Homology, Homotopy and Applications, vol. 16(1), 2014, pp.49–63

SIMPLIFICATION OF COMPLEXES FOR PERSISTENT HOMOLOGY COMPUTATIONS

PAWEŁ DŁOTKO AND HUBERT WAGNER

(*communicated by Graham Ellis*)

Abstract

In this paper we focus on preprocessing for persistent homology computations. We adapt some techniques that were successfully used for standard homology computations. The main idea is to reduce the complex prior to generating its boundary matrix, which is costly to store and process. We discuss the following reduction methods: elementary collapses, coreductions (as defined by Mrozek and Batko), and acyclic subspace methods (introduced by Mrozek, Pilarczyk, and Żelazna).

1. Introduction

Persistent homology is being applied to a wide range of different practical problems, ranging from sensor networks to root architecture analysis. Performance, however, still tends to be a problem. In the following paper efficient preprocessing algorithms are proposed. In the case of standard (i.e., nonpersistent) homology, the following preprocessing techniques were successfully used: elementary collapses by Whitehead [**17**], coreductions by Mrozek et al. [**12**] and the acyclic subspace method by Mrozek et al. [**13**]. The basic idea behind these techniques is to remove a number of cells, without affecting the homology (or affecting it in a controlled way). Importantly, these methods work on raw data; that is, before the boundary matrix is produced.[1]

On a practical note, the memory overhead of storing the boundary matrix is significant. For example, a 3-dimensional image of size 2000^3 voxels with 16b gray-scale values takes roughly 16 GB, while its boundary matrix takes roughly 200 GB. To handle data of such sizes, it is crucial to preprocess before generating the boundary matrix.

The goal of this paper is to extend the mentioned preprocessing methods (elementary collapses, coreductions, and acyclic subspace) to *persistent* homology. The techniques presented in this paper are heuristics, meaning there are no provable bounds

The first author is supported by federal contracts FA9550-12-1-0416 and FA9550-09-1-0643. The second author is supported by Foundation for Polish Science IPP Programme "Geometry and Topology in Physical Models."

Received April 30, 2013, revised September 16, 2013; published on February 26, 2014.

2010 Mathematics Subject Classification: 55-04, 55U05, 57M15, 68R05.

Key words and phrases: homology, persistence, reduction algorithm.

Article available at `http://dx.doi.org/10.4310/HHA.2014.v16.n1.a3`

[1]Of course, we assume that the complexes are stored in some combinatorial way, not as a boundary matrix. This is the case with the output of most (simplicial or cubical) mesh generators.

on how many cells are reduced. They have, however, performed well in practical situations as described in Mischaikow et al. [**8**] and Mrozek et al. [**12, 13**]. The computational complexity of the presented techniques is linear in the number of cells, provided the number of neighbors of every cell in the complex is $O(1)$. Otherwise the complexity of the algorithms is $O(np)$ where n is the number of cells in the complex and p is average number of neighbors of a single cell. Therefore they can be used as an inexpensive preprocessing step.

2. Background

2.1. Chain complexes and homology

The presented methods work for arbitrary *chain complexes* with field coefficients. In the most typical case this chain complex comes from a CW-decomposition of a given space. In practice, simplicial and cubical complexes are used. We fix a field $\mathbb{F}$ of coefficients throughout the paper. We remind the reader that persistence intervals are defined only for field coefficients. One can, for instance, think of $\mathbb{F} = \mathbb{Z}_2$.

Let us fix a chain complex $\mathcal{K}$. Let a *p-chain* be a formal sum of p-cells with the $\mathbb{F}$ coefficients. The p-chains of $\mathcal{K}$, together with addition in $\mathbb{F}$, form a *group of p-chains*, denoted by $C_p(\mathcal{K})$. The boundary operator ∂_p maps p-chains into $(p-1)$-chains, called *faces*. The definition of chain complexes requires that $\partial_p \circ \partial_{p+1} = 0$.

The chain of (co-)faces is called a (co-)boundary. For any p-chain $c = \sum a_i c_i$, we have $\partial_p c = \sum a_i \partial_p c_i$. If a $(p-1)$-cell a has a p-cell B in its coboundary, we say a is a face of B, and B is a coface of a. (Notation: capital letters denote higher dimensional cells where a cell and its face is considered). Let us have a set $A \subset \mathcal{K}$. By the boundary of A we mean $bd\ A = \{b \in \mathcal{K} \mid \exists_{a \in \mathcal{A}} b \in \partial a\}$. We say that A is closed if $bd\ A \subset A$.

To define homology, let us first introduce the group of p-cycles, $Z_p(\mathcal{K}) = \ker \partial_p$ and its subgroup, the group of p-boundaries, $B_p(\mathcal{K}) = \operatorname{im} \partial_{p+1}$. The p-th homology group is the quotient $H_p(\mathcal{K}) = Z_p(\mathcal{K})/B_p(\mathcal{K})$. The p-th Betti number, denoted by $\beta_p(\mathcal{K})$, is the rank of this group.

2.2. Filtrations and persistence

Let us have a finite complex $\mathcal{K}$. By a *filtration* of $\mathcal{K}$ we mean a sequence of subcomplexes:

$$\emptyset \subset \mathcal{K}_0 \subset \mathcal{K}_1 \subset \cdots \subset \mathcal{K}_n = \mathcal{K}$$

Usually in the persistent homology community, the filtration is induced by a filtering function $g\colon \mathcal{K} \to \mathbb{N}$ (note that having function values in $\mathbb{N}$ does not limit the generality of the presented results). We require that $g(a) \leqslant g(B)$ whenever a is a face of B, which implies that for every $t \in \mathbb{N}$, $\mathcal{K}_t$ forms a complex; i.e., the boundary of every cell in $\mathcal{K}_t$ is contained in $\mathcal{K}_t$. We say that g is *filtration by maxima* if for every $A \in \mathcal{K}$, $g(A) = max\{g(b_1), \ldots, g(b_n)\}$, where $b_1, \ldots, b_n$ are the faces of A. Such a filtration is often used when the function values are given only on the vertices and have to be extended to higher-dimensional cells. All the presented methods works for general filtrations, but filtration by maxima is convenient for examples.

