge contraction satisfying the link condition preserves homology groups. Notice that, in general, the converse is not true. So, there exist cases in which link condition is not satisfied but the application of the correspondent edge contraction does not affect the homology of the simplicial complex. Examples of homology-preserving and of homology-modifying edge contraction are depicted in Figure 2.6.

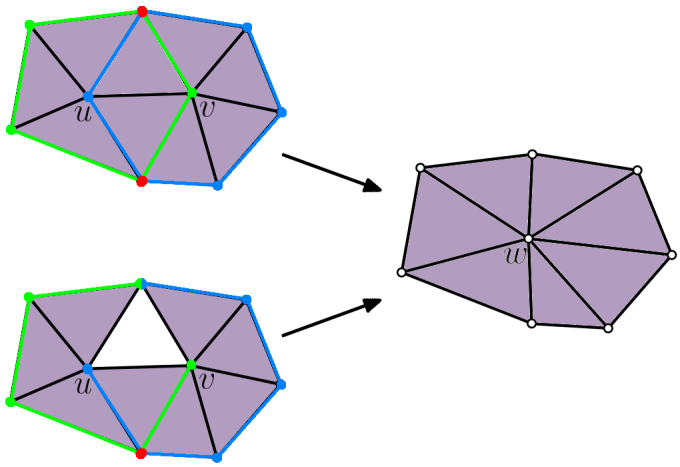


Figure 2.6: Simplices underlined in green, blue and red represent the link of u, v and uv, respectively. Above, the edge contraction of uv into w performed on a configuration satisfying the link condition and so homology-preserving. Below, the edge contraction of uv into w which modifies the homology of the simplicial complex and so not satisfying the link condition.

By using Theorem 2.11, we are able to understand when edge contraction is a homology-preserving operator. This topological operator can be implemented and allows reducing the size of a simplicial complex without losing simpliciality. For this reason, it could be useful in a preprocessing algorithm for homology computation.

As far as we know, in the literature, there is no approach that retrieves persistent homology by exploiting operators of edge contraction. In spite of this, we believe that, similarly to how it has been made for reduction and coreduction operators in [DW$^+$14], a suitable compatible condition between an edge contraction operator and a filtered simplicial complex could be defined.

2.3.3.4 Approaches based on discrete Morse theory

A discrete Morse complex associated with a simplicial complex Σ provides a compact homology-equivalent model of Σ. This equivalence allows us to compute the homology of Σ by applying the SNF reduction algorithm on the discrete Morse complex $\mathcal{M}_*$ instead of on $C_*(\Sigma)$. Since the number of critical simplices generating $\mathcal{M}_*$ is usually negligible with respect to the number of simplices of Σ, this method considerably improves the efficiency of homology computation. By adding some further compatibility conditions, the same reducing strategy can be used to efficiently retrieve persistent homology of a filtered simplicial complex.
In the literature, several algorithms, based on the discrete Morse theory, have been developed to retrieve standard and persistent homology [RWS11, HMMN14, HMM$^+$10]. They mainly differ for the method used to build the discrete Morse complex. The algorithms proposed in [HMMN14, HMM$^+$10] exploit coreduction operators to compute a gradient vector field. In [RWS11], the input complex is endowed with a scalar function on its vertices inducing a filtration on the whole complex and the gradient vector field is retrieved by an algorithm partitioning the complex through the lower stars of its vertices. The method introduced in [RWS11] has also a theoretically relevance. It has been proven that, for regular cell complexes up to dimension 3, the gradient vector field is in some sense optimal with respect to the chosen filtration. In fact, each critical cell of the discrete Morse complex induces a change in the homology of the complex providing a relevant contribute in the persistent homology groups.

In this subsection, we consider the discrete Morse complex associated with a (filtered) simplicial complex as an input. We will focus on the algorithms to actually build it in Subsection 2.4.3.

Homology generator computation through discrete Morse theory
The homological equivalence between chain complex $\mathcal{M}_*$ and simplicial complex Σ implies that, by using the SNF reduction algorithm, we are able to obtain the simplicial homology of Σ. The homology generators of degree k are computed through SNF reduction. Their geometric realization is obtained starting from the critical k-simplices of V and navigating the gradient pairs. Computing the homology generators corresponds to computing the cells of the descending Morse complex.

The computation of a descending k-cell starts from a critical k-simplex σ. All the $(k-1)$-simplices in the immediate boundary of σ are then selected and, among them, only the $(k-1)$-simplices paired with a k-simplex different from σ are considered. Such k-simplices are inserted into a queue, and the traversal of the complex Σ continues in a breadth-first fashion until all the V-paths starting from σ have been visited. In 2D, for example, we start from a critical 2-simplex (maximum) σ and, by following the gradient pairs, we continue adding adjacent 2-simplices until all V-paths from σ have been traversed.

The computation of the descending Morse complex is performed through constant time operations at each simplex on the visited V-paths. In 2D, the extraction of a descending k-cell requires linear time in the number of simplices of Σ forming the k-cell since each simplex is visited at most once. As we will discuss in Section 2.4, in three dimensions and higher, visiting the gradient paths among saddles may have a cubical time complexity.

Persistent homology through discrete Morse theory
As shown in [MN13], discrete Morse theory can be used to efficiently compute persistent homology. Let Σ be a simplicial complex and consider a filtration $F = \{\Sigma^m \,|\, 0 \leq m \leq M\}$ of Σ.

A gradient vector field V of Σ is a *filtered gradient vector field of* F if, for each pair $(\sigma, \tau) \in V$, there exists $m \in \{1, \ldots, M\}$ such that $\sigma, \tau \in \Sigma^m$ and $\sigma, \tau \notin \Sigma^{m-1}$.

Let us denote V^m as the pairs of V whose elements are in Σ^m.

Proposition 2.12. ([MN13], Prop. 4.2). *For each m, $\mathcal{M}_*^m = (\mathcal{M}_k^m, \tilde{\partial}_k|_{\mathcal{M}_k^m})_{k\in\mathbb{N}}$, where $\mathcal{M}_k^m$ is the free Abelian group generated by the simplices of Σ^m not in V^m, is a chain complex and $\{\mathcal{M}_*^m \,|\, 0 \leq m \leq M\}$ is a filtration of $\mathcal{M}_*$.*

It is possible to extend Theorem 1.48 to the level of persistent homology.

Theorem 2.13. ([MN13], Thm. 4.3). *Let $F = \{\Sigma^m \,|\, 0 \leq m \leq M\}$ be a filtration of the simplicial complex Σ endowed with a filtered gradient vector field V. Then, for all m, k and p,*

$$H_k^p(\Sigma^m) \cong H_k^p(\mathcal{M}_*^m)$$

Analogously to the standard homology case, Theorem 2.13 allows to speed up persistent homology computation. In fact, once a discrete Morse complex associated with a filtered gradient vector field has been built, one can retrieve persistent homology groups by applying a matrix reduction algorithm.

2.3.4 Distributed approaches

Another strategy to efficiently recollect homology of a cell complex which is able to handle huge data is the *distributed homology*. The aim of this class of approaches is to partition a simplicial complex or its boundary matrices into local pieces, compute the homology of each piece in parallel and glue the pieces together by retrieving the homological information of the original object.

2.3.4.1 Constructive Mayer-Vietoris algorithm

A first modular algorithm for homology computation of simplicial complexes is the so-called *constructive Mayer-Vietoris (MV) algorithm*. This algorithm has been introduced in [BCA+11] and it has also been the main topic of my Master Thesis [Fug12]. The

constructive MV algorithm has been developed for standard homology and, up to now, no extension to persistent homology computation has been proposed.

Given a topological space, the most common tool in algebraic topology to split it in two parts in order to retrieve the homology of the original space is through the Mayer-Vietoris sequence. The following theorem, whose proof can be found in [Mun84], Thm. 25.1, describes this tool in the case of a simplicial complex.

Theorem 2.14. (Mayer-Vietoris sequence). *Let Σ be a simplicial complex. Let $\mathcal{A}$ and $\mathcal{B}$ be subcomplexes of Σ such that $\Sigma = \mathcal{A} \cup \mathcal{B}$. Then, we have the short exact sequence*

$$0 \to C_*(\mathcal{A} \cap \mathcal{B}) \xrightarrow{i} C_*(\mathcal{A}) \oplus C_*(\mathcal{B}) \xrightarrow{j} C_*(\Sigma) \to 0$$

which induces the following exact sequence in homology

$$\cdots \to H_k(\mathcal{A} \cap \mathcal{B}) \to H_k(\mathcal{A}) \oplus H_k(\mathcal{B}) \to H_k(\Sigma) \to H_{k-1}(\mathcal{A} \cap \mathcal{B}) \to \cdots$$

called the Mayer-Vietoris sequence of $(\mathcal{A}, \mathcal{B})$.

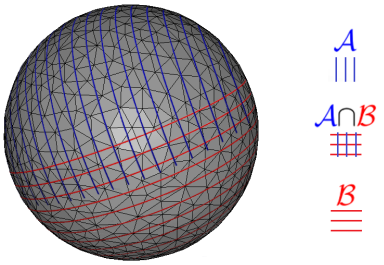


Figure 2.7: A triangulated sphere $\mathbb{S}^2$ and an its decomposition in two subcomplexes $\mathcal{A}$ and $\mathcal{B}$.

In some cases, as in the one depicted in Figure 2.7, the exactness of the Mayer-Vietoris sequence provides us an easy way to compute simplicial homology. If, for instance, the Mayer-Vietoris sequence contains

$$\cdots \to 0 \to H_k(\Sigma) \to 0 \to \cdots$$

we can immediately conclude that $H_k(\Sigma) = 0$. But, if

$$\cdots \to 0 \to \mathbb{Z}_2 \to H_k(\Sigma) \to \mathbb{Z}_2 \to 0 \to \cdots$$

appears in the Mayer-Vietoris sequence, we are not able to conclude if $H_k(\Sigma) \cong \mathbb{Z}_4$ or if $H_k(\Sigma) \cong \mathbb{Z}_2 \oplus \mathbb{Z}_2$.

This problem is due to the non-constructiveness of the proposed method and for this reason we are not able, using the introduced tools, to implement an algorithm based on the Mayer-Vietoris sequence.

Constructive Algebraic Topology
In order to turn Mayer-Vietoris sequences in a constructive method, several theoretical notions and results have to be introduced (see [RS99, RS12] for a detailed discussion).

A first important tool is the notion of *reduction* of chain complexes. Informally, a reduction $\rho : C_* \Rrightarrow C'_*$ is a homology equivalence between a "large" chain complex C_* and a "small" one C'_*. The equivalence provided by a reduction allows to completely retrieve the homological information, homology generators included, of a chain complex from the knowledge of the same information of the other chain complex. The most common example of reduction is the reduction naturally induced by the Smith Normal Form reductions of the boundary maps of a chain complex C_*. It is called *homological Smith reduction* and denoted by $\rho : C_* \Rrightarrow EC_*$.

Another important concept is the notion of cone of a morphism. Let $A_* = (A_k, d^A_k)_{k\in\mathbb{Z}}$ and $B_* = (B_k, d^B_k)_{k\in\mathbb{Z}}$ be two chain complexes and $\phi : A_* \to B_*$ be a chain map between them. We call *cone of* ϕ the chain complex $\mathrm{Cone}(\phi)$ whose group of k-chain is $B_k \oplus A_{k-1}$ and k^{th} boundary map is given by the matrix:

$$d^C_k := \begin{pmatrix} d^B_k & \phi_{k-1} \\ 0 & -d^A_{k-1} \end{pmatrix}$$

i.e., if $a \in A_{k-1}$, $b \in B_k$, we have $d^C_k(b, a) = (d^B_k(b) + \phi_{k-1}(a), -d^A_{k-1}(a))$.

These tools combined together bring to the proof of the following theorem which represents the basis of the constructive MV algorithm.

Theorem 2.15. (Short exact sequence, [BCA+11], Thm. 5.5). *Let Σ be a simplicial complex, let $\mathcal{A}$ and $\mathcal{B}$ be subcomplexes such that $\Sigma = \mathcal{A} \cup \mathcal{B}$. The short exact sequence of Mayer-Vietoris defined as*

$$0 \longrightarrow C_*(\mathcal{A} \cap \mathcal{B}) \underset{i}{\overset{\tau}{\leftrightarrows}} C_*(\mathcal{A}) \oplus C_*(\mathcal{B}) \underset{j}{\overset{\sigma}{\leftrightarrows}} C_*(\Sigma) \longrightarrow 0$$

is an effective short exact sequence of chain complexes and so, it produces the reduction

$$\rho : \mathrm{Cone}(i) \Rrightarrow C_*(\Sigma)$$

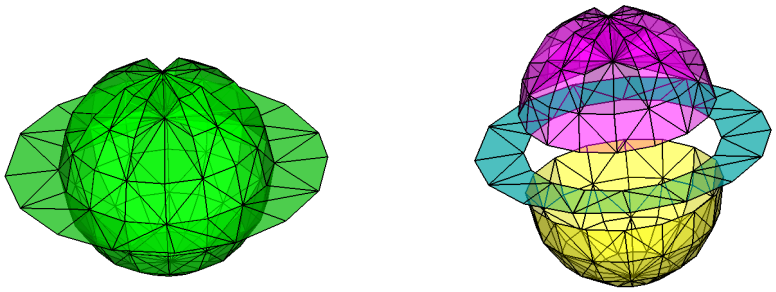
Constructive MV algorithm
We can now describe the idea on which the constructive MV algorithm is based. First of all, the algorithm computes the homological Smith reductions of the chain complexes associated with $\mathcal{A}$ and $\mathcal{B}$. Then, by deeply exploiting the notions and the theorems just introduced, the following equivalence of reductions is obtained.

$$C_*(\Sigma) \Lleftarrow \mathrm{Cone}(i) \Rrightarrow E(\mathrm{Cone}(Ei))_*$$

From this equivalence, knowing the homology of $\mathcal{A}$ and $\mathcal{B}$, it is immediate to retrieve the homology and the homology generators of Σ.
The constructive MV algorithm just presented, here described only in the case of a decomposition of Σ in two pieces, can be generalized for a decomposition with an arbitrary finite number of components (see [Fug12]).

Manifold-connected decomposition. From a theoretical point of view, the constructive MV algorithm can be performed using any kind of decomposition. Since the small size of the intersections between the components helps to reduce the execution time, the algorithm has been implemented using the *manifold-connected decomposition* (introduced in [DMMP03]). In a simplicial d-complex Σ, a $(d-1)$-simplex σ is called *manifold* if there are at most two d-simplices of Σ incident in σ. We call *manifold* $(d-1)$*-path* a sequence consisted of adjacent manifold $(d-1)$-simplices of Σ.

Definition 2.16. A simplicial d-complex Σ in which every pair of d-simplices are connected by a manifold $(d-1)$-path, is called *manifold-connected complex* (*MC complex*) of dimension d.



(a) (b)

Figure 2.8: An example of an MC decomposition: a hollow ball that is pinched at the top and has a circular ring (a), its MC decomposition into three manifold-connected components (b).

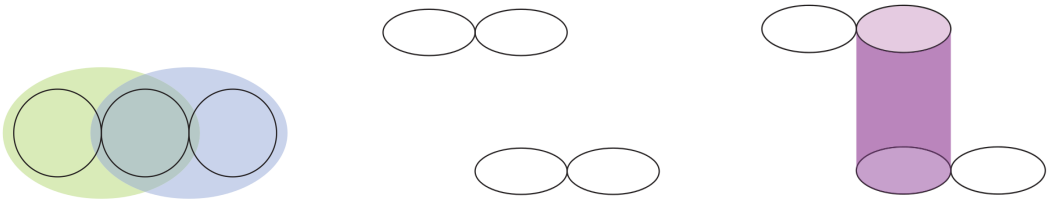
It is easy to prove that any simplicial complex Σ has a unique decomposition in MC complexes called *MC decomposition* (see Figure 2.8 for an example).
As mentioned above, the limited size of the intersections between the MC components on which the simplicial complex is partitioned has led to elect the MC decomposition as the adopted subdivision criterion in the constructive MV algorithm.

In [BCA+11], an implementation of a constructive MV algorithm based on the MC decomposition has been developed and compared with the SNF reduction algorithm. Experimental evaluations have revealed that the considered distributed approach leads to a reduction in the storage cost and in the computational time.

2.3.4.2 Multicore homology

In the literature, another algorithm, similar to the constructive MV algorithm, has been developed to retrieve the homological information of a simplicial complex in a distributed fashion [LZ14, ZC08]. In this work, the authors design and implement a framework, called *multicore homology*, for parallel computation of persistent homology over field coefficients.

The main tool used in [LZ14] is the *blowup complex* (see Figure 2.9). Given a cover of the input complex, a blowup complex is a space homologically equivalent to the input complex which organizes the various subspaces needed for employing the global homology reconstruction. Intuitively, it is built by cutting the input complex into the components in which the cover splits it and connect them along their intersections by “bridges" which allow to duplicate cells belonging to more than one component of the cover.
Blowup complex dues its name to the fact that, given an arbitrary cover of a shape consisting of n components, the resulting blowup complex can be 2^n times larger than the original complex. In spite of the algorithm never explicitly encodes the blowup complex, its potentially huge size affects the efficiency of the algorithm. For this reason, even if the presented algorithm is theoretically valid for any cover defined on the space, in [LZ14] a cover based on graph partitions, producing a blowup complex at most 3 times larger respect to the input complex, has been proposed. Moreover, the authors show that finding a cover which minimizes the size of the blowup of a complex is NP-hard.



(a) Space and cover (b) Local pieces ($t = 0$) (c) Blowup complex ($t = 1$)

Figure 2.9: Given a space equipped with a cover (a), first blow up the space into local pieces (b) and then glue back the pieces to get the blowup complex (c), giving a filtration consisting of two complexes at times $t = 0$ and $t = 1$, respectively.

The main steps of the multicore homology algorithm are the following (see Figure 2.9). First at all, given a cover of the input complex, the complex is split into the components provided by the cover and the homology of each local piece is independently computed. After this process, the blowup complex is built. By computing its persistent homology with respect to a particular filtration, blowup complex allows glueing the homological information of each components quickly retrieving the homology of the input complex.

Localized homology through multicore homology. Another task of multicore homology algorithm is also to retrieve localized homology generators. As mentioned in Subsection 1.2.2, the localization problem is determining the location of topological features within a simplicial complex.
In [ZC08], given a simplicial complex Σ and a cover, a homology class of Σ is defined to be *local* if it exists in one of the pieces of the cover. Given a cover, persistent homology can be used to determine whether a homology class is local. This method provides an algorithm for generating a homology basis localized with respect to the chosen cover. Actually, the blowup complex has the same homology as the original space but, also incorporates the geometric cover information within its structure. By computing persistent homology, one can retrieve homology bases for the blowup complex that are compatible with bases for the local pieces. These localized homology generators naturally reflect the

quality of the given cover, and covers that reflect the geometry of the simplicial complex give better descriptions.
In spite of this interesting attempt, an efficient method to effectively retrieve localized homology of a simplicial complex of arbitrary dimension has yet to come.

2.3.4.3 Approaches based on spectral sequences

The mathematical notion of *spectral sequence* [McC01] allows to retrieve homological information by taking successive approximations. Furthermore, it revealed to be a powerful tool for parallelizing the computation of persistent homology.

A first algorithm based on spectral sequences has been proposed in [EH08]. It divides the boundary matrix of a filtered simplicial complex into blocks and reduces them from the diagonal outwards in different phases that can be executed in parallel.

Analogously to this latter, the algorithm introduced in [LSVJ11] exploits spectral sequences to retrieve persistent homology groups in a distributed manner. It adopts a divide-and-conquer strategy by locally computing information and glueing them together. To achieve this result, differently from other parallelizable methods, it does not use the Mayer-Vietoris long exact sequence but a spectral sequence generalizing it. After dividing the filtered simplicial complex in input according to a cover, the spectral sequence generates a chain complex, called *total complex* similar to the blowup complex used in [LZ14]. The knowledge of the total space gives the machinery necessary to iteratively compute better approximations of the global persistent homology of the filtered complex from the local persistence computations in each components of the cover. In spite of the main focus of the algorithm in [LSVJ11] is on the glueing step more than in the dividing one, some methods to subdivide the input complex have been proposed. The introduced covers, called cube, Voronoi and geodesic Voronoi partitions, minimize the size of the intersections in order to maximally parallelize the computation.

2.3.4.4 Approaches based on a matrix partition in chunks

Recently, the class of the methods performing a distributed computation of the persistent homology has been enriched by two new algorithm [BKR14a, BKR14b].

The algorithm introduced in [BKR14a], called *chunk algorithm*, is very interesting because of its correlations with several other approaches. For instance, it generalizes the methods based on discrete Morse theory, is closely related to the spectral sequence algorithm [EH08] and employs various practical optimization strategies.
The two main contributions of this approach are:

- the introduction of two simple optimization techniques of the standard reduction algorithm, called *clearing* and *compression*, that significantly reduce the number of operations on real-world instances;

- the definition of an algorithm incorporating both the optimizations and also suitable for parallelization.

Besides the worst-case complexity bound is not improved, practical experiments show that significant speed-ups can be achieved through this parallelized approach.

Further improvements to this distributed method have been introduced and discussed in [BKR14b]. The good results reached thanks to these optimizations reveal that in the (persistent) computation the limiting factor is the memory available on the computer rather than the time spent for the calculation.

2.3.5 Annotation-based approaches

Most algorithms for computing topological features of a simplicial complex, such as (co)-homology, persistence and localized homology, have to efficiently reveal the independence or not of a set of cycles. In order to solve this problem, the notion of annotation has been introduced [BCC$^+$12]. An annotation is just a map which assign a (binary) vector to each simplex of a simplicial complex. In this way, it provides the coordinate vector of the homology class of each cycle z in a homology basis and thus, helps us to determine efficiently the topological characterization of z.

In this subsection, first, we formally introduce the notion of annotation and then, we describe how persistent homology computation can be obtained thanks to the use of annotations. For a more detailed discussion about annotations, we refer to [BCC$^+$12, BDM13, DFW12].

2.3.5.1 Annotations

Given a simplicial complex Σ, let Σ_k denote the set of k-simplices in Σ and let $\mathbb{F}$ be an arbitrary field.

Definition 2.17. An *annotation* for Σ_k is a function $a_k : \Sigma_k \to \mathbb{F}^g$. It assigns a vector $a_\sigma := a_k(\sigma)$ of same length g for each k-simplex of Σ.

An annotation for Σ_k induces in a natural way an annotation for the set of k-chains of Σ. Let c be an element in $C_k(\Sigma;\mathbb{F})$, $c = \sum_{i=1}^{q} f_i\sigma_i$ with $f_i \in \mathbb{F}$, $\sigma_i \in \Sigma_k$. We can assign to c vector $a_c := \sum_{i=1}^{q} f_i a_{\sigma_i}$.

Definition 2.18. An annotation $a_k : \Sigma_k \to \mathbb{F}^g$ is called *valid* if the following conditions are satisfied:

- $g = \dim_{\mathbb{F}} H_k(\Sigma;\mathbb{F})$;

- given two k-cycles z_1 and z_2, we have that

$$a_{z_1} = a_{z_2} \iff [z_1] = [z_2]$$

  i.e., z_1 and z_2 belong to the same homology class.

An example of a valid annotation with $\mathbb{F} = \mathbb{Z}_2$ is depicted in Figure 2.10. It is easy to notice the role of annotations to reveal the independence of cycles: 1-cycles only including the left hole have annotation value $(0, 1)$, the ones only including the right hole have value $(1, 0)$, the ones including both the holes take value $(1, 1)$, while the 1-cycles not including any hole have as annotation the null vector $(0, 0)$.

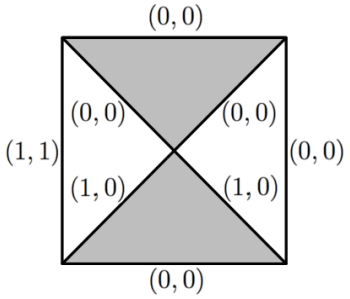


Figure 2.10: A simplicial complex endowed with a valid annotation with $\mathbb{F} = \mathbb{Z}_2$.

The notion of annotation was introduced in [BCC+12] as a tool for computing localized homology. In [BCC+12], the authors propose an algorithm speeding up the computation of an optimal basis for the 1-dimensional homology of a simplicial complex by retrieving a valid annotation of its edges. Furthermore, they provide a method to build a valid annotation for simplicial complexes of any dimension. In spite of this, the proposed method has a high computation time. A more efficient way to build a valid annotation has been introduced with the purpose of speed up persistent homology computation [BDM13]. We describe this method in the next subsection.

2.3.5.2 Persistent homology through annotations

Aside from its importance for quickly checking the independence of cycles, annotations have also further deep connections with homology. Given a simplicial complex Σ and an annotation $a_k : \Sigma_k \to \mathbb{F}^g$, it is possible to define, for each $i = 1, \ldots, g$, a homomorphism $\phi_i : C_k(\Sigma; \mathbb{F}) \to \mathbb{F}$. Given a k-simplex $\sigma \in \Sigma_k$, $\phi_i(\sigma)$ is defined to be $a_\sigma[i]$, the i^{th} element of a_σ.

Proposition 2.19. ([DFW12], Prop. 3.3). *The following statements are equivalent.*

- *An annotation $a_k : \Sigma_k \to \mathbb{F}^g$ is valid.*
- *The cochains ϕ_i, $i = 1, \ldots, g$, are cocycles whose cohomology classes $[\phi_i]$, $i = 1, \ldots, g$, constitute a basis of $H^k(\Sigma; \mathbb{F})$.*

As already mentioned, by taking coefficients in a field, standard and persistent homology groups of a simplicial complex are isomorphic to standard and persistent cohomology groups [DMVJ11].
This duality in combination with Proposition 2.19 is the basis for two algorithms based on annotations for computing persistent homology.

Persistent cohomology algorithm
The algorithm described in [BDM13] retrieves persistent homology of a filtered simplicial complex by iteratively updating a valid annotation of its simplices.
Let Σ be a simplicial complex and let $\emptyset = \Sigma^0 \subseteq \Sigma^1 \subseteq \cdots \subseteq \Sigma^M = \Sigma$ be a filtration of Σ such that $\Sigma^{m+1} = \Sigma^m \cup \{\sigma\}$ with σ a k-simplex. Assume that, for any dimension, a valid annotation a^m is attached to the simplicial complex Σ^m. We describe how to obtain a valid annotation a^{m+1} of Σ^{m+1}.
We compute the annotation $a^m_{\partial\sigma}$ and take actions as follows:

Case 1: $a^m_{\partial\sigma} = 0$. We have that

$$a^m_{\partial\sigma} = 0 \iff [\partial\sigma] = [0] \text{ in } \Sigma^m \iff \beta_k^{\Sigma^{m+1}} = \beta_k^{\Sigma^m} + 1$$

Then, the annotation vector of any k-simplex $\sigma' \in \Sigma^m$ is augmented with a 0 entry so that, if $a^m_{\sigma'} = (f_1, \ldots, f_g)$, $a^{m+1}_{\sigma'}$ is defined to be $(f_1, \ldots, f_g, 0)$. To the new simplex σ is assigned the annotation vector $a^{m+1}_\sigma = (0, \ldots, 0, 1)$. According to Proposition 2.19, this is equivalent to creating a new cohomology class represented by the homomorphism ϕ such that $\phi(\sigma') = 0$ for $\sigma' \neq \sigma$ and $\phi(\sigma) = 1$.

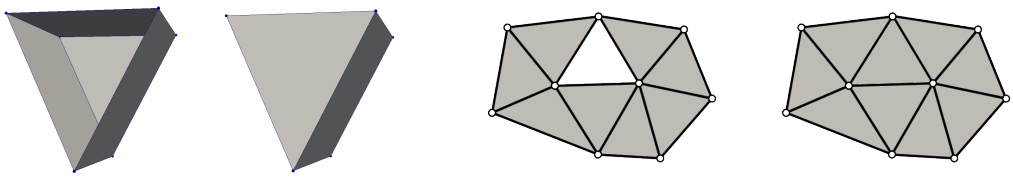
Case 2: $a^m_{\partial\sigma} \neq 0$. We have that

$$a^m_{\partial\sigma} \neq 0 \iff [\partial\sigma] \neq [0] \text{ in } \Sigma^m \iff \beta_{k-1}^{\Sigma^{m+1}} = \beta_{k-1}^{\Sigma^m} - 1$$

Then, we consider the non-zero element f_j of $a^m_{\partial\sigma}$ with maximal index j. We now look for annotations of those $(k-1)$-simplices τ that have a non-zero element at index j and process them as follows. If the element of index j of a^m_τ is $f \neq 0$, we add $-\frac{f}{f_j} a^m_{\partial\sigma}$ to a^m_τ. As a result, $a^m_\tau[j]$ becomes 0 for each $(k-1)$-simplex τ. We define a^{m+1}_τ removing the j^{th} entry of the annotation vector of any $(k-1)$-simplex and set $g \leftarrow g - 1$. We assign to σ the null annotation vector $a^{m+1}_\sigma = 0$. According to Proposition 2.19, this is equivalent to removing the j^{th} cocycle ϕ_j.

As with the standard algorithm, persistent homology is derived from the creation and destruction of the cohomology basis elements.

Complexity and implementation. In order to actually implement this approach, efficient data structures encoding the simplicial complex Σ and the annotation vectors associated with each simplex of Σ have to be developed. For each dimension k, the k^{th} cohomology group can be seen as a valid annotation for the k-simplices of the simplicial complex. Hence, an annotation $a_k : \Sigma_k \to \mathbb{F}^g$ can be represented as a $g \times |\Sigma_k|$ matrix with



(a) (b)

Figure 2.11: The only two possibility of elementary inclusion of a 2-simplex. In (a), according to **Case 1**, the new simplex increases β_2. In (b), according to **Case 2**, the new simplex decreases β_1.

elements in $\mathbb{F}$, where each column is an annotation vector associated with a k-simplex. In most applications, the annotation matrix is sparse and has a lot of duplicate columns. In order to obtain an implementation with good performances, it is necessary to represent this annotation matrix in an efficient way called *compressed annotation matrix*.
A column is represented as the singly-linked list of its non-zero elements, where the list contains a pair (j, f) if the j^{th} element of the column is $f \neq 0$. The pairs in the list are ordered according to row index j. All pairs (j, f) with same row index j are linked in a doubly-linked list. To avoid storing duplicate columns, we use two data structures. The first one, $\mathcal{AV}$, stores the annotation vectors and allows fast search, insertion and deletion. $\mathcal{AV}$ can be implemented as a red-black tree or a hash table. The simplices of the same dimension that have the same annotation vector are now stored in a same set and the sets are stored in a union-find data structure.
Let d be the dimension of the simplicial complex Σ and n the number of its simplices. Let g_n and s_n be the maximal dimension of a cohomology group and the maximal number of distinct columns in the matrix, respectively, along the computation. The total cost for computing the persistent cohomology and the memory complexity for storing the compressed annotation matrices are respectively

$$O(n[\mathcal{C}_\partial^d + d(\alpha(n) + g_n) + s_n(g_n + \mathcal{C}_{\mathcal{AV}} + \alpha(n))]) \text{ and } O(n + dg_n s_n)$$

where $\mathcal{C}_\partial^d$ is the complexity of compute the boundary of a d-simplex, α is the inverse Ackermann function and $\mathcal{C}_{\mathcal{AV}}$ is the complexity of an operation in the data structure $\mathcal{AV}$ which encodes the annotation vectors. Specifically, implementing Σ using an Incidence Graph and the $\mathcal{AV}$ as hash tables, we get $\mathcal{C}_\partial^d = O(d)$ and $\mathcal{C}_{\mathcal{AV}} = O(g_n)$. If we consider $\alpha(n)$ as a small constant and remove it for readability, we get that the total cost for computing persistent cohomology is $O(ng_n(d + s_n))$. Experimental results show that g_n and s_n remain small in practice. Hence, the practical complexity of the algorithm is linear in n for a fixed dimension.

Persistent homology for simplicial maps
While persistent homology handles filtrations only consisting of inclusion maps, the main purpose of the algorithm proposed in [DFW12] is to generalize this situation in order to compute homology modification induced by an arbitrary simplicial map. In [DFW12], it has been proven that any simplicial map $f : \Sigma \to \Sigma'$ can be considered as a composition of elementary simplicial maps. Similarly to [BDM13], by studying how a valid annotation

can be consistently updated after the application of an elementary simplicial map, one can completely describe the homology modifications occurred from Σ to Σ' after the application of simplicial map $f : \Sigma \to \Sigma'$.

2.3.6 Software tools for homology and persistent homology computation

Several software tools for computing homology and persistence homology, have been developed and distributed in the public domain. In [OPT+15], a comparative analysis of such tools is presented. We refer to it for a complete discussion of such tools. The main goal of most of these tools is the retrieval of persistent homological information with respect to $\mathbb{Z}_p$ coefficients. The input of such tools is usually a simplicial complex or a regular grid endowed with a filtration or, alternatively, a set of points from which reconstruct a filtered complex. In most of the cases, the computation of standard homology is achieved by endowing the input complex with a trivial filtration and performing on it the algorithm to retrieve persistent homology.

We can classify software tools computing persistent homology as follows:

- tools based on matrix reduction algorithm,
- tools exploiting coarsening and pruning approaches,
- tools based on annotations.

The first class consists of tools which directly work with the boundary matrices of a cell complex. We need to mention here: *javaPlex* [TVJA12], *GAP persistence* [VJ12] and *Dionysus* [Mor12], which exploit the algorithms proposed in [ZC05] and the optimizations introduced in [DMVJ11, MMS11]; *PHAT* [BKRW14], which implements most of the optimization techniques proposed in literature, including the approach based on chunks [BKR14a] and the one using spectral sequences [LSVJ11]; *DIPHA* [BKR14b], which retrieves persistent homology groups through a distributed computation; *PHOM* [Tau11], written in R, that implements the standard [ZC05] and the dual [DMVJ11] algorithms.
The constant growth of the size of the data has led to the development of another class of tools focused, not on new optimizations of the matrix-based algorithm retrieving persistent homology, but, mainly, on a preliminary simplification step reducing the size of the input complex without altering its topological features. Such a strategy is adopted by the software libraries *RedHom* [JM14] and *Perseus* [Nan]. The first one is based on reduction and coreduction operators for the simplification step and it uses PHAT for the persistent homology computation. The second one preliminary reduces the size of the input complex via discrete Morse theory and then retrieves persistent homology by exploiting the algorithm proposed in [ZC05].
A third class of tools is based on the use of annotations. According to this strategy, *Gudhi* [MBGY14] and *SimpPers* [DFW12] compute persistent homology through annotation

vectors and encode a simplicial complex through the Simplex Tree data structure [BM12] instead of using the Incidence Graph as all the above mentioned libraries. Furthermore, the software Gudhi includes the implementation of the multi-field algorithm proposed in [BM14].

2.4 Algorithms rooted in Morse and discrete Morse theories

Morse theory allows to associated with a shape a complex leading the same topological information of the original object. In this section, we classify and describe the algorithms developed in the literature for building a Morse complex. This classification keeps in consideration various discretizations of Morse theory. Furthermore, since in many real situations the noise in the data leads to the computation of a Morse complex with a too large number of cells, we briefly introduce and discuss the techniques used in the literature to simplify it.

Most of the contributions of this section have been collected in two surveys [DFIM15, DFI15]. The first one gives a general and complete overview about algorithms rooted in Morse theory and in its discretizations [DFIM15]. The second one is mainly devoted to the discrete Morse theory and to the homological shape analysis [DFI15].

2.4.1 Classification

The algorithms for computing Morse complexes can be classified based on different criteria, such as the *input* they accept, the *output* they generate, and the *algorithmic approach* they apply (for a more detailed discussion see [ČDMI14]).

These algorithms may differ in:

- the dimension d of the shape on which a scalar field is defined (they may be specific for $d = 2$ or $d = 3$, or be dimension-independent);
- the discretization of the underlying shape (simplicial complex, regular grid);
- the required properties (e.g., no flat edges).

Algorithms may also differ in:

- the information computed (ascending/descending Morse complex, or Morse-Smale complex);
- the output format (e.g., the cells of such complexes are described as collections of vertices or d-cells of the original complex).

Based on the algorithmic technique they apply, we have the following classification.

- *Boundary-based* approaches, which are based on piecewise linear Morse theory, and compute the Morse-Smale complex by constructing the separatrix lines (in 2D), or the separatrix lines and surfaces (in 3D).
- *Region-growing* approaches, which are generally based on piecewise linear Morse theory, and compute the ascending (or descending) Morse complex by growing the top cells, here called *regions*, from seeds located at the minima (or maxima).
- *Watershed* approaches, which are based on the discrete watershed transform, and compute the ascending Morse complex.
- *Forman-based* approaches, which are based on Forman's discrete Morse theory.

2.4.2 Algorithms based on piecewise linear Morse theory and on watershed transform

In this subsection, we briefly describe the algorithms for computing a Morse complex based on the piecewise linear Morse theory and on the watershed transform. As mentioned above, we classify such algorithms into boundary-based, region-growing and watershed approaches.

2.4.2.1 Boundary-based algorithms

Boundary-based algorithms have been developed for triangle and tetrahedral meshes and for 2D and 3D regular grids. Such algorithms compute an approximation of the Morse-Smale complex through its 1-skeleton in 2D and through its 2-skeleton in 3D.

In the 2D case, a *boundary-based* algorithm generally performs two major steps:

1. *vertex classification* to extract maxima, minima, and saddles;
2. *path computation*: starting from each saddle, ascending separatrix lines are traced until they reach a maximum, and descending separatrix lines are traced until they reach a minimum.

In the smooth case, two ascending and two descending paths are incident in each saddle. In the discrete case, the number of ascending/descending paths in a saddle can be larger. A saddle of multiplicity equal to k has $k + 1$ ascending and $k + 1$ descending paths ($k + 1 = 2$ for simple saddles). Thus, the boundary-based algorithms unfold multiple saddles into simple saddles.

Path tracing should follow the steepest ascent/descent of the function. In the discrete case, this latter may be approximated in different ways. Takashahi et al. [TIKU95] move from the current vertex to its highest/lowest adjacent vertex. Edelsbrunner et al. [EHZ01] consider the edge slopes, and move from the current vertex v to its higher/lower adjacent vertex connected to v through the steepest edge. Such algorithms, which compute separatrix lines by following the edges of the triangle mesh, do not guarantee that traced lines

have the steepest ascent/descent. Bremer et al. [BEHP04] and Pascucci et al. [Pas04] consider the slope of both edges and triangles, and allow separatrix lines to cross triangles. Because tracing separatrix lines across triangles is computationally intensive, de Berg et al. [dBT11] present a hybrid approach which tries to balance computation time and precision.

The approach in [EHZ01] has been extended to 3D [EHNP03]. Given a tetrahedral mesh, the algorithm computes the 1- and 2-cells which bound the 3-cells in the Morse-Smale complex. The extracted complex has the correct combinatorial structure described by a quasi-Morse-Smale complex. Two sweeps are performed over the input mesh. The first sweep (by decreasing function value) computes the descending 1- and 2-cells, and the second sweep (by increasing function value) computes the ascending 1- and 2-cells.

Boundary-based algorithms on regular grids are based on the construction of interpolating functions within each top cell. Two conflicting issues arise: the need of avoiding, or limiting, the creation of new critical points, and that of guaranteeing a certain degree of continuity on the whole domain.

Bajaj et al. [BPS98] use a globally C^1-differentiable Bernstein-Bezier bi-cubic function for two-dimensional regular grids, and a tri-cubic function for 3D regular models. They define an algorithm to reduce the number of newly introduced critical points.

Schneider and Wood [SW04, Sch05] propose two methods for 2D regular grids. The first one uses a bilinear C^0 interpolating function on each top cell, which, thus, cannot introduce additional minima or maxima, but may introduce saddles. The second method uses a bi-quadratic approximation on each top cell. No new critical point is introduced, but the resulting approximation is globally discontinuous, formed by local surface patches.

All algorithms trace four separatrix lines from each saddle point, which can follow grid edges, or go through 2-cells. Both function derivatives and separatrix lines are computed numerically in [BPS98]. The two algorithms in [SW04, Sch05] compute derivatives analytically. The algorithm in [Sch05] uses this information to trace the separatrix lines, while the algorithm in [SW04] uses a step-by-step numerical procedure.

2.4.2.2 Region-growing algorithms

Region-growing algorithms operate on triangle or tetrahedral meshes, and compute the top cells of the ascending (descending) Morse complex, called *regions*, expanding an initial seed, i.e., the minimum (or maximum) the top cell is associated with. This class of algorithms is mostly based on piecewise-linear Morse theory.

Regions are collections of triangles (for 2D scalar fields) or tetrahedra (for 3D scalar fields) from the underlying mesh Σ: each triangle or tetrahedron $t \in \Sigma$ is labeled by the minimum (or maximum) associated with the region containing t. The only exception is the 3D algorithm by Gyulassy et al. [GNPH07]. We focus here on the computation of the descending Morse complex. The computation of the ascending Morse complex is completely dual.

In the two-dimensional case, a region-growing algorithm conceptually contains two major

steps:

1. *extraction of seed vertices*;
2. *region-growing*: from each seed, the corresponding region is built by iteratively adding triangles which are edge-adjacent to it.

In the two algorithms by Danovaro et al. [DDM03b, DDM$^+$03a], the two steps are interlaced: a seed is extracted and its region is immediately grown. Each new seed p is the vertex with maximum function value among the ones with unclassified incident triangles. The region γ of p is initialized with unclassified triangles incident in p and grown by adding edge-adjacent triangles. The algorithm in [DDM03b] decides on adding a triangle t based just on function values, whereas the one in [DDM$^+$03a] uses a discrete gradient defined over triangles. As extracted seeds include other vertices besides the maxima of f, a final merging of regions is required. The algorithm in [DDM03b] works for tetrahedral meshes as well, and can be extended to higher-dimensional scalar fields.

The algorithm in [MDD$^+$09] first extracts all seeds, which are the maxima of function f. Then, their regions are computed one at a time in any order. Similarly to [DDM03b], the criterion for adding a triangle t to γ is based just on function values, but it is less restrictive and permits to build the entire region of a maximum. This algorithm accepts triangulated surfaces with flat edges, by using ad-hoc solutions for handling plateaus.

The algorithm in [GNPH07] computes all the i-cells of the 3D Morse-Smale complex (for all dimensions $i = 0, 1, 2, 3$) as collections of vertices, i.e., through vertex labeling. First, minima are found and an ascending 3-cell is grown from each of them. Then, ascending 2-cells are built starting from boundary vertices of 3-cells. Finally, ascending 1-cells are built starting from boundary vertices of 2-cells. Descending 3-, 2-, and 1-cells (in this order) are computed inside the ascending 3-cells using the same approach in a symmetric way. This algorithm has been presented for 3D scalar fields, but it can be extended to higher dimensions.

2.4.2.3 Watershed algorithms

As described in Section 1.3.2.2, given a labeled graph $H = (N_H, A_H, f)$, watershed algorithms produce the ascending Morse complex as a classification of the nodes of H: each node p is labeled with the minimum corresponding to the ascending top cell containing p. Some methods produce so-called *watershed* nodes, which lie on the boundary between two or more top cells. Watershed algorithms work on regular grids of any dimension or on simplicial meshes. In the first case, they label the top cells of the complex. In the latter case, the nodes in N_H are the vertices of the underlying mesh Σ, the arcs in A_H are the edges of Σ, and the output is a vertex classification.

Watershed algorithms are usually based on *topographic distance* [Mey94], on *simulated immersion* [VS91, Soi04], or on *rain falling simulation* [MW99, SS00]

Watershed algorithms based on topographic distance directly apply the definition of catchment basin in terms of topographic distance and shortest paths (see Section 1.3.2.2). The

image integration algorithm by Meyer [MB90, Mey94] is a variation of the Dijkstra-Moore algorithm [Dij59] for computing the shortest path from a source node to every other node in a graph. In this case, the topographic distance is used. The *hill climbing* algorithm [Mey94] is a simplified version of image integration, which applies to regular grids, since the distance between two adjacent top cells p and q in domain space is constant.

The intuitive idea behind the *simulated immersion* approach [VS91, Soi04] is that of letting water raise from local minima, and label as watershed those nodes of graph H where water coming from different minima merge. The algorithm expands catchment basins by processing the nodes of H by increasing function value. In the stage in which a certain function value h is processed, all catchment basins of minima with value $h' < h$ have been started and, up to now, contain just nodes with function values lower than h. Processing function value h will add new nodes to existing basins, and will start new basins from minima having a function value equal to h.

Watershed approaches presented so far have in common the idea of growing catchment basins upwards from the minima of f. The *rain falling* paradigm [MW99, SS00] uses the opposite idea of letting water fall down from each vertex until it reaches at a minimum. The advantage of the rain-falling approach is that it does not require a preliminary sorting of the vertices, nor a priority queue. The main steps of the rain falling algorithm are:

1. find the minima and label each minimum as belonging to the basin of itself;
2. for each node v, which is still unlabeled, start descending from v to its lowest neighbor u, continue until an already labeled vertex u is found, and give the label of u to all traversed nodes.

The algorithm by Mangan and Whitaker [MW99] is for triangle meshes, and the one by Stoev and Strasser [SS00] for regular grids. An implementation of the rain-falling simulation for triangle meshes has been used in [MDI13], where triangles are then classified based on the labels of their vertices. A similar approach can also be applied for tetrahedral meshes [Iur14].

2.4.2.4 Analysis and comparison

Some algorithms applied to meshes consider both the field difference between two vertices, and their distance in domain space [EHZ01, BEHP04, Mey94]. Other algorithms simply consider field difference. The two boundary-based algorithms in [TIKU95, EHZ01], and the two region-growing ones in [DDM03b, DDM$^+$03a] just differ in this aspect. Note that the two approaches are equivalent on a regular grid.

Region-growing and watershed methods compute the top cells of the Morse complex through a classification of the top cells of the input complex, and, thus, their accuracy cannot go beyond the granularity of the input scalar field model. Boundary based algorithms exist which do the same, or trace separatrix lines also inside top cells, thus splitting an input triangle or pixel across several output top cells [BEHP03, Pas04]. These latter algorithms are less efficient, but avoid a number of violations, such as a path from a

saddle reaching another saddle before arriving at its final maximum or minimum, two overlapping paths in opposite directions, and regions bounded by extracted lines having a disconnected interior.

Boundary-based algorithms guarantee that saddles lie on the 1-cells of the output complex. As two paths going in the same direction join before reaching their final maximum or minimum, 2-cells bounded by such lines may be non-manifold because of so-called *dangling edges*. Region-growing methods suffer of the opposite problem. They build 2-cells that are 2-manifold by construction, but saddle vertices may not lie on their boundary.

Region-growing and watershed approaches do not extract the Morse-Smale complex, which can be obtained by intersecting the top cells of the Morse complexes. On the contrary, boundary-based approaches can easily produce the ascending or descending Morse complex by simply tracing just separatrix lines in one direction.

All algorithms based on locally steepest descent, including algorithms for computing the Forman gradient (see Subsection 2.4.3.1), do not converge to the ground truth smooth function when the underlying discrete domain is refined. Intuitively, the gradient of the input data is affected by a sampling error (due to the data points) and by a quantization error (due to the limited directions of the edges incident in each point). Although the sampling error can be decreased using a denser sampling, this is not the case for the quantization error. For watershed based algorithms, this problem has been studied in [SCP08] where an improved technique is proposed based on a probability propagation scheme. New techniques have also been developed for algorithms computing a Forman gradient [GBP12, RGH$^+$12].

2.4.3 Algorithms rooted in discrete Morse theory

Algorithms based on discrete Morse theory do not compute Morse, or Morse-Smale complexes explicitly, but compute a Forman gradient V from which all the cells of the Morse and Morse-Smale complexes can be extracted if needed. Such algorithms are purely combinatorial, dimension-independent and independent of the type of underlying cell complex. Note that algorithms based on piecewise linear Morse theory are dimension-specific, and specific for simplicial meshes or regular grids. Furthermore, algorithms based on discrete Morse theory can provide all the cells of both the Morse and the Morse-Smale complexes, whereas watershed algorithms compute only the Morse cells.

When working with a Forman gradient, all the Morse and Morse-Smale cells are obtained by traversing the V-paths of the gradient in a dimension-independent way. Some algorithms [RWS11, SMN12, SN12, WIFD13] have been defined for computing a Forman gradient on real world datasets and then are easily parallelizable or have been specifically developed for distributed computation. We can classify algorithms for Forman gradient computation into two categories: *constrained algorithms* [CCL03, KKM05, GBHP11, RWS11, GBP12, GRWH12, SMN12, SN12]; and *unconstrained algorithms* [LLT03, MB09, MW10, DKMW11, HMMN14, BL14].

Constrained algorithms start from a discrete scalar function f defined over the vertices of a cell complex Γ, and aim at constructing a Forman gradient that best fits function f.

They focus on extracting a minimum number of critical simplices in order to avoid spurious cells in the Morse complexes [RWS11, GBP12], or they perform a-posteriori simplifications to reduce them [KKM05, GBHP11]. The typical applications for constrained algorithms are data analysis and visualization, since a Forman gradient provides a computationally efficient way for extracting the Morse and Morse-Smale cells representing the regions of influence of the critical points.

Unconstrained algorithms, on the other hand, compute a Forman gradient on a cell complex when no information about the Forman function is provided. Here, a Forman gradient is extracted with the aim of computing homology and persistent homology of the original complex. The focus of these algorithms is computing a pairing for all the cells of the input cell complex while leaving as few unpaired cells as possible. In [LLT04], Lewiner et al. presented the first algorithm of this type, and showed applications to topology visualization and mesh compression.

A challenging problem, when working with a Forman gradient V, is traversing the V-paths in order to extract information about the Morse or Morse-Smale complexes. In the case of constrained algorithms, traversing the V-paths corresponds to building a segmentation of the underlying shape into Morse or Morse-Smale complexes. In an unconstrained algorithm, V-paths are used for computing homology or homology generators of the underlying cell complex. In 2D, all V-paths can be visited in linear time by traversing all cell pairs at most once. However, for extracting only separatrix V-paths, the gradient paths between saddles and maxima can be visited in reverse order, thus reducing the numbers of pairs to be visited. In higher dimensions the situation is more involved. We will describe in details the algorithms proposed for the efficient extraction of Morse and Morse-Smale complexes from a Forman gradient [GBP12, GRWH12, SN12, WIFD13] in Subsection 2.4.3.2.

Both constrained and unconstrained algorithms are involved, with a different flavor, in persistent homology computation. In constrained approaches, the values of function f at the vertices of cell complex Γ naturally induce a filtration of Γ. In [RWS11], for example, the generic element Γ^m of the filtration induced by the input function f on Γ is the cell complex containing all the cells of Γ that have no vertex with a function value greater than m. This choice guarantees that each cell of the discrete Morse complex corresponds to a change in topology between successive cell complexes of the filtration. In unconstrained approaches, the filtration is set as input, and the construction of the Forman gradient has to comply with the given filtration [MN13, DW$^+$14].

