Zigzag Persistent Homology in Matrix Multiplication Time

Nikola Milosavljevic, Dmitriy Morozov, Primoz Skraba

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Zigzag Persistent Homology in Matrix Multiplication Time

Nikola Milosavljević — Dmitriy Morozov — Primož Škraba

N° 7393

September 2010

Thème COM

ISSN 0249-6399 ISRN INRIA/RR--7393--FR+ENG

INSTITUT NATIONAL
DE RECHERCHE
EN INFORMATIQUE
ET EN AUTOMATIQUE

INRIA

centre de recherche
SACLAY - ÎLE-DE-FRANCE

Zigzag Persistent Homology in Matrix Multiplication Time

Nikola Milosavljević *, Dmitriy Morozov †, Primož Škraba ‡

Thème COM — Systèmes communicants
Équipe-Projet Geometrica

Rapport de recherche n° 7393 — September 2010 — 18 pages

Abstract: We present a new algorithm for computing zigzag persistent homology, an algebraic structure which encodes changes to homology groups of a simplicial complex over a sequence of simplex additions and deletions. Provided that there is an algorithm that multiplies two $n \times n$ matrices in $M(n)$ time, our algorithm runs in $O(M(n) \log n)$ time if $M(n) = O(n^2)$, and $O(M(n))$ time otherwise, for a sequence of n additions and deletions. In particular, the running time is $O(n^{2.376})$, by result of Coppersmith and Winograd. The fastest previously known algorithm for this problem takes $O(n^3)$ time in the worst case.

Key-words: Zig-zag persistence, fast matrix multiplication, complexity

* nikolam@mpi-inf.mpg.de
† dmitriy@mrzv.org
‡ primoz.skraba@inria.fr

Centre de recherche INRIA Saclay – Île-de-France
Parc Orsay Université
4, rue Jacques Monod, 91893 ORSAY Cedex
Téléphone : +33 1 72 92 59 00

Calcul de l'homologie persistante des zigzags par multiplication rapide de matrices

Résumé : Nous prsentons un algorithme pour calculer l'homologie persistante des zigzags, une structure algbrique qui code les changements survenant dans l'homologie d'un complexe simplicial lors d'une squence d'insertions et de suppressions de simplexes. Sous l'hypothse qu'il existe des algorithmes capables de multiplier deux matrices $n \times n$ en temps $M(n)$, notre algorithme tourne en temps $O(M(n)) \log n$ si $M(n) = O(n^2)$ et $O(M(n))$ sinon, pour une squence de n additions et suppressions. En particulier, le temps de calcul est $O(n^{2.376})$ par le rsultat de Coppersmith et Winograd. L'algorithme le plus rapide connu jusqu' prsent pour ce problme prenait un temps $O(n^3)$ dans le cas le pire.

Mots-clés : L'homologie persistante des zigzags, multiplication rapide de matrices, complexitié

1 Introduction

Motivation. Since its introduction a decade ago, *persistent homology* [1] has become an important tool in such wide-ranging domains as computational biology, geometric processing, machine learning, scientific visualization, and sensor networks. Its success rests on two pillars: a solid theoretical foundation [2], including the celebrated stability result [3], and the availability of fast algorithms [4].

A more recent development is *zigzag persistence* [5], which is a generalization of ordinary persistence built on algebraic insights. Together with an efficient algorithm in the homological setting, zigzag persistence has already resolved open questions in the theoretical study of persistence [6]. In addition to the algebraic generality, it brings an appealing property for algorithm design: one is not constrained to study growing families of spaces as in ordinary persistence, but instead is free to choose whether to grow or shrink the space in question on demand. This flexibility has led to a technique for computing ordinary persistent homology of a real-valued function using space that depends only on the size of the largest levelset, rather than the entire domain [6].

Despite the growing interest in persistence, the complexity of its computation is not well understood. Both ordinary and zigzag persistence have algorithms that are cubic in the worst case, but little is known about their optimality. In this paper, we present an algorithm to compute zigzag persistent homology (and therefore also ordinary persistent homology) that, in the worst case, runs in the time it takes to multiply two matrices (unless that time is $\mathrm{O}(n^2)$, in which case our algorithm is $\mathrm{O}(n^2 \log n)$).

Related Work. The computation of homology through Gaussian elimination has been known for some time [7, 8]. The sequential algorithm has the worst case bound of $O(n^3)$, which has been the bound for most persistence computations. The computational complexity equivalence of Gaussian elimination and matrix multiplication [9] was one of the factors in motivating this work. However, most of the past work on speeding up homology computations has been based on combinatorial techniques rather than matrix computations.

The first sub-cubic algorithm incrementally computes the Betti numbers of subcomplexes of triangulations of the three sphere $\mathbb{S}^3$ [10]. The running time is $n\alpha(n)$, where n is the number of simplices and $\alpha(\cdot)$ is the inverse Ackermann function. A different approach was given in [11] to computing Betti numbers using combinatorial Laplacians. It uses the power method on the Laplacian matrix, hence computing it via matrix-vector multiplication. However, the number of multiplications depends on the eigenvalues of the matrix and hence is not easily bounded.

The introduction of persistent homology [1, 2], zigzag persistence [5, 6], and other variants [12, 13] came with algorithms based on the sequential Gaussian elimination and so had running time of $O(n^3)$. In [14], an example complex is given where persistence takes cubic time. However, experimental results show close to linear behavior, most often attributed to the sparsity of the involved matrices.

In the literature, most work to speed up homology computation has had a different flavor, with efforts being made to reduce the size of the input complex using combinatorial operations which preserve the homology [15], most often simplicial collapses. In certain cases, notably 2-manifolds, it is possible to create an optimal order of collapses to obtain the homology [16]. In the end of this procedure we end up only with critical cells, allowing us to simply read off the homology. However, it was also shown that in the general case, there may not exist a way to collapse the complex to the minimal number of cells [17]. Furthermore, it was shown to be NP-hard to find an order which will

result in the minimum number of cells [18]. More recently, a method [19] was given to reduce the size of the complex in the special case of clique complexes. Such complexes are completely determined by their graphs, and the work tries to extract a minimal representation based on maximal cliques. As with the collapsing techniques, it does not come with any guarantees on the size of the output complex.

In this work we use fast matrix multiplication as a black-box, but we briefly recount the work which began with the celebrated result of Strassen [20], showing that the matrix inversion can be done with an exponent of $\omega \approx 2.8$ rather than 3. This has been followed by numerous improvements, with the currently best known algorithm of Coppersmith-Winograd [21]. The current estimate for the exponent is $\omega = 2.376$. Finally, we note that this is still an active area of research, where [22] proves that if a group with certain properties exists, so must a quadratic time algorithm for matrix multiplication.

Outline. We review the necessary background in Section 2. We recast the zigzag persistent homology algorithm of [6] in terms of matrix multiplications in Section 3. By batching some of the operations together in Section 4, we obtain the claimed running time.

2 Background

In this section, we recount the necessary mathematical and algorithmic background. Since the emphasis of this paper is algorithmic, we begin with a simplicial complex $\mathbb{X}$ rather than the usual topological space. Furthermore, everything will be done with simplicial homology over a field $\mathbb{F}$. While mathematically it is often more convenient to work in singular homology, in practice, we always compute in a combinatorial setting.

We now give a brief overview of homology and zigzag persistence. We refer the reader to Munkres [7] or Hatcher [23] for background on homology, Edelsbrunner and Harer [4] for persistence, and Carlsson and de Silva [5] for zigzag persistence. Our primary object is a simplicial complex, which is a set of simplices such that all faces of a simplex are in the complex and the intersection of two simplices is a (possibly empty) face of both.

