



HAL
open science

Computing Persistent Homology of Flag Complexes via Strong Collapses

Jean-Daniel Boissonnat, Siddharth Pritam

► **To cite this version:**

Jean-Daniel Boissonnat, Siddharth Pritam. Computing Persistent Homology of Flag Complexes via Strong Collapses. 2018. hal-01950074

HAL Id: hal-01950074

<https://inria.hal.science/hal-01950074v1>

Preprint submitted on 10 Dec 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Computing Persistent Homology of Flag Complexes via Strong Collapses*

Jean-Daniel Boissonnat

Université Côte d’Azur, INRIA, France

Jean-Daniel.Boissonnat@inria.fr

Siddharth Pritam

Université Côte d’Azur, INRIA, France

siddharth.pritam@inria.fr

1 — Abstract —

2 This paper is a continuation of the research reported in [7] on the usage of strong collapses to
3 accelerate the computation of persistent homology (PH). We show that further decisive progress
4 can be obtained if one restricts the family of simplicial complexes to flag complexes. The resulting
5 method is simple and extremely efficient.

2012 ACM Subject Classification Mathematics of computing, Topological Data Analysis, Computational geometry

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

Lines 493

6 **1** Introduction

7 In this article, we address the problem of computing the Persistent Homology (PH) of a
8 given sequence of simplicial complexes in an efficient way. It is known that computing
9 persistence can be done in $O(n^\omega)$ time, where n is the total number of simplices and $\omega \leq 2.4$
10 is the matrix multiplication exponent [36, 30]. In practice, when dealing with massive and
11 high-dimensional datasets, n can be very large (of order of billions) and computing PH is
12 then very slow and memory intensive. Improving the performance of PH computation is
13 therefore of utmost importance.

14 Much progress has been accomplished in the recent years in two directions. First, a
15 number of clever implementations and optimizations have led to a new generation of software
16 for PH computation [31, 45, 5, 38]. Secondly, a complementary direction has been explored
17 to reduce the size of the complexes in the sequence while preserving or approximating in a
18 controlled way the persistent homology of the sequence. Examples are the work of Mischaikow
19 and Nanda [37] who use Morse theory to reduce the size of a filtration, and the work of Dłotko
20 and Wagner who use simple collapses [25]. Both methods compute the exact PH of the input
21 sequence. Approximations can also be computed with theoretical guarantees. Approaches
22 like interleaving with smaller and easily computable simplicial complexes, or sub-sampling
23 the point sample work well upto certain approximation factors [14, 9, 43, 34, 16, 23].

24 This paper is a continuation of the research reported in [7] on the usage of strong collapses
25 to accelerate the computation of the persistent homology of a sequence of simplicial complexes.

* 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).



26 The basic idea is to simplify the complexes of the input sequence by using strong collapses,
 27 as introduced by J. Barmak and E. Miniam [3], and to compute the PH of an induced
 28 sequence of reduced simplicial complexes that has the same PH as the initial one. A crucial
 29 advantage of the method is that it only needs to store the maximal simplices of the complex,
 30 not the full set of the simplices of all dimensions, which saves space and time by a factor
 31 that is exponential in the dimension of the complex in the worst-case. As a result and as
 32 demonstrated by numerous experiments on publicly available data sets, the approach is very
 33 fast and memory efficient in practice.

34 In this paper, we build on the initial success of [7] and show that further decisive progress
 35 can be obtained if one restricts the family of simplicial complexes to flag complexes. Flag
 36 complexes are fully characterized by their graph (or 1-skeleton), the other faces being obtained
 37 by computing the cliques of the graph. Hence, a flag complex can be represented by its
 38 1-skeleton, which is a very compact representation. Flag complexes are very popular and,
 39 in particular, Vietoris-Rips complexes are by far the most widely used simplicial complexes
 40 in Topological Data Analysis. It has been shown in [7] that the persistent homology of
 41 Vietoris-Rips filtrations can be computed very efficiently using strong collapses. However,
 42 most of the time was devoted to computing the maximal cliques of the complex prior to
 43 their strong collapse. In this paper, we show that the reduced complex obtained by strong
 44 collapsing a flag complex is itself a flag complex. Moreover, this reduced complex can be
 45 computed using only the 1-skeleton of the complex and does not require to compute the set
 46 of its maximal cliques. Finally, we show how to compute the induced sequence of reduced
 47 simplicial complexes using again only the 1-skeleton.

48 The resulting method is simple and extremely efficient. On the theory side, we show
 49 that strong collapses can be computed in time $O(k^2v^2)$ where v is the number of vertices of
 50 the complex and k the maximal degree of its graph. The algorithm described in this paper
 51 has been implemented. Numerous experiments show that the computation of the persistent
 52 homology of flag complexes can be obtained much faster than with previous methods, e.g.
 53 Ripser [5]. The code will be soon released in the Gudhi library [31].

54 2 Preliminaries

55 In this section, we provide a brief review of the notions of simplicial complex and strong
 56 collapse as introduced in [3]. We assume some familiarity with basic concepts like homotopic
 57 maps, homotopy type, homology groups and other algebraic topological notions. Readers
 58 can refer to [32] for a comprehensive introduction to these topics.

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

66 A vertex to vertex map $\psi : K \rightarrow L$ between two simplicial complexes is called a **simplicial**
 67 **map**, if the images of the vertices of a simplex always span a simplex. Simplicial maps are thus
 68 determined by the images of the vertices. In particular, there is a finite number of simplicial
 69 maps between two given finite simplicial complexes. Simplicial maps induce continuous maps
 70 between the underlying geometric realisations of the simplicial complexes. Two simplicial

71 maps $\phi : K \rightarrow L$ and $\psi : K \rightarrow L$ are **contiguous** if, for all $\sigma \in K$, $\phi(\sigma) \cup \psi(\sigma) \in L$. Two
72 contiguous maps are known to be homotopic [39, Theorem 12.5].

73 **Flag complex:** A complex K is a flag or a clique complex if, when a set of its vertices have
74 pair wise edges between them, they span a simplex. It follows that the full structure of K is
75 determined by its 1-skeleton we denote by G . For a vertex v in G , the **open neighborhood**
76 $N_G(v)$ of v in G is defined as $N_G(v) := \{u \in G \mid [uv] \in E\}$. The **closed neighborhood**
77 $N_G[v]$ is $N_G[v] := N_G(v) \cup \{v\}$. We further define the relative closed neighborhood of u by v
78 in G as the set of vertices in $N_G[u]$ that are not in $N_G[v]$. We denote it by $N_G[u \setminus v]$.