2.4.3.1 Computing the discrete Morse gradient

In this subsection, we present first an encoding for the discrete Morse gradient field for a simplicial complex and then we describe algorithms for computing it.

Forman gradient encoding on simplicial complexes

Since a Forman gradient field is a collection of pairs of cells on a cell complex Γ, we need a representation for Γ in which all cells are explicitly encoded as well as their mutual incidence relations, as in the Incidence Graph (IG) (see Subsection 2.1). The Forman

gradient can be easily implemented on the IG as a Boolean function associated with its arcs. For a regular grid, all such relations are encoded implicitly by indexing the cells of the grid. Moreover, since a Forman gradient V defines a pairing between incident cells, V can be defined as a bit vector based on the same indexing [GRWH12].

For simplicial complexes, a data structure encoding all simplices and their incidence relations is too verbose. Other data structures exist for simplicial complexes, which encode only vertices and top simplices [DH05, CDW11], thus being much more compact, and scalable with the dimension. For simplicial complexes of dimension 2 and 3, an encoding for the Forman gradient on such data structures has been developed [FlDFW14, WIFD13]. It is called a *compact gradient* and associates the gradient pairs to the top simplices. The use of such data structures with the compact gradient makes the computation of the Forman gradient and of the Morse and Morse-Smale complexes feasible on simplicial complexes on large size. The compact encoding associates with σ a subset of the pairs involving its faces, namely, all pairs in the discrete vector involving σ or two of its boundary simplices. In 2D, a triangle has 12 possible pairs for a total of $2^{12} = 4,096$ cases. However, for a Forman gradient, there are only 97 valid cases for a triangle. Thus, all possible configurations can be encoded by using only one byte per triangle. Similarly, in 3D, there are 32 pairs for a total of $2^{32} = 4,294,967,296$ possible configurations, and the valid ones are only $51,030$, thus they can be represented with 2 bytes per tetrahedron. We refer to [FlDFW14, WIFD13] for details.

Constrained algorithms

Several algorithm have been recently developed in the literature to compute a Forman gradient when a scalar value is given at the vertices of a cell complex [CCL03, KKM05, GBHP11, RWS11, GBP12, GRWH12, SMN12, SN12]. Such algorithms have been generally developed in 2D or 3D, and mainly on regular grids.

The algorithm proposed in [CCL03] builds a Forman gradient from a triangle mesh Σ endowed with a scalar function, which in this case is the discrete Connolly's function f [Con86]. Function values are extended to edges and triangles as mean value of their incident vertices. They consider the *primal graph* H of Σ, which is the graph having as nodes the vertices of Σ and as arcs its edges, and the *dual graph* H_D of Σ defined as the graph having as nodes the triangles of Σ and the arcs in one-to-one correspondence with edge-adjacent triangles. A spanning forest T_D is created by building a spanning tree on H_D for each local maximum of f on Σ, processing the edges by increasing function value. In a dual fashion a spanning forest T is built on H by creating a spanning tree for each local minimum. The resulting Forman gradient V is computed considering T and T_D. Roots of T are the minima (vertices), roots of T_D are the maxima (triangles) and edges that do not belong to either T and T_D are the saddles. The V-paths of V formed by vertices and edges correspond to paths from a root of T to one of its leaves, while V-paths formed by edges and triangles correspond to the paths from a root of T' to one of its leaves. The algorithm can be extended to d-dimensional complexes but only by restricting to the computation of the pairings between 0-simplices and 1-simplices (forming V-paths connecting minima to 1-saddles) and between $(d-1)$-simplices and d-simplices (forming V-paths connecting maxima to $(d-1)$-saddles).

The algorithm proposed in [KKM05] takes as input a scalar function f defined over the vertices of a 3-dimensional simplicial complex Σ. The algorithm builds the gradient vector field in the lower link $\mathrm{link}^- v$ of each vertex v in Σ, where the *lower link* $\mathrm{link}^- \sigma$ of a simplex σ is the subset of the link of σ containing only simplices with a function value less than or equal to the function value of σ. Then, it extends this function to the cone $(v; \mathrm{link}^- v)$, which is the simplex generated by the union of the vertices of v and $\mathrm{link}^- v$. The Forman gradient computed in this way may have an arbitrary large number of critical simplices compared with the number of actual critical points of the original scalar function f. Thus, the algorithm performs a simplification step for reducing the number of critical cells [For98].

The algorithm proposed by Gyulassy et al. in [GBHP08] is one of the first algorithms defined in a dimension independent way. It computes a Forman gradient starting from a d-dimensional regular grid H with scalar function f defined at the vertices of H. Function f is extended to a Forman function F, defined on all cells of H, such that $F(\eta')$ is slightly larger than $F(\eta)$ for each cell η' and each face η of η'. For such function F, all cells of H are critical. A gradient vector field is computed by assigning gradient arrows in a greedy manner during sweeps over the cells of H according to increasing values of dimension and of F. Each current non-paired and non-critical cell in the sweep is paired with its coface having only one face not marked (as critical or as already paired). If there are several of such cofaces the lowest one is taken. If there is no such coface, the cell is critical. Pairs built in this way define a gradient vector field. The order in which the cells in H are processed is not deterministic, since different k-cells in H may have the same value of function F. As a consequence, some unnecessary critical cells may be produced by the algorithm. In [SMN12] and [SN12], a similar approach based on a *weighted discrete function* has been defined for computing a Forman gradient on 2D and 3D regular grids, respectively. The pairs found by the algorithm are unique and independent of the order in which the cells are considered, thus providing a basis for a parallelization of the algorithm. In [GBP12], a similar algorithm is proposed which focuses on improving the poor geometric approximation of the gradient caused by the local assignment of the gradient arrows. This is especially useful in scalar field analysis, but not for homology computation.

In [RWS11], a dimension-independent algorithm is proposed for a regular grid with scalar function values given at its vertices. In [FlDFW14, WIFD13], this algorithm has been implemented for simplicial complexes in 2D and in 3D by developing very compact representations for the underlying complex, leading to the first efficient algorithm for Forman gradient computation on simplicial complexes. The algorithm processes the lower star of each vertex v in Γ independently, where the *lower star* $\mathrm{star}^- p$ of a cell p is the subset of the star of p containing only cells with a function value less than or equal to the function value of p. Each cell p is considered in ascending order of function values and of dimension, such that each cell p is considered after its faces. All the k-cells incident in the lower star are paired via homotopy expansion. Two cells, k-cell p and $(k+1)$-cell q, are paired via homotopy expansion when: p have no unpaired boundary cells and q has

only one unpaired boundary cell (i.e., p).

Table 2.2 summarizes the algorithms discussed in this section. The only dimension-specific algorithms are the one in [CCL03] specifically defined for 2D simplicial complexes, and the one in [KKM05]. The gradient computation in the algorithm by Kings et al. [KKM05] could be extended to higher dimensions but the simplification step could be problematic in higher dimensions. All of them are implemented for specific complexes (regular grids [GBHP08, RWS11] or simplicial complexes [CCL03, KKM05]). Algorithms implemented for regular grids are typically used for the analysis of gridded volume datasets. However, since they all rely on discrete Morse theory, they can be easily adapted to cell complexes.

Algorithm	Input	Time Complexity
Cazals et al. [CCL03]	2D simplicial complex Σ	$O(\|\Sigma\|(log\|\Sigma\| + \alpha(\|\Sigma\|)))$
King et al. [KKM05]	3D simplicial complex Σ	$O(\|\Sigma_0\|s)$
Gyulassy et al. [GBHP08]	nD cell complex Γ	$O(\|\Gamma\|log\|\Gamma\|)$
Robins et al. [RWS11]	nD cell complex Γ	$O(\|\Gamma_0\|c)$

Table 2.2: Summary of the reviewed algorithms. For each of them the expected input and the worst time complexity are indicated. Note that $|X|$ denotes the cardinality of set X, and X_0 is the set of the vertices of X.

We can classify the algorithms described above in two groups based on their time complexity. Some of them [KKM05, RWS11] are based on an implicit subdivision of the cells of the complex into independent sets (based on the vertices). Both algorithms consider each cell, in the independent set, exactly once. Since all the operations performed are in constant time, the complexity is only dependent on the number of simplices s [cells c] in each independent set. In most of applications, s and c are considered negligible with respect to the number of vertices v and thus the complexity of the entire process is considered linear. The algorithms in [CCL03, GBHP08, GBP12], instead, require as initial step a sorting of the simplices. In [CCL03], all the edges are sorted (with $O(|\Sigma|log|\Sigma|)$ complexity) and a further step, for the forest creation, is performed in $O(|\Sigma|\alpha(|\Sigma|))$ with α the inverse of Ackerman's function. Algorithms in [GBHP08, GBP12] sort the cells based on the Forman function defined starting from the function defined on the vertices.

Unconstrained algorithms

Several algorithms have been proposed for computing a Forman gradient on a cell complex without any constraint such as a scalar value at the vertices of the complex.

The algorithm by Lewiner et al. [LLT03] has been the first algorithm of this kind proposed in the literature with the aim of providing a combinatorial descriptor for 3D shapes. It has been defined on triangle meshes and then extended to 2-dimensional cell complexes Γ. The algorithm is similar to the one described in [CCL03], but here the spanning forests are built by considering a spanning tree for each connected component of the shape, without ordering the edges of Γ based on a function value.

Several algorithms have been proposed based on two simplification operators, called *reduction* and *coreduction* respectively, which delete a pair of cells from a cell complex Γ while preserving the homology groups of Γ (see Subsection 2.3.3.2). Even if these two operators have been defined as simplification operators, they can be seen as pairing operators for building a Forman gradient V on Γ. Once Γ has been fully simplified, obtaining a simplified cell complex Γ_S, the gradient V corresponds to the cell pairs eliminated and the cells in Γ_S to the critical cells of V. Here, we describe these algorithms presenting a dual strategy for computing the Forman gradient [HMM$^+$10, BL14].

In [BL14], an algorithm based on the reduction operator has been proposed for the construction of a Forman gradient. The algorithm builds a gradient vector field V on a cell complex Γ by using reduction pairs and removals of top cells. The algorithm works as follows. V is initialized as the null set. Let us denote as Γ' the set of non-excised cells of Γ and initialize it as Γ. While Γ' admits a reduction pair (p, q), the algorithm excises cells p and q from Γ' and adds the pair (p, q) to set V. When no more reduction is feasible, a top cell is excised from Γ', which becomes a critical cell. The main loop of the algorithm is iterated until Γ' is empty.

The algorithm proposed in [HMM$^+$10, HMMN14] also combines a homology-preserving operator with the construction of a Forman gradient on a cell complex. In this case, the operator involved in the computation of the Forman gradient is *coreduction*. A coreduction [MB09] can be viewed as the dual with respect to a reduction and the two operators combined represent a powerful preprocessing tool to efficiently compute the homology of a cell complex, as described in [MB09, MW10, DKMW11].

In [HMMN14], a Forman gradient is built by removing coreduction pairs and free cells, where a *free cell* is a cell with an empty boundary. In this approach a gradient vector field V on Γ is built as follow. The algorithm looks for available coreduction pairs in the complex. While a coreduction pair is feasible, the pair is excised from the cell complex and it is added to V. When no more coreduction pair is available, a free cell is excised from the complex and considered as a critical cell. The algorithm iterates this process until no more cells can be considered.
Accurate analysis and comparisons of the two approaches have been performed in the context of simplicial complexes in [FID14] and they will be presented in Chapter 3.

2.4.3.2 Computing and representing the discrete Morse complex

The *Morse Incidence Graph* (MIG) [ELZ02, BEHP04, GKK$^+$12, ČomićDI13] is an efficient graph-based representation for a Morse complex. The MIG associated with a d-dimensional descending and ascending Morse complexes is a graph $G = (N, E, \mu)$. The set of nodes N is partitioned into $d + 1$ subsets $N_0, N_1, \ldots, N_d$, such that there is a one-to-one correspondence between nodes in N_k (alsa called k-nodes) and critical k-cells V. There is an arc joining an k-node σ with a $(k + 1)$-node τ if and only if there is a V-path connecting k-saddle σ to $(k + 1)$-saddle τ. Each arc connecting a k-node σ to an $(k+1)$-node τ is labeled by the number of V-paths connecting τ to σ. The label, denoted as $\mu(\tau, \sigma)$, is also called the *multiplicity* of the arc (σ, τ).

In the applications, attributes are attached to the nodes in N and arcs in E storing the geometric information associated with the Morse cells. In [BEHP04, GKK+12], geometrical entities of both the ascending and descending Morse cells are explicitly stored, associated with the nodes in N, as well as the 1-cells forming the 1-skeleton of the Morse-Smale complex associated with each arc of E. In [ČomićDI13], only a subset of the Morse cells geometries are stored, namely the descending [ascending] d-cell for each node in N_d [N_0].

The incidence relations in the MIG are computed by traversing the V-paths of the *compact gradient* V defined on Γ, computing all the Morse cells in one of the two complexes, creating one node for each critical cell and connecting two nodes in the graph with an arc if there is a V-path in V connecting the two corresponding critical cells [WIFD13]. Since only V-paths are needed, ad-hoc strategies can be used to reduce the number of cells traversed. In 2D, the set of V-paths between saddles and minima are visited by starting from each critical 1-cell and following the gradient paths until a minimum is reached. Such paths never branch and, thus, a limited number of 1-cells are visited in practice during their traversal.

The extraction of such subgraphs (also called *extremum graphs*) is performed in a dimension-independent way, leading to the same reduction in complexity. In three and higher dimension, a new step is introduced to compute the saddle connectors, i.e., the arcs of the MIG between k-saddles σ and $(k+1)$-saddles τ, with $k \neq 0, d$. All the gradient paths starting from τ are considered, and all the traversed $(k+1)$-saddles are marked as visited. Then, starting from σ the same process is performed visiting the gradient paths in reverse order and considering only those paths passing by the $(k+1)$-cells previously marked as visited are considered.

In three and higher dimensions gradient paths can branch and merge potentially resulting in many-to-many adjacency relationships between critical k-cells and critical $(k+1)$-cells. Let us consider a simplicial 3-complex Σ with n_0 vertices, whose Forman function contains $O(n_0)$ critical 1-cells, each of which connects to $O(n_0)$ critical 2-cells. This produces a discrete Morse complex containing $O(n_0^2)$ gradient paths between critical 1 and 2-simplices. Since the number of critical 1- and 2-cells is bounded by v, the number of traversals for any cell during the breadth-first search is also bounded by n_0 and so the complexity of the whole extraction process is $O(n_0^3)$. In the literature, some solutions have been proposed to reduce the time required for extracting Morse cells between 1-saddles and 2-saddles [GRWH12, SN12, WIFD13]. In [WIFD13, GRWH12], the space complexity of the algorithm is slightly increased for saving the k-cells visited during a gradient path traversal. In this way, no cell is visited more than once and the time complexity drops to $O(n_0^2)$. This method works well when the computation of the saddle connectors is sequential, but the memory increase becomes too high for a parallel implementation. In [SN12], an algorithm is proposed, specific for parallel computation, with an implementation on 3D regular grids. The algorithm is based on a priority queue which allows counting the number of times a cell is visited, i.e., each cell is inserted in the queue only a constant number of times and the complexity of the resulting algorithm has been proven

to be $O(n_0^2 \log n_0)$.

2.4.4 Simplification of Morse and Morse-Smale complexes

Due to the huge size of available data sets and to the presence of noise, morphological descriptions of scalar fields may contain many uninteresting features. Thus, a fundamental issue is the simplification of such descriptions.
We give here a brief overview on various simplification algorithms proposed in the literature and we refer to Chapter 5 to a complete discussion about the problems affecting the simplification process.

In the literature, several strategies have been proposed for simplifying the morphological representation of a dataset [BDF$^+$08, DFIM15]. We refer to [Iur14] for a detailed discussion. The problem of simplifying a Morse-Smale complex has been addressed in 2D [EHZ01, BEHP04, WG09], 3D [GKK$^+$12, GRWH12] and in nD [ČomićD11].
Each simplification algorithm consists in the iterated application of a simplification operator. The most common simplification operator, called *cancellation*, has been defined in Morse theory for removing pairs of critical points connected by a unique separatrix line [Mat02]. A discrete counterpart of this operator, eliminating pairs of critical cells connected by a unique separatrix V-path, has been introduced in [For98].
A common characteristic of all simplification algorithms is the ordering of the available simplifications based on a priority schema. Priority measures the importance of pairs of critical points which are candidate for deletion, and is defined in such a way to cause the removal of less important critical pairs first. Algorithms have been proposed based on different priority measures. *Persistence* [ELZ02] is the most widely used; it estimates the importance of a pair of critical points according to the absolute difference of function values between the two points. More recently, other methods for measuring the importance of pairs of critical points have been proposed with the purpose of taking into account also the geometry of the underlying complex, namely *separatrix persistence* [WG09, GSW12] and *topological saliency* [DSNW13].

We distinguish between two types of algorithms for simplifying a Morse complex: algorithms working on a graph-based representation of the complex [BEHP04, GKK$^+$12, ČomićDI13] (also called *explicit methods*), like the Morse Incidence Graph, and algorithms based on the Forman gradient [WG09, GRWH12, FlDFW14] (also called *implicit methods*). All algorithms for 2D scalar fields are equivalent in the sense that they can produce the same simplification sequence, and the resulting simplification process is monotonic, i.e., after each simplification, all the new simplifications have higher persistence value. Differences arise when working in three or higher dimensions (see [GRSW13] and Chapter 5).

In [BEHP04, GKK$^+$12], two data structures have been defined implementing the graph representation for triangle meshes and for regular grids, respectively. In both cases, geometric attributes of the Morse cells and of the 1-skeleton of the MS complex are explicitly

encoded (i.e., vertices, edges, triangles and voxels forming such cells). Simplifications are performed by deleting nodes in pairs and merging together the geometrical representations of the Morse cells. In [ČomićDI13], a lightweight version of the same structure has been used encoding only the d- and 0-cells of the two Morse complexes, all the other cells being retrieved by intersection. However, since this latter operation is particularly time-consuming, the resulting data structure is less significative for practical usage.

Algorithms defined in [WG09, FlDFW14, GRWH12] take full advantage of the Forman gradient for defining a simplification algorithm with a low storage consumption. In [WG09, FlDFW14], simplifications are performed on the Forman gradient defined on a 2D regular grid and on a 2D simplicial complex, respectively. In [GRWH12], a similar simplification algorithm is implemented for the Forman gradient defined on a 3D regular grid. Due to the above mentioned inconsistency problem, that we closely discuss in Chapter 5, the incidence relations among the critical simplices need to be locally recomputed after each simplification.

2.5 Concluding remarks

In this chapter, we have provided a complete overview on several topics related to the work developed in this thesis. The comparisons and the considerations presented in this chapter have played a fundamental role in the development of the remainder of the thesis.

The analysis of different topological data structures (Section 2.1) has led us to consider data structures encoding only vertices and top simplices when dealing with high-dimensional simplicial complexes [FID14, FIDon]. The brief overview on multi-resolution models (Section 2.2) has taught us to manage differently the geometrical and morphological modifications in a hierarchical model.
The analysis of the algorithms for retrieving standard and persistent homology (Section 2.3) and for computing Morse complexes (Section 2.4) has provided us with several tools useful for the entire research.

An overview on software tools for the computation of standard and persistent homology has been presented in [OPT$^+$15]. A survey concerning the various algorithmic approaches and strategies to retrieve homological information is lacking in the literature. The classification proposed in Section 2.3 represents a step in this direction. In the near future, we plan to further develop such analysis and to collect its contribution in a survey paper.

The overview on Morse theories presented in Section 2.4 has led to [DFIM15, DFI15]. Note that Section 2.4 does not deal with the investigation of Morse functions having a minimal number of critical cells. This topic will be considered and discussed from an algebraic point of view in Chapter 6.

Chapter 3

Homology Computation through Discrete Morse Theory

As mentioned in the introduction, recently there is a growing need to handle high-dimensional data originated from huge and unorganized clouds of points and to efficiently extract topological information from them [OPT+15]. In this context, several mathematical tools representing simplicial complexes of large size thanks to a topologically-equivalent and more compact object, such as the discrete Morse complex [For98], the size graph [FP99] and the tidy set [Zom10b], have been proposed.
Discrete Morse theory is a very powerful and adaptable tool. It is theoretically independent from the dimension of the complex and it provides a complex whose size is smaller than the original one but still preserving its topological features. In the literature, the importance of discrete Morse theory has been acknowledged in various application domains such as scalar field and function analysis [DFIM15] and computation of standard and persistent homology [RWS11, HMMN14, HMM+10]. The purpose of this chapter is to develop tools for efficiently computing a gradient vector field of a simplicial complex retrieving the associated discrete Morse complex and to show some applications, such as persistent homology computation, in which the knowledge of the discrete Morse complex reveals to be useful. To achieve this goal, connections between the effect of homology-preserving operators and the creation of gradient of a discrete Morse function have been studied and used for efficiently retrieving the topological information of a simplicial complex. Given the huge size of the complexes to be handled, the need of an efficient and compact data structure able to encode them and their associated gradient vector field naturally arises.
This chapter tries to give an answer to these questions by

- describing, introducing and formally comparing various methods to endow a simplicial complex with a gradient vector field;
- encoding the obtained discrete Morse complex thanks to efficient, compact and dimension-independent representations for the Forman gradient and for the underlying simplicial complex;
- developing and implementing an algorithm to build a discrete Morse complex based

on the above methods and data structures;

- applying the proposed algorithm to efficiently retrieve the persistent homology of the original simplicial complex.

The contributions and the results described in this chapter are presented in [FID14, FI-Don].

3.1 Discrete Morse complexes through reductions and coreductions

Given a simplicial complex Σ, a Forman gradient on Σ can be built by using some homology-preserving operators. In this section, we present two methods exploiting reductions and coreductions to retrieve a gradient vector field (see Subsection 2.3.3.2 or refer to [HMMN14, BL14]). Furthermore, we propose another strategy to built a Forman gradient through homology-preserving operators and we provide a theoretical comparison about all these techniques.

3.1.1 Using coreduction sequences or reduction sequences

In this subsection, we briefly recall the definitions of reduction and coreduction and we describe two unconstrained approaches exploiting these operators to build a gradient vector field. As mentioned in Subsection 2.3.3.2, reduction and coreduction operators can be used in a preprocessing approach to compute homology or persistent homology of a simplicial complex, as described in [MB09, MW10, DKMW11, HMMN14, MN13].

Reductions and *coreductions* represent two operators for reducing the size of an S-complex without affecting its homology. Since we will work only with simplicial complexes, we can consider for simplicity an S-complex as a simplicial complex in which some simplices may be not present even if their cofaces are in the complex. Given a simplicial complex Σ (where eventually some simplices have been removed), a pair (σ, τ) of simplices of Σ, such that $\langle \partial\tau, \sigma \rangle = \pm 1$, is called:

- a *reduction pair* if $\mathrm{cbd}_\Sigma\, \sigma = \{\tau\}$,
- a *coreduction pair* if $\mathrm{bd}_\Sigma\, \tau = \{\sigma\}$,

where $\langle \partial\tau, \sigma \rangle$ represents the coefficient of incidence between τ and σ, $\mathrm{cbd}_\Sigma\, \sigma$ consists of the simplices in the immediate coboundary of σ and $\mathrm{bd}_\Sigma\, \tau$ consists of the simplices in the immediate boundary of τ.
Further, we recall the notion of top simplex. We say that a simplex ρ of Σ is a *top simplex* if it is not a proper face of any simplex of Σ. Equivalently, ρ is a top simplex if its coboundary $\mathrm{cbd}_\Sigma\, \rho$ is empty. Dually, ρ is called a *free simplex* if it is not a proper coface of any simplex of Σ or, equivalently, if its boundary $\mathrm{bd}_\Sigma\, \rho$ is empty.

Reduction and coreduction pairs can be successfully used also in the context of discrete Morse theory. Two unconstrained algorithms performing either sequences of coreductions or of reductions to obtain a gradient vector field have been developed in the literature (see Subsection 2.4.3.1). Here, we focus on them describing both algorithms in details. The basic scheme of both algorithms is presented in Algorithm 1.

Algorithm 1 Coreduction-based [reduction-based] algorithm (basic scheme)

INPUT: Σ, simplicial complex
OUTPUT: V, gradient vector field; $\mathcal{A}$, set of critical simplices
$\Sigma' := \Sigma$
$V := \emptyset$
$\mathcal{A} := \emptyset$
while $\Sigma' ! = \emptyset$ **do**
  while Σ' admits a coreduction [reduction] pair (σ, τ) **do**
    $V := V \cup \{(\sigma, \tau)\}$
    $\Sigma' := \Sigma' \setminus \{\sigma, \tau\}$
  Let ρ be a free [top] simplex of Σ'
  $\mathcal{A} := \mathcal{A} \cup \{\rho\}$
  $\Sigma' := \Sigma' \setminus \{\rho\}$

The approach in [HMMN14] is based on the construction of a Forman gradient on a simplicial complex Σ by using coreduction pairs and removals of free simplices. According to Algorithm 1, in order to obtain a Forman gradient V on Σ, the approach in [HMMN14] initializes first V as null and the set of non-excised simplices Σ' as Σ. While Σ' admits a coreduction pair, the algorithm excises a coreduction pair (σ, τ) from Σ' and adds it to V. When no more coreduction is feasible, a free simplex is excised from the complex and labelled as critical. The algorithm repeats these steps until no coreduction can be performed and set Σ' is empty.

The approach in [BL14] is based on the construction of a Forman gradient on a simplicial complex Σ by using reduction pairs and removals of top simplices. This method is dual with respect to the *coreduction-based algorithm*. As described in Algorithm 1, it follows the same pattern as the previous one, but performing and removing reduction pairs and top simplices instead of coreduction pairs and free simplices. While the set of non-excised simplices Σ' admits a reduction pair, the algorithm excises a reduction pair from Σ' and adds it to V. When no more reduction is feasible, a top simplex is excised from the complex and labelled as critical. Then, the algorithm repeats this process until the set Σ' of the remaining simplices of Σ is empty.

The two approaches have been developed in very different contexts with different tasks. The first method belongs to a large project whose purpose is the standard and persistent homology computation and leading to the implementation of the software *Perseus* [Nan]. On the other hand, the main goal of the approach proposed in [BL14] is, thanks to a random algorithm performed a huge number of times, to test the goodness of the triangulation of Σ and to provide a support to the validation of new theoretical results. For both methods it has been proven that the discrete vector field V produced on Σ is free of closed path and, so, it is a gradient vector field.

In order to minimize the size of the resulting discrete Morse complex, in both approaches the creation of a critical simplex is performed only if no more coreduction pair or reduction pair is feasible. Actually, even if this condition is not satisfied, the acyclicity of the obtained gradient paths is still guaranteed. In the following, we refer to this two approaches, also in the case in which critical simplices can be created when it is not strictly necessary, as *coreduction-based algorithm* and *reduction-based algorithm*, respectively.

As described in Subsection 2.3.3.4, an improvement in persistent homology computation can be achieved through the construction of a particular Forman gradient called *filtered gradient vector field*. Let Σ be a simplicial complex and let $F = \{\Sigma^m \,|\, 0 \leq m \leq M\}$ be a filtration of Σ. A Forman gradient V of Σ is called *filtered gradient vector field* if, for each pair $(\sigma, \tau) \in V$ there esists $m \in \{1, \ldots, M\}$ such that σ, $\tau \in \Sigma^m$ and σ, $\tau \notin \Sigma^{m-1}$. Coreduction-based and reduction based algorithms can be easily adapted to the computation of a filtered gradient vector field with respect to a filtration F of a simplicial complex Σ. In this context, both approaches maintain the basic structure described in Algorithm 1. Unlike the standard case, a reduction or a coreduction pair (σ, τ) can be elected as a pair for the filtered gradient vector field only if the simplices σ and τ appear at the same step of filtration F.

3.1.2 Equivalence of reduction and coreduction sequences

In this subsection, we prove the equivalence between the use of reduction and coreduction operators in the construction of a (filtered) gradient vector field and we introduce a class of methods which could operate reductions and coreductions in an interleaved way. To this aim, we need some preliminary results which help us to understand how the removal of a coreduction or of a reduction pair affects the coboundary and the boundary of the simplices of a simplicial complex.

Remark 3.1. Let τ be a simplex and let σ be one of its faces. Then, there exist $\dim \tau - \dim \sigma$ faces of τ in $\mathrm{cbd}_\tau \sigma$.

Lemma 3.2. *In a coreduction-based algorithm, each removal operation does not modify the coboundary of the remaining simplices.*

Proof. Let Σ be a simplicial complex on which the coreduction-based algorithm is executed. Clearly, the removal of a free simplex does not modify the coboundary of any of the remaining simplices. Let us consider only removals of coreduction pairs. Let (σ, τ) be a feasible coreduction pair in the set of non-removed simplices Σ'. The only simplices whose coboundary can be modified by the coreduction pair are those belonging to $\mathrm{bd}_{\Sigma'} \tau$ and to $\mathrm{bd}_{\Sigma'} \sigma$. Since for the feasible coreduction pair (σ, τ) $\mathrm{bd}_{\Sigma'} \tau = \{\sigma\}$, we just need to prove that before performing the coreduction $\mathrm{bd}_{\Sigma'} \sigma = \emptyset$. Suppose that exists $\nu \in \mathrm{bd}_{\Sigma'} \sigma$. By Remark 3.1, there exists in Σ $\sigma' \neq \sigma$ such that $\sigma' \in \mathrm{bd}_\Sigma \tau$ and $\nu \in \mathrm{bd}_\Sigma \sigma'$. Since (σ, τ) is a feasible coreduction pair in Σ', simplex σ' must have been already removed, i.e., $\sigma' \notin \Sigma'$. Let us proceed by induction. If (σ, τ) is the first coreduction pair performed in the coreduction-based algorithm on Σ, then σ' has been removed as a free simplex, but, since $\nu \in \mathrm{bd}_\Sigma \sigma'$ and $\nu \in \Sigma'$, this leads to a contradiction.

Assume now that, for any removal of a coreduction pair performed before (σ, τ), the simplex of lower dimension in the pair was free. Since $\nu \in \mathrm{bd}_\Sigma \sigma'$ and $\nu \in \Sigma'$, σ' cannot be removed as a free simplex or by a coreduction pair removal of the kind (ν', σ'). So, σ' has been removed by operating a coreduction pair removal of the kind (σ', τ'), but this contradicts the inductive hypothesis. □

Lemma 3.3. *In a reduction-based algorithm, each removal operation does not modify the boundary of the remaining simplices.*

Proof. Let Σ be a simplicial complex on which the reduction-based algorithm is executed. Clearly, the removal of a top simplex does not modify the boundary of any remaining simplex. Let us consider only removals of reduction pairs. Let (σ, τ) be a feasible reduction pair in the set of non-removed simplices Σ'. As in Lemma 3.2, it is sufficient to prove that, before performing the coreduction, $\mathrm{cbd}_{\Sigma'} \tau = \emptyset$. If there exists $\nu \in \mathrm{cbd}_{\Sigma'} \tau$, then, by Remark 3.1, there exists $\dim \nu - \dim \sigma \geq 2$ faces of ν in $\mathrm{cbd}_{\Sigma'} \sigma$. But this leads to a contradiction because (σ, τ) is a reduction pair and, thus, $\# \mathrm{cbd}_{\Sigma'} \sigma = 1$. □

We are now ready to formalize and to prove the equivalence between the coreduction-based and reduction-based algorithms.

Proposition 3.4. *Given a simplicial complex Σ and the gradient vector field V produced by a reduction-based algorithm, it is always possible to obtain the same gradient vector field through a coreduction-based algorithm. The converse is also true.*

Proof. For the sake of brevity, we only prove that the Forman gradient produced by a reduction-based algorithm on Σ can be obtained with a coreduction-based algorithm. The proof of the converse is similar (by using Lemma 3.2). Let Σ be a simplicial complex and let

$$R_1^1, R_2^1, \ldots, R_{i_1}^1, R_1^2, R_2^2, \ldots, R_{i_2}^2, \ldots, R_1^n, R_2^n, \ldots, R_{i_n}^n \tag{3.4.1}$$

be the ordered sequence of reduction pairs and top simplices removed during the execution of a reduction-based algorithm, where, for $1 \leq l \leq n$ and $1 \leq j \leq i_l - 1$, R_j^l represents a reduction pair and, for each $1 \leq l \leq n$, $R_{i_l}^l$ represents a top simplex removal (for an example, see Figure 3.1).

We want to prove that, by using the same removals, it is possible to obtain a sequence of coreduction pairs and free simplices compatible with a coreduction-based algorithm producing the same gradient vector field. In order to do this, we consider the following sequence obtained by taking sequence (3.4.1) in reverse order:

$$R_{i_n}^n, R_{i_n-1}^n, \ldots, R_1^n, R_{i_{n-1}}^{n-1}, \ldots, R_{i_1}^1, \ldots, R_2^1, R_1^1 \tag{3.4.2}$$

Consider (3.4.2) as an ordered list of removal operations performed on Σ (refer to Figure 3.2 for an example). The following properties hold:

1. For each $1 \leq l \leq n$ and $1 \leq j \leq i_l - 1$, R_j^l is a feasible coreduction pair.
2. For each $1 \leq l \leq n$, $R_{i_l}^l$ is a free simplex.

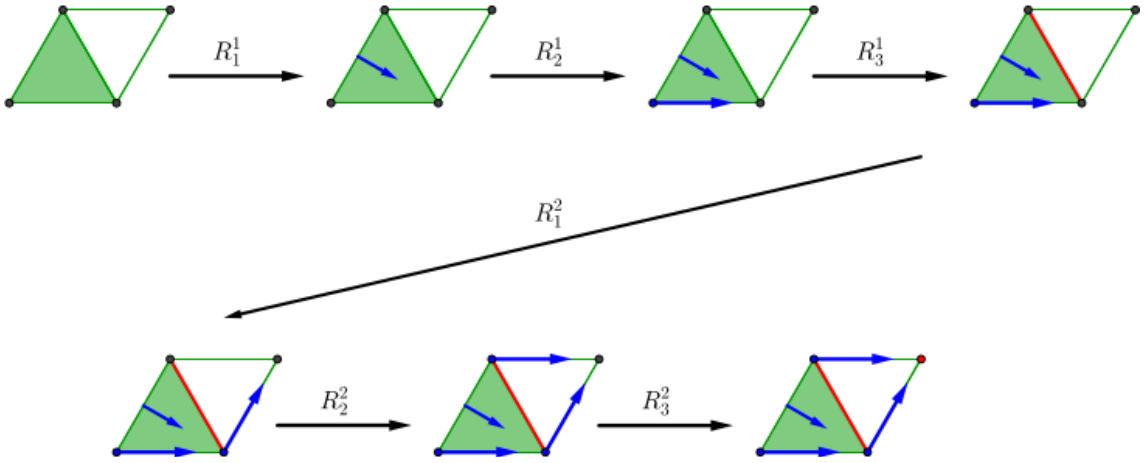


Figure 3.1: A sequence of reduction pairs (blue arrows) and top simplex removals (red simplices) produced by a reduction-based algorithm on a simplicial complex.

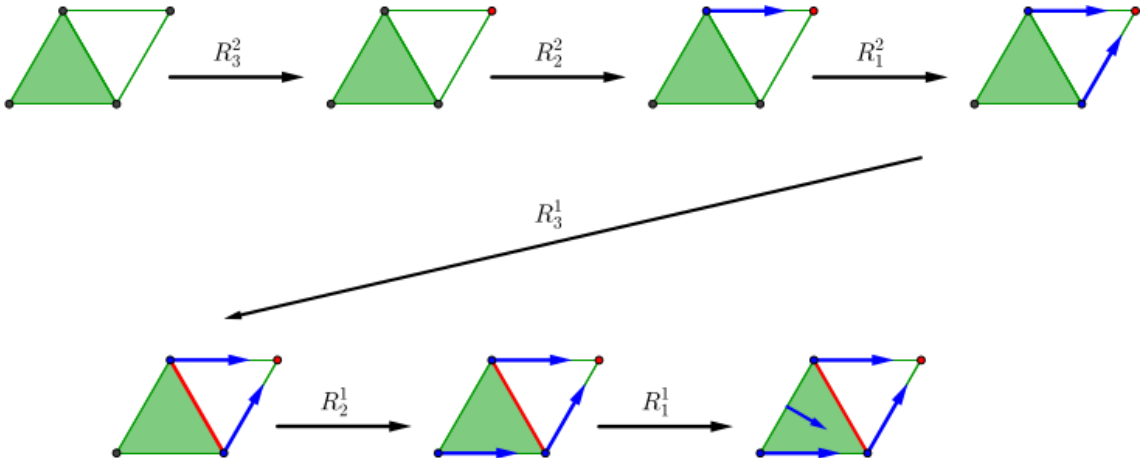


Figure 3.2: The sequence of coreduction pairs (blue arrows) and free simplex removals (red simplices) obtained by taking in the reverse order the reduction-based sequence considered in Figure 3.1.

To prove the two properties, we denote with:

- Σ_j^l the simplicial complex obtained in (3.4.1) after performing all the removal operations up to R_j^l included;
- S_j^l the S-complex obtained in (3.4.2) after performing all the removal operations up to the one preceding R_j^l.

We have that, for each value of l and j,

$$\Sigma_j^l \sqcup S_j^l = \Sigma \tag{3.4.3}$$

1. Let $R_j^l = (\sigma, \tau)$ with $1 \leq l \leq n$ and $1 \leq j \leq i_l - 1$. We have to prove that it represents a coreduction in sequence (3.4.2), i.e., $\mathrm{bd}_{S_j^l} \tau = \{\sigma\}$. By Lemma 3.3, in (3.4.1), τ cannot

be removed before the simplices in $\mathrm{bd}_\Sigma\,\tau$. So, all simplices in $\mathrm{bd}_\Sigma\,\tau \setminus \{\sigma\}$ belong to Σ^l_j. Then, by (3.4.3), $\mathrm{bd}_{S^l_j}\,\tau = \{\sigma\}$ and, thus, (σ, τ) is a feasible coreduction in S^l_j.
2. Let $R^l_{i_l}$ be the simplex σ. We have to prove that it represents a free simplex in sequence (3.4.2), i.e. $\mathrm{bd}_{S^l_{i_l}}\,\sigma = \emptyset$. Similarly to case 1., by Lemma 3.3, in (3.4.1), all simplices belonging to $\mathrm{bd}_\Sigma\,\sigma$ are in $\Sigma^l_{i_l}$. Then, by (3.4.3), $\mathrm{bd}_{S^l_{i_l}}\,\sigma = \emptyset$ and, thus, σ is a free simplex in $S^l_{i_l}$.

Sequence (3.4.2) satisfies properties 1. and 2., so, it represents a sequence of removals compatible with a coreduction-based algorithm producing on Σ the same Forman gradient of (3.4.1). □

It is interesting to understand if the equivalence between reduction-based and coreduction-based algorithms still holds with the further condition that allows the introduction of a critical simplex only if no reduction [coreduction] pair is available. Proposition 3.4 ensures that, given a reduction [coreduction] sequence produced on a simplicial complex Σ by an algorithm imposing a such condition, it is always possible to find a coreduction [reduction] sequence inducing the Forman gradient on Σ. In spite of this, Proposition 3.4 does not guarantee that a sequence produced by an algorithm satisfying the above mentioned condition exists. Figure 3.3 shows that, in general, this does not hold.

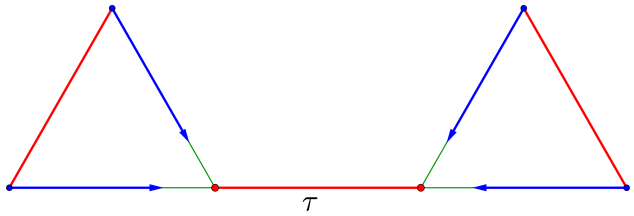


Figure 3.3: A gradient vector field V on a simplicial complex Σ that cannot be produced by a coreduction-based algorithm in which critical simplices are introduced only when no more coreduction pair is feasible.

The Forman gradient V depicted in Figure 3.3 can be considered as produced by a reduction-based algorithm starting with the removal of top simplex τ and introducing critical simplices only when it is strictly necessary. The Forman gradient V cannot be produced by a coreduction-based algorithm in which critical simplices are introduced only when no more coreduction pair is feasible because such an algorithm applied to Σ necessarily produces a gradient vector field with just one critical simplex of dimension 0 and two critical simplices of dimension 1.
For the dual situation, we have not been able to produce a similar counterexample. Precisely, we are not able to show a simplicial complex on which a gradient vector field produced by a coreduction-based algorithm in which critical simplices are introduced only when no more coreduction pair is available cannot be built by a reduction-based algorithm introducing critical simplices only when it is strictly necessary.

3.1.3 Interleaving reductions and coreductions

Another method to build a Forman gradient V on a simplicial complex can be to execute removals of reduction and coreduction pairs in an interleaved way. We call *interleaved-based algorithm* any algorithm producing a discrete vector field by using removals of reduction and coreduction pairs, of top simplices and of free simplices. Here, we prove that an algorithm of a such class actually produces a gradient vector field and that all interleaved methods are equivalent.

Proposition 3.5. *Given a simplicial complex Σ, the discrete vector field V produced by any interleaved-based algorithm is a gradient vector field.*

Proof. Given two pairs (σ, τ), (σ', τ') in V, we define $(\sigma, \tau) \leq (\sigma', \tau')$ if there exists a V-path starting with (σ, τ) and ending with (σ', τ'). In order to prove the thesis, i.e., that V is free of closed V-path, it is enough to prove that $\leq$ define a partial order on V. Consider set V as built in any intermediate step of the proposed algorithm and let (σ, τ) be the last pair inserted in V. The following properties allow us to conclude that the order defined on V is a partial order:

1. (σ, τ) is a minimal element with respect to the elements already inserted in V originating from a coreduction pair;

2. (σ, τ) is a maximal element with respect to the elements already inserted in V originating from a reduction pair.

Suppose that condition 1. does not hold. Then, there must exist an already performed coreduction pair (σ', τ') such that $\sigma \in \mathrm{bd}\,\tau'$. This implies that, at the step in which (σ', τ') has been performed, $\sigma, \sigma' \in \mathrm{bd}\,\tau'$. But this is impossible, otherwise the coreduction pair (σ', τ') could not have been performed. $\lightning$
Suppose that condition 2. does not hold. Then, there must exist an already performed reduction pair (σ', τ') such that $\sigma' \in \mathrm{bd}\,\tau$. This implies that, at the step in which (σ', τ') has been performed, $\tau, \tau' \in \mathrm{cbd}\,\sigma'$. But this is impossible, otherwise the reduction pair (σ', τ') could not have been performed. $\lightning$ □

Since both coreduction- and reduction-based algorithms can be considered as interleaved-based algorithms, the just proven proposition ensures also that any coreduction- or reduction-based algorithm returns a gradient vector field.

Having proven that any possible interleaved method leads to a gradient vector field, we are now interested in understanding if these different approaches could produce equivalent results. As an immediate consequence of Lemmas 3.2 and 3.3, we can claim the following result.

Remark 3.6. In each interleaved-based algorithm, each coreduction pair and free simplex removal cannot make a reduction pair feasible. Dually, each reduction pair and top simplex removal cannot make a coreduction pair feasible.

Proposition 3.7. *Given a simplicial complex Σ and the gradient vector field V on it produced by an interleaved-based algorithm, it is always possible to obtain the same gradient vector field with a reduction-based algorithm or, equivalently, with a coreduction-based algorithm.*

Proof. We prove that the sequence of removals produced by an interleaved-based algorithm on a simplicial complex can be also obtained with a sequence of coreduction pairs and free simplex removals. By Remark 3.6, we can suitably order such sequence, moving all the coreduction pairs and the free simplices at the beginning, thus creating a new sequence equivalent to the previous one. We apply to the last part, composed only of reduction pairs and top simplices, of this new sequence the same sorting strategy proposed in Proposition 3.4 to transform a reduction-based sequence to a coreduction-based sequence, and in this way we obtain the claim. □

From both an application and a theoretical point of view, it is interesting to define an algorithm to build a gradient vector field which minimizes the number of critical simplices. It is known that, in general, this problem is NP-hard [JP06]. Our results show that, from a theoretical point of view, the use of different simplification operators (such as reduction and coreduction pairs), or the combination of more than one, does not actually affect the number of resulting critical simplices.

Analogously to the coreduction- and reduction-based approaches, any algorithm interleaving both these operators can be easily adapted to the construction of a filtered gradient vector field.
Let Σ be a simplicial complex, F a filtration of Σ and V a gradient vector field on Σ. For each pair $(\sigma, \tau) \in V$, the condition required to guarantee that V is a filtered gradient vector field of F is satisfied independently from the fact that (σ, τ) has been created thanks to a reduction or a coreduction pair. This ensures that the validity of the results proven in this section is not affected by considering coreduction- reduction- and interleaved-based algorithms adapted to the construction of a filtered gradient vector field instead of the ones simply retrieving a gradient vector field. So, the proven equivalence between the various strategies to build a Forman gradient still holds in the filtered case.

3.2 Encoding of a simplicial complex endowed with a gradient vector field

To efficiently encode a simplicial complex endowed with a Forman gradient, we need effective data structures to represent the underlying simplicial complex and the gradient vector field defined on it.

As mentioned in Section 2.1, various topological data structures have been developed for encoding a simplicial complex Σ. The Incidence Graph (IG) [Ede87], a graph representation of the Hasse diagram of Σ, encodes all the simplices of Σ and their immediate boundary and coboundary relations. This data structure allows to easily retrieve incidence relations between the encoded entities but has a huge storage cost for large-size

complexes.
The Generalized Indexed data structure with Adjacencies (IA^*) [CDW11] overcomes this limitation by not explicitly representing all the simplices of a simplicial complex Σ but just encoding the vertices and the top simplices of Σ and a subset of its adjacent and boundary relations.
Similarly to the IA^* data structure, the Stellar Tree [Fel15] is based on an explicit encoding of the top simplices of Σ, but, unlike the IA^* data structure, it is built through a block decomposition of Σ. Another data structure taken in account is the Simplex Tree [BM12, BDM13]. The Simplex Tree encodes all the simplices of the represented simplicial complex plus a set of the incidence relations between simplices corresponding to a specific spanning tree of the Hasse diagram. Differently to the previous ones, the Simplex Tree cannot considered as a complete data structure since it does not allow to efficiently perform all the queries to navigate a complex but just a single incidence relation.
Comparisons between the above mentioned data structures have been presented in Section 2.1. They have revealed that, for a simplicial complex Σ, data structures based on a encoding of top simplices of Σ, such as the IA^* data structure and the Stellar Tree, are generally much more compact with respect to data structures which explicitly store all the simplices of Σ.
These considerations have led us to represent the underlying simplicial complex of a simplicial complex endowed with a Forman gradient by using the IA^* data structure.

3.2.1 Compact encoding of a gradient vector field

A fundamental issue, when working with a gradient vector field is encoding it efficiently. As discussed in Subsection 1.3.3, a gradient vector field V on a cell complex Γ is a collection of pairs of cells of Γ such that each cell is involved in at most one pair of V. So, whenever the incidence relations between the cells of a cell complex are naturally or explicitly encoded, the gradient vector field can be easily represented. For instance, for a regular grid H, boundary and adjacency relations can be easily managed through an indexing schema, and so, a Forman gradient V can be compactly encoded as a bitvector [GRWH12]. Since a simplicial complex has not such a fixed connectivity, the efficient encoding of a gradient vector field defined on such a complex becomes an interesting issue. Let Σ be a simplicial complex, a representation for a gradient vector field V encodes the pairing relation between two simplices, for all the simplices in Σ. Different representations have been defined based on the data structure used for representing Σ.

Given the Incidence Graph $G = (N, A)$ of a simplicial complex Σ, the arcs of G encode all the possible pairings that can be defined on Σ by considering two simplices of consecutive dimension. Since a gradient vector field V is a subset of the pairings in A, it can be encoded on the IG by adding 1 bit flag for each arc in G indicating whether such pairing is also a valid pair in V.

The crucial difference between an IG data structure and the IA^* is that the latter only encodes a subset of simplices of Σ. All those simplices that are not vertices nor top simplices can be represented only as tuple of vertices or indicating their position among the faces of a given top simplex. This forces us to compactly represent the configurations of the gradient inside each top simplex.

Generalizing to arbitrary dimensions the gradient encoding proposed in [FlDFW14, WIFD13] and discussed in Subsection 2.4.3.1, we obtain a representation encoding, for each top k-simplex σ, a bitvector of length $\sum_{i=1}^{k} \binom{k+1}{i+1}(i+1)$ representing all the possible pairings on its boundary.

Given the bitvector of a k-simplex σ, its first $k+1$ bits encode the pairing between σ and one of its $(k-1)$-faces. Then, for each of such faces, k bits are stored and so on until, for each 1-face, 2 bits are encoded storing the pairings with one of its vertices. For example, considering a 2-simplex (triangle), 3 bits are reserved for encoding the pairings with the boundary edges. Then, for each of them, 2 bits are reserved for encoding the pairings with the boundary vertices (see Figure 3.4).

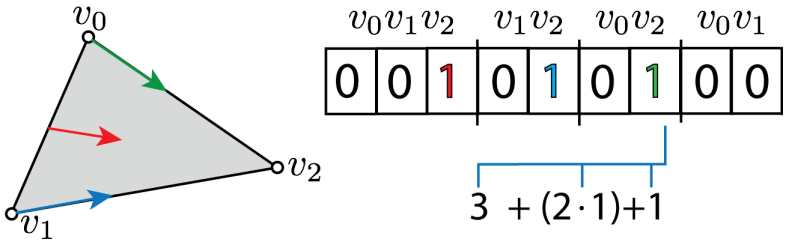


Figure 3.4: Example of the pairs encoded for a triangle.

If two paired simplices ν and τ are both on the boundary of σ, the resulting pair will be encoded in the bitvector of σ. Let j and l (with $j+1=l$) be the dimensions of ν and τ, respectively, we check the bit associated with the corresponding pair computing:

- the position of the bits reserved for l-simplices in σ $\sum_{i=l+1}^{k} \binom{k+1}{i+1}(i+1)$
- the position of ν on the boundary of τ obtained by enumerating the faces of τ (denote pos_ν),
- the position of τ on the boundary of σ obtained by enumerating the faces of σ (denoted pos_τ).

For example in Figure 3.4, considering the pair (v_0, v_0v_2),

- the position of the bits reserved for 1-simplices is 3
- the position of v_0v_2 on the boundary of the triangle is 1,
- the position of vertex v_0 on the boundary of v_0v_2 is 1.

Then, the bit representing their pairing relation is at position $3 + (2 \cdot 1) + 1$.

