Persistent Homology Over Directed Acyclic Graphs

Erin W. Chambers [*†]　　　David Letscher [*]

Abstract

We define persistent homology groups over any set of spaces which have inclusions defined so that the underlying graph between the spaces is directed and acyclic. This method simultaneously generalizes standard persistent homology, zigzag persistence and multidimensional persistence to arbitrary directed acyclic graphs, and it also allows the study of arbitrary families of topological spaces or point-cloud data. We give an algorithm to compute the persistent homology groups simultaneously for all subgraphs which contain a single source and a single sink in $O(n^4)$ time, as well as an algorithm to compute persistence for any arbitrary subgraph in the same running time. We then demonstrate as an application of these tools a method to overlay two distinct filtrations of the same underlying space, which allows us to calculate significant barcodes using considerably fewer points than standard persistence.

1 Introduction

Since its introduction in [11], the concept of topological persistence has become one of the most utilized tools in computational geometry. It has found numerous applications in diverse areas such as surface reconstruction, sensor networks, bioinformatics, and cosmology.

In this paper, we give a generalization of persistence to spaces where the underlying inclusions form a directed acyclic graph. This simultaneously generalizes both zigzag and multidimensional persistence, which can be viewed as special cases of these underlying graphs on the maps between the spaces.

We then give algorithms to compute the persistence homology for specific subgraphs. Our algorithms are analyzed in terms of e and v, which are the number of edges and vertices in the directed acyclic graph G, as well as l, the longest directed path in G. We will in general let $n = v + e$ in our algorithm analysis for simplicity, yielding the $O(n^4)$ time algorithms described in Section 3 and 4.

Potential applications of this are extensive, including any spaces where inclusions are more general than previous settings. We present one such application in Section 5 which deals with barcodes of spaces in such directed acyclic graphs. Our experiment looks at a sampled double torus with 5000 points sampled, as well as a subgraph of two different 200 point samples which include into a 400 points subsample. As expected, the diagram for each smaller sample is quite noisy; however, overlaying the two 200 point samples in our formulation yields a surprisingly good identification of the same features that are obvious in the 5000 point sample.

1.1 Background and related work

For completeness, we briefly survey some results from persistent homology with an emphasis on tools and techniques used in this paper, although a full coverage is beyond the scope of this paper. See any of the recent books or surveys on topological persistence for full coverage of this broad topic and its applications [20, 10, 9, 12].

The original algorithm for computing persistent homology [11] was quickly followed by more work on efficiently computing these groups in several different settings [21] as well as showing desirable properties such as stability under certain assumptions [6] and accurate reconstruction in the presence of noise [5].

The use of barcodes [4] is of particular interest to us, as it relates to our application in Section 5. Through computation of barcodes, one can capture a representation of a topological feature by an interval that represents how long this interval survives in our filtration; at a high level, more important topological features will persist longer in a filtration and will therefore have longer intervals.

Several extensions to topological persistence have appeared, most notably zigzag persistence [1] and multidimensional persistence [3]. In each case, these generalize the simple inclusions from the main algorithm to more general setups.

Zigzag persistence considers spaces with maps of the form $X_1 \leftrightarrow X_2 \leftrightarrow \ldots \leftrightarrow X_n$, where the maps can go in either direction. These maps between the spaces induce maps between chain complexes which pass to homology as homomorphisms $H(X_1) \leftrightarrow H(X_2) \leftrightarrow \ldots \leftrightarrow H(X_n)$; this is known as a zigzag module. Recent work in this setting includes an algorithm which examines the order of the necessary matrix multiplications quite carefully and is able to get a running time for sequence of n simplex deletions or additions which is dominated by the time to multiply two $n \times n$ matrices [16].

Multidimensional persistence extends standard persistence to work not just along a single dimension, but rather on maps between spaces which are parametrized with respect to multiple dimensions [3]. While no analog to the barcode exists in this setting to capture all topological information, the rank invariant is a natural extension of the barcode which captures relevant topological information in many settings.
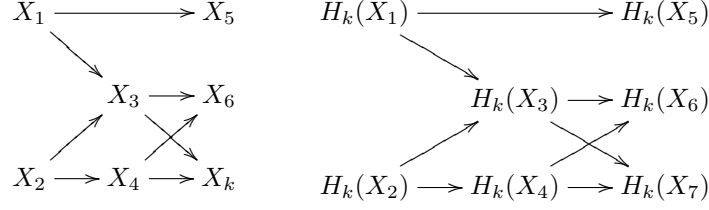
Figure 1: A graph filtration and the induced directed graph of homology groups.

2 Definition

We recall some relevant definitions and background before presenting our definition of persistent homology over directed acyclic graphs. For a full presentation of homology groups see any introductory text in algebraic topology [13, 17].

For a simplicial complex X and an abelian group A, the k-chains $C_k(X)$ is the space of formal linear combinations of the k-simplices of X where the coefficients are in A. $\partial_k : C_k(X) \to C_{k-1}$ is a linear map that calculates the boundary of a chain. The cycle group $Z_k(X) = \{c \in C_k(X) \mid \partial_k(c) = 0\} = ker(\partial_k)$ and the boundary group $B_k(X) = \{c \in C_k(X) \mid \exists d \in C_{k+1}(X) \text{ with } \partial_{k+1}d = c\} = im(\partial_{k+1})$. The homology group is defined as $H_k(X) = Z_k(X)/B_k(X)$. Note that if A is a field then $C_k(X), Z_k(X), B_k(X)$ and $H_k(X)$ are all vector spaces.

Given a filtration $X_0 \subset X_1 \subset \cdots \subset X_n$, the persistent homology group $H_k^p(X_i)$ can defined in multiple ways. Traditionally, it is defined as $Z_k(X_i)/B_k(X_{i+p}) \cap Z_k(X_{i+p})$ and can be viewed as a quotient group of $H_k(X_{i+p})$. An equivalent definition is $H_k^p(X_i) = im(i_*)$, where $i_* : H_k(X_i) \to H_k(X_{i+p})$ is the map induced by the inclusion $i : X_i \to X_{i+p}$. So $H_k^p(X_i)$ can also be thought of as a subgroup of X_{i+p}.

When viewed from the perspective of zigzag persistence and the use of the zigzag module, the persistence group for some interval can be thought as the subgroups which are common to all of the the homology groups. This motivates the definition of persistence for arbitrary directed graphs. The main restriction on this graph is that it must be acyclic, which is a natural for any graph which represents a set of inclusions.

Definition 2.1. *For a directed acyclic graph $G = (V, E)$, a graph filtration $\mathcal{X}_G$ of a topological space X is a pair $(\{X_v\}_{v \in V}, \{f_e\}_{e \in E})$ such that*

1. *$X_v \subset X$ for all $v \in V$*

2. *If $e = (v_1, v_2) \in E$ then $f_e : X_{v_1} \to X_{v_2}$ is continuous embedding (or inclusion) of X_{v_1} into X_{v_2}.*

Given a graph filtration there is a corresponding directed graphs of homology groups, where each edge corresponds to the map induced by inclusion $(f_e)_* : H_k(X_{v_1}) \to H_k(X_{v_2})$, see Figure 1.

We will define persistent homology groups for any connected subgraph of G as the "largest" group that injects into all of the homology groups at the vertices such that the image group is preserved by all of the edges, see figure 2. To make this definition precise, we will will assume that the coefficients for our homology groups are some field $\mathbb{F}$. Throughout this paper, we will assume that $H_k(X)$ means $H_k(X; \mathbb{F})$; this ensures that all of the homology groups are vector spaces over $\mathbb{F}$. In this setting, we can consider one to be "larger" than another if it has larger rank or dimension. See Figure 2 for an example diagram.

Definition 2.2. *Given a graph filtration $\mathcal{X}_G$ and a connected subgraph $G' \subset G$, the G'-persistent homology group, $H_k^{G'}(\mathcal{X}_G)$, is the largest rank vector space $\mathcal{P}$ equipped with linear injections $i_v : \mathcal{P} \to H_k(X_v)$ for all $v \in G'$ such that $f_e \circ i_u = i_v$ for all edges $e = (u, v) \in G'$. In other words, the graph of homology groups along with the inclusions of $\mathcal{P}$ form a commutative diagram.*
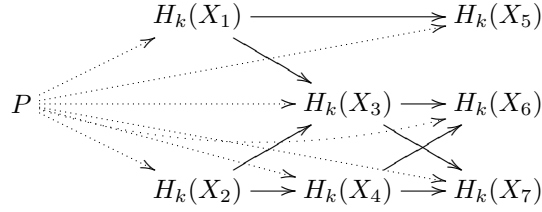
2

Figure 2: Persistence is defined as the "largest" group that injects into every homology group making the diagram commute.

Note that $H_k^{G'}(\mathcal{X}_G)$ always exist if the homology groups are finitely generated and are unique up to isomorphism. The maps, however, are not necessarily unique. In Section 2.3, we connect this definition to persistence to known models and prove that it generalizes standard persistence, zigzag persistence and multidimensional persistence.

2.1 Flow graph view

There is also an equivalent and quite interesting view of the structure of the persistent homology group over G', where G' is again a subgraph of the directed acyclic graph G. In the definition of $H_k^{G'}(\mathcal{X}_G)$, we assumed that there are maps i_v for every vertex of G. However, since the diagram commutes, we only need to define these injections from P to the source vertices of G', and we can then use the edge homomorphisms from G' to construct the remainder of the homomorphisms from P to any vertex in G'.

We note that in the setting, all of the composite maps will also be injective. Also, since $\mathcal{P}$ injects in each of the sink vertices, we can define maps from the homology groups at these sinks to $\mathcal{P}$, such that the composite map $\mathcal{P} \to H_k(X_v) \to \mathcal{P}$ is the identity.

In essence, this means that for any subgraph G', the relevant information can be captured by collapsing G' to a directed bipartite graph, where one independent set consists of the sources in G' and the other is the sinks. The choice of edge homomorphisms from a source to a sink in G' is irrelevant due to the commutativity of the diagram, and therefore we can replace G' with a much simpler conceptual picture. (See Figure 3 for an example.)

This allows an alternate view of this computation of P. Consider collapsing an arbitrary path from each sink to each source of G' into a single edge which is the composition of that path; this results in a bipartite graph H between the sources and sinks of G'. Now in order to compute P in this reduced graph H, we can view this as a flow computation: we place P as a "source" and a "sink" and turn H into a bipartite flow network where the goal is to maximize the rank of P such that the induced homomorphism from each edge in H is a injection and the composite math from P to P through *any* edge is the identity.

2.2 Persistence module

A graph of vector spaces of a fixed directed acyclic graph G form a module structure. For standard persistence this is the same as the persistence module [4] and also coincides with the zigzag module [1].

