# Homology Annotations via Matrix Reduction

Technical Report OSU-CISRC-4/12-TR04 [*]

Oleksiy Busaryev, Tamal K. Dey, Yusu Wang
Department of Computer Science and Engineering
The Ohio State University
Columbus, OH 43210, USA
{busaryev,tamaldey,yusu}@cse.ohio-state.edu

## ABSTRACT

In this work, we propose an alternative algorithm for annotating simplices of a simplicial complex $\mathcal{K}$ with sub-bases of a basis $B$ of its $p$-dimensional homology group $\mathsf{H}_1(\mathcal{K})$. Such annotations, summed over $p$-simplices in any $p$-cycle $z$, provide an expression of $z$ in $B$. This allows us to answer queries of null homology and independence of $p$-cycles efficiently and improve the running time of the greedy algorithm to compute a shortest basis of $\mathsf{H}_1(\mathcal{K})$. The best known algorithm for the shortest basis problem that does not use annotations has a time complexity of $O(n^4)$, where $n$ is the size of the 2-skeleton of $\mathcal{K}$. We improve it to $O(n^3 + n^2g^2)$, where $g$ is the rank of $\mathsf{H}_1(\mathcal{K})$.

Annotating simplices with a homology basis has been considered before [1]. The existing approach can preprocess the simplicial complex and assign annotations in sub-cubic time. However, this involves computing the LSP-decomposition of the boundary matrix, which can be computationally cumbersome. We present a simple and implementation-friendly $O(n^3)$ approach that fits nicely to the family of matrix reduction algorithms such as the persistence algorithm and the classic Smith normal form reduction. Our analysis also reveals interesting connections to the persistence algorithm. Namely, our matrix reduction method computes pairing between simplices under homomorphisms between homology groups that are not necessarily induced by inclusions between the subcomplexes of the filtration of $\mathcal{K}$.

## 1. INTRODUCTION

The $p$-dimensional cycles in simplicial complexes, $p$-cycles in short, are topological generalizations of cycles in graphs. These $p$-cycles play a fundamental role in summarizing the topological information about the underlying space that a complex represents. For example, homology groups, which are well known algebraic structures capturing topology of a space, are defined as the sets of equivalence classes on the space of all cycles. Consequently, questions about topological characterizations of input cycles often come up in computations dealing with topology. For example, to compute a shortest basis of a homology group with greedy approach, one needs to determine if cycles in a given set are *independent*. To determine the topological complexity of a given cycle, a first level test could be that if it is *null homologous*, or equivalently if it is a boundary.

A number of studies has been done that concern with the computation of such topological characterization of cycles [2, 3, 5, 4, 11]. However, most of these studies focus on the special case of 1-cycles on surfaces. Recently, a preprocessing technique was proposed [1] that helps answering some of the basic topological queries about cycles efficiently. Given a simplicial complex $\mathcal{K}$, the algorithm annotates simplices in $\mathcal{K}$ in such a way that one may derive topological information about a given $p$-cycle $z$ in $\mathcal{K}$ quickly. For this purpose, a basis $B$ of the homology group in question is determined, and a sub-basis of $B$ is assigned to each $p$-simplex. The sub-bases, called *annotations*, are such that when added for all $p$-simplices in a $p$-cycle $z$, they form the expression or coordinates of $z$ in $B$. The coordinates of $z$ in a basis $B$ reveal its topological class, and thus help answering questions about its topological characterizations.

The algorithm from [1] can compute annotations in $O(n^\omega)$ matrix multiplication time, where $\omega < 2.376$. However, this approach is not particularly suitable for implementation, since computing LSP-decomposition can be inefficient in practice. We propose a simple alternative that involves only simple matrix operations such as column additions. First, we consider a canonical basis of the $p$-chain group of a simplicial complex $\mathcal{K}$ formed by its $p$-simplices. Using simple chain additions and the boundary operator, we change the basis of the $p$-chain group so that a basis of all $p$-cycles can be identified. We change the basis of $p$-cycles further, again with careful chain additions, so that a basis of the $p$-th homology group can be identified. In the process, we also identify the sub-bases that should annotate each $p$-simplex. The entire algorithm can be implemented by column additions in boundary matrices that represent the boundary operator. In this respect, the algorithm falls in the class of well known Smith normal form reduction algorithm that computes ranks of homology groups [12] and the recent persistence algorithm [10] that computes persistent homology.

[*]Pleasee refer to the paper 'Annotating Simplices with a Homology Basis and Its Applications' [1] for additional information on simplex annotations.