3.3 A coreduction-based algorithm for computing discrete Morse complexes

In this section, we describe an algorithm for computing a discrete Morse complex on a simplicial complex based on a data structure encoding only the top simplices and the

vertices, such as the IA^* data structure.
The algorithm consists of two different phases: the computation of a (filtered) gradient vector field through a coreduction-based algorithm and the extraction of the boundary maps of the discrete Morse complex.

3.3.1 Construction of a (filtered) gradient vector field

In this subsection, we focus on two different algorithms to endow a simplicial complex Σ with a gradient vector field. The theoretical equivalences proven in Section 3.1 reveal that there is not a preferable homology-preserving operator with which build a Forman gradient. This consideration has lead us to develop an algorithm based only on coreductions since this operator is the most suitable to the IA^* data structure which is the data structure chosen for encoding simplicial complexes. The first one allows to retrieve it through a dimension by dimension pairing. The second one is based on a decomposition of Σ and returns a gradient vector field compatible with a filtration induced by a scalar function defined on the vertices of Σ.

Algorithm 2 Coreduction-based algorithm

1: INPUT: Σ, d-dimensional simplicial complex
2: OUTPUT: V, gradient vector field; $\mathcal{A}$, set of critical simplices
3: $V := \emptyset$
4: $\mathcal{A} := \emptyset$
5: $k := 0$ *// working dimension*
6: $simpl := \Sigma_0$ *// consider the set of* 0*-simplices*
7: **while** $k <= d$ **do**
8: $cr := simpl$ *// save unpaired* k*-simplices*
9: $k := k + 1$ *// increase working dimension*
10: $simpl := \Sigma_k$ *// consider the set of* k*-simplices*
11: **while** $cr\,! = \emptyset$ **do**
12: $(\sigma, \tau) := getNextCoreduction(simpl, cr, \Sigma)$
13: **if** $(\sigma, \tau)\,! = \emptyset$ **then**
14: $pair(\sigma, \tau, V)$
15: $remove(\tau, simpl)$ *// remove from pairable*
16: $remove(\sigma, cr)$ *// remove from pairable*
17: **else**
18: $\sigma = getFirst(cr)$
19: $setCritical(\sigma, \mathcal{A})$ *// new critical simplex*
20: $remove(\sigma, cr)$ *// remove from pairable*

The first algorithm, described by Algorithm 2, represents a realistic specialization of the coreduction-based algorithm sketched in Algorithm 1. Starting from the simplices of lower dimension, all possible coreductions are applied to build a gradient vector field V incrementally. Starting from $k = 0$, the algorithm considers the set of unpaired k-simplices (row 8) and the set of the $(k + 1)$-simplices (row 10), respectively denoted as

cr and $simpl$. While cr is not empty (row 11), if a coreduction pair (σ, τ) between a k- and a $(k+1)$-simplex is feasible (row 13), (σ, τ) is added to the gradient vector field V. Otherwise (row 17), a free k-simplex is elected to be critical and the search of coreduction pairs continues. When all the k-simplices have been paired or declared as critical, the working dimension is increased by one. Obviously, since no coreduction pair is feasible on a simplicial complex, at the first step, a vertex v is declared as critical in V.

Algorithm 2 is based on the following functions and procedures. Procedure $pair(\sigma, \tau, V)$ updates the gradient vector field V by adding to it the coreduction pair (σ, τ). By considering the IA^* data structure, $pair(\sigma, \tau, V)$ takes linear time in the number of top simplices incident in σ (the smaller simplex), since, accordingly to the gradient encoding defined in Subsection 3.2.1, we have to update the pairing between σ and τ in all the top simplices incident in them.
Function $getNextCoreduction(simpl, cr, \Sigma)$ iterates on the set of unpaired simplices and takes the first simplex available for a coreduction within its candidate pair. This has linear worst case complexity. Considering the while cycle it brings to a worst case complexity of $O(n_k^2)$, where n_k represents the number of the k-simplices of Σ. However, the whole cycle has a linear complexity in practice.
Function $getFirst$ takes the first element of the set on which it is applied and it has $O(1)$ complexity.
Procedure $remove$ has a complexity depending on the container used for the simplices. By using std::sets, the complexity is $O(\log n_k)$.
Another step to be taken in account is the retrieval of the set of k-simplices of Σ. By using an IG this step requires constant time while, by choosing the IA^* data structure and assuming any top simplex of dimension d, the computation of Σ_k has $O(n_d\binom{d+1}{k+1})$ complexity.

In the specific case of persistent homology computation, an additional condition must assure that each simplex is paired only with another simplex belonging to the same filtration level. Two examples of coreduction-based algorithms implementing this idea can be found in [RWS11] for 2D and 3D regular grids and in [MN13] for simplicial complexes. Even if defined for regular grids, the algorithm in [RWS11] is a valuable approach that can be successfully extended to higher dimensions. We have implemented a dimension-independent version of the latter for simplicial complexes endowed with a scalar function defined on their vertices.

Let us consider a simplicial complex Σ and a function $f : \Sigma_0 \to \mathbb{R}$ associating a scalar value with each vertex of Σ. Under these conditions, Σ can be endowed with a filtration F naturally extending the function f. For each simplex σ in Σ, the filtration value of σ is defined equal to the maximum of the values taken by f on the vertices of σ. According to the definition given in Subsection 2.4.3.1, the *lower star of* v, denoted $\mathrm{star}^- v$, is the set of simplices incident in v for which all its vertices have a filtration value lower than or equal to $f(v)$. Formally,

$$\mathrm{star}^- v = \{\sigma \in \Sigma \,|\, \forall w \subseteq \sigma \wedge w \in \Sigma_0, f(w) \leq f(v)\}$$

Figure 3.5 (a) depicts by using different colors the lower star of each vertex with re-

spect to the function values which label the vertices of the simplicial complex. Under the above notations and supposing that the function f assumes different values on different vertices[1], the lower stars of the vertices of Σ form a partition of the simplices of Σ (see Figure 3.5 (a)). This enables us to subdivide the problem of computing coreduction pairs in disjointed sets.

A simplex σ belongs to the lower star of a single vertex and it can be paired with simplices belonging to the same lower star. Thus we can easily subdivided the computation of the gradient vector field working on the lower star of each vertex.
Algorithm 3 illustrates the procedure for computing a gradient vector field V on Σ filtered with respect to F. The algorithm works, for each dimension, with two sets of simplices:

Algorithm 3 Coreduction-based algorithm (filtered case)

```
1:  INPUT: Σ, d-dimensional simplicial complex
2:  INPUT: f, filtration on vertices of Σ
3:  OUTPUT: V, gradient vector field; A, set of critical simplices
4:  Σ_0 := vertices of Σ
5:  V := ∅
6:  A := ∅
7:  for v ∈ Σ_0 do
8:     k := 0  // working dimension
9:     st_v := {v}
10:    while k <= d do
11:       cr_v := st_v  // save unpaired k-simplices
12:       k := k + 1  // increase working dimension
13:       st_v := LowerStar(v, Σ, f, k)  // compute k-simplices of lower star
14:       while cr_v ! = ∅ do
15:          (σ, τ) := getNextCoreduction(st_v, cr_v, Σ)
16:          if (σ, τ) ! = ∅ then
17:             pair(σ, τ, V)
18:             remove(τ, st_v)  // remove from pairable
19:             remove(σ, cr_v)  // remove from pairable
20:          else
21:             σ = getFirst(cr_v)
22:             setCritical(σ, A)  // new critical simplex
23:             remove(σ, cr_v)  // remove from pairable
```

the set of k-simplices that can be paired with bigger $(k+1)$-simplices or declared as critical (row 11) and the set of $(k+1)$-simplices (row 13), called cr_v and st_v in the algorithm, respectively.
Candidate simplex for coreduction is extracted from the set st_v (row 15) and paired with its unique unpaired face (row 17). Recall that a simplex τ can be paired with another simplex σ by coreduction if σ is the only unpaired simplex on the boundary of τ. If there

[1]The injectivity hypothesis for function f can be easily ensured by a slight perturbation of the values taken by f.

are no coreductions available (row 20) a new critical simplex is taken from cr_v. Every time a simplex is paired or set as critical it is also removed from st_v or cr_v. When the set cr_v is empty, the working dimension increase. The algorithm stops when all the simplices in the lower star of v have been paired or set as critical.

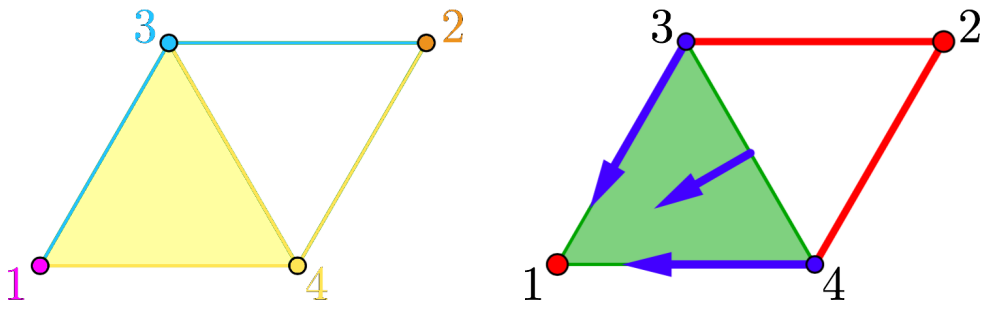


Figure 3.5: (a) A simplicial complex Σ decomposed in the lower stars of its vertices (different lower stars are depicted by using different colors) where, the labels of the vertices represent the scalar values taken by the function f. (b) The filtered Forman gradient V obtained by applying Algorithm 3 on the simplicial complex Σ depicted in (a). The red simplices denote the simplices declared as critical while, the blue arrows represent the pairs of V.

Let us consider the simplicial complex Σ and the scalar function f defined on its vertices represented in Figure 3.5 (a). Applied to this example, Algorithm 3 works independently on the lower star of each vertex of Σ returning the filtered gradient vector field V depicted in Figure 3.5 (b). Since the lower stars of the vertices labeled by 1 and 2 consist of only one vertex, these vertices are declared as critical. Then, the lower star of the vertex labeled by 3 is considered. It consists of the vertex 3 and of the 1-simplices 13 and 23. Vertex 3 is paired with the edge 13. The working dimension is increased by 1. Since the lower star does not contain any 2-simplex, no more coreduction pair is feasible and so, the edge 23 is declared as critical. Finally, the lower star of vertex 4 is computed. It consists of the vertex 4, of the 1-simplices 14, 24, 34 and of the 2-simplex 134. First, the vertex 4 is paired with the edge 14. Then, the working dimension is increased by 1 and the algorithm start to look for coreduction pairs composed by 1- and 2-simplices. So, the coreduction pair $(34, 134)$ is add to the gradient V and the unpaired 1-simplex 24 is declared as critical.

For computing the lower star of a vertex, specifically considering the IA^* data structure, we extract the top simplices incident in it and then, navigating their boundary, we collect all those simplices that are in the lower star. Let t_v be the number of top simplices incident in v and s_v the number of k-simplices in the lower star of v the procedure $LowerStar(v, \Sigma, f, k)$ takes $O(t_v \cdot s_v)$ in the worst case since some simplices are contained in the boundary of more than one top simplex (and those visited more than once). Analogously to Algorithm 2,

- procedure $pair(\sigma, \tau, V)$ takes linear time in the number of top simplices incident in σ;

- function $getNextCoreduction(st_v, cr_v, \Sigma)$ iterates on the set of unpaired simplices and so, it has linear worst case complexity; it brings to a worst case complexity of $O(s_v^2)$ if we consider the while cycle (linear complexity in practice);
- function $getFirst$ has $O(1)$ complexity;
- procedure $remove$ has $O(\log s_v)$ complexity by choosing std::sets.

It is immediate to verify that Algorithm 2 is a coreduction-based algorithm. Even if it is less obvious, also Algorithm 3 is a coreduction-based algorithm. Furthermore, it allows to retrieve a filtered gradient vector field.

Proposition 3.8. *Let Σ be a simplicial complex, $f : \Sigma_0 \to \mathbb{R}$ be an injective function and F be the filtration of Σ naturally induced by f. Given Σ and f as input, Algorithm 3 returns a filtered gradient vector field with respect to F.*

Proof. Algorithm 3 processes the lower stars independently. So, without loss of generality, we can assume that lower stars are processed in a sequence ordered by ascending function values. In this way, we obtain an ordered sequence of simplices added to V and to $\mathcal{A}$. We prove that this sequence, denoted as S, actually represents a feasible sequence of coreduction pairs and free simplices for Σ.
Let us consider a pair of simplices (σ, τ) elected as a pair of V during the processing of $\mathrm{star}^- v$. Let σ' be a simplex in $\mathrm{bd}_\Sigma \tau$ different from σ. If $\sigma' \in \mathrm{star}^- v$, then σ' has to be already added to V or to $\mathcal{A}$. Otherwise, if $\sigma' \notin \mathrm{star}^- v$, then there exists a vertex w of Σ such that $\sigma' \in \mathrm{star}^- w$ and $f(w) < f(v)$. So, σ' has to be already added to V or to $\mathcal{A}$ during the processing of $\mathrm{star}^- w$. In both cases, (σ, τ) can be considered as a feasible coreduction pair in the sequence S. Analogously, any simplex σ added to $\mathcal{A}$ during the processing of a lower star can be considered as a free simplex in the sequence S. So, Algorithm 3 is a coreduction-based algorithm and then, thanks to Proposition 4.14, it returns a gradient vector field.
Further, since Algorithm 3 pairs only simplices belonging to the same lower star and by the definition of F these simplices have the same filtration value, the returned gradient vector field V is necessarily filtered with respect to F. □

Algorithm 3 results to be especially optimized when working with Vietoris-Rips (VR) complexes (see Subsection 1.1.1). VR complexes represent a class of simplicial complexes particularly relevant in various applications and, since they are computed from clouds of points, they are quite often filtered according to a scalar function defined on their vertices. Algorithm 3 can be suitable adapted for an arbitrary simplicial complex Σ endowed with a different filtration. In this case, a filtered Forman gradient is retrieved by processing the lower stars of all the simplices of Σ instead of considering only the lower stars of the vertices of Σ.

As already mentioned, the construction of a gradient vector field V on a simplicial complex Σ can be exploited to speed up the computation of the homology of Σ. Algorithm 2 is a simple way to build a gradient vector field V. In spite of this, its time complexity is strongly affected by the function $getNextCoreduction(simpl, cr, \Sigma)$ which iterates on

the set of simplices of Σ of a certain dimension.
In order to overcome this limitation, a Forman gradient for Σ can be obtained by applying Algorithm 3 on Σ. Since, in this case, we do not have an available scalar function f, we can arbitrarily declare any ordering of the vertices of Σ as the function f. By using Algorithm 3, the number of retrieved critical simplices is usually greater than the one expected by performing Algorithm 2. For instance, the Forman gradient represented in Figure 3.5 (b) and obtained by performing Algorithm 3 on the depicted simplicial complex has superfluous critical simplices. In spite of this, experimental evaluations have led us to choose Algorithm 3 instead of Algorithm 2 not only for computing persistent homology but also for retrieving standard simplicial homology. This is mainly due to the fact that, in Algorithm 3, the function $getNextCoreduction$ iterates on a subset of the simplices belonging to a lower star instead of iterating on the set of the simplices of Σ of a certain dimension.

3.3.2 Extraction of the boundary maps

As discussed in Subsection 2.4.3.2, a discrete Morse complex can be extracted from the corresponding gradient vector field by navigating its paths. Specifically, the boundary maps $\tilde{\partial}_k : \mathcal{M}_k \to \mathcal{M}_{k-1}$ of the discrete Morse complex $\mathcal{M}_*$ associated with a (filtered) gradient vector field built on Σ, are obtained counting the multiplicity of each gradient path between two critical simplices.

Algorithm 4 Descending visit

```
1:  INPUT: Σ, d-dimensional simplicial complex
2:  INPUT: V, gradient vector field; A, set of critical simplices
3:  OUTPUT: M, set of simplices marked as visited
4:  for σ ∈ A do
5:     queue Q := ∅
6:     Q.push(σ)  // enqueue σ
7:     setMarked(σ, M)  // mark σ as visited
8:     while Q.notEmpty() do
9:        τ := Q.pop()
10:       for σ1 ∈ getImmediateBoundary(τ, Σ) do
11:          if isPaired(σ1) then
12:             τ1 := getPair(σ1)
13:             if !isMarked(τ1) then
14:                // each simplex is visited only once
15:                Q.push(τ1)
16:                setMarked(τ1, M)
```

Algorithm 4 illustrates the descending traversal of a gradient vector field. Starting from a critical k-simplex σ, all the $(k-1)$-simplices in the boundary of σ are selected (row 11) and, among them, only the $(k-1)$-simplices paired with a k-simplex not visited are considered (row 14). From such k-simplices a breadth-first traversal continues until all the V-paths starting from σ have been visited.

Procedure $setMarked(\sigma, M)$ annotates a simplex σ adding it to the set M. To do this, we use a bitvector similar to the one described in Section 3.2.1. The time complexity for annotating a simplex is then constant.
Function $getImmediateBoundary(\tau, \Sigma)$ returns the immediate boundary $\mathrm{bd}_\Sigma \tau$ of τ in the simplicial complex Σ. Extracting the immediate boundary of a simplex σ has a negligible time complexity since it is linear in the number of simplices in the boundary of σ. Since each simplex is visited at most once, the time complexity of the whole process is linear.

Using only the simplices marked as visited by Algorithm 4, and navigating the V-paths between two critical simplices in reverse (ascending) order, the boundary maps can be efficiently computed accordingly to Definition 1.46. As an example, we show in Figure 3.6 the two steps performed for computing the V-path between the critical simplices τ and σ. Starting from τ, the descending traversal is performed marking as visited all the triangles reached by a V-path starting at τ. Then, the (trivial) ascending traversal is performed starting at σ and navigating only the triangles previously visited until τ is reached.

As shown in the example of Figure 3.6, the simplices visited during the ascending traversal are a subset of the ones visited during the descending traversal. Thus the time complexity is still linear.

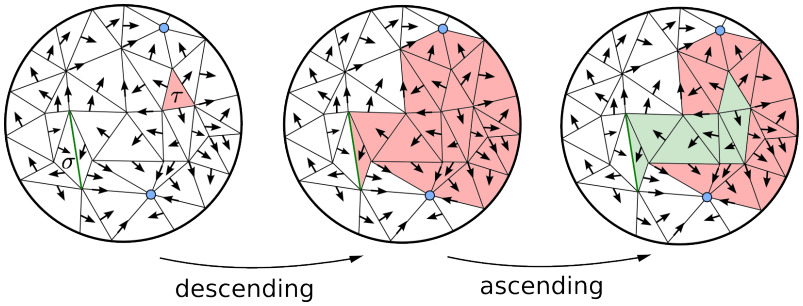


Figure 3.6: Descending and ascending traversals used during the computation of the V-path connecting τ and σ.

3.4 Experimental results

We have implemented Algorithm 3 and Algorithm 4 by choosing to represent simplicial complexes through the IA^* data structure. Given a simplicial complex Σ, the obtained tool allows us to achieve two different tasks:

- compute a discrete Morse complex associated with Σ,
- retrieve homology and persistent homology of Σ.

The discrete Morse complex has been used in many applications with different goals. Generally speaking, computing the discrete Morse complex guarantees a way for reducing the size of the original complex. In this work, we have considered two problems

concerning the computation of a discrete Morse complex in arbitrary dimensions. The first one is the storage cost required by the underlying simplicial complex, that increases exponentially with the growing of the dimension; the second one is the storage cost of the Forman gradient and the memory consumption required for computing it.

We evaluate the performances of our coreduction-based algorithm for both these points of view, considering real and synthetic datasets originated by different thematic areas. The hardware configuration used is an Intel i7 3930K CPU at 3.2Ghz with 64GB of RAM.

When working in higher dimensions, the gradient vector field V has been used as a homology-equivalent model of the original complex Σ rather than a combinatorial representation of the function gradient. Specifically, homology and persistent homology are computed on V exploiting its compactness with respect to Σ.

We have used three kinds of dataset in our experiments. The first ones are volumetric datasets that have been tetrahedralized. Each vertex of the dataset has a scalar value associated with. The DTI-scan is a Diffusion Tensor MRI Scan of a human brain, the VisMale dataset is a CT-scan of a man's head and the Ackley dataset is a synthetic function discretizing the Ackley's function [Ack87]. The second ones are networks obtained from real data on which the cliques are computed. Two of these datasets (AMAZON1, AMAZON2) are graphs representing the "Customers Who Bought This Item Also Bought" feature of the Amazon website. If a product i is frequently co-purchased with product j, the graph contains a directed edge from i to j (notice, we are considering the graph undirected). The third graph represent a road network of California where intersections and endpoints are represented by nodes and the roads connecting these intersections or road endpoints are represented by undirected edges (ROADNET). For simplicity we use the enumeration of the input vertices as field value. The third ones are point clouds extracted from a 2-sphere on which a VR complex is computed (datasets SPHERE-1.0, SPHERE-1.2, SPHERE-1.3).

3.4.1 Computing the discrete Morse complex

We have compared the performances of our implementation in computing the discrete Morse complex with the ones obtained by using the software tool *Perseus* [Nan]. Perseus is a software tool able to build a discrete Morse complex and to retrieve homology and persistent homology group for a large class of cell complexes containing simplicial complexes. In Perseus, the construction of a Forman gradient is based on a coreduction-based algorithm (quite similar to the one depicted in Algorithm 3) and simplicial complexes are encoded through the Incidence Graph (IG).

Table 3.1 summarizes the results obtained. Column Cp indicates the number of critical simplices in the computed Forman gradient, as well as the compression factor with respect to the simplicial complex in input. The compression factor generally depends on the homological changes in the filtration of a dataset and, as we can notice, it varies depending on the type of the dataset. Volumetric datasets benefit from a compression of about two orders of magnitude, network datasets are compressed only by a factor of ten, while higher dimensional complexes are compressed by five to eight orders of magnitude.

Dataset	d	$\|\Sigma\|$	Cp	Space (GB)					Time		
				Σ *static*	V	IA^* *run.*	IA^*_p *run.*	IG	IA^*	IA^*_p	IG
DTI-SCAN	3	24M	0.14M (171x)	0.97	0.47	1.5	1.69	19.9	3.1m	0.7m	77.3h
VISMALE	3	118M	0.94M (125x)	4.72	1.02	6.3	6.49	-	29.2m	6.5m	-
ACKLEY4	4	204M	0.01M (10^4x)	6.8	1.1	8.0	8.7	-	1.1h	19.7m	-
AMAZON01	6	2.2M	0.16M (13.7x)	0.12	0.11	0.28	0.5	1.8	14.5s	3.7s	20.94h
AMAZON02	7	18.4M	0.37M (49.7x)	0.28	0.25	0.68	1.3	9.8	281.9s	68.3s	>200h
ROADNET	3	4.8M	0.75M (6.4x)	0.8	0.6	1.6	2.5	3.3	15.8s	6.06s	>200h
SPHERE-1.0	16	0.6M	16 (10^5x)	0.0009	0.002	0.018	0.031	1.0	56.8s	22.1s	61.7s
SPHERE-1.2	21	26M	12 (10^7x)	0.0032	0.067	0.29	1.2	-	4.2h	1.8h	-
SPHERE-1.3	23	197M	7 (10^8x)	0.0034	0.09	1.8	7.0	-	173h	74.3h	-

Table 3.1: Memory consumption and timings obtained computing the discrete Morse complex with our library (IA^*, IA^*_p) and with Perseus (IG). The *Space* column indicates the memory consumption obtained running the three programs; Σ and V indicate the memory required by storing the simplicial complex and the Forman gradient, respectively. Column *run.* indicates the total memory consumption at runtime. The *Time* columns indicate the time needed to compute the discrete Morse complex; timings are reported in seconds (s), minutes (m) and hours (h). Some runs went out of memory (indicated with -) and some other have been stopped when the computation time was above 200 hours (indicated with >200h).

Regarding the memory consumption (column *Space*), we can notice that differences between the IA^* and the IG are still evident while considering the memory used at run-time. Looking at the IG implemented in the Perseus library the memory consumption at runtime is always comparable to the one used for storing the data structure statically. This is due to the fact that all the simplices are already represented at the beginning of computation and, thus, the maximum peak is reached before starting the reduction algorithm.

This is no longer true using the IA^*. For studying the performances of the representation, we are reporting the memory consumption of the static simplicial complex (column Σ), of the gradient vector field (column V) and of the total memory consumption used at runtime (column IA^*) fractioned. Summing up the storage cost of Σ and V we obtain the storage cost of the data structures from a static point of view (i.e. how much RAM is used for storing those information). Nevertheless, computing the Forman gradient requires an additional cost. Using the IA^* we are encoding only a fraction of the total number of simplices, the remaining have to be explicitly represented at runtime when extracting the lower star on each vertex (see Algorithm 3, lines 15-16) and encoded in the sets cr_v and st_v.

The total memory consumption is represented in Table 3.1 (column IA^* *run.*). We can notice that, while working on low dimensional complexes (like DTI-SCAN or ROADNET) the difference between the static storage cost (Σ + V) and the memory consumption at runtime (column IA^* *run.*) is low. Indeed in these cases, the lower star for each vertex is small. Working on higher dimensional complexes instead, the number of simplices in the lower star grows exponentially. In the worst case (Sphere-1.3), we end up occupying 1.8GB at runtime while for storing the simplicial complex and the Forman gradient were required less than 100MB.

Considering the timings, we can see that our approach is generally faster, in particular when the size of the complex is big. With the growing of the dimensions, we see that the complexity of the Forman gradient computation reaches its limits taking also 173 hours to finish in the worst case (dataset SPHERE-1.3). However, the peculiarity of the algorithm should be the easy parallelization of computation. In particular, the pairings between simplices belonging to different lower stars can be performed independently from each other. To investigate this aspect, we have implemented a parallel version of our algorithm based on the OpenMP library. The results obtained are reported in Table 3.1 (column IA^*_p). Running the same experiments as before but using eight threads we are basically able to process 8 vertices at the same time and this speeds up the computation 2 to 5 times. Since all the processes are executed on the same machine the maximum peak of memory experienced is higher with respect to the single thread version, up to 6 times bigger when working in high dimensions. (see Table 3.1 column Space IA^*_p).

3.4.2 Computing persistent homology

In this subsection, we compare the actual performances of three different libraries in the specific case of persistent homology computation. The libraries involved in our experimental evaluation have been our implementation based on the IA^* data structure, Perseus based on the IG and the *Gudhi library* [MBGY14].
The Gudhi library is de-facto the state-of-the-art library for what concern performances and compactness when working on simplicial complexes in high dimensions. Specifically, it has been used for computing persistent homology by exploiting an approach based on annotations and the Simplex Tree (ST) data structure [BDM13].

Dataset	d	$\lvert\Sigma\rvert$	Space (GB)				Time			
			IA^*	IA^*_p	IG	ST	IA^*	IA^*_p	IG	ST
AMAZON01	6	2.2M	0.28	0.5	1.8	0.4	14.5s	3.7s	20.94h	2.6s
AMAZON02	7	18.4M	0.68	1.3	9.8	2.2	281.9s	68.3s	>200h	44.05s
ROADNET	3	4.8M	1.6	2.5	3.3	1.3	15.8s	6.06s	>200h	3.3s
SPHERE-1.0	16	0.6M	0.018	0.031	1.0	0.06	56.8s	22.1s	61.7s	1.05s
SPHERE-1.2	21	26M	0.29	1.2	-	2.1	4.2h	1.8h	-	1.32m
SPHERE-1.3	23	197M	1.8	7.0	-	16.04	173h	74.3h	-	19.2m

Table 3.2: Memory consumption and timings obtained computing the Forman gradient with our library (IA^*, IA^*_p) and with Perseus (IG) and computing the persistent homology with Gudhi library (ST). The *Space* column indicates the memory consumption obtained running the four programs. Column *run.* indicates the total memory consumption at run-time. The *Time* columns indicate the time needed to compute the discrete Morse complex; timings are reported in seconds (s), minutes (m) and hours (h). Some runs went out of memory (indicated with -) and some other have been stopped when the computation time was above 200 hours (indicated with >200h).

Table 3.2 summarizes the obtained results. Concerning the memory consumption (column *Space*), we can notice that differences between the IA^* and the ST significantly decrease while considering the memory used at run-time. This is mainly due to the different approaches that are implemented in our library (based on the encoding of the Forman

gradient) and in the Gudhi library (based on the annotation matrices whose compactness has been discussed in [BDM13]). Considering the IG implemented in the Perseus library, the memory consumption at run-time is always comparable to the one used for storing the data structure statically. Considering the timings and comparing our library to Perseus, we can see that our approach is generally faster in particular when the size of the complex is big. With the growing of the dimensions, we see the computational complexity of the Forman gradient computation reaches its limits taking also 173 hours to finish in the worst case (dataset SPHERE-1.3).

Compared to the Gudhi library, our implementation is always much slower. This was an expected result caused by the compactness of our data structure, that requires time for extracting data not explicitly encoded, and by the goodness of the approach based on annotation matrices already documented in literature. However, the peculiarity of a Forman based implementation should be the easy parallelization of computation. To investigate this aspect, we have implemented a parallel version of our algorithm based on the OpenMP library. The results obtained are reported in Table 3.2 (column IA^*_p). Running the same experiments as before but using eight threads, we are basically able to process 8 vertices at the same time. This speeds up the computation 2 to 5 times. Since all the processes are executed on the same machine, the maximum peak of memory experienced is higher with respect to the single thread version, up to 6 times bigger when working in high dimensions (dataset SPHERE-1.3).

Even though the Gudhi library is the best implementation currently available in the literature, the fact that for high dimensional complexes it represents all the simplices of the simplicial complex explicitly could lead to serious limitations when computing persistent homology with real datasets. Conversely, the Forman gradient computed with our approach can be used for computing a complex that shares the same persistent homology of the simplicial complex, still being much smaller. The compact encoding and the algorithm, working independently on the lower star of each vertex, will be at the center of investigations for the development of a distributed approach.

3.5 Concluding remarks

We have analyzed and proposed different strategies to endow a simplicial complex with a gradient vector field through the use of homology-preserving operator and to extract the corresponding discrete Morse complex. We have formally proven the theoretical equivalence of such methods which allow reducing the complexity of the computation through reductions and coreductions. We have developed and implemented an algorithm to efficiently build a discrete Morse complex based on coreductions, on a space-efficient representation of the simplicial complex and on a compact encoding of the Forman gradient. Finally, we have showed the application of the proposed algorithm to the persistent homology computation providing experimental results and comparisons.

We take into account here three different developments of the work presented in this chapter:

- development of a distributed implementation of the coreduction-based algorithm,
- application of our algorithm to the extraction and the visualization of extremum graphs,
- design and development of an algorithm for computing multi-dimensional persistent homology.

A first future development could be to design and implement an efficient encoding for a simplicial complex in arbitrary dimensions based on the Stellar Tree [Fel15], a compact topological data structure based on a spatial index, which stores only the vertices and the top simplices of the complex. This latter would not only reduce the storage cost further, but also allow an efficient localized computation of the homology, overcoming the limitation in this of the IA^* data structure.
Since the results obtained from the parallel implementation, we will consider a distributed implementation. Subdividing the computation on different machines we should be able to get a boost on timings without affecting memory consumptions.

The representation of a Forman gradient will be at the base of many application domains. When working with scientific simulations and in particular with high dimensional functions, the *topological spines* have been proposed as a visual representation of the extremum graphs [CLB11]. The extremum graph is a simplified substructure of the Morse-Smale complex encoding information only about the critical points of index 0, 1, $d-1$ and d where d is the dimension of the underlying complex. The topological spines can be thought as an extremum graph presenting informative visual descriptions of the function behavior along the paths connecting maxima and minima with saddles.
For representing this structure, computing the 1-skeleton of the Morse-Smale complex as well as computing the volume of the area surrounding maxima and minima is a necessary step. These structures are implicitly represented by the Forman gradient that can be exploited for navigating the simplices spanning those regions at wish (following the algorithm in Section 3.3.2), providing the perfect framework for interactively studying subsets of the domain then retrieving (and storing) such information under request.
In the near future, we plan to suitably adapt our work to quickly extract the extremum graph of a scalar field defined on a simplicial complex and to visually represent it through a topological spine [CLB11].

As already mentioned in Subsection 1.2.2, *multi-dimensional persistent homology* [CZ09, CSZ09] represents a generalization of classical persistent homology capturing the topology of a family of shapes parameterized along multiple scalar functions. As recently addressed in [AKL16, AKLM15], multi-dimensional persistent homology can be obtained by exploiting discrete Morse theory. We plan to apply our method for efficiently building discrete Morse complexes in order to improve the computation of multi-dimensional persistent homology.
In the framework of multi-dimensional persistence, the lower stars of the vertices does not form a partition of the original complex Σ. We should adapt our algorithm to find weaker properties that still provide a total ordering on the entire complex. This is a relevant aspect since it means that boundary relations of the reduced Morse complex can be theoretically retrieved from the gradient vector field with the same technique we applied

in the one-dimensional setting. Allili et al. [AKL16, AKLM15] have already adapted both the algorithms in [KKM05] and [RWS11], which compute persistent homology through a discrete Morse complex, to a vector-valued filtering function, producing an implementation working on 2-dimensional simplicial complexes.

Chapter 4

Homology Computation through Multi-resolution Models

A multi-resolution model is a tool to encode and handle shape at different levels of detail. The main purpose of this chapter is the development of methods for efficiently computing homological information of a cell or of a simplicial complex by exploiting a multi-resolution model. Besides improving the efficiency in homology computation, these techniques allow the explicit retrieval of homology generators at different levels of detail. In order to achieve this goal for cell and simplicial complexes, a general notion of multi-resolution model is introduced first.

A key ingredient in the definition of a multi-resolution is represented by the operator used to refine the input complex. In Section 4.1, two classes of topological operators, acting on cell and simplicial complexes respectively, are introduced. Section 4.2 is devoted to provide a consistent definition of a general geometry-based model able to represent a shape at various levels of detail. In Section 4.3 and Section 4.4, this general model is specialized for cell and simplicial complexes. For both situations, a multi-resolution model based on homology-preserving operators is defined. The multi-resolution model for cell complexes has been implemented and used to retrieve homological information of each complexes which can be extracted from the model.

The cellular topological operators and the homology computation through a cellular multi-resolution model have been presented in [ČomićDIF14]. For a more detailed treatment of simplicial operators please refer to [DFW12]. The definition of a general multi-resolution model and its specialization for simplicial complexes, presented in Section 4.2 and 4.4, will be soon collected in a work currently in preparation.

4.1 Topological operators for cell and simplicial complexes

In the literature, several operators for manipulating cell and simplicial complexes have been introduced [DL03, DGDP12, BADSM08, EW79, BHS80, MS82, Man88, LL01, MSNK89, Mas93, Gom04, LPT$^+$03, Mat02, LT97]. Most of the proposed classes of

operators have been defined for low dimensional complexes, or they do not represent a set able to describe all the possible modifications of a complex. In this section, we propose two complete classes of dimension-independent operators for cell and simplicial complexes, respectively, that will play an important role in the multi-resolution models we are going to define.

4.1.1 Operators for cell complexes

In this subsection, we present a class of operators to modify cell complexes in arbitrary dimensions [ČomićDIF14]. Further, we show that this class of modifications forms a minimally complete basis for the set of all possible operators that modify cell complexes in a topologically-consistent manner.

These operators can be easily classified according to their effect on the homology of the cell complex on which they are applied:

- *homology-preserving operators*: $KiC(i+1)C$ (***Kill i-Cell and (i+1)-Cell***), which removes an i-cell and an $(i+1)$-cell, and its inverse $MiC(i+1)C$ (***Make i-Cell and (i+1)-Cell***);
- *homology-modifying operators*: $KiCiCycle$ (***Kill i-Cell and i-Cycle***), which removes an i-cell and an i-cycle, and its inverse $MiCiCycle$ (***Make i-Cell and i-Cycle***).

There are in total d homology-preserving and $d+1$ homology-modifying operators on cell complexes of dimension d.

4.1.1.1 Homology-preseving operators

Given a cell complex Γ, *homology-preserving* operators $KiC(i+1)C$ (***Kill i-Cell and (i+1)-Cell***) change the number of cells in Γ by decreasing the number n_i of i-cells and the number n_{i+1} of $(i+1)$-cells by one unit. There are two types of homology-preserving operators:

- *contract*, denoted as $KiC(i+1)C_{co}(q,p,p')$;
- *remove*, denoted as $KiC(i+1)C_{re}(q,p,p')$.

Operator $KiC(i+1)C_{co}(q,p,p')$ is feasible on Γ under the following conditions:

- the $(i+1)$-cell q of Γ is bounded by two i-cells of Γ (the i-cell p to be deleted and the i-cell p' which will remain),
- the i-cell p is a regular face of the $(i+1)$-cell q.

Under these assumptions, the effect of *contract* operator $KiC(i+1)C_{co}(q,p,p')$ on Γ is as follows:

- cells p and q are removed,
- i-cell p is replaced by i-cell p' on the boundary of each $(i+1)$-cell r in the coboundary of i-cell p.

An example of *contract* operator $KiC(i+1)C_{co}(q,p,p')$ is depicted in Figure 4.1. The illustrated $K0C1C_{co}(q,p,p')$ operator removes vertex p and 1-cell q and replace p by p' in the boundary of 1-cell r.

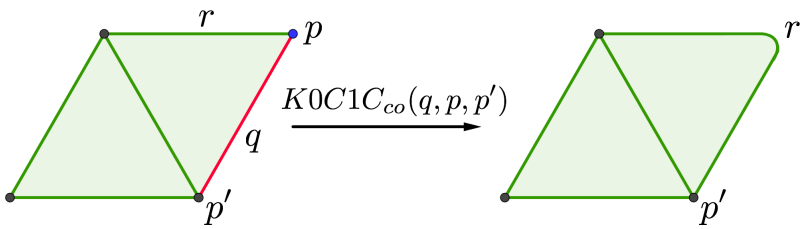


Figure 4.1: Effect of *contract* operator $K0C1C_{co}(q,p,p')$ on a 2-dimensional cell complex.

Contract operator can be applied even if i-cell p' does not exists and the boundary of the $(i+1)$-cell q to be deleted consist of just the i-cell p. In this situation, under the condition that the i-cell p is a regular face of the $(i+1)$-cell q, *contract* operator is denoted as $KiC(i+1)C_{co}(q,p)$ and its effect is to delete both cells p and q from the complex.

The second type of homology-preserving operator is called *remove* and it is denoted as $KiC(i+1)C_{re}(q,p,p')$. *remove* operator $KiC(i+1)C_{re}(q,p,p')$ is feasible on Γ under the following conditions:

- i-cell q of Γ is a face of two $(i+1)$-cells (the $(i+1)$-cell p to be deleted and the $(i+1)$-cell p' which will remain),
- i-cell q is a regular face of the $(i+1)$-cell p.

The effect of *remove* operator $KiC(i+1)C_{re}(q,p,p')$ is completely dual with respect to that of *contract* operator:

- cells p and q are removed,
- $(i+1)$-cell p is replaced by $(i+1)$-cell p' on the coboundary of each i-cell r in the boundary of $(i+1)$-cell p.

An example of *remove* operator $KiC(i+1)C_{re}(q,p,p')$ is depicted in Figure 4.2. The illustrated $K0C1C_{re}(q,p,p')$ operator removes 1-cell p and 2-cell q and replace p by p' in the coboundary of 1-cells r_1 and r_2.

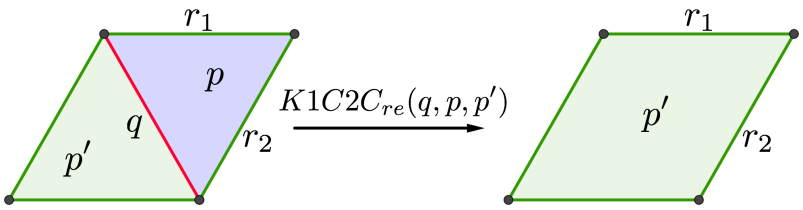


Figure 4.2: Effect of *remove* operator $K0C1C_{re}(q,p,p')$ on a 2-dimensional cell complex.

Remove operator can be applied even if $(i+1)$-cell p' does not exists and the coboundary of i-cell q to be deleted consist of just $(i+1)$-cell p. In this situation, under the condition that the i-cell q is a regular face of the $(i+1)$-cell p, *remove* operator is denoted as $KiC(i+1)C_{re}(q,p)$ and its effect is to delete both cells p and q from the complex.

Inverse $MiC(i+1)C$ (*Make i-Cell and (i+1)-Cell*) operators adds an i-cell and an $(i+1)$-cell to the complex Γ. Again, we have operators of two different types:

- *expand*, denoted as $MiC(i+1)C_{ex}(q,p,p')$, inverse of *contract* operator $KiC(i+1)C_{co}(q,p,p')$;
- *insert*, denoted as $MiC(i+1)C_{in}(q,p,p')$, inverse of *remove* operator $KiC(i+1)C_{re}(q,p,p')$.

Expand operator $MiC(i+1)C_{ex}(q,p,p')$

- subdivides the existing i-cell p' into two by splitting its coboundary,
- creates $(i+1)$-cell q bounded by the two i-cells p and p'.

Insert operator $MiC(i+1)C_{in}(q,p,p')$

- subdivides the existing $(i+1)$-cell p' into two by splitting its boundary,
- creates i-cell q separating the two $(i+1)$-cells p and p'.

In both cases, the new i-cell is a regular face of the new $(i+1)$-cell. Examples of *expand* and *insert* operators are depicted in Figure 4.1 and 4.2, respectively. In such figures, the two operators are not explicitly represented but they can be considered as the operators transforming the cell complex depicted in the right into the cell complex in the left.

The inverse operators of *contract* $KiC(i+1)C_{co}(q,p)$ and of *remove* $KiC(i+1)C_{re}(q,p)$ are denoted as $MiC(i+1)C_{ex}(q,p)$ and $MiC(i+1)C_{in}(q,p)$, respectively.

Operator $MiC(i+1)C_{ex}(q,p)$ creates

- an i-cell p and an $(i+1)$-cell q bounded only by p.

Dually, operator $MiC(i+1)C_{in}(q,p)$ creates

- an $(i+1)$-cell q and an i-cell p bounding only q.

In both cases, the new i-cell appears exactly once on the boundary of the new $(i+1)$-cell.

By using discrete Morse theory, it is easy to prove that the application of one of the above defined operators on a cell complex Γ actually preserves its homology.

Proposition 4.1. *Operators $KiC(i+1)C$ and $MiC(i+1)C$ preserve the homology with coefficients in any Abelian group.*

Proof. Since $MiC(i+1)C$ are the inverse operators of $KiC(i+1)C$, it is sufficient to prove that $KiC(i+1)C$ are homology-preserving operators. Let Γ be a cell complex, $KiC(i+1)C$ the operator that deletes an i-cell p and an $(i+1)$-cell q from Γ and let Γ' be the resulting cell complex. Let $f : \Gamma \to \mathbb{R}$ be the discrete Morse function defined by

$$f(x) = \begin{cases} \dim x & \text{if } x \in \Gamma \setminus \{p, q\} \\ \frac{\dim p + \dim q}{2} = i + \frac{1}{2} & \text{otherwise} \end{cases}$$

The chain complex associated with Γ' represents the discrete Morse complex associated with Γ with respect to function f. By Theorem 1.48, we conclude that $H_*(\Gamma; G) \cong H_*(\Gamma'; G)$ for any Abelian group G. □

4.1.1.2 Homology-modifying operators

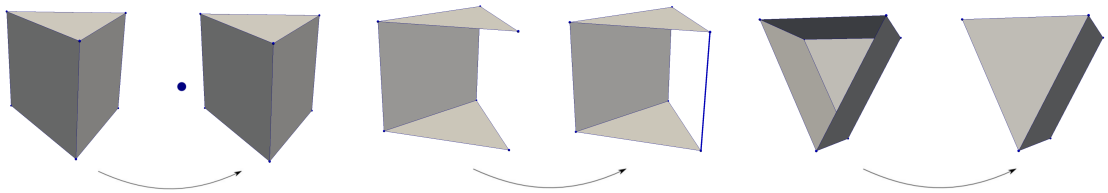
Homology-modifying operators change the number of cells in a complex Γ plus its Betti numbers. Given a cell complex Γ, homology-modifying operator $KiCiCycle$ (***Kill i-Cell and i-Cycle***) deletes an i-cell and destroys an i-cycle, thus decreasing the numbers n_i of i-cells in Γ and the i^{th} Betti number β_i of Γ by one unit. It is feasible on a cell complex Γ if the coboundary of the cell to be deleted is empty.

The inverse $MiCiCycle$ (***Make i-Cell and i-Cycle***) increases the number n_i of i-cells and the number β_i of independent non-bounding i-cycles by a unit. It is feasible on a complex Γ if all the cells on the boundary of the cell to be created are in Γ.

An example of application of such operators is illustrated in Figure 4.3. Operator $M0C0Cycle$ creates a new vertex and a new connected component, it increases by a unit the number of 0-cells and the 0^{th} Betti number β_0. It is used as an initialization operator to create a new complex Γ. Operator $M1C1Cycle$ creates a new edge and a new 1-cycle, thus increasing both the number of 1-cells and the first Betti number β_1 by one unit. Operator $M2C2Cycle$ creates a new face and a new 2-cycle, thus increasing the number of 2-cells and the second Betti number β_2 by one unit.

4.1.1.3 Minimality and completeness

The operators described in this subsection form a minimal set of operators on cell complexes in arbitrary dimensions, in such a way that any other operator can be expressed as



(a) (b) (c)

Figure 4.3: Examples of homology-modifying operators on a 2-complex: $M0C0Cycle$ (*Make 0-Cell and 0-Cycle*) (a); $M1C1Cycle$ (*Make 1-Cell and 1-Cycle*) (b); $M2C2Cycle$ (*Make 2-Cell and 2-Cycle*) (c).

suitable combination of those in the minimal set.

Proposition 4.2. *The operators $KiC(i+1)C$, $KiCiCycle$ and their inverse $MiC(i+1)C$, $MiCiCycle$ form a minimal and complete basis of operators for creating and updating d-dimensional cell complexes.*

Proof. To prove this fact, we interpret such operators as ordered $(2d+2)$-tuples $(x_0, x_1, \ldots, x_d, c_0, c_1, \ldots, c_d)$ in an integer grid, belonging to the hyperplane Π: $\sum_{i=0}^{d} (-1)^i x_i = \sum_{i=0}^{d} (-1)^i c_i$ defined by the Euler-Poincaré formula (Proposition 1.29 in Subsection 1.2.1.2). The first $d+1$ coordinates denote the number of i-cells created or deleted by the operator, depending on the sign of the coordinate, while the last $d+1$ coordinates denote the change in the Betti numbers of the complex induced by the operator. Operator $MiC(i+1)C$, $0 \leq i \leq d-1$, has coordinates

$$x_j = \begin{cases} 1 & \text{for } j = i, i+1 \\ 0 & \text{otherwise} \end{cases}$$
$$c_j = 0 \text{ for } j \in \{0, 1, \ldots, d\}$$

Operator $MiCiCycle$, $0 \leq i \leq d$, has coordinates

$$x_j = c_j = \begin{cases} 1 & \text{for } j = i \\ 0 & \text{otherwise} \end{cases}$$

In the following, the operators $MiC(i+1)C$ and $MiCiCycle$ endowed with a negative coefficient have to be considered as the operators $KiC(i+1)C$ and $KiCiCycle$, respectively.
In order to prove the proposition, we have to show that any $(2d+2)$-tuple in the hyperplane Π can be expressed as a linear combination of $2d+1$ $(2d+2)$-tuples corresponding to our operators. A tuple $(a_0, a_1, \ldots, a_d, b_0, b_1, \ldots, b_d)$ in the hyperplane Π (i.e., such that $\sum_{i=0}^{d} (-1)^i a_i = \sum_{i=0}^{d} (-1)^i b_i$) can be expressed through the $2d+1$ independent $(2d+2)$-tuples corresponding to our operators as $\sum_{i=0}^{d-1} \alpha_i MiC(i+1)C + \sum_{i=0}^{d} \beta_i MiCiCycle$ if

$$(a_0, a_1, \ldots, a_d, b_0, b_1, \ldots, b_d) =$$
$$= (\alpha_0 + \beta_0, \alpha_0 + \alpha_1 + \beta_1, \alpha_1 + \alpha_2 + \beta_2, \ldots, \alpha_{d-2} + \alpha_{d-1} + \beta_{d-1}, \alpha_{d-1} + \beta_d, \beta_0, \beta_1, \ldots, \beta_d)$$

It follows that $\beta_i = b_i$, $0 \leq i \leq d$, and

$$\begin{cases} \alpha_0 = a_0 - b_0 \\ \alpha_1 = a_1 - b_1 - \alpha_0 = (a_1 - a_0) - (b_1 - b_0) \\ \alpha_2 = a_2 - b_2 - \alpha_1 = (a_2 - a_1 + a_0) - (b_2 - b_1 + b_0) \\ \vdots \\ \alpha_{d-1} = (a_{d-1} - a_{d-2} + \cdots + (-1)^d a_0) - (b_{d-1} - b_{d-2} + \cdots + (-1)^d b_0) = a_d - b_d \end{cases}$$

In short, $\alpha_i = \sum_{j=0}^{i} (-1)^{i-j} a_j - \sum_{j=0}^{i} (-1)^{i-j} b_j$, $0 \leq i \leq d-1$ ($\alpha_{d-1} = a_d - b_d$) and $\beta_i = b_i$, $0 \leq i \leq d$. Thus, each operator satisfying Euler-Poincaré formula on a cell complex Γ can be expressed as a linear combination of our operators.
In the space (hyperplane) of dimension $2d+1$, a generating set consisting of $2d+1$ independent tuples forms a basis for the hyperplane.

□

4.1.1.4 Comparison with other operators for cell complexes

In the literature, several operators modifying cell complexes have been introduced. The completeness result just proven allows to describe such operators as combinations of the operators introduced in this subsection. In particular, we can express in terms of these operators the following remarkable modifications: removal and contraction operators, Euler operators and handle operators. Please refer to [ČD12, ČomićDIF14] for a more detailed discussion.

Removal and contraction operators
Removal and contraction operators have been introduced in digital geometry as simplification operators on n-G-maps [DL03]. A *Generalized (G-) map* is a combinatorial structure for a cellular subdivided object of arbitrary dimension. The definition of a G-map is based on a set of elements, called darts, and on some involution relations between them. In this framework, the notion of cell is described in terms of the orbit of a dart while adjacency and incidence relations between the cells are implicitly encoded though the notion of involutions.