We define the *chain group* C_p as an Abelian group on the set of oriented p-simplices in the simplicial complex. A p-chain is a linear combination of simplices with coefficients in a group. We restrict ourselves to the case where the coefficients lie in a field, therefore each C_p is a vector space. We also define the boundary operator $\partial_p : \mathsf{C}_p \to \mathsf{C}_{p-1}$. In a simplicial complex, the boundary operator is defined on a simplex σ as $\partial_p \sigma = \sum_i (-1)^i [v_0, v_1, \ldots, \hat{v}_i, \ldots, v_p]$, where $\hat{v}_i$ is deleted from the sequence. Since the operator is linear, boundary of a chain is the linear combination of the boundaries of the simplices. The boundary operator connects the chain groups into the *chain complex*: $\ldots \to \mathsf{C}_{p+1} \xrightarrow{\partial_{p+1}} \mathsf{C}_p \xrightarrow{\partial_p} \mathsf{C}_{p-1} \to \ldots$ The boundary operator is a map between vector spaces and is most naturally represented as a matrix where the rows represent p-simplices and the columns represent $(p+1)$-simplices.

To define homology, we require two subgroups: the *cycle group* Z and the *boundary group* B. The cycle group Z_p, is the kernel of the ∂_p, which is the null space of the boundary matrix. The boundary group B_p is the image of ∂_{p+1}. The *homology* is the quotient group of the two subspaces: $\mathsf{H}_p = \mathsf{Z}_p / \mathsf{B}_p$. By the property that $\partial_p \partial_{p+1} = 0$, it is not difficult to see that $\mathsf{B}_p \subseteq \mathsf{Z}_p \subseteq \mathsf{C}_p$, meaning the above quotient is well defined. Since these are vector spaces, standard Gaussian elimination can be used to find a basis for each space.

A powerful extension to homology was introduced called *persistent homology* and more recently *zigzag persistence*. Since we often construct simplicial complexes from geometry, a single complex does not always capture the underlying topology of a noisy space. The notion of persistence was introduced to counter this problem. Rather than building a single complex, we can build several and if we can relate them in a certain way, persistence tells us which features live across multiple parameters, and therefore represent prominent features of the underlying space.

The starting point of zigzag persistent homology is a sequence of spaces connected with maps

$$\mathbb{X}_1 \leftrightarrow \mathbb{X}_2 \leftrightarrow \ldots \leftrightarrow \mathbb{X}_n. \tag{1}$$

The above notation means that the map can point in either direction. For each of these spaces we can form a a chain complex, which induces maps between the vector spaces, $\mathsf{C}(\mathbb{X}_1) \leftrightarrow \mathsf{C}(\mathbb{X}_2) \leftrightarrow \ldots \leftrightarrow \mathsf{C}(\mathbb{X}_n)$. Finally, by passing to homology, we obtain a sequence of vector spaces connected by homomorphisms.

$$\mathsf{H}(\mathbb{X}_1) \leftrightarrow \mathsf{H}(\mathbb{X}_2) \leftrightarrow \ldots \leftrightarrow \mathsf{H}(\mathbb{X}_n). \tag{2}$$

This object is called zigzag module, denoted by $\mathbb{V}$. It is the collection of vector spaces $\mathsf{H}(\mathbb{X}_i)$ along with the maps between them. As shown in [5], $\mathbb{V}$ has a nice decomposition into a direct sum of interval modules, $\mathbb{V} = \bigoplus \mathbb{I}_{[b,d]}$. Each interval module $\mathbb{I}_{[b,d]}$ represents a homology class which exists in all the spaces from $\mathsf{H}(\mathbb{X}_b)$ to $\mathsf{H}(\mathbb{X}_d)$ inclusive. Therefore, we say this class persists from b to d. Note that persistent homology is a special case of zigzag persistence where all the arrows point one way.

To compute this decomposition, we require compatible bases in all of the individual spaces and track how they change as we apply the maps. Surprisingly, all the information we require can be encoded in a single filtration.

Definition. The *right filtration* of a space $\mathsf{H}(\mathbb{X}_i)$ is the collection of subspaces of V_i taking the form $R^i = (R_0, R_1, R_2, \ldots, R_i)$, satisfying inclusion relations $R_k \subseteq R_l$ for $k \leq l$. The filtration is then defined inductively. If the map points forward: $\mathsf{H}(\mathbb{X}_i) \xrightarrow{f} \mathsf{H}(\mathbb{X}_{i+1})$, then the filtration is updated with the images of the map $R^{i+1} = (f(R_0), f(R_1), \ldots, f(R_i), \mathsf{H}(\mathbb{X}_{i+1}))$. If the map points in the other direction: $\mathsf{H}(\mathbb{X}_i) \xleftarrow{g} \mathsf{H}(\mathbb{X}_{i+1})$, then the preimages are added. $R^{i+1} = (0, g^{-1}(R_0), g^{-1}(R_1), \ldots, g^{-1}(R_i))$.

By keeping track of the changes in the right filtration as we process the zigzag in (1), we obtain the interval decomposition. In the algorithmic setting, without loss of generality, we can assume that in the sequence of simplicial complexes (1) consecutive complexes differ by a single simplex, i.e. either $\mathbb{X}_{i+1} = \mathbb{X}_i \cup \sigma_i$, or $\mathbb{X}_i = \mathbb{X}_{i+1} \cup \sigma_i$.

3 Sequential Algorithm

In this section we express the algorithm of [6] in terms of matrix update operations.

Representation. We start with a sequence of simplicial complexes (1), where consecutive spaces differ by a single simplex. Enumerating all the simplices $\sigma_1, \ldots, \sigma_k$ that appear in the sequence, we ignore multiplicities: if the same simplex enters, leaves, and then re-enters, we treat it as two simplices in the enumeration. We associate to each simplex σ_i its interval $\mathbb{I}_{[b_i,d_i]}$ in the chain complex zigzag, i.e. $\sigma_i \in \mathbb{X}_j$ iff $j \in [b_i, d_i]$.

As in [6], we maintain the right filtration of the zigzag (2), and represent it after j steps via the matrices Z^j, B^j as well as the matrix C^j, which maintains the bounding chains. Denoting the boundary matrix of the simplicial complex with D, we have $0 = DZ^j$, $Z^j B^j = DC^j$. Furthermore, the right filtration of the group $\mathsf{H}(\mathbb{X}_j) = (R_0^j, R_1^j, \ldots)$ is represented via

$$R_k^j = \operatorname{span}\left(\{z + \mathsf{B} \mid z \in \mathsf{Z}_k^j \text{ and } \mathsf{B} = \operatorname{span}(Z^j B^j)\}\right),$$

where Z_k^j denotes the subset of the cycle group spanned by the first k columns of the matrix Z^j. We also implicitly maintain the birth vector associated to the right filtration; however, this detail is irrelevant to our argument, and we ignore it from now on.

In addition to the matrices D, Z^j, C^j, B^j of [6], we introduce matrices E^j, F^j. The matrices E^j and F^j keep track of the "past" and "future" simplices, respectively. Their columns correspond to the simplices that have been removed in the case of E^j, or not yet added in the case of F^j. Their boundaries are expressed using the auxiliary matrices $G^j, H^j, K^j, L^j, R^j, S^j$, as defined by Equation (3). We say that the matrix H^j is *independent* of the matrix B^j if any non-zero row in matrix B^j is zero in matrix H^j.

