Computing Persistent Homology of Flag Complexes via Strong Collapses

Jean-Daniel Boissonnat
Université Côte d'Azur, INRIA, Sophia Antipolis, France
Jean-Daniel.Boissonnat@inria.fr

Siddharth Pritam
Université Côte d'Azur, INRIA, Sophia Antipolis, France
siddharth.pritam@inria.fr

Abstract

In this article, we focus on the problem of computing Persistent Homology of a flag tower, i.e. a sequence of flag complexes connected by simplicial maps. We show that if we restrict the class of simplicial complexes to flag complexes, we can achieve decisive improvement in terms of time and space complexities with respect to previous work. We show that strong collapses of flag complexes can be computed in time $O(k^2v^2)$ where v is the number of vertices of the complex and k is the maximal degree of its graph. Moreover we can strong collapse a flag complex knowing only its 1-skeleton and the resulting complex is also a flag complex. When we strong collapse the complexes in a flag tower, we obtain a reduced sequence that is also a flag tower we call the core flag tower. We then convert the core flag tower to an equivalent filtration to compute its PH. Here again, we only use the 1-skeletons of the complexes. The resulting method is simple and extremely efficient.

2012 ACM Subject Classification Mathematics of computing; Theory of computation → Computational geometry; Mathematics of computing → Algebraic topology

Keywords and phrases Computational Topology, Topological Data Analysis, Strong Collapse, Persistent homology

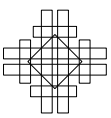
Digital Object Identifier 10.4230/LIPIcs.SoCG.2019.55

Funding This research has received funding from the European Research Council (ERC) under the European Union's Seventh Framework Programme (FP/2007- 2013) / ERC Grant Agreement No. 339025 GUDHI (Algorithmic Foundations of Geometry Understanding in Higher Dimensions).

1 Introduction

This paper is a continuation of the research reported in [8] on the usage of strong collapses to accelerate the computation of the Persistent Homology (PH) of a sequence of simplicial complexes. Computing PH efficiently is a very well studied problem in Computational Topology and Topological Data Analysis. The time complexity to compute persistent homology is $O(n^\omega)$, where n is the total number of simplices and $\omega \leq 2.4$ is the matrix multiplication exponent [38, 31]. In practice, when we encounter massive and high-dimensional datasets, n may be very large n (of order of billions) and computing PH is then slow and memory intensive. This especially occurs for flag complexes since the simplices of a flag complex are the cliques of the 1-skeleton of the complex.

There has been a lot of progress in the recent years along two directions. On one hand, efficient implementations and optimizations have led to a new generation of software for PH computation [32, 6, 4, 40]. Another complementary direction has been explored to reduce the size of the complexes in the sequence while preserving (or approximating in a controlled way) the persistent homology of the sequence [39, 26, 14, 9, 45, 36, 16, 23]. Most of the methods are for general simplicial complexes except [4, 45] which focus on the Vietoris-Rips complex, an example of a flag complex. In this paper, we build on the initial success of

35th International Symposium on Computational Geometry (SoCG 2019).
Editors: Gill Barequet and Yusu Wang; Article No. 55; pp. 55:1–55:15


Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

[8] and show that further decisive progress can be obtained if one restricts the family of simplicial complexes to flag complexes. Flag complexes are fully characterized by their graph (or 1-skeleton), the other faces being obtained by computing the cliques of the graph. Hence, a flag complex can be represented by its 1-skeleton, which is a very compact representation. Flag complexes are very popular and, in particular, Vietoris-Rips complexes are widely used in Topological Data Analysis.

It has been shown in [8] that the persistent homology of a sequence of simplicial complexes can be computed very efficiently using *strong collapses*. The basic idea is to simplify the complexes of the input sequence by using strong collapses, as introduced by J. Barmak and E. Miniam [3], and to compute the PH of an induced sequence of reduced simplicial complexes that has the same PH as the initial one. This reduced sequence is unique up to isomorphism and is called the *core sequence*. A crucial advantage of the method is that it only needs to store the *maximal simplices* of the complex, not the full set of the simplices of all dimensions, which saves space and time by a factor that can be exponential in the dimension of the complex. Still, in the case of a flag complex and, in particular, in the case of the widely used Vietoris-Rips complex, the maximal simplices are the maximal cliques of the 1-skeleton of the complex and their number can be very large (exponential in the dimension of the complex). As a result, the method of [8] devoted most of the time to compute the maximal faces of the complexes prior to their strong collapse.

In this paper, we avoid computing maximal cliques and show that we can strong collapse any flag complex using only the *1-skeleton* or graph of the complex. Another crucial observation is that the reduced complex obtained by strong collapsing a flag complex is itself a flag complex.

Furthermore, if we consider a sequence of flag complexes connected by simplicial maps (a flag tower), we can strong collapse all the complexes of the sequence. The obtained core sequence is also a flag tower with smaller complexes that are connected by simplicial maps that are induced from the maps of the original sequence and the strong collapses. In the general case where the core sequence is not a filtration (which usually happens even if the original sequence is a filtration), we need to convert the flag tower to an equivalent filtration to compute its PH using known algorithms. To do so, we build on the work of [22, 35] that we restrict to flag complexes and strong collapses. This allows us to convert a flag tower to an equivalent flag filtration using again only the 1-skeleton.

The major advantages of our approach are:

- We can compute PH for large complexes of high dimensions as we don't need to compute the maximal simplices.
- The dimension of the original sequence is in fact irrelevant and what matters is the dimension of the core sequence, which is usually quite small.
- Instead of computing the exact PH, we can compute an approximate PH which is substantially faster at a very minimal cost. We will explain more about the approximation scheme in Sections 2 and 5.

The resulting method is simple and extremely efficient. On the theory side, we show that strong collapses can be computed in time $O(k^2v^2)$ where v is the number of vertices of the complex and k the maximal degree of its graph. The algorithm described in this paper has been implemented. Numerous experiments show that the computation of the persistent homology of flag complexes can be obtained much faster than with previous methods, e.g. Ripser [4]. The code will be soon released in the Gudhi library [32].

2 Preliminaries

In this section, we provide a brief review of the notions of simplicial complex and strong collapse as introduced in [3] and recalled in [8]. We use the same notations and conventions as in [8] and just recall, in addition, some basic facts about *Flag complexes*. Readers can refer to [33] for a comprehensive introduction to these topics.

Simplex, simplicial complex and simplicial map. An **abstract simplicial complex** K is a collection of subsets of a non-empty finite set X, such that for every subset A in K, all the subsets of A are in K. From now on we will call an *abstract simplicial complex* simply a *simplicial complex* or just a *complex*. An element of K is called a **simplex**. An element of cardinality $k+1$ is called a k-simplex and k is called its **dimension**. A simplex is called **maximal** if it is not a proper subset of any other simplex in K. A sub-collection L of K is called a **subcomplex**, if it is a simplicial complex itself.