Note that once a filtration $\emptyset \subset \mathcal{K}_0 \subset \mathcal{K}_1 \subset \cdots \subset \mathcal{K}_n = \mathcal{K}$ of a complex is given, we can naturally associate with it a filtering function $g\colon \mathcal{K} \to \mathbb{N}$ by setting $g(A) =$

$\min\{i \in \{1, \ldots, n\} \mid A \in \mathcal{K}_i\}$. Such a function naturally associated with filtration will be used in the algorithms presented in this paper.

Persistent homology captures the birth and death times of homology classes of the sublevel complexes $\mathcal{K}_t$, as t grows from 0 to $+\infty$. By birth, we mean that a homology class is created; by death, we mean it either becomes trivial or becomes identical to some other class born earlier. The *persistence*, or lifetime of a class, is the difference between the death and birth times. Often a multiset of *persistence intervals* is used to represent persistence in a given dimension. A single interval encodes the lifetime of a homology class. We say that two spaces have the same persistence if their persistence intervals are the same in the corresponding dimensions. We disregard zero-length persistence intervals, because they carry virtually no information.

The formal definition is as follows (after Edelsbrunner et al. [**5**]): The p-th persistent homology groups of filtered complex $\mathcal{K}$ are the images of the homomorphisms induced by inclusion, $H^{i,j}(\mathcal{K}) = \operatorname{im} H(f^{i,j})$, where $f^{i,j} \colon \mathcal{K}_i \hookrightarrow \mathcal{K}_j$ for every $i, j \in \mathbb{N}$, $i \leqslant j$. By $H^p(\mathcal{K})$ we denote the persistent homology of a filtered cell complex $\mathcal{K}$. So-called *persistence diagrams* encode the complete information about the persistence of a filtered complex. For more details see Edelsbrunner et al. [**5**].

We want to reiterate a theorem saying when the persistence of two filtered complexes are equal.

Theorem 2.1 (Persistence equivalence theorem, Edelsbrunner et al. [**5**])**.** *Consider the persistent homology of two filtered complexes X and Y. Let $\phi_i \colon H_*(X_i) \to H_*(Y_i)$:*

$$\begin{tikzcd}
\cdots H_*(X_0) \arrow[r] \arrow[d, "\phi_0"'] & H_*(X_1) \arrow[r] \arrow[d, "\phi_1"'] & \cdots \arrow[r] & H_*(X_{n-1}) \arrow[r] \arrow[d, "\phi_{n-1}"'] & H_*(X_n) \cdots \arrow[d, "\phi_n"'] \\
\cdots H_*(Y_0) \arrow[r] & H_*(Y_1) \arrow[r] & \cdots \arrow[r] & H_*(Y_{n-1}) \arrow[r] & H_*(Y_n) \cdots
\end{tikzcd}$$

If the ϕ_i are isomorphisms and all the squares commute, then the persistent homology of X and Y is the same.

The standard algorithm to compute persistent homology is a matrix reduction algorithm presented in Edelsbrunner et al. [**5**]. The other algorithms to compute persistence are discussed in the Section 2.3. The output of the persistent homology computations is a list of *persistence pairs* of the form (birth, death).

An important justification for the usage of persistence is the stability theorem. Cohen-Steiner et al. [**3**] proved that for any two filtering functions f and g, the so-called bottleneck distance (d_B, see [**3**]) between the persistence of $\mathcal{K}$ with respect to f (denoted as $H^p(\mathcal{K}, f)$) and the persistence of $\mathcal{K}$ with respect to g ($H^p(\mathcal{K}, g)$) is upper bounded by the L^∞ norm of the difference between f and g:

$$d_B(H^p(\mathcal{K}, f), H^p(\mathcal{K}, g)) \leqslant \|f - g\|_\infty := \max_{x \in \mathcal{K}} |f(x) - g(x)|. \tag{1}$$

Simply put, small changes in the filtration values cause small changes in persistence. This enables robust estimation of how persistence is affected by perturbation of the input (e.g., noise).

2.3. Existing algorithms and their complexity

Applying the matrix-reduction algorithm described in Edelsbrunner et al. [**5**] to the boundary matrix of the input complex is the standard way to compute persistent homology groups. It works for general complexes in arbitrary dimensions. The worst-case complexity is $O(n^3)$, where n is the size of the input complex. Milosavljevic et al. [**10**] showed that persistent homology can be computed in matrix multiplication time $O(n^\omega)$, where the currently best estimation of ω is 2.373. Chen et al. [**1**] proposed a randomized algorithm to compute only pairs with persistence above a chosen threshold. Despite improving the theoretical complexity, it is unclear whether these methods are better in practice.

When focusing on 0-dimensional homology, union–find data structures can be used to compute 0-dimensional persistence in time $O(n\alpha(n))$, as discussed in Edelsbrunner et al. [**5**], where α is the inverse of the Ackermann function and n the size of the input.

A recent variation of the standard algorithm, introduced by Chen et al. [**2**], significantly reduces the amount of computation. This idea was also used by Wagner et al. [**16**] to compute persistence for n-dimensional images.

One can also compute persistence by computing homology of the inclusion map, as discussed in Mrozek et al. [**14**] and Żelazna [**18**]. This approach gives even more complete information than persistence. It is also very time-consuming, as noted in Wagner et al. [**16**].

Discrete Morse theory was also used by Ellis et al. [**6**], in conjunction with so-called zig-zag homotopy retractions, to compute homology, persistence, and fundamental groups.

Another class of methods involves preprocessing the input complex using discrete Morse theory, as proposed by Robins et al. [**15**]. Such a preprocessing significantly reduces the size of the boundary matrix, while preserving persistence. In the case of 3D grayscale images, an efficient parallel implementation was proposed in Günther et al. [**7**], allowing for handling large ($\approx 1200^3$) images on commodity hardware. The standard matrix-reduction algorithm is used in the final step of computation.