Sequential Invariant. After step j of the zigzag, the following conditions must hold:

1. $$D_p \left[\begin{array}{c|c|c|c} E_p^j & 0 & 0 & 0 \\ \hline 0 & Z_p^j & C_p^j & F_p^j \end{array}\right] = \left[\begin{array}{c|c|c|c} E_{p-1}^j & 0 & 0 & 0 \\ \hline 0 & Z_{p-1}^j & C_{p-1}^j & F_{p-1}^j \end{array}\right] \left[\begin{array}{c|c|c|c} G_p^j & 0 & 0 & 0 \\ \hline R_p^j & 0 & B_p^j & H_p^j \\ \hline S_p^j & 0 & 0 & K_p^j \\ \hline 0 & 0 & 0 & L_p^j \end{array}\right] \tag{3}$$

   For convenience, we abbreviate the matrices in (3) as $D\Phi_p^j = \Phi_{p-1}^j \Gamma_p^j$.
2. Matrix B_p^j has exactly one non-zero element per column, and at most one per row. Matrix H_p^j is independent of B_p^j.
3. Matrix Z_p^j forms a basis for the p-cycles $\mathsf{Z}_p(\mathbb{X}_j)$. The boundary DC_p^j forms a basis for the $(p-1)$-boundaries $\mathsf{B}_p(\mathbb{X}_j)$.
4. $R_k^j = \operatorname{span}\left(\{z + \mathsf{B} \mid z \in \mathsf{Z}_k^j \text{ and } \mathsf{B} = \operatorname{span}(Z^j B^j)\}\right)$.

Additionally, we choose a particular ordering for the rows and columns of our matrices.

Ordering. The rows of the matrices Z^j, C^j, E^j, and F^j correspond to the individual simplices in the complex, and we order them by the *removal* time of those simplices. Consequently, the rows and the columns of the boundary matrix D are also ordered by the removal time. The columns of the matrix F^j represent future simplices, and we order them by the *addition* time.

From the invariant and ordering requirements it follows that, initially, L_p^0 is the boundary matrix with rows and columns ordered by addition, while F_p^0 is a permutation matrix, its columns and rows are indexed by simplices ordered by addition and removal, respectively. The above ordering is insignificant in the rest of this section, however, it becomes critical in Section 4 when we describe how to divide and conquer the necessary computation.

Four cases. In the remainder of the section, we express the update performed to the matrices after each step as matrix multiplication.

$$\left[\begin{array}{c|c|c|c} E_p^{j+1} & 0 & 0 & 0 \\ \hline 0 & Z_p^{j+1} & C_p^{j+1} & F_p^{j+1} \end{array}\right] = \left[\begin{array}{c|c|c|c} E_p^{j} & 0 & 0 & 0 \\ \hline 0 & Z_p^{j} & C_p^{j} & F_p^{j} \end{array}\right] M_p^j \tag{4}$$

$$\left[\begin{array}{c|c|c|c} G_p^{j+1} & 0 & 0 & 0 \\ \hline R_p^{j+1} & 0 & B_p^{j+1} & H_p^{j+1} \\ \hline S_p^{j+1} & 0 & 0 & K_p^{j+1} \\ \hline 0 & 0 & 0 & L_p^{j+1} \end{array}\right] = (M_{p-1}^j)^{-1} \left[\begin{array}{c|c|c|c} G_p^{j} & 0 & 0 & 0 \\ \hline R_p^{j} & 0 & B_p^{j} & H_p^{j} \\ \hline S_p^{j} & 0 & 0 & K_p^{j} \\ \hline 0 & 0 & 0 & L_p^{j} \end{array}\right] M_p^j \tag{5}$$

Step j of the zigzag can be one of the two possibilities: an addition of a simplex, $\mathbb{X}_{j+1} = \mathbb{X}_j \cup \sigma_i$, or a removal of a simplex, $\mathbb{X}_j = \mathbb{X}_{j+1} \cup \sigma_i$. Each of the two cases can have one of the two outcomes: a birth or a death of a homology class. Before we describe the four possibilities, we establish an equivalence between the matrices.

Lemma 1 (Forward Step Equivalence Lemma). If the $(j+1)$-st step of the zigzag is the addition of the simplex σ_i, then (i) $L_{p-1}^j[i] = 0$, (ii) $K_{p-1}^j[i] = 0$, and (iii) $DF_p^j[i] = 0 \Leftrightarrow H_{p-1}^j[i] = 0$.

Proof. $L_{p-1}^j[i] = 0$ is a trivial consequence of $\mathbb{X}_j$ being a simplicial complex: when we are adding a simplex σ_i, its boundary cannot lie in the future. The column $K_{p-1}^j[i]$ is zero if $DF_p^j[i] = 0$ since the boundaries DC_{p-1}^j form a basis for the $(p-2)$-boundaries; consequently, no linear combination of the columns of C_{p-1}^j is zero. If $DF_p^j[0] \neq 0$, it is a cycle, and therefore not in the span of the bounding chains C_{p-1}^j. Finally, since the columns of Z_{p-1}^j are linearly independent, the boundary of $F_p^j[i]$ is zero iff $H_{p-1}^j[i]$ is zero. □

Birth after addition. This case is characterized by $DF^j[i] = 0$, or equivalently $H^j[i] = 0$. In particular, $F^j[i]$ is the newly born cycle (indeed, it contains the new simplex σ_i). We append it to the cycle matrix Z^j. The update matrix for this operation is

$$M_p^j = \left[\begin{array}{ccc|c|c} I & 0 & 0 & 0 & 0 \\ \hline 0 & I & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & I & 0 \\ \hline 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & I \end{array}\right]. \tag{6}$$

Death after addition. This case is characterized by $DF_p^j[i] \neq 0$, or equivalently $H_p^j[i] \neq 0$. Moreover, $DF_p^j[i] = Z_{p-1}^j H_p^j[i]$ represents the cycle that gets killed by the addition of σ_i. Indeed, since $H_p^j[i]$ is independent of B_p^j, and non-zero, it means that $Z_{p-1}^j H_p^j[i]$ is not a boundary before step j. On the other hand, by definition, $DF_p^j[i]$ is a boundary after the addition of simplex σ_i. Let a be the last non-zero entry in the column $H_p^j[i]$. We split the column using this entry into $H_p^j[i] = [\mathbf{h}^T \ a \ \mathbf{0}^T]^T$, where $\mathbf{h}$ is a vector. Suppose a is in the row k. $F_p^j[i]$ becomes a bounding chain, and therefore moves to the matrix C_p^j. Its boundary replaces the column k in matrix Z_{p-1}^j

(which is a change of basis in Z^j_{p-1}). We insert a column into the matrix B^j_p with a single 1 in row k to reflect the death of the cycle $Z^j_{p-1}[k]$. The necessary update operations are

$$M^j_{p-1} = \left[\begin{array}{c|c|ccc|c} I & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & I & \mathbf{h}/a & 0 & 0 & 0 \\ 0 & 0 & 1/a & 0 & 0 & 0 \\ 0 & 0 & 0 & I & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & I & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & I \end{array}\right], \quad M^j_p = \left[\begin{array}{c|c|ccc} I & 0 & 0 & 0 & 0 \\ \hline 0 & I & 0 & 0 & 0 \\ \hline 0 & 0 & I & 0 & 0 \\ \hline 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & I \end{array}\right]. \tag{7}$$

We are missing one more operation after both steps. Namely, to maintain the invariant we must make the matrix H^j_p independent of B^j_p. This update can be performed by multiplication with the matrix below. In the next section, we will batch these reductions together, and will rely on the individual matrices M^j to have the structure above. However, in the sequential algorithm, the actual update matrix is the product $M^j_p W^a_p$ of the matrices above with

$$W^a_p = \left[\begin{array}{c|c|c|c} I & 0 & 0 & 0 \\ \hline 0 & I & 0 & 0 \\ \hline 0 & 0 & I & -B^T_p H_p \\ \hline 0 & 0 & 0 & I \end{array}\right]. \tag{8}$$

(The product $-B^T_p H_p$ is zero in the case of birth, since the removal of a column does not affect the independence of H_p. This product contains a single non-zero row, if death happens. This row is exactly the negative of row k of H_p. We write the full matrix product for consistency with the next section.)