A map $\psi : K \to L$ between two simplicial complexes is called a **simplicial map**, if it always maps a simplex in K to a simplex in L. Simplicial maps are induced by vertex-to-vertex maps. In particular, there is a finite number of simplicial maps between two given finite simplicial complexes. Simplicial maps induce continuous maps between the underlying geometric realisations of the simplicial complexes. Any general simplicial map can be decomposed into more elementary simplicial maps, namely **elementary inclusions** (i.e., inclusions of a single simplex) and **elementary contractions** $\{\{u, v\} \mapsto u\}$ (where a vertex is mapped onto another vertex). Two simplicial maps $\phi : K \to L$ and $\psi : K \to L$ are **contiguous** if, for all $\sigma \in K$, $\phi(\sigma) \cup \psi(\sigma) \in L$. Two contiguous maps are known to be homotopic [41, Theorem 12.5].

Flag complex. A complex K is a flag or a clique complex if, when a subset of its vertices has pairwise edges between them, they span a simplex. It follows that the full structure of K is determined by its 1-skeleton we denote by G. For a vertex v in G, the **open neighborhood** $N_G(v)$ of v in G is defined as $N_G(v) := \{u \in G \mid [uv] \in E\}$. The **closed neighborhood** $N_G[v]$ is $N_G[v] := N_G(v) \cup \{v\}$. We further define the relative closed neighborhood of u by v in G as the set of vertices in $N_G[u]$ that are not in $N_G[v]$. We denote it by $N_G[u \setminus v]$.

Dominated vertex. Let σ be a simplex of a simplicial complex K, the **closed star** of σ in K, $st_K(\sigma)$ is a subcomplex of K which is defined as follows, $st_K(\sigma) := \{\tau \in K |\ \tau \cup \sigma \in K\}$. The **link** of σ in K, $lk_K(\sigma)$ is defined as the set of simplices in $st_K(\sigma)$ which do not intersect with σ, $lk_K(\sigma) := \{\tau \in st_K(\sigma) | \tau \cap \sigma = \emptyset\}$.

Taking a join with a vertex transforms a simplicial complex into a **simplicial cone**. Formally if L is a simplicial complex and a is a vertex not in L then the simplicial cone aL is defined as $aL := \{a, \tau \mid \tau \in L \text{ or } \tau = \sigma \cup a; \text{ where } \sigma \in L\}$. A vertex v in K is called a **dominated vertex** if the link of v in K, $lk_K(v)$ is a simplicial cone, that is, there exists a vertex $v' \neq v$ and a subcomplex L in K, such that $lk_K(v) = v'L$. We say that the vertex v' is dominating v and v is dominated by v'. The symbol $\boldsymbol{K \setminus v}$ (deletion of v from K) refers to the subcomplex of K which has all simplices of K except the ones containing v. Below is an important remark from [3, Remark 2.2], which proposes an alternative definition of dominated vertices.

▶ Remark 1. A vertex $v \in K$ is dominated by another vertex $v' \in K$, *if and only if* all the maximal simplices of K that contain v also contain v' [3].

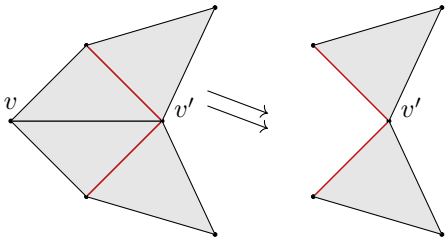


Figure 1 Illustration of an *elementary strong collapse*. In the complex on the left, v is dominated by v'. The link of v is highlighted in red. Removing v leads to the complex on the right.

Strong collapse. An **elementary strong collapse** is the deletion of a dominated vertex v from K, which we denote with $K \searrow\searrow K \setminus v$. Figure 1 illustrates an easy case of an elementary strong collapse. There is a **strong collapse** from a simplicial complex K to its subcomplex L, if there exists a series of elementary strong collapses from K to L, denoted as $K \searrow\searrow L$. The inverse of a strong collapse is called a **strong expansion**. If there exists a combination of strong collapses and/or strong expansions from K to L, then K and L are said to have the same **strong homotopy type**.

The notion of strong homotopy type is stronger than the notion of simple homotopy type in the sense that if K and L have the same strong homotopy type, then they have the same simple homotopy type, and therefore the same homotopy type [3]. There are examples of contractible or simply collapsible simplicial complexes that are not strong collapsible.

A complex without any dominated vertex will be called a **minimal complex**. A **core** of a complex K is a minimal subcomplex $K^c \subseteq K$, such that $K \searrow\searrow K^c$. *Every simplicial complex has a* ***unique core*** *up to isomorphism. The core decides the strong homotopy type of the complex*, and two simplicial complexes have the same strong homotopy type *if and only if* they have isomorphic cores [3, Theorem 2.11].

Retraction map. If a vertex $v \in K$ is dominated by another vertex $v' \in K$, the vertex map $r : K \to K \setminus v$ defined as: $r(w) = w$ if $w \neq v$ and $r(v) = v'$, induces a simplical map that is a *retraction* map. The homotopy between r and the identity $i_{K\setminus v}$ over $K \setminus v$ is in fact a strong deformation retract. Furthermore, the composition $(i_{K\setminus v})r$ is contiguous to the identity i_K over K [3, Proposition 2.9].

Sequences of complexes. A **sequence** of simplicial complexes $\mathcal{T} : \{K_1 \xrightarrow{f_1} K_2 \xrightarrow{f_2} K_3 \xrightarrow{f_3} \cdots \xrightarrow{f_{(m-1)}} K_m\}$, connected through simplicial maps f_i is called a **simplicial tower** or simply a *tower*. We call a tower a **flag tower** if all the simplicial complexes K_i are flag complexes. When all the simplicial maps f_is are inclusions, then the tower is called a filtration and a flag tower is called a **flag filtration**.

Persistent homology. If we compute the homology classes of all the K_i, we get the sequence $\mathcal{P}(\mathcal{T}) : \{H_p(K_1) \xrightarrow{f_1^*} H_p(K_2) \xrightarrow{f_2^*} H_p(K_3) \xrightarrow{f_3^*} \cdots \xrightarrow{f_{(m-1)}^*} H_p(K_m)\}$. Here $H_p()$ denotes the homology class of dimension p with coefficients from a field $\mathbb{F}$ and $*$ denotes an induced homomorphism. $\mathcal{P}(\mathcal{T})$ is a sequence of vector spaces connected through homomorphisms, called a **persistence module**. More formally, a *persistence module* $\mathbb{V}$ is a sequence of vector spaces $\{V_1 \to V_2 \to V_3 \to \cdots \to V_m\}$ connected with homomorphisms $\{\to\}$ between them. A persistence module arising from a sequence of simplicial complexes captures the evolution of the topology of the sequence.