The approach by Robins et al. works for arbitrary complexes and in dimension three the preprocessing results in the smallest possible boundary matrix [**15**] (counting the number of rows/columns). The algorithm used in Robins et al. [**15**] depends crucially on simple-homotopy theory, which makes it hard to directly generalize the optimality result to higher dimensions. A recent paper by Mischaikow et al. [**11**] proposes a handy theoretical framework, where discrete Morse theory is extended from complexes to filtrations.

It should be noted that the simplification methods based on discrete Morse theory aim at optimizing the number of cells. The total size of the resulting complex can in fact grow, since the number of connections can grow even quadratically in the number of cells. In other words, even if the initial boundary matrix is sparse, the matrix of the simplified complex can be dense. The methods presented in this paper do not have this property. Also, they can be used prior to persistence computations with discrete Morse theory.

3. Elementary collapses

Elementary collapse is an old concept introduced by Whitehead [**17**] in the context of simple homotopy types. It is often used in the context of homology theory. It aims in finding a pair of cells $(A, b) \in \mathcal{K} \times \mathcal{K}$ (called elementary collapse pair) such that b has only one coface A in $\mathcal{K}$ (in this case b is called a *free face*). Removing such a pair does not change the homology or homotopy type of $\mathcal{K}$, since such a removal corresponds to a deformation retraction.

In this section we show that the elementary collapses can also be used in the case of persistent homology. However, the elementary collapse pair (A, b) must fulfill two extra assumptions:

1. b is a free face for every $\mathcal{K}_n$, $n \in \mathbb{N}$;
2. $g(A) = g(b)$.

The first condition indicates that at every filtration level (A, b) is a elementary collapse pair. It suffices to check the size of the coboundary of b in the final complex in the filtration. The second condition indicates that filtration levels of A and b need to be equal. In Figure 1 we show that these are in fact necessary conditions in order to preserve persistent homology information.

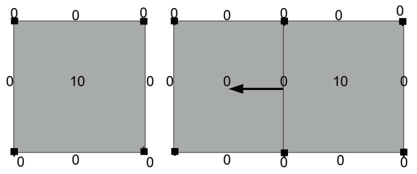

Figure 1: On the left the complex consisting of a single cube is depicted. All the vertices and edges have filtration level 0, while the two-dimensional cube has filtration level 10. Consequently, in the case of one-dimensional persistent homology an interval $[0, 10]$ appears. However, when any elementary collapse is performed, the interval does not appear anymore. Therefore the assumption $g(A) = g(b)$ is necessary. In the right picture, if an elementary collapse is performed at level 0 as indicated by the arrow, at sublevel 10 we are missing the shared edge and the 2-cell having value 0, so the homology (and consequently persistence) is changed.

Let us now prove that the presented extra assumptions guarantee that after removing a pair (A, b), persistent homology of a complex $\mathcal{K}$ does not change.

Theorem 3.1. *Let $(A, b) \in \mathcal{K} \times \mathcal{K}$ be a elementary collapse pair and b be a free face in $\mathcal{K}_n$ for every $n \in \mathbb{N}$. Moreover let $g(A) = g(b)$. Then $H^p_*(\mathcal{K}) = H^p_*(\mathcal{K} \setminus \{A, b\})$.*

Proof. The proof is based on Theorem 2.1. Let us consider maps in homology induced by inclusions $\mathcal{K}_1 \subset \mathcal{K}_2 \subset \cdots \subset \mathcal{K}_n$ before and after removal of a pair (A, b):

$$\begin{tikzcd}
\cdots \arrow[r, "H(i_{l-1})"] \arrow[d] & H_*(\mathcal{K}_l) \arrow[r, "H(i_l)"] \arrow[d, "H(j_l)"] & H_*(\mathcal{K}_{l+1}) \arrow[r, "H(i_{l+1})"] \arrow[d, "H(j_{l+1})"] & \cdots \arrow[d] \\
\cdots \arrow[r, "H(k_{l-1})"] & H_*(\mathcal{K}_l \setminus \{A, b\}) \arrow[r, "H(k_l)"] & H_*(\mathcal{K}_{l+1} \setminus \{A, b\}) \arrow[r, "H(k_{l+1})"] & \cdots
\end{tikzcd}$$

The horizontal maps are the maps induced in homology by the inclusion map

$i_l\colon \mathcal{K}_l \hookrightarrow \mathcal{K}_{l+1}$ and $k_l\colon \mathcal{K}_l \setminus \{A, b\} \hookrightarrow \mathcal{K}_{l+1} \setminus \{A, b\}$. The vertical maps $j_l\colon H_*(\mathcal{K}_l) \to H_*(\mathcal{K}_l \setminus \{A, b\})$ are isomorphisms, since a pair (A, b) is an elementary collapse pair in every $\mathcal{K}_l$ and $g(A) = g(b)$.

To show that all squares commute, let us pick a chain $c \in C(\mathcal{K}_l)$. Then $j_l(c) = c - \langle c, A\rangle A - \langle c, b\rangle b$; i.e., the generators A and b are removed from the chain c. We see that $k_l(j_l(c))$ is simply $j_l(c) \in C(\mathcal{K}_{l+1} \setminus \{A, b\})$. Let us now compute $j_{l+1}(i_l(c))$. We have $i_l(c) = c \in C(\mathcal{K}_{l+1})$. To obtain $j_{l+1}(i_l(c))$ it again suffices to remove from $i_l(c)$ the generators A and b, which is $j_{l+1}(i_l(c)) = c - \langle c, A\rangle A - \langle c, b\rangle b$. Therefore $j_{l+1}(i_l(c)) = k_l(j_l(c))$, so all the squares commute.

Therefore, due to Theorem 2.1, persistence of the lower and upper filtered complexes are equal, which completes the proof. □

It might seem that there could be just a few elementary collapse pairs to remove in the external boundary of the considered complex. However, the reduction process usually creates new collapse pairs and it can be continued.