Removal and contraction operators defined on n-G-maps acts as follows. An i-cell q, $0 \leq i \leq n-1$, can be removed in two cases: if it bounds exactly two different $(i+1)$-cells p and p' and it appears exactly once on the boundary of both p and p'; or if it bounds exactly one $(i+1)$-cell p and it appears exactly twice on the boundary of p. The contraction operator is dual.
The first instance of the removal operator is a special case of $KiC(i+1)C_{re}(q, p, p')$, as it requires that the i-cell q appears exactly once not only on the boundary of the $(i+1)$-cell p but also on the boundary of the $(i+1)$-cell p'. The effect of the first instance of the removal operator is the same as the effect of $KiC(i+1)C_{re}$. The second instance of the removal operator may, but is not guaranteed to, preserve the topological characteristics of the complex (it may produce cells that are not topological cells, or it may discon-

nect the complex). Thus, it cannot be classified either as a homology-preserving, or as a homology-modifying operator.

In [DGDP12], homology generators of a cell complex are computed using two homology-preserving simplification operators: the removal of an i-cell incident in exactly two $(i+1)$-cells (which is the same as $KiC(i+1)C_{re}(q,p,p')$ and as the first instance of the removal operator in [DL03]) and the removal of a dangling cell (which is the same as $KiC(i+1)C_{re}(q,p)$). The inverse (refinement) insertion and expansion operators have been introduced in [BADSM08]. They are the same as $MiC(i+1)C_{in}(q,p,p')$ and $MiC(i+1)C_{ex}(q,p,p')$, respectively.

Euler operators

Virtually all the proposed sets of basis Euler operators on cell 2- and 3-complexes contain MEV (Make Edge and Vertex) and MEF (Make Edge and Face) operators, which are the same as $M0C1C$ (Make 0-cell and 1-cell) and $M1C2C$ (Make 1-cell and 2-cell), respectively.

Several homology-modifying operators have been proposed for cell 2-complexes that define the boundary of a solid in $\mathbb{R}^3$, called *boundary models*. In these models, there is only one implicitly represented volumetric cell, which is not necessarily homeomorphic to a 3D ball.

The *glue* operator in [EW79] merges two 2-cells and deletes both of them. It corresponds to the connected sum operator on surfaces. If the two glued 2-cells belong to two different connected components of the complex, one of the components is deleted (and β_0 is decreased by a unit). If the two glued faces belong to the same connected components, a handle or through-hole is created (and β_1 is increased by two units).

In [BHS80, MS82, Man88], the homology-modifying operator is called $MRKF$ (*Make Ring, Kill Face*). It is similar to the *glue* operator in [EW79], but it imposes less restrictive conditions on the 2-cells to be glued, and it deletes only one of the 2-cells. The 2-cells are not supposed to be topological (homeomorphic to a 2D ball).

Homology-modifying operators defined for arbitrary cell 2-complexes in $\mathbb{R}^3$ [LL01] are called $MECh$ (*Make Edge and Complex Hole*), $MFKCh$ (*Make Face, Kill Complex Hole*) and $MFCc$ (*Make Face and Complex Cavity*). They are the same as operators $M1C1Cycle$, $M2CK1Cycle$ (*Make 2-Cell Kill 1-Cycle*, which can be expressed as $K1C1Cycle$, $M1C2C$) and $M2C2Cycle$, respectively. For 3-complexes in $\mathbb{R}^3$ [MSNK89, Mas93], an additional homology-modifying operator is defined, called $MVlKCc$ (*Make Volume, Kill Complex Cavity*). It is the same as $M3CK2Cycle$ (*Make 3-Cell, Kill 2-Cycle*) operator, and can be expressed as $K2C2Cycle$, $M2C3C$.

In [Gom04], the operators defined in [MSNK89] have been extended to complexes called *stratifications*, in which the cells, called *strata*, are defined by analytic equalities and inequalities. The cells are not necessarily homeomorphic to a ball, and they may have incomplete boundaries. Among the operators on stratifications proposed in [Gom04], operators on topological cells (that are homeomorphic to a ball) with complete boundaries can be classified as homology-preserving (called *cell subdividers*) and homology-modifying (called *global hole shapers*). Both types of operators are instances of the introduced atomic operators. A *cell subdivider* subdivides an i-cell by inserting into it an $(i-1)$-cell.

This operator is the same as the $M(i-1)CiC$ operator.
A *global hole shaper* either attaches or detaches a cell, thus creating a hole. There are two instances of this operator: the attached topological i-cell creates an i-hole or the detached topological i-cell creates an $(i-1)$-hole. The first instance of this operator corresponds to $MiCiCycle$. The second instance corresponds to $KiCM(i-1)Cycle$ (***Kill i-Cell, Make (i-1)-Cycle***), and can be expressed as $M(i-1)C(i-1)Cycle$, $K(i-1)CiC$.
The inverse homology-modifying operators attach or detach a cell, thus deleting a hole. They correspond to $KiCiCycle$ and $MiCK(i-1)Cycle$ (inverse to $KiCM(i-1)Cycle$), respectively.

Handle operators
(Homology-modifying) *handle operators* on a cell 2-complex Γ triangulating a surface S have been introduced in [LPT+03]. They are based on the *handlebody theory* for surfaces [Mat02], stating that any surface S can be obtained from a 2-ball by iteratively attaching handles (0-, 1- and 2-handles).
Attachment of a 0-handle creates a new surface with one face, three edges and three vertices. It can be expressed as one $M0C0Cycle$ operator, two $M0C1C$ operators and one $M1C2C$ operator, which together create a triangle. The operator that corresponds to the attachment of a 1-handle identifies two boundary edges of Γ (incident in exactly one face) with no vertices in common. The operator that corresponds to the attachment of a 2-handle identifies two boundary edges of Γ with two vertices in common. They can be expressed through our operators in a similar manner.

Handle operators have been extended to 3D in [LT97]. The operator that creates a new 3-ball (initialization operator) corresponds to the attachment of a 0-handle. Other operators identify two boundary faces (incident in exactly one 3-cell) of a cell 3-complex Γ triangulating a solid S. The attachment of a 1-handle (2-handle, or 3-handle) can be applied if the two identified boundary faces have no edges (some edges, or all edges) in common. The handle operators in 3D generalize the glue operator in [EW79]. They can be expressed in terms of the introduced atomic operators in a similar manner.

4.1.2 Operators for simplicial complexes

We present here a complete set of dimension-independent operators for modifying and updating simplicial complexes [DFW12].
They cannot explicitly be classified in homology-preserving and homology-modifying operators. We refer to Subsection 4.1.2.2 for the discussion about their effects on the homology.

The class of simplicial operators we consider consists of:

- *(elementary) excision*, which removes from the input simplicial complex a top simplex, and its inverse called *(elementary) inclusion*;

- *edge contraction*, which collapses two vertices of the input simplicial complex and the edge connecting them into a new vertex, and its inverse called *vertex split*.

Given a simplicial complex Σ and a top simplex σ in Σ, the (elementary) excision $excision(\sigma)$ consists of the removal of σ from Σ. Conversely, the (elementary) inclusion $inclusion(\sigma)$ on a simplicial complex Σ, already containing the boundary of σ, adds a new simplex σ to Σ .

An example of elementary excision is illustrated in Figure 4.4. Conversely, the transformation of the simplicial complex depicted on the right into the simplicial complex on the left represents an example of elementary inclusion.

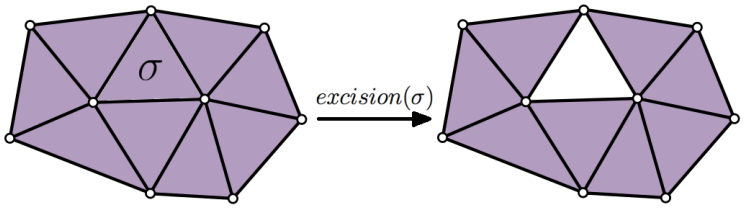


Figure 4.4: Effect of the elementary excision $excision(\sigma)$ on a 2-dimensional simplicial complex.

Edge contraction operator has been already introduced in Subsection 2.3.3.3. From now on, we just consider edge contraction operators collapsing two vertices of a simplicial complex Σ actually forming an edge of Σ. Given a simplicial complex Σ and two of its vertices u and v, the edge contraction $contraction(u, v, w)$ can be seen as the simplicial map f induced by the vertex map f_V taking vertices u and v to w and all other vertices to themselves. According to this description, the effect of $contraction(u, v, w)$ on Σ can be described as the substitution of each simplex σ containing u or v with a new simplex obtained from σ in which vertices u and v have been replaced by vertex w.

The vertex split operator is the inverse of the edge contraction and it is denoted as $split(u, v, w)$. Given a simplicial complex Σ and a vertex w of Σ, $split(u, v, w)$ creates an edge uv in place of a vertex w. In order to actually perform vertex split $split(u, v, w)$, the following information is required:

1. for each coface σ of w, i.e., $\sigma = w\, v_1 \cdots v_k$, simplices $u\, v_1 \cdots v_k$ or $v\, v_1 \cdots v_k$ or both with which σ will be substituted;

2. the simplices which will contain edge uv forming its star.

The effect of $split(u, v, w)$ on a simplicial complex Σ is the following:

- substitute, accordingly to 1., each coface $\sigma = w\, v_1 \cdots v_k$ of w with simplices $u\, v_1 \cdots v_k$ or $v\, v_1 \cdots v_k$ or both;

- introduce the simplices declared in 2. containing the edge uv.

A vertex split $split(u, v, w)$ operated on a simplicial Σ is considered valid if and only if its effect actually produces a simplicial complex.
An example of edge contraction is illustrated in Figure 4.5. Conversely, the transformation of the simplicial complex depicted on the right into the simplicial complex on the left represents an example of a vertex split.

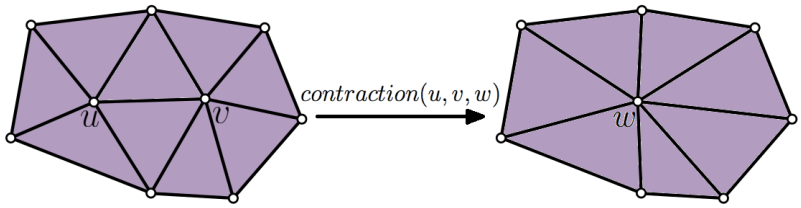


Figure 4.5: Effect of the edge contraction $contraction(u, v, w)$ on a 2-dimensional simplicial complex.

In the following, without loss of generality, we will often consider edge contractions $contraction(u, v, w)$ for which w coincides with v. We will simply denote this operator as $contraction(u, v)$, and its inverse operator as $split(u, v)$.

4.1.2.1 Minimality and completeness

By iteratively performing operators of (elementary) excision, any simplicial complex Σ can be transformed into the empty simplicial complex $\emptyset$. Dually, Σ can be retrieved starting from the empty simplicial complex $\emptyset$ and iteratively performing operators of (elementary) inclusion. As immediate consequence of these considerations we can claim the following proposition.

Proposition 4.3. *The operators of (elementary) excision and inclusion form a minimal and complete basis of operators for creating and updating simplicial complexes.*

Proposition 4.3 trivially ensures that the set of modifications consisting of elementary excisions, edge contractions and their inverse operators forms a complete basis for the set of operators for creating and updating simplicial complexes. At the same time, Proposition 4.3 implies that the considered basis is not minimal since each edge contraction and vertex split can be expressed as a combination of elementary excisions and inclusions.
Specifically, let us consider operator $contraction(u, v, w)$. According to the above definition, the execution of $contraction(u, v, w)$ on a simplicial complex Σ can be described as the performance of:

- the excisions, ordered by descending dimension, of all the simplices of Σ containing u or v;
- the inclusions, ordered by ascending dimension, of the just excised simplices in which vertices u and v have been replaced by w.

The reason why we choose to include edge contractions and vertex splits in our discussion is related to our purpose of building a multi-resolution model with an high expressive power. In fact, even if an edge contraction can be expressed as a suitable combination of excisions and inclusions, it cannot be obtained by using just one kind of these operators, for instance using only excisions. So, if during the construction of a multi-resolution model we choose as simplifications operators only elementary excisions, no edge contraction will be performed in the simplifying process. Since, as we will see in Section 4.2, the expressivity of a multi-resolution model deeply depends on the variety of simplification operators we can apply, the choice of not involving edge contraction operators could cause serious limitations in the obtained model.

4.1.2.2 Homology-preserving simplicial operators

Similarly to the cellular case, we would like to characterize the introduced simplicial operators according to their effects on the homology of the simplicial complex on which they are performed. To understand in which conditions these operators preserve the homology will be useful for the construction of an homology-preserving multi-resolution model for simplicial complexes.

According to the incremental method proposed in [DE93], adding to a simplicial complex a single simplex of dimension k either creates a non-bounding k-cycle or kills a $(k-1)$-homology class. In both the cases, the homology of the obtained simplicial complex is modified with respect to the one of the original simplicial complex. For this reason, an elementary inclusion and its undo (elementary excision) are homology-modifying operators.
In spite of this, a suitable combination of two elementary excisions gives rise to a homology-preserving simplicial operator called *(elementary) reduction* [MB09]. As already described in Subsection 2.3.3.2, given a simplicial complex Σ, an (elementary) reduction $reduction(\sigma, \tau)$ removes a pair of simplices σ and τ of dimension k and $k+1$ respectively if the immediate coboundary $\mathrm{cbd}_\Sigma\, \sigma$ of σ consists only of τ. Given a reduction operator $reduction(\sigma, \tau)$, we denote its inverse, which suitably adds the pair of simplices σ, τ of consecutive dimensions, as $expansion(\sigma, \tau)$. As mentioned above and depicted in Figure 4.6, a reduction operator, and so its inverse, does not modify the homology and it can be considered as the combination of the elementary excisions of τ and of σ.

Edge contraction is not, in general, a homology-preserving operator. Given a simplicial complex Σ, an edge uv in Σ satisfies the *link condition* if and only if

$$\mathrm{link}_\Sigma\, uv = \mathrm{link}_\Sigma\, u \cap \mathrm{link}_\Sigma\, v$$

As stated by Theorem 2.11 in Subsection 2.3.3.3, if the edge uv satisfies the link condition, then the edge contraction collapsing uv is homology-preserving. See Figure 2.6 for an example.

The above considerations lead us to choose as privileged class of homology-preserving

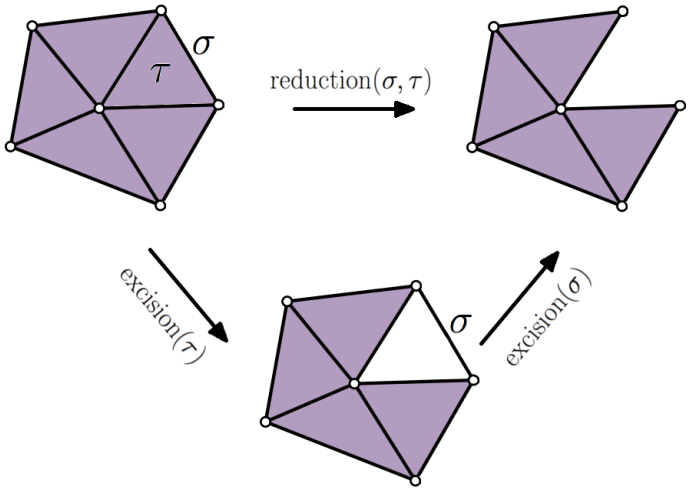


Figure 4.6: The reduction $reduction(\sigma, \tau)$ removing the simplices σ and τ and its decomposition in elementary excisions $excision(\tau)$, $excision(\sigma)$.

simplicial operators the one consisting of elementary reductions and edge contractions satisfying the link condition.

4.2 A general multi-resolution model

As mentioned before, the main goal of this chapter is to exploit multi-resolution models to efficiently retrieve cellular or simplicial homology of a complex and its homological generators at different levels of detail. To achieve this task, we provide a definition of multi-resolution model able to generalize both the cellular and the simplicial case.

4.2.1 Operators

Fundamental tools to build a multi-resolution model are a cell complex Γ and a class of operators. Up to now, we have informally considered an *operator (or modification)* μ as an operation that transforms a cell complex Γ into a new cell complex by replacing a set of cells Γ_1 of Γ with another set of cells Γ_2. Given an operator μ, we have referred to the *inverse* operator μ^{-1} with respect to μ as the one which transforms a cell complex containing Γ_2 by replacing the cells in Γ_2 with the ones in Γ_1 and obtaining a new cell complex.

In order to describe a multi-resolution model, these intuitive notions have to be turned in more rigorous definitions. A formal definition of an operator can be obtained by focusing on its effects in terms of the Hasse diagram of the cell complexes involved in its application. As we know, there is a complete correspondence between the Hasse diagram of a cell complex and its representation as an Incidence Graph (see Section 2.1). Thanks to this, in order to simplify the discussion, we can define the notion of operators in terms of Incidence Graph modifications without losing generality or creating ambiguities. In the

following, we will make use of the notion of graph subset.

Definition 4.4. Given a graph $G = (N, A)$, a *graph subset* H_1 of G is a pair (N_1, A_1) of sets such that $N_1 \subseteq N$ and $A_1 \subseteq A$.

Since the above definition does not ensure that the nodes of an arc of A belonging to A_1 are contained in N_1, a graph subset of G is not, in general, a subgraph of G. In spite of this, with a little abuse of notation, in the following we will denote the elements of N_1 and A_1 as the nodes and the arcs of graph subset H_1, respectively.

Let μ be an operator that applied to the cell complex Γ returns the cell complex Γ' by replacing Γ_1 with Γ_2 and let $G = (N, A)$, $G' = (N', A')$ be the Incidence Graphs of Γ, Γ', respectively. The effect of the operator μ can be described as a substitution of a graph subset of G in order to retrieve G'. Let $H_1 = (N_1, A_1)$ be the graph subset of G whose set of nodes $N_1 \subseteq N$ corresponds to the subset of cells $\Gamma_1 \subseteq \Gamma$ to be removed and whose set of arcs $A_1 \subseteq A$ represents the immediate boundary and coboundary relations in Γ of the cells in Γ_1. Let $H_2 = (N_2, A_2)$ be the graph subset of G' whose set of nodes $N_2 \subseteq N$ corresponds to the subset of cells $\Gamma_2 \subseteq \Gamma$ to be created and whose set of arcs $A_2 \subseteq A$ represents the immediate boundary and coboundary relations in Γ' of the cells in Γ_2. Thus, we can consider operator μ as the pair (H_1, H_2) and its effect as the replacement of H_1 of G with H_2 obtaining the Incidence Graph G' of Γ'.

Definition 4.5. An *operator* μ is a pair (H_1, H_2) of graph subsets $H_i = (N_i, A_i)$.
Let Γ be a cell complex and let $G = (N, A)$ be the Incidence Graph of Γ. The operator $\mu = (H_1, H_2)$ is said to be *feasible* on Γ if

- H_1 is a graph subset of G of Γ, and
- $G' := (N', A')$, where $N' = (N \setminus N_1) \cup N_2$ and $A' = (A \setminus A_1) \cup A_2$, is the Incidence Graph of a cell complex Γ'.

The above definition allows us to consider two operators performing exactly the same substitution of cells and boundary and coboundary relations even if applied on different cell complexes as the same operator. For instance, the two transformations depicted in Figure 4.7 collapsing edge e_2 into the vertex v' are actually the same operator μ, but applied on different cell complexes. At the same time, we want to distinguish between two operators which behave analogously but which involve different cells. The given definition allows, for instance, us to distinguish and to differently name the operator collapsing edge e_1 and the one collapsing edge e_2 of the cell complex depicted in the top left corner of Figure 4.7.

According to the intuitive description provided above, given an operator $\mu = (H_1, H_2)$, we define the *inverse* operator μ^{-1} of μ as the pair (H_2, H_1). An operator μ is called a *simplification operator* if it reduces the number of cells of cell complex Γ on which is applied. Conversely, if μ increases the number of cells of Γ, it is called a *refinement operator*.

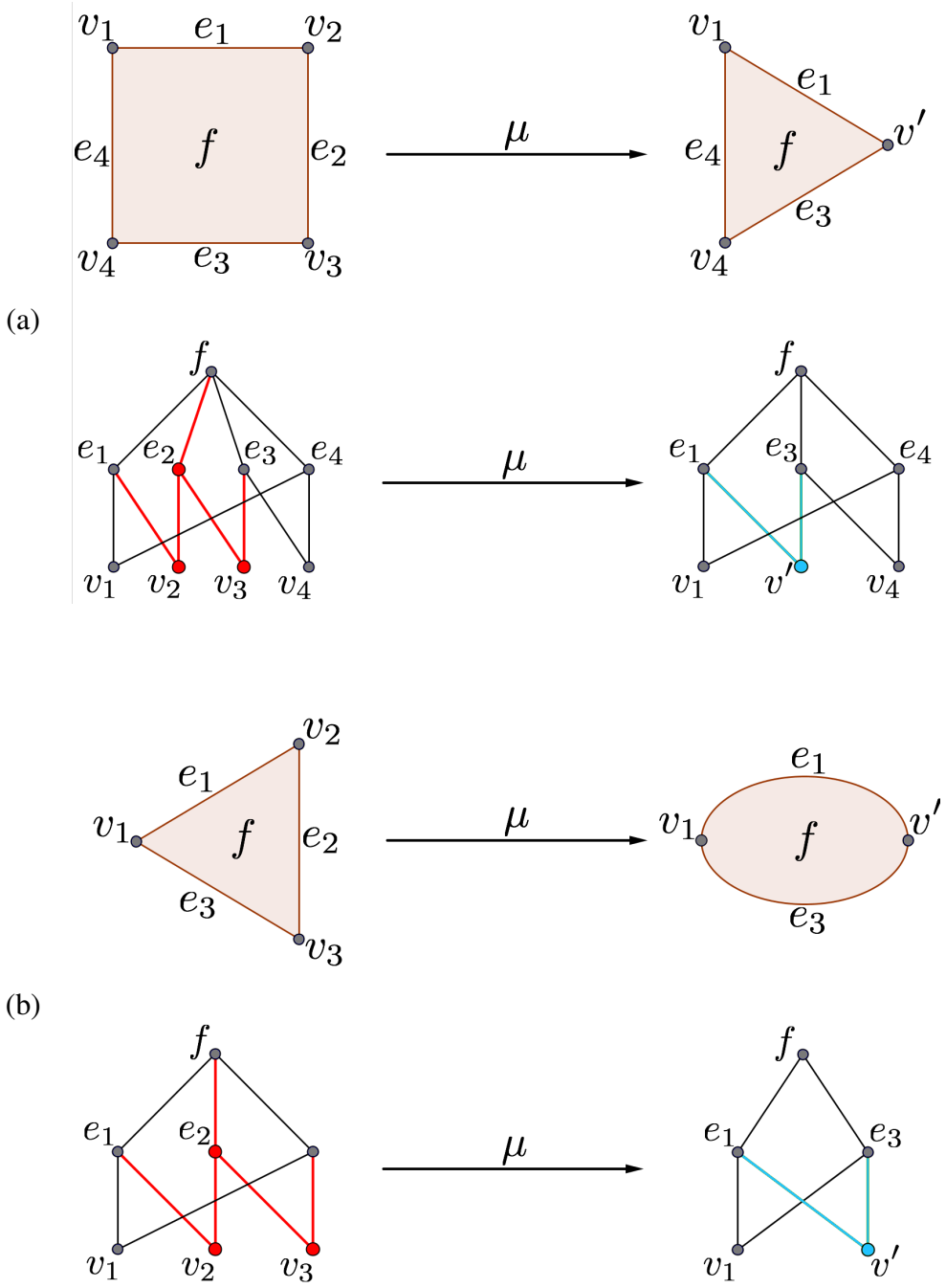


Figure 4.7: The operator μ contracting the edge e_2 applied in two different domains (a) and (b). For both the situations, the operation of μ is depicted in terms of modifications on the cell complex and on the Incidence Graph. The graph subsets $H_1 = (N_1, A_1)$ and $H_2 = (N_2, A_2)$ consist of the nodes and the arcs represented in red and light blue, respectively.

4.2.2 Multi-resolution cell complexes

Let Γ be a cell complex considered at full resolution. Suppose to iteratively apply on Γ a sequence of feasible simplification operators which do not reintroduce any already

removed cell. We define as

- *base complex* the coarse complex Γ_B obtained at the end of the simplification process,
- the set of the *refinement modifications* the set $\mathcal{M}$ of the refinement operators inverse with respect to the simplifications that have produced Γ_B from Γ.

For simplicity, we consider the creation of the coarse complex Γ_B as a refinement modification in $\mathcal{M}$. We denote this dummy refinement modification as μ_0 (i.e., $\mu_0 = (\emptyset, G_B)$, where G_B is the Incidence Graph of Γ_B).

Given an operator $\mu = (H_1, H_2)$ in $\mathcal{M}$, we define C_μ as the set of the cells corresponding to the extreme nodes of the arcs in A_1 and in A_2 which do not belongs to N_1. Intuitively, the set C_μ consists of the cells whose immediate boundary or immediate coboundary has been modified by the execution of μ, including also the cells removed by the application of μ.

Definition 4.6. Given two refinement modifications μ, μ' in $\mathcal{M}$, we say that μ *directly depends* on μ' if μ' creates a cell belonging to C_μ.

This notion of dependency between refinement modifications allows us to define a dependency relation $\mathcal{R}$ between the elements of $\mathcal{M}$. The *dependency relation* $\mathcal{R}$ is defined to be the transitive closure of the relation of direct dependency introduced above. Note that, during the coarsening step to obtain Γ_B from Γ, a cell can be removed just once, the same cell is never introduced twice by the refinements in $\mathcal{M}$. Thus, $\mathcal{R}$ is a partial order relation.

The three components introduced above allow us to define a multi-resolution model.

Definition 4.7. According to the introduced notations, we define the *Multi-Resolution Cell Complex* ($MRCC$) as the triple $(\Gamma_B, \mathcal{M}, \mathcal{R})$.

Since the dependency relation $\mathcal{R}$ is a partial order relation, an $MRCC$ can be encoded through a graph structure representing the set of refinement modifications and the direct dependency relation between pairs of modifications. Thus, an $MRCC$ can be encoded by using a *Direct Acyclic Graph* (DAG), in which

- the root encodes the creation of the base complex Γ_B (i.e., modification μ_0);
- the nodes are in one-to-one correspondence with the refinement modifications in $\mathcal{M}$;
- the arcs represent the direct dependency relation generating $\mathcal{R}$, i.e., given two distinct nodes μ and μ', arc (μ', μ) exists iff μ directly depends on μ'.

The data structure encoding such a DAG has to store the information required to retrieve the associated $MRCC$. Specifically, each node of the DAG has to contain the necessary data to correctly perform the corresponding refinement modification. This can be

obtained by encoding, for each node of the DAG, the two graph subsets H_1, H_2 which completely describe the correspondent refinement $\mu = (H_1, H_2)$. For the root, representing the modification $\mu_0 = (\emptyset, G_B)$, storing this information is equivalent to encode the Incidence Graph G_B of the coarse complex Γ_B.

Mainly due to the general context in which the $MRCC$ has been defined, the proposed encoding leads to a data structure with a high storage cost. Specific compact encoding of a multi-resolution model can be defined by considering specific refinement modifications (see Subsection 4.3.2 and 4.4.2).

The DAG encoding an $MRCC$ can be built by starting from cell complex Γ at the finest resolution. A sequence of simplifications is applied on Γ. Information about all the simplifications performed are saved in a stack. Once the sequence of simplifications terminates, the DAG is initialized with the root node storing the Incidence Graph G_B of the cell complex Γ_B at the coarsest resolution. Then, for each element in the stack, a new node in the DAG and the arcs connecting it to the nodes already in the DAG are created and the information required to correctly perform the refinement corresponding to the created node are stored.

4.2.3 Selective refinement extraction

From an $MRCC$, a large number of complexes at intermediate resolution can be obtained by applying sequences of refinement modifications in $\mathcal{M}$ to the base complex Γ_B.

Definition 4.8. A sequence $U = (\mu_0, \mu_1, \mu_2, \cdots, \mu_m)$ of refinements in $\mathcal{M}$ is *feasible* if μ_1 is feasible on the base complex Γ_B and each refinement μ_i, $1 \leq i \leq m$ is feasible on the complex Γ_{i-1} obtained from the base complex Γ_B by applying on it the sequence $(\mu_1, \mu_2, \cdots, \mu_{i-1})$.

Definition 4.9. Given a feasible sequence $U = (\mu_0, \mu_1, \mu_2, \cdots, \mu_m)$ of refinements in $\mathcal{M}$, the *front complex* Γ_U associated with U is the complex obtained from the base complex Γ_B by applying on it the sequence of refinements $(\mu_1, \mu_2, \cdots, \mu_m)$.

The front complex represents the topological structure of the cell complex Γ at an intermediate level of detail. If the feasible sequence U contains all the refinements in $\mathcal{M}$, then the front complex Γ_U associated with U is the same as the complex Γ at full resolution.
The following results characterize the set of the front complexes that can be extracted from an $MRCC$ (Proposition 4.11) and ensure us that a feasible sequence of refinements produces the same front complex independently from the order in which the modifications are operated (Proposition 4.13).

Thanks to the dependency relation $\mathcal{R}$, the feasibility of a refinement on a front complex Γ_U can be quickly checked.

Lemma 4.10. *Let Γ_U be the cell complex obtained by applying a feasible sequence U of refinements on the base complex Γ_B. We have that a refinement μ in $\mathcal{M}$ is feasible on the complex Γ_U if and only if U contains the set of the refinement modifications from which μ directly depends.*

Proof. Since during the simplification process no removed cell can be reintroduced, U contains the set of the refinement modifications from which $\mu = (H_1, H_2)$ directly depends if and only if the set of cells C_μ is contained in Γ_U. Further, the set C_μ consists exactly of the cells required to perform the substitution of H_1 with H_2 in the Incidence Graph of Γ_U. So, Γ_U contains the cells of C_μ if and only if the feasibility conditions of μ are satisfied. Finally, the fact that the operator μ is the inverse of an operator performed in the simplification process guarantees that the performance of μ on Γ_U returns a cell complex. □

A large number of adaptive morphological representations can be extracted from the multi-resolution model defined by the triple $(\Gamma_B, \mathcal{M}, \mathcal{R})$ by considering the closed sets of refinements in $\mathcal{M}$ under the dependency relation $\mathcal{R}$.

Proposition 4.11. *The sequence $U = (\mu_0, \mu_1, \mu_2, \cdots, \mu_m)$ of refinements in $\mathcal{M}$ is feasible if and only if the set $\mathcal{U} = \{\mu_0, \mu_1, \mu_2, \cdots, \mu_m\}$ is closed with respect to the dependency relation $\mathcal{R}$.*

Proof. $U = (\mu_0, \mu_1, \mu_2, \cdots, \mu_m)$ is a feasible sequence of refinements in $\mathcal{M}$ if and only if, for each i, the refinement μ_i is feasible on the cell complex $\Gamma_{U_{i-1}}$, where $U_{i-1} = (\mu_0, \mu_1, \mu_2, \cdots, \mu_{i_1})$. By Lemma 4.10, this holds if and only if, for each i, the set U_{i-1}, and thus the set U, contains all refinements on which the refinement μ_i directly depends. This is true if and only if the set $\mathcal{U} = \{\mu_0, \mu_1, \mu_2, \cdots, \mu_m\}$ is closed with respect to the dependency relation $\mathcal{R}$. □

Now, we want to show that the execution of a sequence of refinements with respect to two different feasible orders leads to the same front complex.
Let (μ, μ'), (μ', μ) be two feasible sequences of refinement operators on a front complex Γ_U. Since both sequences are feasible on Γ_U, both the operators have to be feasible on Γ_U. This ensures that μ and μ' are independent with respect to the dependency relation $\mathcal{R}$. The operators μ and μ' are said *interchangeable* if the sequences (μ, μ') and (μ', μ) performed on Γ_U produce the same cell complex.

Lemma 4.12. *Let μ, μ' be two feasible refinements on a front complex Γ_U. If μ and μ' are independent, then they are interchangeable.*

Proof. Let us consider the operators μ and μ' as the pairs (H_1, H_2) and (H'_1, H'_2), respectively. Since they are independent, no cell created by μ belongs to $C_{\mu'}$ and, conversely, no cell created by μ' belongs to C_μ. So, in this context, both sequences of operators (μ, μ') and (μ', μ) can be described as a single refinement operator acting on the Incidence Graph of Γ_U and substituting the graph subset obtained by the union of H_1 and H'_1 with the one obtained by the union of H_2 and H'_2. □

Proposition 4.13. *Let $U = (\mu_0, \mu_1, \mu_2, \cdots, \mu_m)$ be a feasible sequence of refinements in $\mathcal{M}$. If, applying the permutation $\mu_0, \mu_{i_1}, \mu_{i_2}, \cdots, \mu_{i_m}$ of the refinements in U, the sequence $V = (\mu_0, \mu_{i_1}, \mu_{i_2}, \cdots, \mu_{i_m})$ is a feasible sequence of refinements in $\mathcal{M}$, then the front complexes Γ_U and Γ_V are equal.*

Proof. The sequence V is feasible if the permutation that defines V starting from U is such that each refinement μ_{i_j} is feasible in sequence V. This means that each refinement μ_{i_k} on which μ_{i_j} depends has a position $i_k < i_j$ in V. The permutation defining V from U is a composition of adjacent transpositions of two independent refinements (composition of permutations obtained by reversing the order of two consecutive refinements). By Lemma 4.12, for each such transposition, the associated front complex before and after the transposition remains unchanged. Thus, the front complex Γ_V associated with the sequence V is the same as the front complex Γ_U associated with the sequence U. □

From a multi-resolution model, it is possible to dynamically extract representations of the original cell complex Γ at uniform and variable resolutions. The basic query for extracting a single-resolution representation from a multi-resolution model is known as selective refinement. A *selective refinement* query consists of extracting from a multi-resolution model a complex with the minimum number of cells, satisfying some application-dependent criterion. This criterion can be formalized by defining a Boolean function φ over all nodes of a multi-resolution model, such that the value of φ is true on nodes which satisfy the criterion, and false otherwise. The same value of φ is associated with the cells created by the modification encoded in the node of the $MRCC$.

The selective refinement query consists of extracting from the $MRCC$ an intermediate complex of minimum size among the complexes encoded in the $MRCC$ that satisfies φ. Equivalently, it consists of extracting a minimal closed set $\mathcal{U}$ of modifications in $\mathcal{M}$, such that the corresponding complex satisfies φ. Such closed set of modifications uniquely determines a front complex, which is obtained from the base complex Γ_B, by applying to it all modifications from $\mathcal{U}$ in any order that is consistent with the partial order defined by the dependency relation.

Criterion φ can be defined based on various conditions posed on the cells in the extracted complex, like the size of the cell (which may be expressed as the maximum distance between its vertices or the diameter of its bounding box) or the portion of the complex in which full resolution is required (while in the rest of the complex, the resolution may be arbitrarily low).

4.3 Cellular homology computation through a multi-resolution model

In this section, we define a multi-resolution model for cell complexes based on the set of atomic modeling operators introduced in Subsection 4.1.1. Further, we define a version of this model based on the subset of the proposed modeling operators which preserve homology and we describe its implementation. We apply the homology-preserving multi-resolution model to enhance the efficiency in extracting homology generators at different resolutions. To this aim, we propose an algorithm which computes homology generators on the coarsest representation of the original complex, and uses the hierarchical model to propagate them to complexes at any intermediate resolution, and we prove its correctness. For a more detailed description of the contributions presented in this section, please refer

to [ČomićDIF14].

4.3.1 The Hierarchical Cell Complex (*HCC*)

In this subsection, we introduce a multi-resolution model for cell complexes based on the simplification and refinement operators described in Subsection 4.1.1. We call this model *Hierarchical Cell Complex (HCC)*, and we discuss its major properties.

This model is generated from a cell complex Γ considered at full resolution by iteratively applying simplification operators $KiC(i+1)C$ and $KiCiCycle$. By applying first the homology-preserving operators, we obtain a complex Γ' having the same homology as the original complex Γ but with fewer cells and such that no homology-preserving operator is feasible on Γ'. By applying next the homology-modifying operators to iteratively remove the cells of Γ', the homology is affected at each step and the process is repeated until a complex is obtained that has at least one i-cell, $0 \leq i \leq d$. At each step, when we apply a homology-modifying operator, we remove a top cell from the complex. After each application of a homology-modifying operator, we perform feasible homology-preserving ones.
Analogously to the description of multi-resolution model given in Section 4.2, we define a *Hierarchical Cell Complex (HCC)* as a triple $(\Gamma_B, \mathcal{M}, \mathcal{R})$, where, given a cell complex Γ, Γ_B is the result of the sequence of simplification operators, $\mathcal{M}$ is the set of the refinement modifications which are inverse with respect to the performed simplification modifications and $\mathcal{R}$ is the dependency relation between the modifications in $\mathcal{M}$.

A first step to properly build this model and retrieve its dependency relations is to describe the involved operators in the form proposed in Subsection 4.2.1. Let us consider the simplification operators $KiC(i+1)C_{co}(q,p,p')$ and $KiCiCycle$. The representation of the remaining instances and of the inverse operators can be easily recollected to the one of the considered operators. Our task is to represent the above mentioned operators as a pair (H_1, H_2) of graph subsets, where $H_i = (N_i, A_i)$.
The operator $KiC(i+1)C_{co}(q,p,p')$ can be described as the pair (H_1, H_2) where:

- N_1 is the set of nodes corresponding to cells p and q;
- A_1 consists of the arcs representing the relations of immediate boundary and coboundary of p and q;
- N_2 is the empty set;
- A_2 consists of the arcs representing the newly introduced boundary relations between the cells in the immediate boundary of p and p'.

The operator $KiCiCycle$, creating a cell p, can be described as the pair (H_1, H_2) where:

- N_1 consists of the node corresponding to cell p;

- A_1 consists of the arcs representing the relation of immediate boundary of p;

- the sets N_2 and A_2 are both empty.

As mentioned in Subsection 4.2.2, in order to define the relation $\mathcal{R}$, it is enough to characterize, for each refinement modification $\mu \in \mathcal{M}$, the set C_μ. With respect to the operators considered in this section,

- if μ is the homology-preserving refinement $MiC(i+1)C$ creating cells p and q, C_μ consists of the cells in the immediate boundary and coboundary of q and p except for q and p themselves;

- if μ is the homology-modifying refinement $MiCiCycle$, creating a cell p, C_μ consists of the cells in the immediate boundary of p.

This definition and the results proven in Subsection 4.2.3 ensure us that the multi-resolution model HCC is well-defined and different selective refinements can be extracted from it. Analogously to the general model $MRCC$, the operation of any sequence U of refinement modifications performed in a consistent order and closed with respect to $\mathcal{R}$ leads to the front complex Γ_U representing Γ at a certain level of detail. As before, HCC model allows to manage different refinement queries and to extract representations of the original cell complex at uniform and variable resolutions.

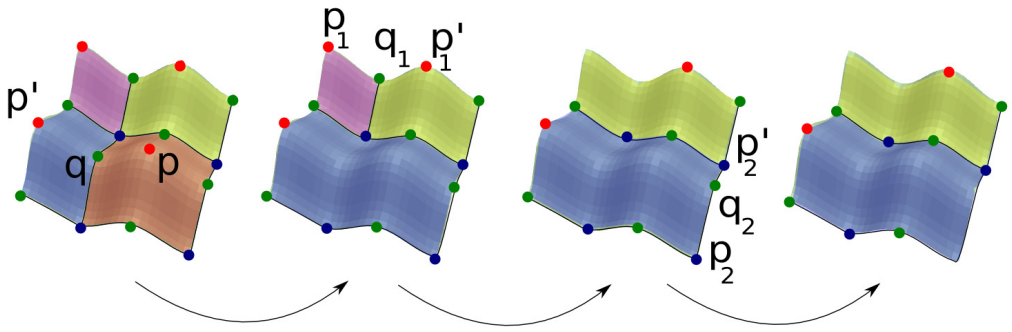


Figure 4.8: A sequence of operators consisting of (from left to right) $K1C2C_{re}\,(q, p, p')$, $K1C2C_{re}(q_1, p_1, p'_1)$ and $K0C1C_{co}(q_2, p_2, p'_2)$ on a 2-dimensional cell complex. Blue dots (e.g., p_2 and p'_2) correspond to 0-cells, green dots (e.g., q, q_1 and q_2) to 1-cells and red dots (e.g., p, p', p_1 and p'_1) to 2-cells.

Figure 4.8 illustrates a sequence consisting of the simplification operators $K1C2C_{re}$ (q, p, p'), $K1C2C_{re}$ (q_1, p_1, p'_1) and $K0C1C_{co}$ (q_2, p_2, p'_2) operated on a 2-dimensional cell complex Γ. In Figure 4.9, the HCC built from this sequence of simplifications is depicted. We can notice that each node, with the exception of the root, represents a refinement dual to a simplification applied in Figure 4.8. Each closed subset of refinement modifications produces a different cell complex at intermediate resolution.

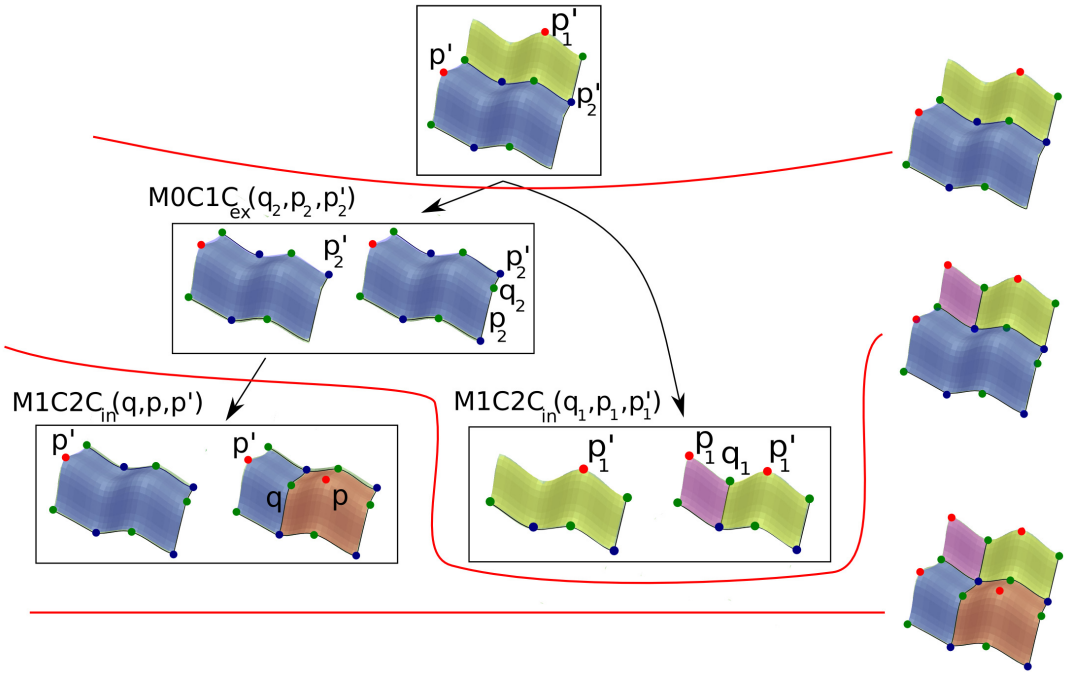


Figure 4.9: An example of an HCC built from the simplification process illustrated in Figure 4.8. The top level of the HCC is the root node encoding the complex at the coarsest resolution. At the bottom level are two $M1C2C_{in}$ operators. The $M1C2C_{in}(q, p, p')$ depends on the $M0C1C_{ex}(q_2, p_2, p'_2)$ and the $M1C2C_{in}(q_1, p_1, p'_1)$ depends only on the root. Blue dots (e.g., p_2 and p'_2) correspond to 0-cells, green dots (e.g., q, q_1 and q_2) to 1-cells and red dots (e.g., p, p' and p'_1) to 2-cells. On the right, three different complexes are shown, obtained by performing different closed sets of refinements on the HCC as indicated by the red lines.

4.3.2 The Homology-preserving Hierarchical Cell Complex (*HHCC*)

Since our task is to use the defined multi-resolution model to quickly retrieve the homological information of any complexes represented in it, we focus here on the encoding and the implementation of a multi-resolution model based only on homology-preserving operators. This model is called *Homology-preserving Hierarchical Cell Complex (HHCC)* and it can be built iteratively performing feasible homology-preserving operators $KiC(i{+}1)C$ on a cell complex Γ.

4.3.2.1 Encoding an *HHCC*

As mentioned in Subsection 4.2.2, each multi-resolution model can be represented by a Direct Acyclic Graph (DAG). Here, we describe dimension-independent encoding for an $HHCC$ that we have developed [ČomićDIF14]. The two data structures, called *explicit* and *implicit*, differentiate in the way the cells, involved in each refinement, are encoded. In the explicit data structure, we encode all the cells involved in each refinement explicitly. This leads to an efficient reconstruction of the information required to perform each

refinement μ. On the other side, we tried to reduce the memory consumption of the whole data structure designing a more compact representation for the set of cells. The two encodings should represent the usual tradeoff between memory consumption and runtime efficiency.

Both explicit and implicit encodings store the coarse cell complex Γ_B in their root encoded as an Incidence Graph G_B. Each node of the DAG represents a refinement modification $\mu = MiC(i+1)C$; μ encodes information about the refinement as well as its dependencies on the other nodes.

The information stored for each node μ, in both explicit and implicit representations, are:

- a flag indicating the type of the refinement modification ($expand$ or $insert$), and the value of i;
- cells p and q to be reintroduced;
- a pointer to the DAG node corresponding to the modification μ' introducing p';
- an array, called $ancestors$, containing the pointers to the parents of μ
- an array, called $descendants$, containing the pointers to the children of μ.

These information identify the refinement operator represented by the DAG node μ.

Both our data structures encoding an $HHCC$ are built by starting from the Incidence Graph G describing the cell complex Γ at the finest resolution. A sequence of simplifications is applied on G until there are no more feasible simplifications, or the size of the resulting complex is below a predefined threshold. Information about all the simplifications performed are saved in a stack. Each element of the stack represents a generic simplification and stores all the nodes of G involved in its performance. Once there are no more feasible simplifications, the base graph G_B at the coarsest resolution, is stored in the root of the DAG. To perform a refinement μ correctly we need to represent the set of cells involved in the refinement. On their representation the two data structures differ. In the following, without loss of generality, we consider the case in which μ in $\mathcal{M}$ is a $MiC(i+1)C_{ex}(q,p,p')$ operator. Let us suppose that the operator μ^{-1}, performed during the simplification sequence generating the $HHCC$, has been applied to the cell complex Γ'' transforming it into the cell complex Γ'. In order to retrieve the refinement operators from which μ directly depends and to correctly perform the operator μ, we consider the following sets of cells:

- S consisting of the $(i+2)$-cells in Γ'' which are cofaces of q;
- Z consisting of the $(i-1)$-cells in Γ'' which are faces of p;
- R consisting of the $(i+1)$-cells in Γ'' which are cofaces of p but not of p';
- R' consisting of the $(i+1)$-cells in Γ'' different from q which are cofaces both of p and p'.

By knowing these sets of cells, p, p' and q, the dependency relations of μ and the effect of the execution of μ can be easily described. Specifically, C_μ coincides with $S \cup Z \cup R \cup R' \cup \{p'\}$.
Let us consider a front complex Γ_U of the $HHCC$ on which the refinement operator μ is feasible. The effect of μ on Γ_U is the following:

- cells p and q are added to Γ_U;
- set S is declared as the immediate coboundary of q;
- cells p and p' are declared as the cells in the boundary of p
- set $R \cup R' \cup \{q\}$ is declared as the immediate coboundary of q;
- set Z is declared as the immediate boundary of q;
- immediate coboundary of p' is updated by removing the cells in R and adding cell q.

In the *explicit data structure*, cells in S, Z, R and R' are encoded explicitly for each node in the structure. During the building process, a new node in the DAG is created for each element in the stack. The type of the node is initialized based on the type of the simplification operator, and also the cells p and q removed by the operator and the pointer to the third cell p' are encoded. Then, all cells in sets S, Z, R and R' are explicitly referred to a pointer (4 bytes per cell) and one list of pointers is stored for each set S, Z, R and R'. The resulting structure allows for a faster application of the refinement modifications since, for each modification, the sets S, Z, R and R' are ready to be used. The resulting storage cost for the cells in sets S, Z, R and R' in the explicit data structure is equal to $4|SZRR'|$ bytes, where $|SZRR'|$ indicates the number of cells in the union of the sets S, Z, R and R'.

Implicit encoding
The nodes encoded in the implicit data structure represent cells in the sets S, Z, R and R' in an implicit manner referring to the DAG nodes introducing them or to the coarse Incidence Graph if present in G_B from the beginning. Intuitively, each node in the Incidence Graph G_U of a front complex Γ_U, on which the refinement μ will be performed, has been inserted in G_U from another refinement μ_k or it was in the base graph G_B. To implicitly refer the cells introduced by another modification, in each node μ a two-bit-flag vector $pq\text{-}ancestors$ with the same size as $ancestors$ is defined. Let us consider the refinement $\mu = MiC(i+1)C(q,p,p')$ depending from a refinement $ancestors[j] = \mu_1 = MiC(i+1)C(q_1,p_1,p'_1)$,

- $pq\text{-}ancestors[j] = 0$ if μ depends on cell p_1 introduced by μ_1
- $pq\text{-}ancestors[j] = 1$ if μ depends on cell q_1 introduced by μ_1
- $pq\text{-}ancestors[j] = 2$ if μ depends on both p_1 and q_1.

Thus, vector $pq\text{-}ancestors[j]$ offers a compact way to refer cells introduced by other nodes in the DAG. When the cells to refer correspond to nodes in the base graph G_B, four bitvectors are stored in DAG node μ. Let i, $i+1$ be the dimension of cells p and q introduced by $\mu = MiC(i+1)C(q,p,p')$, respectively:

- bitvector B_S has length equal to the number of $(i+2)$-nodes in G_B;
- bitvector B_Z has length equal to the number of $(i-1)$-nodes in G_B;
- bitvectors B_R and $B_{R'}$ have length equal to the number of $(i+1)$-nodes in G_B.

For each bitvector (e.g. B_S), the j^{th} bit flag is equal to 1 if the j^{th} $(i-2)$-node of G_B is in S for modification μ. B_Z, B_R and $B_{R'}$ are similarly defined. Vector $pq\text{-}ancestors$, bitvectors B_S, B_Z, B_R and $B_{R'}$, and array $ancestor$ provide an implicit encoding of sets S, Z, R and R' required by modification μ. These latter sets are reconstructed when applying the modification in the extraction phase.