Definition 2.3. *For a directed acyclic graph $G = (V, E)$, a* commutative *G-module is the pair $(\{W_v\}_{v \in V}, \{f_e\}_{e \in E}$ where V_v is a vector space and for any edge $e = (v, w)$, $f_e : W_v \to W_w$ is a linear map with the condition that the resulting diagram is commutative.*

This definition provides the framework for discussing the persistence module for a graph that extends the definition for the zigzag persistence module.
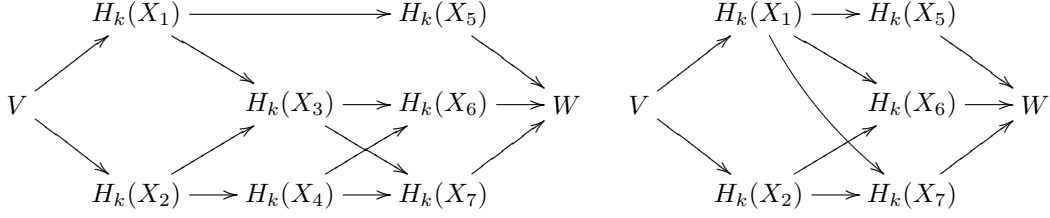
3

Figure 3: Persistence can also be defined in terms of these flow graphs as the maximal dimension of the image of the composite map $V \to W$.
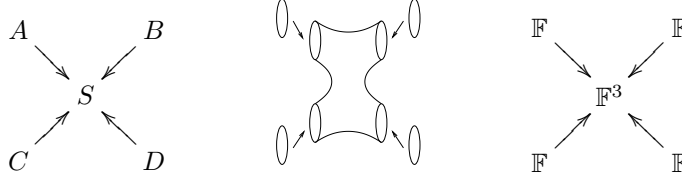

Figure 4: (a) The persistence module for the graph G is not finite-type. (b,c) The persistence module for these four curves in a punctured sphere is indecomposable.

Definition 2.4. *The persistence module for $\mathcal{X}_G$, $\mathcal{PH}_k(\mathcal{X}_G)$, is the commutative G-module formed from the graph of homology groups.*

The theory for G-modules is very similar to the zigzag persistence module. Given a G-module $\mathcal{W} = (\{W_v\}, \{f_e\})$, a G-module $\mathcal{V} = (\{V_v\}, \{g_e\})$ is a *submodule* if $V_v \subset W_v$ for all v and $f_e|_{V_v} = g_e$. Similarly, given two commutative G-modules $\mathcal{V} = (\{V_v\}, \{f_e\})$ and $\mathcal{W} = (\{W_v\}, \{g_e\})$, we can define their connected sum $\mathcal{V} \oplus \mathcal{W}$ as $(\{V_v \oplus W_v\}, \{f_e \oplus g_e\})$. A commutative G-module is said to be *indecomposable* if it cannot be written as a non-trivial connected sum. Any commutative G-module, $\mathcal{V}$, can be written as $\mathcal{V} = \mathcal{V}_1 \oplus \cdots \oplus \mathcal{V}_n$, where each $\mathcal{V}$ is indecomposable. For a connected subgraph G' of G, we will define the commutative G-module $\mathbb{F}_{G'}$ as the module with a copy of $\mathbb{F}$ at each vertex of G'; we will put the identity map on each edge of G' and make every other map trivial. We will call this module *elementary*.

In representation theory, a directed graph is known as a quiver [7] and a representation of that quiver is an assignment of vector spaces to each vertex and a linear map for each edge. The study of quivers provides the underlying theory for the zigzag persistence module. If we add the conditions that the quiver representation must be a commutative diagram, then we get a quiver with relations. Every finite dimensional algebra occurs as a quiver with relations. The following theorem allows us to decompose our persistence modules.

Theorem 2.5 (Krull-Remak-Schmidt Theorem for Finite-Dimensional Algebras [14])**.** *The decomposition of a commutative G-module is unique up to isomorphism and permutation of the summands.*

In the case of zigzag persistence the relevant indecomposable modules are always are elementary. This is implied by Gabriel's theorem [7] which provides an enumeration of indecomposable modules for particular graph types. Figure 4 gives an example of an indecomposable module that is not elementary. The example consists of a sphere with four punctures, and the inclusion of each of the four boundary components. Unfortunately, the existence of such examples tells us that there is no simple "barcode" representation for DAG persistence. However, the indecomposable summands can be thought of as an "annotated" barcode.

4

2.3 Relationship with other models of persistence

We saw in Section 2 the definition for standard persistence. Multidimensional persistence can also be defined in terms of the image of a map. Consider a multifiltration, or d-dimensional grid of spaces equipped with a partial ordering on vertices, where $u = (u_1, \ldots, u_d) \leq v = (v_1, \ldots, v_d)$ if and only if $u_i \leq v_i$ for all i. The *rank invariant*, $\rho_{X,k}(u,v)$, is defined as the dimension of the image of the induced map $H_k(X_u) \to H_k(X_v)$ [1]. Since the diagram commutes, this map can be found following any path from u to v in the graph. Zigzag persistence is defined in terms of the zigzag module. The definition of a commutative G-module and a τ-module for zigzag persistence are identical for a zigzag graph. The following proposition specifies how DAG persistence generalizes these three notions of persistence.

Proposition 2.6. *Suppose $\mathcal{X}_G$ is a graph filtration of X then*

1. *(Standard persistence) If G is a the graph corresponding to the filtration $X_0 \to X_1 \to \cdots \to X_n$ and $I_{i,p}$ is the subgraph consisting of vertices $\{X_i, \ldots, X_{i+p}\}$ then $H_k^{I_{i,p}}(\mathcal{X}_G) \cong H_k^p(X_i)$. Furthermore, $\mathcal{PH}_k(\mathcal{X}_G)$ coincides with the persistence module.*

2. *(Zigzag persistence) If G is the graph for zigzag persistence $\mathbb{X} = X_0 \leftrightarrow X_1 \leftrightarrow \cdots \leftrightarrow X_n$ where each arrow could go in either direction then $H_k(\mathbb{X}) \cong \mathcal{PH}_k(\mathcal{X}_G)$.*

3. *(Multidimensional persistence) If $\mathcal{X} = \{X_v\}_{v \in \{0,\ldots,m\}^d}$ is a multifiltration with underlying graph G. If $G_{u,v}$ is the subgraph with vertices $\{w \in G \mid u \leq w \leq v\}$ then the rank invariant $\rho_{X,k}(u,v) = \dim H_k^{G_{u,v}}(\mathcal{X}_G)$.*

2.4 An example

In Figure 5 we see an example of a set of spaces with inclusions that form a directed acyclic graph. At the top level is a genus two surface; the directed arrows indicate the inclusion maps in our directed acyclic graph, down to our two source vertices in the graph which include one space with two disjoint annuli and one space that is a disk with three boundaries. The graph forms a poset that demonstrates the non-trivial intersections and unions of the three surfaces. In Figure 5, middle, we see the persistence diagram for the entire space, and on the left are diagrams showing the indecomposable summands of the module. Notice that each of these submodules is elementary, yielding a "barcode" like description.

3 Single Source-Single Sink Subgraphs

In this section, we consider $\mathcal{X}_G$ where G is a directed acyclic graph with a single source vertex s. To limit the number of cells which f_e can introduce (where $e = (u,v) \in G$), we assume that each inclusion f_e adds a single cell to the underlying space. As a result, we will also assume that the space X_s will consist of the empty set. This assumption is both standard in most persistence algorithms [21, 16] and quite natural given that we can decompose any inclusion map into a series of inclusions of one simplex at a time.

First, note if we consider a single source-single sink subgraph, it suffices to take any directed path between them in G and compose the maps, since the persistent homology diagram commutes. Therefore, a straightforward application of the $O(n^3)$ standard persistence algorithm would yield a $O(n^5)$ algorithm. Using tools from recent work to compute zigzag persistence in matrix multiple time [16] would give a running time of $O(n^2(M(l) + l^2 \log^2 l))$, where l is the length of the longest path between any source and sink and $M(l)$ is the time to multiply two $l \times l$ matrices.

Theorem 3.1. *If G is a directed acyclic graph whose longest path has length l, then the persistent homology groups with coefficients from a finite field for all single source-single sink subgraphs of G can be calculated in $O(v^2 l^2)$ time.*
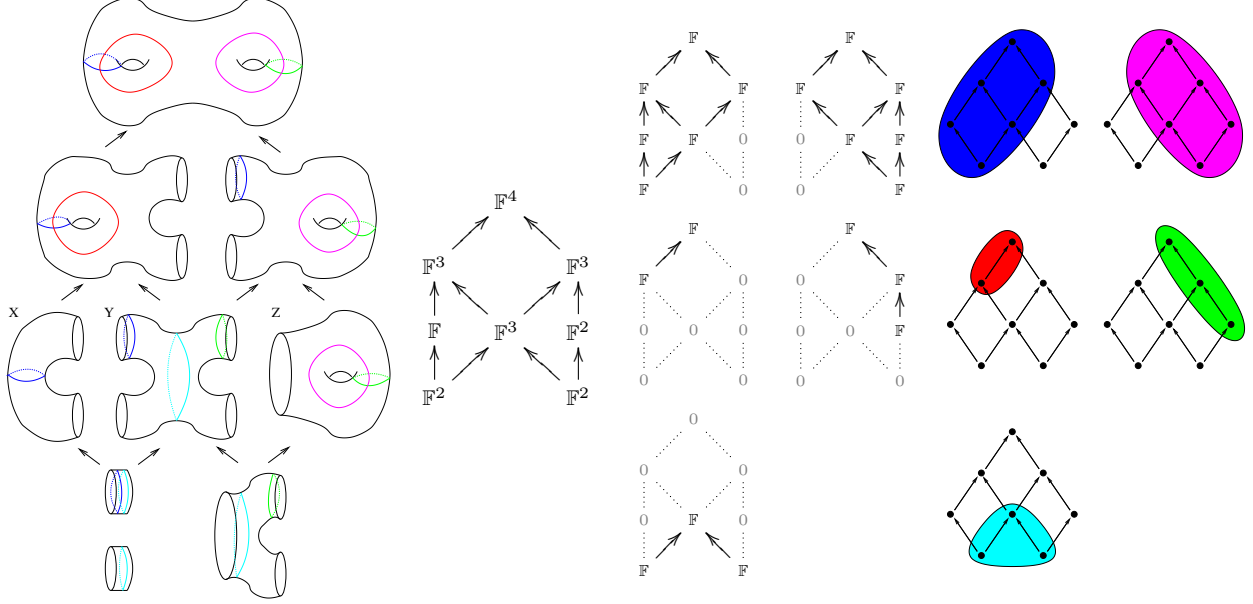
Figure 5: (a) A genus two surface that is the union of three subsurfaces (X, Y and Z), (b) its persistence module $PM_1(\mathcal{X}_G)$, (c) the indecomposable summands of the persistence module . Each colored curve on the left corresponds to a generator of an indecomposable submodule on the right and (d) the "barcodes" for this example.