However, there are important differences that enable our algorithm to compute annotations which neither of the other two algorithms does. Incidentally, the persistence algorithm can be viewed as a special case of our annotation algorithm. In fact, we obtain a slightly different reduction sequence for the persistence algorithm which enables it to run faster on some special input. Furthermore, our results suggest that simplices can be paired differently while still remaining consistent with a notion of persistent homology which allows homomorphisms between the homology groups that are not necessarily induced by inclusions in a filtration.

We show how the annotations of the simplices can be leveraged to answer some of the topological queries efficiently. In particular, with an $O(n^3)$ preprocessing time for a simplicial complex of $n$ simplices, we achieve the following. A $p$-cycle with $m$ simplices can be checked for null homology in $O(mg)$ time, where $g$ is the rank of $\mathsf{H}_p$. A set of $k$ $p$-cycles with $m$ simplices in total can be verified for independence in $O(m + kg^2)$ time. Lastly, we can improve the time complexity of the greedy algorithm to compute a shortest basis for the one-dimensional homology group of a simplicial complex. The greedy approach, first proposed for surfaces [11] and later extended to simplicial complexes [8], needs independence checks. Using annotations, one can carry out the entire greedy algorithm in $O(n^3 + n^2g^2)$ time, improving the previously known time complexity of $O(n^4)$.

## 2. BACKGROUND AND NOTATIONS

We briefly introduce the notations for chains, cycles, boundaries, and homology groups of a simplicial complex; one may obtain the details from any standard book on algebraic topology such as [12].

Let $\mathcal{K}$ be a simplicial complex. A $p$-chain in $\mathcal{K}$ is a formal sum of $p$-simplices. The additions are assumed to be $\mathbb{Z}_2$-additions in this work. The set of $p$-chains forms an abelian group $\mathsf{C}_p = \mathsf{C}_p(\mathcal{K})$ under $\mathbb{Z}_2$-additions where the empty chain plays the role of identity.

A boundary operator $\partial_p$ acting on a $p$-simplex provides a $(p-1)$-chain that is the sum of the $(p-1)$-faces of $\sigma$. This boundary operator linearly extends to provide a homomorphism $\partial_p : \mathsf{C}_p \to \mathsf{C}_{p-1}$. The kernel $\ker \partial_p$ is called the *$p$-cycle* group of $\mathcal{K}$, denoted $\mathsf{Z}_p = \mathsf{Z}_p(\mathcal{K})$. The image $\operatorname{im} \partial_p$ is called the $(p-1)$-*boundary* group of $\mathcal{K}$, denoted $\mathsf{B}_{p-1} = \mathsf{B}_{p-1}(\mathcal{K})$. By definitions, $\mathsf{Z}_p$ and $\mathsf{B}_{p-1}$ are subgroups of $\mathsf{C}_p$ and $\mathsf{C}_{p-1}$ respectively. Also, since $\partial_{p+1} \circ \partial_p = 0$, we have $\mathsf{B}_p \subseteq \mathsf{Z}_p \subseteq \mathsf{C}_p$. Therefore, the quotient group $\mathsf{H}_p = \mathsf{Z}_p / \mathsf{B}_p$ is well defined, which is called the *$p$-dimensional homology group* of $\mathcal{K}$.

The groups $\mathsf{C}_p$, $\mathsf{Z}_p$, and $\mathsf{B}_p$ are all free abelian groups generated finitely. Thus, they have finite bases. The homology group $\mathsf{H}_p$ under $\mathbb{Z}_2$ additions becomes a finite vector space. Consequently, it also admits a finite basis. An element $v$ in a group with a finite basis $B = \{v_1, \ldots, v_k\}$ has the *coordinate* $\{\alpha_1, \ldots, \alpha_k\}$ where $v = \sum_{i=1}^{k} \alpha_i v_i$. Letting $\langle \cdot, \cdot \rangle$ denote the scalar product associated to the basis $B$, we have $\alpha_i = \langle v, v_i \rangle$ as $B$ becomes orthonormal. The only scalar product we use here is that associated to the *canonical* basis of $\mathsf{C}_p$ formed by its $p$-simplices. Thus, for a $p$-chain $z$ and a $p$-simplex $b$, $\langle z, b \rangle = 1$ if and only if $z$ contains $b$.

## 3. BOUNDARY AND CYCLE BASIS

In this section, we describe a so-called *pivot* operation and how it can be used to produce a certain basis for the boundary and cycle groups. An example is provided in Section 6 to help illustrating our descriptions.