Remark 1. A subtle, but important consequence of the matrix H^j being independent of the matrix B^j is that the column $F^j[i]$ arrives "ready": we do not need to reduce it at the beginning of an individual step. This fact is crucial for the speed-up explained in the next section.

Removing a simplex σ_i corresponds to nullifying the rows i in the matrices Z^j, C^j, and F^j.

Birth after removal. This case is characterized by the row i of matrix Z^j_p being zero, i.e. there are no cycles containing simplex σ_i. Let a be the rightmost non-zero entry in row i of matrix C^j_p, and suppose it lies in column k, we have $C^j_p[i,\cdot] = [\mathbf{c}^T\ a\ \mathbf{0}^T]$; specifically, $C^j_p[i, 1..k-1] = \mathbf{c}^T$, $C^j_p[i,k] = a$. Let $B^j_{\mathbf{c}}$ be the part of the matrix B^j_p corresponding to the columns $\mathbf{c}^T$ in Equation (3), i.e. $B^j_{\mathbf{c}} = B^j_p[\cdot, 1..k]$.

1: The boundary $DC^j_p[k] = Z^j_{p-1} B^j_p[k]$ is the newly born cycle. We prepend it to the matrix Z^j. In the operations below, suppose the 1 in the column $B^j_p[k]$ is in row l.
2: We subtract column $C^j_p[k]/a$ from the other columns in C^j_p.
3: To keep track of the "past", we move column k from matrix C^j_p to E^j_p (with the corresponding update in the matrix Γ_p).
4: The term $-(\mathbf{c}^T/a)(B^j_{\mathbf{c}})^T$ below undoes the effect of Step 2 on matrix B^j_p.

Operations

$$M^j_{p-1} = \; l \left[\begin{array}{c|c|c|c|c|c} I & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & I & 0 & 0 & 0 \\ 0 & 1 & -(\mathbf{c}^T/a)(B^j_{\mathbf{c}})^T & 0 & 0 & 0 \\ 0 & 0 & 0 & I & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & I & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & I \end{array}\right], M^j_p = \left[\begin{array}{cc|c|cc|c} I & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & I & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & I & 0 & 0 \\ 0 & 1 & 0 & -\mathbf{c}^T/a & 0 & 0 \\ 0 & 0 & 0 & 0 & I & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & I \end{array}\right]. \quad (9)$$

Death after removal. This case is characterized by the row i of matrix Z^j_p being non-zero, i.e. there is a cycle that contains σ_i. Let a be the leftmost non-zero entry in this row, we have $Z^j_p[i,\cdot] = [\mathbf{0}^T\ a\ \mathbf{z}^T]$. Furthermore, let $C^j_p[i,\cdot] = \mathbf{c}^T$ be the i-th row of matrix C^j_p. We zero out this row by adding multiples of the column containing element a to the other columns, while moving that column into the "past" matrix E^j.

Operations

$$M^j_p = \left[\begin{array}{cc|c|cc|c} I & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & I & 0 & 0 & 0 \\ 0 & 1 & 0 & -\mathbf{z}^T/a & -\mathbf{c}^T/a & 0 \\ 0 & 0 & 0 & I & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & I & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & I \end{array}\right]. \quad (10)$$

In order to maintain the invariant, we must perform one more update after the removal of a simplex. Namely, we need to make sure that the row of the freshly removed simplex σ_i is zero in the future matrix F^j_p. As in the case of addition, in the next section, we rely on the matrices M^j_p to have the structure above, and batch such annihilations of the future together. In the sequential algorithm, however, the actual updates are the product $M^j_p W^r_p$. Writing $\mathbf{f}^T = F^j_p[i,\cdot]$ for the i-th row of the matrix F^j_p, we have

$$W^r_p = \left[\begin{array}{cc|c|c|c} I & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & -\mathbf{f}^T/a \\ \hline 0 & 0 & I & 0 & 0 \\ \hline 0 & 0 & 0 & I & 0 \\ \hline 0 & 0 & 0 & 0 & I \end{array}\right]. \quad (11)$$

Remark 2. In the addition cases, we only inspect the column of the matrix H^j_p corresponding to the j-th step of the zigzag, both to make a decision about birth or death, and to construct the update matrices. Similarly, in the removal cases we only inspect the rows of the matrices Z^j_p or C^j_p corresponding to the j-th step of the zigzag.

In the next section, we divide our matrices into smaller blocks, always ensuring that the necessary parts are available to the procedure responsible for the j-th step.

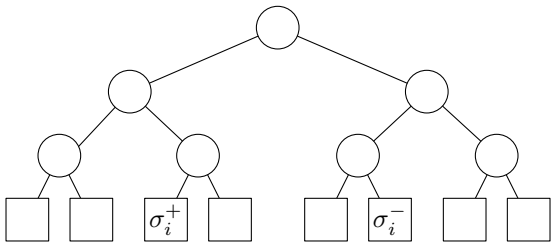
Remark 3. When processing the last step of the zigzag (or, equivalently, if we cut our zigzag at step $n = j+1$), W^a_p and W^r_p are identity matrices, since no more future steps remain. Therefore, the updates for this last step are captured completely by the matrices M^j_p.

Correctness. The above operations are almost the same as in [6]; in Appendix A, we focus on verifying the new parts of the invariant.

4 Hierarchical Algorithm

In this section we take advantage of the Remarks 1, 2, and 3 above, and construct a divide-and-conquer algorithm for processing a zigzag of simplicial complexes. For simplicity, we assume that the number of steps in the zigzag is a power of two. Throughout this section, $M(n)$ denotes time to multiply two $n \times n$ matrices.

Operation tree. The recursive algorithm shown in Figure 1 can be seen as an in-order traversal of the binary tree in the same figure. The operations performed at the leaves correspond to the



algorithm H-ZZPH(v)
if v is a leaf **then**
  return the basic update matrix M^j
if v is an internal node **then**
  l ← left child of v, r ← right child of v
  $M^{\mathrm{l}} \leftarrow$ H-ZZPH(l)
  $M^{\mathrm{l}} \leftarrow$ Left-to-Right$(M^{\mathrm{l}}, \mathrm{r})$
  $M^{\mathrm{r}} \leftarrow$ H-ZZPH(r)
  $M^{\mathrm{c}} \leftarrow$ Combine$(M^{\mathrm{l}}, M^{\mathrm{r}})$
  return M^{c}

Figure 1: Leaves correspond to the individual steps in the zigzag. The internal nodes represent combination operations, where update matrices from the steps below are combined via multiplications into larger update matrices. Left-to-Right applies M^{l} to parts of matrices corresponding to r's leaves.

individual steps of the zigzag. The computation is the same as in the previous section, except that the updates expressed by matrices M_p^j are not applied immediately, but rather propagated up the tree. The internal nodes combine and apply the updates in batches, as well as perform the (restricted) updates expressed by the matrices W_p^a and W_p^r.