We note that if we remove the assumption that coefficients are from a finite field, our theorem still accurately counts the number of arithmetic operations, although running times for each operation might take longer. In 3 dimensions, however, no information is lost when persistent homology is taken with respect to a finite field [15], so this is not an overly restrictive assumption.

This algorithm is actually a relatively straightforward extension of a standard persistence algorithm over finite fields [21]. In this paper they initially represent the homology basis as a matrix, but they are able to eliminate the need for row operations. This results in a simpler data structure, namely an array where each simplex is annotated and marked appropriately (as basis elements are created and destroyed) during the course of the algorithm. The running time is equivalent to Gaussian elimination, or $O(m^3)$ where m is the number of simplices in the filtration.

We will rely on their matrix representation (rather than on their improved data structure) for our algorithm. We first fix a single source vertex s, and compute a (directed) tree T which reaches all possible sinks in G. If our underlying tree is a path, then we can directly apply their algorithm in $O(l^3)$ time. In the more general case where it is not a path, however, we must consider the matrix representation more carefully.

In general, the homology basis at dimension k will be a matrix with each $(k+1)$-simplex represented as a column and each k-simplex represented as a row. While in their algorithm this resulted in a running time of $O(n^2)$ for each operation (due to the Gaussian elimination step), we note that in our algorithm, each such operation is bounded by $O(l^2)$, since the fact that G' only has a path of length l means that our matrix has size at most $l \times l$.

We can also store a representation of the current matrix at each branch point in T, which means we will not need to repeat our calculation starting at the source each time. Since we have at most n edges in T and adding each edge triggers an $O(l^2)$ computation, we take overall $O(nl^2)$ time to compute each path from

6

a single source to all sinks. Repeating this for each source vertex yields the running time in Theorem 3.1. We do note that all of the matrices involved are sparse, as in [21], so we expect the running time in this setting to be faster in practice. We can also adapt this algorithm to multidimensional persistence to give an improvement over the known polynomial time algorithm [2].

Proposition 3.2. *All of the rank invariants for d-dimension lattice with n nodes can be calculated in $O(n^{4-1/d})$ time.*

4 General Subgraphs

There are potentially infinitely many subgraphs of a graph G, so it is unreasonable to expect to be able to calculate persistent homology for all subgraphs in an efficient manner. However, in this section we will given an algorithm calculate $H_k^{G'}(\mathcal{X}_G)$ for fixed G' in $O(n^4)$ time.

Our algorithm is based on the Gram-Schmidt orthonormalization procedure [19]. Gram-Schmidt does not work if $\mathbb{F}$ is a finite field, since we do not have an inner product space in finite fields. Instead, we will work with rational coefficients for our homology groups. As in the last section, we will assume that G has a single source and each edge adds a single simplex in the graph filtration $\mathcal{X}_G$.

At a high level, our algorithm is simply an inductive one that adds each simplex to the subgraph one at a time. At each stage, we will maintain the persistent homology for the current subgraph; however, this requires that we store a matrix representing the boundary, chain, and cycle groups at each vertex so that we can track changes to the persistent homology group across the entire graph as each edge is added. The end result is the following:

Theorem 4.1. $H_k^{G'}(\mathcal{X}_G)$ *can be found using $O(el^3)$ arithmetic operations.*

The running time for this algorithm could actually be much worse since since we are not working in a finite field, so there is the potential of an exponential growth in the numbers involved in these calculations. Ideally, we would work in a finite field to avoid this issue, but as previously noted the Gram-Schmidt process will not work in finite fields.

Let $G_0, \ldots, G_m$ be a sequence of subgraphs of G' where G_0 consists only of the vertices of G', $G_m = G'$ and G_{a+1} is G_a with a single additional edge. We will calculate persistent homology the G_a inductively.

For each vertex i of G', we start with an $s \times s$ matrix M_i, where s is the number of k simplices of X_i. The columns of M_i will be partitioned into three subset: $M_i = (B_i \mid Z_i \mid C_i)$. The rows of M_i correspond to the k-simplices of X_i. Initially, the columns in B_i span the boundaries $B_k(X_i)$ and the the cycle space $Z_k(X_i)$ is spanned by the combination of the columns of Z_i and B_i. The entries of C_i fill out a basis of $\mathbb{Q}^s$. Let b_i, z_i and c_i be the number of columns in each portion of M_i. These entries can be found by calculating the homology groups $H_k(X_i)$ in $O(M(s))$ time [8]. We will abuse notation slightly and use B_i and Z_i to represent both the specified columns of the matrix and the subspace spanned by those columns. Similarly, we use $Z_i + B_i$ to denote the subspace spanned by the union of Z_i and B_i.

The first step of the algorithm will be to use Gram-Schmidt to turn M_i into an orthonormal matrix, which is a matrix with the norm of each column equal to one and the dot product between distinct columns equal to zero. Let $(M_i)_c$ denote the c-th column of M_i. The orthonormalization procedure proceeds as follows:

1. Replace $(M_i)_1$ with $\frac{(M_i)_1}{||(M_i)_1||}$

2. For $j = 2..s$, let $v = (M_i)_j - \sum_{l=1}^{j-1}((M_i)_j \cdot (M_i)_l)(M_i)_l$ and replace $(M_i)_j$ with $\frac{v}{||v||}$.

This process does not change the span of the first j columns. So the first b_i columns still span $B_k(X_i)$, and the first $b_i + z_i$ columns span $Z_k(X_i)$.

The invariants we will maintain through this inductive algorithm are:

- For every component C of G_a and vertex $i \in C$, $H_k^C(\mathcal{X}_G) \cong (Z_i + B_i)/B_i$.

- For a component C of G_a, the possible maps $\{g_i : H_k^C(\mathcal{X}_G) \to C_k(X_i)\}$ that realize persistence for the graph C are precisely the ones with $im(g_i) \subset Z_i + B_i$ and $im(g_i) \cap B_i = 0$ and satisfy the commutativity assumptions $f_e \circ g_i = g_j$ for every edge $e = (i, j)$ of C.

- For any edge $e = (i, j)$ of G_a, we have $f_e(Z_i + B_i) = Z_j + B_j$ and $f_e^{-1}(B_j) = B_j$.

- M_i is orthonormal.

As edges are added to the subgraphs, the groups $Z_i + B_i$ will shrink and the group B_i will grow, but these invariants will be maintained.

Consider adding an edge e connecting vertices i and j, and consider injections that realize persistence at both vertices and yield a commutative diagram with the following conditions:

$$
\begin{array}{ccc}
 & V & \\
g_i \swarrow & & \searrow g_j \\
C_k(X_i) & \xrightarrow{\quad f_e \quad} & C_k(X_j)
\end{array}
\qquad\qquad
\begin{array}{l}
g_i(V) \subset B_i + Z_i \\
g_i(V) \cap Z_i = 0 \\
g_j(V) \subset B_j + Z_j \\
g_j(V) \cap Z_j = 0
\end{array}
$$

To realize persistence for the components of the graph G_a, the above conditions must be met. Any realization of persistence for the larger graph, G_{a+1}, must also satisfy a few additional properties:

1. For the diagram to commute, $g_j(V)$ must be completely contained in the image $f_e(Z_i + B_j)$ which can be guaranteed if we replace $Z_j + B_j$ with $(Z_j + B_j) \cap f_e(Z_i + B_i)$.

2. Also, $g_i(V)$ must be contained in $f_e^{-1}(Z_j + B_j)$. To ensure this we will replace $Z_i + B_i$ with $(Z_i + B_i) \cap f_e^{-1}(Z_j + B_j)$.

3. If $g_i(V)$ intersects $f_e^{-1}(B_j)$ non-trivially then the composition $g_j = f_e \circ g_i$ intersections B_j non-trivially. Thus we must ensure that $g_i(V) \cap f_e^{-1}(B_j) = 0$ so we will replace B_i with $B_i \cup f_e^{-1}(B_j)$.

4. Finally, if $g_i(V) \cap B_i = 0$ then $g_j(V) \cap f_e(B_i) = 0$, so $g_j(V)$ must meet $f_e(B_i)$ trivially. So we will replace B_j with $B_j \cup f_e(B_i)$.

Notice that after these changes $f_e(Z_i + B_i) = Z_j + B_j$ and $f_e^{-1}(B_j) = B_i$. However, this property is no longer guaranteed at all of the other edges. To fix this, we will traverse the graph identifying deficiencies. At any such edge $e = (i, j)$ we will make the following replacements:

1. Replace $Z_j + B_j$ with $(Z_j + B_j) \cap f_e(Z_i + B_j)$.

2. Replace B_i with $B_i \cup f_e^{-1}(B_j)$.

This process must terminate since any time changes are made either the dimension of some $Z_i + B_i$ or Z_i decreases; we bound the exact number of iterations later when discussing the specifics of the algorithm.

Lemma 4.2. *The invariants are satisfied for the graph G_{a+1} if they are met for G_a.*

To implement the algorithm, we must perform matrix operations on each M_i. Our basic operations are calculating intersections and the union of subspaces; we also note that unions are the same as doing intersections of perpendicular subspaces.

As a map on chain complexes, f_e is either the identity map (when there is a death of a k-cycle) or its image has codimension one (when there is a birth of a cycle). In the second case, the coordinate corresponding to the new simplex is always zero. Because of this, it is natural to think of chains in $C_k(X_i)$ also as chains in $C_k(X_j)$. Furthermore, chains in $C_k(X_j)$ that are zero in the extra coordinate belong to $C_k(X_i)$. We will denote the coordinate vector for this extra coordinate by u_{ij}. With this perspective, we can summarize our operations as follows:

$Z_j + B_j \Rightarrow (Z_j + B_j) \cap f_e(Z_i + B_i)$	Add u_{ij} and each column of C_i to C_j and re-orthonormalize
$Z_i + B_i \Rightarrow (Z_i + B_i) \cap f_e^{-1}(Z_j + B_j)$	Add each column of C_j projected to $u_{ij}^{\perp}$ to C_i and re-orthonormalize
$B_i \Rightarrow B_i \cup f_e^{-1}(B_j)$	Add each column of B_j to B_i and re-orthonormalize
$B_j \Rightarrow B_j \cup f_e(B_i)$	Add each column of B_i to B_j and re-orthonormalize

Notice instead of shrinking $Z_i + B_i$, we perform the equivalent operation of enlarging its perpendicular subspace $(Z_i + B_i)^{\perp} = C_i$. The process of adding a column v to C_i proceeds like a step of Gram-Schmidt. First, project v to a vector, $\bar{v}$, that is perpendicular to the rest of C_i. If $\bar{v}$ is non-zero, it is added to C_i. At this point M_i is no longer square. However, the vectors in B_i and then Z_i can be reduced to be perpendicular to $\bar{v}$. At some point, one of these vectors will be zero and can be removed from M_i. Adding a vector to B_i works in the same manner. Since the matrices involved are at most $l \times l$, this single update can be performed in $O(l^2)$ arithmetic operations.