The idea of the described procedure is summarized in Algorithm 1. We want to point out that Algorithm 1 and Algorithm 2 (presented in the next section) have a very similar general structure. First of all the whole complex $\mathcal{K}$ is searched for an initial pairs to be removed. All those pairs are put into a queue, which is later processed as long as it remains nonempty. During the process some new candidates for pairs may be added to this queue.

Algorithm 1 Elementary collapses for persistence.

```
Input: $\mathcal{K}$ - cell complex with filtration $g : \mathcal{K} \to \mathbb{N}$;
Output: Reduced complex $\mathcal{K}'$ with the same persistence as $\mathcal{K}$.
  $\mathcal{K}' = \mathcal{K}$
  List of cells Q= $\emptyset$;
  for every cell $b \in \mathcal{K}'$ do
    if $b$ has unique cell $A$ in coboundary and $g(b) = g(A)$ then
      Q.$enqueue(b)$;
  while Q is not empty do
    $b$ = Q.$dequeue()$;
    if $b$ has unique cell $A$ in coboundary and $b, A \in \mathcal{K}'$ and $g(b) = g(A)$ then
      $\mathcal{K}' = \mathcal{K}' \setminus \{A, b\}$;
      for every element $c \in \mathcal{K}'$ in boundary of $A$ or boundary of $b$ do
        if $c$ has unique cell $D \in \mathcal{K}'$ in coboundary and $g(c) = g(D)$ then
          Q.$enqueue(c)$;
```

4. Coreductions

The concept of coreductions was introduced by Mrozek et al. in [**12**]. It is based on the idea to search for a homologically trivial subcomplex in the given complex, starting from lowest possible dimension (bottom-up). The approach to find acyclic subcomplexes that uses only top-dimensional cells is shown in Section 5. The formal presentation requires the concept of S-complex, which is a chain complex with a fixed basis. This concept is not recalled here for the sake of brevity. For further details consult Mrozek et al. [**12**].

A pair of cells (A, b) is a *coreduction pair* if b is the unique element in the boundary of A. Such a pair cannot exist in a simplicial or cubical complex. Therefore in [**12**] first a single vertex is removed from each connected component of the considered complex. This removal changes only the zero-dimensional homology. Now, the process of coreduction is iterated as long as one can find a coreduction pair in the considered complex.

The coreduction algorithm was already used to compute homology of inclusions and persistent homology in Mrozek et al. [**14**]. However the approach there was different: in [**14**] the coreduction was used to remove as many elements as possible before computing the map in homology induced by the inclusion map between levelsets. To be precise: at each level of filtration the homology generators were computed and then, by solving a system of linear equations, one obtained the matrix mapping generators at the nth level to generators at the $(n+1)$st level. This procedure is more general than computing persistence. As indicated in Wagner et al. [**16**], this strategy is in general not efficient for computing only persistence. Here we use a coreduction algorithm as preprocessing for the standard algorithm to compute persistence, as presented in Edelsbrunner et al. [**5**].

First we show that removing a coreduction pair (A, b) such that $g(A) = g(b)$ does not change the persistent homology of the filtered complex $\mathcal{K}$. Then we show that removing an initial vertex from the complex changes only 0-dimensional persistence.

Since Corollary 4.2. from Mrozek et al. [**12**] is extensively used in this section, we present it here.

Theorem 4.1 (Corollary 4.2 [**12**]). *Let $\mathcal{K}$ be a chain complex (without filtration). If (A, b) is a coreduction pair in $\mathcal{K}$, then $H(\mathcal{K})$ and $H(\mathcal{K} \setminus \{A, b\})$ are isomorphic.*

Now we are ready to present the main theorem of this section, analogous to Threorem 3.1:

Theorem 4.2. *Let $(A, b) \in \mathcal{K} \times \mathcal{K}$ be a coreduction pair. Moreover let $g(A) = g(b)$. Then $H^p_*(\mathcal{K}) = H^p_*(\mathcal{K} \setminus \{A, b\})$.*

Proof. First we point out that if (A, b) is a coreduction pair and $g(A) = g(b)$, then it is a coreduction pair in every filtration level in which A and b exist. Let us consider the filtered complex $\mathcal{K}$ before and after removal of a coreduction pair (A, b):

$$\begin{tikzcd}
\cdots \arrow[r, "H(i_{l-1})"] \arrow[d] & H_*(\mathcal{K}_l) \arrow[r, "H(i_l)"] \arrow[d, "H(j_l)"] & H_*(\mathcal{K}_{l+1}) \arrow[r, "H(i_{l+1})"] \arrow[d, "H(j_{l+1})"] & \cdots \arrow[d] \\
\cdots \arrow[r, "H(k_{l-1})"] & H_*(\mathcal{K}_l \setminus \{A, b\}) \arrow[r, "H(k_l)"] & H_*(\mathcal{K}_{l+1} \setminus \{A, b\}) \arrow[r, "H(k_{l+1})"] & \cdots
\end{tikzcd}$$

The horizontal maps $i_l \colon \mathcal{K}_l \hookrightarrow \mathcal{K}_{l+1}$ and $k_l \colon \mathcal{K}_l \setminus \{A, b\} \hookrightarrow \mathcal{K}_{l+1} \setminus \{A, b\}$ are inclusion maps. The vertical maps $H(j_l) \colon H_*(\mathcal{K}_l) \to H_*(\mathcal{K}_l \setminus \{A, b\})$ are isomorphisms due to Theorem 4.1 and the fact that $\{A, b\}$ is a coreduction pair in both $\mathcal{K}_l$ and $\mathcal{K}_{l+1}$ and $g(A) = g(b)$. Analogously, as in Theorem 3.1, one can show that all the squares commute. Therefore, due to Theorem 2.1, persistence defined by lower and upper sequence of complexes is equal, which completes the proof. □

It remains to show that we can remove a single vertex V from the complex and this changes only the zero-dimensional persistent homology. The fact that higher-dimensional persistence is not affected follows from the definition of reduced homology and persistence. To be precise, let us recall the augmented chain complex in the setting of reduced homology,

$$\xrightarrow{\partial_{n+1}} C_n(\mathcal{K}) \xrightarrow{\partial_n} \cdots \xrightarrow{\partial_2} C_1(\mathcal{K}) \xrightarrow{\partial_1} C_0(\mathcal{K}) \xrightarrow{\epsilon} \mathbb{F} \to 0,$$

where $\epsilon(\sum_i \alpha_i t_i) = \sum_i \alpha_i$ for $t_i \in \mathcal{K}$.