Ordering. Notice that because of the row ordering assumed earlier, the two recurrences (on the left and right children) require only restrictions of the full matrices. In particular, splitting the matrices Φ_p and Γ_p as follows, we obtain the necessary restrictions:

$$\Phi_p = \left[\begin{array}{c|c|c|cc} & Z_p^l & C_p^l & {}^lF_p^l & {}^rF_p^l \\ & Z_p^r & C_p^r & {}^lF_p^r & {}^rF_p^r \end{array}\right], \quad \Gamma_p = \left[\begin{array}{c|c|c|cc} & & & & \\ \hline & 0 & B_p & {}^lH_p & {}^rH_p \\ \hline & 0 & 0 & {}^lK_p & {}^rK_p \\ \hline & 0 & 0 & {}^lL_p^l & {}^rL_p^l \\ & 0 & 0 & 0 & {}^rL_p^r \end{array}\right]. \tag{12}$$

In other words, H-ZZPH(v) ignores the past, and ignores the future beyond the steps that correspond to v's leaves[1]. Here, matrices with the superscript l correspond to the removals or additions during the first half of the sequence. So, for example, submatrix Z_p^l is the restriction of Z_p to the simplices removed during the steps corresponding to l.

Left-to-Right. Recurring on l with the matrix $\left[\ \middle|\ Z_p^l \mid C_p^l \mid {}^lF_p^l\right]$, we obtain an update matrix M^{l}. The first step of LEFT-TO-RIGHT is to apply M^{l} to input matrices Φ and Γ.

$$\Phi_p \leftarrow \Phi_p M_p^{\mathrm{l}}\,, \qquad \Gamma_p \leftarrow (M_{p-1}^{\mathrm{l}})^{-1}\Gamma_p M_p^{\mathrm{l}}\,. \tag{13}$$

The result of this is

$$\Phi_p = \left[\begin{array}{c|c|c|c} {}^lE_p^l & 0 & 0 & {}^rF_p^l \\ {}^lE_p^r & Z_p^r & C_p^r & {}^rF_p^r \end{array}\right] \qquad \Gamma_p = \left[\begin{array}{c|c|c|c} {}^lG_p^l & 0 & 0 & X \\ \hline {}^lR_p & 0 & B_p & {}^rH_p \\ \hline {}^lS_p & 0 & 0 & {}^rK_p \\ \hline 0 & 0 & 0 & {}^rL_p^r \end{array}\right]\,. \tag{14}$$

where X is some matrix (which only exists temporarily, has no topological meaning and hence needs no special notation). Submatrices ${}^lE_p^*$, ${}^lG_p^l$, ${}^lR_p$, ${}^lS_p$ represent cycles and chains that were destroyed in the subsequence corresponding to l.

The only operation that remains are the updates carried out by matrices W_p^a and W_p^r in the previous section. Recall that the former makes the matrix H_p independent of the matrix B_p, while the latter zeroes out the rows of the removed simplices in E_p in the future matrices ${}^*F_p^*$. We can express the two operations as a combined matrix W, and update matrix M^{l} to include the operations of W:

$$\Phi_p \leftarrow \Phi_p W_p\,, \qquad \Gamma_p \leftarrow W_{p-1}^{-1}\Gamma_p W_p\,.$$

$$M_p^{\mathrm{l}} \leftarrow \left[\begin{array}{ccc|c} & M_p^{\mathrm{l}} & & \begin{array}{c}0\\ \hline 0\\ \hline 0\end{array} \\ \hline 0 & 0 & 0 & I \end{array}\right] W_p\,, \quad W_p = \left[\begin{array}{ccc|c} & I & & \begin{array}{c} -({}^lE_p^l)^{-1}({}^rF_p^l) \\ \hline 0 \\ \hline -B_p^T({}^rH_p - {}^lR_p({}^lE_p^l)^{-1}({}^rF_p^l)) \end{array} \\ \hline 0 & 0 & 0 & I \end{array}\right]\,. \tag{15}$$

Combine. To combine the update matrices M^{l} and M^{r} into a matrix M^{c} that represents all the updates performed for steps corresponding to the node on which H-ZZPH is being executed, we simply multiply the two matrices together, first padding M^{r} to represent the fact that steps corresponding to r do not affect cycles and chains removed during steps corresponding to l.

$$M_p^{\mathrm{c}} \leftarrow M_p^{\mathrm{l}} \left[\begin{array}{c|ccc} I & 0 & 0 & 0 \\ \hline 0 & & & \\ 0 & & M_p^{\mathrm{r}} & \\ 0 & & & \end{array}\right]\,. \tag{16}$$

[1]We emphasize that this is a purely notational convention (to avoid equations with big matrices that stay almost unchanged throughout the execution). In a real implementation, entire Φ_p and Γ_p would be passed to the procedure by reference, it would modify small parts of it, and undo the changes before terminating (to simulate pass-by-value).

Correctness. We prove correctness of the hierarchical algorithm by applying the following more general statement to each leaf of the operation tree.

LEMMA 2. Let v be a tree node whose leaf descendants are $[j+1, j+k]$. Assume that when H-ZZPH(v) invoked (i.e., after step j), Φ_p^j and Γ_p^j satisfy the sequential invariant for step j, and zigzag restricted to steps $[1, j+k]$. Then at termination of H-ZZPH(v), its output, denoted by Φ_p^{j+k} and Γ_p^{j+k}, satisfies the sequential invariant for step $j+k$, and zigzag restricted to steps $[1, j+k]$.

Proof. We proceed by natural induction on the tree.

The base case is $k = 1$ (v is a leaf). By assumption, Φ_p^j and Γ_p^j satisfy the sequential invariant for step j and zigzag restricted to steps $[1, j+1]$. By definition of the hierarchical algorithm, M_p^c are given by (6), (7), (9), (10). By definition of the sequential algorithm, these are also update matrices of the sequential algorithm when $j+1$ is the last step of the zigzag (cf. Remark 3). By correctness of the sequential algorithm, the sequential invariant is satisfied by Φ_p^{j+1} and Γ_p^{j+1}. In particular, Φ_p^{j+1} and Γ_p^{j+1} are correct when restricted to steps $[1, j+1]$.

Now suppose $k > 1$ (v is not a leaf). Let l and r be the left and right child of v, respectively. By inductive hypothesis on l, at termination of H-ZZPH(l), $\Phi_p^{j+\frac{k}{2}}$ and $\Gamma_p^{j+\frac{k}{2}}$ satisfy the sequential invariant for step $j+\frac{k}{2}$ and zigzag restricted to steps $[1, j+\frac{k}{2}]$. At termination of LEFT-TO-RIGHT(M^{l}, r), we claim that they satisfy the sequential invariant for step $j+\frac{k}{2}$ and zigzag restricted to steps $[1, j+k]$. Once this is established, the main claim follows by inductive hypothesis on r.