The implicit data structure is built by starting from the Incidence Graph G of the cell complex Γ at the finest resolution. A stack of simplifications is constructed iteratively by applying the operators in a sequence. Once no more feasible simplifications are feasible, the base graph G_B at the coarsest resolution is stored in the root of the DAG. For each element in the stack, a new node in the DAG is created, the type of the node is initialized based on the type of the simplification operator. Also cells p and q removed by a simplification as well as the pointer to the third cell p' are encoded. Then, the following process is repeated for all cells in sets S, Z, R and R'. For each cell p_1, belonging to any of these sets:

- if p_1 belongs to G_B, we set its corresponding bit in the bitvector (S, Z, R or R') to 1;
- otherwise we store in $ancestors$ a pointer to modification μ_1 in the DAG that introduces p_1 and we insert in the corresponding position in $pq\text{-}ancestors$ the value 0,1 or 2 depending on whether μ depends on first, second or both cells introduced by μ.

The storage cost for each node in the implicit data structure depends on the number of cells in sets S, Z, R and R' and on the number of nodes in the base graph G_B. Specifically, for each DAG node, it requires:

- 1 bit for each node in N_B, with a total cost of $\frac{1}{8}|N_B|$ bytes, where $|N_B|$ is the total number of nodes of G_B,
- 2 bits for each cell in $S \cup Z \cup R \cup R'$, total cost $\frac{1}{4}|SZRR'|$ byte, where $|SZRR'|$ is the cardinality of set $S \cup Z \cup R \cup R'$.

Thus, to encode such information in the implicit data structure $\frac{1}{8}(|G_B|+2|SZRR'|)$ bytes are required. Then, comparing the two storage costs, the implicit data structure is more

efficient in terms of memory requirements when, approximating the number of nodes in the base graph, $|G_B| < 30|SZRR'|$.

Experimental evaluations comparing explicit and implicit data structures have been developed for 2- and 3-dimensional cell complexes [Iur14]. The explicit data structure requires 20-30% more memory with respect to the implicit one in 2D, but differences are higher in the 3D case, where the explicit data structure requires three times more memory than the implicit structure. On the other hand, the direct access to the nodes in sets S, Z, R and R' for each modification μ speeds up the navigation inside the explicit data structure, and thus we obtain lower extraction times.

An implementation of the $HHCC$, based on an implicit encoding, has been developed in [ČomićDIF14]. by using a desktop computer with a 3.2Ghz processor and 16GB of memory. All complexes are simplicial complexes, that become cell complexes after undergoing some simplification.

The storage cost of the $HHCC$ encoding structure is about 25% less than the storage cost of the Incidence Graph representing the complex at full resolution (the original complex). We have considered complexes with between 40K and 3.2M top cells in 2D case, and between 700K and 6M top cells in the 3D case, as shown in Table 4.1 (column $Cells$). The storage cost of the original cell complex is between 4.8MB and 398MB for 2D complexes, and between 118MB and 980MB for 3D complexes (column *Complex cost*). The storage cost of the corresponding $HHCC$ is between 3.3MB and 273MB, and between 84MB and 720MB (column *HHCC cost*), respectively.

Dataset	Cells	Complex cost (MB)	$HHCC$ cost (MB)	Homology
Genus3	40K	4.8	3.3	(1,6,1)
Fertility	1.4M	176	122	(1,8,1)
Hand	2.1M	256	177	(1,2,0)
Buddha	3.2M	398	273	(1,208,1)
Skull	748K	118	84	(1,2,1,0)
Fert-Solid	6.2M	980	720	(1,4,0,0)

Table 4.1: Four 2D shapes and two volumetric datasets used in our experiments. The columns from left to right indicate: the name of the dataset (*Dataset*), the number of the top cells in the datasets (*Cells*), the storage cost of the original cell complex (*Complex cost*), the storage cost of the $HHCC$ (*HHCC cost*), the Betti numbers (*Homology*).

4.3.3 Homology computation through an *HHCC*

An $HHCC$ can be exploited for computing homology and homology generators of a cell complex at various resolutions.

4.3.3.1 Computing homology and homology generators

In this subsection, we are interested in computing the homology groups $H_k(\Gamma;\mathbb{Z}_2)$ of a cell complex Γ with the coefficients in $\mathbb{Z}_2$. As described in [Hat02], this corresponds to computing the Betti numbers of Γ with coefficients in $\mathbb{Z}_2$. Moreover, for each $k = 0,\ldots,d$, we are interested in computing the homology generators of degree k, that we call H_k *generators*. The H_k generators are the generators of the $\mathbb{Z}_2$-vector space $H_k(\Gamma;\mathbb{Z}_2)$, and they represent the independent non-bounding k-cycles in Γ. Each H_k generator of a cell complex Γ is a linear combination of k-cells in Γ with coefficients in $\mathbb{Z}_2$. In Figure 4.10(a), two H_1 generators are shown as linear combination of 1-cells. The first generator is composed of the set of blue (bold) edges and the other one of the set of red (dotted) edges.

In an $HHCC$, any front complex Γ_U is obtained from the base complex Γ_B by applying a sequence of homology-preserving refinement modifications $MiC(i+1)C$. In an $HHCC$ thus, the homology of the base complex is the same as the homology of any other complex implicitly encoded in the $HHCC$. We use the Smith Normal Form (SNF) reduction algorithm (Subsection 1.2.1.3) to compute homology and homology generators with coefficients in $\mathbb{Z}_2$ on the base complex Γ_B. Then, at each application of the refinement, we modify the homology generators in the currently extracted front complex Γ_U according to Algorithm 5 and to Proposition 4.14 described below.

Proposition 4.14. *Let Γ' be a d-dimensional cell complex, Γ'' the cell complex obtained from Γ' by applying $MiC(i+1)C(q,p,p')$. For a fixed $k \in \{0,\cdots,d\}$, let $B' = \{[c'_1]_{\Gamma'},\cdots,[c'_l]_{\Gamma'}\}$ be a basis for $H_k(\Gamma';\mathbb{Z}_2)$, then*

1) *if $k \neq i+1$, $[c'_1]_{\Gamma''},\cdots,[c'_l]_{\Gamma''}$ is a basis for $H_k(\Gamma'';\mathbb{Z}_2)$;*
2) *if $k = i+1$, $B'' = \{[c''_1]_{\Gamma''},\cdots,[c''_l]_{\Gamma''}\}$ is a basis for $H_{i+1}(\Gamma'';\mathbb{Z}_2)$, where, if $[c']_{\Gamma'} \in B'$, $[c'']_{\Gamma''} \in B''$ is defined by*

$$c'' = \begin{cases} c' & \text{if } \partial_{\Gamma''}c' \equiv 0 \ (\text{mod } 2) \\ c'+q & \text{otherwise} \end{cases}$$

Proof. See Appendix A. □

As we have formally proven, the operation of a refinement modification $MiC(i+1)C$ only affects the H_{i+1} generators.
Let us consider refinement modification $MiC(i+1)C_{ex}(q,p)$, which creates an i-cell p and an $(i+1)$-cell q (the case of a refinement $MiC(i+1)C_{in}$ is entirely dual). Operator $MiC(i+1)C_{ex}(q,p)$ is applied on a complex Γ' producing a refined complex Γ''. Algorithm 5 checks if the introduced $(i+1)$-cell q in Γ'' breaks an $(i+1)$-cycle corresponding to an H_{i+1} generator in Γ'. This is done by considering the number of $(i+1)$-cells in the coboundary of i-cell p that are involved in H_{i+1} generators. This idea is illustrated in Figures 4.10(b) and (c), where we show two different applications of operator $M0C1C_{ex}$ to the same 2-complex (torus), depicted in Figure 4.10(a). The application of operator $M0C1C_{ex}(q_1,p_1,p')$, illustrated in Figure 4.10(b), modifies one of the two H_1 generators

in the torus. We can notice that the new 0-cell p_1 has exactly one incident 1-cell belonging to the blue (bold) 1-chain. Thus the 1-cycle has been broken by the refinement and 1-cell q_1 is added to the 1-chain to reconstruct the cycle. On the contrary, the application of operator $M0C1C_{ex}(q_2, p_2, p')$, illustrated in Figure 4.10(c), does not affect the generators. Note that 0-cell p_2 has no incident 1-cell belonging to some H_1 generator.

Algorithm 5 $ExpandGenerators(p, q, \mathcal{G}_i)$

```
1: INPUT: p, i-cell
2: INPUT: q, (i + 1)-cell
3: INPUT: G_i, set of H_{i+1} generators
4: OUTPUT: G_i, set of H_{i+1} updated generators
5: // C is a map from a generator g to an integer m
6: C := empty map
7: // Extract the (i + 1)-cells on the coboundary of p
8: for all cofaces r of p do
9:    // G_{i,r} is the set of generators containing r
10:   G_{i,r} := getGeneratorsOn(r, G_i)
11:   // Consider the number of incidences between p and r
12:   for all generators g in G_{i,r} do
13:      C[g] := getIncidence(p, r) + C[g]
14: // Expand the generators on q if necessary
15: for all pairs (g, m) in C do
16:   if m is odd then
17:      addGenerator(g, q, G_i)
```

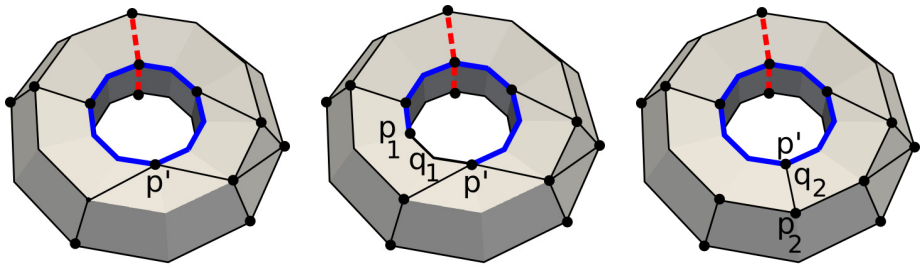
In the description of Algorithm $ExpandGenerators(p, q, \mathcal{G}_i)$, p and q denote, respectively, the i-cell and the $(i + 1)$-cell introduced by the refinement operator, and $\mathcal{G}_i$ represents the set of H_{i+1} generators of Γ'. The algorithm makes use of a map C from a generator g to an integer m, that, for each generator g, stores the number of $(i + 1)$-cells in the coboundary of i-cell p which also belong to g.

Algorithm 5 uses the following three functions:

- $getGeneratorsOn(r, \mathcal{G}_i)$, which returns the set of generators $\mathcal{G}_{i,r}$ containing cell r in their chain,
- $getIncidence(p, r)$, which returns the number of times i-cell p appears on the boundary of $(i + 1)$-cell r,
- $addToGenerator(g, q, \mathcal{G}_i)$, which updates the generators in $\mathcal{G}_i$ by adding $(i + 1)$-cell q to the $(i + 1)$-chain corresponding to g.

Algorithm $ExpandGenerators(p, q, \mathcal{G}_i)$ considers only the $(i + 1)$-cells in the coboundary of p that are part of one or more H_{i+1} generators. For each such $(i+1)$-cell r, $\mathcal{G}_{i,r}$ is the

set of generators that contain r ($getGeneratorsOn(r, \mathcal{G}_i)$). For each generator $g \in \mathcal{G}_{i,r}$, map C is updated by adding the number of times the i-cell p appears on the boundary of r ($getIncidence(p, r)$). Once all the $(i+1)$-cells in the coboundary of p have been examined, cell q is added to generator g only if the number m of incidences for g is odd ($addGenerator(g, q, \mathcal{G}_i)$).



(a) (b) (c)

Figure 4.10: (a) A cell complex representing a torus. Black dots represent 0-cells. Red (dotted) and blue (bold) edges correspond to the two H_1 generators. (b) Application of operator $M0C1C_{ex}(q_1, p_1, p')$, which affects one of the homology generators. (c) Application of operator $M0C1C_{ex}(q_2, p_2, p')$, which does not affect the homology generators.

By considering the used homology-preserving operators as compositions of the elementary operators introduced in [KMS98], the above presented method could be suitable extended to homology with coefficients in $\mathbb{Z}$.

4.3.3.2 Homology computation and generator extraction: experimental results

Several experiment have been performed to check the efficiency of the proposed approach. In a first set of experiments we have evaluated the time required to compute the homology and its generators of the original complex (the one at full resolution) by using the $HHCC$. To this aim, we first compute the homology generators on the base complex, encoded in the root of the $HHCC$. This computation requires between 8.3×10^{-5} and 8.8 seconds depending on the dataset (column *SNF Time* in Table 4.2). Then, we perform all the refinements in the $HHCC$, by applying when necessary the refinement of the generators as described in Subsection 4.3.3. This produces the representation of the complex at full resolution together with the homology generators. The total cost of this computation is the sum of the time required to compute the homology of the base complex (column *SNF Time*) and the time needed to fully refine the complex and its generators (column *Tot. Ref. Time*). This takes from a minimum of 0.15 to a maximum of 83.3 seconds. Applying the same SNF reduction directly on the original complex, requires about 2.6 hours on a relatively small complex (the dataset *Genus3*), while it results in very high computation times for the other datasets.

In Figure 4.11, we show the H_1 generators computed on two 2D shapes *Fertility* and *Hand* and, in Figures 4.13 (b) and (c), we show the H_1 and H_2 generators computed on the 3D *Skull* dataset.

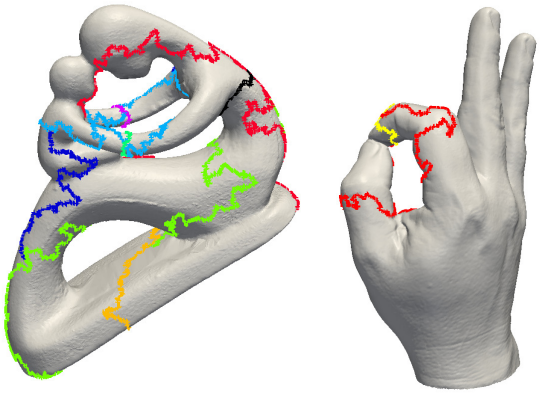
In a second set of experiments we have focused on extracting different representations of

Dataset	SNF Time	Tot. Ref. Time	Uniform Ref.	Uniform Time	Generators Ref.	Generators Time
Genus3	9.2×10^{-5}s	0.15s	4K 10K 16K	0.03s 0.07s 0.12s	5K	0.03s
Fertility	8.3×10^{-5}s	9.31s	144K 362K 579K	1.8s 4.6s 7.52s	68K	1.48s
Hand	9.8×10^{-5}s	14.9s	200K 500K 800K	2.6s 6.8s 11.2s	19K	1.6s
Buddha	0.02s	23.7s	320K 800K 1.2M	0.5s 4.3s 19.2s	162K	3.6s
Skull	0.007s	6.4s	75K 187K 299K	1.0s 2.9s 5.0s	191K	2.6s
Fert-Solid	8.8s	74.5s	1.2M 3.1M 4.9M	7.5s 29.1s 69.3s	267K	10.9s

Table 4.2: Experimental results obtained by refining four 2D shapes and two volumetric datasets and by computing homology generators on them through the Smith Normal Form (SNF) reduction. The columns from left to right indicate: the name of the dataset (*Dataset*), time required to compute the homology generators on the base complex (*SNF Time*), the time needed to extract the complex at full resolution and to expand all the generators (*Tot Ref Time*), the number of refinements and the time needed to extract the complex and the geometry of the generators at uniform level of detail (*Uniform Ref.* and *Uniform Time*) and the number of refinements and the time needed to extract the complex and the generators concentrating the resolution only in the neighborhood of the generators (*Generators Ref.*) and (*Generators Time*). The time is expressed in seconds.

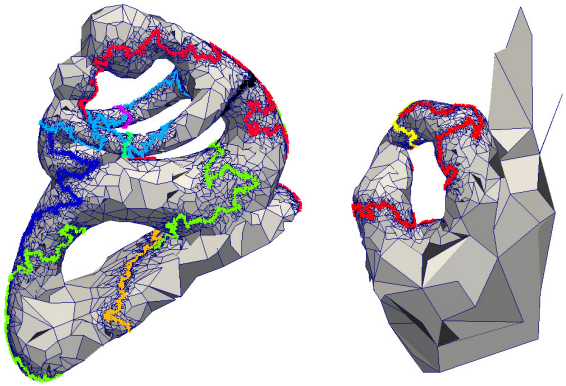
the complex by expanding the computed generators at different resolutions. We have considered first the extraction of representations at uniform resolution: we have extracted representations obtained from the base complex by applying approximatively 20%, 50% and 80% of the total possible refinements (column *Uniform Ref.* in Table 4.2).Refinements are forced to be evenly distributed inside the complex in order to obtain a uniformly refined complex. We can notice (see column *Uniform Time*) that the time required depends on the number of refinements performed and is between 0.03 and 7.5 seconds for extraction at 20% resolution and between 0.12 and 69.3 seconds for extraction at 80% resolution.

Then, we have extracted representations of the complexes varying the resolution inside the domain. The objective has been to obtain a cell complex, and the corresponding homology generators, with a maximum resolution only in a neighborhood of a specific homology class. This corresponds to computing the H_i generators on the base complex and, by traversing the $HHCC$, to performing only those refinements that create an i-



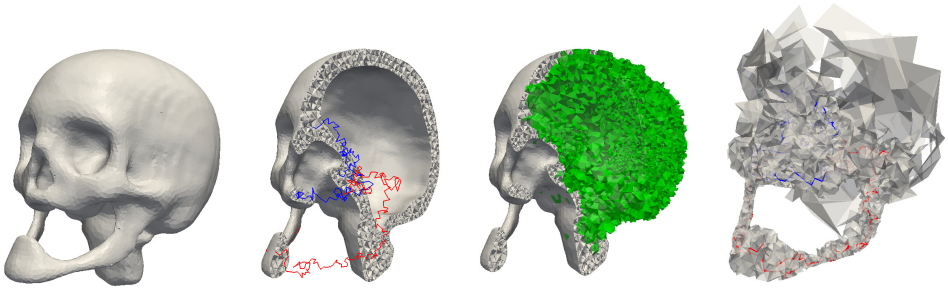
(a) (b)

Figure 4.11: The H_1 generators computed on the *Fertility* dataset (a) and on the *Hand* dataset (b) by fully refining the cell complex.



(a) (b)

Figure 4.12: The H_1 generators computed on the *Fertility* dataset and on the *Hand* dataset. In (a) and (b) the generators obtained by refining the cell complex only in a neighborhood of the generators.



(a) (b) (c) (d)

Figure 4.13: The H_1 and H_2 generators computed on the *Skull* dataset. In (a) the original dataset, in (b) and (c) the H_1 and H_2 generators computed at full resolution and in (d) the H_1 generators extracted at variable resolution and visualized inside the extracted cell complex.

cell belonging to some H_i generator (and the refinements on which they depend). This kind of selective refinement produces cell complexes with a low number of cells outside the area around the generators and thus leads to a further saving (15-30%) with respect to extracting generators and complexes at maximum resolution. Note that the extraction at variable resolution is a distinctive feature of the $HHCC$ which cannot be performed on other hierarchical models. Examples of variable resolution extractions are shown in Figure 4.12 and in Figure 4.13 (d).

4.4 Simplicial homology computation through a multi-resolution model

In this section, we define a multi-resolution model for simplicial complexes based on the set of simplification and refinement operators introduced in Subsection 4.1.2. Then, specializing the previous model, we describe a multi-resolution model for simplicial complexes based on the subset of the proposed modeling operators which preserve homology and we discuss about some possible implementations of this model. Finally, exploiting the properties of the homology-preserving multi-resolution model, we propose an algorithmic technique to efficiently extract homology and homology generators of a simplicial complex at different resolutions.

4.4.1 The Hierarchical Simplicial Complex (*HSC*)

In this subsection, we introduce a hierarchy of simplicial complexes, that we call *Hierarchical Simplicial Complex (HSC)*. An HSC is generated from a simplicial complex Σ, considered at full resolution, by iteratively applying the elementary simplification operators, (elementary) excisions and edge contractions, introduced in Subsection 4.1.2.

Analogously to the general case described in Subsection 4.2.2 an HSC is formally a triple $(\Sigma_B, \mathcal{M}, \mathcal{R})$. The first component, Σ_B, of the HSC is the coarse complex obtained at the end of the simplification step and it is called base complex. The second component is the set $\mathcal{M}$ of the refinement modifications which are the inverse operators with respect to the simplifications that have produced Σ_B from Σ. The third component $\mathcal{R}$ is the dependency relation between the modifications in the set $\mathcal{M}$ of all refinement modifications. As in the previous cases, we consider, for simplicity, the creation of the coarse complex Σ_B as a dummy refinement modification that we denote as μ_0.

Let us consider the modifying operators used to construct an HSC as a pair of graph subsets (H_1, H_2), where $H_i = (N_i, A_i)$ (see Subsection 4.2.1).
If μ is the excision $excision(\sigma)$,

- N_1 consists of the node corresponding to σ;
- A_1 consists of the arcs representing the relation of immediate boundary of σ;

- the sets N_2 and A_2 are both empty.

If μ is the edge contraction $contraction(u, v)$,

- N_1 consists of the nodes corresponding to the cofaces of u;
- A_1 is the set of arcs representing the relations of immediate boundary and coboundary of the nodes in N_1;
- N_2 consists of the nodes corresponding to the cofaces of u having at least a vertex v' such that $v' \notin \overline{\text{star}}\, v$ and in which vertex u has been replaced by v;
- A_2 is the set of arcs representing the relations of immediate boundary and coboundary of the nodes in N_2.

In order to define the dependency relation $\mathcal{R}$ between the refinement modifications in $\mathcal{M}$, it is enough to notice that if the refinement μ is

- the inclusion $inclusion(\sigma)$, the set C_μ consists of the simplices in the immediate boundary of σ;
- the expansion $expansion(\sigma, \tau)$, the set C_μ consists of the simplices in the immediate boundary of σ and of τ except for the simplex σ;
- the vertex split $split(u, v)$, the set C_μ consists of the simplices whose vertices are in $\overline{\text{star}}\, u$ and in which vertex u has been replaced by v.

Thanks to the results presented in Subsection 4.2.3, the introduced model HSC is well-defined and different selective refinements can be extracted from it. Similarly to the general model $MRCC$ and to the cellular model HCC, the performance of a sequence U of refinement modifications operated in a consistent order and closed with respect to $\mathcal{R}$ leads to the front complex Σ_U representing Σ at a certain resolution level. This allows to the HSC model to manage different refinement queries and to dynamically extract representations of the original cell complex at uniform and variable resolutions.

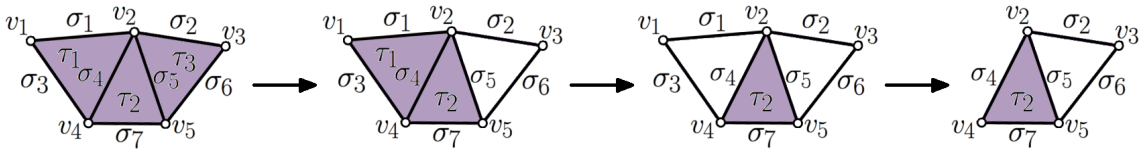


Figure 4.14: An example of simplification process consisting of two elementary excisions and an edge contraction.

An example of the HSC built from the simplification process illustrated in Figure 4.14 is illustrated in Figure 4.15. The refinement μ_0 creates the base complex Σ_B at the coarsest resolution. The refinement μ_1 is the elementary inclusion of the simplex τ_3, i.e., $\mu_1 = inclusion(\tau_3)$. For μ_1, the set $C_{\mu_1} = \{\sigma_2, \sigma_5, \sigma_6\}$. The refinement μ_2 is the vertex split $\mu_2 = split(v_1, v_4)$. For μ_2, the set $C_{\mu_2} = \{v_2, v_4, \sigma_4\}$. The refinement μ_3 is the elementary

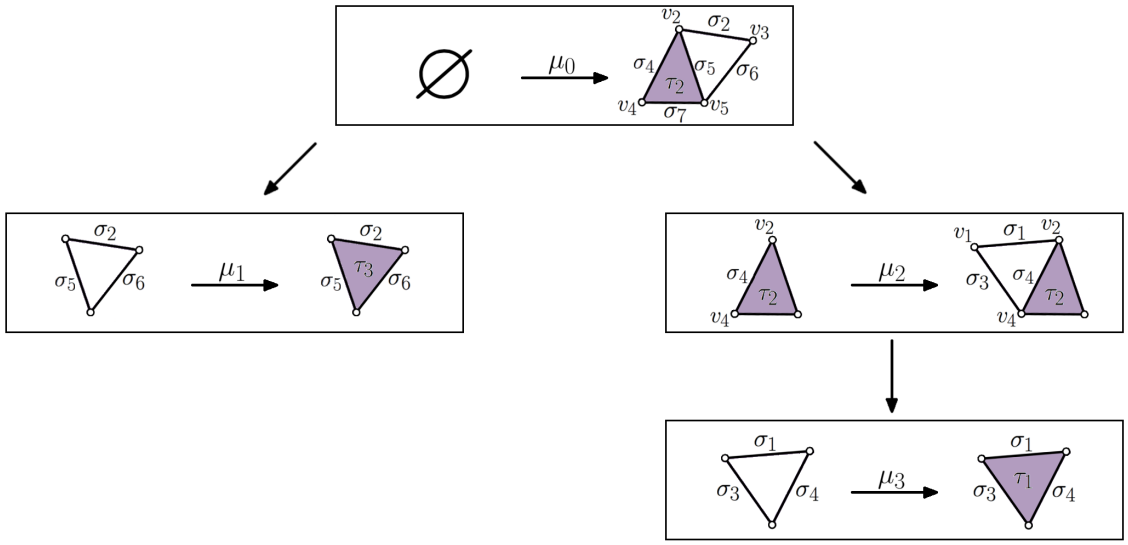


Figure 4.15: An example of HSC built from the simplification process illustrated in Figure 4.14.

inclusion of the simplex τ_1, i.e., $\mu_3 = inclusion(\tau_1)$. For μ_3, the set $C_{\mu_3} = \{\sigma_1, \sigma_3, \sigma_4\}$. The refinements μ_1 and μ_2 are independent. The refinement μ_3 depends on the refinement μ_2, since σ_1, σ_3 are in C_{μ_3}, and it does not depend on μ_1. Each closed subset of refinement modifications produces a different cell complex at intermediate resolution.

4.4.2 The Homology-preserving Hierarchical Simplicial Complex (*HHSC*)

Analogously to the cellular case, our interest is to use the HSC as a tool to efficiently retrieve homology of a simplicial complex at various levels of detail. In order to reach this purpose, we define an HSC based only on homology-preserving simplicial operator. This model, called *Homology-preserving Hierarchical Simplicial Complex (HHSC)*, can be built iteratively performing feasible homology-preserving simplifications on a simplicial complex Σ. Specifically, from now on, the used operators are just reductions and edge contractions satisfying the link condition.

An $HHSC$ can be represented as a Direct Acyclic Graph (DAG). This DAG can be iteratively built from the stack of the simplification operators performed on a simplicial complex Σ up to obtain a simplicial complex Σ_B called base complex. In each node of the DAG, the information required to correctly perform the correspondent refinement modification μ has to be stored. A simple but space-consuming way to do it is by encoding the two graph subsets H_1, H_2 completely characterizing μ. This solution is theoretically valid but it will probably lead to a non practically implementable multi-resolution model. Analogously to the cellular case (see Subsection 4.3.2), each node of the DAG could be more efficiently stored by using an implicit encoding able to represent the required information in a compact way. In order to further improve the spatial occupation of an $HHSC$, we propose here an encoding based on the IA^* data structure which has been

elected by experimental evaluations as one of the most compact data structures for simplicial complexes (see Section 2.1 and Subsection 3.4.2).

Analogously to the previous cases, the encoding of an $HHSC$ is achieved by representing it by the Direct Acyclic Graph of the dependency relations of the refinement modifications in $\mathcal{M}$. In this case, the root of the DAG, representing the dummy refinement modification μ_0, stores the base complex Σ_B by encoding it through the IA^* data structure. Other nodes of the DAG represent expansion and vertex split modifications and encode the refinements as well as their dependencies. By using the IA^* data structure, this information have to be expressed in terms of vertices and top simplices.

Let μ be a modification in $\mathcal{M}$ of the $HHSC$. Let us suppose that the operator μ^{-1}, performed during the simplification sequence generating the $HHSC$, has been applied to the simplicial complex Σ'' transforming it into the simplicial complex Σ'.
If $\mu = expansion(\sigma, \tau)$, with $\dim \sigma = k$, we store an ordered list of the vertices of τ in which the vertices of σ are stored in the first k positions.

If $\mu = split(u, v)$, we store:

- vertices u and v;
- list L_{uv} of the vertices in $\mathrm{link}_{\Sigma''} uv$;
- list L_{u^-} of the vertices in $\mathrm{link}_{\Sigma''} u \setminus \mathrm{link}_{\Sigma''} uv$;
- list T of the top simplices of Σ' whose vertices are in $L_{u^-} \cup L_{uv} \cup \{v\}$.

Similarly to the cellular case, each node μ encodes also a label describing if μ is a expansion or a vertex split modification, and arrays containing the pointers to the parents and the children of μ.
Analogously to previous cases, the DAG can be iteratively built from the information, saved in a stack, of a sequence of simplifications by adding one node at a time.

In order to check the validity of the proposed encoding, we have to prove that the information stored for a modification μ allows retrieving set C_μ and performing the operator μ.
If $\mu = expansion(\sigma, \tau)$, the stored list of vertices allows retrieving σ, τ and the simplices in their immediate boundary. The knowledge of these simplices allows immediately retrieving set C_μ and the information required to perform the operator μ.

If $\mu = split(u, v)$, the set C_μ (described in Subsection 4.4.1) consists of the faces of the simplices in T.
According to Subsection 4.1.2, the information required to perform μ is:

- for each coface σ of v, i.e., $\sigma = v\, v_1 \cdots v_k$, a list L_σ indicating which of the simplices (eventually both) in $\{u\, v_1 \cdots v_k,\ v\, v_1 \cdots v_k\}$will substitute σ;

- the simplices which will form the star of edge uv.

Let $\sigma = v\, v_1 \cdots v_k$ be a coface of v. The list L_σ can be retrieved as follows:

- if $v_i \in L_{uv}$ for each i, then $L_\sigma = \{u\, v_1 \cdots v_k,\ v\, v_1 \cdots v_k\}$;
- if there exists at least one vertex $v_i \in L_{u^-}$, then $L_\sigma = \{u\, v_1 \cdots v_k\}$;
- otherwise, $L_\sigma = \{v\, v_1 \cdots v_k\}$.

The star of edge uv consists of the simplices $u\, v\, v_1 \cdots v_k$ where $v_1 \cdots v_k$ is a simplex of the simplicial complex on which μ is applied and each of its vertices v_i belongs to L_{uv}.

4.4.3 Homology computation through an *HHSC*

In this subsection, we present an approach for computing simplicial homology and homology generators at various resolutions using the version of the hierarchical model based only on the homology-preserving operators.
Given a simplicial complex Σ and an $HHSC$ of Σ, since any front complex Σ_U is obtained from the base complex Σ_B by applying a sequence of homology-preserving refinement modifications, the homology of all the complexes implicitly encoded in the $HHSC$ is the same. We use the *Smith Normal Form (SNF) reduction algorithm* [Mun84] to compute homology and homology generators on the base complex Σ_B. In order to retrieve the homology generators of a front complex Σ_U, it is enough to understand how the refinement modifications in U affect the homology generators of Σ_B.
Without loss of generality, we can just describe how the homology generators change after a single refinement modification. To do it, we need to consider refinement and simplification modifications in terms of some elementary homology-preserving operators introduced in [KMS98].

Let Γ be a cell complex and let σ, τ be two cells in Γ of dimension k, $k+1$, respectively, such that $\langle \partial\tau, \sigma \rangle$ is invertible, where $\langle \quad , \quad \rangle$ represents the scalar product between chains in $C_*(\Gamma)$. We define as *elementary collapse* [1] the homology-preserving operator that, starting from $C_*(\Gamma)$, builds a new chain complex D_* by removing the two cells σ and τ from $C_*(\Gamma)$ and modifying the boundary $\partial\nu$ of a cell ν into $\tilde{\partial}\nu$ as follows.

$$\tilde{\partial}\nu = \begin{cases} \partial\nu - \frac{\langle \partial\nu, \sigma \rangle}{\langle \partial\tau, \sigma \rangle} \partial\tau & \text{if } \dim(\nu) = k+1 \\ \partial\nu - \langle \partial\nu, \tau \rangle \tau & \text{if } \dim(\nu) = k+2 \\ \partial\nu & \text{otherwise} \end{cases}$$

Elementary reductions and edge contractions satisfying the link condition can be seen in terms of elementary collapses: $reduction(\sigma, \tau)$ is just a specific instance of an elementary collapse and $contraction(u, v)$, if the link condition holds, corresponds to

[1] In [KMS98], elementary collapses are called reductions. We do not use such a term in order to avoid confusions.

the performance of the elementary collapses removing the pairs of cells $(u\sigma, uv\sigma)$ with $\sigma \in \operatorname{link} u \cap \operatorname{link} v$ and, at last, the pair (u, uv).
Let us denote the chain complex $C_*(\Gamma)$ as C_*. If we know the homology generators of chain complex D_* obtained after an elementary collapse of pair (σ, τ), we are interested in retrieving the homology generators of the original chain complex C_*. To reach this purpose, we need to explicit formulas for the chain maps between C_* and D_* inducing an isomorphism in homology.
Let $\psi_*^{(\sigma,\tau)} : C_* \to D_*$, $\iota_*^{(\sigma,\tau)} : D_* \to C_*$ be two chain maps defined by

$$\psi^{(\sigma,\tau)}(c) = \begin{cases} c - \frac{\langle c,\sigma\rangle}{\langle \partial\tau,\sigma\rangle}\partial\tau & \text{if } \dim c = k \\ c - \langle c,\tau\rangle\tau & \text{if } \dim c = k+1 \\ c & \text{otherwise} \end{cases}$$

$$\iota^{(\sigma,\tau)}(c) = \begin{cases} c - \frac{\langle \partial c,\sigma\rangle}{\langle \partial\tau,\sigma\rangle}\tau & \text{if } \dim c = k+1 \\ c & \text{otherwise} \end{cases}$$

The chain maps $\psi_*^{(\sigma,\tau)}$ and $\iota_*^{(\sigma,\tau)}$ establish a homology equivalence between the chain complexes C_* and D_* (see Theorem 2 in [KMS98]). Furthermore, since $\psi_*^{(\sigma,\tau)} \cdot \iota_*^{(\sigma,\tau)} = \mathrm{id}_{D_*}$, the homology generators of C_* can be retrieved by computing the generators of D_* and applying to them the map $\iota_*^{(\sigma,\tau)}$.

The simplicial operators used to build the $HHSC$ can be consider as compositions of specific instances of elementary collapses. The above rules for retrieving the homology generators of a refined complex knowing the generators of a coarse one can be used to suitably modify the homology generators after a refinement modification μ and to correctly expand the generators of the base complex Σ_B up to any front complex Σ_U encoded in the $HHSC$.
According to the formula above, the performance of $reduction(\sigma, \tau)$, or of its inverse $expansion(\sigma, \tau)$, does not affect the homology generators of a simplicial complex.
Let Σ' be a simplicial complex, μ be the operator $split(u, v)$ feasible on Σ' and Σ'' be the simplicial complex obtained performing μ on Σ'. Suppose to know the set $\mathcal{G}$ of the (geometric realizations of the) homology generators of Σ', in which each generator is expressed as the set of the simplices of which it consists. Algorithm 6 returns the set of homology generators of Σ'' by updating the generators in $\mathcal{G}$ according to the above described formula.

Algorithm 6 makes use of a bitvector B, which describes if a generator g in $\mathcal{G}$ has been updated in first step of the algorithm. Algorithm 6 uses the following functions:

- $toBeModified(u, v)$, which returns the set Σ'_u of the simplices of Σ' that will be modified by μ;
- $getGeneratorsOn(\sigma, \mathcal{G})$, which returns the set $\mathcal{G}_\sigma$ of the generators containing the simplex σ;

Algorithm 6 $ExpandGenerators(u, v, \mathcal{G})$

1: INPUT: u, v, vertices
2: INPUT: $\mathcal{G}$, set of generators
3: OUTPUT: $\mathcal{G}$, set of updated generators
4: *// B is a bitvector of length $|\mathcal{G}|$*
5: $B :=$ null vector
6: *// Σ'_u is the set of simplices that will be modified by $split(u, v)$*
7: $\Sigma'_u := toBeModified(u, v)$
8: *// Update the generators*
9: **for all** simplices σ of Σ'_u **do**
10: *// $\mathcal{G}_\sigma$ is the set of generators containing σ*
11: $\mathcal{G}_\sigma := getGeneratorsOn(\sigma, \mathcal{G})$
12: *// Update the generator containing σ*
13: **for all** generators g in $\mathcal{G}_\sigma$ **do**
14: $\sigma' :=$ simplex σ in which v is replaced by u
15: $replaceInGenerator(g, \sigma, \sigma', \mathcal{G})$
16: $B[g] := 1$
17: *// Expand the generators if necessary*
18: **for all** generators g in $\mathcal{G}$ **do**
19: **if** $B[g] = 1$ **then**
20: *// Bd is the set of simplices on the boundary of g*
21: $Bd := getBoundary(g)$
22: **for all** simplices ρ in Bd **do**
23: **if** $v \in \rho$ **then**
24: $addToGenerator(g, u\rho, \mathcal{G})$

- $replaceInGenerator(g, \sigma, \sigma', \mathcal{G})$, which updates the generator g in $\mathcal{G}$ by replacing in it the simplex σ with the simplex σ';

- $getBoundary(g)$, which returns the set Bd of the simplices ρ of Σ'' such that, for the generator g, $\langle \partial_{\Sigma''} g, \rho \rangle \neq 0$;

- $addToGenerator(g, \sigma, \mathcal{G})$, which updates the generator g in $\mathcal{G}$ by adding to it the simplex σ.

Algorithm 6 considers the simplices of Σ' that will be modified by the performance of μ ($toBeModified(u, v)$). For each of these simplices, the algorithm retrieves the generators containing it ($getGeneratorsOn(\sigma, \mathcal{G})$) and suitably modifies them ($replaceInGenerator(g, \sigma, \sigma', \mathcal{G})$). For each generator g updated in the first step of the algorithm, the boundary of g is computed ($getBoundary(g)$) in order to check if g is still a cycle in Σ''. If not, g is updated by suitably adding some of the simplices newly created by μ ($addToGenerator(g, \sigma, \mathcal{G})$).

4.5 Concluding remarks

In this chapter, we have combined the compactness and the expressive power of multi-resolution models to quickly retrieve the homological information of cell and simplicial complexes. First, various atomic modifying operators have been investigated. This study has involved both homology-preserving and homology-modifying operators and it has led to the definition of two complete bases of operators in terms of which any cellular and, respectively, simplicial modification can be represented. Then, we have proposed the definition of a general multi-resolution model and specialized it for cellular and simplicial cases. For cellular complexes, a homology-preserving hierarchical cell complex has been implemented and used to efficiently extract the homology generators at any level of detail. Experimental evaluations have been performed revealing the efficiency and the compactness of the proposed approach. Since the promising results obtained, a hierarchical model dealing with the homology at different resolutions of a simplicial complex has been proposed.

Further developments of our work will mainly involve the simplicial case. In particular, we plan to design and develop an efficient and compact implementation of the homology-preserving multi-resolution model for simplicial complexes based on reductions and homology-preserving edge contractions introduced in Section 4.4. In order to obtain a model with a compact storage cost, we plan to work with a data structure encoding only the top simplices and the vertices of a simplicial complex Σ, such as the Generalized Indexed data structure with Adjacencies (IA^*) [CDW11] and the Stellar Tree [Fel15]. The difficulty here is to have an efficient tool to perform homology-preserving edge contractions to generate the sequence of simplifications which are the first step in generating the multi-resolution model.

When implementing a homology-preserving edge contraction, we have to check the link condition and update the data structure. The former is the most challenging task when working on a data structure based on top simplices: it requires the extraction of the links of the edge involved in the simplification and its two vertices. Thus, we have to represent all the simplices, and this is an operation than becomes extremely costly when working in high dimensions. On the other hand, updating the data structure has a contained complexity. We need to remove the top simplices incident in the edge removed (operation performed efficiently by both IA^* data structure the Stellar Tree) and we need to introduce the top simplices possibly created (always less or equal to the number of top simplices eliminated). The key point of this step is that it only involves top simplices and, thus, it is well suited for such data structures.

Similarly to the homology-preserving multi-resolution model for a cell complex defined in Section 4.3, we are confident that the $HHSC$ could actually lead to interesting results. First at all, we want to perform experimental evaluations in order to compare the storage cost of the encoding of an $HHSC$ with the storage cost required to encode the corresponding simplicial complex at full resolution. Second, we anticipate that the time required to compute the homology of the base complex and to fully refine the complex and its generators could be much better with respect to the time needed for executing the SNF reduction on the original simplicial complex Σ. Moreover, we believe that the

computation of homology generators of the original simplicial complex Σ can be further improved by expanding them not up to the full resolution simplicial complex but just reaching the simplicial complex in the $HHSC$ having maximum resolution only in the vicinity of the homology generators.

Chapter 5

Topologically-consistent Simplification of Discrete Morse Complexes

A fundamental issue when working with real data is the presence of noise. Using Morse theory as a tool for studying these data, the resulting oversegmentations produced require a device to eliminate uninteresting features. Multi-resolution models are then defined to provide domain experts with an interactive tool for the exploration of such data. At the base of the definition of a multi-resolution model stands the definition of a simplification algorithm, used for building the model.

As mentioned in Subsection 2.4.4, in Morse theory the morphological simplification of a dataset is driven by an operator called *cancellation*. Cancellation removes two critical points connected by a unique separatrix line through local modifications of the integral lines originating and converging in the two points [Mat02]. Two different approaches have been defined for applying such operator on real data. The first approach is based on modifications of the *Morse Incidence Graph* (MIG), i.e., a graph representation of the connectivity of the critical points [EHZ01]. The geometry of the Morse complexes is explicitly stored in the representation, attached to the graph nodes. For this reason, this approach is also known as *explicit*. Removing two critical points corresponds to deleting two nodes of the graph and merging the attached entities.

The second approach is based on discrete Morse theory [For98] (see Subsection 1.3.3). The Forman gradient is used here as a discrete counterpart of the gradient of a smooth function f. In this context, the integral lines and the critical points of f are represented as gradient paths and critical simplices, respectively. From a gradient vector field, the Morse cells can be computed navigating the gradient paths and, thus, they do not need to be stored explicitly.
Alongside with the notion of critical point and Morse complex also the *cancellation* operator has been defined in this combinatorial framework. Applying the $cancellation$ operator on a Forman gradient corresponds to eliminating a pair of critical simplices connected by a unique separatrix V-path and changing the direction of the gradient arrows along the path between them. This update implicitly modifies the Morse cells accordingly. Thus, this approach is also know as *implicit*.

Simplifications performed by using the *explicit* method are generally faster thanks to the graph-based representation, and thus preferable when high performances are required. On the other hand, the *implicit* method avoids the extraction of the Morse cells and is preferable when compactness is more relevant. However, even if the two methods are equivalent in the two dimensional case, the implicit representation may present inconsistencies when working in higher dimensions. The origin of the problem, described in [GRSW13] for 3D scalar fields, is ascribed to the structure of the discrete gradient pairs along the paths connecting 1-saddles to 2-saddles. This makes the *implicit* approach useless in practice when simplifying volumetric data.

Different multi-resolution models have been defined in the literature based on an *explicit* simplification sequence [GKK$^+$12, ČDI12]. The resulting models have been proven to be efficient for interactively modifying and visualizing Morse cells but they are lacking in compactness. In this direction, the Forman gradient would be a perfect candidate for defining a compact multi-resolution model but, due to the inconsistency problem described in [GRSW13], no such models have been yet defined for volumetric data.

We consider here the problem of defining a simplification algorithm for the implicit method based on an efficient graph-based representation and free from the topologically inconsistencies that affect the standard implicit method. Our approach is described and implemented for 3-dimensional simplicial complexes, but it is entirely dimension-independent. Thus, the major contributions are:

1. the definition of a compact data structure for the efficient simplification of a Forman gradient;
2. a method for removing gradient paths causing topological inconsistencies;
3. an algorithm combining the two previous contributions to perform a topologically-consistent simplification of a discrete Morse complex.

Specifically, in Section 5.1, we consider the problem of representing a discrete Morse complex by presenting the two representations used in the literature and by introducing a new efficient and compact data structure. Section 5.2 describes the $cancellation$ operator in terms of gradient-based and graph-based representations and points out the inconsistencies between the two approaches. Section 5.3 is devoted to solve this problem. The goal is reached thanks to a disambiguation algorithm able to clean up the gradient from paths causing inconsistencies and thanks to the use of a simplification operator, called $remove$, whose performance does not introduce any bad configuration in the gradient. In Section 5.4, the proposed data structure and the disambiguation algorithm are combined together in order to obtain a topologically-consistent simplification algorithm. Furthermore, experimental results obtained from an implementation of the just introduced algorithm are presented and discussed. Finally, in Section 5.5, we summarize our contributions and we focus on some immediate improvements in persistent homology computation and in expressiveness of a multi-resolution model that the proposed simplification process produces. Moreover, we discuss about the development of a geometric simplification algorithm and about the definition of a multi-resolution model combing morphological and

geometrical simplifying operators.

The contributions of this Chapter has been collected in [IFD15]. This paper has been presented at the conference Shape Modeling International (SMI) 2015 and awarded with an honorable mention.

5.1 Representing discrete Morse complexes

Two kinds of representation are used for Morse complexes: a graph-based representation [ELZ02, BEHP04, GKK+12], which *explicitly* encodes the cells of the Morse complexes and their topological relations, and one based on the encoding of the Forman gradient, which represents such relations *implicitly* [GRWH12, GRSW13]. In this section, we motivate why the implicit representation is preferable when aiming at a compact data structure for simplifying Morse complexes. Moreover, we propose a new compact graph-based representation coupling the efficiency of the explicit graph-based representation and the compactness of gradient-based one.

Gradient-based representation
The standard gradient-based representation encodes the arrows defining a Forman gradient field V on a cell complex Γ. Since a Forman gradient field is a collection of pairs of cells on Γ, we need a representation for Γ, in which all cells and their mutual incidence relations are explicitly encoded, as in the *Incidence Graph (IG)* [Ede87] (see Section 2.1). The Forman gradient V can be implemented in a straightforward way on an IG as a Boolean function associated with the arcs of the IG. For a regular grid, the arcs of the IG are encoded implicitly by indexing the cells of the complex. Moreover, since V defines a pairing between incident cells, V is encoded as a bitvector based on the same indexing [GRWH12].

For simplicial complexes, compact representations such as the IA^* data structure [CDW11] and the Stellar Tree [Fel15] are characterized by an explicit encoding of the vertices and the top simplices. In such data structures, immediate boundary and coboundary relations are not explicitly stored. So, a different encoding of the Forman gradient is required. As discussed in Subsection 3.2.1, a dimension-independent encoding for the Forman gradient, which associates the gradient pairs with the top simplices, has been defined [FID14]. It represents a generalization to arbitrary dimensions of the ones proposed for 2- and 3-dimensional simplicial complexes [FlDFW14, WIFD13]. This encoding associates with each top simplex σ of a simplicial complex Σ a bitvector representing all the possible pairings on the boundary of σ. All the gradient pairs inside each top simplex can be represented in a very compact way, e.g., one byte for triangle for a simplicial 2-complex or two bytes for tetrahedron for a simplicial 3-complex.

Graph-based representation
The graph representation is the so-called *Morse Incidence Graph (MIG)*. As described

in Subsection 2.4.3.2, an MIG is a weighted graph $G = (N, E, \mu)$ in which:

- the set of nodes N is partitioned into $d + 1$ subsets $N_0, N_1, \ldots, N_d$;
- each node in N_k, denoted as the set of k-*nodes*, represents both a k-cells of the descending complex Σ_D and a $(d - k)$-cells of the ascending complex Σ_A;
- each arc in E, connecting a k-node σ to a $(k + 1)$-node τ, represents the incidence relation between the Morse cells corresponding to σ and τ;
- the label $\mu(\tau, \sigma)$ of an arc (τ, σ) coincides with the multiplicity of the incidences between k-cell σ and $(k + 1)$-cell τ.

The MIG is an incidence-based representation of the two Morse complexes and provides also a combinatorial representation of the 1-skeleton of the Morse-Smale complex. In the applications, attributes are attached to the nodes in N storing the geometric information associated with the Morse cells. In Figure 5.1, an example of a 2D MIG is shown, representing the combinatorial structure of the 1-skeleton of the MS complex depicted in Figure 5.2(b).

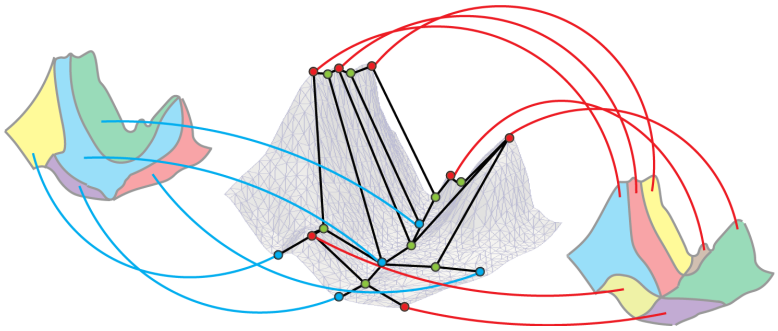


Figure 5.1: The MIG computed on the terrain dataset shown in Figure 5.2(b). The nodes of the graph are the maxima (red nodes), saddles (green nodes) and minima (blue nodes) of the scalar field function. Arcs (black lines) connect two nodes if there exist a separatrix line connecting the corresponding critical points. Nodes corresponding to maxima are enhanced with the geometrical representation of the corresponding descending 2-cells (relation depicted with red lines) while minima nodes refer to the ascending 2-cells (relation depicted with blue lines).

In [GKK+12], an *extended* MIG has been defined storing the cells of both the ascending and descending Morse complexes explicitly. Given a d-dimensional simplicial complex Σ endowed with a gradient vector field V, let us consider the subset of the simplices in Σ belonging to the cells of either the ascending Morse complex Σ_A or the descending one Σ_D. Let σ be such a simplex of dimension k. The extended MIG associates with σ a label indicating the $(d - k)$-cells of Σ_A and the k-cells in Σ_D which σ belongs to. If Σ is represented by a data structure explicitly encoding all its simplices, this labeling is easy to be stored.
A different strategy has to be developed for compact data structures encoding only the

vertices and top simplices of Σ, such as the *Generalized Indexed data structure with Adjacencies (IA^*)* [CDW11] (see Section 2.1). In this framework, in order to associate with a simplex a label indicating the corresponding cells in the ascending and descending Morse complexes, a compact representation of any simplex σ of Σ is required. This can be achieved by representing the simplex σ as a pair (t, bv) where t is the index of a top simplex τ containing σ and bv is a bitvector of length $\dim \tau + 1$ representing the vertices of τ spanning σ.
Independently from the chosen data structure, the extended MIG can be used for fast rendering of Morse cells during a simplification or a refinement process [GKK+12]. However, computing and storing all the Morse cells during simplification leads to inefficiencies in terms of storage.