Any persistence module can be *decomposed* into a collection of intervals of the form $[i, j)$ [10]. The multiset of all the intervals $[i, j)$ in this decomposition is called the **persistence diagram** of the persistence module. An interval of the form $[i, j)$ in the persistence diagram of $\mathcal{P}(\mathcal{T})$ corresponds to a homological feature (a 'cycle') which appeared at i and disappeared at j. The persistence diagram (PD) completely characterizes the persistence module, that is, there is a bijective correspondence between the PD and the equivalence class of the persistence module [10, 51].

Two different persistence modules $\mathbb{V} : \{V_1 \to V_2 \to \cdots \to V_m\}$ and $\mathbb{W} : \{W_1 \to W_2 \to \cdots \to W_m\}$, connected through a set of homomorphisms $\phi_i : V_i \to W_i$ are **equivalent** if the ϕ_i are isomorphisms and the following diagram commutes [10, 20].

$$\begin{array}{ccccccc} V_1 & \longrightarrow & V_2 & \cdots & \longrightarrow V_{m-1} & \longrightarrow & V_m \\ \downarrow{\scriptstyle \phi_1} & & \downarrow{\scriptstyle \phi_2} & & \downarrow{\scriptstyle \phi_{m-1}} & & \downarrow{\scriptstyle \phi_m} \\ W_1 & \longrightarrow & W_2 & \cdots & \longrightarrow W_{m-1} & \longrightarrow & W_m \end{array}$$

The equivalent persistence modules will have the same interval decomposition, therefore the same diagram.

Overview of the algorithm and approximation scheme. The algorithm presented in this paper adopts the same strategy as the algorithm reported in [8]. However we just focus on *flag towers* instead of general sequences.

Let $\mathcal{T} : \{K_1 \xrightarrow{f_1} K_2 \xrightarrow{f_2} \cdots \xrightarrow{f_{(m-1)}} K_m\}$ be a flag tower of which we want to compute the persistence diagram. We first strong collapse the various complexes K_i, $i = 1, ..., m$ as suggested in [8]. Since each K_i is a flag complex, the computation of its core can be computed much more efficiently using only its 1-skeleton. The new algorithm is discussed in detail in Section 3.

We then compute a *core tower* that connects the K_is through induced simplicial maps. This is again an adaptation of what has been done in [8] for general simplicial complexes to the case of flag complexes. Lastly, we compute a flag *filtration* with the same PH as the core tower. This is similar to what has been done in [22, 35], and is detailed in Section 4. This filtration can then be sent to any algorithm that computes the persistence homology of a flag filtration [40, 4, 6, 32].

It is important to note that the algorithm computes the exact PH of the input flag tower $\mathcal{T}$. However, instead of considering all the K_is and the associated simplicial maps, we can select some of the K_is we rename $K'_1, ..., K'_q$, and compose the original maps $f_1, ..., f_{m-1}$ to obtain simplicial maps $f'_1, ..., f'_{q-1}$ connecting the selected complexes. We thus obtain a sub-tower $\mathcal{T}'$ and the algorithm will then compute the exact PH of $\mathcal{T}'$. An application of this idea will be presented for Vietoris-Rips filtrations in Section 5. By rounding the values of the threshold parameter to a given number of snapshots of the threshold value, we will be able to approximate in a controlled way the PH of the initial filtration $\mathcal{T}$.

3 Strong Collapse of a Flag complex

In this section, we show that the core of a flag complex K is itself a flag complex whose graph is called the *core graph* of K. The core graph of K can be computed from the 1-skeleton G of K in time $\mathcal{O}(v^2k^2)$, where v is the number of vertices in K and k is an upper bound on the degree of G (i.e. the number of edges that are incident on a vertex in K).

Although this change wrt the algorithm in [8] might look minor, it is crucial in practice as the time to compute all the maximal simplices of a flag complex from its graph is exponential in the number of its vertices. We thus reduce immensely the time and space complexity of the general algorithm of [8] whose complexity is $\mathcal{O}(v^2\Gamma_0 d + m^2\Gamma_0 d)$, where Γ_0 is an upper bound on the number of *maximal simplices* incident to a vertex.

In the following lemma, we describe a condition in terms of the closed neighborhood $N_G[v]$ of a vertex v of a flag complex K under which v will be dominated by another vertex v' of K. This result has been studied in another context in [30, Lemma 4.1].

▶ **Lemma 2.** *Let K be a flag complex. A vertex $v \in K$ is dominated by v' iff $N_G[v] \subseteq N_G[v']$.*

Proof. If v is dominated by v', then, according to Remark 1, the set of maximal simplices that contain v is a subset of the set of maximal simplices that contain v'. It follows that $N_G[v] \subseteq N_G[v']$.

Now we prove the other direction. Let σ be a maximal simplex of K containing v. Any other vertex x of σ is joined to v by an edge $[x, v] \in \sigma$. Moreover, since $N_G[v] \subseteq N_G[v']$, $[v, v']$ and $[x, v']$ are in K. It follows that every vertex in σ has an edge with both v and v' and, since K is a flag complex and σ is maximal, v' must be in σ. This implies that all the maximal simplices that contain v also contain v'. Hence v is dominated by v'. ◀

As mentioned before, an elementary strong collapse consists in removing a dominated vertex, and it can be easily observed that removing a vertex does not affect the 'flagness' of the residual complex $K \setminus v$. In other words, if σ is a maximal clique with vertex v, the resultant clique $\sigma \setminus v$ is still a maximal clique in $K \setminus v$. Moreover, all the other cliques that do not contain v still span the complete simplices. This implies that the core K^c of a flag complex K with graph G is a flag complex of a sub-graph G^c of G.

In what follows next, we describe an algorithm to compute the core graph $G^c \subseteq G$ whose flag complex is the core K^c of K.

Data structure. We represent G with its adjacency matrix M, where the rows and the columns of M represent the vertices of G. An entry $M[v_i][v_j]$ associated with vertices v_i and v_j is set to 1 if either the edge $[v_i, v_j] \in G$ or $i = j$, and to 0 otherwise. Note that we set $M[v_i][v_j] = 1$ for $i = j$ to be able to consider closed neighborhood. We will say that a row v is contained in another row v' if the set of column indices of the non-zero entries of v is a subset of the indices of the non-zero entries of v'. It is clear that if a row v is contained in another row v', we have $N_G[v] \subseteq N_G[v']$ and therefore the vertex v is dominated by the vertex v'.

Core graph algorithm. Given the adjacency matrix M of G, we compute the adjacency matrix C of the core graph G^c. In view of Lemma 2, we can easily compute C from M using basic row removal operations. Loosely speaking, we remove the rows of M that are contained in another row. After removing the row associated to v, we simultaneously update the matrix by removing the column associated to v. The process is iterated as long as the matrix can be reduced. Upon termination, we output the reduced matrix C, which is the adjacency matrix of the core graph G^c of K. Since the core of a complex is always unique, the order in which vertices are removed does not matter [3].