In this setting, removal of the initial vertex V can be interpreted as a removal of a coreduction pair $(V, \emptyset)$, where $\emptyset$ is the unique generator of $\mathbb{F}$ in dimension -1. From the properties of the reduced homology, it follows that removing V changes only the zero-dimensional homology (and persistence).

In the standard homology, once the coreduction pairs are removed as long as possible, all vertices in a given connected component are removed. The example in Figure 2 shows that this does not hold in the case of persistent homology: Even if we start from the vertex with the lowest filtration value and perform the coreductions as long as possible, some vertices may remain in the considered connected component.

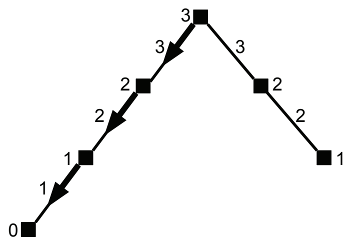

Figure 2: One-dimensional complex, filtered with maxima. Numbers indicate the filtration values. Initially the vertex having value 0 is removed. Arrows indicate pairings done by coreductions. This example shows that, unlike the case of standard homology, not all vertices from the connected component can be removed during coreductions.

As can be seen in Figure 3, all information about zero-dimensional persistence is lost when the coreductions are done. However, it is fairly easy to compute zero-dimensional persistence in near-linear time from the original complex by using union–find data structures, as described in Edelsbrunner et al. [**5**].

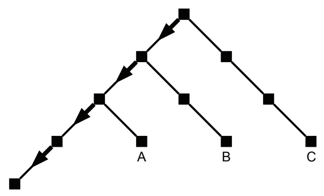

Figure 3: An example showing that when the coreductions are done the information about zero-dimensional persistence is lost. For the sake of clarity the filtration is simply a height function (it is not depicted with numbers). We assume the filtration of an edge is the maximal filtration level of its vertices. The coreductions are started by removing the left-bottom vertex and are indicated by arrows. It is easy to see that in the coreduced complex, vertices A, B, and C generate infinite persistence intervals in zero-dimensional persistent homology. Those infinite intervals are not present in persistent homology of the initial complex.

As can be seen in Figure 4, removing a coreduction pair (A, b) such that $g(A) > g(b)$ does change the persistent homology of the given complex. Therefore the assumptions presented in Theorem 4.2 are crucial.

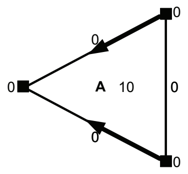

Figure 4: Simple demonstration showing that the condition $g(A) = g(b)$ is necessary when removing coreduction pair (A, b). Suppose a coreduction pair (A, b) presented in the picture is removed (the vertex on the left and the edges paired with remaining vertices are already removed from the complex). It is then clear that once pair (A, b) is removed, an interval $[0, 10]$ is lost from one-dimensional persistence.

To summarize this section, the coreductions for persistence are presented in Algorithm 2.

Algorithm 2 Coreductions for persistence.

```
Input: $\mathcal{K}$ - cell complex with filtration $g : \mathcal{K} \to \mathbb{N}$;
Output: Reduced complex $\mathcal{K}'$ with the same persistence for dimensions $\geqslant 1$ as $\mathcal{K}$.
  $\mathcal{K}' = \mathcal{K}$;
  List of cells Q= $\emptyset$;
  for every connected component of $\mathcal{K}'$ do
    Remove a single vertex $V$ from considered connected component of $\mathcal{K}'$;
  for every cell $B \in \mathcal{K}'$ do
    if $B$ has unique cell $a$ in boundary and $g(B) = g(a)$ then
      Q.$enqueue(B)$;
  while Q is not empty do
    $B$ = Q.$dequeue()$;
    if $B$ has unique cell $a$ in boundary and $B, a \in \mathcal{K}'$ and $g(B) = g(a)$ then
      $\mathcal{K}' = \mathcal{K}' \setminus \{B, a\}$;
      for every element $C \in \mathcal{K}'$ in coboundary of $a$ or coboundary of $B$ do
        if $C$ has unique cell $d \in \mathcal{K}'$ in boundary and $g(C) = g(d)$ then
          Q.$enqueue(C)$;
```

5. Acyclic subspace

The acyclic subspace method follows from the excision property. In standard homology theory it states that for a complex $\mathcal{K}$ and a subcomplex $\mathcal{A}$ of $\mathcal{K}$ such that $H_n(\mathcal{A}) = 0$ for $n > 0$ and $\mathcal{A}$ intersects every connected component of $\mathcal{K}$, we have $H_n(\mathcal{K}) = H_n(\mathcal{K}, \mathcal{A})$ for $n > 0$. Since $\mathcal{A}$ is closed we can use the following theorem from Mrozek et al. [**12**].

Theorem 5.1 (Theorem 3.1 [**12**]). *If $\mathcal{A}$ is closed in $\mathcal{K}$, then $H_*(\mathcal{K} \setminus \mathcal{A}) = H_*(\mathcal{K}, \mathcal{A})$.*

Therefore it suffices to compute homology of a chain complex obtained by removing all elements in $\mathcal{A}$ from $\mathcal{K}$. If one is able to efficiently find the large acyclic subcomplex $\mathcal{A}$, then this approach offers great performance gains. For instance, this method has been used to speed up cubical homology computations in Mrozek et al. [**13**] and to compute cohomology generators efficiently in Dłotko et al. [**4**]. A similar technique has been used to speed up computations of homology of inclusions by Żelazna [**18**].

The presented strategy can be especially useful in the case of cubical data, for example 3D images. In this case we work only on top-dimensional cells. This is important, as the number of faces (of all dimensions) of a given cell is exponential in its dimension. Even though this method is limited to low dimensions, the performance gains can be significant.