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

83 Taking a join with a vertex transforms a simplicial complex into a **simplicial cone**.
84 Formally if L is a simplicial complex and a is a vertex not in L then the simplicial cone aL
85 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
86 **dominated vertex** if the link of v in K , $lk_K(v)$ is a simplicial cone, that is, there exists a
87 vertex $v' \neq v$ and a subcomplex L in K , such that $lk_K(v) = v'L$. We say that the vertex v'
88 is dominating v and v is dominated by v' . The symbol $K \setminus v$ (deletion of v from K) refers
89 to the subcomplex of K which has all simplices of K except the ones containing v . Below
90 is an important remark from [3, Remark 2.2], which proposes an alternative definition of
91 dominated vertices.

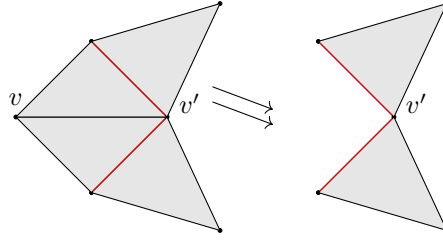
92 **Remark 1:** A vertex $v \in K$ is dominated by another vertex $v' \in K$, *if and only if* all
93 the maximal simplices of K that contain v also contain v' [3].

94 **Strong collapse:** An **elementary strong collapse** is the deletion of a dominated vertex
95 v from K , which we denote with $K \searrow \searrow K \setminus v$. Figure 1 illustrates an easy case of an
96 elementary strong collapse. There is a **strong collapse** from a simplicial complex K to its
97 subcomplex L , if there exists a series of elementary strong collapses from K to L , denoted as
98 $K \searrow \searrow L$. The inverse of a strong collapse is called a **strong expansion**. If there exists a
99 combination of strong collapses and/or strong expansions from K to L , then K and L are
100 said to have the same **strong homotopy type**.

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

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

112 **Retraction map:** If a vertex $v \in K$ is dominated by another vertex $v' \in K$, the vertex map
113 $r : K \rightarrow K \setminus v$ defined as: $r(w) = w$ if $w \neq v$ and $r(v) = v'$, induces a simplicial map that is a
114 *retraction* map. The homotopy between r and the identity $i_{K \setminus v}$ over $K \setminus v$ is in fact a strong
115 deformation retract. Furthermore, the composition $(i_{K \setminus v})r$ is contiguous to the identity i_K
116 over K [3, Proposition 2.9].



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

117 **Persistent homology** A sequence of simplicial complexes $\mathcal{T} : \{K_1 \xrightarrow{f_1} K_2 \xrightarrow{f_2} K_3 \xrightarrow{f_3} \dots \xrightarrow{f_{(m-1)}} K_m\}$,
 118 connected through simplicial maps f_i s is called a **simplicial tower** or
 119 simply a *tower*. We call a tower a **flag tower** if all the simplicial complexes K_i are flag
 120 complexes. When all the simplicial maps f_i s are inclusions, then the tower is called a filtration
 121 and a flag tower will be called a **flag filtration**.

122 If we compute the homology classes of all the K_i , we get the sequence $\mathcal{P}(\mathcal{T}) : \{H_p(K_1) \xrightarrow{f_1^*} H_p(K_2) \xrightarrow{f_2^*} H_p(K_3) \xrightarrow{f_3^*} \dots \xrightarrow{f_{(m-1)}^*} H_p(K_m)\}$. Here $H_p(\cdot)$ denotes the homology class of
 123 dimension p with coefficients from a field \mathbb{F} and $*$ denotes an induced homomorphism. $\mathcal{P}(\mathcal{T})$
 124 is a sequence of vector spaces connected through homomorphisms, called a **persistence**
 125 **module**. More formally, a *persistence module* \mathbb{V} is a sequence of vector spaces $\{V_1 \rightarrow V_2 \rightarrow V_3 \rightarrow \dots \rightarrow V_m\}$
 126 connected with homomorphisms $\{\rightarrow\}$ between them. A persistence module
 127 arising from a sequence of simplicial complexes captures the evolution of the topology of the
 128 sequence.
 129

130 Any persistence module can be *decomposed* into a collection of intervals of the form $[i, j)$
 131 [10]. The multiset of all the intervals $[i, j)$ in this decomposition is called the **persistence**
 132 **diagram** of the persistence module. An interval of the form $[i, j)$ in the persistence diagram
 133 of $\mathcal{P}(\mathcal{T})$ corresponds to a homological feature (a ‘cycle’) which appeared at i and disappeared
 134 at j . The persistence diagram completely characterizes the persistence module, that is, there
 135 is a bijective correspondence between them [10, 49].

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

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

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

142 2.1 Overview of the algorithm

143 The algorithm presented in this paper adopts the same strategy as the algorithm reported
 144 in [7]. By focussing on *flag towers* instead of general sequences of simplicial complexes, we
 145 obtain a significantly more efficient algorithm regarding space and time complexity.

146 Let $\mathcal{T} : \{K_1 \xrightarrow{f_1} K_2 \xrightarrow{f_2} \dots \xrightarrow{f_{(m-1)}} K_m\}$ be a flag tower of which we want to compute
 147 the persistence diagram. We first compute a *reduced core tower* by computing the cores of
 148 each K_i and connecting them through induced simplicial maps as suggested in [7]. Since each
 149 K_i is a flag complex, the computation of its core can be computed using only the 1-skeleton
 150 of the complex. This is much more efficient than what is done in [7] where the complex is
 151 represented by its maximal faces. The new algorithm is discussed in detail in Section 3.

152 We then compute an associated flag *filtration* we can send to any algorithm that computes
 153 the persistence homology of a flag filtration [38, 5, 45, 31]. This is similar to what has been
 154 done in [21, 33] and [7], and is detailed in Section 4.

155 **3 Strong Collapse of a Flag complex**

156 In this section, we show that the core of a flag complex K is itself a flag complex whose graph
 157 is called the *core graph* of K . The core graph of K can be computed from the 1-skeleton G
 158 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
 159 the degree of G (i.e. the number of edges that are incident on a vertex in K).