Retraction map computation. We can easily compute the retraction map r defined in Section 2 using the above core graph algorithm. A row v being removed in M corresponds to a dominated vertex in K and the row which contains v corresponds to a dominating vertex. Therefore we map the dominated vertex to the dominating vertex.

Domination tests optimization. Let us observe that, to check if a row v is dominated by some other row v', it is sufficient to compare v with its neighbors, which are at most k in number, if k denotes the maximum degree of the vertices in G.

We define a row v to be a **candidate row** for the next iteration if at least one of its neighbors has been removed in a previous row removal iteration. We observe that the candidate rows are the only rows that need to be considered in the domination tests of the algorithm. Indeed, a row w of M whose set of neighbors has not been modified at the previous *iteration* cannot be dominated by another row v' of M, as w was not dominated in the previous iteration and all other vertices can only loose neighbors. This ensures that w will still remain un-dominated.

We maintain a *queue*, for the candidate rows (rowQueue) which is implemented as a First in First out (FIFO) queue. At each iteration, we *pop out* a candidate row from rowQueue for domination test. After each successful domination test, we push the new candidate rows in the queue in preparation for the subsequent iteration. In the first iteration, we push all the rows in rowQueue. Algorithm 1 gives the pseudo code of our algorithm.

Algorithm 1 Core graph algorithm.

```
1:  procedure CORE(M)
2:      input : the adjacency matrix M of the graph of a flag complex K
3:      rowQueue ← push all rows of M (all vertices of K)
4:      while rowQueue is not empty do
5:          v ← pop(rowQueue)
6:          N_G[v] ← the non-zero columns of v
7:          for w in N_G[v] do
8:              if N_G[v] ⊆ N_G[w] then
9:                  Remove from M the column and the row associated to v
10:                 push all the entries of N_G(v) to rowQueue if not pushed before
11:                 break
12:             end if
13:         end for
14:     end while
15:     return M                          ▷ M is now the adjacency matrix of the core of K
16: end procedure
```

Time Complexity. Let us start by analyzing the most basic operation in our algorithm which is to determine if a row is dominated by another row. We store the rows of the matrix as sorted lists. Deciding if a sorted list is included in another sorted list (Line 8) can be done in time $\mathcal{O}(l)$, where l is the size of the longer list. In our case, the length of a row list is at most $k+1$ where k denotes as before the maximal degree of any vertex. Hence lines 8-12 takes $\mathcal{O}(k)$ time.

As explained in the paragraph *Domination tests optimization*, each row is checked against at most k other rows. Hence the for loop (Lines 7-13) is executed at most k times. Moreover, since at each iteration we ought to remove at least one row, the total number of iterations on the rows, i.e. the number of times the while loop is executed, is at most $O(v^2)$, where v is the total number of vertices of the complex K. It follows that the worst-case time complexity of our algorithm is $\mathcal{O}(v^2k^2)$.

4 From a Flag Tower to a Flag Filtration

In this section, we show that, thanks to the notion of strong collapses, we can efficiently turn a flag *tower* into a flag *filtration* using only edge inclusions over the 1-skeletons of the complexes.

4.1 Previous work

It is known that any general simplicial map can be decomposed into elementary inclusions and elementary contractions. Hence if we can replace an elementary contraction $\{\{u, v\} \mapsto u\}$ by an equivalent (not necessarily elementary) inclusion, we transform a tower into an equivalent filtration. This was the philosophy introduced by Dey et al. [22]. This idea has been further refined by Kerber and Schreiber [35] who introduced a slightly different approach based on coning. They provided theoretical bounds on the size and time to construct the final equivalent filtration. Specifically, they proved that the size of the equivalent filtration is $\mathcal{O}(d * n * \log n_0)$, where d is the maximal dimension of the complexes in the input tower and n (resp. n_0) the total number of elementary inclusions (resp. vertex inclusions) in the input tower [35, Theorem 2].

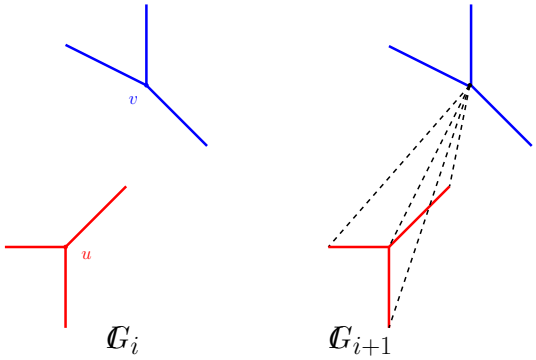
4.2 A new construction

We now present an algorithm that turns a flag tower into a flag filtration with the same PH. Our work builds upon the above mentioned previous works [22, 35]. The difference is that we use strong expansion which is the inverse operation of a strong collapse. The main advantage of strong expansions is that, when the input is a flag tower, we can use the domination criterion of Lemma 2. This leads to a simple algorithm that only deals with edges. The output filtration is a flag filtration, which can be represented very compactly. Moreover, since a strong expansion is a coning, we will be able to use the theoretical results of [35]. Now we describe our construction.

Let K_i be a flag complex and G_i be its 1-skeleton. We associate to K_i an augmented complex $\mathbb{K}_i \supseteq K_i$. As will be seen below, $\mathbb{K}_i$ is also a flag complex whose 1-skeleton will be denoted by $\mathbb{G}_i$. Following the terminology of [35], we call a vertex $v \in \mathbb{K}_i$ to be **active** if it is currently *not dominated*. The active closed neighborhood $ActN_{G_i}[v]$ is then defined as the set of all active vertices in $N_{G_i}[v]$. Similarly, $ActN_{G_i}[v \setminus u]$ denotes the set of active vertices in the closed neighborhood $N_{G_i}[v]$ of v that are not in $N_{G_i}[u]$. Finally, let $\{[u, ActN_{G_i}[v \setminus u]]\}$ denote the set of edges between u and $ActN_{G_i}[v \setminus u]$.

Using the notions defined above, we now explain how to inductively construct a filtration associated to a given flag tower. The construction operates in a streaming fashion on the 1-skeleton. We distinguish two kinds of inputs: elementary inclusions (of a vertex or an edge) and elementary contractions. For $i = 0$, we set $\mathbb{G}_0 = \emptyset$. We then define $\mathbb{G}_i$ as follows.

1. if $G_i \xrightarrow{\cup\sigma} G_{i+1}$ is an elementary *inclusion* where σ is either a vertex or an edge, we set $\mathbb{G}_{i+1} := \mathbb{G}_i \cup \sigma$.
2. if $G_i \xrightarrow{\{u,v\}\mapsto u} G_{i+1}$ is an elementary *contraction*
   - 2.1. if $|ActN_{\mathbb{G}_i}[v \setminus u]| \leq |ActN_{\mathbb{G}_i}[u \setminus v]|$, we set $\mathbb{G}_{i+1} := \mathbb{G}_i \cup \{[u, ActN_{\mathbb{G}_i}[v \setminus u]\}$ and v as contracted
   - 2.2. otherwise, we set $\mathbb{G}_{i+1} := \mathbb{G}_i \cup \{[v, ActN_{\mathbb{G}_i}[u \setminus v]\}$ and u as contracted.