Let $\mathcal{K}$ be a simplicial complex and $p$ be a non-negative integer not more than the dimension of $\mathcal{K}$. Canonical bases of the chain groups $\mathsf{C}_{p-1}$ and $\mathsf{C}_p$ are given by the set of $(p-1)$-simplices $\{a_1, \ldots, a_n\}$ and the set of $p$-simplices $\{b_1, \ldots, b_m\}$ respectively in $\mathcal{K}$. The boundary group $\mathsf{B}_{p-1} \subseteq \mathsf{C}_{p-1}$ is the image $\partial_p(\mathsf{C}_p)$ of the boundary homomorphism $\partial_p : \mathsf{C}_p \to \mathsf{C}_{p-1}$. Let $\{a_{i_1}, \ldots, a_{i_{p+1}}\} \subseteq \{a_1, \ldots, a_n\}$ denote the set of $(p-1)$-simplices in the boundary of a $p$-simplex $b_i$. We drop the subscript $p$ from $\partial_p$ when it is obvious from the context. Clearly, the boundary group $\mathsf{B}_{p-1}$ is generated by

$$\{\partial b_i = a_{i_1} + \cdots + a_{i_{p+1}} \mid i = 1, \ldots, m\}.$$

We *pivot* the boundary cycles $\{d_i = \partial b_i\}$ in any order which updates them as follows (see Section 6 for an example). Suppose we are pivoting $d_i$ now. If $d_i = 0$ because of previous pivoting, no updates occur. Otherwise, we choose any $a_{i_k}$ as a *sentinel* where

$$d_i = a_{i_1} + \cdots + a_{i_k} + \cdots + a_{i_r}$$

and eliminate $a_{i_k}$ from all other chains by the following sequence of *reductions*:

$$\forall j \neq i : d_j \leftarrow d_j + d_i \text{ if } d_j \text{ contains } a_{i_k}. \qquad \text{(R1)}$$

After the application of (R1), $a_{i_k}$ can exist only in $d_i$ and no other $d_j$'s for $j \neq i$. Observe also that after any reduction, a boundary cycle $d_j$ remains a sum of the boundaries of the $p$-simplices that were added to it for reductions (R1). Therefore, at any point of the pivot sequence

$$d_j = \partial b_{j_1} + \cdots + \partial b_{j_k} = \partial(b_{j_1} + \cdots + b_{j_k}),$$

where $\{j_1, \ldots, j_k\} \subseteq \{1, \ldots, m\}$. Let $b'_j = b_{j_1} + \cdots + b_{j_k}$. As the $(p-1)$-chains $\{d_j\}$ change due to reductions, so do the $p$-chains $\{b'_j\}$. The set $\{b'_j\}$ continues to be a basis of $\mathsf{C}_p$ since a reduction replaces a basis element by a linear combination of the element itself with other basis elements. In particular, if $d^*_i$ and $b^*_i$ are the final chains where $d^*_i = \partial b^*_i$ after all pivots are complete, we have a new basis for $\mathsf{C}_p$.

PROPOSITION 1. *$\{b^*_1, \ldots, b^*_m\}$ form a basis of $\mathsf{C}_p$.*

Recall that the pivot operation may render zero some of the boundary cycles in $\{d^*_1, \ldots, d^*_m\}$. Let $U \subseteq \{b^*_1, \ldots, b^*_m\}$ be the maximal subset of the new basis where $\partial b^*_i = 0$ ($U = \{b^*_3, b^*_5\}$ in the example in Section 6). Let $V = \{b^*_1, \ldots, b^*_m\} \setminus U$ be the rest of the basis. Recall that $\langle \cdot, \cdot \rangle$ denotes the scalar product in the canonical base $\{b_1, \ldots, b_m\}$ of $\mathsf{C}_p$.

CLAIM 1. *If $b^*_i \in V$ and $a_{i_k}$ is the sentinel to pivot $d_i$, then $\langle \partial b^*_j, a_{i_k} \rangle = 1$ if and only if $i = j$. Similarly, if $b^*_i \in U$, then $\langle b^*_j, b_i \rangle = 1$ if and only if $i = j$.*

PROOF. Consider $b^*_i \in V$. By definition $d^*_i = \partial b^*_i$ is a nonzero chain. Then, $d_i$ must have been pivoted since the pivot operation skips a chain only if it is zero, and a zero chain remains to be so throughout the pivot sequence. Observe that, once a chain $d_i$ is pivoted, it contains the *unique* sentinel $a_{i_k}$ which does not appear in any other chain $d_j$, $j \neq i$, for the rest of the pivot sequence. Since $d_i = \partial b'_i$, we have the first claim.