In this section we demonstrate how this technique can be used to speed up computations of persistent homology. To do this, we need to introduce the concept of an *acyclic subcomplex compatible with filtration.* Let $\mathcal{A}$ be a subcomplex of a filtered complex $\mathcal{K}_0 \subset \cdots \subset \mathcal{K}_n = \mathcal{K}$. We say that $\mathcal{A}$ is an *acyclic subcomplex compatible with filtration* if $\mathcal{A}_i = \mathcal{K}_i \cap \mathcal{A}$ is an acyclic subcomplex in $\mathcal{K}_i$ for every $i \in \{0, \ldots, n\}$.

In order to obtain an acyclic subcomplex compatible with filtration, first the maximal acyclic subcomplex $\mathcal{A}_0$ is constructed in $\mathcal{K}_0$ as described, for instance, in Mrozek et al. [**13**].[2] Then elements in $\mathcal{K}_1$ are processed to construct $\mathcal{A}_1$. The element $B \in \mathcal{K}_1$ is added to to the complex iff:

1. The intersection with the acyclic complex constructed so far is acyclic, and
2. there are no elements in boundary of B intersected with $\mathcal{K}_0$ that are not in acyclic subcomplex $\mathcal{A}_0$.

For the higher values of filtration we use analogous criterion. Condition (2) for $g(B) = i$ should be then replaced with:

2. there are no elements in boundary of B intersected with $\mathcal{K}_j$ for $j < i$ that are not in acyclic subcomplex $\mathcal{A}_{i-1}$.

In this way we ensure that the closure of the final acyclic subcomplex intersected with every level of filtration is an acyclic subcomplex at this level of filtration, as shown in Theorem 5.2. This condition is necessary, as explained in Figure 5.

Theorem 5.2. *As a result of the presented procedure, an acyclic subcomplex compatible with filtration is obtained.*

Proof. Due to (1), the obtained complex is acyclic. From point (2) we have that $\mathcal{K}_i \setminus \mathcal{A} = \mathcal{K}_i \setminus \mathcal{A}_i$. Therefore we obtain an acyclic subcomplex compatible with filtration. □

Let us show that such a procedure does not change persistence in dimension greater or equal 1.

[2]The idea of the procedure is as follows. First, a top-dimensional cell A in $\mathcal{K}_0$ is chosen. Then all its neighbor elements in $\mathcal{K}_0$ are processed. Element $B \in \mathcal{K}_0$ is added to the acyclic subcomplex iff its intersection with the current acyclic subcomplex is acyclic. For fast tests, tabulated configurations for cubes and simplices can be used.

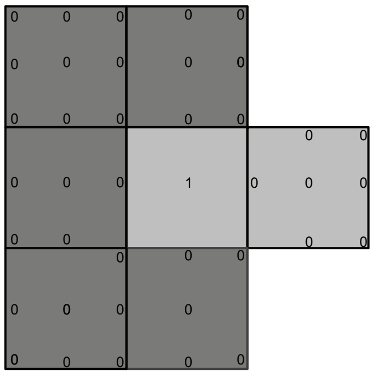

Figure 5: At the first filtration level the 2-dimensional elements with filtration 0 (marked with darker gray) belong to the acyclic subcomplex. If, as in Mrozek et al. [**13**], we considered only intersection with an acyclic subcomplex when generating the acyclic subcomplex at the filtration level 1, the middle 2-dimensional element would be added to the acyclic subset at filtration level 1. In the end, when the whole (closed) acyclic subspace is removed, we lose the $[0, 1]$ persistence interval in dimension one. This cannot happen when the extra restrictions are imposed.

Theorem 5.3. *Let $\mathcal{K}_0 \subset \cdots \subset \mathcal{K}_n = \mathcal{K}$ be a filtered complex and let $\mathcal{A}_0 \subset \cdots \subset \mathcal{A}_n = \mathcal{A}$, such that $\mathcal{A}_i = \mathcal{K}_i \setminus \mathcal{A}$ is an acyclic subcomplex compatible with filtration. Then persistent homologies $H^p_l(\mathcal{K})$ and $H^p_l(\mathcal{K} \setminus \mathcal{A})$ are the same for $l > 0$.*

Proof. Again, let us write the following diagram:

$$\begin{tikzcd}
\cdots \arrow[r, "H(i_{l-1})"] \arrow[d] & H_*(\mathcal{K}_l) \arrow[r, "H(i_l)"] \arrow[d, "H(j_l)"] & H_*(\mathcal{K}_{l+1}) \arrow[r, "H(i_{l+1})"] \arrow[d, "H(j_{l+1})"] & \cdots \arrow[d] \\
\cdots \arrow[r, "H(k_{l-1})"] & H_*(\mathcal{K}_l \setminus \mathcal{A}_l) \arrow[r, "H(k_l)"] & H_*(\mathcal{K}_{l+1} \setminus \mathcal{A}_{l+1}) \arrow[r, "H(k_{l+1})"] & \cdots
\end{tikzcd}$$

The horizontal arrows are induced by inclusion. While it is straightforward for the upper sequence, the lower one requires some explanation, since in general both $\mathcal{K}_l \subset \mathcal{K}_{l+1}$ and $\mathcal{A}_l \subset \mathcal{A}_{l+1}$. The k_l is an inclusion since $\mathcal{A}_{l+1} \setminus \mathcal{A}_l \subset \mathcal{K}_{l+1} \setminus \mathcal{K}_l$.