160 Although this change wrt to the algorithm in [7] might look minor, it is crucial in
 161 practice as the time to compute all the maximal simplices of a flag complex from its graph is
 162 exponential in the number of its vertices. We thus reduce immensely the time and space
 163 complexity of the general algorithm of [7] whose complexity is $\mathcal{O}(v^2\Gamma_0d + m^2\Gamma_0d)$, where Γ_0
 164 is an upper bound on the number of *maximal simplices* incident to a vertex.

165 In the following lemma, we describe a condition in terms of the closed neighborhood
 166 $N_G[v]$ of a vertex v of a flag complex K under which v will be dominated by another vertex
 167 v' of K . This result has been independently studied in [29, Lemma 4.1].

168 **► Lemma 1.** *Let K be a flag complex. A vertex $v \in K$ is dominated by v' iff $N_G[v] \subseteq N_G[v']$.*

169 **Proof.** If v is dominated by v' , then, according to Remark 1, all the maximal simplices that
 170 contain v also contain v' . Since K is a flag complex, it follows that all edges incident on v
 171 are also incident on v' . In other words, $N_G[v] \subseteq N_G[v']$.

172 Now we prove the other direction. Let σ be a maximal simplex of K containing v . Any
 173 other vertex x of σ is joined to v by an edge $[x, v] \in \sigma$. Moreover, since $N_G[v] \subseteq N_G[v']$,
 174 $[v, v']$ and $[x, v']$ are in K . It follows that every vertex in σ has an edge with both v and v'
 175 and, since K is a flag complex and σ is maximal, v' must be in σ . This implies that all the
 176 maximal simplices that contains v also contains v' . Hence v is dominated by v' . ◀

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

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

185 **Data structure:** We represent G with its adjacency matrix M , where the rows and the
 186 columns of M represent the vertices of G . An entry $M[v_i][v_j]$ associated with vertices v_i and
 187 v_j is set to 1 if either the edge $[v_i, v_j] \in G$ or $i = j$, and to 0 otherwise. We will say that
 188 a row v is contained in another row v' if the set of indices of the non-zero entries of v is a

189 subset of the indices of the non-zero entries of v' . It is clear that if a row v is contained
 190 in another row v' , we have $N_G[v] \subseteq N_G[v']$ and therefore the vertex v is dominated by the
 191 vertex v'

192 **Core algorithm:** Given the adjacency matrix M of G , we compute the adjacency matrix C
 193 of the core graph G^c . In view of Lemma 1, we can easily compute C from M using basic
 194 row removal operations. Loosely speaking, we remove the rows of M that are contained in
 195 another row. After removing the row associated to v , we simultaneously update the matrix
 196 by removing the column associated to v . The process is iterated as long as the matrix can be
 197 reduced. Upon termination, we output the reduced matrix C , which is the adjacency matrix
 198 of the core graph G^c of K . Since the core of a complex is always unique, the order in which
 199 vertices are removed does not matter [3].

200 **Computing the retraction map r :** The algorithm also provides a direct way to compute
 201 the retraction map r defined in Section 2. It can be constructed as follows. A row v
 202 being removed in M corresponds to a dominated vertex in K and the row which contains
 203 v corresponds to a dominating vertex. Therefore we map the dominated vertex to the
 204 dominating vertex.

205 **Reducing the number of domination tests:** We first observe that, when one wants to
 206 determine if a row v is dominated by some other row v' , we don't need to test v with all
 207 other rows but only with its neighbors, i.e. at most k of them. Here k is the upper bound on
 208 the degree of the vertices in G .

209 A second observation is that we don't need to test all rows for domination, but only the
 210 so-called candidate rows. We define a row v to be a **candidate row** for the next iteration if
 211 at least one of its neighbors has been removed in a previous row removal iteration. Candidate
 212 rows are the only rows that need to be considered in the domination tests of the algorithm.
 213 Indeed, a row w of M whose neighbors are present from the previous *iteration* cannot
 214 be dominated by another row v' of M , as w was not dominated in the previous iteration
 215 and all other vertices can only lose their neighbors. This ensures that v will still remain
 216 un-dominated.

217 We maintain a *queue*, for the candidate rows (rowQueue). These queues are implemented
 218 as First in First out (FIFO) queues. At each iteration, we *pop out* a candidate row from
 219 rowQueue and test whether it is dominated or not. After each successful domination test,
 220 we push the new candidate rows in the queue in preparation for the subsequent iteration. In
 221 the first iteration, we push all the rows in rowQueue. Algorithm 1 gives the pseudo code of
 222 our algorithm.

239 **Time Complexity:** The most basic operation in our algorithm is to determine if a row is
 240 dominated by another given row. In our implementation, the rows of the matrix that are
 241 considered by the algorithm are stored as sorted lists. Checking if one sorted list is a subset
 242 of another sorted list can be done in time $\mathcal{O}(l)$, where l is the size of the longer list. Note
 243 that the length of a row list is at most $k + 1$ where k denotes an upper bound on the degree
 244 of any vertex. Hence checking if a row is dominated by another row takes $\mathcal{O}(k)$ time.

245 At each iteration on the rows (Lines 6-12 of Algorithm 1), each row is checked against at
 246 most k other rows (maximum number of neighbors). Moreover, since at each iteration on
 247 the rows we remove at least one row, the total number of iterations on the rows is at most

223 **Algorithm 1** Core algorithm

224 1: **procedure** CORE(M) \triangleright Returns the adjacency matrix corresponding to the core of K

225 2: $rowQueue \leftarrow$ *push* all rows of M (all vertices of K)

226 3: **while** $rowQueue$ is not empty **do**

227 4: $v \leftarrow pop(rowQueue)$

228 5: $N_G[v] \leftarrow$ the non-zero columns of v

229 6: **for** w in $N_G[v]$ **do**

230 7: **if** $N_G[v] \subseteq N_G[w]$ **then**

231 8: Remove v from M \triangleright Both the associated column and the associated row

232 9: *push* all the entries of $N_G(v)$ to $rowQueue$ if not pushed before

233 10: **break**

234 11: **end if**

235 12: **end for**

236 13: **end while**

237 14: **return** M \triangleright The core consists of the remaining rows and columns

238 15: **end procedure**

248 $O(v^2)$, where v is the total number of vertices of the complex K . Therefore, the worst-case
 249 time complexity of our algorithm is $\mathcal{O}(v^2k^2)$.