Next, consider $b^*_i \in U$. By definition $d^*_i = \partial b^*_i$ is a zero chain. Then, the chain $d_i$ was never pivoted since otherwise it would contain a unique $a_{i_k}$ till the end. The chain $b'_i$ is simply $b_i$ at the beginning of the pivot sequence. Since $d_i$ is not pivoted, $b_i$ is not added to any $b'_j$, $j \neq i$ during the entire pivot sequence. Therefore, no $p$-chain $b'_j$, $j \neq i$, contains $b_i$ at any point of the pivot operation. Hence, $\langle b^*_j, b_i\rangle = 0$ for $j \neq i$. Clearly, the chain $b'_i$ cannot lose $b_i$ due to reductions since no other $b'_j$ contains $b_i$. Thus, $\langle b^*_i, b_i\rangle = 1$, completing the proof of the second claim. $\square$

PROPOSITION 2. *The chains in $\partial V = \{\partial b^*_i \mid b^*_i \in V\}$ form a basis of the boundary group $\mathsf{B}_{p-1}$ whereas the chains in $U$ form a basis of $\mathsf{Z}_p$.*

PROOF. First, since $\{b^*_i \mid i = 1, \ldots, m\}$ is a basis of $\mathsf{C}_p$, $\{\partial b^*_i \mid i = 1, \ldots, m\}$ is a set of cycles that generates $\mathsf{B}_{p-1}$. Among these cycles, the ones in $\partial U$ are zero chains. Therefore, $\mathsf{B}_{p-1}$ is generated by $\partial V$. Furthermore, the chains in $\partial V$ are linearly independent since the chain $\partial b^*_i$ for any $b^*_i \in V$ has a *unique* element $a_{i_k}$ which does not appear in any other $\partial b^*_i$ (Claim 1). Hence $\partial V$, a generator for $\mathsf{B}_{p-1}$, must be its basis.

Now consider $U$, which is a set of $p$-cycles as $\partial b^*_i = 0$ for each $b^*_i \in U$. Cycles in $U$ are linearly independent since each $b^*_i \in U$ has a unique element $b_i$ which does not appear in any $b^*_j$ for $j \neq i$ (Claim 1). Furthermore, it is well-known that $\operatorname{rank} \mathsf{C}_p = \operatorname{rank} \mathsf{B}_{p-1} + \operatorname{rank} \mathsf{Z}_p$. It then follows that $|U| = \operatorname{rank} \mathsf{Z}_p$ since $U \cup V$ forms a basis for $\mathsf{C}_p$ and cardinality of $V$ equals that of $\partial V$ which is $\operatorname{rank}(\mathsf{B}_{p-1})$. Hence $U$ forms a basis of $\mathsf{Z}_p$. $\square$

Let *the index-set $I(U)$ of $U$* denote the set of indices $I(U) = \{i \mid d^*_i \in U\}$. The index-set $I(V)$ is defined similarly. It may be of independent interest to observe that the index-sets of $U$ and $V$ remain independent of the choice of the sentinels. They only depend on the choice of the order by which the chains $\{d_i\}$ are pivoted. Without loss of generality, assume that the chains $\{d_i\}$ have been pivoted in order $\{d_1, \ldots, d_m\}$.

PROPOSITION 3. *The index-sets of $U$ and $V$ remain independent of the choice of the sentinels $\{a_{i_k}\}$.*

PROOF. Assume inductively that the index set of zero chains $U_{i-1} \subset \{d_1, \ldots, d_{i-1}\}$ and hence that of non-zero chains $V_{i-1} = \{d_1, \ldots, d_{i-1}\} \setminus V_{i-1}$ remain independent of the choice of the sentinels after pivoting up to the chain $d_{i-1}$. The assertion is true when $i = 2$ since $d_1$ is not zero no matter which sentinel is chosen for it. To continue induction, consider the chain $d_i$. The fact whether $d_i$ is zero or not does not depend on the choice of the previous sentinels. To see this, consider the complex $\mathcal{K}_i$ made by $p$-simplices $b_1, \ldots, b_i$ and their faces. According to Proposition 2, the cardinality of $U_i$ determines the rank of cycle group $\mathsf{Z}_p(\mathcal{K}_i)$. Since this rank and $I(U_{i-1})$ remain independent of the choice of sentinels, the nullity of $d_i$ and hence inclusion of $i$ into the index set does not depend on them either. We have the claim as $I(U) = I(U_m)$ and $I(V) = I(V_m)$. $\square$

## 4. ANNOTATION

Consider a simplicial complex $\mathcal{K}$ with $(p-1)$-, $p$-, $(p+1)$-simplices $\{a_1, \ldots, a_n\}$, $\{b_1, \ldots, b_m\}$, and $\{c_1, \ldots, c_\ell\}$ respectively. We now describe the procedure to annotate each $p$-simplex in $\mathcal{K}$. We will show in next section that our pivot operation and annotation procedure can be easily implemented using a simple reduction algorithm.