Discrete Morse Incidence Graph
To overcome the lack of compactness affecting the graph-based representations proposed in the literature, we have defined a *Discrete Morse Incidence Graph* ($DMIG$) combining the MIG with the compact representation provided by the Forman gradient V. This latter differs from the extended MIG by the geometric attributes attached to the graph nodes. Using the compact representation provided by V, we associate with each node n in N_k the corresponding critical k-simplex σ in V instead of the entire geometrical embedding for the Morse cell corresponding to σ. If we consider the case of 3-dimensional simplicial complexes encoded in a compact data structure, like the IA^* data structure, implicitly representing the geometrical embedding with the critical simplices requires only one integer for each critical maximum or minimum, corresponding to a tetrahedron and a vertex respectively, and two integers for each 1- and 2-saddle (corresponding to triangles and edges, respectively, where triangles are represented as pairs of tetrahedra and edges as pairs of vertices).

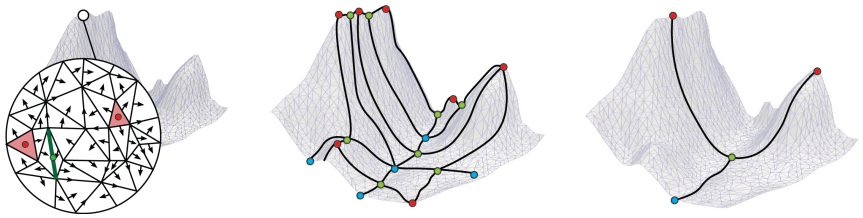
Dataset	$\lvert\Sigma_0\rvert$	$\lvert\Sigma_3\rvert$	$\#C$	$\frac{MIG}{DMIG}$	$\frac{DMIG}{Gradient.}$
SHOCKWAVE	2M	12M	3.2K	6.9x	1.001x
BONSAI	4.2M	24.4M	0.8M	27.6x	1.17x
VISMALE	4.6M	26.5M	1.2M	28.6x	1.12x
FOOT	5.0M	29.5M	1.98M	30.1x	1.24x

Table 5.1: Evaluation on 3-dimensional simplicial complexes of the storage costs using the $DMIG$ compared to the extended MIG and the Forman gradient. For each dataset, we indicate the number of vertices and tetrahedra (columns Σ_0 and Σ_3), the number of critical simplices ($\#C$) and the compression factor of the $DMIG$ with respect to the MIG and the Forman gradient.

In Table 5.1, we compare, for 3-dimensional simplicial complexes, the storage cost of the $DMIG$ with respect to the storage cost of the extended MIG, as described above, and versus the space required by just encoding the Forman gradient. The $DMIG$ results to be 7 to 30 times more compact than the extended MIG and its size is always comparable with that of the direct encoding of the Forman gradient.

5.2 Simplifying discrete Morse complexes

Morphological simplification of scalar fields [ELZ02, GNP+05] is a powerful tool known in the literature for removing insignificant features while preserving relevant parts of the data (see Figure 5.2). For a detailed discussion please refer to [Iur14].



(a) (b) (c)

Figure 5.2: A Forman gradient defined on a simplicial complex (a) and the 1-skeleton of its Morse-Smale complex (b). Effects of topological simplification performed on the 1-skeleton of the Morse-Smale complex (c). Note that function values (height values of the terrain) are not modified by the topological simplification; the simplified 1-skeleton represents the two main peaks and the pit only.

As mentioned in Subsection 2.4.4, an operator (called *cancellation*) has been defined in the literature for removing pairs of critical points [Mat02]. The discrete counterpart of this operator has been introduced in [For98] and allows for the elimination of a pair of critical simplices. The effects of $cancellation$ operator can be described in terms of gradient-based or graph-based representations. In this section, we describe this operators for both the implicit and explicit representations indicating where the two approaches present inconsistencies.

Gradient-based *cancellation*

Let Σ be a simplicial complex endowed with a Forman gradient V. Given two critical simplices τ and σ of Σ, with dimension $k+1$ and k respectively, (σ, τ) is a valid $cancellation$ pair for (Σ, V) if $\mu(\tau, \sigma) = 1$ i.e., if the two simplices are connected through a unique separatrix V-path. Under such assumption, $k\text{-}cancellation(\sigma, \tau)$ is the operator which removes the critical simplices σ and τ, reversing the gradient arrows along the unique separatrix V-path from σ to τ. More precisely, if $[\tau, (\sigma_1, \tau_1), (\sigma_2, \tau_2), \ldots, (\sigma_r, \tau_r), \sigma]$ is a separatrix V-path, a new V-path on Σ is created as $[(\sigma, \tau_r), (\sigma_r, \tau_{r-1}), \ldots, (\sigma_2, \tau_1), (\sigma_1, \tau)]$. The Forman gradient V' obtained in this way is still a Forman gradient on Σ with the same critical simplices with the exception of σ and τ.

Figure 5.3 shows the effect of $1\text{-}cancellation(\sigma, \tau)$ on a Forman gradient V defined on a 2-dimensional simplicial complex Σ: σ is a critical edge and τ and τ' are two critical triangles. Starting from σ, the separatrix V-path, connecting τ to σ, is reversed. As a consequence, τ and σ are not critical. The two separatrix V-paths, connecting τ to α_1 and α_2, are extended with the reversed V-path, and now connect τ' to α_1 and α_2. The two separatrix V-paths starting from σ and reaching ρ_1 and ρ_2 become non-separatrix V-paths.

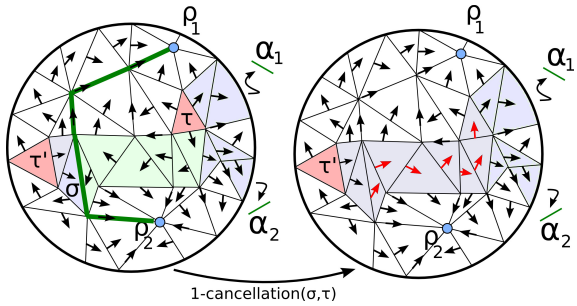


Figure 5.3: Effect of the 1-$cancellation(\sigma, \tau)$ on a Forman gradient V defined on a 2-dimensional simplicial complex, The original V (left side) has two critical triangles τ and τ' (in red) and one critical edge σ (in green). Red arrows indicate the V-path involved in the simplification.

Graph-based *cancellation*

As for the Forman gradient, also the Morse Incidence Graph (MIG) can be simplified by means of the *cancellation* operator. We consider an MIG $G = (N, E, \mu)$, and a pair of nodes τ and σ in N of dimension $k+1$ and k, respectively, connected through an arc in E. We denote as $\mathrm{A} = \{\alpha_i, i = 1, \ldots, i_{max}\}$ the k-nodes of the MIG different from σ and connected to node τ, and as $\mathrm{B} = \{\beta_j, j = 1, \ldots, j_{max}\}$ the $(k+1)$-nodes of the MIG different from τ and connected to the node σ.

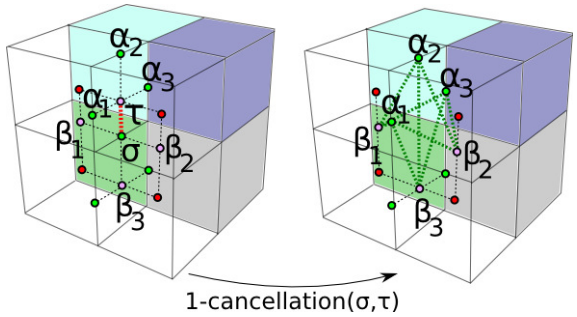


Figure 5.4: Example of a 1-$cancellation(\sigma, \tau)$ operator. Red dots correspond to maxima, purple dots to 2-saddles, green dots to 1-saddles. Dotted lines corresponds to the arcs of the MIG.

A $cancellation$ pair (σ, τ) is feasible on an MIG G if $\mu(\tau, \sigma) = 1$. Its effect is as follows (see Figure 5.4):

- delete nodes τ and σ,
- delete all arcs incident in either node τ or node σ,
- introduce an arc (β_j, α_i) for each $\alpha_i \in \mathrm{A}$ and each $\beta_j \in \mathrm{B}$ (if such arc does not already exist),
- set $\mu(\beta_j, \alpha_i) = \mu(\beta_j, \sigma)\mu(\tau, \alpha_i) + \mu(\beta_j, \alpha_i)$.

Topological inconsistencies
As investigated in [GRSW13], the simplification of the same pair of critical simplices performed on an MIG and on the corresponding Forman gradient may give different results on the connectivity of the critical simplices when working in three dimensions or higher.

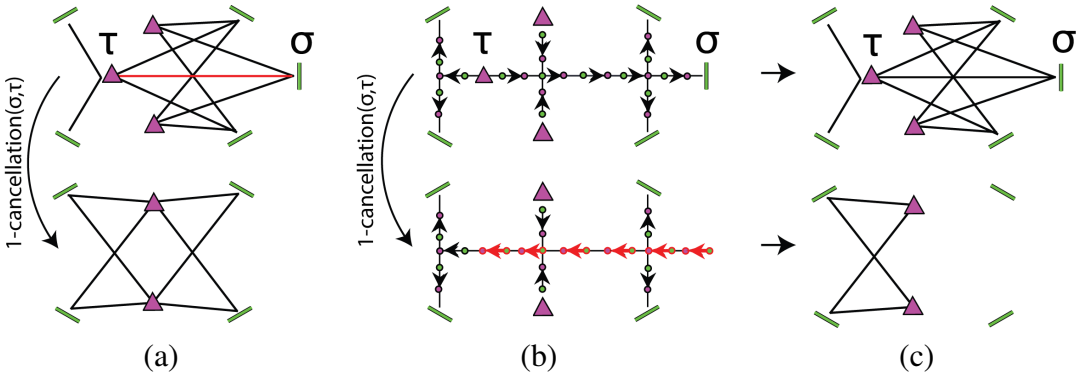


Figure 5.5: Morse Incidence Graph (a) and Forman gradient (b) before and after the 1-$cancellation(\sigma, \tau)$ operator and (c) MIG computed from the Forman gradient. Green edges denote 1-saddles and purple triangles denote 2-saddles. In (b), simplices forming the V-paths are depicted with green (edges) and purple (triangles) dots. Arrows between two dots indicate a gradient pair, while a straight line between two dots indicates the incidence relation between the corresponding simplices.

We illustrate this problem by using the example in Figure 5.5. Recall that the weighted arcs in the MIG are in correspondence with the separatrix V-paths in the Forman gradient. Figure 5.5(a) shows a $cancellation$ applied to delete 1-saddle σ and 2-saddle τ on the MIG. As a result of the $cancellation$, all the arcs connected to either σ or τ are deleted, and the new arcs introduced connect nodes which were previously connected with σ and τ. In Figure 5.5(b), the same configuration is depicted on a Forman gradient showing the separatrix V-paths between the critical simplices connected to σ and τ. When performing the same $cancellation$ as before, the arrows in the separatrix V-path between σ and τ are swapped. As a consequence, following the gradient arrows outgoing from the remaining 2-saddles (purple triangles), the new separatrix V-paths will end at the two 1-saddles (green edges), on the left, only. The MIG configuration extracted from the original Forman gradient and the one simplified are shown in Figure 5.5(c).

We can observe that this situation occurs each time a $cancellation$ involves a separatrix V-path originating from different critical simplices and converging to different critical simplices, which merge and split in a common V-path, that we call a *shared V-path*. More precisely, a V-path π is called a *shared V- path* if it is contained in at least two separatrix V-paths π', between τ' and σ', and π'', between τ'' and σ'', such that $\tau' \neq \tau''$ and $\sigma' \neq \sigma''$.

5.3 Solving topological inconsistencies

The aim of this section is to provide a solution to the inconsistency problem caused by $cancellation$ operators performed on different representations of a discrete Morse complex.
Our solution is based on a preprocessing step, able to free the gradient vector field from shared V-paths, and on a new simplification operator, called $remove$, preventing the generation of gradient configurations leading to inconsistencies.

5.3.1 Shared *V*-paths and the *remove* operator

A $Cancellation$ produces the removal of two Morse cells, both in the ascending and descending Morse complexes, as well as the local modifications of the incidence relations between the remaining Morse cells. In spite of this, a $cancellation$ may increase the incidence relations among such cells when applied on a complex in dimension higher than two.
Let us considered, for instance, the 1-$cancellation(\sigma, \tau)$ of a 1-saddle and a 2-saddle operated on a 3-dimensional Morse complex. As described in the previous section, the effect of 1-$cancellation(\sigma, \tau)$ on a graph-based representation consists of deleting nodes σ and τ, as well as all the arcs incident in nodes σ and τ, and adding one arc for each pair (β_j, α_i) where α belongs to A and β_j belongs to B. Thus, the 1-$cancellation$ operator deletes two nodes from N, but possibly increasing the number of arcs connecting 1-nodes to 2-nodes in the graph by deleting $|\,\mathrm{A}\,| + |\,\mathrm{B}\,| + 1$ of such arcs, but adding $|\,\mathrm{A}\,| \cdot |\,\mathrm{B}\,|$ of them. Thus, it is not a simplification operator, since it does not reduce the size of the graph. In [GKK$^+$12], this issue has been discussed at length, since it can cause computational problems and, more importantly, make the application of k-$cancellation$ operator unfeasible on large-scale data sets. Several strategies are proposed in [GKK$^+$12], which aim at postponing an k-$cancellation$ that would introduce a number of arcs greater than a predefined threshold, or vertices with valence greater than a predefined threshold.
Another solution proposed to overcome this issue make use of the *remove* operators. A $remove$ modification, denoted as k-$remove(\sigma, \tau)$, is a dimension-independent simplification operator firstly introduced in [ČomićD11]. Unlike $cancellation$, the $remove$ operator always reduces the size of a graph-based representation of the Morse complex.
For $k = 0$ or $k = d - 1$, k-$remove(\sigma, \tau)$ is equivalent to k-$cancellation(\sigma, \tau)$. When $1 < k < d - 1$, k-$remove(\sigma, \tau)$ operator can be consider as a $cancellation$ with stronger feasibility conditions.

A k-$remove(\sigma, \tau)$ collapses a k-saddle σ and a $(k+1)$-saddle τ, that are connected through a unique separatrix V-path, if there is at most one k-saddle, different from σ, connected with τ or, at most one $(k+1)$-saddle, different from τ, connected with σ. Because of the similarity between a k-$remove(\sigma, \tau)$ and a k-$cancellation(\sigma, \tau)$, we can describe the effects of a k-$remove(\sigma, \tau)$ on the MIG as a k-$cancellation(\sigma, \tau)$ in which $\#\,\mathrm{A} \leq 1$ or $\#\,\mathrm{B} \leq 1$ (see Section 5.2). Its effect in terms of updates on the Forman gradient or on the MIG are the same as the $cancellation$ operator.

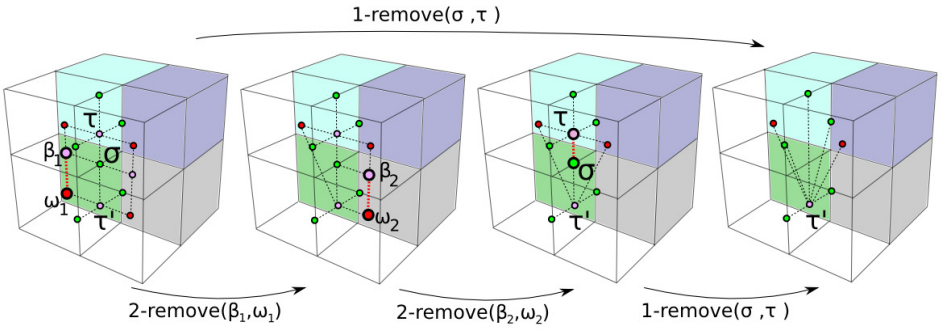


Figure 5.6: Examples of a 1-$remove(\sigma, \tau)$ operator. Red points correspond to maxima, purple points to 2-saddles, green points to 1-saddles. Dotted lines corresponds to the arcs of the MIG.

When the feasibility conditions are not satisfied, i.e., when $\#\,\mathrm{A} > 1$ and $\#\,\mathrm{B} > 1$, a suitable sequence of extremum-saddle operators is performed to obtain a valid configuration for k-$remove(\sigma, \tau)$. Such sequence of simplifications forms a *macro-operator*. As an example, we consider the macro-operator which collapses a 2-saddle τ and a 1-saddle σ into another 2-saddle τ' (see Figure 5.6). For all the 2-saddles β_j connected to σ and different from τ and τ', a 2-$remove$ involving β_j is performed. When τ and τ' are the only 2-saddles connected to σ, the 1-$remove(\sigma, \tau)$ is performed.

$remove$ operator is still affected by the problems of inconsistencies arising when performing the graph-based or the gradient-based simplification. However, it guarantees a fundamental property that makes k-$remove(\sigma, \tau)$ the first ingredient for our simplification algorithm: a k-$remove(\sigma, \tau)$ never introduces shared V-paths in V.

Proposition 5.1. *Let Σ be a simplicial complex endowed with a Forman gradient V, which does not contain any shared V-path. Let (σ, τ) be a valid $cancellation$ pair for (Σ, V), let V' be the Forman gradient obtained from V by applying k-$cancellation(\sigma, \tau)$. Then, V' does not contain any shared V-path if and only if k-$cancellation(\sigma, \tau)$ is a feasible k-$remove(\sigma, \tau)$ for (Σ, V).*

Proof. " $\Leftarrow$ ". Let us assume that k-$remove(\sigma, \tau)$ is feasible for (Σ, V). By hypothesis, V has no shared V-path. Thus, any shared V-path in V' should be contained in one of the separatrix V-paths newly created by k-$remove(\sigma, \tau)$. Since k-$remove(\sigma, \tau)$ is feasible for (Σ, V), at least one of the sets A and B has cardinality equal to one, so no shared V-path can be created in V'.

An example of this is shown in Figure 5.7. Since 1-$remove(\sigma, \tau)$ is feasible, at most one simplex α_1 of the same dimension of τ is connected to σ. Thus, the new created V-paths cannot be *shared V-paths* since they will have a common origin (i.e., α_1).

" $\Rightarrow$ ". Assume that k-$cancellation(\sigma, \tau)$ is not a feasible k-$remove(\sigma, \tau)$ for (Σ, V). Let us call π the separatrix V-path $[\tau = \tau_0, (\sigma_1, \tau_1), (\sigma_2, \tau_2), \ldots, (\sigma_r, \tau_r), \sigma_{r+1} = \sigma]$. Let σ_l be the first simplex of π which belongs to a separatrix V-path between $\beta_j \in \mathrm{B}$ and

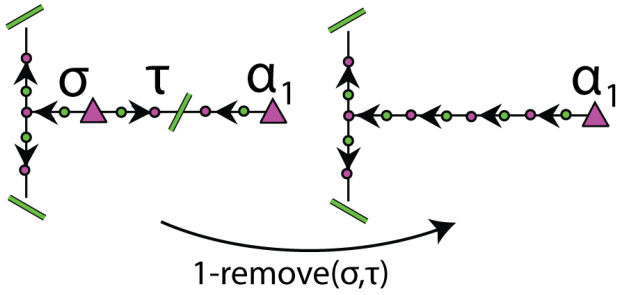


Figure 5.7: 1-$remove(\sigma, \tau)$ operator not introducing any shared V-path.

σ. Dually, let τ_m be the last simplex of π which belongs to a separatrix V-path between $\alpha_i \in \mathrm{A}$ and τ. Since V has no shared V-path, $m < l$ and each newly created separatrix V-path between β_j and α_i will contain the V-path $\pi' = [(\sigma_l, \tau_{l-1}), \ldots, (\sigma_{m+1}, \tau_m)]$. Since both $\#\,\mathrm{A}$ and $\#\,\mathrm{B}$ are greater than 1, π' is a shared V-path for (Σ, V').

Conversely to the example shown in Figure 5.7, the configuration depicted in Figure 5.8 is not valid for 1-$remove(\sigma, \tau)$ since multiple 2-saddles are connected with σ (i.e. α_1,α_2 and α_3). As a result of applying 1-$cancellation(\sigma, \tau)$ we introduce a shared V-path, depicted in red, created overlapping the new V-paths having different origin and destination.

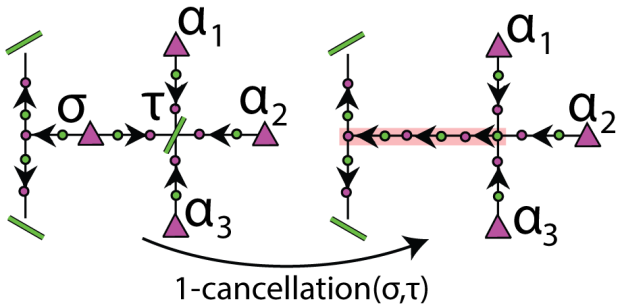


Figure 5.8: 1-$cancellation(\sigma, \tau)$ operator introducing a shared V-path.

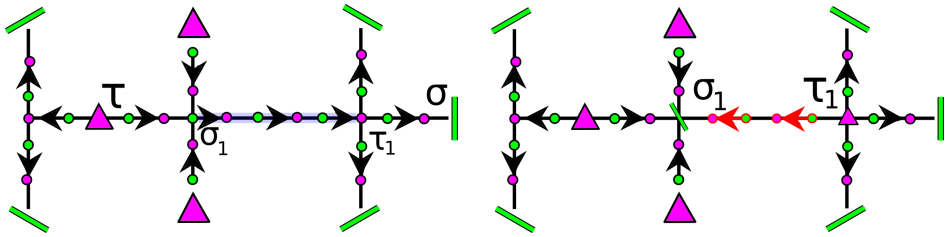
□

5.3.2 Shared *V*-path disambiguation algorithm

In this subsection, we propose a preprocessing step aimed to untie the shared V-paths in a simplicial complex Σ endowed with a Forman gradient V. We describe here such algorithm for the case of 3-dimensional simplicial complexes, but it can be extended to simplicial complexes of arbitrary dimension as well.
The idea at the basis of the shared V-path disambiguation algorithm is to modify the separatrix V-paths between 1-saddles and 2-saddles, inserting, by the performance of the undo of a $cancellation$, new dummy critical simplices in such a way that all the separatrix V-paths sharing the same path will end (or start) at the same critical saddle. When looking at the separatrix V-paths connecting maxima with 2-saddles and minima with 1-saddles, this property is guaranteed by construction, i.e., V-paths starting from a maximum can

only split, while V-paths reaching a minimum can only merge.

Figure 5.9 illustrates the key ideas of the algorithm. The traversal starts from critical edge σ and continues visiting the triangles in the separatrix V-path by navigating the arrows in reverse order. At triangle τ_1, three separatrix V-paths split, then the triangle is identified as part of the shared path. Continuing the traversal, on edge σ_1 different separatrix V-paths merge. Thus, σ_1 is identified as the beginning of the shared path, and τ_1 and σ_1 are introduced as critical (see Figure 5.9(b)).



(a) (b)

Figure 5.9: Shared V-path identified (a) and disambiguated inserting dummy critical simplices σ_1 and τ_1 (b).

Algorithm 7 and Algorithm 8 show the pseudocode description of the shared V-path disambiguation process. They are based on the following functions and procedures:

- $startingPaths(\sigma, V)$ returns the simplices different to the one paired with σ (if it exists), belonging to a separatrix V-path and on the immediate coboundary of σ;
- $countSplittingSeparatrix(\tau_i, V)$ counts the number of separatrix V-paths outgoing from the simplices in the immediate boundary of τ_i;
- $adjacentPaired(\tau_i, V)$ returns the simplices different to τ_i, belonging to a separatrix V-path and on the immediate coboundary of the simplex with which τ_i is paired;
- $V.getPair(\tau_i)$ returns the simplex paired in V with τ_i;
- $reversePath(\sigma_i, \tau_i, V, \mathcal{A})$ declares as critical the simplices σ_i and τ_i and updates V by reversing the gradient path between them.

Algorithm 7 describes the algorithm for disambiguation of V-paths. Starting from a critical edge σ, the separatrix V-paths converging in it are considered (lines 2-4). For each separatrix V-path, the first triangle incident into σ and belonging to the path is pushed onto a stack S (lines 4-6). While S is not empty, the first triangle τ_i is popped from the stack and the number of separatrix V-paths outgoing from its boundary edges are computed (function $countSplittingSeparatrix(\tau_i, V)$). If there are multiple separatrix V-paths that split at τ_i (see τ_1 in Figure 5.9(a)), the visit of a shared path begins (line 11).

Algorithm 7 $IdentifySharedPath(V, \mathcal{A})$

```
1: INPUT: V, gradient vector field; A, set of critical simplices
2: for all critical edges σ in A do
3:    F := startingPaths(σ, V)
4:    for all triangles τ_i in F do
5:       Stack S := ∅
6:       S.push(τ_i)
7:       while S ! = ∅ do
8:          τ_i := S.pop()
9:          nSplit := countSplittingSeparatrix(τ_i, V)
10:         if nSplit > 1 then
11:            visitSharedPath(τ_i, V, A)
12:
13:         F := adjacentPaired(τ_i, V)
14:         for all triangles τ_j in F do
15:            S.push(τ_j)
```

Algorithm 8 $VisitSharedPath(\tau_i, V, \mathcal{A})$

```
1: INPUT: τ_i is a triangle
2: INPUT: V, gradient vector field; A, set of critical simplices
3: σ_i := V.getPair(τ_i)
4: F := startingPaths(σ_i, V)
5: if #F = 1 then
6:    τ_j := F
7:    nSplit := countSplittingSeparatrix(τ_j, V)
8:    if nSplit > 1 then
9:       // if a new splitting face is found τ_i is updated
10:      τ_i := τ_j
11:   return visitSharedPath(τ_j, V, A)
12: if #F > 1 then
13:   reversePath(σ_i, τ_i, V, A)
```

Algorithm 8 describes the traversal of a shared V-path. Starting from the triangle τ_i on which the shared V-path splits, the edge σ_i, paired with it, is extracted (line 3). Function $startingPaths(\sigma_i, V)$ returns the set F of triangles different from τ_i and incident in σ_i that are in some separatrix V-path (line 4). If F has cardinality equal to one, we are still visiting the shared V-path (line 5). Otherwise, if the cardinality of F is greater than one, we are on an edge σ_i on which multiple separatrix V-paths are collapsing (see σ_1 in Figure 5.9(a)). If this is the case, τ_i and σ_i are introduced as dummy critical simplices and the arrows between them are reversed (lines 12-13). If $\#F$ was zero, we ended into a single critical triangle, thus we were not on a real shared V-path and no critical simplices are introduced. Note that, during the visit of a shared V-path, triangle τ_i can be updated if another triangle, closer to τ_i, is found on which separatrix V-paths split (lines 7-10).

Algorithm 8 visits the simplices of the shared V-path it identifies by performing constant time operations. Then, it reverses the visited shared V-path. This latter procedure and, consequently, Algorithm 8 have a linear time complexity in the number of simplices in the identified shared V-path.
Algorithm 7, instead, visits all the separatrix V-paths once for each 1-saddle. Thus, it has a worst-case time complexity of $O(s_1 \cdot s_V)$, where s_1 is the number of 1-saddles and s_V the number of simplices forming the separatrix V-paths.

Once all shared V-paths have been identified and disambiguated, we perform a simplification step for removing all the dummy critical simplices. Since the insertion of a pair of critical simplices (σ,τ) can be seen as the undo of a $cancellation$, performing cancellations would restore the initial inconsistency situations in the complex. Thus, we use only $remove$ operators that will trigger macro-operators working on extremum-saddle pairs.

Dummy critical simplices and obstructions
Obstructions are critical point configurations that cannot be simplified either using a $cancellation$ or a $remove$ operator. Specifically an ***obstruction*** is a pair of critical points, of consecutive index connected by multiple paths. The presence of obstructions can lead to degenerate configurations, called ***fingers***, that cannot be simplified. Such configurations typically do not appear in the initial state of the dataset but arise, with the undergoing of simplifications, in flat areas [GDN$^+$07]. Even if flat areas are not allowed, when computing a Forman gradient with the algorithm described in [RWS11], obstructions are still present in the data since they describe the natural behavior of the field.

Let us consider the shared V-paths. When the obstruction is present inside a shared V-path, the introduction of a dummy critical simplex can be avoided (since any simplification passing by that part will be unfeasible). When obstructions involve critical simplices in the neighborhood, there is a degenerate configuration that could prevent the removal of the dummy critical simplices. We show such configuration in Figure 5.10. The 2-saddles (purple triangles) and the 1-saddles (green edges) are all connected with extrema (maxima and minima, respectively) through multiple paths. Thus, the macro-operator cannot remove the two dummy critical simplices since none of the 2- or 1-saddles in the neighborhood can be removed. However, it is still important to introduce the pair since otherwise, the shared V-path could be affected by a swap during the simplification algorithm. Note that if this is the case, it means that the dummy pair will be removed in the future.

Dummy critical simplices that have not been removed during the simplification process can be removed at the end, with a $cancellation$, avoiding the visualization of spurious cells.

Even if this can be seen as a degenerate problem that could bring to uncontrolled results, we have noticed that the introduction of a dummy pair never inhibits the application of other $remove$ operators. In other words, we can always assume that the number of $remove$ operators, preserving the shared V-paths, which can be applied on a Forman gradient V without a dummy pair, is always less or equal to the number of $remove$ operators

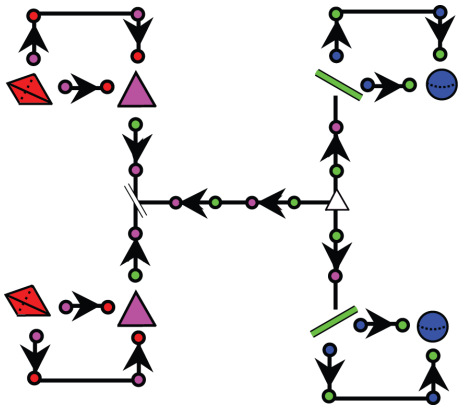


Figure 5.10: Example of obstructions preventing the removal of a dummy critical pair. Red tetrahedra correspond to maxima, purple triangles to 2-saddles, green edges are 1-saddles and blue spheres correspond to minima. Red, purple, green and blue dots correspond to (non critical) tetrahedra, triangles, edges and vertices, respectively. The white triangle and edge are the dummy critical simplices.

which can be applied on V after the insertion of the dummy pair. This is important to guarantee that the simplification is never obstructed by our disambiguation method.

5.4 Experimental results

The compact data structure introduced in Section 5.1 and the process described in Subsection 5.3.2 to untie shared V-paths have been exploited to define a new topologically-consistent algorithm for the morphological simplification of a discrete Morse complex. In order to obtain such a result, we have combined the shared V-path disambiguation algorithm with a simplification algorithm based on the $remove$ operator. We discuss here the results obtained when simplifying real datasets. Experiments have been performed on a desktop computer with a 3.2Ghz processor and 16GB of memory. The datasets chosen for our experiments are originated from regularly distributed data. The 3-dimensional simplicial complexes are obtained by removing points (and tetrahedra) corresponding to the empty space and removing flat areas (adjacent vertices with the same field value) through edge contractions.

We use a Discrete Morse Incidence Graph ($DMIG$) (see Section 5.1) for representing the pairs of critical simplices connected by a separatrix V-path. $remove$ operators are applied in ascending order of persistence using a priority queue. At each step, the simplification with the lowest persistence value is performed, the gradient arrows along the path are updated as well as the $DMIG$, and the new available simplifications are inserted in the priority queue. Once the queue is empty, or all the valid simplifications have a persistence value higher than a user defined threshold, the simplification algorithm ends.

We have studied the preprocessing step by evaluating its impact on the overall compu-

Dataset	*Size*	$\|\Sigma_0\|$	$\|\Sigma_3\|$	$\#C$	*Preprocessing*		*Simplification*		*Mem. Peak*
					$\#C_{ins}$	*Time*	*Rem*	*Time*	(GB)
BUCKY	32^3	32K	0.17M	2K	156	2.4 sec	1K	6.39 sec	0.09
FUEL	64^3	13K	0.06M	2.7K	54	0.65 sec	1.3K	4.13 sec	0.05
SILICIUM	$98x34x34$	66K	0.36M	2.1K	290	1.6 sec	1K	17.5 sec	0.1
NEGHIP	64^3	0.12M	0.64M	12.6K	234	10.7 sec	6.3K	3.8 min	0.2
SHOCKWAVE	$64x64x512$	1.2M	7M	1.1K	55	20.1 sec	582	2.8 min	2.4
BLUNT	$256x128x64$	1.0M	6M	11.2K	1378	10.4 min	5.5K	22.2 min	1.9
HYDROGEN	128^3	0.6M	3.9M	15.1K	2133	24.1 min	7.5K	24.3 min	2.2

Table 5.2: Evaluation of the preprocessing step and the remove-based simplification. For each dataset we indicate, the original size and the number of vertices, tetrahedra and critical points (columns $Size$, $|\Sigma_0|$, $|\Sigma_3|$ and $\#C$, respectively) in the tetrahedral mesh. In column *Preprocessing*, we show the number of critical points introduced during the preprocessing step and the timings for: identifying the shared V-paths, insert the critical points and remove them. Column *Simplification* shows the total number of simplifications performed and the time required by the algorithm. Column *Mem. Peak* indicates the maximum amount of memory used.

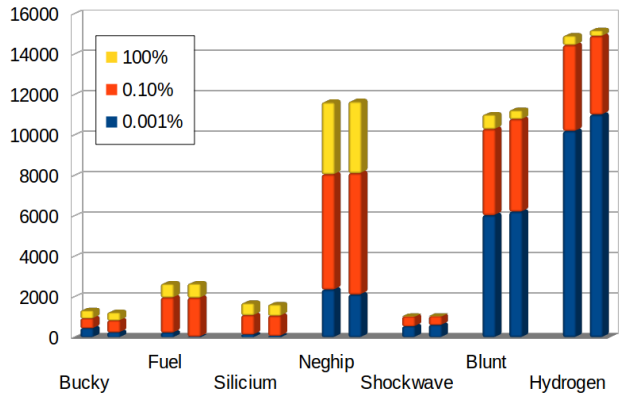
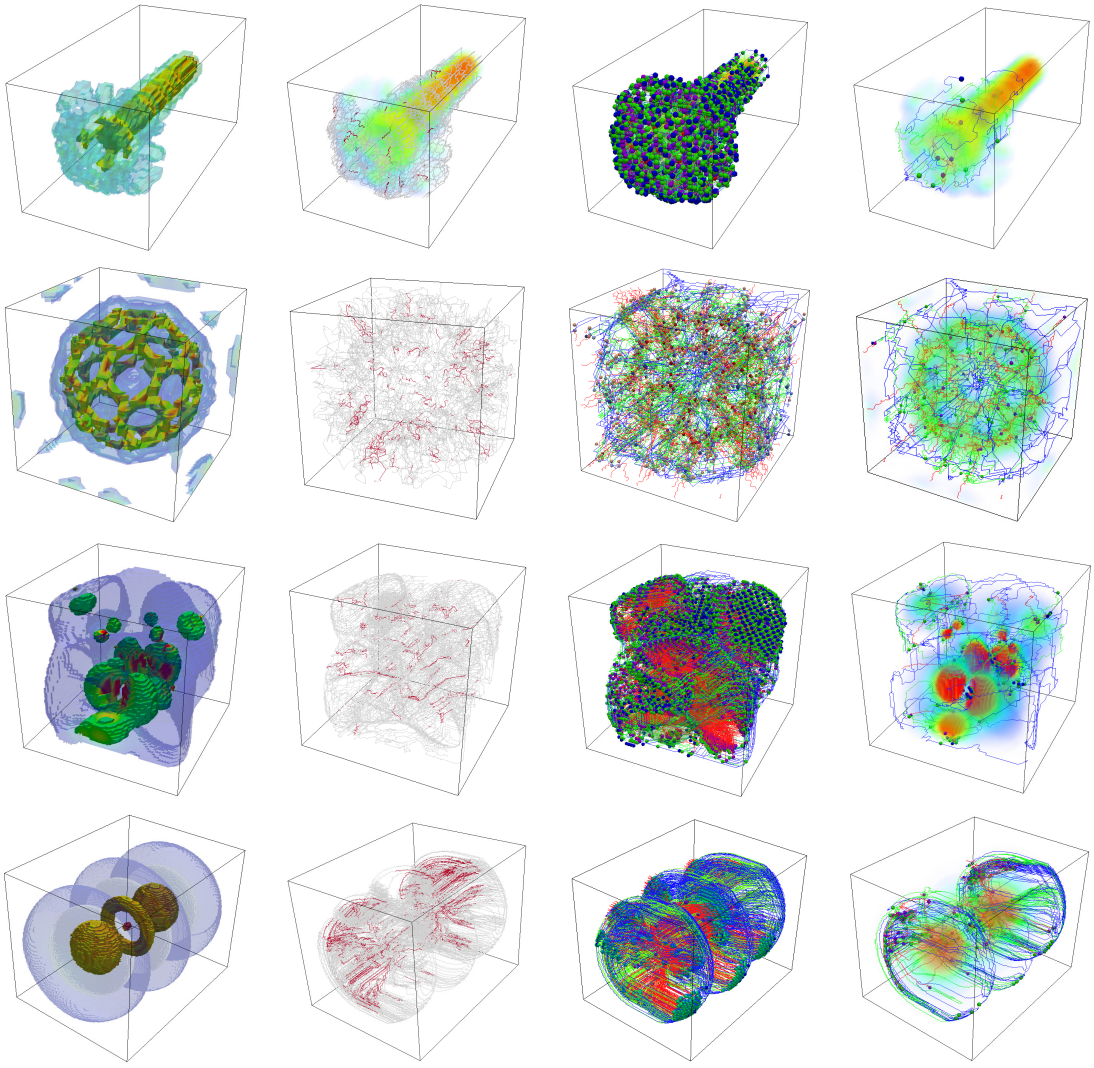


Figure 5.11: Nodes deleted by the $remove$-based (columns on the right) and the $cancellation$-based (columns on the left) algorithms using different simplification errors.

tation. In Table 5.2, we present the results obtained. We can notice that the number of critical simplices artificially introduced (column $\#C_{ins}$) varies depending on the dataset and is between 2-13% of the total number of critical simplices and all of them are removed during this phase. The timings of the preprocessing algorithms can be relevant with respect to the whole simplification process and, in a worst-case scenario (HYDROGEN), the time required for identifying and disambiguating shared V-paths and removing the dummy critical simplices is equal to the time required for simplifying the entire complex. The complexity of the preprocessing step depends on the number of separatrix V-paths between saddles and on their size, i.e., on the number of simplices forming them. In Figure 5.12, we show the results obtained by simplifying FUEL, BUCKY, NEGHIP and HYDROGEN 3-dimensional simplicial complexes. For HYDROGEN dataset, we can notice that shared V-paths are quite numerous and spread around the entire complex, unlike what happens with FUEL, BUCKY and NEGHIP.



(a) (b) (c) (d)

Figure 5.12: Topologically-consistent simplification of the FUEL, BUCKY, NEGHIP and HYDROGEN. The original scalar field (a) and the shared paths depicted in red (b). The original 1-skeleton of the MS complex (c) and its simplified version (d) computed with a persistence threshold of 0.01% with respect to the maximum persistence for FUEL, 0.2 for BUCKY and HYDROGEN and 0.3% for NEGHIP.

We have also studied the $remove$ operations triggered by the macro-operators during the removal of the dummy critical simplices. Specifically, we focus on studying the persistence associated with the deleted nodes in order to ensure that interesting features were not deleted during the preprocessing step. As discussed in [GRSW13], there is a correlation between noise and shared V-paths. We have found that 98% of the removals applied during the preprocessing step delete nodes that would be removed by the classical algorithm using a persistence threshold lower than 0.01% of the maximum persistence. Nodes in the remaining 2% have a persistence lower than 0.1% of the maximum persistence. Typically, values of persistence lower than 0.2% of the maximum persistence are considered noise.

Studying the entire simplification algorithm, we have verified experimentally the correctness of our approach comparing the graph updated during the simplification process and the one extracted from the simplified Forman gradient after each simplification step.

Moreover, we have compared our $remove$-based simplification algorithm with a standard $cancellation$-based algorithm testing whether the number of critical simplices in the fully simplified Forman gradient are comparable. The graph depicted in Figure 5.11 shows the number of critical simplices deleted using different simplification errors. For each dataset, the column on the right indicates the results obtained with the $remove$-based algorithm, while the results obtained with the $cancellation$-based algorithm are shown in the columns on the left. As we can notice, the number of critical simplices removed is comparable in both approaches. This result guarantees that the simplification sequence obtained using the $remove$ operator removes features in a controlled and progressive way, as the $cancellation$-based method.

5.5 Concluding remarks

We have presented a new simplification algorithm for a discrete Morse gradient that guarantees the topological consistency of Morse and Morse-Smale complexes generated from the simplification. Fundamental steps of the proposed algorithm are the shared V-path disambiguation process and the use of the $remove$ operator. The first one allows freeing the gradient from all the configurations leading to inconsistencies, the second one to simplify the discrete Morse complex in a topologically-consistent way. We have proved the correctness of our approach, and we have evaluated experimentally its performances with respect to a classical $cancellation$-based approach. Another contribution of our work is the introduction of a new graph-based data structure representing Morse complexes based on the Morse Incidence Graph (MIG). This structure has been obtained by removing almost all the geometric attributes associated with each node of the MIG. Experimental evaluations have revealed that this data structure allows to encode a Morse complex efficiently with a minimum loss in storage cost.

The simplification algorithm proposed in this chapter can lead to immediate improvements in various application domains. We focus here on two of them on which we plan to work in the immediate future. The first one concerns persistent homology computation, the second one is related to the expressive power of a multi-resolution model for a Morse complex.

As described in Subsection 2.3.3.4 and further developed in Chapter 3, discrete Morse theory can be exploited to efficiently retrieve standard and persistent homology of a simplicial complex. In this context, discrete Morse theory allows us to reach two goals: a time improvement in the computation of homological information, and the retrieval of a compact but homologically equivalent representation of the input simplicial complex.
Differently to the standard homology, persistent homology is not a topological invariant

and it strongly depends on the filtration considered to analyze the complex. In some situations, for instance, when the Forman gradient is built starting from a noisy function defined on the vertices of the considered complex, this dependency leads to the construction of a oversize discrete Morse complex and consequently to the retrieval of groups of persistent homology affected by noise. In a similar situation, noise can be easily detected and removed once the persistent homology is computed but, by adopting this strategy, the algorithm retrieving persistent homology has to deal with a large-size complex and its execution can require too much time. In order to speed up the computation of a noise-free persistent homology, a simplification step can be useful. Let us suppose to have computed for a simplicial complex a filtered Forman gradient. Instead of directly compute persistent homology of the associated Morse complex and then denoise the obtained result, we propose of remove the redundant critical simplices that will likely cause noise by performing a simplification preprocessing on the filtered Forman gradient before to compute persistent homology. Since a simplification process is much less time-consuming than the algorithm to retrieve persistent homology groups, the time required for the entire computation of a noise-free persistent homology decreases by adopting this strategy.

As widely discuss in Chapter 4, a multi-resolution model permits to obtain different representations of any spatial object at different levels of detail. The level of detail can be uniform or vary other the object. In [BEHP04] and [BPH05], two different multi-resolution models have been defined for representing the cells of the Morse complexes computed on a terrain. In [GKK$^+$12] a similar model has been defined for volumetric datasets while in [ČDI12] it has been defined for simplicial meshes embedded in nD. Aside from the working dimension, all these models are built by applying a sequence of simplifications on a graph-based representation of the Morse complexes. The coarsest graph obtained is then stored in combination with the set of refinements, undo of the simplifications performed. The dependency relation for a refinement operator can be seen as the set of nodes/arcs that have to be in the graph for the operator to be successfully applied. The definition of a weak dependency relation (i.e., a dependency relation involving less nodes as possible) is crucial for a multi-resolution model since it augments its adaptivity.

From an application point of view, the compactness of the Forman gradient would make it a perfect candidate for representing the Morse cells in a multi-resolution framework. In this case, the base complex would be the coarse Forman gradient (i.e., the gradient with the minimum number of critical simplices) obtained through a sequence of simplifications applied to the initial Forman gradient, while the set of refinement operators would reintroduce pairs of critical simplices.
Following this approach, however, the dependency relation is defined in terms of critical simplices, present in the Forman gradient, and of their connections. Unfortunately, the undecidability introduced by the shared V-paths forces us to check those properties on the fly, navigating the Forman gradient. The resulting loss of efficiency would clearly make the model useless for an interactive experience.
The method we have proposed for disambiguating the shared V-paths solves this problem. Removing the uncertainty behind each simplification, allows us to define a multi-resolution model that is efficient since free of the local checks at runtime. Moreover, we believe that the implicit approach used in combination with a $remove$ operator and with

a Forman gradient free of the shared V-paths, let us to define the weakest dependency relation ever defined for these kinds of models that results in the multi-resolution model with a high expressive power.

In our plans, the proposed simplification algorithm is a first step in the definition of a tool able to represent a discrete Morse complex at different levels of detail. More precisely, we plan to develop a multi-resolution model for discrete Morse complex combining both geometrical and morphological modifications. This goal has been achieved in the context of 2-dimensional simplicial complex leading to the definition of a multi-resolution model for triangulated scalar field [ID14].

As discuss above, the resolution of the inconsistency problem addressed in [GRSW13] leads to a good management of the morphological modifications and to an increase of the expressive power of a multi-resolution model. Aside from formally proving the validity of the weak dependency relation above defined, the next steps required to reach our purpose are the development of a geometrical simplification algorithm for simplicial complexes endowed with a gradient vector field, and the definition and the design of a multi-resolution model based on geometrical and morphological simplification operators.

Simplification algorithms for discrete Morse complexes proposed in the literature reduce the size of such a complex only from a morphological point of view. Actually, they are based on simplification operators removing pairs of critical simplices while the underlying simplicial complex remains unchanged. In order to obtain a multi-resolution model with a wider expressive power, the possibility of simplifying discrete Morse complex also from a geometrical point of view is required. Our task is to simplify the underlying geometrical structure of a complex by reducing its size through homology-preserving operators while we maintain not only the homology but also the Forman gradient. Analogously to [ID14], we want to consider the edge contraction as simplifying operator.

Once that simplification operators able to handle the geometry of a discrete Morse complex without affecting the gradient behavior will be defined, we plan to combine them with morphology-modifying operators in order to design and implement a multi-resolution model for discrete Morse complexes based on both these kinds of modifications.
This model will be a tool on which both geometric and morphological simplifications can operate concurrently to reduce the complexity and to enhance the understanding of available volume datasets.

Chapter 6

Relations between Perfect Discrete Morse Functions and Betti Splittings

The main goal of the work developed in this chapter is to establish a first connection between the notions of perfect discrete Morse function and Betti splitting by exploiting the link between commutative algebra and algebraic topology provided by the Stanley-Reisner correspondence [Rei76, Sta75].

Thanks to this correspondence and to Hochster's formula [Hoc77], the homology groups of a simplicial complex can be related to some algebraic invariants obtained through the computation of the minimal graded free resolution of a suitable monomial ideal. For an ideal I, the existence of a particular decomposition, called Betti splitting, ensures that the minimal graded free resolution of I can be recovered from the resolutions of the ideals in which it has been decomposed.
Specifically, our task is to correlate the good property for a simplicial complex of admitting a perfect Morse function with the desirable property for a monomial ideal of admitting a splitting. The literature ensures us that there exist examples of simplicial complexes on which no perfect Morse function can be defined but whose associated ideal admits a Betti splitting [Bol15, LB13]. For this reason, in this chapter we focus on finding out under which assumptions admitting a perfect discrete Morse function implies admitting also a homological or a Betti splitting.

The chapter is organized as follow. Section 6.1 is devoted to introduce various background notions useful for the rest of the chapter, such as the Stanley-Reisner correspondence, the definition of Betti and homological splitting and the property of perfection for a discrete Morse function.
Then, Section 6.2 and Section 6.3 aim to present the two main contributions of this chapter. In both the cases, it has been proven that for a simplicial complex with non null top homology, admitting a perfect discrete Morse function ensures that the correspondent ideal can be suitable split. In Section 6.2, we prove that under the conditions mentioned above a homological splitting can be retrieved. In Section 6.3, instead, we describe which further hypotheses have to be added in order to obtain a Betti splitting. The first result

can be actually used as an algorithmic criterion to verify that a simplicial complex does not admit a perfect discrete Morse function, while the second one represents a first interesting connection between discrete Morse theory and the relevant algebraic notion of Betti splitting. Both the results have been first introduced in a general context in which some technical hypotheses are required. Then, by exploiting the Poincaré duality, they have been specialized in the more familiar context of the orientable manifolds. Another interesting fact is that all the proposed proofs are constructive. So, the knowledge of a perfect discrete Morse function allows us to actually build the desired splitting.
Moreover, the proven results ensure the existence of a Betti splitting for the wide class of the 2-manifolds. Finally, in Section 6.4, we discuss about the obtained results and we give an overview of the open questions and of the goals we would like to achieve in the near future.
Currently, a paper summarizing the results presented in this chapter is in progress [BFRDon].

6.1 Background

In this section, we provide some basic notions in combinatorics, algebra and topology useful for the rest.

Simplicial complexes and squarefree monomial ideals
Recall that an abstract simplicial complex Σ on V is a collection of finite subsets of V, called simplices, such that if $\tau \in \Sigma$, $\sigma \subseteq \tau$, then $\sigma \in \Sigma$. Since the two notions of simplicial complex and abstract simplicial complex are equivalent, in the following, when this does not create any ambiguity, we omit the term "abstract".

In the following, we always consider abstract simplicial complexes on $\underline{n} := \{1, \ldots, n\}$,we denote as $< \sigma_1, \ldots, \sigma_r >$ the (abstract) simplicial complex generated (by inclusion closure) by the top simplices $\sigma_1, \ldots, \sigma_r$, and we declare the empty set $\emptyset$ as a simplex of dimension -1 of any considered simplicial complex. A subset $\sigma \subseteq \underline{n}$ such that $\sigma \notin \Sigma$ is called a *nonface* of Σ. We denote by $N(\Sigma)$ the set of minimal (with respect to inclusion) nonfaces of Σ.

Given an (abstract) simplicial complex Σ on $\underline{n}$ and a field $\mathbb{F}$, we can associate with Σ two different monomial squarefree ideals in the polynomial ring $R = \mathbb{F}[x_1, \ldots, x_n]$.

Definition 6.1. The *Stanley-Reisner ideal* of Σ in R is

$$I_\Sigma = (x_\sigma \,|\, \sigma \in N(\Sigma))$$

where $x_\sigma = \prod_{i \in \sigma} x_i$.