It remains to prove the claim about the effect of LEFT-TO-RIGHT. Observe that the first update (13) of LEFT-TO-RIGHT(M^{l}, r) applies M^{l} to $\Phi^{j+\frac{k}{2}}$ and $\Gamma^{j+\frac{k}{2}}$, restricted to steps $[1, j+k]$. So after those updates the matrices restricted to steps $[1, j+k]$ are given by (14). After the last update (15) of RIGHT-TO-LEFT, the matrices are

$$\Phi_p^{j+\frac{k}{2}} = \left[\begin{array}{c|c|c|c} {}^{l}E_p^{l} & 0 & 0 & 0 \\ {}^{l}E_p^{r} & Z_p^{r} & C_p^{r} & {}^{r}F_p^{r} \end{array}\right] \qquad \Gamma_p^{j+\frac{k}{2}} = \left[\begin{array}{c|c|c|c} {}^{l}G_p^{l} & 0 & 0 & 0 \\ \hline {}^{l}R_p & 0 & B_p & {}^{r}H_p \\ \hline {}^{l}S_p & 0 & 0 & {}^{r}K_p \\ \hline 0 & 0 & 0 & {}^{r}L_p^{r} \end{array}\right]. \tag{17}$$

This can be easily verified by applying (15) to (14). Notice that ${}^{r}F_p^{l}$ has been zeroed out. Of course, some of the other submatrices have changed too. In particular, B_p and ${}^{r}H_p$ are independent. Now we are ready to verify parts of the sequential invariant.

1: Equality in (3) follows from the fact that we apply updates to both sides of the equation. The format of $\Phi_p^{j+\frac{k}{2}}$ and $\Gamma_p^{j+\frac{k}{2}}$ follows from (17).

2: The structure of $B_p^{j+\frac{k}{2}}$ is not affected by LEFT-TO-RIGHT. Independence of $B_p^{j+\frac{k}{2}}$ and $H_p^{j+\frac{k}{2}}$ was established above.

3, 4: The matrices involved are not affected by LEFT-TO-RIGHT, and the spaces that they should represent do not change, because they only depend on the current step, which remains $j+\frac{k}{2}$.

□

Running time. We show that the hierarchical algorithm runs on an n-step zigzag in time $O(M(n)\log n)$ if $M(n) = O(n^2)$ and in time $M(n)$ otherwise. The key step is showing that

H-ZZPH(v), excluding its recursive calls, can be implemented to run in time proportional to the number of leaf descendents of v. The missing proofs of lemmas below appear in Appendix B.

THEOREM 1. Let n be the length of the zigzag. Consider the invocation H-ZZPH(v) where v is an ancestor of k leaves. The output M_p^c of this procedure is of the form $M_p^c = P_p(I + U_p V_p^T)$, where P_p is a permutation matrix, and U_p, V_p have at most k columns and full rank. Furthermore, the output can be computed in $O(\frac{n}{k}M(k))$ time[2].

From this we easily get the main result of the paper.

COROLLARY 1. Zigzag persistent homology for a sequence of n steps can be computed in time $O(M(n)\log n)$ if $M(n) = O(n^2)$, and in time $O(M(n))$ otherwise.

Before proving Theorem 1, we provide tools for efficient computation that will be invoked repeatedly.

DEFINITION. An $n \times n$ matrix A has *width* k, for some $k \leq n$, if it can be written as $A = P(I + UV^T)$, where P is an $n \times n$ permutation matrix and U, V are $n \times k$ matrices.

LEMMA 3. Let U, V be $n \times k$ matrices, $k \leq n$, and let W be a $k \times k$ matrix. One can compute $U^T V$ and UW in $O(\frac{n}{k}M(k))$ time.

LEMMA 4. If A has width k_1 and B has width k_2, then AB has width $k_1 + k_2$. Given low-width representations A and B, low-width representation of AB can be computed in $O(\frac{n}{\max\{k_1,k_2\}}M(\max\{k_1,k_2\}))$ time. If low width representations of A and B have U-matrices of full rank, the same can be ensured for the low-width representation of AB within the same asymptotic time bound.

LEMMA 5. Let B be an $n \times k$ matrix. Given a width-k representation of A, AB can be computed in $O(\frac{n}{k}M(k))$ time.

LEMMA 6. Given a low-width representation of a width-k matrix, the low-width representation of its inverse can be computed in $O(\frac{n}{k}M(k))$ time.

Proof. (of Theorem 1) Let l and r be the left and right child of v, respectively.

By inductive hypothesis applied to l, $M^{\mathrm{l}} = P(I + UV^T)$, where U, V have full rank and at most $\frac{k}{2}$ columns. It is clear from (15) that W_p also has this structure. Furthermore, one can show that *columns of the top-right submatrix of W_p are spanned by columns of the U-matrix from the low-width representation of M^{l}*. To see why, first note that rows of $-({}^lE_p^l)^{-1}({}^rF_p^l)$ correspond to p-cycles/p-chains *removed* in the subsequence of l. Similarly, *nonzero* rows of $-B_p^T({}^rH_p - {}^lR_p({}^lE_p^l)^{-1}({}^rF_p^l))$ correspond to p-chains *added* in the subsequence of l. Now, observe that for each of these nonzero rows, the U-matrix contains a column which is nonzero only in that row. This column was added by the elementary update for the step in which that cycle/chain was added/removed, because the corresponding column of Φ_p/Γ_p was the pivot column in that elementary update.

So if W_p is applied to M_p^{l} in (15) using Lemma 4 (full rank version), we have that after the update M_p^{l} still has width $\frac{k}{2}$. Combining this with the inductive hypothesis applied to r, the result M_p^{c}, has width at most $\frac{k}{2} + \frac{k}{2} = k$. This proves the structure part of the claim.

[2]Assuming that M_p is represented in the above mentioned form, and that B_p is represented in some standard compact form for binary sparse matrices.

It remains to bound the running time of H-ZZPH. If $k = 1$, the claim is easy to verify – the algorithm manipulates $O(1)$ rows and columns, by inspecting, moving or copying them to compute M^j.

Suppose $k > 1$. Assume that $M(n) \notin O(n^2)$, and that for all $k' < k$ the algorithm runs in time bounded by $A\frac{n}{k'}M(k')$ for an absolute constant A to be determined later. By inductive hypothesis applied to l, H-ZZPH(l) runs in time $A\frac{2n}{k}M(\frac{k}{2})$.

First we discuss computing the updates in (13). From (12) it can be seen that Φ_p is of size at most $k \times n$ by construction, while Γ_p has width k. Since M^{l} has width at most $\frac{k}{2}$, $\Phi_p M_p^{\mathrm{l}}$ and $\Gamma_p M_p^{\mathrm{l}}$ can be computed in time $O(\frac{n}{k}M(k))$ by Lemma 5 and Lemma 4, respectively. Observe that the rank of $\Gamma_p M_p^{\mathrm{l}}$ is now bounded by $\frac{3k}{2}$. By Lemma 6, so $(M_{p-1}^{\mathrm{l}})^{-1}$ can be computed in $O(\frac{n}{k}M(\frac{k}{2}))$ time, and has rank at most $\frac{k}{2}$. Therefore, multiplication by $(M_{p-1}^{\mathrm{l}})^{-1}$ from the left can be done in time $O(\frac{n}{k}M(2k))$ by Lemma 4. However, after this left multiplication the rank of the result (updated Γ_p) goes back to at most k. This is because B_p goes back to being sparse, due to construction of elementary updates (6), (7), (9), (10).