### 4.1 Chains to cycles

Consider the chain group $\mathsf{C}_p$ with the basis $\{b^*_1, \ldots b^*_m\}$ and, after re-indexing if necessary, let $U = \{b^*_1, \ldots b^*_r\}$ and $V = \{b^*_{(r+1)}, \ldots, b^*_m\}$ be the subbases as defined before. In light of Proposition 2, $U$ is a basis of $\mathsf{Z}_p$. We first observe (also see the example in Section 6):

CLAIM 2. *For any $z \in \mathsf{Z}_p$, $z = \sum_{i=1}^r \langle z, b_i\rangle b^*_i$.*

PROOF. Since $U$ is a basis of $\mathsf{Z}_p$, we can represent $z$ uniquely as $z = \sum_{i=1}^r \alpha_i b^*_i$. It then follows from Claim 1 that $\langle z, b_i\rangle = \langle \sum_{j=1}^r \alpha_j b^*_j, b_i\rangle = \sum_{j=1}^r \langle \alpha_j b^*_j, b_i\rangle = \alpha_i$. $\square$

### 4.2 Cycles to homology bases

Our goal is to find a basis for $\mathsf{H}_p = \mathsf{Z}_p/\mathsf{B}_p$, and an annotation of simplices so that the class of any $p$-cycle can be coordinatized easily in this precomputed basis. To achieve this, we will first compute a subbasis of $\mathsf{Z}_p$ that is a basis of the boundary group $\mathsf{B}_p$.

Specifically, consider $(p+1)$-simplices $\{c_1, \ldots, c_\ell\}$ which form a basis of $\mathsf{C}_{p+1}$. The boundary cycles $e_i := \partial c_i = b_{i_1} + \cdots + b_{i_{(p+2)}}$ for $i = 1, \ldots, \ell$ generate the boundary group $\mathsf{B}_p$. We extract a basis out of these boundary cycles similar to what we did for $\mathsf{B}_{p-1}$ using the boundary chains $\{\partial b_i \mid i \in [1, m]\}$, but based on a slightly modified pivot operation as follows. The pivot operation considers a chain $e_i$ if it has not yet been pivoted and it is not a zero chain. The reductions in this pivot sequence are distinguished from the ones in (R1) in that the sentinel for $e_i$, say $b_{i_k}$, must be from $\{b_1, \ldots, b_r\}$ so that $b^*_{i_k} \in U$. We claim in Proposition 4 below that such an element exists in any chain $e_i$ chosen for pivot. After choosing the sentinel $b_{i_k}$ in $e_i$, we reduce other chains as:

$$\forall j \neq i : e_j \leftarrow e_j + e_i \text{ if } e_j \text{ contains } b_{i_k} \qquad \text{(R2)}$$

PROPOSITION 4. *Let $e_i = b_{i_1} + \cdots + b_{i_s}$ be any non-zero chain chosen for pivot. There must exist $b_{i_k} \in \{b_{i_1}, \ldots, b_{i_s}\}$ such that no $e_i$ has been pivoted with $b_{i_k}$ and $b_{i_k} \in \{b_1, \ldots, b_r\}$.*

PROOF. Since $e_i$ is a $p$-cycle, we can write $e_i = \sum_{j=1}^r \alpha_j b^*_j$. Since $e_i$ is non-zero, at least one coordinate, say $\alpha_j$ of $e_i$ is 1. By Claim 2, $\langle e_i, b_j\rangle = \langle e_i, b^*_j\rangle = \alpha_j = 1$, implying that $b_j$ is from $\{b_{i_1}, \ldots, b_{i_s}\}$ and we can set $i_k = j$. Furthermore, this $b_{i_k}$ cannot be a sentinel chosen previously for some chain since then $b_{i_k}$ would have been eliminated from $e_i$. The claim then follows. $\square$

At the end of pivot sequence with reductions in (R2), the boundary cycles $\{e_1, \ldots, e_\ell\}$ are split into two disjoint sets $S$ and $T$ where $S$ is the set of all zero chains and $T$ is the rest. Letting $T$ take the role of $\partial V$ in Proposition 2, one can show that $T$ is a basis of $\mathsf{B}_p$ (see the example in Section 6).