Remark 6.2. (Stanley-Reisner correspondence, [MS05]). The function described in Definition 6.1 mapping a simplicial complex into a squarefree monomial ideal constitutes a bijection from the set of abstract simplicial complexes on $\underline{n}$ to the set of squarefree monomial ideals in $R = \mathbb{F}[x_1, \ldots, x_n]$.

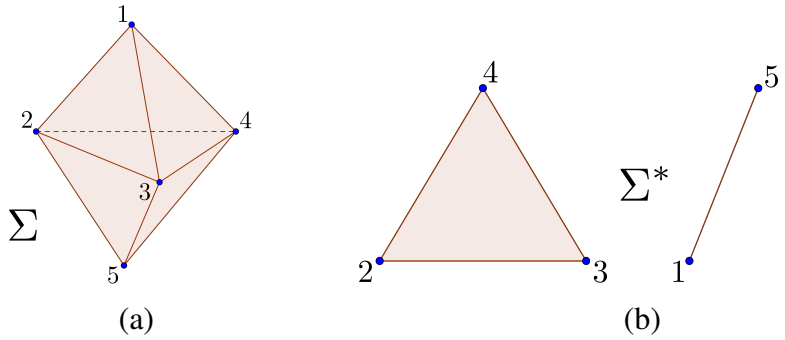


Figure 6.1: A simplicial complex Σ (a) and its Alexander dual Σ^* (b).

In order to define the second ideal associated with Σ, we need to associate with Σ another simplicial complex.

Definition 6.3. The *Alexander dual* of Σ is the simplicial complex defined as

$$\Sigma^* =< \underline{n} \setminus \sigma \,|\, \sigma \in N(\Sigma) >$$

Definition 6.4. The *Alexander dual ideal* of Σ in R is the Stanley-Reisner ideal I_{Σ^*} of Σ^*.

In the following, we denote the ideal I_{Σ^*} as I^*_Σ.

Example 6.5. Let Σ be the simplicial complex on $\underline{5} = \{1, \ldots, 5\}$ defined as $< 123, 124, 134, 235, 245, 345 >$ and depicted in Figure 6.1 (a).
According to the above definitions,

- the Stanley-Reisner ideal of Σ in R is $I_\Sigma = (x_1x_5, x_2x_3x_4)$;
- the Alexander dual of Σ is the simplicial complex $\Sigma^* =< 15, 234 >$ depicted in Figure 6.1 (b);
- the Alexander dual ideal of Σ in R is $I^*_\Sigma = (x_1x_2, x_1x_3, x_1x_4, x_2x_5, x_3x_5, x_4x_5)$.

Let $\mathbb{F}$ be a field, $R = \mathbb{F}[x_1, \ldots, x_n]$ be the polynomial ring on n variables and $\mathfrak{M} = (x_1, \ldots, x_n)$ its maximal homogeneous ideal. Let R be with the standard grading and M,N finitely generated graded R-module. A homogeneous homomorphism $\phi : M \to N$ of graded R-modules of degree d is a homomorphism such that $\phi(M_i) \subseteq N_{i+d}$ for each $i \in \mathbb{N}$. Given a integer r, denote by $M(r)$ the shifted R-module whose graded components are given by $M(r)_i := M_{r+i}$.

Definition 6.6. Let M be a finitely generated graded R-module. A *minimal graded free resolution* of M as R-module is a free resolution of M of the form

$$0 \to F_p \xrightarrow{\phi_p} \cdots \xrightarrow{\phi_2} F_1 \xrightarrow{\phi_1} F_0 \xrightarrow{\phi_0} M \to 0$$

where $F_i = \bigoplus_{j \in \mathbb{N}} R(-j)^{\beta_{i,j}(M)}$ are shifted free modules, ϕ_i are homogeneous of degree 0 and $Im(\phi_i) \subseteq \mathfrak{M}F_{i-1}$ for each integer i.

A minimal graded free resolution of a finitely generated graded R-module M always exists and it is unique up to isomorphism of chain complexes [Eis13]. Adjective "minimal" denotes that the number of basis elements of each shifted free module F_i is minimal. The invariants $\beta_{i,j}(M) = \dim_{\mathbb{F}}(F_i)_j$ are called *graded Betti numbers* of M. We denote as $\beta_i(M) = \sum_{j\in\mathbb{N}} \beta_{i,j}(M)$ the i^{th} *total Betti number* of M.

The *projective dimension* of an R-module M, denoted as $pd(M)$, measures the length of the minimal graded free resolution of M as R-module and it is given by $pd(M) = \sup\{i \in \mathbb{N} \,|\, \beta_{i,j}(M) \neq 0 \text{ for some } j \in \mathbb{N}\}$. One of the most important general results about resolutions is given by Hilbert's syzygies.

Theorem 6.7. (Hilbert's syzygies Theorem, [Hil90]). *Let M be a graded R-module. Then, $pd(M) \leq n$.*

Example 6.8. Let us consider the Alexander dual ideal I^*_Σ in R of the simplicial complex Σ considered in Example 6.5. So, I^*_Σ is the ideal $(x_1x_2, x_1x_3, x_1x_4, x_2x_5, x_3x_5, x_4x_5)$. A minimal graded free resolution of I^*_Σ is

$$0 \to R(-5) \to R(-4)^5 \to R(-3)^9 \to R(-2)^6 \to I^*_\Sigma \to 0$$

From it, the graded Betti numbers of I^*_Σ can be easily retrieved, e.g., $\beta_{0,2}(I^*_\Sigma) = 6$. Further, $pd(I^*_\Sigma) = 3$ and, according to Theorem 6.7, it is less than 5.

In [Hoc77], M. Hochster proved a remarkable formula that is the main bridge between combinatorics and commutative algebra allowing to establish a relation between the homology of a simplicial complex and the Betti numbers of its associated ideals. The version of Hochster's formula that we introduce is due to Eagon and Reiner [ER98].

Theorem 6.9. (Hochster's formula, graded version, [ER98]). *Let Σ be a simplicial complex on $\underline{n}$. Then,*

$$\beta_{i,j}(I^*_\Sigma) = \sum_{\sigma\in\Sigma, \#\sigma = n-j} \tilde{\beta}_{i-1}(\text{link}_\Sigma\, \sigma; \mathbb{F})$$

where, for each k, $\tilde{\beta}_k$ represents the k^{th} Betti number of reduced homology [MS05].

Remark 6.10. As mentioned above, Hochster's formula allows to link explicitly the homology of a simplicial complex Σ with the graded Betti numbers of I^*_Σ. By imposing $j = n$, we obtain $\beta_{i,n}(I^*_\Sigma) = \tilde{\beta}_{i-1}(\Sigma; \mathbb{F})$ which is equivalent to $\beta_0(\Sigma; \mathbb{F}) = \beta_{1,n}(I^*_\Sigma) + 1$ and $\beta_k(\Sigma; \mathbb{F}) = \beta_{k+1,n}(I^*_\Sigma)$ for $k > 0$.

Example 6.11. The just mentioned relation allows retrieving, for instance, the Betti numbers of the simplicial complex Σ introduced in Example 6.5 from the minimal graded free resolution of I^*_Σ described in Example 6.8. In this case, we obtain

$$\beta_k(\Sigma; \mathbb{F}) = \begin{cases} \beta_{1,5}(I^*_\Sigma) + 1 = 1 & \text{if } k = 0 \\ \beta_{2,5}(I^*_\Sigma) = 0 & \text{if } k = 1 \\ \beta_{3,5}(I^*_\Sigma) = 1 & \text{if } k = 2 \\ \beta_{k+1,5}(I^*_\Sigma) = 0 & \text{if } k \geq 3 \end{cases}$$

Homological and Betti splittings
For studying the graded Betti numbers of a monomial ideal I, it is natural to split I into smaller monomial ideals J, K where $I = J + K$ trying to recover $\beta_{i,j}(I)$ from $\beta_{i,j}(J)$, $\beta_{i,j}(K)$ and $\beta_{i-1,j}(J \cap K)$.

Given a monomial ideal in R, we denote as $G(I)$ the minimal system of monomial generators of I.

Definition 6.12. Let I, J and K be monomial ideals such that $I = J + K$ and $G(I)$ is the disjoint union of $G(J)$ and $G(K)$. Then, $J + K$ is a *Betti splitting* of I with respect to $\mathbb{F}$ if, for all $i, j \in \mathbb{N}$,

$$\beta_{i,j}(I) = \beta_{i,j}(J) + \beta_{i,j}(K) + \beta_{i-1,j}(J \cap K)$$

Example 6.13. Let us consider as I the ideal $(x_1x_2, x_1x_3, x_1x_4, x_2x_5, x_3x_5, x_4x_5)$. A minimal graded free resolution of I has been showed in Example 6.8. Let $J = (x_4x_5)$, $K = (x_1x_2, x_1x_3, x_1x_4, x_2x_5, x_3x_5)$ be two ideals such that $I = J + K$ and $G(I)$ is the disjoint union of $G(J)$ and $G(K)$.
Minimal graded free resolutions of J, K and $J \cap K$ are, respectively,

$$\begin{array}{rcccl} & & 0 \to R(-2) & \to \quad J & \to 0 \\ 0 \to R(-4)^2 & \to R(-3)^6 & \to R(-2)^5 & \to \quad K & \to 0 \\ 0 \to R(-5) & \to R(-4)^3 & \to R(-3)^3 & \to J \cap K & \to 0 \end{array} \tag{6.13.1}$$

Then, it is easy to check that $J + K$ is a Betti splitting of I.

By using the Stanley-Reisner correspondence (Remark 6.2) and the Alexander duality (Definition 6.3), we can state the above condition for a simplicial complex Σ. First at all, we need the analogous of the request that two minimal systems of generators are disjoint.

Definition 6.14. Let Σ be a simplicial complex. Consider a partition of the set of its top simplices in two disjoint subsets T_1 and T_2. Let $\Sigma_1 =< \tau \,|\, \tau \in T_1 >$ and $\Sigma_2 =< \tau \,|\, \tau \in T_2 >$ be the two simplicial complexes generated by the simplices in T_1 and T_2, respectively.
We call the decomposition $\Sigma = \Sigma_1 \cup \Sigma_2$ a *standard decomposition* of Σ.

Here, the analogous for a simplicial complex Σ of the notion of Betti splitting introduced in Definition 6.12.

Definition 6.15. Let $\Sigma = \Sigma_1 \cup \Sigma_2$ be a standard decomposition of a simplicial complex Σ. We call the decomposition $\Sigma = \Sigma_1 \cup \Sigma_2$ a *Betti splitting* of Σ with respect to $\mathbb{F}$ if $I^*_\Sigma = I^*_{\Sigma_1} + I^*_{\Sigma_2}$ is a Betti splitting of I^*_Σ with respect to $\mathbb{F}$.

It is possible to express the Betti splitting condition for the Alexander dual ideal I^*_Σ of a simplicial complex Σ directly on the complex, introducing the notion of homological splitting.

Definition 6.16. ([Bol15], Definition 5.7). Let $\Sigma = \Sigma_1 \cup \Sigma_2$ be a standard decomposition of a simplicial complex Σ. We say that $\Sigma = \Sigma_1 \cup \Sigma_2$ is a *homological splitting* of Σ with respect to $\mathbb{F}$ if the following conditions are satisfied.

$$\beta_0(\Sigma;\mathbb{F}) = \begin{cases} \beta_0(\Sigma_1;\mathbb{F}) + \beta_0(\Sigma_2;\mathbb{F}) & \text{if } \Sigma_1 \cap \Sigma_2 = \emptyset \\ \beta_0(\Sigma_1;\mathbb{F}) + \beta_0(\Sigma_2;\mathbb{F}) - 1 & \text{otherwise} \end{cases}$$

$$\beta_1(\Sigma;\mathbb{F}) = \begin{cases} \beta_1(\Sigma_1;\mathbb{F}) + \beta_1(\Sigma_2;\mathbb{F}) & \text{if } \Sigma_1 \cap \Sigma_2 = \emptyset \\ \beta_1(\Sigma_1;\mathbb{F}) + \beta_1(\Sigma_2;\mathbb{F}) + \beta_0(\Sigma_1 \cap \Sigma_2;\mathbb{F}) - 1 & \text{otherwise} \end{cases}$$

$$\beta_k(\Sigma;\mathbb{F}) = \beta_k(\Sigma_1;\mathbb{F}) + \beta_k(\Sigma_2;\mathbb{F}) + \beta_{k-1}(\Sigma_1 \cap \Sigma_2;\mathbb{F}) \text{ for } k > 1$$

By Hochster's formula (Theorem 6.9 and Remark 6.10), we can consider a homological splitting as a Betti splitting in degree n. So, if $\Sigma = \Sigma_1 \cup \Sigma_2$ is a Betti splitting of Σ, then $\Sigma = \Sigma_1 \cup \Sigma_2$ is a homological splitting of Σ. In general, the converse does not hold ([Bol15], Example 5.8).

Example 6.17. Let us consider the simplicial complex $\Sigma =< 123, 124, 134, 235, 245, 345 >$ depicted in Figure 6.1 (a). Let Σ_1, Σ_2 be the simplicial complexes $< 123 >$ and $< 124, 134, 235, 245, 345 >$, respectively. $\Sigma_1 \cup \Sigma_2$ is a standard decomposition of Σ. Moreover, since ideals I^*_Σ, $I^*_{\Sigma_1}$ and $I^*_{\Sigma_2}$ coincide respectively with ideals I, J and K considered in Example 6.13, equation (6.13.1) ensures that $\Sigma_1 \cup \Sigma_2$ is a Betti splitting (and so, also a homological splitting) of Σ.

The following theorem describes how admitting a Betti splitting can be characterized in terms of homological splittings.

Theorem 6.18. ([Bol15], Theorem 5.14). *Let $\Sigma = \Sigma_1 \cup \Sigma_2$ be a standard decomposition of a simplicial complex Σ. Then, the following statements are equivalent:*

- *$\Sigma = \Sigma_1 \cup \Sigma_2$ is a Betti splitting of Σ with respect to $\mathbb{F}$;*
- *$\operatorname{link}_\Sigma \sigma = \operatorname{link}_{\Sigma_1} \sigma \cup \operatorname{link}_{\Sigma_2} \sigma$ is a homological splitting with respect to $\mathbb{F}$ of $\operatorname{link}_\Sigma \sigma$, for each $\sigma \in \Sigma_1 \cap \Sigma_2$.*

Perfect discrete Morse functions

According to the definition provided in Subsection 1.3.3, a discrete Morse function f on a simplicial complex Σ is an increasing function with respect to the dimension of the simplices which, for each k-simplex, admits at most one exception to the rule above among the simplices of dimension $k-1$, or among the simplices of dimension $k+1$. Simplices of dimension k which do not admit any exception are called critical k-simplices of f. Let us denote as $C_k(f)$ the set of the critical k-simplices of f.

Theorem 6.19. (Weak Morse inequality, [For98]). *Let Σ be a simplicial complex and let f be a discrete Morse function defined on it. Then, for each k and each field $\mathbb{F}$,*

$$\beta_k(\Sigma;\mathbb{F}) \leq C_k(f)$$

Definition 6.20. If the discrete Morse function f is such that, for each k, $\beta_k(\Sigma;\mathbb{F}) = C_k(f)$, f is said *perfect* with respect to $\mathbb{F}$.

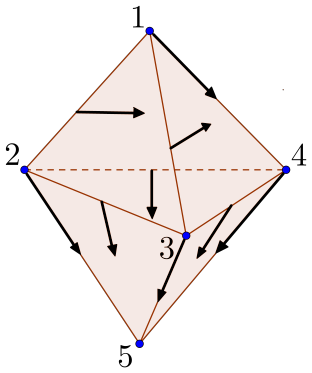


Figure 6.2: An acyclic matching defined on a simplicial complex and inducing a perfect discrete Morse function.

Example 6.21. The simplicial complex Σ triangulating a sphere depicted in Figure 6.1 (a) can be easily endowed with a perfect discrete Morse function. For instance, the acyclic matching on Σ depicted in Figure 6.2 induces a perfect discrete Morse function having as critical simplices the edge 123 and the vertex 5.

Example 6.22. Each collapsible simplicial complex Σ (i.e., Σ can be simplified to a single point space by using elementary reductions) admits a perfect Morse function. Moreover, a simplicial complex Σ is collapsible if and only if Σ admits a discrete Morse function with exactly one critical simplex.

Topological and combinatorial manifolds

Intuitively, a manifold is a topological space which locally behaves like an Euclidean space. Manifolds form an important class of topological spaces with applications in several mathematical fields. Here, we focus our attention on topological manifolds but, in general, a manifold can be equipped with other additional structures, such as a differentiable structure. More details about the topics of this section can be found in [Hud69, BR03].

Definition 6.23. A topological space X is called *locally homeomorphic to* $\mathbb{E}^d$ if every point $x \in X$ has a neighbourhood which is homeomorphic to the d-dimensional Euclidean space $\mathbb{E}^d$.

Definition 6.24. A *(topological) d-manifold* M is a Hausdorff space that is locally homeomorphic to the d-dimensional Euclidean space $\mathbb{E}^d$.
A *(topological) d-manifold with boundary* M is a Hausdorff space in which every point has a neighbourhood homeomorphic to the d-dimensional Euclidean space $\mathbb{E}^d$ or to the d-dimensional Euclidean half-space $\mathbb{H}^d := \{x \in \mathbb{R}^d \mid x_d \geq 0\}$.

Remark 6.25. Since any metric space is a Hausdorff space, for a d-manifold M which is a subset of an Euclidean space, the request that M is a Hausdorff space is superfluous.

For simplicial complexes, the notion of manifold can be described in a more combinatorial way. To do that, we need a tool which can locally describe the behavior of a simplicial complex by representing a neighbourhood for each simplex of Σ.

Definition 6.26. A simplicial d-complex Σ is a *combinatorial d-manifold [with boundary]* if the link of every vertex is homeomorphic [either] to the $(d-1)$-sphere $\mathbb{S}^{d-1} := \left\{x \in \mathbb{R}^d \mid d_2(x,0) = 1\right\}$ [or to the $(d-1)$-disk $D^{d-1} := \left\{x \in \mathbb{R}^{d-1} \mid d_2(x,0) \leq 1\right\}$].

A combinatorial manifold can be also described in terms of the links of all its simplices and not only of the links of its vertices.

Proposition 6.27. *Let Σ be a combinatorial d-manifold and let σ be a k-simplex of Σ. Then, $\mathrm{link}_\Sigma \sigma$ is homeomorphic to the $(d-k-1)$-sphere $\mathbb{S}^{d-k-1}$.*

Proof. By induction on the dimension k of σ. For the inductive step, it enough to notice that, writing σ as $v\sigma'$ where v and σ' are simplices of Σ of dimension 0 and $k-1$ respectively, $\mathrm{link}_\Sigma \sigma = \mathrm{link}_{\mathrm{link}_\Sigma \sigma'} v$. □

Proposition 6.28. *Let Σ be a simplicial complex. If Σ is a combinatorial d-manifold [with boundary], then $|\Sigma|$ is a topological d-manifold [with boundary].*

Remark 6.29. In general, the converse is not true. The equivalence between the notion of topological and combinatorial manifold holds for simplicial complexes of dimension $d \leq 3$. For $d > 4$, there exist simplicial d-complexes which are topological manifold but not combinatorial manifold. For $d = 4$, the problem is still open.

Intuitively, a d-manifold M is called *orientable* if it has a global consistent choice of orientation. For instance, the Möebius strip and the Klein bottle are two example of non-orientable spaces. For a formal definition of the notion of orientability we refer to Chapter 3 of [Hat02].

Definition 6.30. A topological space X is called *compact* if every open cover of X (i.e., a collection of open sets of X whose union contains X) has a finite subcover.

In the following, we will denote as *closed d-manifold* any compact d-manifold without boundary.

Theorem 6.31. (Poincaré duality Theorem). *Let M be an orientable closed d-manifold. Then, for any integer k and any field $\mathbb{F}$,*

$$H_k(M;\mathbb{F}) \cong H_{d-k}(M;\mathbb{F})$$

Poincaré duality Theorem is usually enunciated in a more general formulation not only for coefficients in a field [Hat02]. In that case, the theorem states that the k^{th} cohomology group of M is isomorphic to the $(d-k)^{th}$ homology group of M.

Remark 6.32. For a closed d-manifold not necessarily orientable, the isomorphism stated in Theorem 6.31 between $H_k(M;\mathbb{F})$ and $H_{d-k}(M;\mathbb{F})$ is ensured only for $\mathbb{F} = \mathbb{Z}_2$.

6.2 Perfect discrete Morse functions and homological splittings

The main aim of this chapter is to investigate the connections between admitting Betti splittings and perfect discrete Morse functions.

Example 6.33. The simplicial complex $\Sigma =< 123, 124, 134, 235, 245, 345 >$, elected as our reference example, satisfies both the just mentioned properties. Let us endow Σ with the perfect discrete Morse function described in Example 6.21 and depicted in Figure 6.2. In this example, we notice that the decomposition of Σ obtained by the removal of the critical top simplex 123 represents a Betti splitting for Σ (Example 6.17) and so, for the ideal I^*_Σ (Example 6.13).

Inspired by this example, in this and in the next section, we enunciate and prove that, under suitable conditions, admitting a perfect discrete Morse function implies admitting a homological or a Betti splitting.

If the top homology of a simplicial complex is not null, then the existence of a perfect discrete Morse function ensures the existence of a homological splitting.

Proposition 6.34. ([DE93], Section 3). *Let Σ be a simplicial complex, τ be a top simplex of dimension d in Σ and $\mathbb{F}$ be a field. Denoting the simplicial complex $\Sigma \setminus \{\tau\}$ as Σ', we have that:*

- *if τ belongs to a d-cycle of Σ, then*

$$\beta_k(\Sigma';\mathbb{F}) = \begin{cases} \beta_k(\Sigma;\mathbb{F}) - 1 & \text{if } k = d \\ \beta_k(\Sigma;\mathbb{F}) & \text{otherwise} \end{cases}$$

- *if τ does not belong to a d-cycle of Σ, then*

$$\beta_k(\Sigma';\mathbb{F}) = \begin{cases} \beta_k(\Sigma;\mathbb{F}) + 1 & \text{if } k = d-1 \\ \beta_k(\Sigma;\mathbb{F}) & \text{otherwise} \end{cases}$$

Remark 6.35. Proposition 6.34 ensures us that removing from Σ a top simplex of dimension d only affects either the d^{th} or the $(d+1)^{th}$ homology group of Σ and that in each situation just one of these modifications can happen.

Theorem 6.36. *Let Σ be a simplicial complex of dimension d such that $\beta_d(\Sigma;\mathbb{F}) \neq 0$. If Σ admits a perfect discrete Morse function with respect to $\mathbb{F}$, then Σ admits a homological splitting with respect to $\mathbb{F}$.*

Before presenting a proof of this theorem, we need to prove some preliminary results.
In order to prove that Σ admits a homological splitting with respect to $\mathbb{F}$, we have to provide two subcomplexes of Σ, generated by a partition of the set of top simplices of Σ, satisfying the equations described in Definition 6.16.
In the following, we will call f a perfect discrete Morse function with respect to $\mathbb{F}$ defined

on Σ, τ a critical d-simplex of Σ with respect to f, Σ_1 the simplicial complex $\Sigma \setminus \{\tau\}$ and Σ_2 the simplicial complex generated by τ. Furthermore, when this does not create ambiguity, we will omit the field $\mathbb{F}$.
To prove Theorem 6.36, we want to show that Σ_1, Σ_2 are two subcomplexes providing a homological splitting of the simplicial complex Σ. The information required to verify this claim are the assurance that Σ_1 is generated by the top simplices of Σ different from τ and the knowledge of the homology of Σ_1, Σ_2 and $\Sigma_1 \cap \Sigma_2$.

Lemma 6.37. *With the above notation, we have that*

$$\beta_k(\Sigma_1) = \begin{cases} \beta_k(\Sigma) - 1 & \text{if } k = d \\ \beta_k(\Sigma) & \text{otherwise} \end{cases}$$

Proof. Since f is a discrete Morse function on Σ, its restriction $f|_{\Sigma_1}$ to Σ_1 is still a discrete Morse function and its critical simplices are the same of f except for τ. So,

$$\begin{aligned} \beta_d(\Sigma_1) \leq \#C_d(f|_{\Sigma_1}) = & && \text{(weak Morse inequality)} \\ = \#C_d(f) - 1 & && (\Sigma_1 = \Sigma \setminus \{\tau\}) \\ = \beta_d(\Sigma) - 1 & && (f \text{ is perfect on } \Sigma) \end{aligned}$$

Then, by using Proposition 6.34, we reach the thesis. □

In order to prove that $\Sigma_1 = \Sigma \setminus \{\tau\}$ coincides with the simplicial complex generated by the top simplices of Σ different from τ, we need to show that τ is not on a simplex on the boundary (as topological space) of Σ.

Lemma 6.38. *With the above notation, there does not exist any $\sigma \in \mathrm{bd}_\Sigma \tau$ such that $\mathrm{cbd}_\Sigma \sigma = \{\tau\}$.*

Proof. Let us suppose that there exists in Σ such a simplex σ. Since $\mathrm{cbd}_\Sigma \sigma = \{\tau\}$, (σ, τ) represents a reduction pair whose removal preserves the homology. So, Σ and $\Sigma' := \Sigma \setminus \{\sigma, \tau\}$ are homologically equivalent.
On the other hand, we can consider Σ' as $\Sigma_1 \setminus \{\sigma\}$. So, by Proposition 6.34, we have that either

$$\beta_k(\Sigma') = \begin{cases} \beta_k(\Sigma_1) - 1 & \text{if } k = d-1 \\ \beta_k(\Sigma_1) & \text{otherwise} \end{cases} \quad \text{or} \quad \beta_k(\Sigma') = \begin{cases} \beta_k(\Sigma_1) + 1 & \text{if } k = d-2 \\ \beta_k(\Sigma_1) & \text{otherwise} \end{cases}$$

In both the cases, $\beta_d(\Sigma) = \beta_d(\Sigma') = \beta_d(\Sigma_1)$. But this is in contradiction with Lemma 6.37. □

As immediate consequence of Lemma 6.38, we obtain the following result.

Lemma 6.39. *With the above notation, Σ_1 coincides with the simplicial complex generated by the top simplices of Σ different from τ, i.e.,*

$$\Sigma \setminus \{\tau\} = < \tau' \mid \tau' \textit{ top simplex of } \Sigma, \tau' \neq \tau >$$

Proof. The inclusion "$\supseteq$" is obvious. Let us focus on the inclusion "$\subseteq$".
In order to verify that inclusion "$\subseteq$" holds, to show that each $\sigma \in \mathrm{bd}_\Sigma \tau$ also belongs to $< \tau' \,|\, \tau'$ top simplex of $\Sigma, \tau' \neq \tau >$ is enough. Lemma 6.38 ensures that, for each $\sigma \in \mathrm{bd}_\Sigma \tau$, $\#\,\mathrm{cbd}_\Sigma \sigma > 1$. So, there exists a top simplex $\tau' \neq \tau$ in Σ containing σ. □

Proof of Theorem 6.36

Proof. According to the above notation, we claim that the decomposition of Σ in $\Sigma_1 \cup \Sigma_2$ provides a homological splitting of Σ.
By Lemma 6.39, we have that the decomposition $\Sigma_1 \cup \Sigma_2$ is actually a partition of the set of the top simplices of Σ, i.e. a standard decompostion.
By Lemma 6.37, we have that

$$\beta_k(\Sigma_1) = \begin{cases} \beta_k(\Sigma) - 1 & \text{if } k = d \\ \beta_k(\Sigma) & \text{otherwise} \end{cases}$$

Trivially,

$$\beta_k(\Sigma_2) = \begin{cases} 1 & \text{if } k = 0 \\ 0 & \text{otherwise} \end{cases}$$

If $d = 0$, $\Sigma_1 \cap \Sigma_2$ is equal to the empty set, otherwise it coincides with the simplicial complex generated by the simplices in the boundary of τ and it is homeomorphic to the $(d-1)$-sphere $\mathbb{S}^{d-1}$. Therefore,

$$\beta_k(\Sigma_1 \cap \Sigma_2) = \begin{cases} 2 & \text{if } k = 0 \\ 0 & \text{otherwise} \end{cases} \text{ if } d = 1 \quad \beta_k(\Sigma_1 \cap \Sigma_2) = \begin{cases} 1 & \text{if } k = 0, d-1 \\ 0 & \text{otherwise} \end{cases} \text{ if } d > 1$$

It is easy to check that the above Betti numbers satisfy the equations described in Definition 6.16. Here, the explicit calculation.
Case $d = 0$.
$\beta_0(\Sigma_1) + \beta_0(\Sigma_2) = \beta_0(\Sigma) - 1 + 1 = \beta_0(\Sigma)$
Case $d = 1$.
$\beta_0(\Sigma_1) + \beta_0(\Sigma_2) - 1 = \beta_0(\Sigma) + 1 - 1 = \beta_0(\Sigma)$
$\beta_1(\Sigma_1) + \beta_1(\Sigma_2) + \beta_0(\Sigma_1 \cap \Sigma_2) - 1 = \beta_1(\Sigma) - 1 + 0 + 2 - 1 = \beta_1(\Sigma)$
Case $d > 1$.
$\beta_0(\Sigma_1) + \beta_0(\Sigma_2) - 1 = \beta_0(\Sigma) + 1 - 1 = \beta_0(\Sigma)$
$\beta_1(\Sigma_1) + \beta_1(\Sigma_2) + \beta_0(\Sigma_1 \cap \Sigma_2) - 1 = \beta_1(\Sigma) + 0 + 1 - 1 = \beta_1(\Sigma)$
$\beta_k(\Sigma_1) + \beta_k(\Sigma_2) + \beta_{k-1}(\Sigma_1 \cap \Sigma_2) - 1 = \beta_k(\Sigma) + 0 + 0 = \beta_k(\Sigma)$ if $1 < k < d$
$\beta_d(\Sigma_1) + \beta_d(\Sigma_2) + \beta_{d-1}(\Sigma_1 \cap \Sigma_2) - 1 = \beta_k(\Sigma) - 1 + 0 + 1 = \beta_d(\Sigma)$ □

Example 6.40. Theorem 6.36 can be applied to simplicial complex $\Sigma =< 123, 124, 134, 235, 245, 345 >$ depicted in Figure 6.1 (a). Considering the perfect discrete Morse function proposed in Example 6.21 and depicted in Figure 6.2, Theorem 6.36 ensures us that $< 123 > \cup < 124, 134, 235, 245, 345 >$ represents a homological splitting of Σ (as confirmed in Example 6.17).

Remark 6.41. In general, the converse of Theorem 6.36 is false. The simplicial complex $S_{18,125}$ considered in [LB13] and representing a triangularization of a 3-dimensional sphere is an example of this. Being $S_{18,125}$ a sphere, its top homology is non-null and it admits a homological splitting [Bol15]. In spite of this, $S_{18,125}$ does not admit any perfect discrete Morse function (see [LB13], Theorem 5.2).

The technical hypotheses required by Theorem 6.36 are satisfied by a large and interesting class of simplicial complexes: the orientable manifolds.

Corollary 6.42. *Let the simplicial complex Σ be an orientable d-manifold. If Σ admits a perfect discrete Morse function with respect to $\mathbb{F}$, then Σ admits a homological splitting with respect to $\mathbb{F}$.*

Proof. Each finite simplicial complex is compact. So, Σ satisfies the hypotheses of Poincaré duality Theorem 6.31. Since $\beta_0(\Sigma)$ is necessarily different from 0, by Theorem 6.31, we have that $\beta_d(\Sigma) = \beta_0(\Sigma) \neq 0$. By applying Theorem 6.36, we reach the thesis. □

Thanks to Remark 6.32, even if the orientability hypothesis in Corollary 6.42 is not satisfied, we can still reach the following result.

Corollary 6.43. *Let the simplicial complex Σ be a d-manifold. If Σ admits a perfect discrete Morse function with respect to $\mathbb{Z}_2$, then Σ admits a homological splitting with respect to $\mathbb{Z}_2$.*

6.3 Perfect discrete Morse functions and Betti splittings

By exploiting the results of the previous section, we describe here under which hypotheses the obtained homological splitting for Σ is also a Betti splitting. Furthermore, the reached result will be useful to prove that any 2-dimensional manifold admits a Betti splitting.

This section is mainly devoted to proving the following theorem.

Theorem 6.44. *Let the simplicial complex Σ be a manifold of dimension $d \leq 3$ such that $\beta_d(\Sigma; \mathbb{F}) \neq 0$. If Σ admits a perfect discrete Morse function with respect to $\mathbb{F}$, then Σ admits a Betti splitting with respect to $\mathbb{F}$.*

In order to prove Theorem 6.44, we need the following preliminary result.

Lemma 6.45. *Let Σ be a simplicial complex homeomorphic to the k-sphere $\mathbb{S}^k$ with $k \leq 2$. Given any k-simplex $\nu \in \Sigma$ and a field $\mathbb{F}$, there exists a perfect discrete Morse function with respect to $\mathbb{F}$ on Σ for which ν is a critical simplex.*

Proof. The simplicial complex $\Sigma \setminus \{\nu\}$ is homeomorphic to a k-disk and each k-disk, for $k \leq 2$, is collapsable [LB13]. □

Remark 6.46. Since, for each $k > 2$, there exist non-collapsible k-balls [LB13], the above lemma is not true in dimensions higher than 2.

Proof of Theorem 6.44

Proof. Let f be a perfect discrete Morse function with respect to $\mathbb{F}$ defined on Σ and let τ be a critical d-simplex of Σ with respect to f. Analogously to the proof of Theorem 6.36, we want to show that the decomposition of Σ in $\Sigma_1 = \Sigma \setminus \{\tau\}$ (also equal to $< \tau' \,|\, \tau'$ top simplex of $\Sigma, \tau' \neq \tau >$ by Lemma 6.39) and $\Sigma_2 =< \tau >$ provides a Betti splitting of Σ.
Thanks to Proposition 6.18, it is enough to prove that, for each $\sigma \in \Sigma_1 \cap \Sigma_2$, $\mathrm{link}_\Sigma \sigma = \mathrm{link}_{\Sigma_1} \sigma \cup \mathrm{link}_{\Sigma_2} \sigma$ is a homological splitting of $\mathrm{link}_\Sigma \sigma$.
If $\sigma = \emptyset$, then $\mathrm{link}_{\Sigma_1} \sigma = \Sigma_1$, $\mathrm{link}_{\Sigma_2} \sigma = \Sigma_2$ and $\mathrm{link}_\Sigma \sigma = \Sigma$. So, by Theorem 6.36, we reach the thesis.
Let k be the dimension of σ. Since the dimension of the manifold Σ is $d \leq 3$, Remark 6.29 ensures us that Σ is a combinatorial manifold. So, by Proposition 6.27, $\mathrm{link}_\Sigma \sigma$ is homeomorphic to the $(d-k-1)$-sphere $\mathbb{S}^{d-k-1}$. Furthermore, thanks to Lemma 6.45, $\mathrm{link}_\Sigma \sigma$ satisfies the hypotheses of Theorem 6.36 and, choosing any $(d-k-1)$-simplex ν in $\mathrm{link}_\Sigma \sigma$, there exists a perfect discrete Morse function with respect to $\mathbb{F}$ on $\mathrm{link}_\Sigma \sigma$ for which ν is a critical simplex. Let us set ν as the $(d-k-1)$-simplex of Σ such that $\mathrm{link}_{\Sigma_2} \sigma =< \nu >$ (i.e. ν is the simplex of Σ such that $\nu\sigma = \tau$). Then, by applying Theorem 6.36, $\mathrm{link}_\Sigma \sigma = \mathrm{link}_{\Sigma_1} \sigma \cup \mathrm{link}_{\Sigma_2} \sigma$ is a homological splitting of $\mathrm{link}_\Sigma \sigma$. □

Example 6.47. Similarly to the previous section, simplicial complex $\Sigma =< 123, 124, 134, 235, 245, 345 >$ represents an example in which Theorem 6.44 can be applied ensuring that the decomposition $< 123 > \cup < 124, 134, 235, 245, 345 >$ is a Betti splitting of Σ.

Remark 6.48. Since the 3-dimensional sphere $S_{18,125}$ admits a Betti splitting but not a perfect discrete Morse function, the converse of Theorem 6.44 does not hold in general.

Thanks to the Poincaré duality Theorem 6.31, the hypothesis $\beta_0(\Sigma) \neq 0$ in Theorem 6.44 is automatically satisfied if we consider an orientable manifold.

Corollary 6.49. *Let the simplicial complex Σ be an orientable manifold of dimension $d \leq 3$. If Σ admits a perfect discrete Morse function with respect to $\mathbb{F}$, then Σ admits a Betti splitting with respect to $\mathbb{F}$.*

Thanks to Remark 6.32, even if the orientability hypothesis in Corollary 6.49 is not satisfied, we can still reach the following result.

Corollary 6.50. *Let the simplicial complex Σ be a manifold of dimension $d \leq 3$. If Σ admits a perfect discrete Morse function with respect to $\mathbb{Z}_2$, then Σ admits a Betti splitting with respect to $\mathbb{Z}_2$.*

For some classes of simplicial complexes, the hypotheses required by the proven propositions are always satisfied. For example, Proposition 3.4 in [AFTV12] states that any compact connected surface admits a perfect discrete Morse function with respect to $\mathbb{Z}_2$. By combining this result with Corollary 6.50, the following proposition can easily deduced.

Proposition 6.51. *Any 2-manifold simplicial complex Σ admits a Betti splitting with respect to $\mathbb{Z}_2$.*

6.4 Concluding remarks

The main contribution of this chapter is the direct connection between the notions of splitting and of perfect discrete Morse function established by Theorem 6.36 and Theorem 6.44. Moreover, thanks to Proposition 6.51, the existence of a Betti splitting is ensured for any 2-manifold.

Theorem 6.36 plays a crucial role mainly in the proof of Theorem 6.44. In spite of this, we believe that Theorem 6.36 could become a fundamental theoretical tool for the development of an algorithmic test for checking if a simplicial complex admits or not a perfect discrete Morse function. Its statement ensures that if a simplicial complex Σ with non-null top homology does not admit any homological splitting, then Σ does not admit a perfect discrete Morse function too. So, if we algorithmically check that any possible decomposition of Σ does not lead to a homological splitting, then we have the assurance no perfect discrete Morse function can be built on Σ. Moreover, since the homological splitting decompositions returned by Theorem 6.36 are obtained through a removal of a single top simplex, the number of decompositions to be checked by the algorithm will be drastically reduced.

In the near future, our task is to understand if it is possible to generalize the presented results to higher dimensions and which hypotheses are really necessary to ensure the validity of the theses. Currently, our efforts are devoted to understand if the connection established by Theorem 6.36 and Theorem 6.44 still holds for simplicial complexes with vanishing top homology. In particular, we plan to firstly investigate the case of simplicial complexes having trivial homology, trying to understand if a collapsible simplicial complex necessarily admits a Betti splitting.

Concluding Remarks

In this thesis, we have investigated some tools in topological data analysis focusing our attention on the simplicial homology and the discrete Morse theory. The main contributions of this thesis are in these directions:

- an analysis and a comparison of different topological data structures for encoding simplicial complexes of arbitrary dimension [FID14, FIDon];
- a complete classification of various approaches and strategies to compute standard and persistent homology;
- a systematic overview on Morse theory and its discretization and on algorithms to compute Morse complexes [DFIM15, DFI15];
- a theoretical comparison between various methods to build a Forman gradient through homology-preserving operators [FID14, FIDon];
- the definition and the development of a compact encoding of the Forman gradient for a data structure based on top simplices [FID14, FIDon];
- the development and the implementation of an algorithm for the construction of a discrete Morse complex and the retrieval of standard and persistent homology based on coreduction operators and on a compact encoding of the Forman gradient and of the underlying simplicial complex [FID14, FIDon];
- a study of topological operators for modifying cell and simplicial complexes [ČomićDIF14];
- the formal definition of a general geometry-based multi-resolution model for cell complexes;
- the definition and the implementation of a multi-resolution model for cell complexes for the efficient retrieval of homology and the extraction of homology generators [ČomićDIF14];
- the definition of a multi-resolution model for simplicial complexes for the efficient retrieval of homology and the extraction of homology generators;
- the development of a new compact and efficient data structure for encoding discrete Morse complexes [IFD15];

- the development and the implementation of a topologically-consistent algorithm for the simplification of discrete Morse complexes [IFD15];
- a study of the connections between simplicial complexes and squarefree monomial ideals, proving that the satisfaction of suitable properties in discrete Morse theory for a simplicial complex ensures desirable algebraic decompositions of the corresponding ideal [BFRDon].

Thanks to the results obtained during the Ph.D. studies, we are planning to further study many of the topics addressed during the thesis work.

We can subdivide the planned research activity according to the task we want to achieve in:

- methods to improve standard and persistent homology computation;
- methods based on discrete Morse theory to represent and analyze scalar fields;
- investigation of the topological properties of a simplicial complex through an algebraic point of view.

Knowledge of **standard and persistent homology** allows us to describe the core topological information of a simplicial complex or of a sequence of them. Our analysis of the methods proposed in the literature has revealed several efficient approaches to retrieve homological information. In the near future, we will continue the study of the methods to compute homological information of a complex and in order to fill a gap in the literature, we plan to describe and classify them in a survey.

Currently, the Gudhi library [MBGY14] and the approach based on annotations [BDM13] are the state-of-the-art method for what concern performances and compactness in persistent homology computation. In spite of this, we believe that methods based on data structures encoding all the simplices of a simplicial complex will be affected by serious limitations when dealing with datasets of large size, huge dimension and built starting from a cloud of points. These limitations can be overcome by the use of compact data structures and the use of discrete Morse theory as a tool providing a compact and homology-equivalent model of the simplicial complex under investigation. The coreduction-based algorithm described in Chapter 3 represents a first attempt in this direction. In the next, we would like to further develop this strategy and to exploit it in various applications. A first development we plan to do is the implementation of a distributed version of the presented algorithm. This will be possible by the use of the Stellar Tree[Fel15], a compact topological data structure based on a spatial index, which stores only the vertices and the top simplices of the complex. This latter would not only reduce the storage cost further, but also allow an efficient localized computation of the homology, overcoming the limitation in this of the IA^* data structure. Improvements in the retrieval of persistent homology could be also obtained by performing a simplification step on the discrete Morse complex obtained during the execution of the coreduction-based algorithm. This result could be achieved by exploiting the topologically-consistent simplification algorithm introduced in Chapter 5.

Multi-dimensional persistent homology [CZ09, CSZ09] represents a generalization of the persistent homology allowing to describe the changes in homology of a simplicial complex parameterized along multiple scalar functions. As recently proposed in [AKL16, AKLM15], the computation of multi-dimensional persistent homology can be improved by exploiting discrete Morse theory. We expect that, after suitable modifications, the Morse-based algorithm developed in Chapter 3 could enhance the efficiency allowing a practical applicability of this method.

Multi-resolution models represent an interesting tool able to compactly represent and analyze a dataset at different levels of detail. In our work, we have combined for the first time the retrieval of homology with a multi-resolution model. Since the promising results obtained in the case of cell complexes, we plan to actually implement a hierarchical model for simplicial complexes. We believe that, analogously to the cellular case, this model will allow both a compact encoding of different representations of a simplicial complex and an efficient extraction of homology and of homology generators at any level of detail. In order to achieve this task, a dimension independent edge contraction is being implemented in the context of Stellar Tree.

The compact representation provided by **discrete Morse theory** has been recognized as a fundamental tool in both the study of the morphology of scalar fields and the topological analysis of simplicial complexes.

When working with data with scalar functions defined on high dimensional domains, such as 3D and time-varying scalar fields, a simplified substructure of the Morse-Smale complex, called extremum graph, has been proposed [CLB11]. The extremum graphs and their visual representations, the topological spines, can be extracted from the Forman gradient and provide the perfect framework for interactively studying subsets of the domain. We plan to suitably adapt the algorithms presented in Chapter 3 to quickly extract the extremum graph of a scalar field defined on a simplicial complex and to visually represent it through a topological spine.

Multi-resolution models represent another useful tool to enhance our knowledge of the morphology and the topology of a scalar field. In our plans, we aim to develop a multi-resolution model for discrete Morse complexes combining both geometrical and morphological modifications. This goal has been achieved in the context of 2-dimensional simplicial complex leading to the definition of a multi-resolution model for triangulated terrains [ID14]. The resolution of the inconsistency problem addressed in [GRSW13] and affecting the simplifications of discrete Morse complex of dimension greater than 2 leads to a good management of the morphological modifications and to an increase of the expressive power of a multi-resolution model. So, the topologically-consistent simplification algorithm proposed in Chapter 5 represents a first step in the definition of a dimension-independent version of such a multi-resolution model for discrete Morse complexes.

Further steps required to reach such goal are the development of simplification operators modifying the geometry of a simplicial complex without affecting the Forman gradient defined on it and the definition of a multi-resolution model based on geometrical and morphological simplification operators.
Typically, a simplification of a discrete Morse complex reduces its size, but leaves the

geometry of the underlying simplicial complex unchanged. In order to obtain a complete multi-resolution model, the possibility of simplifying discrete Morse complex also from a geometrical point of view is required. In this framework, our task will be to simplify the underlying geometrical structure of a complex by reducing its size through homology-preserving operators, while we maintain not only the homology of the complex but also the Forman gradient defined on it. Analogously to [ID14], we plan to consider as simplifying operator the edge contraction and to understand under which conditions the performance of this operator does not affect the morphology of the Forman gradient. Once this problem will be solved, we will combine these operators with morphology-modifying operators in order to design and implement a multi-resolution model for discrete Morse complexes based on both these kinds of modifications.

Another field on which we plan to keep on working is the study of the simplicial complexes through an **algebraic point of view**. The results obtained in Chapter 6 represent just a first step in this direction. We believe that an in-depth investigation of the bridge between topology and algebra established by the Stanley-Reisner correspondence could lead to create strong relations between topological and geometrical properties of simplicial complexes and algebraic and combinatorial properties of squarefree monomial ideals. In the short term, our task is to understand if it is possible to generalize the results obtained in Chapter 6 to higher dimensions and to relax their assumptions in order to prove more general propositions. Further, we would like to understand if the theorems proven could be useful for the development of an algorithmic test to check if a simplicial complex does not admit any perfect Morse function.

Appendix A

In this appendix, we present some notions and results useful for the thesis. These contributions have been discussed separately from the rest of the thesis in order to avoid to burden the reading.

Statement and proof of Proposition 4.14 (Subsection 4.3.3)

Proposition. *Let Γ be a d-dimensional cell complex, Γ' the cell complex obtained from Γ by applying $MiC(i+1)C(q,p,p')$. For a fixed $k \in \{0,\cdots,d\}$, let $B = \{[c_1]_\Gamma,\cdots,[c_l]_\Gamma\}$ be a basis for $H_k(\Gamma;\mathbb{Z}_2)$, then*

1) *if $k \neq i+1$, $[c_1]_{\Gamma'},\cdots,[c_l]_{\Gamma'}$ is a basis for $H_k(\Gamma';\mathbb{Z}_2)$;*
2) *if $k = i+1$, $B' = \{[c'_1]_{\Gamma'},\cdots,[c'_l]_{\Gamma'}\}$ is a basis for $H_{i+1}(\Gamma';\mathbb{Z}_2)$, where, if $[c]_\Gamma \in B$, $[c']_{\Gamma'} \in B'$ is defined by*

$$c' = \begin{cases} c & \text{if } \partial_{\Gamma'} c = 0 \\ c+q & \text{otherwise} \end{cases}$$

Proof. Throughout the proof and the statement we denote as ∂_Γ and $\partial_{\Gamma'}$ the boundary maps $\partial_\Gamma \otimes_{\mathbb{Z}} \mathbb{Z}_2$ and $\partial_{\Gamma'} \otimes_{\mathbb{Z}} \mathbb{Z}_2$ respectively and all calculations are to be considered modulo 2. We give the proof for the case when the refinement operator is of the type $expand$ ($MiC(i+1)C_{ex}(q,p,p')$). The case when the operator is of the type $insert$ ($MiC(i+1)C_{in}(q,p,p')$) is dual.

In order to prove that a set of elements of $C_k(\Gamma';\mathbb{Z}_2)$ is a basis for the $\mathbb{Z}_2$ vector space $H_k(\Gamma';\mathbb{Z}_2)$ we have to show that:

a) each element is in $Z_k(\Gamma';\mathbb{Z}_2)$;

b) each element is not in $B_k(\Gamma';\mathbb{Z}_2)$;

c) the elements are linearly independent.