Note that $\mathbb{G}_i \subseteq \mathbb{G}_{i+1}$ and thus $\mathbb{K}_i \subseteq \mathbb{K}_{i+1}$. We continue the construction until the end of our input tower.

Complexity Analysis. The vertices marked *contracted* during our construction are exactly the same as the inactive vertices defined in [35]. By construction, any contracted vertex will be dominated permanently in the filtration. Since such a vertex stops existing in the tower later on, its neighborhood stays the same and the vertex remains dominated. Therefore, at any point in our construction, the number of active vertices is less than the number of active vertices that are used in [35]. Moreover, since a strong expansion is a coning, the size of the final filtration in our construction is at most that obtained by the construction prescribed in [35]. Moreover, since we are working with 1-skeletons only, the space and time complexity of our method is much lower than that of [35].

Let us now analyze the time complexity of the algorithm. Notice that there are two basic operations on the 1-skeleton, elementary inclusions in Line 1, and the computation and comparison of $ActN_{\mathbb{G}_i}[v \setminus u]$ and $ActN_{\mathbb{G}_i}[u \setminus v]$ in Lines 2.1 and 2.2. Elementary inclusions can be performed in constant time $\mathcal{O}(1)$. To compute $ActN_{\mathbb{G}_i}[v \setminus u]$ we need to compute $N_{\mathbb{G}_i}[v]$, $N_{\mathbb{G}_i}[u]$ and the subset of vertices that are dominated (inactive) in $N_{\mathbb{G}_i}[v \setminus u]$. We can access $N_{\mathbb{G}_i}[v]$ and $N_{\mathbb{G}_i}[u]$ in constant time $\mathcal{O}(1)$ and compute the set-difference $N_{\mathbb{G}_i}[v \setminus u]$ in $\mathcal{O}(k \log k)$ time, where k is the maximum degree of a vertex. Finally computing the dominated vertices in $N_{\mathbb{G}_i}[v \setminus u]$ can be done in $\mathcal{O}(k^3)$ time as $|N_{\mathbb{G}_i}[v \setminus u]| \leq k$. Therefore computing $ActN_{\mathbb{G}_i}[v \setminus u]$ and $ActN_{\mathbb{G}_i}[u \setminus v]$ takes $\mathcal{O}(k^3)$ time. Since the size of active relative closed neighborhoods are bounded by k, for each elementary contraction, we include at most k edges in $\mathcal{O}(k)$ time. It follows that the worst-case time complexity for each elementary contraction is $\mathcal{O}(k^3)$.

We conclude that the time complexity of the algorithm is $\mathcal{O}(|\mathbb{G}_m| + n_c * k^3)$ where n_c is the number of elementary contractions in the input tower and $|\mathbb{G}_m|$ is the size of the 1-skeleton of the output flag filtration. The space complexity of our construction is $\mathcal{O}(n_0 * k)$ which is the size of a sparse adjacency matrix of a flag complex with n_0 vertices.

Correctness. Finally we prove few lemmas and state our main result Theorem 7 to certify the correctness of the output of our construction.

▶ **Lemma 3.** *Let $f_i : K_i \xrightarrow{\{u,v\}\mapsto u} K_{i+1}$ be the first elementary contraction in the tower $\mathcal{T} : K_0 \xrightarrow{f_0} K_1 \xrightarrow{f_1} \ldots \xrightarrow{f_{m-1}} K_m$. Then the complex K_{i+1} is a subcomplex of $\mathbb{K}_{i+1}$ and $\mathbb{K}_{i+1} \searrow\searrow K_{i+1}$.*

Proof. We prove the second part $\mathbb{K}_{i+1} \searrow\searrow K_{i+1}$ of the statement which then implies the first part $K_{i+1} \subset \mathbb{K}_{i+1}$. Since f_i is the first contraction $\mathbb{K}_i = K_i$ and $\mathbb{G}_i = G_i$. Let $\mathbb{G}_{i+1} := G_i \cup \{[u, ActN_{G_i}[v \setminus u]\}$ be the graph defined in the construction Line 2.1. By construction, contracting v to u in both graphs G_i and $\mathbb{G}_{i+1}$ yields the same graph G_{i+1}.

Let $x' \in ActN_{G_i}[v \setminus u]$. We observe that adding the edge $[ux']$ to G_i does not change the fact that all $x \in \{N_{G_i}[v \setminus u] \setminus ActN_{G_i}[v \setminus u]\}$ are dominated since the addition of $[ux']$ only adds neighbors to $N_{G_i}[x']$ and $N_{G_i}[u]$. Removing all the dominated vertices in $N_{G_i}[v \setminus u]$ thus provides a sequence of elementary strong collapses. By performing all such elementary strong collapses, $\mathbb{K}_{i+1}$ is eventually transformed into a complex K^0_{i+1}. By doing so, we have removed all the dominated vertices from $N_{G_i}[v \setminus u]$ and added edges between u and the non dominated vertices that are in $N_{G_i}[v \setminus u]$. This implies that v is dominated by u in K^0_{i+1}. The elementary strong collapse of v onto u, implies $K^0_{i+1} \searrow\searrow K_{i+1}$ and therefore $\mathbb{K}_{i+1} \searrow\searrow K_{i+1}$. ◀

▶ **Lemma 4.** *Let $f_i : K_i \xrightarrow{\{u,v\}\mapsto u} K_{i+1}$ be the first elementary contraction in the tower $\mathbb{T} : K_0 \xrightarrow{f_0} K_1 \xrightarrow{f_1} ... \xrightarrow{f_{m-1}} K_m$. Then the following diagram commutes*

$$\begin{array}{ccc} H_p(K_i) & \xrightarrow{f_i^*} & H_p(K_{i+1}) \\ & \searrow^{i^*} & \downarrow{\scriptstyle (i')^*} \\ & & H_p(\mathbb{K}_{i+1}) \end{array}$$

Here $i' : K_{i+1} \hookrightarrow \mathbb{K}_{i+1}$ is the inclusion induced by the strong collapse. i^* and $(i')^*$ are homomorphisms induced by the inclusion maps.

Proof. Using the fact that f_i is the first contraction, we have the inclusion $\mathbb{K}_i = K_i \subseteq \mathbb{K}_{i+1}$. Let K^0_{i+1} be the complex as defined in the proof of Lemma 3. Consider the following diagram of simplicial complexes, and note that $i' = i_1 \circ i_0$ where i_0 and i_1 are both inclusions induced by the respective strong collapses.

$$\begin{array}{ccc} K_i & \xrightarrow{f_i} & K_{i+1} \\ \downarrow{\scriptstyle i} & & \downarrow{\scriptstyle i_0} \\ \mathbb{K}_{i+1} & \underset{i_1}{\hookleftarrow} & K^0_{i+1} \end{array}$$

We claim that the maps $i' \circ f_i$ and i are contiguous, which we denote $i' \circ f_i \sim i$. Indeed, let σ be any simplex in K_i. Since i is an inclusion, $i(\sigma) = \sigma$.