To construct a basis for $\mathsf{H}_p = \mathsf{Z}_p/\mathsf{B}_p$ we set $T$, a basis of $\mathsf{B}_p$, to zero. This means setting $e_i = 0$ for each $e_i \in T$. Consider any $e_i \in T$. By Claim 2, we can write $e_i = \sum_{j=1}^{r} \langle e_i, b_j\rangle b_j^*$. Let $\{i_1, \ldots, i_t\}$ be the set of indices from $[1, r]$ such that $\langle e_i, b_{i_u}\rangle = 1$ for $u \in [1, t]$; that is, $e_i = b_{i_1}^* + \cdots + b_{i_t}^*$ with each $i_u \in [1, r]$. Let $b_j$ be the sentinel we used to pivot $e_i$. It follows from Proposition 4 that $j \in \{i_1, \ldots, i_t\}$ (as $\langle e_i, b_j\rangle = 1$ and $j \in [1, r]$) and assume without loss of generality that $j = i_t$. Setting $e_i = 0$ is equivalent to setting

$$b_{i_t}^* = b_{i_1}^* + \cdots + b_{i_{t-1}}^*. \tag{1}$$

Since the sentinel $b_{i_t}$ only appears in chain $e_i$ after all pivot operations, none of the chains that are on the righthand side of the relations in (1) are on the left. Let $P = \{b_{i_t}^*\}$ be the set of chains that are on the left of equations in (1). By definition of $b_{i_t}^*$, we have $P \subseteq \{b_1^*, \ldots, b_r^*\}$. Let $W = \{b_1^*, \ldots, b_r^*\} \setminus P$. In the example in Section 6, we have $b_3^* = b_5^*$ by setting $e_1 = 0$ and thus $P = \{b_3^*\}$ and $W = \{b_5^*\}$.

The group $\mathsf{H}_p$ is the image of the homomorphism $\eta\colon \mathsf{Z}_p \to \mathsf{H}_p$ where $\eta$ takes a cycle $z \in \mathsf{Z}_p$ to its equivalence class $[z]$. Using relations in (1), it can be defined the basis of $\mathsf{Z}_p$ as:

$$\eta(b_i^*) = \begin{cases} [b_i^*] & \text{if } b_i^* \in W \\ [b_{i_1}^*] + \cdots + [b_{i_{t-1}}^*] & \text{otherwise} \end{cases}$$

Proposition 5. *$[W] = \{[b_i^*] \mid b_i^* \in W\}$ is a basis for $\mathsf{H}_p$.*

## 4.3 Annotation

Finally, we *annotate* the $p$-simplices with $p$-chains that mimic the map $\eta$:

$$\sharp b_i = \begin{cases} b_i & \text{if } b_i^* \in W, \\ b_{i_1} + \cdots + b_{i_{t-1}} & \text{if } b_i^* = b_{i_t}^* \in P, \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

Since $[W]$ is a basis for $\mathsf{H}_p$, its cardinality is $g$. By reindexing if necessary, we assume that $W = \{b_1^*, \ldots, b_g^*\}$.

Theorem 1. *Given a $p$-cycle $z \in \mathsf{Z}_p$ with $[z] = \sum_i^g \alpha_i [b_i^*]$, we have that $\alpha_i = \langle\ \sum_{j=1}^m \langle z, b_j\rangle \sharp b_j, b_i\rangle$.*

Proof. Since $\{b_1, \ldots, b_m\}$ forms a basis for $\mathsf{C}_p$, we can write the $p$-cycle $z$ as $\sum_{j=1}^m \langle z, b_j\rangle b_j$. By Claim 2, $z = \sum_{j=1}^r \langle z, b_j\rangle b_j^*$, where all the simplices with index higher than $r$ can be safely ignored. The homology class of $z$ expressed using basis $[W]$ is $\eta(z) = \sum_{j=1}^r \langle z, b_j\rangle \eta(b_j^*)$. By the way we define the $\sharp$-operator, this is equivalent to

$$\begin{aligned} \eta(z) &= \sum_{j=1}^{r} \left\{ \langle z, b_j\rangle \sum_{i=1}^{g} \langle \sharp b_j, b_i\rangle [b_i^*] \right\} \\ &= \sum_{i=1}^{g} \left\{ \sum_{j=1}^{r} \langle z, b_j\rangle \cdot \langle \sharp b_j, b_i\rangle [b_i^*] \right\} \\ &= \sum_{i=1}^{g} \langle \sum_{j=1}^{m} \langle z, b_j\rangle \sharp b_j, b_i\rangle [b_i^*]. \quad \square \end{aligned}$$