250 4 Flag Tower to Flag Filtration

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

254 4.1 Previous work

255 It is well known that any general simplicial map can be decomposed into more elementary
 256 simplicial maps, namely elementary inclusions (i.e., inclusions of a single simplex) and
 257 elementary contractions (where a vertex is mapped onto another vertex). It can be observed
 258 that if we can replace an elementary contraction $\{\{u, v\} \mapsto u\}$ with an equivalent (not
 259 necessarily elementary) inclusion, we thus transform a tower into an equivalent filtration.
 260 This was the philosophy introduced by Dey et. al. in [21]. Given an elementary contraction
 261 $K_i \xrightarrow{\{u, v\} \mapsto u} K_{i+1}$, their strategy was to fill the simplices around the edge $[uv]$ such that it
 262 satisfies the *link condition*. Let $\hat{K}_{i+1} := \{K_i \cup S\}$, where S contains the missing simplices
 263 around $[uv]$ such that $[uv]$ satisfies the link condition in \hat{K}_{i+1} . The crucial observation is that
 264 including S does not create new topological changes. More precisely, the following diagram
 265 commutes and the inclusion $(i')^* : H_p(K_{i+1}) \hookrightarrow H_p(\hat{K}_{i+1})$ is an isomorphism.

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

267 Applying this construction as many times as required, we can see that any tower can be
 268 transformed into an equivalent filtration.

269 The work of Kerber and Schreiber [33] uses a slightly different approach where instead of
 270 the link condition they use a coning strategy. They define $\hat{K}_{i+1} := \{K_i \cup (u * St_{K_i}(v))\}$. The
 271 differently defined \hat{K}_{i+1} and the associated inclusion map $(i')^*$ also satisfy the aforementioned
 272 commutativity and isomorphism. However, if one uses the coning strategy naively, the size
 273 of the final resultant filtration may not be optimal. They address this issue using two crucial
 274 observations. They first observe that one does not need to cone u with the complete star
 275 $st_{K_i}(v)$ of v but only with a subset $ActSt_{K_i}(v)$ of it, called an active star. The **active star**
 276 $ActSt_{K_i}(v)$ of a vertex v in K_i is the set of simplices in the star $st_{K_i}(v)$ that doesn't contain
 277 any vertex whose star has been coned before. This is implemented as follows. Initially, all
 278 vertices are marked as active. Then, after the coning $\hat{K}_{i+1} := \{K_i \cup (u * St_{K_i}(v))\}$, the vertex
 279 v is marked *inactive*. They also observed that mapping $\{\{u, v\} \mapsto u\}$ or $\{\{u, v\} \mapsto v\}$ yields
 280 isomorphic simplicial complexes \hat{K}_i upto renaming the vertices. Choosing the representative
 281 whose active star is smaller can lead to a smaller complex \hat{K}_{i+1} . Finally, they prove that,
 282 given a tower $\mathcal{T} : K_0 \xrightarrow{f_0} K_1 \xrightarrow{f_1} \dots \xrightarrow{f_{m-1}} K_m$ of elementary inclusions and elementary
 283 contractions, the size of the equivalent filtration is $\mathcal{O}(d * n * \log n_0)$, where d is the maximal
 284 dimension of the K_i s, n the total number of elementary inclusions in \mathcal{T} and n_0 is the number
 285 of vertices included in \mathcal{T} [33, Theorem 2]. Further they show that the time and space
 286 complexities of their algorithm are $\mathcal{O}(d * |\hat{K}_m| * \mathbb{C}_\omega)$ and $\mathcal{O}(d * \omega)$ respectively, where \mathbb{C}_ω is
 287 the cost of an operation in a dictionary with ω elements, where ω is the maximal number of
 288 simplices in K_i s [33, Theorem 2].

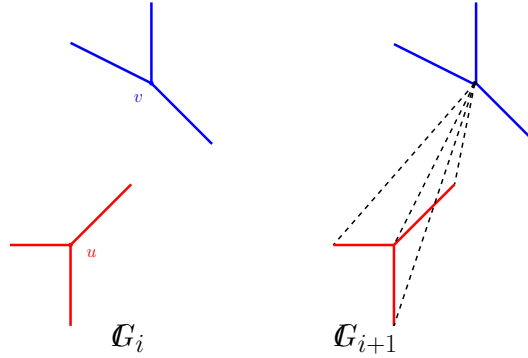
289 4.2 A new construction

290 Our work builds upon the above mentioned previous works [21, 33]. The difference is that
 291 we use strong expansions instead of coning, a strong expansion being the inverse operation of
 292 a strong collapse. The main advantage of strong expansions is that, when the input is a flag
 293 tower, we can use the domination criterion of Lemma 1. This leads to a simple algorithm
 294 that only deals with edges. The final filtration is a flag filtration, which can be represented
 295 very compactly. Moreover, a strong expansion being a coning, we will be able to use the
 296 theoretical results of [33]. Now we describe our construction.

297 Let K_i be a flag complex and G_i be its 1-skeleton for $i = 0, \dots, m$. As in [21, 33], we
 298 associate to K_i an augmented complex noted \mathbb{K}_i which plays a role similar to \hat{K}_i in the
 299 previous subsection. As will be seen below, \mathbb{K}_i is also a flag complex whose 1-skeleton will
 300 be denoted by \mathbb{G}_i . Following the terminology of [33], we call a vertex $v \in \mathbb{K}_i$ to be **active**
 301 if it is currently not *dominated* and has never been *contracted* before. The active closed
 302 neighborhood $ActN_{G_i}[v]$ is then defined as the set of all active vertices in $N_{G_i}[v]$. Similarly,
 303 $ActN_{G_i}[v \setminus u]$ denotes the set of active vertices in the closed neighborhood $N_{G_i}[v]$ of v that
 304 are not in $N_{G_i}[u]$. Finally, let $\{[u, ActN_{G_i}[v \setminus u]]\}$ denote the set of edges between u and
 305 $ActN_{G_i}[v \setminus u]$.

306 Using the notions defined above, we now explain how to inductively construct a filtration
 307 associated to a given flag tower. For $i = 0$, we set $\mathbb{G}_0 = \emptyset$. We then define \mathbb{G}_i as follows.

- 308 ■ if $G_i \xrightarrow{\cup \sigma} G_{i+1}$ is an elementary *inclusion* where σ is either a vertex or an edge, we set
 309 $\mathbb{G}_{i+1} := \mathbb{G}_i \cup \sigma$.
- 310 ■ if $G_i \xrightarrow{\{u, v\} \mapsto u} G_{i+1}$ is an elementary *contraction*
 - 311 ■ 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
 312 contracted
 - 313 ■ otherwise, we set $\mathbb{G}_{i+1} := \mathbb{G}_i \cup \{[v, ActN_{\mathbb{G}_i}[u \setminus v]]\}$ and u as contracted.