Case 1. If $v \notin \sigma$, then $i' \circ f_i(\sigma) = \sigma = i(\sigma)$.

Case 2. If $v \in \sigma$, $f_i(\sigma)$ is a simplex $\gamma \in K_{i+1}$ that contains u and, since i_0 is an inclusion, $i_0 \circ f_i(\sigma) = \gamma$. Observe that, in the retraction map associated to the strong collapse $r_1 : \mathbb{K}_{i+1} \searrow\searrow K^0_{i+1}$, v is not contracted (by construction of K^0_{i+1}). Therefore $r_1 \circ i(\sigma)$ is a simplex $\gamma' \in K^0_{i+1}$ that contains v.
Now, as mentioned in the proof of Lemma 3, u dominates v in K^0_{i+1}. Therefore all the maximal simplices in K^0_{i+1} that contain v also contain u (Remark 1). Therefore, γ' is a face of a maximal simplex $\tau \in K^0_{i+1}$ that contains u (in addition to v).
Since γ is obtained by contracting v to u, γ must be a face of τ that contains both u and v. This implies that $\gamma' \cup \gamma \subseteq \tau$, which in turn implies that $r_1 \circ i(\sigma)$ is contiguous to $i_0 \circ f_i(\sigma)$. After composing both sides with i_1, we get $i_1 \circ r_1 \circ i(\sigma) \sim i_1 \circ i_0 \circ f_i(\sigma)$. Now since $\mathbb{K}_{i+1} \searrow\searrow K^0_{i+1}$, $i_1 \circ r_1 \sim 1_{\mathbb{K}_{i+1}}$ [3], where $1_{\mathbb{K}_{i+1}}$ is the identity over $\mathbb{K}_{i+1}$. As $i_1 \circ i_0 = i'$, we have $i' \circ f_i(\sigma) \sim i(\sigma)$.

Combining both the cases we conclude $i' \circ f_i \sim i$.

Since contiguous maps are homotopic at the level of geometric realizations, the diagram in the lemma commutes. ◀

Proceeding by induction, Lemma 3 then immediately implies the following result.

▶ **Lemma 5.** *Given a tower* $\mathcal{T} : K_0 \xrightarrow{f_0} K_1 \xrightarrow{f_1} ... \xrightarrow{f_{m-1}} K_m$. *For each* $0 \leq i \leq m$, $\mathbb{K}_i \searrow\searrow K_i$.

Again, using an inductive argument along with Lemmas 4 and 5, we can deduce the following result.

▶ **Theorem 6.** *The following diagram commutes and all the vertical maps* ϕ_i^* *are isomorphisms. As a consequence, the tower* $\mathcal{T} : K_0 \xrightarrow{f_0} K_1 \xrightarrow{f_1} ... \xrightarrow{f_{m-1}} K_m$ *and the constructed filtration* $\mathcal{F} : \mathbb{K}_0 \hookrightarrow \mathbb{K}_1 \hookrightarrow ... \hookrightarrow \mathbb{K}_m$ *have the same persistence diagram.*

$$\begin{array}{ccccccc} H_p(\mathbb{K}_1) & \xhookrightarrow{*} & H_p(\mathbb{K}_2) & \xhookrightarrow{*} & \cdots & \longrightarrow H_p(\mathbb{K}_{m-1}) & \xhookrightarrow{*} H_p(\mathbb{K}_m) \\ \downarrow{\scriptstyle \phi_1^*} & & \downarrow{\scriptstyle \phi_2^*} & & & \downarrow{\scriptstyle \phi_{m-1}^*} & \downarrow{\scriptstyle \phi_m^*} \\ H_p(K_1) & \xrightarrow{f_1^*} & H_p(K_2) & \xrightarrow{f_2^*} & \cdots & \longrightarrow H_p(K_{m-1}) & \xrightarrow{f_{m-1}^*} H_p(K_m) \end{array}$$

Here ϕ_i *is a strong collapse for each* $i \in \{0, \cdots, m\}$ *and* $*$ *indicates the induced homomorphisms.*

We summarize our result in the following theorem. We write $\mathcal{T} : K_0 \xrightarrow{f_0} K_1 \xrightarrow{f_1} ... \xrightarrow{f_{m-1}} K_m$ for the given flag tower where, w.l.o.g., $K_0 = \emptyset$. and each f_i is either an inclusion or an elementary contraction. The inclusions are not necessarily elementary but corresponds to an elementary inclusion on the graphs G_i. We denote by d the maximal dimension of the K_is in $\mathcal{T}$, and by n the total number of elementary inclusions of simplices in $\mathcal{T}$, by n_c the total number of elementary contractions and by n_0 the number of vertex inclusions in $\mathcal{T}$.

▶ **Theorem 7.** *There exists a filtration* $\mathcal{F} : \mathbb{K}_0 \hookrightarrow \mathbb{K}_1 \hookrightarrow ... \hookrightarrow \mathbb{K}_m$, *where the inclusions are not necessarily elementary, such that* $\mathcal{T}$ *and* $\mathcal{F}$ *have the same persistence diagram and the size of the filtration* $|\mathbb{K}_m|$ *is at most* $\mathcal{O}(d * n * \log n_0)$. *Moreover,* $\mathcal{F}$ *is a flag filtration which can be computed from* $\mathcal{T}$ *using only the 1-skeletons* G_i*s of the* K_i*s. The time complexity of the algorithm is* $O(|\mathbb{G}_m| + n_c * k^3)$ *time and its space complexity* $\mathcal{O}(n_0 * k)$, *where* $\mathbb{G}_m$ *denotes the 1-skeleton of* $\mathbb{K}_m$ *and* k *is an upper bound on the degree of the vertices in* $\mathbb{G}_m$.

5 Computational experiments

Our algorithm has been implemented for Vietoris-Rips (VR) filtrations as a C++ module named RipsCollapser. Recall that, for VR-filtrations, the filtration value of a simplex is the length of the longest edge of the simplex. RipsCollapser takes as input a VR filtration and returns the reduced flag filtration as shown above. RipsCollapser can be used in two modes: in the exact mode, the output filtration has the same PD as the input filtration while, in the approximate mode to be described below, a certified approximation is returned. The output filtration can then be sent to any software that computes the PD of a VR-filtration such as the Gudhi library (the one we chose for our experiments) [32] or Ripser [4]. Therefore RipsCollapser is to be considered as an ad-on to software computing PD. RipsCollapser will be available as an open-source package of a next release of the Gudhi library.