In other words, given any cycle $z$, its homology class $[z]$ can be expressed in the basis $[W]$ as $\Sigma_{i=1}^g \alpha_i [b_i^*]$ where its *coordinate vector* $\theta = \{\alpha_1, \ldots, \alpha_g\}$ is

$$\theta = \{\langle \Sigma_i \langle z, b_i\rangle \sharp b_i, b_1\rangle, \ldots, \langle \Sigma_i \langle z, b_i\rangle \sharp b_i, b_g\rangle\}.$$

Once the annotation $\sharp b_i$ of each simplex $b_i$ is given, the coordinate vector of any $p$-cycle can be computed in $O(ng)$ time where $n$ is the number of $p$-simplices and $g = \operatorname{rank} \mathsf{H}_p$.

Corollary 1. *For a set of cycles $z_1, \ldots, z_k$, we have (i) $z_1 + \cdots + z_k = 0$ iff $\theta_1 + \cdots + \theta_g = 0$ (ii) $[z_1], \ldots, [z_k]$ are independent if and only if the coordinate vectors $\theta_1, \ldots, \theta_k$ are independent.*

## 5. ALGORITHM

To compute the annotations for $p$-simplices, we perform two pivot sequences, one reducing $(p-1)$-chains (R1) and another reducing $p$-chains (R2). Let $\{a_i \mid i = 1, \ldots, n\}$, $\{b_i \mid i = 1, \ldots, m\}$, and $\{c_i \mid i = 1, \ldots, \ell\}$ be arbitrarily ordered sets of $(p-1)$-, $p$-, and $(p+1)$-simplices respectively in a given simplicial complex $\mathcal{K}$. The boundary homomorphism $\partial_p : \mathsf{C}_p \to \mathsf{C}_{p-1}$ is represented by an $n \times m$ boundary matrix, denoted $[\partial_p]$, where $[\partial_p][i, j] = \langle a_i, \partial b_j\rangle$. This means that the $i$-th row and $j$-th column of $[\partial_p]$ represent the $(p-1)$-simplex $a_i$ and the $p$-simplex $b_j$ respectively. Therefore, the reductions of (R1) can be carried out by column additions of $[\partial_p]$. For reductions of (R2), we consider $[\partial_{p+1}]$.

**Algorithm 1** Pivot$(D, m, n)$

```
1:  $J \leftarrow \{1, \ldots, m\}$;
2:  while $J \neq \emptyset$ do
3:    choose any $j \in J$; $J \leftarrow J \setminus j$;
4:    if $\mathrm{col}_D[j] \neq \mathbf{0}$ then
5:      choose any $i$ such that $D[i, j] \neq 0$;
6:      $\mathrm{pivot}[j] \leftarrow i$;
7:      for all $k : D[i, k] \neq 0, k \neq j$ do
8:        $\mathrm{col}_D[k] \leftarrow \mathrm{col}_D[k] + \mathrm{col}_D[j]$
9:      end for
10:   end if
11: end while
12: return $(D, \mathrm{pivot})$;
```

First, we write the algorithm Pivot for pivoting a general $n \times m$ matrix $D$ which we call in the two specific contexts of implementing pivot sequences. Let $\mathrm{col}_D[j]$ and $\mathrm{row}_D[j]$ denote the column $j$ and row $j$ of $D$ respectively. In all pseudo-codes we assume that arrays are initialized appropriately. The algorithm Pivot picks any column $j$ to pivot that has not yet been pivoted (lines 3,4). A sentinel $i$ representing the $(p-1)$-simplex $a_i$ is chosen (line 5) and all columns containing $a_i$ other than column $j$ are reduced (lines 7-8). We record that the chain for column $j$ has been pivoted with $a_i$ by setting the $j$th entry to $i$ in an array pivot (line 6). This pivot array is used later as explained below.

The routine Reduce performs two pivot sequences. First, it reduces $[\partial_p]$ by calling Pivot (line 1) which returns a pivot array pivot1. The columns of $[\partial_p]$ that are reduced to zero represent the set $U$ after the first pivot sequence (R1). The rest of the columns represent the set $V$. This information is implicitly recorded in pivot1. Assuming that all entries of