314

315 There are two different kinds of inactive vertices; the ones that have been marked as
 316 *contracted* and the others which are currently dominated. The first type is permanent while
 317 the second type may change over time. Observe that the construction checks for domination
 318 only before the elementary contractions not before the elementary inclusions. Therefore,
 319 before each such contraction $\{u, v\} \mapsto u$, we visit all the vertices to see if it is dominated
 320 in G_i . In fact, we only consider the vertices that have gained new edges after the previous
 321 contraction. Indeed, they are the only ones whose neighborhood has changed and can start
 322 dominating other vertices. To implement this optimization, we maintain and update a flag for
 323 each vertex that indicates whether a new edge became incident on it after the last contraction.
 324 We continue the construction until the end of our input tower.

325 **► Lemma 2.** Let $f_i : K_i \xrightarrow{\{u,v\} \mapsto u} K_{i+1}$ be the first elementary contraction in the tower
 326 $\mathcal{T} : K_0 \xrightarrow{f_0} K_1 \xrightarrow{f_1} \dots \xrightarrow{f_{m-1}} K_m$. Then the complex K_{i+1} is a subcomplex of \mathbb{K}_{i+1} and
 327 $\mathbb{K}_{i+1} \searrow \swarrow K_{i+1}$.

328 **Proof.** We prove the second part $\mathbb{K}_{i+1} \searrow \swarrow K_{i+1}$ of the statement which then implies
 329 the first part $K_{i+1} \subset \mathbb{K}_{i+1}$. Since f_i is the first contraction $\mathbb{K}_i = K_i$ and $G_i = G_i$. Let
 330 $G_{i+1} := G_i \cup \{[u, ActN_{G_i}[v \setminus u]]\}$ be the graph defined above. By construction, contracting
 331 the pair $\{u, v\}$ to u in both the graphs G_i, G_{i+1} yields the same graph G_{i+1} .

332 Let $x' \in ActN_{G_i}[v \setminus u]$. We observe that adding the edge $[ux']$ in G_i does not change the
 333 domination of $x \in \{N_{G_i}[v \setminus u] \setminus ActN_{G_i}[v \setminus u]\}$ as it only add neighbors in $N_{G_i}[x']$ and $N_{G_i}[u]$.
 334 Therefore, adding the edges $\{[u, ActN_{G_i}[v \setminus u]]\}$ to G_i does not change the domination status
 335 of the vertices of \mathbb{K}_{i+1} that are in the set $\{N_{G_i}[v \setminus u] \setminus ActN_{G_i}[v \setminus u]\}$. Therefore, removing
 336 all the dominated vertices in $N_{G_i}[v \setminus u]$ provides a sequence of elementary strong collapses.
 337 By performing all such elementary strong collapses, \mathbb{K}_{i+1} is eventually transformed into
 338 a complex K_{i+1}^0 . Moreover, we have removed all the dominated vertices from $N_{G_i}[v \setminus u]$
 339 and added edges between u and the remaining vertices in $ActN_{G_i}[v]$. This implies that v
 340 is dominated by u in K_{i+1}^0 . Collapsing v onto u , implies $K_{i+1}^0 \searrow \swarrow K_{i+1}$ and therefore
 341 $\mathbb{K}_{i+1} \searrow \swarrow K_{i+1}$.

342

343 **► Lemma 3.** Let $f_i : K_i \xrightarrow{\{u,v\} \mapsto u} K_{i+1}$ be the first elementary contraction in the tower
 344 $\mathbb{T} : K_0 \xrightarrow{f_0} K_1 \xrightarrow{f_1} \dots \xrightarrow{f_{m-1}} K_m$. Then the following diagram commutes

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

345

XX:10 Computing Persistent Homology of Flag Complexes via Strong Collapses

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

348 **Proof.** As mentioned before, since f_i is the first contraction $\mathbb{K}_i = K_i \subseteq \mathbb{K}_{i+1}$. Let K_{i+1}^0 be
349 the complex as defined in the proof of Lemma 2. Consider the following diagram of the
350 simplicial complexes, and note that $i' = i_1 \circ i_0$ where both i_0 and i_1 are inclusions induced
351 by the respective strong collapses.

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

352

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

356 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,
357 $i_0 \circ f_i(\sigma) = \gamma$. Observe that, in the retraction map associated to the strong collapse
358 $r_1 : \mathbb{K}_{i+1} \searrow \searrow K_{i+1}^0$, v is not contracted (by construction of K_{i+1}^0). Therefore $r_1 \circ i(\sigma)$ is a
359 simplex $\gamma' \in K_{i+1}^0$ containing v .

360 Now, as mentioned in the proof of Lemma 2, u dominates v in K_{i+1}^0 . Therefore all the
361 maximal simplices in K_{i+1}^0 that contain v also contain u . Therefore, γ' will be a face of a
362 maximal simplex $\tau \in K_{i+1}^0$ that contains u .

363 Since γ is obtained by contracting v to u , γ must be a face of such a $\tau \in K_{i+1}^0$, which
364 contains both u and v . This implies that $\gamma' \cup \gamma \subseteq \tau$ for some maximal simplex $\tau \in K_{i+1}^0$,
365 which in turn implies that $r_1 \circ i(\sigma)$ is contiguous to $i_0 \circ f_i(\sigma)$. After composing both sides
366 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 $\mathbb{K}_{i+1} \searrow \searrow K_{i+1}^0$, $i_1 \circ r_1 \sim 1_{\mathbb{K}_{i+1}}$ [3],
367 where $1_{\mathbb{K}_{i+1}}$ is the identity over \mathbb{K}_{i+1} . As $i_1 \circ i_0 = i'$, we conclude $i' \circ f_i \sim i$.

368 Since contiguous maps are homotopic at the level of geometric realizations, the diagram
369 in the lemma commutes. \blacktriangleleft

370 The following lemma is a more general version of Lemma 2 stated above. Its proof
371 follows from simple inductive arguments.

372 **► Lemma 4.** Given a tower $\mathbb{T} : K_0 \xrightarrow{f_0} K_1 \xrightarrow{f_1} \dots \xrightarrow{f_{m-1}} K_m$. for each $0 \leq i \leq m$,
373 $\mathbb{K}_i \searrow \searrow K_i$.