Each update either increases c_i or increases z_i, and we will avoid adding the same column twice. So, at most $O(l)$ updates can occur for each matrix. Each update that occurs for a matrix triggers potential updates for all of its neighbors. If d_i is the degree of that vertex then it takes $O(d_i l^3)$ operations to do all of the updates to vertex i and perform checks on each neighbor to see if they must be updated. So the total number of arithmetic operations is $\sum O(d_i l^3) = O(e l^3)$, as claimed in Theorem 4.1.

5 Applications

In Figure 6, we show a compelling example of the utility of our algorithm. Each subfigure is a persistence diagram, where each cycle is considered as a pair of birth and death times. The persistence of each cycle is the difference in these times which is equal to the distance to the diagonal. The space considered is simple a double torus sampled with 5000 points, at which level the persistent features, which are the 4 generators of homology, are clearly seen as significant. Note that these four points blur together in pairs in the figure and are difficult to distinguish from each other. In the remaining pictures, we calculate persistent homology for a simple directed acyclic graph that consists of two different 200 point subsamples including into a larger 400 point sample. We note that individually, each sample's persistent homology is quite noisy and does not indicate the 4 generators at all. However, the persistent homology for the directed acyclic graph (shown on the top right) clearly separates the 4 main generators from the noise, at a far lower level of sampling than is possible with standard persistent homology. The table shows the persistence values for the four generating cycles for the surface and the next largest cycle. The values have been renormalized to the length of the cycle with largest persistence in the original sample.

Additional possible applications of this are numerous. For example, we could use directed acyclic graphs of various witness complexes and seek improved results using a DAG over a small set of witness complexes; recent work using zigzag persistence seems relevant in this setting [18]. Also, the example in Section 2.4 demonstrates how homology classes from pieces of a space can be "aligned" similar to the bootstrapping method of [18].

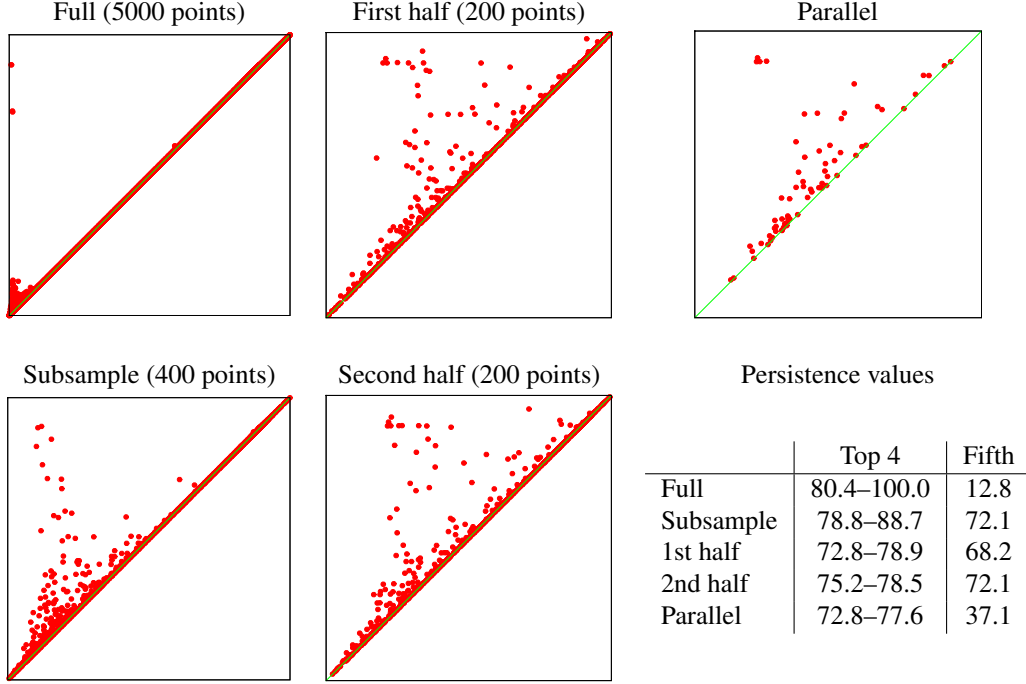|			Top 4	Fifth
Full			80.4–100.0	12.8
Subsample			78.8–88.7	72.1
1st half			72.8–78.9	68.2
2nd half			75.2–78.5	72.1
Parallel			72.8–77.6	37.1

Figure 6: Top left: the persistent homology for a 5000 point sample of a double torus. Bottom left: The persistent homology for a subsample of 400 points. Middle: The persistent homology for two disjoint subsamples of the 400 points. Top right: The homology for the directed acyclic graph that includes each 200 point sample into the 400 point sample.

6 Future Work

A practical algorithm for decomposition the persistence module into indecomposibles would be useful in finding an annotated barcode representation of the persistence module. Such an algorithm could generalize the right filtrations used in [1, 16].

There is the potential to improve our algorithm for all single source-single sink subgraphs using tools from recent work that computes zigzag homology in matrix multiply time [16]. Consider a directed tree T in the DAG for a single sink s; this tree is composed of paths which are zigzag homology complexes, so we can use the recent zigzag algorithm for each path. If this tree has common subpaths from previous calls to the zigzag algorithm, we could potentially speed up our algorithm by using this information. On the other hand, if the tree has no such common subpaths, then intuitively we should be able to balance the fact that the disjoint paths sum to n (so that we either have many very short paths or the tree is a single path). Balancing this recursion based on their analysis, however, give no better running time than the naive $O(n^2 M(l))$ algorithm we briefly outlined in Section 3, since the time to combine the information from a previous call with the new recursive call will take longer than the call itself. A better algorithm that uses this information successfully is an interesting direction to consider.

Acknowledgments

References

[1] G. Carlsson and V. de Silva. Zigzag persistence. *Foundations of Computational Mathematics*, 10:367–405, 2010.

[2] G. Carlsson, G. Singh, and A. Zomorodian. Computing multidimensional persistence. *Journal of Computational Geometry*, 1(1), 2010. Preliminary version appeared in ISAAC 2009.

[3] G. Carlsson and A. Zomorodian. The theory of multidimensional persistence. *Discrete Comput. Geom.*, 42(1):71–93, May 2009.

[4] G. Carlsson, A. Zomorodian, A. Collins, and L. Guibas. Persistence barcodes for shapes. *International Journal of Shape Modeling*, 11:149–187, 2005.

[5] F. Chazal and A. Lieutier. Weak feature size and persistent homology: computing homology of solids in rn from noisy data samples. In *Proceedings of the twenty-first annual symposium on Computational geometry*, SCG '05, pages 255–262, New York, NY, USA, 2005. ACM.

[6] D. Cohen-Steiner, H. Edelsbrunner, and J. Harer. Stability of persistence diagrams. In *Proceedings of the twenty-first annual symposium on Computational geometry*, SCG '05, pages 263–271, New York, NY, USA, 2005. ACM.

[7] H. Derksen and J. Weyman. Quiver representations. *Notices of the AMS*, 52(2):200–206, 2005.

[8] W. Eberly, M. Giesbrecht, and G. Villard. Computing the determinant and smith form of an integer matrix. In *FOCS*, pages 675–685. IEEE Computer Society, 2000.

[9] H. Edelsbrunner and J. Harer. Persistent homology—a survey. In J. E. Goodman, J. Pach, and R. Pollack, editors, *Essays on Discrete and Computational Geometry: Twenty Years Later*, number 453 in Contemporary Mathematics, pages 257–282. American Mathematical Society, 2008.

[10] H. Edelsbrunner and J. Harer. *Computational Topology, An Introduction*. American Mathematical Society, 2010.

[11] H. Edelsbrunner, D. Letscher, and A. Zomorodian. Topological persistence and simplification. *Discrete & Computational Geometry*, 28(4):511–533, 2002.

[12] R. Ghrist. Barcodes: The persistent topology of data. *Bulletin of the American Mathematical Society*, 45:61–75, 2008.

[13] A. Hatcher. *Algebraic Topology*. Cambridge University Press, 2002.

[14] S. Lang. *Algebra*. Graduate Texts in Mathematics. Springer, 2002.

[15] D. Letscher. On persistent homotopy, knotted complexes and the alexander module. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ITCS '12, pages 428–441, New York, NY, USA, 2012. ACM.

[16] N. Milosavljević, D. Morozov, and P. Skraba. Zigzag persistent homology in matrix multiplication time. In *Proceedings of the 27th annual ACM symposium on Computational geometry*, SoCG '11, pages 216–225, 2011.

[17] J. Munkres. *Elements Of Algebraic Topology*. Advanced book program. Perseus Books, 1984.

[18] A. Tausz and G. Carlsson. Applications of zigzag persistence to topological data analysis. *CoRR*, abs/1108.3545, 2011.

[19] L. N. Trefethen and D. Bau. *Numberical Linear Algebra*. SIAM, 1997.

[20] A. Zomorodian. *Topology for Computing*. Cambridge Univ. Press, 2005.

[21] A. Zomorodian and G. Carlsson. Computing persistent homology. *Discrete Comput. Geom.*, 33(2):249–274, Feb. 2005.

Appendix: Full Version of Paper
Persistent Homology Over Directed Acyclic Graphs

Erin W. Chambers [*†]   David Letscher [*]

Abstract

We define persistent homology groups over any set of spaces which have inclusions defined so that the underlying graph between the spaces is directed and acyclic. This method simultaneously generalizes standard persistent homology, zigzag persistence and multidimensional persistence to arbitrary directed acyclic graphs, and it also allows the study of arbitrary families of topological spaces or point-cloud data. We give an algorithm to compute the persistent homology groups simultaneously for all subgraphs which contain a single source and a single sink in $O(n^4)$ time, as well as an algorithm to compute persistence for any arbitrary subgraph in the same running time. We then demonstrate as an application of these tools a method to overlay two distinct filtrations of the same underlying space, which allows us to calculate significant barcodes using considerably fewer points than standard persistence.

1 Introduction

Since its introduction in [11], the concept of topological persistence has become one of the most utilized tools in computational geometry. It has found numerous applications in diverse areas such as surface reconstruction, sensor networks, bioinformatics, and cosmology.

In this paper, we give a generalization of persistence to spaces where the underlying inclusions form a directed acyclic graph. This simultaneously generalizes both zigzag and multidimensional persistence, which can be viewed as special cases of these underlying graphs on the maps between the spaces.

We then give algorithms to compute the persistence homology for specific subgraphs. Our algorithms are analyzed in terms of e and v, which are the number of edges and vertices in the directed acyclic graph G, as well as l, the longest directed path in G. We will in general let $n = v + e$ in our algorithm analysis for simplicity, yielding the $O(n^4)$ time algorithms described in Section 3 and 4.