Approximate persistence diagram. Given a VR filtration, one can choose to collapse the original complexes after each edge inclusion. However, as mentioned in Section 2, we can also choose to strong collapse the complexes less often, i.e. after several edge inclusions rather than just one. This will result in a faster algorithm but comes with a cost: the computed PD

is then only approximate. We call **snapshots** the values of the scale parameter at which we choose to strong collapse the complex. The difference between two consecutive snapshots is called a **step**. We approximate the filtration value of a simplex as the value of the snapshot at which it first appears. We can observe that our algorithm will report all persistence pairs that are separated by at least one snapshot. Hence if all steps are equal to some $\epsilon > 0$, we will compute all the persistence pairs whose lengths are at least ϵ. It follows that the l_∞-*bottleneck distance* between the computed PD and the exact one is at most ϵ. If instead the ratio between any two consecutive steps is taken to a constant $\rho > 1$, the l_∞-bottleneck distance will be at most $\log \rho$ after reparameterizing the filtrations on a log-log-scale [45].

Experimental setup. We present results on five datasets **netw-sc**, **senate**, **eleg**, **HIV** and **drag 1** that are publicly available [18]. Each dataset is given as the interpoint distance matrix. The reported time includes the time of RipsCollapser and the time to compute the persistent diagram (PD). The time of RipsCollapser includes: 1. The time taken to compute the largest 1-skeleton associated to the maximum threshold value, 2. The time taken to collapse all the sub-skeletons and assemble their cores. 3. To transform them into an equivalent flag-filtration.

The code has been compiled using the compiler 'clang-900.0.38' and all computations were performed on a '2.8 GHz Intel Core i5' machine with 16 GB of available RAM. We took all steps to be equal. Parameter *Step* controls the quality of the approximation : if *Step* $= 0$, we obtain the exact PD, otherwise *Step* is an upper bound on the l_∞-*bottleneck distance* between the output diagram and the exact one. RipsCollapser works irrespective of the dimension of the input complexes. However, the size of the complexes in the reduced filtration, even if much smaller than in the original filtration, might exceed the capacities of the PD computation algorithm. For this reason, we introduced a parameter *dim* and restricts PD computation to dimension at most *dim*.

Some experimental results are reported in Table 1. We first observe that the reduction done by RipsCollapser is enormous. The reduced complexes are small and of low dimension (column Size/Dim) compared to the input VR-complexes which are of dimensions respectively 57, 54 and 105 for the first three datasets **netw-sc**, **senate** and **eleg**. We also observe that, while the time taken by RipsCollapser is large for exact persistence computation, very good approximations can be obtained fast. Moreover the computing time mildly increases with the number of snapshots. This suggests that implementing the collapses in parallel would lead to further substantial improvement.

Comparison with Ripser. Ripser [4] is the state of the art software to compute persistent diagrams of VR filtrations. It computes the *exact* PD associated to the input filtration up to dimension *dim*. Although RipsCollapser is more complementary to Ripser than a competitor, we run Ripser[1] on the same datasets as in Table 1 to demonstrate the benefit of using RipsCollapser.

Results are presented in Table 2. The main observation is that Ripser performs quite well in low dimensions but its ability to handle higher dimensions is limited.

[1] We used the command
`<./ripser inputData -format distances -threshold inputTh -dim inputDim >`.

Table 1 The columns are, from left to right: dataset (Data), number of points (Pnt), maximum value of the scale parameter (Thrsld), number of simplices (Size) and dimension of the final filtration (Dim), parameter (*dim*), time (in seconds) taken by RipsCollapser, total time (in seconds) including PD computation (Tot-Time), parameter *Step* and the number of snapshots used (Snaps).

Data	Pnt	Thrsld	RipsCollapser +PD					
			Size/Dim	*dim*	Pre-Time	Tot-Time	*Step*	Snaps
netw-sc	379	5.5	155/3	∞	7.28	7.38	0.02	263
”	”	”	155/3	∞	13.93	14.03	0.01	531
”	”	”	175/3	∞	366.46	366.56	0	8420
senate	103	0.415	405/4	∞	2.53	2.54	0.001	403
”	”	”	417/4	∞	15.96	15.98	0	2728
eleg	297	0.3	577K/15	∞	11.65	26.02	0.001	284
”	”	”	835K/16	∞	518.36	540.40	0	9850
HIV	1088	1050	127.3M/?	4	660	3,955	4	184
drag1	1000	0.05	478.3M/?	4	687	14,170	0.0002	249

Table 2 Time is the total time (in seconds) taken by Ripser. ∞ means that the experiment ran longer than 12 hours or crashed due to memory overload.

Data	Pnt	Threshold	Val		Val		Val	
			dim	Time	*dim*	Time	*dim*	Time
netw-sc	379	5.5	4	25.3	5	231.2	6	∞
senate	103	0.415	3	0.52	4	5.9	5	52.3
”	”	”	6	406.8	7	∞		
eleg	297	0.3	3	8.9	4	217	5	∞
HIV	1088	1050	2	31.35	3	∞		
drag1	1000	0.05	3	249	4	∞		

References

1 M. Adamaszek and J. Stacho. Complexity of simplicial homology and independence complexes of chordal graphs. *Computational Geometry: Theory and Applications*, 57:8–18, 2016.

2 D. Attali, A. Lieutier, and D. Salinas. Efficient data structure for representing and simplifying simplicial complexes in high dimensions. *International Journal of Computational Geometry and Applications (IJCGA)*, 22:279–303, 2012.

3 J. A. Barmak and E. G. Minian. Strong homotopy types, nerves and collapses. *Discrete and Computational Geometry*, 47:301–328, 2012.

4 U. Bauer. Ripser. URL: `https://github.com/Ripser/ripser`.

5 U. Bauer, M. Kerber, and J. Reininghaus. Clear and compress: Computing persistent homology in chunks. In *Topological Methods in Data Analysis and Visualization III, Mathematics and Visualization*, pages 103–117. Springer, 2014.

6 U. Bauer, M. Kerber, J. Reininghaus, and H. Wagner. PHAT – persistent homology algorithms toolbox. *Journal of Symbolic Computation*, 78, 2017.

7 J-D. Boissonnat and C. S. Karthik. An Efficient Representation for Filtrations of Simplicial Complexes. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2017.

8 J-D. Boissonnat, S.Pritam, and D. Pareek. Strong Collapse for Persistence. In *26th Annual European Symposium on Algorithms (ESA 2018)*, volume 112, 2018.

9 M. Botnan and G. Spreemann. Approximating persistent homology in Euclidean space through collapses. *Applicable Algebra in Engineering, Communication and Computing*, 26:73–101, 2015.

10 G. Carlsson and V. de Silva. Zigzag Persistence. *Found Comput Math*, 10, 2010.

11 G. Carlsson, V. de Silva, and D. Morozov. Zigzag persistent homology and real-valued functions. *SOCG*, pages 247–256, 2009.