From the exact sequence of a pair $(\mathcal{K}_l, \mathcal{A}_l)$:

$$\to H_n(\mathcal{A}_l) \to H_n(\mathcal{K}_l) \to H_n(\mathcal{K}_l, \mathcal{A}_l) \to$$

Since $H_n(\mathcal{A}_l) = 0$ for $n > 0$ we have that $H_n(\mathcal{K}_l)$ is isomorphic to $H_n(\mathcal{K}_l, \mathcal{A}_l)$. From Theorem 5.1 we have that $H_n(\mathcal{K}_l, \mathcal{A}_l)$ is isomorphic to $H_n(\mathcal{K}_l \setminus \mathcal{A}_l)$. Therefore all the vertical arrows are isomorphisms for $n > 0$. Again, as in Theorem 3.1, one can show that all the squares commute since $\mathcal{A}_{l+1} \setminus \mathcal{A}_l \subset \mathcal{K}_{l+1} \setminus \mathcal{K}_l$. Consequently, from Theorem 2.1, the persistence of the lower and upper complexes are equal for $n > 0$. □

For the zero-dimensional persistence, the same situation as the one presented in Figure 3 occurs. Therefore all the information about zero-dimensional persistence is lost. They can be retrieved in near linear time by using a union–find data structure.

Algorithm 3 summarizes the acyclic subspace method for persistence. In this algorithm only the top dimensional elements need to be represented.

Algorithm 3 Acyclic subspace algorithm for persistence.

```
Input: $\mathcal{K}$-cell complex with filtration $g\colon \mathcal{K} \to \mathbb{N}$
Output: $\mathcal{A}$-acyclic subcomplex of $\mathcal{K}$ compatible with filtration $g$;
  $min$ = minimal value of $g$ on $\mathcal{K}$;
  $\mathcal{A} = \emptyset$, the acyclic subcomplex;
  for every connected component $\mathcal{C}$ of $\mathcal{K}$ do
    Pick any top dimensional cell $T \in \mathcal{C}$ having minimal filtration value in its connected
    component. Set $\mathcal{A} = \mathcal{A} \cup \{T\}$;
  for every filtration value $i$ starting from $min$ in increasing order do
    List of cells $\mathtt{Q} = \emptyset$;
    for every $T$ such that $g(T) = i$ do
      if $T \cap \mathcal{A}$ is acyclic and for every element $a \in T \setminus (T \cap \mathcal{A})$, $g(a) = i$ then
        $\mathtt{Q} = \mathtt{Q} \cup T$;
    while $\mathtt{Q}$ is not empty do
      $T = \mathtt{Q}.dequeue()$;
      if $T \cap \mathcal{A}$ is acyclic then
        $\mathcal{A} = \mathcal{A} \cup T$;
        for every cell $T'$ incident to $T$ such that $g(T') = i$ and $T' \notin \mathcal{A}$ do
          if $T' \cap \mathcal{A}$ is acyclic and for every element $a \in T' \setminus (T' \cap \mathcal{A})$, $g(a) = i$ then
            $\mathtt{Q} = \mathtt{Q} \cup T'$;
```

The complexity of this algorithm is linear, provided the cells of the complex and the neighboring cells can be iterated according to the filtration levels of the function g.

6. Smoothing the data

Let us fix a complex $\mathcal{K}$ and a filtering function f. The number of elements reduced with the described algorithms strongly depends on the filtering function f. It may happen, especially for noisy data, that no reduction can be made, because the value of the cell and some of its faces differs infinitesimally. In this section we show a heuristic for *denoising* such data, controlling the changes of persistence. This way we increase the effectiveness of the presented reduction methods. This idea is based on stability results for persistence described in Cohen-Steiner et al. [**3**]. As recalled in Equation 1, stability of persistence states that under the ϵ change (in the maximum norm) of the filtering function, the persistence diagrams will change by no more than ϵ in the so-called bottleneck distance [**3**].

We start with free face collapses and coreductions. One can view the denoising procedure as constructing a perturbed function $f'\colon \mathcal{K} \to \mathbb{N}$ (we assume that $f' = f$ on all cells in which the perturbation did not take place). Suppose we have an elementary collapse or a coreduction pair (A, b) such that

1. $f(A) - f(b) < \epsilon$, and
2. for every B being a coface of b we have $f(B) \geqslant f(A)$.

Then the filtration value of the cell b can be perturbed to $f'(b) = f(A)$ and a reduction of a pair (A, b) can possibly be made when a complex $\mathcal{K}$ with a filtering function f' is considered. We assume that value of a cell is perturbed at most once.

Let us show that f' is a filtration and $\|f - f'\|_\infty = max_{a\in\mathcal{K}}|f(a) - f'(a)| \leqslant \epsilon$. The fact that f' is a filtration of $\mathcal{K}$ follows from condition (2) above, and the fact that if the filtration value of a coface of b, such that $f(b) \neq f'(b)$, is changed, then the value of the coface is increased. The fact that, for every $a \in \mathcal{K}$, $|f(a) - f'(a)| \leqslant \epsilon$ follows from condition (1) and the fact that we are allowed to perturb the value of a cell only once. Therefore, from the stability of persistence by Cohen-Steiner et al. [**3**], we have $d_B(H^p(\mathcal{K}, f), H^p(\mathcal{K}, f')) \leqslant \epsilon$.

It is known from Leviner et al. [**9**] that maximizing the number of reduced elements is NP-complete, so a greedy strategy seems to be a viable option. One can perform a greedy reduction together with changing the value of the function if the conditions presented above are satisfied. We want to stress that once we want to change the value of a cell $a \in \mathcal{K}$ from $f(a)$ to $f'(a)$, then the inequality $f(B) \geqslant f'(a)$, for every B coface of a, needs to be checked in the initial complex (so we are not allowed to disregard the elements that have already been reduced if we do the perturbations at the same time as reduction). Otherwise the changes will accumulate and the ϵ precision will be lost, as presented in Figure 6.