Potential applications of this are extensive, including any spaces where inclusions are more general than previous settings. We present one such application in Section 5 which deals with barcodes of spaces in such directed acyclic graphs. Our experiment looks at a sampled double torus with 5000 points sampled, as well as a subgraph of two different 200 point samples which include into a 400 points subsample. As expected, the diagram for each smaller sample is quite noisy; however, overlaying the two 200 point samples in our formulation yields a surprisingly good identification of the same features that are obvious in the 5000 point sample.

1.1 Background and related work

For completeness, we briefly survey some results from persistent homology with an emphasis on tools and techniques used in this paper, although a full coverage is beyond the scope of this paper. See any of the recent books or surveys on topological persistence for full coverage of this broad topic and its applications [20, 10, 9, 12].

The original algorithm for computing persistent homology [11] was quickly followed by more work on efficiently computing these groups in several different settings [21] as well as showing desirable properties such as stability under certain assumptions [6] and accurate reconstruction in the presence of noise [5].

The use of barcodes [4] is of particular interest to us, as it relates to our application in Section 5. Through computation of barcodes, one can capture a representation of a topological feature by an interval that represents how long this interval survives in our filtration; at a high level, more important topological features will persist longer in a filtration and will therefore have longer intervals.

Several extensions to topological persistence have appeared, most notably zigzag persistence [1] and multidimensional persistence [3]. In each case, these generalize the simple inclusions from the main algorithm to more general setups.

Zigzag persistence considers spaces with maps of the form $X_1 \leftrightarrow X_2 \leftrightarrow \ldots \leftrightarrow X_n$, where the maps can go in either direction. These maps between the spaces induce maps between chain complexes which pass to homology as homomorphisms $H(X_1) \leftrightarrow H(X_2) \leftrightarrow \ldots \leftrightarrow H(X_n)$; this is known as a zigzag module. Recent work in this setting includes an algorithm which examines the order of the necessary matrix multiplications quite carefully and is able to get a running time for sequence of n simplex deletions or additions which is dominated by the time to multiply two $n \times n$ matrices [16].

Multidimensional persistence extends standard persistence to work not just along a single dimension, but rather on maps between spaces which are parametrized with respect to multiple dimensions [3]. While no analog to the barcode exists in this setting to capture all topological information, the rank invariant is a natural extension of the barcode which captures relevant topological information in many settings.

2 Definition

We recall some relevant definitions and background before presenting our definition of persistent homology over directed acyclic graphs. For a full presentation of homology groups see any introductory text in algebraic topology [13, 17].

For a simplicial complex X and an abelian group A, the k-chains $C_k(X)$ is the space of formal linear combinations of the k-simplices of X where the coefficients are in A. $\partial_k : C_k(X) \to C_{k-1}$ is a linear map that calculates the boundary of a chain. The cycle group $Z_k(X) = \{c \in C_k(X) \mid \partial_k(c) = 0\} = ker(\partial_k)$ and the boundary group $B_k(X) = \{c \in C_k(X) \mid \exists d \in C_{k+1}(X) \text{ with } \partial_{k+1}d = c\} = im(\partial_{k+1})$. The homology group is defined as $H_k(X) = Z_k(X)/B_k(X)$. Note that if A is a field then $C_k(X), Z_k(X), B_k(X)$ and $H_k(X)$ are all vector spaces.

Given a filtration $X_0 \subset X_1 \subset \cdots \subset X_n$, the persistent homology group $H_k^p(X_i)$ can defined in multiple ways. Traditionally, it is defined as $Z_k(X_i)/B_k(X_{i+p}) \cap Z_k(X_{i+p})$ and can be viewed as a quotient group of $H_k(X_{i+p})$. An equivalent definition is $H_k^p(X_i) = im(i_*)$, where $i_* : H_k(X_i) \to H_k(X_{i+p})$ is the map induced by the inclusion $i : X_i \to X_{i+p}$. So $H_k^p(X_i)$ can also be thought of as a subgroup of X_{i+p}.

When viewed from the perspective of zigzag persistence and the use of the zigzag module, the persistence group for some interval can be thought as the subgroups which are common to all of the the homology groups. This motivates the definition of persistence for arbitrary directed graphs. The main restriction on this graph is that it must be acyclic, which is a natural for any graph which represents a set of inclusions.

Definition 2.1. *For a directed acyclic graph $G = (V, E)$, a graph filtration $\mathcal{X}_G$ of a topological space X is a pair $(\{X_v\}_{v \in V}, \{f_e\}_{e \in E})$ such that*

1. *$X_v \subset X$ for all $v \in V$*

2. *If $e = (v_1, v_2) \in E$ then $f_e : X_{v_1} \to X_{v_2}$ is continuous embedding (or inclusion) of X_{v_1} into X_{v_2}.*
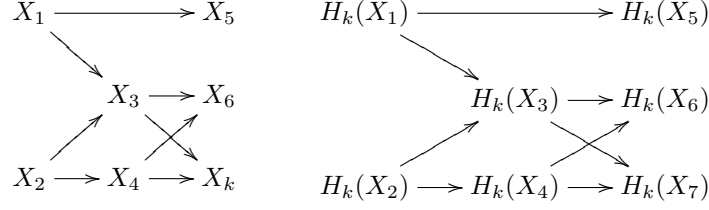
Figure 1: A graph filtration and the induced directed graph of homology groups.
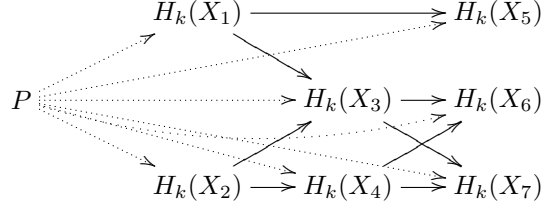

Figure 2: Persistence is defined as the "largest" group that injects into every homology group making the diagram commute.

Given a graph filtration there is a corresponding directed graphs of homology groups, where each edge corresponds to the map induced by inclusion $(f_e)_* : H_k(X_{v_1}) \to H_k(X_{v_2})$, see Figure 1.

We will define persistent homology groups for any connected subgraph of G as the "largest" group that injects into all of the homology groups at the vertices such that the image group is preserved by all of the edges, see figure 2. To make this definition precise, we will will assume that the coefficients for our homology groups are some field $\mathbb{F}$. Throughout this paper, we will assume that $H_k(X)$ means $H_k(X; \mathbb{F})$; this ensures that all of the homology groups are vector spaces over $\mathbb{F}$. In this setting, we can consider one to be "larger" than another if it has larger rank or dimension. See Figure 2 for an example diagram.

Definition 2.2. *Given a graph filtration $\mathcal{X}_G$ and a connected subgraph $G' \subset G$, the G'-persistent homology group, $H_k^{G'}(\mathcal{X}_G)$, is the largest rank vector space $\mathcal{P}$ equipped with linear injections $i_v : \mathcal{P} \to H_k(X_v)$ for all $v \in G'$ such that $f_e \circ i_u = i_v$ for all edges $e = (u, v) \in G'$. In other words, the graph of homology groups along with the inclusions of $\mathcal{P}$ form a commutative diagram.*

Note that $H_k^{G'}(\mathcal{X}_G)$ always exist if the homology groups are finitely generated and are unique up to isomorphism. The maps, however, are not necessarily unique. In Section 2.3, we connect this definition to persistence to known models and prove that it generalizes standard persistence, zigzag persistence and multidimensional persistence.

2.1 Flow graph view

There is also an equivalent and quite interesting view of the structure of the persistent homology group over G', where G' is again a subgraph of the directed acyclic graph G. In the definition of $H_k^{G'}(\mathcal{X}_G)$, we assumed that there are maps i_v for every vertex of G. However, since the diagram commutes, we only need to define these injections from P to the source vertices of G', and we can then use the edge homomorphisms from G' to construct the remainder of the homomorphisms from P to any vertex in G'.

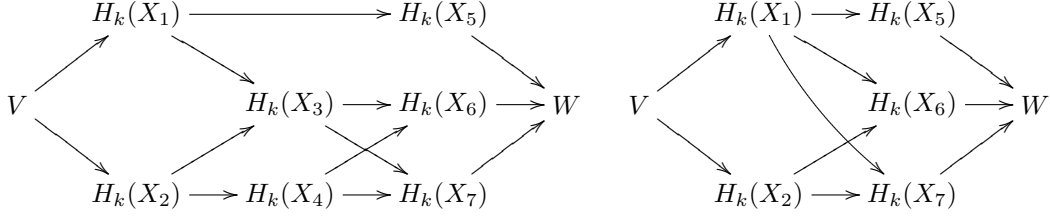We note that in the setting, all of the composite maps will also be injective. Also, since $\mathcal{P}$ injects in

Figure 3: Persistence can also be defined in terms of these flow graphs as the maximal dimension of the image of the composite map $V \to W$.

each of the sink vertices, we can define maps from the homology groups at these sinks to $\mathcal{P}$, such that the composite map $\mathcal{P} \to H_k(X_v) \to \mathcal{P}$ is the identity.

In essence, this means that for any subgraph G', the relevant information can be captured by collapsing G' to a directed bipartite graph, where one independent set consists of the sources in G' and the other is the sinks. The choice of edge homomorphisms from a source to a sink in G' is irrelevant due to the commutativity of the diagram, and therefore we can replace G' with a much simpler conceptual picture. (See Figure 3 for an example.)

This allows an alternate view of this computation of P. Consider collapsing an arbitrary path from each sink to each source of G' into a single edge which is the composition of that path; this results in a bipartite graph H between the sources and sinks of G'. Now in order to compute P in this reduced graph H, we can view this as a flow computation: we place P as a "source" and a "sink" and turn H into a bipartite flow network where the goal is to maximize the rank of P such that the induced homomorphism from each edge in H is a injection and the composite math from P to P through *any* edge is the identity.

2.2 Persistence module

A graph of vector spaces of a fixed directed acyclic graph G form a module structure. For standard persistence this is the same as the persistence module [4] and also coincides with the zigzag module [1].

Definition 2.3. *For a directed acyclic graph $G = (V, E)$, a* commutative G-module *is the pair* $(\{W_v\}_{v \in V}, \{f_e\}_{e \in E}$ *where V_v is a vector space and for any edge $e = (v, w)$, $f_e : W_v \to W_w$ is a linear map with the condition that the resulting diagram is commutative.*

This definition provides the framework for discussing the persistence module for a graph that extends the definition for the zigzag persistence module.