12 G. Carlsson, T. Ishkhanov, V. de Silva, and A. Zomorodian. On the Local Behavior of Spaces of Natural Images. *In: International Journal of Computer Vision*, 76:1–12, 2008.

13 J. M. Chan, G. Carlsson, and R. Rabadan. Topology of viral evolution. *In: Proceedings of the National Academy of Sciences*, 110:18566–18571, 2013.

14 F. Chazal and S. Oudot. Towards Persistence-Based Reconstruction in Euclidean Spaces. *SOCG*, 2008.

15 C. Chen and M. Kerber. Persistent homology computation with a twist. *In European Workshop on Computational Geometry (EuroCG)*, pages 197–200, 2011.

16 Aruni Choudhary, Michael Kerber, and Sharath Raghvendra. Polynomial-Sized Topological Approximations Using the Permutahedron. In Sándor Fekete and Anna Lubiw, editors, *32nd International Symposium on Computational Geometry (SoCG 2016)*, volume 51 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 31:1–31:16, Dagstuhl, Germany, 2016. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. URL: `http://drops.dagstuhl.de/opus/volltexte/2016/5923`, `doi:10.4230/LIPIcs.SoCG.2016.31`.

17 H. Edelsbrunner D. Cohen-Steiner and J. Harer. Stability of Persistence Diagrams. *Discrete and Compututaional Geometry*, 37:103–120, 2007.

18 Datasets. URL: `https://github.com/n-otter/PH-roadmap/''`.

19 V. de Silva and R. Ghrist. Coverage in sensor networks via persistent homology. *In: Algebraic and Geometric Topology*, 7:339 – 358, 2007.

20 H. Derksen and J. Weyman. Quiver representations. *Notices of the American Mathematical Society*, 52(2):200–206, February 2005.

21 T. K. Dey, H. Edelsbrunner, S. Guha, and D. Nekhayev. Topology preserving edge contraction. *Publications de l'Institut Mathematique (Beograd)*, 60:23–45, 1999.

22 T. K. Dey, F. Fan, and Y. Wang. Computing Topological Persistence for Simplicial Maps. In *Symposium on Computational Geometry (SoCG*, pages 345–354, 2014.

23 T. K. Dey, D. Shi, and Y. Wang. *SimBa: An efficient tool for approximating Rips-filtration persistence via Simplicial Batch-collapse*. In European Symp. on Algorithms (ESA), pages 35:1–35:16, 2016.

24 T. K. Dey and R. Slechta. Filtration Simplification for Persistent Homology via Edge Contraction. *International Conference on Discrete Geometry for Computer Imagery*, 2019.

25 C. H. Dowker. Homology groups of relations. *The Annals of Mathematics*, 56:84–95, 1952.

26 P. Dłotko and H. Wagner. Simplification of complexes for persistent homology computations,. *Homology, Homotopy and Applications*, 16:49–63, 2014.

27 H. Edelsbrunner and J. Harer. *Computational Topology: An Introduction.* American Mathematical Society, 2010.

28 H. Edelsbrunner, D. Letscher, and A. Zomorodian. Topological persistence and simplification. *Discrete and Compututational Geometry*, 28:511–533, 2002.

29 B. T. Fasy, J. Kim, F. Lecci, and C. Maria:. Introduction to the R package TDA. *CoRR abs/1411.1830*, 2014.

30 E. Fieux and J. Lacaze. Foldings in graphs and relations with simplicial complexes and posets. *Discrete Mathematics*, 312(17):2639–2651, 2012.

31 F. Le Gall. Powers of tensors and fast matrix multiplication. *ISSAC '*, 14:296–303, 2014.

32 Gudhi: Geometry understanding in Higher Dimensions. URL: `http://gudhi.gforge.inria.fr/`.

33 A. Hatcher. *Algebraic Topology.* Univ. Press Cambridge, 2001.

34 C. S. Karthik J-D. Boissonnat and S. Tavenas. Building Efficient and Compact Data Structures for Simplicial Complexes. *Algorithmica*, 79:530–567, 2017.

35 M. Kerber and H. Schreiber:. Barcodes of Towers and a Streaming Algorithm for Persistent Homology. *33rd International Symposium on Computational Geometry*, 2017. `arXiv:1701.02208`.

36 M. Kerber and R. Sharathkumar. Approximate Čech Complex in Low and High Dimensions. In *Algorithms and Computation*, pages 666–676. by Leizhen Cai, Siu-Wing Cheng, and Tak-Wah Lam. Vol. 8283. Lecture Notes in Computer Science, 2013.

37 C. Maria and S. Oudot. Zigzag Persistence via Reflections and Transpositions. In *Proc. ACM-SIAM Symposium on Discrete Algorithms (SODA) pp.* 181–199, January 2015.

38 N. Milosavljevic, D. Morozov, and P. Skraba. Zigzag persistent homology in matrix multiplication time. In *Symposium on Computational Geometry (SoCG)*, 2011.

39 K. Mischaikow and V. Nanda. Morse Theory for Filtrations and Efficient Computation of Persistent Homology. *Discrete and Computational Geometry*, 50:330–353, September 2013.

40 D. Mozozov. Dionysus. URL: `http://www.mrzv.org/software/dionysus/`.

41 J. Munkres. *Elements of Algebraic Topology.* Perseus Publishing, 1984.

42 N. Otter, M. Porter, U. Tillmann, P. Grindrod, and H. Harrington. A roadmap for the computation of persistent homology. *EPJ Data Science, Springer Nature*, page 6:17, 2017.

43 J. Perea and G. Carlsson. A Klein-Bottle-Based Dictionary for Texture Representation. *In: International Journal of Computer Vision*, 107:75–97, 2014.

44 H. Schreiber. Sophia. URL: `https://bitbucket.org/schreiberh/sophia/`.

45 D. Sheehy. Linear-Size Approximations to the Vietoris–Rips Filtration. *Discrete and Computational Geometry*, 49:778–796, 2013.

46 M. Tancer. Recognition of collapsible complexes is NP-complete. *Discrete and Computational Geometry*, 55:21–38, 2016.

47 J. H. C Whitehead. Simplicial spaces nuclei and m-groups. *Proc. London Math. Soc*, 45:243–327, 1939.

48 A. C. Wilkerson, H. Chintakunta, and H. Krim. Computing persistent features in big data: A distributed dimension reduction approach. In *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 11–15, 2014.

49 A. C. Wilkerson, T. J. Moore, A. Swami, and A. H. Krim. Simplifying the homology of networks via strong collapses. In *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 11–15, 2013.

50 A. Zomorodian. The Tidy Set: A Minimal Simplicial Set for Computing Homology of Clique Complexes. In *Proceedings of the Twenty-sixth Annual Symposium on Computational Geometry*, pages 257–266, 2010.

51 A. Zomorodian and G. Carlsson. Computing persistent homology. *Discrete and Computational Geometry*, 33:249–274, 2005.