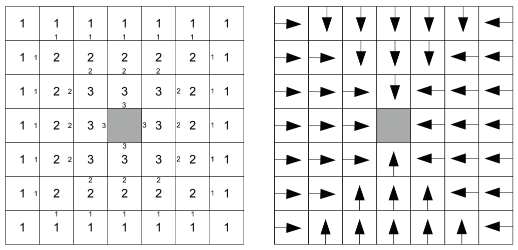

Figure 6: Example showing that when changing the filtering function values after some reductions were made, one should always check the presented conditions on the initial complex, not the reduced one. Otherwise the changes could accumulate and one would lose the bounds on the distances of the output persistence, as presented in this example. On the left is the initial complex with the initial function values. Filtration is given on top-dimensional cells and the lower-dimensional cell has the filtration equal to minimum of filtration of incident 2-dimensional cells. Suppose we want to obtain persistence with a tolerance $\epsilon = 1$. Then, if the reduced elements are forgotten and they are not taken care of in checking $f(B) \geqslant f'(b)$ for b being a face of B, then all the reductions presented on the right can be made, which clearly gives an interval $[3, \infty]$ in the dimension 1. The correct one, $[1, \infty]$, is farther than ϵ from the provided answer. By enlarging this example the difference can be made arbitrarily large.

A similar idea can be applied to the acyclic subcomplex method for persistence. Let us assume that the filtration f is given only on top-dimensional cells of a complex $\mathcal{K}$. Then f is assumed to be extended to other cells by using lower star filtration (i.e., every cell has a filtration equal to the minimal filtration of the incident top dimensional cells). Suppose one perturbs the values of the top-dimensional cells by introducing a perturbed function f', such that, for every top-dimensional cell $A \in \mathcal{K}$, we have $|f(A) - f'(A)| \leqslant \epsilon$. Once a lower star filtration is constructed for f', for every $a \in \mathcal{K}$ we have $|f(a) - f'(a)| \leqslant \epsilon$. Consequently, $d_B(H^p(\mathcal{K}, f), H^p(\mathcal{K}, f')) \leqslant \epsilon$. Due to this property one can do two things to increase the efficiency of acyclic subcomplex method.

1. Reduce the number of filtration levels of f' with respect to f as much as possible. This allows a better complexity of the algorithm (we remind the reader that the complexity of this algorithm is dependent on the number of filtration levels).
2. Use a greedy strategy when constructing the maximal acyclic subcomplex by changing the value of neighboring cells used to construct locally larger acyclic subcomplexes.

In both cases we face a hard optimization problem. Therefore, the only feasible solution would be to use a greedy strategy.

7. Conclusions

In this paper, we adopted existing reduction techniques, which were beneficial in homology computation, to the context of persistent homology. In many practical cases such a reduction should be used as a preprocessing step to avoid storing and processing the entire boundary matrix of the initial data. As in the case of standard homology computation, one can use the presented reductions in a sequence: acyclic subspace, then elementary collapses and coreductions.

References

[1] C. Chen and M. Kerber, *An output-sensitive algorithm for persistent homology*, 27th Annual Symposium on Computational Geometry (SoCG 2011), 2011.

[2] C. Chen and M. Kerber, *Persistent homology computation with a twist*, 27th European Workshop on Computational Geometry (EuroCG 2011), 2011.

[3] D. Cohen-Steiner, H. Edelsbrunner, and J. Harer, *Stability of Persistence Diagrams*, Discrete Comput. Geom., **37**(1) (2007), 103–120.

[4] P. Dłotko and R. Specogna, *Efficient cohomology computation for electromagnetic modeling*, CMES: Computer Modeling in Engineering & Sciences, vol. 60, no. 3 (2010), 247–278.

[5] H. Edelsbrunner and J. Harer, *Computational Topology. An Introduction.* Amer. Math. Soc., Providence, Rhode Island, 2010.

[6] G. Ellis and F. Hegarty, *Computational homotopy of finite regular CW-spaces*, Journal of Homotopy and Related Structures, published online May 2013.

[7] D. Günther, J. Reininghaus, H. Wagner, and I. Hotz, *Effcient Computation of 3D Morse-Smale Complexes and Persistent Homology using Discrete Morse Theory*, The Visual Computer **28**(10) (2012), 959–969.

[8] T. Kaczynski, K. Mischaikow, and M. Mrozek, *Computational Homology*, Applied Mathematical Sciences 157, Springer-Verlag, 2004.

[9] Th. Lewiner, H. Lopes, and G. Tavares, *Optimal discrete Morse functions for 2-manifolds*, Computational Geometry: Theory and Applications, **26**(3) (2003), 221–233.

[10] N. Milosavljevic, D. Morozov, and P. Skraba, *Zigzag Persistent Homology in Matrix Multiplication Time*, Proceedings of the 27th Annual Symposium on Computational Geometry (SCG'11), 2011.

[11] K. Mischaikow and V. Nanda, *Morse theory for filtrations and efficient computation of persistent homology*, Discrete & Computational Geometry, vol. 50, no. 2 (2013), 330–353.

[12] M. Mrozek and B. Batko, *Coreduction homology algorithm*, Discrete & Computational Geometry, **41**(1) (2009), 96–118.

[13] M. Mrozek, P. Pilarczyk, and N. Żelazna, *Homology Algorithm Based on Acyclic Subspace*, Computers and Mathematics with Applications **55** (2008), 2395–2412.

[14] M. Mrozek and Th. Wanner, *Coreduction homology algorithm for inclusions and persistent homology*, Computers and Mathematics with Applications **60**(10) (2010), 2812–2833.

[15] 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, vol. 33, no. 8 (2011), 1646–1658.

[16] H. Wagner, C. Chen, and E. Vucini, *Efficient computation of persistent homology for cubical data*, Proceedings of the 4th Workshop on Topology-based Methods in Data Analysis and Visualization (TopoInVis 2011), 2011.

[17] J.H.C. Whitehead, *Simple homotopy types*, Amer. J. Math. **72** (1950), 1–57.

[18] N. Żelazna, *Acyclic Subspace Homology Algorithm for Inclusions*, Schedae Informaticae, 2007.

Paweł Dłotko `dlotko@sas.upenn.edu`

Department of Mathematics, University of Pennsylvania, Philadelphia, PA 19104-6395, USA, and Institute of Computer Science, Jagiellonian University, Lojasiewicza 6, Krakow, Poland

Hubert Wagner `hubert.wagner@uj.edu.pl`

Institute of Computer Science, Jagiellonian University, Lojasiewicza 6, Krakow, Poland