374 Again using an inductive argument along with Lemmas 3 and 4, we can deduce the
375 following result.

376 **► Theorem 5.** The following diagram commutes and therefore the tower $\mathcal{T} : K_0 \xrightarrow{f_0} K_1 \xrightarrow{f_1} \dots$
377 $\dots \xrightarrow{f_{m-1}} K_m$ and the constructed filtration $\mathcal{F} : \mathbb{K}_0 \hookrightarrow \mathbb{K}_1 \hookrightarrow \dots \hookrightarrow \mathbb{K}_m$ have the same
378 persistence diagram.

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

379

380 Here ϕ_i is a strong collapse for each $i \in \{0, \dots, m\}$ and $*$ indicates the induced homomorph-
381 isms.

382 **Complexity Analysis:** The *contracted* vertices defined here are exactly the same as the
 383 inactive vertices in [33]. By construction, any contracted vertex will be dominated per-
 384 manently in the filtration. As such a vertex stops being existent in the tower later on, its
 385 neighborhood stays the same and the vertex remains dominated. Therefore, all the *active*
 386 vertices in our construction are the vertices that are currently not dominated. However, we
 387 choose to differentiate the two different types of inactive vertices (dominated and contracted)
 388 to emphasise that, at any point in our construction, the number of active vertices is less than
 389 the number of active vertices that are used in [33]. Moreover, since a strong expansion is a
 390 coning, the size of the final filtration in our construction is at most that obtained by the
 391 construction prescribed in [33]. Moreover, since we are working with 1-skeletons only, the
 392 space and time complexity of our method is much lower than that of [33].

393 To analyze the time complexity, observe that each edge inclusion can be performed
 394 in constant time $\mathcal{O}(1)$ and, before each contraction, the domination relationships can be
 395 updated in $\mathcal{O}(v * k^2)$ time. Note that we are not computing the core here we are just
 396 computing the currently dominated vertices. For each strong expansion in the augmented
 397 graphs corresponding to the contraction $(\{w, u\} \mapsto u)$ in the original graph, we look at the
 398 set-difference between at most k neighbors of the two vertices. This can be performed in $\mathcal{O}(k)$
 399 time. Therefore, given a tower with n_c elementary contractions, and $|\mathbb{G}_m|$ the size of the
 400 skeleton of the final equivalent flag filtration can be computed in at most $\mathcal{O}(|\mathbb{G}_m| + n_c * v * k^3)$.
 401 The space complexity of our construction is $\mathcal{O}(n_0 * k)$ which is the size of the sparse adjacency
 402 matrix of the final flag filtration of n_0 vertices. We summarize our result in the following
 403 theorem.

404 ► **Theorem 6.** Let $\mathcal{T} : K_0 \xrightarrow{f_0} K_1 \xrightarrow{f_1} \dots \xrightarrow{f_{m-1}} K_m$ be a flag tower where, w.l.o.g., $K_0 = \emptyset$
 405 and each f_i is either an inclusion (not necessarily elementary but corresponds to an elementary
 406 inclusion on the graphs G_i) or an elementary contraction. Let d denote the maximal dimension
 407 of K_i s in \mathcal{T} , and let n denote the total number of elementary inclusions of simplices in
 408 \mathcal{T} , n_c total number of elementary contraction and n_0 the number of vertex inclusions in
 409 \mathcal{T} . Then, there exists a filtration $\mathcal{F} : \mathbb{K}_0 \hookrightarrow \mathbb{K}_1 \hookrightarrow \dots \hookrightarrow \mathbb{K}_m$, where the inclusions are not
 410 necessarily elementary, such that \mathcal{T} and \mathcal{F} have the same persistence diagram and the size of
 411 the filtration $|\mathbb{K}_m|$ is at most $\mathcal{O}(d * n * \log n_0)$. Moreover, \mathcal{F} is a filtered flag complex and can
 412 be computed from \mathcal{T} using only the 1-skeletons G_i s of K_i s in $\mathcal{O}(|\mathbb{G}_m| + n_c * n_0 * k^3)$ time and
 413 $\mathcal{O}(n_0 * k)$ space complexity, here k is the upper bound on the degree of the vertices in \mathbb{G}_m .

414 5 Computational experiments

415 We compute the persistence diagram (PD) of VR-filtrations associated to different data. The
 416 filtration value of a simplex in a VR filtration is the length of the longest edge of the simplex.

417 **Approximate persistence diagram** Given a VR filtration, one can choose to collapse the
 418 original complexes after each edge inclusion. However, we can also choose to strong collapse
 419 the complexes less often, i.e. after several edge inclusions rather than just one. This will result
 420 in a faster algorithm but comes with a cost: the computed PD is then only approximate. We
 421 call **snapshots** the values of the scale parameter at which we choose to strong collapse the
 422 complex. The difference between two consecutive snapshots is called a **step**. We approximate
 423 the filtration value of a simplex as the value of the snapshot at which it first appears. We can
 424 observe that our algorithm will report all persistence pairs that are separated by at least one
 425 snapshot. Hence if all steps are equal to some $\epsilon > 0$, we will compute all the persistence pairs

426 whose lengths are at least ϵ . It follows that the *bottleneck distance* between the computed
 427 PD and the exact one is at most ϵ .

428 **Experimental setup** Now we present some experimental results comparing our software
 429 named RipsCollapser with Ripser [5], which is the state of the art software to compute the
 430 PD of VR filtrations. RipsCollapser has been coded in C++. The code has been compiled
 431 using the compiler ‘clang-900.0.38’ and all computations were performed on a ‘2.8 GHz Intel
 432 Core i5’ machine with 16 GB of available RAM.

433 The comparison is done on three datasets **netw-sc**, **senate** and **eleg** from [18]. RipsCol-
 434 lapser takes as input a VR filtration, constructs the associated collapsed sequence and then
 435 computes the equivalent flag filtration. The output filtration is then sent to the Gudhi library
 436 to compute persistence. The reported time is the total time which includes: 1. The time
 437 taken to compute the largest 1-skeleton associated to the maximum threshold value, 2. The
 438 time taken to collapse all the sub-skeletons and assemble their cores. 3. To transform them
 439 into an equivalent flag-filtration. 4. To compute the PD of the equivalent flag-filtration. We
 440 also note the time taken from 1 to 3 as pre-process time.