Definition 2.4. *The persistence module for $\mathcal{X}_G$, $\mathcal{PH}_k(\mathcal{X}_G)$, is the commutative G-module formed from the graph of homology groups.*

The theory for G-modules is very similar to the zigzag persistence module. Given a G-module $\mathcal{W} = (\{W_v\}, \{f_e\}$, a G-module $\mathcal{V} = (\{V_v\}, \{g_e\}$ is a *submodule* if $V_v \subset W_v$ for all v and $f_e|_{V_v} = g_e$. Similarly, given two commutative G-modules $\mathcal{V} = (\{V_v\}, \{f_e\})$ and $\mathcal{W} = (\{W_v\}, \{g_e\})$, we can define their connected sum $\mathcal{V} \oplus \mathcal{W}$ as $(\{V_v \oplus W_v\}, \{f_e \oplus g_e\})$. A commutative G-module is said to be *indecomposable* if it cannot be written as a non-trivial connected sum. Any commutative G-module, $\mathcal{V}$, can be written as $\mathcal{V} = \mathcal{V}_1 \oplus \cdots \oplus \mathcal{V}_n$, where each $\mathcal{V}$ is indecomposable. For a connected subgraph G' of G, we will define the commutative G-module $\mathbb{F}_{G'}$ as the module with a copy of $\mathbb{F}$ at each vertex of G'; we will put the identity map on each edge of G' and make every other map trivial. We will call this module *elementary*.

In representation theory, a directed graph is known as a quiver [7] and a representation of that quiver is an assignment of vector spaces to each vertex and a linear map for each edge. The study of quivers
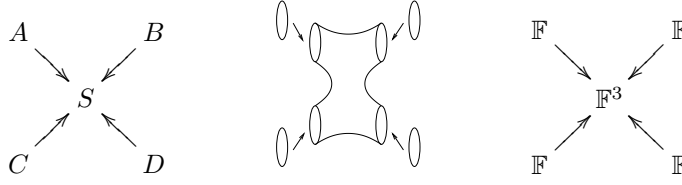
4

Figure 4: (a) The persistence module for the graph G is not finite-type. (b,c) The persistence module for these four curves in a punctured sphere is indecomposable.

provides the underlying theory for the zigzag persistence module. If we add the conditions that the quiver representation must be a commutative diagram, then we get a quiver with relations. Every finite dimensional algebra occurs as a quiver with relations. The following theorem allows us to decompose our persistence modules.

Theorem 2.5 (Krull-Remak-Schmidt Theorem for Finite-Dimensional Algebras [14]). *The decomposition of a commutative G-module is unique up to isomorphism and permutation of the summands.*

In the case of zigzag persistence the relevant indecomposable modules are always are elementary. This is implied by Gabriel's theorem [7] which provides an enumeration of indecomposable modules for particular graph types. Figure 4 gives an example of an indecomposable module that is not elementary. The example consists of a sphere with four punctures, and the inclusion of each of the four boundary components. Unfortunately, the existence of such examples tells us that there is no simple "barcode" representation for DAG persistence. However, the indecomposable summands can be thought of as an "annotated" barcode.

2.3 Relationship with other models of persistence

We saw in Section 2 the definition for standard persistence. Multidimensional persistence can also be defined in terms of the image of a map. Consider a multifiltration, or d-dimensional grid of spaces equipped with a partial ordering on vertices, where $u = (u_1, \ldots, u_d) \leq v = (v_1, \ldots, v_d)$ if and only if $u_i \leq v_i$ for all i.

$$
\begin{array}{ccc}
X_x & \dashrightarrow \cdots \dashrightarrow & X_v \\
\uparrow & & \uparrow \\
\vdots & & \vdots \\
\uparrow & & \uparrow \\
X_u & \dashrightarrow \cdots \dashrightarrow & X_y
\end{array}
$$

The *rank invariant*, $\rho_{X,k}(u, v)$, is defined as the dimension of the image of the induced map $H_k(X_u) \to H_k(X_v)$ [1]. Since the diagram commutes, this map can be found following any path from u to v in the graph. Zigzag persistence is defined in terms of the zigzag module. The definition of a commutative G-module and a τ-module for zigzag persistence are identical for a zigzag graph. The following proposition specifies how DAG persistence generalizes these three notions of persistence.

Proposition 2.6. *Suppose $\mathcal{X}_G$ is a graph filtration of X then*

1. *(Standard persistence) If G is a the graph corresponding to the filtration $X_0 \to X_1 \to \cdots \to X_n$ and $I_{i,p}$ is the subgraph consisting of vertices $\{X_i, \ldots, X_{i+p}\}$ then $H_k^{I_{i,p}}(\mathcal{X}_G) \cong H_k^p(X_i)$. Furthermore, $\mathcal{PH}_k(\mathcal{X}_G)$ coincides with the persistence module.*

5

2. *(Zigzag persistence) If G is the graph for zigzag persistence $\mathbb{X} = X_0 \leftrightarrow X_1 \leftrightarrow \cdots \leftrightarrow X_n$ where each arrow could go in either direction then $H_k(\mathbb{X}) \cong \mathcal{PH}_k(\mathcal{X}_G)$.*

3. *(Multidimensional persistence) If $\mathcal{X} = \{X_v\}_{v \in \{0,\dots,m\}^d}$ is a multifiltration with underlying graph G. If $G_{u,v}$ is the subgraph with vertices $\{w \in G \mid u \leq w \leq v\}$ then the rank invariant $\rho_{X,k}(u,v) = \dim H_k^{G_{u,v}}(\mathcal{X}_G)$.*

Proof. First consider standard persistence and the relevant portion of the filtration of the homology groups: $H_k(X_i) \rightarrow \cdots \rightarrow H_k(X_{i+p})$ and let $i_* : H_k(X_i) \rightarrow H_k(X_{i+p})$ be the composition of the maps. Since $H_k^p(X_i) = im(i_*)$ is a subgroup of $H_k(X_{i+p})$ there is a natural inclusion $g : H_k^p(X_i) \rightarrow H_k(X_{i+p})$. Since i_* surjects on the persistence group there exists an injection. $f : H_k^p(X_i) \rightarrow H_k(X_i)$ such that $i_* \circ f = g$. This shows us that $H_k^{I_{i,p}}(\mathcal{X}_G)$ is at least as large as $H_k^p(X_i)$. And it cannot be larger since any map in $H_k(X_i)$ that realizes persistence must have its image in $im(i_*)$ which is equal to $H_k^p(X_i)$.

The proof for multidimensional persistence is almost identical as the one for standard persistence since the rank invariant is defined as the rank of the image of any path from $H_k(X_u)$ to $H_k(X_v)$. Note that for a graph for zigzag persistence the definition of a commutative G-module coincides with the definition of the zigzag module, so the statement follows immediately. $\qquad\square$

2.4 An example

In Figure 5 we see an example of a set of spaces with inclusions that form a directed acyclic graph. At the top level is a genus two surface; the directed arrows indicate the inclusion maps in our directed acyclic graph, down to our two source vertices in the graph which include one space with two disjoint annuli and one space that is a disk with three boundaries. The graph forms a poset that demonstrates the non-trivial intersections and unions of the three surfaces. In Figure 5, middle, we see the persistence diagram for the entire space, and on the left are diagrams showing the indecomposable summands of the module. Notice that each of these submodules is elementary, yielding a "barcode" like description. From these indecomposables it is possible to read off the persistence for any subgraph G' by counting how many of the elementary modules have G' as a subgraph.

3 Single Source-Single Sink Subgraphs

In this section, we consider $\mathcal{X}_G$ where G is a directed acyclic graph with a single source vertex s. To limit the number of cells which f_e can introduce (where $e = (u,v) \in G$), we assume that each inclusion f_e adds a single cell to the underlying space. As a result, we will also assume that the space X_s will consist of the empty set. This assumption is both standard in most persistence algorithms [21, 16] and quite natural given that we can decompose any inclusion map into a series of inclusions of one simplex at a time.

First, note if we consider a single source-single sink subgraph, it suffices to take any directed path between them in G and compose the maps, since the persistent homology diagram commutes. Therefore, a straightforward application of the $O(n^3)$ standard persistence algorithm would yield a $O(n^5)$ algorithm. Using tools from recent work to compute zigzag persistence in matrix multiple time [16] would give a running time of $O(n^2(M(l) + l^2 \log^2 l))$, where l is the length of the longest path between any source and sink and $M(l)$ is the time to multiply two $l \times l$ matrices.

Theorem 3.1. *If G is a directed acyclic graph whose longest path has length l, then the persistent homology groups with coefficients from a finite field for all single source-single sink subgraphs of G can be calculated in $O(v^2 l^2)$ time.*

We note that if we remove the assumption that coefficients are from a finite field, our theorem still accurately counts the number of arithmetic operations, although running times for each operation might take
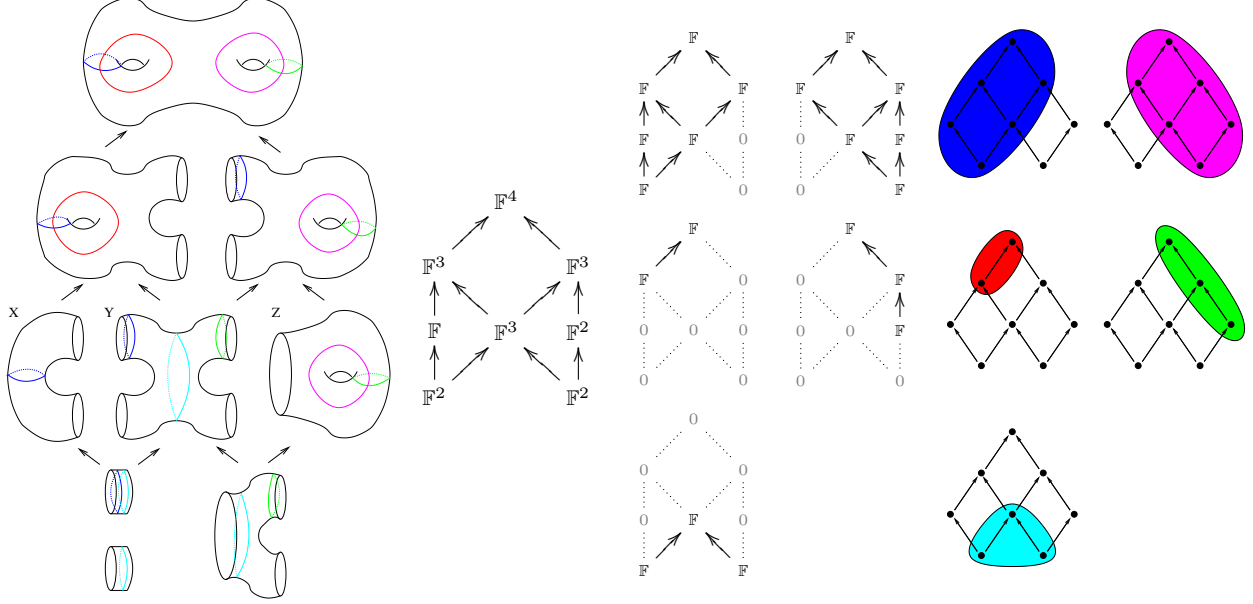
Figure 5: (a) A genus two surface that is the union of three subsurfaces (X, Y and Z), (b) its persistence module $PM_1(\mathcal{X}_G)$, (c) the indecomposable summands of the persistence module . Each colored curve on the left corresponds to a generator of an indecomposable submodule on the right and (d) the "barcodes" for this example.

longer. In 3 dimensions, however, no information is lost when persistent homology is taken with respect to a finite field [15], so this is not an overly restrictive assumption.