Next we discuss computing the updates in (15). Matrix ${}^{l}E_p^l$ has at most $\frac{k}{2}$ rows (by construction), and at most $\times\frac{k}{2}$ columns (because at most one cycle or chain can be destroyed per zigzag step). Hence it can be inverted in $O(M(\frac{k}{2}))$ time using the algorithm of Bunch and Hopcroft [9]. Multiplication by $(B_p)^T$ is performed in $O(nk)$ time using sparsity of B_p. All other matrix multiplications in (15) involve a matrix of size at most $\frac{k}{2} \times \frac{k}{2}$ and a matrix of size at most $\frac{k}{2} \times n$, and can be computed in time $O(\frac{n}{k}M(\frac{k}{2}))$ by Lemma 3.

It is easy to check that $W_p^{-1} = 2I - W_p$ (changing signs in the top-right submatrix), so W_p^{-1} can be computed in $O(nk)$ time. Also note that both W_p and W_p^{-1} have the low-width structure, as defined in the statement of the theorem.

It remains to apply W_p and W_p^{-1} to M^{l}, Φ_p and Γ_p. It was already mentioned above that applying to M^{l} is done using Lemma 4, and hence takes $O(\frac{n}{k}M(k))$ time. The structure of Φ_p and Γ_p is given by (14). Φ_p is of size at most $k \times n$, so $\Phi_p W_p$ can be computed in time $O(\frac{n}{k}M(\frac{k}{2}))$ by invoking Lemma 5 twice. Γ_p has width at most k, so $W_{p-1}^{-1}\Gamma_p W_p$ can be computed in time $O(\frac{n}{k}M(k))$ by invoking Lemma 4 some small constant number of times.

Finally, the product in Equation (16) of Combine can be computed in $O(\frac{n}{k}M(k))$ time by Lemma 4, since it involves two matrices of width at most $\frac{k}{2}$, as mentioned above.

So we have shown that the total running time is at most $2A\frac{2n}{k}M(\frac{k}{2}) + C\frac{n}{k}M(2k)$, where the first term represents the two recursive calls, and the second term represents Left-to-Right and Combine (C depends on the number of times the above proof invokes the technical lemmas, and the big-Oh constants of the latter). This is easily verified to be less than $A\frac{n}{k}M(k)$, as long as $A \geq \frac{CM(2k)}{M(k)-4M(k/2)}$, which is bounded by a constant independent of k for large enough k.

If $M(n) = O(n^2)$, we postulate running time of $A\frac{n}{k}M(k)\log k$, and repeating the analysis under that assumption we get that H-ZZPH(v) runs in time $2A\frac{2n}{k}M(\frac{k}{2})\log\frac{k}{2} + C\frac{n}{k}M(2k)$, which is less than $A\frac{n}{k}M(k)\log k$ as long as $A \geq \frac{4C}{\log 2}$. □

References

[1] Herbert Edelsbrunner, David Letscher, and Afra Zomorodian. Topological persistence and simplification. *Discrete Comput. Geom.*, 28:511–533, 2002.

[2] Afra Zomorodian and Gunnar Carlsson. Computing Persistent Homology. *Discrete Comput. Geom.*, 33(2):249–274, 2005.

[3] D. Cohen-Steiner, H. Edelsbrunner, and J. Harer. Stability of persistence diagrams. *Discrete and Computational Geometry*, 37(1):103–120, 2007.

[4] H. Edelsbrunner and J. Harer. Persistent homology — a survey. *Surveys on Discrete and Computational Geometry. Twenty Years*, pages 257–282, 2008.

[5] Gunnar Carlsson and Vin de Silva. Zigzag persistence. *Foundations of Computational Mathematics*, 10(4):367–405, August 2010.

[6] Gunnar Carlsson, Vin de Silva, and Dmitriy Morozov. Zigzag persistent homology and real-valued functions. In *Proceedings of the Annual Symposium on Computational Geometry*, pages 247–256, 2009.

[7] James R. Munkres. *Elements of Algebraic Topology*. Westview Press, 1984.

[8] Tomasz Kaczynski, Konstantin Mischaikow, and Marian Mrozek. *Computational Homology*. Springer, 2004.

[9] James R. Bunch and John E. Hopcroft. Triangular Factorization and Inversion by Fast Matrix Multiplication. *Mathematics of Computation*, 28(125):231–236, 1974.

[10] Cecil Jose A. Delfinado and Herbert Edelsbrunner. An incremental algorithm for Betti numbers of simplicial complexes. In *SCG '93: Proceedings of the ninth annual symposium on Computational geometry*, pages 232–239, New York, NY, USA, 1993. ACM.

[11] Joel Friedman. Computing Betti numbers via combinatorial Laplacians. In *STOC '96: Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 386–391, New York, NY, USA, 1996. ACM.

[12] David Cohen-Steiner, Herbert Edelsbrunner, and John Harer. Extending Persistence Using Poincaré and Lefschetz Duality. *Found. Comput. Math.*, 9(1):79–103, 2009.

[13] David Cohen-Steiner, Herbert Edelsbrunner, John Harer, and Dmitriy Morozov. Persistent homology for kernels, images, and cokernels. In *SODA '09: Proceedings of the twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1011–1020, Philadelphia, PA, USA, 2009. Society for Industrial and Applied Mathematics.

[14] Dmitriy Morozov. *Homological Illusions of Persistence and Stability*. PhD thesis, Duke University, August 2008.

[15] Marian Mrozek, Paweł Pilarczyk, and Natalia Zelazna. Homology algorithm based on acyclic subspace. *Comput. Math. Appl.*, 55(11):2395–2412, 2008.

[16] Thomas Lewiner, Hélio Lopes, and Geovan Tavares. Optimal discrete Morse functions for 2-manifolds. *Comput. Geom. Theory Appl.*, 26(3):221–233, 2003.

[17] Ömer Eğecioğlu and Teofilo F. Gonzalez. A computationally intractable problem on simplicial complexes. *Comput. Geom. Theory Appl.*, 6(2):85–98, 1996.

[18] Michael Joswig and Marc E. Pfetsch. Computing Optimal Morse Matchings. *SIAM J. Discret. Math.*, 20(1):11–25, 2006.

[19] A. Zomorodian. The Tidy Set: A Minimal Simplicial Set for Computing Homology of Clique Complexes. In *Proc. ACM Symposium of Computational Geometry*, pages 257–266, 2010.

[20] Volker Strassen. Gaussian Elimination is not Optimal. *Numer. Math.*, 13:354–356, 1969.

[21] Don Coppersmith and Shmuel Winograd. Matrix multiplication via arithmetic progressions. *J. Symb. Comput.*, 9(3):251–280, 1990.

[22] Henry Cohn, Robert Kleinberg, Balazs Szegedy, and Christopher Umans. Group-theoretic Algorithms for Matrix Multiplication. In *FOCS '05: Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science*, pages 379–388, Washington, DC, USA, 2005. IEEE Computer Society.

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

A Correctness of the Sequential Algorithm

Since the operations in Section 3 are almost the same as in [6], we focus our proof of correctness on verifying the new parts of the invariant.

1: $D\Phi_p^j = \Phi_{p-1}^j \Gamma_p^j$. The equality is satisfied in Equation (3) because we apply the same updates to both sides of the equation. However, we must still ensure that the specific forms prescribed to matrices Φ_p^j and Γ_p^j remain intact: namely, that the blocks marked zero in (3) remain zero.

If there is a *birth after an addition*, the product $\Phi_p^j M_p^j$ moves the newly born cycle from submatrix F_p^j into Z_p^{j+1}; its boundary is zero by the case assumption. $\Gamma_p^j M_p^j$ moves the zero column into the portion of Γ_p^{j+1} corresponding to the cycle matrix Z_p^j. The product $(M_p^j)^{-1}\Gamma_{p+1}$ performs the corresponding row update in Γ_{p+1}, safely moving a row from the fourth to the second section of Γ_{p+1}. In birth after addition, matrix W_p^a is an identity, and therefore has no effect on the update.