441 As mentioned above, we approximate the filtration value of a simplex as the value of the
 442 snapshot parameter at which it appears for the first time, whereas, in the case of Ripser, it
 443 is the length of the longest edge (1-simplex) it contains. Therefore, the computed PD by
 444 RipsCollapser is not exactly the same as the one computed by Ripser. However, in the above
 445 experiments, we choose steps that are very small so that the bottleneck distance between the
 446 two PD returned by Ripser and RipsCollapser for a given data set is also very small.

447 Command `<./riper inputData -format distances -threshold inputTh -dim inputDim >`
 448 was used to run Ripser and we used the distance matrix format for all the datasets. Both
 449 RipsCollapser and Ripser have a parameter `-dim` until which they compute the PD. For a
 450 given *threshold* (maximum scale parameter) and *dim*, Ripser computes the *dim*-skeleton of
 451 the VR complex and then computes the PD of the *dim*-skeleton. Differently, RipsCollapser
 452 computes the 1-skeleton of the VR complex for the maximum scale parameter (Threshold).
 453 During the preprocessing, the dimension of the original complex doesn’t come into con-
 454 sideration. Therefore the preprocessing is done for all the dimensions. However, we can
 455 restrict the dimension (using parameter `-dim`) of the smaller equivalent flag filtration. As
 456 the experiments will show, this does not matter much since the collapsed cores have small
 457 size and dimension.

458 **Results** Table 1 contains the results of the experiments using RipsCollapser and Table
 459 2 contains the results using Ripser. Ripser performs quite well for computing PD for low
 460 values of *dim*. However, as we move to intermediate values, it slows down quite considerably
 461 and in some cases (*dim* above 7), the size of the complex is so huge that Ripser crashed due
 462 to memory overload. Differently RipsCollapser is not much affected by the choice of *dim*
 463 both in terms of space and time and we can compute PD for large values of the threshold
 464 and of *dim*.

490 As an additional remark, we note that, in our current implementation, the collapses are
 491 performed in sequence. A further improvement would be to perform them in parallel.

492 RipsCollapser will be available as an open-source package of a next release of the Gudhi
 493 library [31].

Data	Pnt	Threshold	RipsCollapser(Gudhi)				
			Dim	Preprocess-Time	Total-Time	Steps	TotSnaps
netw-sc	379	4.5	5	4.99s	5.09s	0.02	213
netw-sc	379	4.5	all	4.99s	5.08s	0.02	213
"	"	5.5	6	9.55s	9.65s	0.02	263
"	"	5.5	all	9.46s	9.56s	0.02	263
senate	103	0.415	all	2.71s	2.73s	0.001	403
eleg	297	0.3	5	11.14s	13.32s	0.001	284
eleg	297	0.3	all	11.15s	25.3s	0.001	284

Table 1 The columns are, from left to right: dataset (Data), number of points (Pnt), maximum scale parameter (Threshold), dimension of the collapsed flag-Complex (Dim), time taken to collapse and compute equivalent flag-filtration (Preprocess-Time), total time taken by RipsCollapser (Gudhi) (Total-Time), incremental steps of subcomplexes (Steps) and total number of snapshots used (TotSnaps). All times are averaged over five trials

Data	Pnt	Threshold	Val		Val		Val	
			Dim	Time	Dim	Time	Dim	Time
netw-sc	379	4.5	4	3.8s	5	21.5s	7	357s
"	"	5.5	4	25.3s	5	231.2s	6	∞
senate	103	0.415	3	0.52s	4	5.9s	5	52.3s
"	"	"	6	406.8s	7	∞		
eleg	297	0.3	3	8.9s	4	217s	5	∞

Table 2 The columns are, from left to right: dataset (Data), number of points (Pnt), maximum scale parameter (Threshold), input dimension for Ripser (Dim), total time taken by Ripser (Time). Most results are averaged over five trials except the longer ones. ∞ in the Time column means that the experiment ran longer than 12hrs or crashed due to memory overload.

References

- M. Adamaszek and J. Stacho. Complexity of simplicial homology and independence complexes of chordal graphs. *Computational Geometry: Theory and Applications*, 57:8–18, 2016.
- 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.
- J. A. Barmak and E. G. Minian. Strong homotopy types, nerves and collapses. *Discrete and Computational Geometry*, 47:301–328, 2012.
- 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. 2014.
- Ulrich Bauer. Ripser. URL: <https://github.com/Ripser/ripser>.
- Jean-Daniel Boissonnat, C. S. Karthik, and Sébastien Tavenas. Building efficient and compact data structures for simplicial complexes. *Algorithmica*, 79:530–567, 2017.
- Jean-Daniel Boissonnat, Siddharth Pritam, and Divyansh Pareek. Strong Collapse for Persistence. In *26th Annual European Symposium on Algorithms (ESA 2018)*, volume 112, 2018.