This algorithm is actually a relatively straightforward extension of a standard persistence algorithm over finite fields [21]. In this paper they initially represent the homology basis as a matrix, but they are able to eliminate the need for row operations. This results in a simpler data structure, namely an array where each simplex is annotated and marked appropriately (as basis elements are created and destroyed) during the course of the algorithm. The running time is equivalent to Gaussian elimination, or $O(m^3)$ where m is the number of simplices in the filtration.

We will rely on their matrix representation (rather than on their improved data structure) for our algorithm. We first fix a single source vertex s, and compute a (directed) tree T which reaches all possible sinks in G. If our underlying tree is a path, then we can directly apply their algorithm in $O(l^3)$ time. In the more general case where it is not a path, however, we must consider the matrix representation more carefully.

In general, the homology basis at dimension k will be a matrix with each $(k+1)$-simplex represented as a column and each k-simplex represented as a row. While in their algorithm this resulted in a running time of $O(n^2)$ for each operation (due to the Gaussian elimination step), we note that in our algorithm, each such operation is bounded by $O(l^2)$, since the fact that G' only has a path of length l means that our matrix has size at most $l \times l$.

We can also store a representation of the current matrix at each branch point in T, which means we will not need to repeat our calculation starting at the source each time. Since we have at most n edges in T and adding each edge triggers an $O(l^2)$ computation, we take overall $O(nl^2)$ time to compute each path from a single source to all sinks. Repeating this for each source vertex yields the running time in Theorem 3.1. We do note that all of the matrices involved are sparse, as in [21], so we expect the running time in this

7

setting to be faster in practice. We can also adapt this algorithm to multidimensional persistence to give an improvement over the known polynomial time algorithm [2].

Proposition 3.2. *All of the rank invariants for d-dimension lattice with n nodes can be calculated in $O(n^{4-1/d})$ time.*

Proof. The improvement in the run-time for lattices is that we do not need to run the tree based algorithm for every vertex of the graph. Instead we can choose trees with roots at the vertices $X_{(0,v_2,...,v_d)}$ that include all of the paths of the form $X_{(0,v_2,...,v_d)} \to X_{(1,v_2,...,v_d)} \to \cdots \to X_{(m,v_2,...,v_d)}$. Every pair of vertices that can be connected in the lattice by a directed path can also be connected in one of these trees. There are $n^{\frac{d-1}{d}} = n^{1-1/d}$ such trees, yielding the improved running time. $\square$

4 General Subgraphs

There are potentially infinitely many subgraphs of a graph G, so it is unreasonable to expect to be able to calculate persistent homology for all subgraphs in an efficient manner. However, in this section we will given an algorithm calculate $H_k^{G'}(\mathcal{X}_G)$ for fixed G' in $O(n^4)$ time.

Our algorithm is based on the Gram-Schmidt orthonormalization procedure [19]. Gram-Schmidt does not work if $\mathbb{F}$ is a finite field, since we do not have an inner product space in finite fields. Instead, we will work with rational coefficients for our homology groups. As in the last section, we will assume that G has a single source and each edge adds a single simplex in the graph filtration $\mathcal{X}_G$.

At a high level, our algorithm is simply an inductive one that adds each simplex to the subgraph one at a time. At each stage, we will maintain the persistent homology for the current subgraph; however, this requires that we store a matrix representing the boundary, chain, and cycle groups at each vertex so that we can track changes to the persistent homology group across the entire graph as each edge is added. The end result is the following:

Theorem 4.1. $H_k^{G'}(\mathcal{X}_G)$ *can be found using $O(el^3)$ arithmetic operations.*

The running time for this algorithm could actually be much worse since since we are not working in a finite field, so there is the potential of an exponential growth in the numbers involved in these calculations. Ideally, we would work in a finite field to avoid this issue, but as previously noted the Gram-Schmidt process will not work in finite fields.

Let $G_0, \ldots, G_m$ be a sequence of subgraphs of G' where G_0 consists only of the vertices of G', $G_m = G'$ and G_{a+1} is G_a with a single additional edge. We will calculate persistent homology the G_a inductively.

For each vertex i of G', we start with an $s \times s$ matrix M_i, where s is the number of k simplices of X_i. The columns of M_i will be partitioned into three subset: $M_i = (B_i \mid Z_i \mid C_i)$. The rows of M_i correspond to the k-simplices of X_i. Initially, the columns in B_i span the boundaries $B_k(X_i)$ and the the cycle space $Z_k(X_i)$ is spanned by the combination of the columns of Z_i and B_i. The entries of C_i fill out a basis of $\mathbb{Q}^s$. Let b_i, z_i and c_i be the number of columns in each portion of M_i. These entries can be found by calculating the homology groups $H_k(X_i)$ in $O(M(s))$ time [8]. We will abuse notation slightly and use B_i and Z_i to represent both the specified columns of the matrix and the subspace spanned by those columns. Similarly, we use $Z_i + B_i$ to denote the subspace spanned by the union of Z_i and B_i.

The first step of the algorithm will be to use Gram-Schmidt to turn M_i into an orthonormal matrix, which is a matrix with the norm of each column equal to one and the dot product between distinct columns equal to zero. Let $(M_i)_c$ denote the c-th column of M_i. The orthonormalization procedure proceeds as follows:

1. Replace $(M_i)_1$ with $\frac{(M_i)_1}{||(M_i)_1||}$

8

2. For $j = 2..s$, let $v = (M_i)_j - \sum_{l=1}^{j-1} ((M_i)_j \cdot (M_i)_l) \, (M_i)_l$ and replace $(M_i)_j$ with $\frac{v}{||v||}$.

This process does not change the span of the first j columns. So the first b_i columns still span $B_k(X_i)$, and the first $b_i + z_i$ columns span $Z_k(X_i)$.

The invariants we will maintain through this inductive algorithm are:

- For every component C of G_a and vertex $i \in C$, $H_k^C(\mathcal{X}_G) \cong (Z_i + B_i)/B_i$.

- For a component C of G_a, the possible maps $\{g_i : H_k^C(\mathcal{X}_G) \to C_k(X_i)\}$ that realize persistence for the graph C are precisely the ones with $im(g_i) \subset Z_i + B_i$ and $im(g_i) \cap B_i = 0$ and satisfy the commutativity assumptions $f_e \circ g_i = g_j$ for every edge $e = (i, j)$ of C.

- For any edge $e = (i, j)$ of G_a, we have $f_e(Z_i + B_i) = Z_j + B_j$ and $f_e^{-1}(B_j) = B_j$.

- M_i is orthonormal.

As edges are added to the subgraphs, the groups $Z_i + B_i$ will shrink and the group B_i will grow, but these invariants will be maintained.

Consider adding an edge e connecting vertices i and j, and consider injections that realize persistence at both vertices and yield a commutative diagram with the following conditions:

$$
\begin{array}{c}
\begin{array}{ccc}
 & V & \\
{}^{g_i}\swarrow & & \searrow{}^{g_j} \\
C_k(X_i) \xrightarrow{\quad f_e \quad} & & C_k(X_j)
\end{array}
\end{array}
\qquad
\begin{array}{l}
g_i(V) \subset B_i + Z_i \\
g_i(V) \cap Z_i = 0 \\
g_j(V) \subset B_j + Z_j \\
g_j(V) \cap Z_j = 0
\end{array}
$$

To realize persistence for the components of the graph G_a, the above conditions must be met. Any realization of persistence for the larger graph, G_{a+1}, must also satisfy a few additional properties:

1. For the diagram to commute, $g_j(V)$ must be completely contained in the image $f_e(Z_i + B_j)$ which can be guaranteed if we replace $Z_j + B_j$ with $(Z_j + B_j) \cap f_e(Z_i + B_i)$.

2. Also, $g_i(V)$ must be contained in $f_e^{-1}(Z_j + B_j)$. To ensure this we will replace $Z_i + B_i$ with $(Z_i + B_i) \cap f_e^{-1}(Z_j + B_j)$.

3. If $g_i(V)$ intersects $f_e^{-1}(B_j)$ non-trivially then the composition $g_j = f_e \circ g_i$ intersections B_j non-trivially. Thus we must ensure that $g_i(V) \cap f_e^{-1}(B_j) = 0$ so we will replace B_i with $B_i \cup f_e^{-1}(B_j)$.

4. Finally, if $g_i(V) \cap B_i = 0$ then $g_j(V) \cap f_e(B_i) = 0$, so $g_j(V)$ must meet $f_e(B_i)$ trivially. So we will replace B_j with $B_j \cup f_e(B_i)$.

Notice that after these changes $f_e(Z_i + B_i) = Z_j + B_j$ and $f_e^{-1}(B_j) = B_i$. However, this property is no longer guaranteed at all of the other edges. To fix this, we will traverse the graph identifying deficiencies. At any such edge $e = (i, j)$ we will make the following replacements:

1. Replace $Z_j + B_j$ with $(Z_j + B_j) \cap f_e(Z_i + B_i)$.

2. Replace B_i with $B_i \cup f_e^{-1}(B_j)$.

This process must terminate since any time changes are made either the dimension of some $Z_i + B_i$ or Z_i decreases; we bound the exact number of iterations later when discussing the specifics of the algorithm.

Lemma 4.2. *The invariants are satisfied for the graph G_{a+1} if they are met for G_a.*

Proof. Before making changes to the matrices M_i and M_j we have a full characterization of all maps realizing persistence for the components of G_a. The steps we made to change these two matrices yield the overlap in the options. This yields the largest group that injects into both. So, after these changes the new matrices M_i and M_j completely characterize the maps realizing persistence for the components of G_{a+1}. The propagation process ensures that any choice of realization at one vertex can be extended to the adjacent vertex. Thus, after the propagation is complete, the invariants are satisfied at all of the vertices. $\square$

To implement the algorithm, we must perform matrix operations on each M_i. Our basic operations are calculating intersections and the union of subspaces; we also note that unions are the same as doing intersections of perpendicular subspaces.