1) The only non-trivial cases are for $k = i+2,\ i,\ i-1$.
Case $k = i+2$.

a) Let $c \in C_{i+2}(\Gamma;\mathbb{Z}_2)$ be such that $[c]_{\Gamma}$ is a basis element for $H_{i+2}(\Gamma;\mathbb{Z}_2)$. We can consider c as an element in $C_{i+2}(\Gamma';\mathbb{Z}_2)$. We have that

$$\partial_{\Gamma'}c = \partial_{\Gamma}c + mq = mq$$

where $m \in \{0,1\}$.
Suppose that $m = 1$, i.e., $\partial_{\Gamma'}c = q$. Then, since $\partial^2_{\Gamma'} = 0$,

$$\partial_{\Gamma'}q = \partial^2_{\Gamma'}c = 0$$

But, $\langle \partial_{\Gamma'}q, p\rangle = 1$, so

$$\partial_{\Gamma'}q = p + \cdots \neq 0 \quad \lightning$$

Hence, $c \in Z_{i+2}(\Gamma';\mathbb{Z}_2)$.

b) Suppose $\exists\, b \in C_{i+3}(\Gamma';\mathbb{Z}_2)$ such that $\partial_{\Gamma'}b = c$. Since no $(i+3)$-cell and no $(i+2)$-cell is created in Γ', we have that $\partial_{\Gamma',i+3} = \partial_{\Gamma,i+3}$, and $\partial_{\Gamma}b = \partial_{\Gamma'}b = c$. So, $c \in B_{i+2}(\Gamma;\mathbb{Z}_2)$. $\lightning$

c) $[c_1]_{\Gamma'}, \cdots, [c_l]_{\Gamma'}$ are linearly dependent in $H_{i+2}(\Gamma';\mathbb{Z}_2)$
$\Leftrightarrow$ for each $j = 1, \cdots, l$, $\exists\, \alpha_j \in \{0,1\}$ such that $[\alpha_1 c_1 + \cdots + \alpha_l c_l]_{\Gamma'} = [0]_{\Gamma'}$ and at least one $\alpha_j \neq 0$
$\Leftrightarrow \exists\, b \in C_{i+3}(\Gamma';\mathbb{Z}_2)$ such that $\partial_{\Gamma'}b = \sum_{j=1}^{l} \alpha_j c_j$
$\Leftrightarrow \exists\, b \in C_{i+3}(\Gamma;\mathbb{Z}_2)$ such that $\partial_{\Gamma}b = \sum_{j=1}^{l} \alpha_j c_j$
$\Leftrightarrow$ for each $j = 1, \cdots, l$, $\exists\, \alpha_j \in \{0,1\}$ such that $[\alpha_1 c_1 + \cdots + \alpha_l c_l]_{\Gamma} = [0]_{\Gamma}$ and at least one $\alpha_j \neq 0$
$\Leftrightarrow [c_1]_{\Gamma}, \cdots, [c_l]_{\Gamma}$ are linearly dependent in $H_{i+2}(\Gamma;\mathbb{Z}_2)$ $\lightning$

Case $k = i$.

a) Let $c \in C_i(\Gamma;\mathbb{Z}_2)$ such that $[c]_{\Gamma}$ is a basis element for $H_i(\Gamma;\mathbb{Z}_2)$. Since p does not appear in c, we have that

$$\partial_{\Gamma'}c = \partial_{\Gamma}c = 0$$

b) Suppose $\exists\, b \in C_{i+1}(\Gamma';\mathbb{Z}_2)$ such that $\partial_{\Gamma'}b = c$. There are two cases.
If q does not appear in b, then $b \in C_{i+1}(\Gamma;\mathbb{Z}_2)$ and

$$\partial_{\Gamma}b = \partial_{\Gamma'}b + \langle \partial_{\Gamma'}q, p'\rangle mp' = c + \langle \partial_{\Gamma'}q, p'\rangle mp'$$

where $m = \sum_{s \in \{r\ (i+1)\text{-cell in } \Gamma' \mid r \text{ is in } b\}} \langle \partial_{\Gamma'}s, p\rangle$ is the number of the $(i+1)$-cells in b in which p is incident in Γ'. Since $\partial_{\Gamma'}b = \cdots + mp$ and we know that $\partial_{\Gamma'}b = c$, then m has to be an even number and so $\partial_{\Gamma}b = c$. $\lightning$
If q appears in b, then $b - q \in C_{i+1}(\Gamma;\mathbb{Z}_2)$ and

$$\begin{aligned}\partial_{\Gamma}(b-q) &= c - \langle \partial_{\Gamma'}q, p'\rangle p' + \langle \partial_{\Gamma'}q, p'\rangle mp' \\ &= c + \langle \partial_{\Gamma'}q, p'\rangle mp'\end{aligned}$$

where $m = \sum_{s \in \{r\ (i+1)\text{-cell in } \Gamma' \mid r \neq q,\ r \text{ is in } b\}} \langle \partial_{\Gamma'}s, p\rangle$ is the number of the $(i+1)$-cells in $b-q$ in which p is incident in Γ'. Since $\partial_{\Gamma'}(b-q) = \cdots + mp$ and we know that $\partial_{\Gamma'}(b-q) = \partial_{\Gamma'}b - \partial_{\Gamma'}q = c + \cdots + p$, then m has to be an odd number and so $\partial_{\Gamma}(b-q) = c$. $\lightning$

c) Suppose that $[c_1]_{\Gamma'}, \cdots, [c_l]_{\Gamma'}$ are linearly dependent in $H_i(\Gamma'; \mathbb{Z}_2)$. This implies that, for each $j = 1, \cdots, l$, $\exists\, \alpha_j \in \{0, 1\}$ such that $[\alpha_1 c_1 + \cdots + \alpha_l c_l]_{\Gamma'} = [0]_{\Gamma'}$ and at least one $\alpha_j \neq 0$, i.e., $\exists\, b \in C_{i+1}(\Gamma'; \mathbb{Z}_2)$ such that $\partial_{\Gamma'} b = \sum_{j=1}^{l} \alpha_j c_j$.
With arguments analogous to those used in b), we can conclude that either $\partial_{\Gamma} b = \sum_{j=1}^{l} \alpha_j c_j$ or $\partial_{\Gamma}(b - q) = \sum_{j=1}^{l} \alpha_j c_j$ and so, $[c_1]_{\Gamma}, \cdots, [c_l]_{\Gamma}$ are linearly dependent in $H_i(\Gamma; \mathbb{Z}_2)$. $\lightning$

Case $k = i - 1$**.**

a) Let $c \in C_{i-1}(\Gamma; \mathbb{Z}_2)$ such that $[c]_{\Gamma}$ is a basis element for $H_{i-1}(\Gamma; \mathbb{Z}_2)$. We can consider c as an element in $C_{i-1}(\Gamma'; \mathbb{Z}_2)$. Since no $(i-1)$-cell and no $(i-2)$-cell is created in Γ', we have that $\partial_{\Gamma', i-1} = \partial_{\Gamma, i-1}$ and

$$\partial_{\Gamma'} c = \partial_{\Gamma} c = 0$$

b) Suppose $\exists\, b \in C_i(\Gamma'; \mathbb{Z}_2)$ such that $\partial_{\Gamma'} b = c$. For all i-cell $s \neq p$ of Γ', we have that

$$\partial_{\Gamma} s = \partial_{\Gamma'} s \qquad (*)$$

There are two cases.
If p does not appear in b, then, by $(*)$, we have that

$$\partial_{\Gamma} b = \partial_{\Gamma'} b = c \quad \lightning$$

Otherwise, if p appears in b, since $\partial^2_{\Gamma'} = 0$, we have that

$$0 = \partial^2_{\Gamma'} q = \partial_{\Gamma'}(p + (\partial_{\Gamma'} q - p)) = \partial_{\Gamma'} p + \partial_{\Gamma'}(\partial_{\Gamma'} q - p)$$

Hence,

$$\partial_{\Gamma'} p = \partial_{\Gamma'}(\partial_{\Gamma'} q - p) = \partial_{\Gamma}(\partial_{\Gamma'} q - p) \qquad (**)$$

Then, $b + \partial_{\Gamma'} q \in C_i(\Gamma; \mathbb{Z}_2)$ and

$$\begin{aligned}
\partial_{\Gamma}(b + \partial_{\Gamma'} q) &= \partial_{\Gamma}(b + p - p + \partial_{\Gamma'} q) \\
&= \partial_{\Gamma}(b + p) + \partial_{\Gamma}(\partial_{\Gamma'} q - p) \\
&= \partial_{\Gamma'} b + \partial_{\Gamma'} p + \partial_{\Gamma}(\partial_{\Gamma'} q - p) && \text{by } (*) \\
&= \partial_{\Gamma'} b + \partial_{\Gamma'} p + \partial_{\Gamma'} p && \text{by } (**) \\
&= c \quad \lightning
\end{aligned}$$

c) Analogous to the previous case.

2) Let's prove now the case $k = i + 1$. We have to show that if $B = \{[c_1]_{\Gamma}, \cdots, [c_l]_{\Gamma}\}$ is a basis for $H_{i+1}(\Gamma; \mathbb{Z}_2)$, then $B' = \{[c'_1]_{\Gamma'}, \cdots, [c'_l]_{\Gamma'}\}$ is a basis for $H_{i+1}(\Gamma'; \mathbb{Z}_2)$.

a) Let $c \in B$, i.e., $c \in C_{i+1}(\Gamma; \mathbb{Z}_2)$ such that $[c]_\Gamma$ is a basis element for $H_{i+1}(\Gamma; \mathbb{Z}_2)$. We want to prove that $\partial_{\Gamma'} c' = 0$.
If $\partial_{\Gamma'} c = 0$, then $c' = c$ and $\partial_{\Gamma'} c' = \partial_{\Gamma'} c = 0$.
Otherwise $\partial_{\Gamma'} c \neq 0$ and $c' = c + q$. In order to conclude, we want to show that $\partial_{\Gamma'} c = \partial_{\Gamma'} q$. This is indeed true, because we have that

$$\begin{aligned} \partial_{\Gamma'} c &= \partial_\Gamma c - m\langle \partial_{\Gamma'} q, p' \rangle p' + mp \\ &= m(\langle \partial_{\Gamma'} q, p' \rangle p' + p) \end{aligned}$$

where $m = \sum_{s \in \{r \text{ } (i+1)\text{-cell in } \Gamma' \mid r \text{ is in } c\}} \langle \partial_{\Gamma'} s, p \rangle$ is the number of the $(i+1)$-cells in c in which p is incident in Γ'. Since $\partial_{\Gamma'} c \neq 0$, m is an odd number, and

$$\partial_{\Gamma'} c = \langle \partial_{\Gamma'} q, p' \rangle p' + p = \partial_{\Gamma'} q$$

As a consequence,

$$\partial_{\Gamma'}(c') = \partial_{\Gamma'}(c + q) = \partial_{\Gamma'} c + \partial_{\Gamma'} q = 0$$

b) For each $(i+2)$-cell t in Γ', we have that $\partial_\Gamma t = \partial_{\Gamma'} t - \langle \partial_{\Gamma'} t, q \rangle q$. If $\exists\, b \in C_{i+2}(\Gamma'; \mathbb{Z}_2)$ such that $\partial_{\Gamma'} b = c'$, then

$$\partial_\Gamma b = \partial_{\Gamma'} b + mq = c' + mq$$

where $m = \sum_{t \in \{r \text{ } (i+2)\text{-cell in } \Gamma' \mid r \text{ is in } b\}} \langle \partial_{\Gamma'} t, q \rangle$.
Since m is even if $c' = c$ and m is odd if $c' = c + q$, we can conclude that

$$\partial_\Gamma b = c \quad \lightning$$

c) Analogous to the previous case.

□

List of Figures

List of Tables

Bibliography

[Ack87] D. H. Ackley. *A Connectionist Machine for Genetic Hillclimbing*. Kluwer Academic Publishers, Norwell, MA, USA, 1987.

[AFTV12] R. Ayala, D. Fernández-Ternero, and J. A. Vilches. Perfect discrete Morse functions on 2-complexes. *Pattern Recogn. Lett.*, 33(11):1495–1500, August 2012.

[Ago05] M. K. Agoston. *Computer Graphics and Geometric Modeling: Mathematics*. Springer Verlag London Ltd., 2005.

[AH35] P. Alexandroff and H. Hopf. *Topologie I*, volume 1035. Berlin, 1935.

[AKL16] M. Allili, T. Kaczynski, and C. Landi. Reducing complexes in multidimensional persistent homology theory. *Journal of Symbolic Computation*, 2016.

[AKLM15] M. Allili, T. Kaczynski, C. Landi, and F. Masoni. A new matching algorithm for multidimensional persistence. *CoRR*, abs/1511.05427v1, 2015.

[ALS11] D. Attali, A. Lieutier, and D. Salinas. Efficient data structure for representing and simplifying simplicial complexes in high dimensions. In *Proceedings of the 27th ACM Symposium on Computational Geometry, Paris, France, June 13-15, 2011*, pages 501–509, 2011.

[Art91] M. Artin. *Algebra*. Prentice Hall, 1991.

[BADSM08] M. Baba-Ali, G. Damiand, X. Skapin, and D. Marcheix. *Discrete Geometry for Computer Imagery: 14th IAPR International Conference, DGCI 2008, Lyon, France, April 16-18, 2008. Proceedings*, chapter Insertion and Expansion Operations for n-Dimensional Generalized Maps, pages 141–152. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[Ban67] T. Banchoff. Critical points and curvature for embedded polyhedra. *J. of Differential Geometry*, 1:245–256, 1967.

[Ban70] T. Banchoff. Critical points and curvature for embedded polyhedral surfaces. *American Mathematical Monthly*, 77(5):475–485, 1970.

[BCA+11] D. Boltcheva, D. Canino, S. Merino Aceituno, J.-C. Léon, L. De Floriani, and F. Hétroy. An iterative algorithm for homology computation on simplicial shapes. *Computer-Aided Design*, 43(11):1457 – 1467, 2011.

[BCC+12] O. Busaryev, S. Cabello, C. Chen, T. K. Dey, and Y. Wang. Annotating simplices with a homology basis and its applications. In *Algorithm Theory–SWAT 2012*, pages 189–200. Springer Verlag, 2012.

[BDF+08] S. Biasotti, L. De Floriani, B. Falcidieno, P. Frosini, D. Giorgi, C. Landi, L. Papaleo, and M. Spagnuolo. Describing shapes by geometrical-topological properties of real functions. *ACM Computing Surveys*, 40(4):Article 12, 2008.

[BDM13] J.-D. Boissonnat, T. K. Dey, and C. Maria. The compressed annotation matrix: An efficient data structure for computing persistent cohomology. In *Algorithms–ESA 2013*, pages 695–706. Springer Verlag, 2013.

[BDMZ12] P. Brendel, P. Dłotko, M. Mrozek, and N. Żelazna. Homology computations via acyclic subspace. In *Proceedings of the 4th international conference on Computational Topology in Image Context*, CTIC'12, pages 117–127, Berlin, Heidelberg, 2012. Springer Verlag.

[BEHP03] P.-T. Bremer, H. Edelsbrunner, B. Hamann, and V. Pascucci. A multi-resolution data structure for two-dimensional Morse functions. In *Proc. IEEE Visualization 2003*, pages 139–146. IEEE Computer Society, 2003.

[BEHP04] P.-T. Bremer, H. Edelsbrunner, B. Hamann, and V. Pascucci. A topological hierarchy for functions on triangulated surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 10(4):385–396, July/August 2004.

[BEK10] P. Bendich, H. Edelsbrunner, and M. Kerber. Computing robustness and persistence for images. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1251–1260, 2010.

[BFRDon] D. Bolognini, U. Fugacci, M. E. Rossi, and E. De Negri. Betti splittings for triangulated manifolds, In preparation.

[BHS80] I. C. Braid, R. C. Hillyard, and I. A. Stroud. Stepwise construction of polyhedra in geometric modelling. In K.W.Brodlie, editor, *Mathematical Methods in Computer Graphics and Design*, pages 123–141. Academic Press, 1980.

[BKR14a] U. Bauer, M. Kerber, and J. Reininghaus. Clear and compress: Computing persistent homology in chunks. In Peer-Timo Bremer, Ingrid Hotz, Valerio Pascucci, and Ronald Peikert, editors, *Topological Methods in Data Analysis and Visualization III*, Mathematics and Visualization, pages 103–117. Springer International Publishing, 2014.

[BKR14b] U. Bauer, M. Kerber, and J. Reininghaus. Distributed computation of persistent homology. In *ALENEX'14*, pages 31–38, 2014.

[BKRW14] U. Bauer, M. Kerber, J. Reininghaus, and H. Wagner. PHAT - Persistent Homology Algorithms Toolbox. In Hoon Hong and Chee Yap, editors,

Mathematical Software – ICMS 2014, volume 8592 of *Lecture Notes in Computer Science*, pages 137–143. Springer Berlin Heidelberg, 2014.

[BL14] B. Benedetti and F. H. Lutz. Random discrete Morse theory and a new library of triangulations. *Experimental Mathematics*, 23(1):66–94, 2014.

[BM12] J.-D. Boissonnat and C. Maria. The Simplex Tree: an efficient data structure for general simplicial complexes. In *Algorithms–ESA 2012*, pages 731–742. Springer, 2012.

[BM14] J.-D. Boissonnat and C. Maria. Computing persistent homology with various coefficient fields in a single pass. In Andreas S. Schulz and Dorothea Wagner, editors, *Algorithms - ESA 2014*, volume 8737 of *Lecture Notes in Computer Science*, pages 185–196. Springer Berlin Heidelberg, 2014.

[Bol15] D. Bolognini. *Betti splittings of monomial ideals and simplicial complexes*. PhD thesis, University of Genova, Italy, 2015.

[BPH05] P.-T. Bremer, V. Pascucci, and B. Hamann. Maximizing adaptivity in hierarchical topological models. In A.G. Belyaev, A.A. Pasko, and M. Spagnuolo, editors, *Proc. Int. Conf. on Shape Modeling and Applications 2005 (SMI '05)*, pages 300–309, Los Alamitos, California, 2005. IEEE Computer Society Press.

[BPS98] C. L. Bajaj, V. Pascucci, and D. R. Shikore. Visualization of scalar topology for structural enhancement. In *Proc. IEEE Visualization'98*, pages 51–58. IEEE Computer Society, 1998.

[BR03] S. Buoncristiano and C. Rourke. *Fragments of geometric topology from the sixties*. University of Warwick, Mathematics Institute, 2003.

[Can12a] D. Canino. The Mangrove TDS Library: a C++ tool for the fast prototyping of topological data structures, http://mangrovetds.sourceforge.net, 2012.

[Can12b] D. Canino. *Tools for Modeling and Analysis of Non-manifold shapes*. PhD thesis, University of Genova, Italy, 2012.

[CBK09] M. K. Chung, P. Bubenik, and P. T. Kim. Persistence diagrams of cortical surface data. In *Information Processing in Medical Imaging*, pages 386–397. Springer, 2009.

[CCL03] F. Cazals, F. Chazal, and T. Lewiner. Molecular shape analysis based upon the Morse-Smale complex and the Connolly function. In *Proc. 9th Annual Symposium on Computational Geometry*, pages 351–360, New York, USA, 2003. ACM Press.

[ČD12] L. Čomić and L. De Floriani. Topological operators on cell complexes in arbitrary dimensions. In *Computational Topology in Image Context*, pages 98–107. Springer Verlag, 2012.

[CD13] D. Canino and L. De Floriani. Representing simplicial complexes with Mangrove. *Proceedings of the 22nd Iinternational Meshing Roundtable*, pages 465–483, 2013.

[CDEG10] E. W. Chambers, V. De Silva, J. Erickson, and R. Ghrist. Vietoris-Rips complexes of planar point sets. *Discrete & Computational Geometry*, 44(1):75–90, 2010.

[ČDI12] L. Čomić, L. De Floriani, and F. Iuricich. Dimension-independent multi-resolution Morse complexes. *Computers & Graphics*, 36(5):541–547, 2012.

[CDI13] L. Comic, L. De Floriani, and F. Iuricich. Modeling three-dimensional Morse and Morse-Smale complexes. In *Innovations for Shape Analysis, Models and Algorithms*, pages 3–34. Springer, 2013.

[ČDMI14] L. Čomić, L. De Floriani, P. Magillo, and F. Iuricich. *Morphological Modeling of Terrains and Volume Data*. Springer Briefs in Computer Science. Springer, 2014.

[CDW11] D. Canino, L. De Floriani, and K. Weiss. IA*: An adjacency-based representation for non-manifold simplicial shapes in arbitrary dimensions. *Computers & Graphics*, 35(3):747–753, 2011.

[CF08] C. Chen and D. Freedman. Quantifying homology classes. In *Proceedings of the International Symposium on Theoretical Aspects of Computer Science (STACS)*, page 169–180, 2008.

[CFG06] A. Cerri, M. Ferri, and D. Giorgi. Retrieval of trademark images by means of size functions. *Graphical Models*, 68(5):451–471, 2006.

[CH13] C. J. Carstens and K. J. Horadam. Persistent homology of collaboration networks. *Mathematical Problems in Engineering*, 2013.

[CIDZ08] G. Carlsson, T. Ishkhanov, V. De Silva, and A. J. Zomorodian. On the local behavior of spaces of natural images. *International journal of computer vision*, 76(1):1–12, 2008.

[CK11] C. Chen and M. Kerber. Persistent homology computation with a twist. In *Proceedings 27th European Workshop on Computational Geometry*, 2011.

[CK13] C. Chen and M. Kerber. An output-sensitive algorithm for persistent homology. *Comput. Geom. Theory Appl.*, 46(4):435–447, May 2013.

[CLB11] C. Correa, P. Lindstrom, and Peer-Timo Bremer. Topological spines: A structure-preserving visual representation of scalar fields. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):1842–1851, 2011.

[ČomićD11] L. Čomić and L. De Floriani. Dimension-independent simplification and refinement of Morse complexes. *Graphical Models*, 73(5):261–285, Sep 2011.

[ČomićDI13] L. Čomić, L. De Floriani, and F. Iuricich. Simplification operators on a dimension-independent graph-based representation of Morse complexes. In C. L. Luengo Hendriks, G. Borgefors, and R. Strand, editors, *ISMM*, volume 7883 of *Lecture Notes in Computer Science*, pages 13–24. Springer, 2013.

[ČomićDIF14] L. Čomić, L. De Floriani, F. Iuricich, and U. Fugacci. Topological modifications and hierarchical representation of cell complexes in arbitrary dimensions. *Computer Vision and Image Understanding*, 121:2–12, 2014.

[Con86] M. L. Connolly. Measurement of protein surface shape by solid angles. *J. of Molecular Graphics*, 4(1):3 – 6, 1986.

[CSZ09] G. Carlsson, G. Singh, and A. J. Zomorodian. Computing multidimensional persistence. In Yingfei Dong, Ding-Zhu Du, and Oscar Ibarra, editors, *Algorithms and Computation*, volume 5878 of *Lecture Notes in Computer Science*, pages 730–739. Springer Berlin Heidelberg, 2009.

[CZ09] G. Carlsson and A. J. Zomorodian. The theory of multidimensional persistence. *Discrete & Computational Geometry*, 42(1):71–93, 2009.

[DAE+08] M.-L. Dequeant, S. Ahnert, H. Edelsbrunner, T. Fink, E. Glynn, G. Hattem, A. Kudlicki, Y. Mileyko, J. Morton, A. Mushegian, et al. Comparison of pattern detection methods in microarray time series of the segmentation clock. *PLoS One*, 3(8):e2856, 2008.

[dBT11] M. de Berg and C. Tsirogiannis. Exact and approximate computations of watersheds on triangulated terrains. In *Proc. 19th ACM SIGSPATIAL Int. Conf. on Advances in Geographic Information Systems*, pages 74–83, 2011.

[DC04] V. De Silva and G. Carlsson. Topological estimation using witness complexes. In *Proceedings of the First Eurographics Conference on Point-Based Graphics*, SPBG'04, pages 157–166, Aire-la-Ville, Switzerland, Switzerland, 2004. Eurographics Association.

[DDM+03a] E. Danovaro, L. De Floriani, P. Magillo, M. M. Mesmoudi, and E. Puppo. Morphology-driven simplification and multiresolution modeling of terrains. In E. Hoel and P. Rigaux, editors, *Proc. ACM GIS 2003 - The 11th Int. Symposium on Advances in Geographic Information Systems*, pages 63–70. ACM Press, 2003.

[DDM03b] E. Danovaro, L. De Floriani, and M. M. Mesmoudi. Topological analysis and characterization of discrete scalar fields. In T. Asano, R. Klette, and C. Ronse, editors, *Geometry, Morphology, and Computational Imaging*, volume 2616 of *Lecture Notes in Computer Science*, pages 386–402. Springer Verlag, 2003.

[DDM+06] E. Danovaro, L. De Floriani, P. Magillo, E. Puppo, and D. Sobrero. Level-of-detail for data analysis and exploration: A historical overview and some new perspectives. *Computers & Graphics*, 30(3):334–344, 2006.

[DDMP03] E. Danovaro, L. De Floriani, P. Magillo, and E. Puppo. Data structures for 3D multi-tessellations: an overview. In *Data Visualization: The State of the Art*, pages 239–256, 2003.

[De 03] V. De Silva. A weak definition of Delaunay triangulation. Technical report, Manuscript, Dept. Mathematics, 2003.

[DE93] C. J. Delfinado and H. Edelsbrunner. An incremental algorithm for Betti numbers of simplicial complexes, 1993.

[DE95] C. J. Delfinado and H. Edelsbrunner. An incremental algorithm for Betti numbers of simplicial complexes on the 3-sphere. *Computer Aided Geometric Design*, 12(7):771–784, 1995.

[DEGN99] T. K. Dey, H. Edelsbrunner, S. Guha, and D. V. Nekhayev. Topology preserving edge contraction. *Publ. Inst. Math.(Beograd)(NS)*, 66(80):23–45, 1999.

[DFI15] L. De Floriani, U. Fugacci, and F. Iuricich. Homological shape analysis through discrete Morse theory. In M. Breuß, A. Bruckstein, P. Maragos, and S. Wuhrer, editors, *Perspectives in Shape Analysis - Dagstuhl Seminar on New Perspectives in Shape Analysis*, Mathematics and Visualization. Springer Berlin Heidelberg, 2015.

[DFIM15] L. De Floriani, U. Fugacci, F. Iuricich, and P. Magillo. Morse complexes for shape segmentation and homological analysis: Discrete models and algorithms. *Computer Graphics Forum*, 2015.

[DFW12] T. K. Dey, F. Fan, and Y. Wang. Computing topological persistence for simplicial maps. *arXiv preprint arXiv:1208.5018*, 2012.

[DFW13] T. K. Dey, F. Fan, and Y. Wang. Graph induced complex on point data. In *Proceedings of the Twenty-ninth Annual Symposium on Computational Geometry*, SoCG '13, pages 107–116, New York, NY, USA, 2013. ACM.

[DG07a] V. De Silva and R. Ghrist. Coverage in sensor networks via persistent homology. *Algebraic & Geometric Topology*, 7(339-358):24, 2007.

[DG07b] V. De Silva and R. Ghrist. Homological sensor networks. *Notices of the American Mathematical Society*, 54, 2007.

[DGDP12] G. Damiand, R. Gonzalez-Diaz, and S. Peltier. Removal Operations in nD Generalized Maps for Efficient Homology Computation. In *Proc. of 4th International Workshop on Computational Topology in Image Context (CTIC)*, volume 7309 of *Lecture Notes in Computer Science*, pages 20–29. Springer Berlin/Heidelberg, 2012.

[DGH04] L. De Floriani, D. Greenfieldboyce, and A. Hui. A data structure for non-manifold simplicial d-complexes. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 83–92. ACM, 2004.

[DH05] L. De Floriani and A. Hui. Data structures for simplicial complexes: An analysis and a comparison. In Mathieu Desbrun and Helmut Pottmann, editors, *Proc. 3rd Eurographics Symposium on Geometry Processing*, volume 255 of *ACM Int. Conf. Proceeding Series*, pages 119–128, Aire-la-Ville, Switzerland, 2005. Eurographics Association.

[DH07] L. De Floriani and A. Hui. Shape representations based on simplicial and cell complexes. In *Eurographics 2007 - State of the Art Reports, Prague, Czech Republic, September 3-7, 2007*, pages 63–87, 2007.

[DHPC10] L. De Floriani, A. Hui, D. Panozzo, and D. Canino. A dimension-independent data structure for simplicial complexes. *Proceedings of the 19th International Meshing Roundtable*, pages 403–420, 2010.

[Dij59] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[DKMW11] P. Dłotko, T. Kaczynski, M. Mrozek, and T. Wanner. Coreduction homology algorithm for regular CW-complexes. *Discrete & Computational Geometry*, 46(2):361–388, 2011.

[DL03] G. Damiand and P. Lienhardt. Removal and contraction for n-dimensional generalized maps. In *DGCI*, volume 2886 of *Lecture Notes in Computer Science*, pages 408–419. Springer, 2003.

[DM02] L. De Floriani and P. Magillo. Multiresolution mesh representation: models and data structures. In *Tutorials on Multiresolution in Geometric Modelling*, pages 363–418. Springer-Verlag, 2002.

[DMMP03] L. De Floriani, M. M. Mesmoudi, F. Morando, and E. Puppo. Decomposing non-manifold objects in arbitrary dimensions. *Graphical Models*, 65(1):2–22, 2003.

[DMVJ11] V. De Silva, D. Morozov, and M. Vejdemo-Johansson. Dualities in persistent (co) homology. *Inverse Problems*, 27(12):124003, 2011.

[DSNW13] H. Doraiswamy, N. Shivashankar, V. Natarajan, and Y. Wang. Topological saliency. *Computers & Graphics*, 37(7):787–799, 2013.

[DW12] P. Dłotko and H. Wagner. Computing homology and persistent homology using iterated Morse decomposition. *arXiv preprint arXiv:1210.1429*, 2012.

[DW+14] P. Dłotko, H. Wagner, et al. Simplification of complexes for persistent homology computations. *Homology, Homotopy and Applications*, 16(1):49–63, 2014.

[Ede87] H. Edelsbrunner. *Algorithms in Combinatorial Geometry*. Springer Verlag, Berlin, 1987.

[EH08] H. Edelsbrunner and J. Harer. Persistent homology - a survey. *Contemporary mathematics*, 453:257–282, 2008.

[EH10] H. Edelsbrunner and J. Harer. *Computational topology: an introduction*. American Mathematical Soc., 2010.

[EHNP03] H. Edelsbrunner, J. Harer, V. Natarajan, and V. Pascucci. Morse-Smale complexes for piecewise linear 3-manifolds. In *Proc. 19th ACM Symposium on Computational Geometry*, pages 361–370, 2003.

[EHZ01] H. Edelsbrunner, J. Harer, and A. J. Zomorodian. Hierarchical Morse complexes for piecewise linear 2-manifolds. In *Proc. 17th ACM Symposium on Computational Geometry*, pages 70–79, 2001.

[Eis13] D. Eisenbud. *Commutative Algebra: with a view toward algebraic geometry*, volume 150. Springer Science & Business Media, 2013.

[EKS83] H. Edelsbrunner, D. G. Kirkpatrick, and R. Seidel. On the shape of a set of points in the plane. *IEEE Transactions on Information Theory*, 29(4):551–559, 1983.

[ELZ02] H. Edelsbrunner, D. Letscher, and A. J. Zomorodian. Topological persistence and simplification. *Discrete & Computational Geometry*, 28(4):511–533, 2002.

[ER98] J. A. Eagon and V. Reiner. Resolutions of Stanley-Reisner rings and Alexander duality. *Journal of Pure and Applied Algebra*, 130(3):265–275, 1998.

[EW79] C. M. Eastman and K. Weiler. Geometric Modeling Using the Euler Operators. In *1st Annual Conference on Computer Graphics in CAD/CAM Systems*, MIT, May 1979.

[Fel15] R. Fellegara. *A spatio-topological approach to the representation of simplicial complexes and beyond*. PhD thesis, University of Genova, Italy, 2015.

[FID14] U. Fugacci, F. Iuricich, and L. De Floriani. Efficient computation of simplicial homology through acyclic matching. In *Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), 2014 16th International Symposium on*, pages 587–593, Sept 2014.

[FIDon] U. Fugacci, F. Iuricich, and L. De Floriani. Computing discrete Morse complexes through reductions and coreductions, In preparation.

[FlDFW14] R. Fellegara, F. luricich, L. De Floriani, and K. Weiss. Efficient computation and simplification of discrete Morse decompositions on triangulated terrains. In *Proceedings of the 22Nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, SIGSPATIAL '14, pages 223–232, New York, NY, USA, 2014. ACM.

[For98] R. Forman. Morse theory for cell complexes. *Advances in Mathematics*, 134(1):90–145, 1998.

[For02] R. Forman. A user's guide to discrete Morse theory. *Sém. Lothar. Combin*, 48:35, 2002.

[FP99] P. Frosini and M. Pittore. New methods for reducing size graphs. *International journal of computer mathematics*, 70(3):505–517, 1999.

[Fug12] U. Fugacci. Constructive methods for the computation of simplicial homology. *Master Thesis*, 2012.

[GBHP08] 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.

[GBHP11] A. Gyulassy, P.-T. Bremer, B. Hamann, and V. Pascucci. Practical considerations in Morse-Smale complex computation. In V. Pascucci, X. Tricoche, H. Hagen, and J. Tierny, editors, *Topological Methods in Data Analysis and Visualization: Theory, Algorithms, and Applications*, Mathematics and Visualization, pages 67–78. Springer Verlag, Heidelberg, 2011.

[GBP12] A. Gyulassy, P.-T. Bremer, and V. Pascucci. Computing Morse-Smale complexes with accurate geometry. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2014–2022, 2012.

[GDN+07] A. Gyulassy, M. A. Duchaineau, V. Natarajan, V. Pascucci, E. Bringa, A. Higginbotham, and B. Hamann. Topologically clean distance fields. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1432–1439, 2007.

[Ghr08] R. Ghrist. Barcodes: the persistent topology of data. *Bulletin of the American Mathematical Society*, 45(1):61–75, 2008.

[Gie96] M. Giesbrecht. Probabilistic computation of the Smith Normal Form of a sparse integer matrix. In *Algorithmic Number Theory*, pages 173–186. Springer Verlag, 1996.

[GKK+12] A. Gyulassy, N. Kotava, M. Kim, C.D. Hansen, H. Hagen, and V. Pascucci. Direct feature visualization using Morse-Smale complexes. *IEEE Transactions on Visualization and Computer Graphics*, 18(9):1549–1562, Sept 2012.

[GNP+05] A. Gyulassy, V. Natarajan, V. Pascucci, P.-T. Bremer, and B. Hamann. Topology-based simplification for feature extraction from 3D scalar fields. In *Proc. IEEE Visualization'05*, pages 275–280. ACM Press, 2005.

[GNPH07] A. Gyulassy, V. Natarajan, V. Pascucci, and B. Hamann. Efficient computation of Morse-Smale complexes for three-dimensional scalar functions. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1440–1447, 2007.

[GO08] L. J. Guibas and S. Y. Oudot. Reconstruction using witness complexes. *Discrete & Computational geometry*, 40(3):325–356, 2008.

[Gom04] A. J. P. Gomes. Euler operators for stratified objects with incomplete boundaries. In *Proceedings of the ninth ACM symposium on Solid modeling and applications*, SM '04, pages 315–320, Los Alamitos, CA, USA, 2004. IEEE Computer Society.

[GRSW13] D. Günther, J. Reininghaus, H.-P. Seidel, and T. Weinkauf. Notes on the simplification of the Morse-Smale complex. In *Proc. TopoInVis*, Davis, U.S.A., March 2013.

[GRWH12] D. Günther, J. Reininghaus, H. Wagner, and I. Hotz. Efficient computation of 3D Morse-Smale complexes and persistent homology using discrete Morse theory. *The Visual Computer*, 28(10):959–969, 2012.

[GSW12] D. Günther, H.-P. Seidel, and T. Weinkauf. Extraction of dominant extremal structures in volumetric data using separatrix persistence. *Comput. Graph. Forum*, 31(8):2554–2566, 2012.

[Hat02] A. Hatcher. *Algebraic topology*. `https://www.math.cornell.edu/~hatcher/AT/ATpage.html`, 2002.

[Hil90] D. Hilbert. Ueber die theorie der algebraischen formen. *Mathematische Annalen*, 36(4):473–534, 1890.

[HM91] J. L. Hafner and K. S. McCurley. Asymptotically fast triangularization of matrices over rings. *SIAM Journal on Computing*, 20(6):1068–1083, 1991.

[HMM+10] S. Harker, K. Mischaikow, M. Mrozek, V. Nanda, H. Wagner, M. Juda, and P. Dłotko. The efficiency of a homology algorithm based on discrete Morse theory and coreductions. In *Proceedings 3rd International Workshop on Computational Topology in Image Context (CTIC 2010). Image A*, volume 1, pages 41–47, 2010.

[HMMN14] S. Harker, K. Mischaikow, M. Mrozek, and V. Nanda. Discrete Morse theoretic algorithms for computing homology of complexes and maps. *Foundations of Computational Mathematics*, 14(1):151–184, 2014.

[HMR09] D. Horak, S. Maletić, and M. Rajković. Persistent homology of complex networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2009(03):P03034, 2009.

[Hoc77] M. Hochster. Cohen-Macaulay rings, combinatorics, and simplicial complexes. In *Ring theory, II (Proc. Second Conf., Univ. Oklahoma, Norman, Okla., 1975)*, pages 171–223, 1977.

[Hud69] J. Hudson. Piecewise linear topology. *New York*, 1969.

[ID14] F. Iuricich and L. De Floriani. A combined geometrical and topological simplification hierarchy for terrain analysis. In *Proceedings of the 22Nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, SIGSPATIAL '14, pages 493–496, New York, NY, USA, 2014. ACM.

[IFD15] F. Iuricich, U. Fugacci, and L. De Floriani. Topologically-consistent simplification of discrete Morse complex. *Computers & Graphics*, 51:157 – 166, 2015. Honorable Mention at SMI 2015.

[Iur14] F. Iuricich. *Multi-resolution shape analysis based on discrete Morse decompositions*. PhD thesis, University of Genova, Italy, 2014.

[JM14] M. Juda and M. Mrozek. CAPD::redhom v2 - homology software based on reduction algorithms. In Hoon Hong and Chee Yap, editors, *Mathematical Software – ICMS 2014*, volume 8592 of *Lecture Notes in Computer Science*, pages 160–166. Springer Berlin Heidelberg, 2014.

[JP06] Michael Joswig and Marc E. Pfetsch. Computing optimal Morse matchings. *SIAM J. Discret. Math.*, 20(1):11–25, January 2006.

[KB79] R. Kannan and A. Bachem. Polynomial algorithms for computing the Smith and Hermite normal forms of an integer matrix. *SIAM Journal on Computing*, 8(4):499–507, 1979.

[KKM05] H. King, K. Knudson, and N. Mramor. Generating discrete Morse functions from point data. *Experimental Mathematics*, 14(4):435–444, 2005.

[KMM04] T. Kaczynski, K. Mischaikow, and M. Mrozek. *Computational homology*, volume 157. Springer Verlag, 2004.

[KMS98] T. Kaczynski, M. Mrozek, and M. Slusarek. Homology computation by reduction of chain complexes. *Computers & Mathematics with Applications*, 35(4):59–70, 1998.

[KSSM13] K. F. Kee, L. Sparks, D. C. Struppa, and M. Mannucci. Social groups, social media, and higher dimensional social structures: A simplicial model of social aggregation for computational communication research. *Communication Quarterly*, 61(1):35–58, 2013.

[LB13] F. H. Lutz and B. Benedetti. Knots in collapsible and non-collapsible balls. *Electr. J. Comb.*, 20(3):P31, 2013.

[LL01] S. H. Lee and K. Lee. Partial entity structure: a fast and compact non-manifold boundary representation based on partial topological entities. In *Proceedings Sixth ACM Symposium on Solid Modeling and Applications*, pages 159–170. Ann Arbor, Michigan, June 2001.

[LLT03] T. Lewiner, H. Lopes, and G. Tavares. Optimal discrete Morse functions for 2-manifolds. *Computational Geometry*, 26(3):221 – 233, 2003.

[LLT04] T. Lewiner, H. Lopes, and G. Tavares. Applications of Forman's discrete Morse theory to topology visualization and mesh compression. *IEEE Transactions on Visualization and Computer Graphics*, 10(5):499–508, 2004.

[LPT+03] H. Lopes, S. Pesco, G. Tavares, M. Maia, and A. Xavier. Handlebody Representation for Surfaces and Its Applications to Terrain Modeling. *International Journal of Shape Modeling*, 9(1):61–77, 2003.

[LSVJ11] D. Lipsky, P. Skraba, and M. Vejdemo-Johansson. A spectral sequence for parallelized persistence. *CoRR*, abs/1112.1245, 2011.

[LT97] H. Lopes and G. Tavares. Structural Operators for Modeling 3-Manifolds. In *Proceedings Fourth ACM Symposium on Solid Modeling and Applications*, pages 10–18. ACM Press, May 1997.

[LW69] A. T. Lundell and S. Weingram. *The topology of CW complexes*. Van Nostrand Reinhold Company, 1969.

[LZ14] R. H. Lewis and A. J. Zomorodian. Multicore homology via Mayer Vietoris. *CoRR*, abs/1407.2275, 2014.

[Man88] M. Mantyla. *An Introduction to Solid Modeling*. Computer Science Press, 1988.

[Mas91] W. S. Massey. *A basic course in algebraic topology*, volume 127. Springer Verlag New York, 1991.

[Mas93] H. Masuda. Topological Operators and Boolean Operations for Complex-Based Non-Manifold Geometric Models. *Computer-Aided Design*, 25(2):119–129, feb 1993.

[Mat02] Y. Matsumoto. *An Introduction to Morse Theory*, volume 208 of *Translations of Mathematical Monographs*. American Mathematical Society, 2002.

[MB90] F. Meyer and S. Beucher. Morphological segmentation. *J. of Visual Communication and Image Representation*, 1:21–45, 1990.

[MB09] M. Mrozek and B. Batko. Coreduction homology algorithm. *Discrete & Computational Geometry*, 41(1):96–118, 2009.

[MBGY14] C. Maria, J.-D. Boissonnat, M. Glisse, and M. Yvinec. The GUDHI library: simplicial complexes and persistent homology. In Hoon Hong and Chee Yap, editors, *Mathematical Software – ICMS 2014*, volume 8592 of *Lecture Notes in Computer Science*, pages 167–174. Springer Berlin Heidelberg, 2014.

[McC01] J. McCleary. *A user's guide to spectral sequences*. Cambridge University Press, 2001.

[MDD+09] P. Magillo, E. Danovaro, L. De Floriani, L. Papaleo, and M. Vitali. A discrete approach to compute terrain morphology. *Computer Vision and Computer Graphics Theory and Applications*, 21:13–26, 2009.

[MDI13] P. Magillo, L. De Floriani, and F. Iuricich. Morphologically-aware elimination of flat edges from a TIN. In *Proc. 21th ACM SIGSPATIAL Int. Conf. on Advances in Geographic Information Systems*, pages 244–253, 2013.

[Mey94] F. Meyer. Topographic distance and watershed lines. *Signal Processing*, 38:113–125, 1994.

[Mil63] J. Milnor. *Morse Theory*. Princeton University Press, New Jersey, 1963.

[MMS11] N. Milosavljević, D. Morozov, and P. Skraba. Zigzag persistent homology in matrix multiplication time. In *Proceedings of the twenty-seventh Annual Symposium on Computational Geometry*, pages 216–225. ACM, 2011.

[MN13] K. Mischaikow and V. Nanda. Morse theory for filtrations and efficient computation of persistent homology. *Discrete & Computational Geometry*, 50(2):330–353, 2013.

[MNV13] N. A. Murty, V. Natarajan, and S. Vadhiyar. Efficient homology computations on multicore and manycore systems. In *High Performance Computing (HiPC), 2013 20th International Conference on*, pages 333–342. IEEE, 2013.

[Mor12] D. Morozov. Dionysus library for computing persistent homology, 2012.

[MPZ08] M. Mrozek, P. Pilarczyk, and N. Żelazna. Homology algorithm based on acyclic subspace. *Comput. Math. Appl.*, 55(11):2395–2412, June 2008.

[MS82] M. Mantyla and R. Sulonen. Gwb: A Solid Modeler with Euler Operators. *IEEE Computer Graphics and Applications*, 2:17–31, 1982.

[MS05] E. Miller and B. Sturmfels. *Combinatorial commutative algebra*, volume 227. Springer Science & Business Media, 2005.

[MSNK89] H. Masuda, K. Shimada, M. Numao, and S. Kawabe. A Mathematical Theory and Applications of Non-Manifold Geometric Modeling. In *IFIP WG 5.2/GI International Symposium on Advanced Geometric Modeling*

for Engineering Applications, pages 89–103, Berlin, Germany, nov 1989. North-Holland.

[MSS06] M. A. Mannucci, L. Sparks, and D. C. Struppa. Simplicial models of social aggregation I. *arXiv preprint cs/0604090*, 2006.

[MTCW10] S. Martin, A. Thompson, E. A. Coutsias, and J.-P. Watson. Topology of cyclo-octane energy landscape. *Journal of Chemical Physics*, 132(23):234115, 2010.

[Mun84] J. R. Munkres. *Elements of Algebraic Topology*. Advanced book classics. Perseus Books, 1984.

[MW99] A. Mangan and R. Whitaker. Partitioning 3D surface meshes using watershed segmentation. *IEEE Transactions on Visualization and Computer Graphics*, 5(4):308–321, 1999.

[MW10] M. Mrozek and T. Wanner. Coreduction homology algorithm for inclusions and persistent homology. *Comput. Math. Appl.*, 60(10):2812–2833, November 2010.

[Nan] V. Nanda. The Perseus software project for rapid computation of persistent homology, http://www.math.rutgers.edu/ vidit/perseus/index.html.

[OPT+15] N. Otter, M. A. Porter, U. Tillmann, P. Grindrod, and H. A. Harrington. A roadmap for the computation of persistent homology. *ArXiv e-prints*, June 2015.

[Pas04] V. Pascucci. Topology diagrams of scalar fields in scientific visualization. In S. Rana, editor, *Topological Data Structures for Surfaces*, pages 121–129. John Wiley & Sons Ltd, 2004.

[PS03] S. Pemmaraju and S. Skiena. *Computational Discrete Mathematics: Combinatorics and Graph Theory with Mathematica*. Cambridge University Press, New York, NY, USA, 2003.

[Rei76] G. A. Reisner. Cohen-Macaulay quotients of polynomial rings. *Advances in Mathematics*, 21(1):30 – 49, 1976.

[RGH+12] J. Reininghaus, D. Günther, I. Hotz, T. Weinkauf, and H.-P. Seidel. Combinatorial gradient fields for 2D images with empirically convergent separatrices. *CoRR*, abs/1208.6523, 2012.

[RL14] B. Rieck and H. Leitte. Structural analysis of multivariate point clouds using simplicial chains. *Computer Graphics Forum*, 33(8):28–37, 2014.

[RM00] J. Roerdink and A. Meijster. The watershed transform: Definitions, algorithms, and parallelization strategies. *Fundamenta Informaticae*, 41:187–228, 2000.

[Rot70] J. J. Rotman. *Notes on homological algebra*. Van Nostrand Reinhold mathematical studies. Van Nostrand Reinhold, 1970.

[RS99] J. Rubio and F. Sergeraert. Constructive algebraic topology. *SIGSAM Bull*, pages 13–25, 1999.

[RS12] J. Rubio and F. Sergeraert. Constructive Homological Algebra and Applications. *ArXiv e-prints*, 2012.

[RWS11] V. Robins, P. J. Wood, and A. P. Sheppard. Theory and algorithms for constructing discrete Morse complexes from grayscale digital images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(8):1646–1658, 2011.

[Sam05] H. Samet. *Foundations of Multidimensional and Metric Data Structures (The Morgan Kaufmann Series in Computer Graphics and Geometric Modeling)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005.

[Sch05] B. Schneider. Extraction of hierarchical surface networks from bilinear surface patches. *Geographical Analysis*, 37(2):244–263, 2005.

[SCP08] T. Sousbie, S. Colombi, and C. Pichon. The fully connected n-dimensional skeleton: probing the evolution of the cosmic web. *CoRR*, abs/0809.2423, 2008.

[SG98] Oliver G. Staadt and Markus H. Gross. Progressive tetrahedralizations. In *Proceedings of the Conference on Visualization '98*, VIS '98, pages 397–402, Los Alamitos, CA, USA, 1998. IEEE Computer Society Press.

[SMN12] N. Shivashankar, S. Maadasamy, and V. Natarajan. Parallel computation of 2D Morse-Smale complexes. *IEEE Transactions on Visualization and Computer Graphics*, 18(10):1757–1770, 2012.

[SN12] N. Shivashankar and V. Natarajan. Parallel computation of 3D Morse-Smale complexes. *Computer Graphics Forum*, 31(3):965–974, 2012.

[Soi04] P. Soille. *Morphological Image Analysis: Principles and Applications*. Springer-Verlag, Berlin and New York, 2004.

[SS00] S. L. Stoev and W. Strasser. Extracting regions of interest applying a local watershed transformation. In *Proc. IEEE Visualization'00*, pages 21–28. ACM Press, 2000.

[Sta75] R. P. Stanley. Cohen-Macaulay rings and constructible polytopes. *Bull. Amer. Math. Soc.*, 81(1):133–135, 01 1975.

[Sto96] A. Storjohann. Near optimal algorithms for computing Smith normal forms of integer matrices. In *Proceedings of the 1996 international symposium on Symbolic and algebraic computation*, ISSAC '96, pages 267–274, New York, NY, USA, 1996. ACM.

[SW04] B. Schneider and J. Wood. Construction of metric surface networks from raster-based DEMs. In S. Rana, editor, *Topological Data Structures for Surfaces*, pages 53–70. John Wiley & Sons Ltd, 2004.

[Tau11] A. Tausz. PHOM: Persistent homology in R, Version 1.0. 1. 2011. *Available at CRAN http://cran. r-project. org*, 2011.

[TIKU95] S. Takahashi, T. Ikeda, T. L. Kunii, and M. Ueda. Algorithms for extracting correct critical points and constructing topological graphs from discrete geographic elevation data. In *Computer Graphics Forum*, volume 14, pages 181–192, 1995.

[TVJA12] A. Tausz, M. Vejdemo-Johansson, and H. Adams. javaPlex: a research platform for persistent homology. *Book of Abstracts Minisymposium on Publicly Available Geometric/Topological Software*, 7, 2012.

[VCY12] G. Vegter, A. Chattopadhyay, and C. K. Yap. Certified computation of planar Morse-Smale complexes. In *Proceedings of the Twenty-eighth Annual Symposium on Computational Geometry*, SoCG '12, pages 259–268, New York, NY, USA, 2012. ACM.

[vdWVE+11] R. van de Weygaert, G. Vegter, H. Edelsbrunner, B. Jones, P. Pranav, C. Park, W. A. Hellwing, B. Eldering, N. Kruithof, E. G. P. Bos, et al. Alpha, Betti and the megaparsec universe: On the topology of the cosmic web. In *Transactions on Computational Science XIV*, pages 60–101. Springer-Verlag, 2011.

[VJ12] M. Vejdemo-Johansson. GAP persistence - a computational topology package for GAP. *Book of Abstracts Minisymposium on Publicly Available Geometric/Topological Software*, 43, 2012.

[VS91] L. Vincent and P. Soille. Watershed in digital spaces: An efficient algorithm based on immersion simulation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(6):583–598, 1991.

[WAB+05] Y. Wang, P. K. Agarwal, P. Brown, H. Edelsbrunner, and J. Rudolph. Coarse and reliable geometric alignment for protein docking. In *In Proceedings of Pacific Symposium on Biocomputing*, volume 10, pages 65–75, 2005.

[WDFV11] K. Weiss, L. De Floriani, R. Fellegara, and M. Velloso. The PR-star octree: a spatio-topological data structure for tetrahedral meshes. In *GIS*, pages 92–101, 2011.

[WG09] T. Weinkauf and D. Günther. Separatrix persistence: Extraction of salient edges on surfaces using topological methods. *Comput. Graph. Forum*, 28(5):1519–1528, 2009.

[WIFD13] K. Weiss, F. Iuricich, R. Fellegara, and L. De Floriani. A primal/dual representation for discrete Morse complexes on tetrahedral meshes. *Computer Graphics Forum*, 32(3):361–370, 2013.

[ZC05] A. J. Zomorodian and G. Carlsson. Computing persistent homology. *Discrete & Computational Geometry*, 33(2):249–274, 2005.

[ZC08] A. J. Zomorodian and G. Carlsson. Localized homology. *Computational Geometry*, 41(3):126–148, 2008.

[Zom05] A. J. Zomorodian. *Topology for computing*. Cambridge University Press, 2005.

[Zom10a] A. J. Zomorodian. Fast construction of the Vietoris-Rips complex. *Computer and Graphics*, pages 263–271, 2010.

[Zom10b] A. J. Zomorodian. The Tidy Set: a minimal simplicial set for computing homology of clique complexes. In *Proceedings of the 2010 Annual Symposium on Computational Geometry*, pages 257–266. ACM, 2010.