- 512 **8** Jean-Daniel Boissonnat and Karthik C. S. An efficient representation for filtrations of
513 simplicial complexes. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2017.
- 514 **9** M. Botnan and G Spreemann. Approximating persistent homology in euclidean space
515 through collapses. In: *Applicable Algebra in Engineering, Communication and Computing*,
516 26:73–101.
- 517 **10** Gunnar Carlsson and Vin de Silva. Zigzag persistence. *Found Comput Math*, 10, 2010.
- 518 **11** Gunnar Carlsson, Vin de Silva, and Dmitriy Morozov. Zigzag persistent homology and
519 real-valued functions. *SOCG*, pages 247–256, 2009.
- 520 **12** Gunnar Carlsson, Tigran Ishkhanov, Vin de Silva, and Afra Zomorodian. On the local be-
521 havior of spaces of natural images. In: *International Journal of Computer Vision*, 76:1–12,
522 2008.
- 523 **13** Joseph Minhow Chan, Gunnar Carlsson, and Raul Rabadan. Topology of viral evolution.
524 In: *Proceedings of the National Academy of Sciences*, 110:18566–18571, 2013.
- 525 **14** F. Chazal and S. Oudot. Towards persistence-based reconstruction in euclidean spaces.
526 *SOCG*, 2008.
- 527 **15** C. Chen and M. Kerber. Persistent homology computation with a twist. In *European*
528 *Workshop on Computational Geometry (EuroCG)*, pages 197–200, 2011.
- 529 **16** Aruni Choudhary, Michael Kerber, and Sharath Raghvendra:. In *Polynomial-Sized To-*
530 *pological Approximations Using The Permutedron*. 32nd International Symposium on
531 Computational Geometry (SoCG), 2016.
- 532 **17** David Cohen-Steiner and Herbert Edelsbrunner. Harer, john. stability of persistence
533 diagrams. *Discrete Comput. Geom*, 37:103–120, 2007.
- 534 **18** Datasets. URL: <https://github.com/n-otter/PH-roadmap/> .
- 535 **19** Vin de Silva and Robert Ghrist. Coverage in sensor networks via persistent homology. In:
536 *Algebraic and Geometric Topology*, 7:339 – 358, 2007.
- 537 **20** Harm Derksen and Jerzy Weyman. Quiver representations. *Notices of the American Math-*
538 *ematical Society*, 52(2):200–206, February 2005.
- 539 **21** T. Dey, F. Fan, and Y. Wang. Computing topological persistence for simplicial maps. In
540 *Symposium on Computational Geometry (SoCG)*, pages 345–354, 2014.
- 541 **22** T. K. Dey, H. Edelsbrunner, S. Guha, and D. Nekhayev. Topology preserving edge con-
542 traction. *Publications de l’Institut Mathematique (Beograd)*, 60:23–45, 1999.
- 543 **23** Tamal Dey, Dayu Shi, and Yusu Wang. *SimBa: An efficient tool for approximating Rips-*
544 *filtration persistence via Simplicial Batch-collapse*. In *European Symp. on Algorithms*
545 (ESA), pages 35:1–35:16, 2016.
- 546 **24** C. H. Dowker. Homology groups of relations. *The Annals of Mathematics*, 56:84–95, 1952.
- 547 **25** P. Dłotko and H. Wagner. Simplification of complexes for persistent homology computa-
548 tions,. *Homology, Homotopy and Applications*, 16:49–63, 2014.
- 549 **26** H. Edelsbrunner, D. Letscher, and A. Zomorodian. Topological persistence and simplifica-
550 tion. *Discrete Comput. Geom*, 28:511–533, 2002.
- 551 **27** Herbert Edelsbrunner and John Harer. *Computational Topology: An Introduction*. Amer-
552 ican Mathematical Society, 2010.
- 553 **28** Brittany Terese Fasy, Jisu Kim, Fabrizio Lecci, and Clément Maria:. Introduction to the r
554 package tda. *CoRR abs/1411.1830*, 2014.
- 555 **29** E. Fieux and J. Lacaze. Foldings in graphs and relations with simplicial complexes and
556 posets. *Discrete Mathematics*, 312(17):2639 – 2651, 2012.
- 557 **30** François Le Gall. Powers of tensors and fast matrix multiplication. *ISSAC ’*, 14:296–303,
558 2014.
- 559 **31** Gudhi: Geometry understanding in higher dimensions. URL: [http://gudhi.gforge.](http://gudhi.gforge.inria.fr/)
560 [inria.fr/](http://gudhi.gforge.inria.fr/).
- 561 **32** A. Hatcher. *Algebraic Topology*. Univ. Press Cambridge, 2001.

- 562 **33** Michael Kerber and Hannah Schreiber:. Barcodes of towers and a streaming algorithm
563 for persistent homology. *33rd International Symposium on Computational Geometry*, 2017.
564 arXiv:1701.02208.
- 565 **34** Michael Kerber and R. Sharathkumar. Approximate Čech complex in low and high dimen-
566 sions. In *Algorithms and Computation*, pages 666–676. by Leizhen Cai, Siu-Wing Cheng,
567 and Tak-Wah Lam. Vol. 8283. Lecture Notes in Computer Science, 2013.
- 568 **35** C. Maria and S. Oudot. Zigzag persistence via reflections and transpositions. In *Proc.*
569 *ACM-SIAM Symposium on Discrete Algorithms (SODA)* pp. 181–199, January 2015.
- 570 **36** Nikola Milosavljevic, Dmitriy Morozov, and Primož Skraba. Zigzag persistent homology in
571 matrix multiplication time. In *Symposium on Computational Geometry (SoCG)*, 2011.
- 572 **37** K. Mischaikow and V. Nanda. Morse theory for filtrations and efficient computation of
573 persistent homology. *DCG*, 50:330–353, September 2013.
- 574 **38** Dmitriy Mozozov. Dionysus. URL: <http://www.mrzv.org/software/dionysus/>.
- 575 **39** J. Munkres. *Elements of Algebraic Topology*. Perseus Publishing, 1984.
- 576 **40** N. Otter, M. Porter, U. Tillmann, P. Grindrod, and H. Harrington. A roadmap for the
577 computation of persistent homology. *EPJ Data Science, Springer Nature*, page 6:17, 2017.
- 578 **41** Jose Perea and Gunnar Carlsson. A klein-bottle-based dictionary for texture representation.
579 In: *International Journal of Computer Vision*, 107:75–97, 2014.
- 580 **42** Hannah Schreiber. Sophia. URL: <https://bitbucket.org/schreiberh/sophia/>.
- 581 **43** Donald Sheehy. Linear-size approximations to the victoris–rips filtration. *Discrete and*
582 *Computational Geometry*, 49:778–796, 2013.
- 583 **44** M. Tancer. Recognition of collapsible complexes is np-complete. *Discrete and Computa-*
584 *tional Geometry*, 55:21–38, 2016.
- 585 **45** J.Reininghaus U. Bauer, M. Kerber and Hagner:. Phat – persistent homology algorithms
586 toolbox. *Journal of Symbolic Computation*, 78, 2017.
- 587 **46** J. H. C Whitehead. Simplicial spaces nuclei and m-groups. *Proc. London Math. Soc.*,
588 45:243–327, 1939.
- 589 **47** A. C. Wilkerson, H. Chintakunta, and H. Krim. Computing persistent features in big data:
590 A distributed dimension reduction approach. *ICASSP - Proceedings*, pages 11–15, 2014.
- 591 **48** A. C. Wilkerson, T. J. Moore, and and A. H. Krim A. Swami. *Simplifying the homology of*
592 *networks via strong collapses*. ICASSP - Proceedings, 2013.
- 593 **49** A. Zomorodian and G. Carlsson. Computing persistent homology. *Discrete Comput. Geom.*,
594 33:249–274, 2005.
- 595 **50** Afra Zomorodian. The tidy set: A minimal simplicial set for computing homology of
596 clique complexes. In *Proceedings of the Twenty-sixth Annual Symposium on Computational*
597 *Geometry*, pages 257–266, SoCG’10. Snowbird, Utah, USA . isbn, 2010.