If there is a *death after an addition*, M_p^j is identity that merely repartitions the matrices: the product $\Phi_p^j M_p^j$ moves the bounding chain from submatrix F_p^j into C_p^{j+1}. M_{p-1}^j performs a basis change in Z_{p-1}^j (with a matching update in B_p^j) that does not affect the structure of the matrices. Multiplication by W_p^a subtracts a multiple of the new column in B_p^j from H_p^j; these subtractions only introduce zeros. The effect of $(W_p^a)^{-1}$ on Γ_{p+1}^j is adding rows of L_{p+1}^j to K_{p+1}^j, a benign update as far as zeros are concerned.

If a simplex is *removed*, multiplying by matrix M_p^j zeros out the outgoing row of matrix C_p^j (and also Z_p^j in case of a death), while multiplying by W_p^r zeros out the remainder of the row in F_p^j. As a result, once the row is moved to the past, all but its first portion (now belonging to matrix E_p^{j+1}) is zero. The effect of the matrix $(M_{p-1}^j)^{-1}$ (from Equation (9)) in the product $(M_{p-1}^j)^{-1}\Gamma_p^j$ does not cross the partition boundaries.

2: *B_p^j has exactly one non-zero element per column, and at most one per row. H_p^j is independent of B_p^j.* In the removal cases, the only change to the matrix H_p^j is that it loses a row, and so does B_p^j; this loss does not affect the invariant. It's important to note that in case of the change of basis in Z_p^j after a death, B_p^j cannot stop satisfying the invariant, since the columns of Z_p^j that have a matching 1 in B_p^j cannot contain the outgoing simplex (the 1 would indicate that the cycles were boundaries, but the outgoing simplex has no cofaces). In case of a birth after removal, M_{p-1}^j performs a change of basis in Z_{p-1}^{j+1} such that its inverse updates B_p^{j+1} to once again satisfy the invariant.

In the case of addition, if birth occurs, matrix H_p^j loses a zero column, which again requires no update. If there is a death, B_p^j acquires a single column. Pre-multiplication of Γ_p^j by $(M_{p-1}^j)^{-1}$ ensures that B_p^j has a single 1 in that column, which is in the row of the lowest non-zero entry in the outgoing column of H_p^j; therefore, B_p^{j+1} remains a permutation matrix. We explicitly zero this row out via matrix W_p^a. Therefore, in all cases statement 2 of the Invariant is maintained.

3, 4: In the case of a removal of a simplex, the main updates performed to matrices Z and C are the same as in [6], with the exception of a few additional changes of basis in Z via left to right operations (which respect the right filtration). Therefore, the last two parts of the invariant are maintained.

In the case of an addition, the only difference from [6] is that the first column of the matrix F_p^j already has the correct form, and does not require any additional processing.

In the birth case, this is easy to see: the column $F_p^0[i]$ corresponding to the simplex σ_i contains a 1 in the row corresponding to this simplex. None of the operations ever remove this element from this column. Therefore, when we add σ_i we have a new cycle (recorded as the column $F_p^j[i]$), which we append to the matrix Z_p^j.

In the death case, the boundary of the column $F_p^j[i]$ is in the kernel of the map on the homology. If the lowest element in the matching column $H_p^j[i]$ is in the row l, then this cycle is in the span of the first l elements of the right filtration given by the matrix Z_{p-1}^j. Furthermore, since the matrix H_p^j is reduced with respect to B_p^j, the row l in B_p^j is zero, by definition. Therefore, there does not exist a cycle in the span of the first $l-1$ elements of the right filtration that's in the kernel of the homology map. Therefore, our update that adds to the matrix B_p^j a column with a 1 in row l, indicating that the updated cycle Z_{p-1}^{j+1} dies after the addition of simplex σ_i is correct. (This analysis reflects the Reduction Lemma in [6].)

In summary, the operations in Section 3 closely mimic the original zigzag persistent homology algorithm [6], and satisfy the additional Sequential Invariant requirements (1, 2).

B Proofs of running time lemmas

Proof of Corollary 1. This is a special case for $k = n$ (i.e. root of the tree), with Z_p, C_p, B_p, H_p and K_p "empty" (Z_p, C_p, B_p with zero columns, H_p, K_p with zero rows), with F_p equal to a permutation matrix that maps addition order to removal order, and $L_p = F_{p-1}^T D_p F_p$. □

Proof sketch of Lemma 3. Split matrices into $k \times k$ blocks, and treat blocks as entries. Use a fast $k \times k$ matrix multiplication algorithm to multiply blocks. □

Proof sketch of Lemma 4. Let $A = P_1(I + U_1V_1^T)$, $B = P_2(I + U_2V_2^T)$ be low-width representations of A and B. It is easy to check that $AB = P(I + UV^T)$, where

$$P = P_1P_2 \qquad U = \begin{bmatrix} P_2^TU_1 & U_2 \end{bmatrix}, \qquad V = \begin{bmatrix} P_2^TV_1 + V_2(U_2^T(P_2^TV_1)) & V_2 \end{bmatrix}.$$

is width-$(k_1 + k_2)$ representation of AB. If U_1 and U_2 have full rank, then we claim that $UV^T = U'(V')^T$ where U' has full rank. To get U', V', simply subtract from columns of U_2 their projections onto the column space of $P_2^TU_1$, i.e., compute

$$U' = U \begin{bmatrix} I & 0 \\ -(U_2^TU_2)^{-1}U_2^T & I \end{bmatrix} \qquad V' = V \begin{bmatrix} I & (U_2^TU_2)^{-1}U_2^T \\ 0 & I \end{bmatrix}$$

and then remove zero columns of U' and corresponding columns of V'. Since $U_2^TU_2$ is of size $k_2 \times k_2$, its inverse can be computed using Bunch and Hopcroft's algorithm [9] in time $O(M(\max\{k_1, k_2\}))$. The running time claim follows by applying Lemma 3 to the above expressions. □

Proof sketch of Lemma 5. Let $A = P(I + UV^T)$ be the width-k representation of A. Then one can compute AB as $W + U(V^TW)$, using Lemma 3. □

Proof sketch of Lemma 6. Let $P(I + UV^T)$ be given width-k representation. We have

$$P' = P^T, \qquad U' = -(PU)(I + V^TU)^{-1}, \qquad V' = PV.$$

Note that $I+V^TU$ is of size at most $k\times k$ and can be inverted using Bunch and Hopcroft's algorithm in $O(M(k))$ time. Everything else is computed by Lemma 3. □

Centre de recherche INRIA Saclay – Île-de-France
Parc Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 Orsay Cedex (France)

Centre de recherche INRIA Bordeaux – Sud Ouest : Domaine Universitaire - 351, cours de la Libération - 33405 Talence Cedex
Centre de recherche INRIA Grenoble – Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier
Centre de recherche INRIA Lille – Nord Europe : Parc Scientifique de la Haute Borne - 40, avenue Halley - 59650 Villeneuve d'Ascq
Centre de recherche INRIA Nancy – Grand Est : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex
Centre de recherche INRIA Paris – Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex
Centre de recherche INRIA Rennes – Bretagne Atlantique : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex
Centre de recherche INRIA Sophia Antipolis – Méditerranée : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399