As a map on chain complexes, f_e is either the identity map (when there is a death of a k-cycle) or its image has codimension one (when there is a birth of a cycle). In the second case, the coordinate corresponding to the new simplex is always zero. Because of this, it is natural to think of chains in $C_k(X_i)$ also as chains in $C_k(X_j)$. Furthermore, chains in $C_k(X_j)$ that are zero in the extra coordinate belong to $C_k(X_i)$. We will denote the coordinate vector for this extra coordinate by u_{ij}. With this perspective, we can summarize our operations as follows:

$Z_j + B_j \Rightarrow (Z_j + B_j) \cap f_e(Z_i + B_i))$	Add u_{ij} and each column of C_i to C_j and re-orthonormalize
$Z_i + B_i \Rightarrow (Z_i + B_i) \cap f_e^{-1}(Z_j + B_j))$	Add each column of C_j projected to $u_{ij}^\perp$ to C_i and re-orthonormalize
$B_i \Rightarrow B_i \cup f_e^{-1}(B_j)$	Add each column of B_j to B_i and re-orthonormalize
$B_j \Rightarrow B_j \cup f_e(B_i)$	Add each column of B_i to B_j and re-orthonormalize

Notice instead of shrinking $Z_i + B_i$, we perform the equivalent operation of enlarging its perpendicular subspace $(Z_i + B_i)^\perp = C_i$. The process of adding a column v to C_i proceeds like a step of Gram-Schmidt. First, project v to a vector, $\bar{v}$, that is perpendicular to the rest of C_i. If $\bar{v}$ is non-zero, it is added to C_i. At this point M_i is no longer square. However, the vectors in B_i and then Z_i can be reduced to be perpendicular to $\bar{v}$. At some point, one of these vectors will be zero and can be removed from M_i. Adding a vector to B_i works in the same manner. Since the matrices involved are at most $l \times l$, this single update can be performed in $O(l^2)$ arithmetic operations.

Each update either increases c_i or increases z_i, and we will avoid adding the same column twice. So, at most $O(l)$ updates can occur for each matrix. Each update that occurs for a matrix triggers potential updates for all of its neighbors. If d_i is the degree of that vertex then it takes $O(d_i l^3)$ operations to do all of the updates to vertex i and perform checks on each neighbor to see if they must be updated. So the total number of arithmetic operations is $\sum O(d_i l^3) = O(e l^3)$, as claimed in Theorem 4.1.

5 Applications

In Figure 6, we show a compelling example of the utility of our algorithm. Each subfigure is a persistence diagram, where each cycle is considered as a pair of birth and death times. The persistence of each cycle is the difference in these times which is equal to the distance to the diagonal. The space considered is simple a double torus sampled with 5000 points, at which level the persistent features, which are the 4 generators of homology, are clearly seen as significant. Note that these four points blur together in pairs in the figure and are difficult to distinguish from each other. In the remaining pictures, we calculate persistent homology for a simple directed acyclic graph that consists of two different 200 point subsamples including into a larger 400 point sample. We note that individually, each sample's persistent homology is quite noisy and does not indicate the 4 generators at all. However, the persistent homology for the directed acyclic graph (shown on the top right) clearly separates the 4 main generators from the noise, at a far lower level of sampling than is possible with standard persistent homology. The table shows the persistence values for the four generating cycles for the surface and the next largest cycle. The values have been renormalized to the length of the cycle with largest persistence in the original sample.

Additional possible applications of this are numerous. For example, we could use directed acyclic graphs of various witness complexes and seek improved results using a DAG over a small set of witness complexes; recent work using zigzag persistence seems relevant in this setting [18]. Also, the example in Section 2.4

| | Full (5000 points) | First half (200 points) | Parallel |
| Subsample (400 points) | Second half (200 points) | Persistence values |

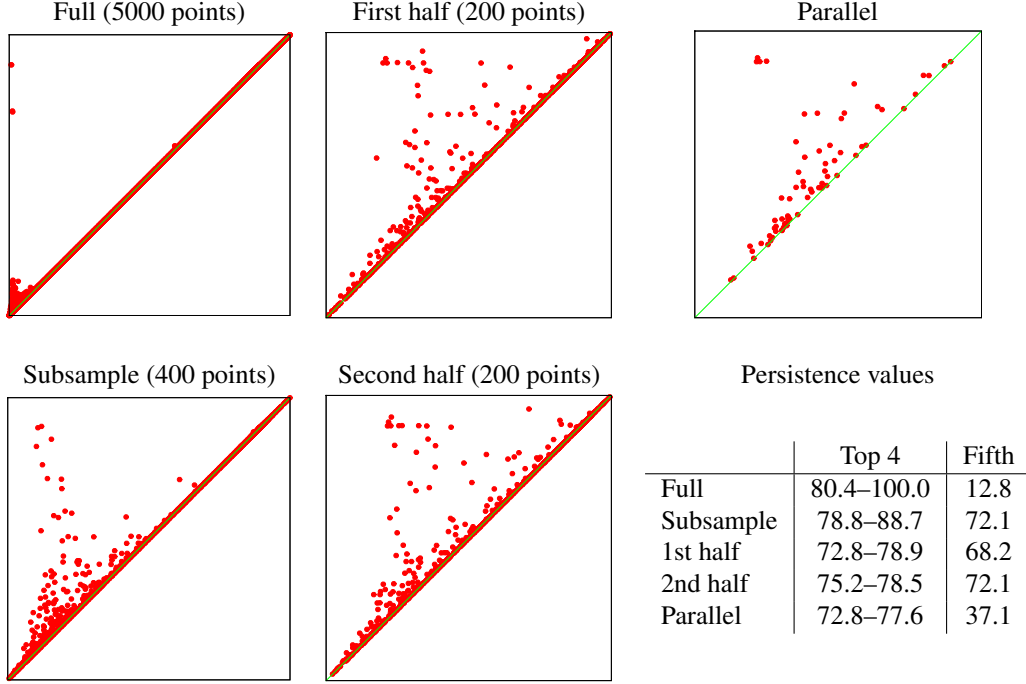|	Top 4	Fifth
Full	80.4–100.0	12.8
Subsample	78.8–88.7	72.1
1st half	72.8–78.9	68.2
2nd half	75.2–78.5	72.1
Parallel	72.8–77.6	37.1

Figure 6: Top left: the persistent homology for a 5000 point sample of a double torus. Bottom left: The persistent homology for a subsample of 400 points. Middle: The persistent homology for two disjoint subsamples of the 400 points. Top right: The homology for the directed acyclic graph that includes each 200 point sample into the 400 point sample.

demonstrates how homology classes from pieces of a space can be "aligned" similar to the bootstrapping method of [18].

6 Future Work

A practical algorithm for decomposition the persistence module into indecompossibles would be useful in finding an annotated barcode representation of the persistence module. Such an algorithm could generalize the right filtrations used in [1, 16].

There is the potential to improve our algorithm for all single source-single sink subgraphs using tools from recent work that computes zigzag homology in matrix multiply time [16]. Consider a directed tree T in the DAG for a single sink s; this tree is composed of paths which are zigzag homology complexes, so we can use the recent zigzag algorithm for each path. If this tree has common subpaths from previous calls to the zigzag algorithm, we could potentially speed up our algorithm by using this information. On the other hand, if the tree has no such common subpaths, then intuitively we should be able to balance the fact that the disjoint paths sum to n (so that we either have many very short paths or the tree is a single path). Balancing this recursion based on their analysis, however, give no better running time than the naive $O(n^2 M(l))$ algorithm we briefly outlined in Section 3, since the time to combine the information from a previous call with the new recursive call will take longer than the call itself. A better algorithm that uses this information successfully is an interesting direction to consider.

Another potential for algorithm improvement follows for our flow characterization in Section 2.1. It is unclear how to adapt standard flow algorithm to the computation of this group P; nonetheless, the picture is

quite compelling, and the literature on maximum flow computations has many algorithms and techniques. It will be interesting to examine this connection more carefully to see if max flow techniques could provide additional insight or improvements.

Acknowledgments

The authors would like to thank Afra Zomorodian for suggesting this problem during a visit, as well as for additional comments and suggestions along the way. We would also like to thank Greg Marks and Michael May for helpful conversions during the course of this work.

References

[1] G. Carlsson and V. de Silva. Zigzag persistence. *Foundations of Computational Mathematics*, 10:367–405, 2010.

[2] G. Carlsson, G. Singh, and A. Zomorodian. Computing multidimensional persistence. *Journal of Computational Geometry*, 1(1), 2010. Preliminary version appeared in ISAAC 2009.

[3] G. Carlsson and A. Zomorodian. The theory of multidimensional persistence. *Discrete Comput. Geom.*, 42(1):71–93, May 2009.

[4] G. Carlsson, A. Zomorodian, A. Collins, and L. Guibas. Persistence barcodes for shapes. *International Journal of Shape Modeling*, 11:149–187, 2005.

[5] F. Chazal and A. Lieutier. Weak feature size and persistent homology: computing homology of solids in rn from noisy data samples. In *Proceedings of the twenty-first annual symposium on Computational geometry*, SCG '05, pages 255–262, New York, NY, USA, 2005. ACM.

[6] D. Cohen-Steiner, H. Edelsbrunner, and J. Harer. Stability of persistence diagrams. In *Proceedings of the twenty-first annual symposium on Computational geometry*, SCG '05, pages 263–271, New York, NY, USA, 2005. ACM.

[7] H. Derksen and J. Weyman. Quiver representations. *Notices of the AMS*, 52(2):200–206, 2005.

[8] W. Eberly, M. Giesbrecht, and G. Villard. Computing the determinant and smith form of an integer matrix. In *FOCS*, pages 675–685. IEEE Computer Society, 2000.

[9] H. Edelsbrunner and J. Harer. Persistent homology—a survey. In J. E. Goodman, J. Pach, and R. Pollack, editors, *Essays on Discrete and Computational Geometry: Twenty Years Later*, number 453 in Contemporary Mathematics, pages 257–282. American Mathematical Society, 2008.

[10] H. Edelsbrunner and J. Harer. *Computational Topology, An Introduction*. American Mathematical Society, 2010.

[11] H. Edelsbrunner, D. Letscher, and A. Zomorodian. Topological persistence and simplification. *Discrete & Computational Geometry*, 28(4):511–533, 2002.

[12] R. Ghrist. Barcodes: The persistent topology of data. *Bulletin of the American Mathematical Society*, 45:61–75, 2008.

[13] A. Hatcher. *Algebraic Topology*. Cambridge University Press, 2002.

[14] S. Lang. *Algebra*. Graduate Texts in Mathematics. Springer, 2002.

[15] D. Letscher. On persistent homotopy, knotted complexes and the alexander module. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ITCS '12, pages 428–441, New York, NY, USA, 2012. ACM.

[16] N. Milosavljević, D. Morozov, and P. Skraba. Zigzag persistent homology in matrix multiplication time. In *Proceedings of the 27th annual ACM symposium on Computational geometry*, SoCG '11, pages 216–225, 2011.

[17] J. Munkres. *Elements Of Algebraic Topology*. Advanced book program. Perseus Books, 1984.

[18] A. Tausz and G. Carlsson. Applications of zigzag persistence to topological data analysis. *CoRR*, abs/1108.3545, 2011.

[19] L. N. Trefethen and D. Bau. *Numberical Linear Algebra*. SIAM, 1997.

[20] A. Zomorodian. *Topology for Computing*. Cambridge Univ. Press, 2005.

[21] A. Zomorodian and G. Carlsson. Computing persistent homology. *Discrete Comput. Geom.*, 33(2):249–274, Feb. 2005.