**Algorithm 2** REDUCE($[\partial_{p+1}], [\partial_p], \ell, m, n$)

```
1: $([\partial_p], \text{pivot1}) \leftarrow$ PIVOT($[\partial_p], m, n$);
2: for all $i \leftarrow 1$ to $m$ do
3:   if $\text{pivot1}[i] \neq 0$ then
4:     $\text{row}_{[\partial_{p+1}]}[i] \leftarrow \mathbf{0}$;
5:   end if
6: end for
7: $([\partial_{p+1}], \text{pivot2}) \leftarrow$ PIVOT($[\partial_{p+1}], \ell, m$);
8: return $([\partial_{p+1}], \text{pivot1}, \text{pivot2})$;
```

pivot1 are initialized to 0, any nonzero entry at location, say $j$, indicates that the $j$th $p$-chain $b_j^*$ is in $V$. The second pivot sequence (R2) is performed on the matrix $[\partial_{p+1}]$. In this sequence, we choose the sentinels whose indices correspond to the elements in $U$. Complementary, the $p$-simplices whose indices correspond to the chains in $V$ cannot be chosen as sentinels. We implement this constraint by zeroing out the row $i$ of $[\partial_{p+1}]$ if the column $i$ in the reduced $[\partial_p]$ is not a zero column, or equivalently, $\text{pivot}[i] \neq 0$ (lines 2-4). After modifying $[\partial_{p+1}]$ as above, we pass it to PIVOT for the second pivot sequence (line 5). Next, we call the routine ANNOTATE which completes the annotation using an array $A$.

**Algorithm 3** ANNOTATE($[\partial_{p+1}], [\partial_p], \ell, m, n$)

```
1: $([\partial_{p+1}], \text{pivot1}, \text{pivot2}) \leftarrow$ REDUCE($[\partial_{p+1}], [\partial_p], \ell, m, n$);
2: for $j \leftarrow 1$ to $\ell$ do
3:   if $(i \leftarrow \text{pivot2}[j]) \neq 0$ then
4:     $A[i] \leftarrow \{k \mid [\partial_{p+1}][k, j] = 1, k \neq i\}$;
5:   end if
6: end for
7: for $i \leftarrow 1$ to $m$ do
8:   if $\text{pivot1}[i] \neq 0$ then
9:     $A[i] \leftarrow 0$;
10:  else
11:    if $A[i] = 0$ then
12:      $A[i] \leftarrow i$;
13:    end if
14:  end if
15: end for
```

The routine ANNOTATE first pivots the columns of $[\partial_p]$ and $[\partial_{p+1}]$ and obtains pivot arrays pivot1 and pivot2 (line 1). At the end of pivoting, the columns of $[\partial_{p+1}]$ that are not zeroed out represent the chains forming the basis $T$ of $\mathsf{B}_p$. Then, if any such column $j$ was pivoted using the row $i$, we annotate $b_i$ with the chain $b_{i_1} + \cdots + b_{i_s}$, where $[\partial_{p+1}][i_1, j]$, $\ldots$, $[\partial_{p+1}][i_s, j]$ are remaining non-zero entries in the column $\text{col}_{[\partial_{p+1}]}[j]$. Specifically, line 3 in the **for** loop (line 2-4), checks if $j$-th column is zero. If not, we know $j$-th column is pivoted with $i$th row (correspond to sentinel $b_i$) since $\text{pivot2}[j] = i$. Therefore, $A[i]$ is set to the list of row indices (representing $b^*_{i_j}s$) having non-zero entries in the column $j$.

A simplex $b_i$ among the rest of the $p$-simplices is annotated with 0 if $i$ is a pivoted column of $[\partial_p]$, i.e., $\text{pivot1}[i] \neq 0$ (line 6-7). Notice that this assignment does not conflict with the previous assignments in line 4 since if column $i$ is not zero in $[\partial_p]$, it is not chosen as a sentinel row in $[\partial_{p+1}]$. The rest of the entries in $A$ which are not set in line 4 and 7, are set to $i$ ($b_i$ itself) following the annotation assignments in (2).

## 5.1 Connection to persistence

It turns out that the matrix based persistence algorithm [9] can be obtained as a special case of our pivoting algorithm. Interestingly, we also obtain a new variant of pairing between simplices.

A *filtration* of a simplicial complex $\mathcal{K}$ is a sequence of nested sub-complexes: $\emptyset = \mathcal{K}_0 \subset \cdots \subset \mathcal{K}_n = \mathcal{K}$. Assume that the difference between any two consecutive sub-complexes is a simplex $\sigma_i = \mathcal{K}_i \setminus \mathcal{K}_{i-1}$, and let $\{\sigma_1, \ldots, \sigma_n\}$ denote the sequence of simplices in $\mathcal{K}$ induced by the filtration. The persistence algorithm inserts the simplices in the filtration order, and computes the changes in homology groups defined under $\mathbb{Z}_2$ coefficients. Specifically, when a $p$-simplex $\sigma$ is brought in, it either creates a $p$-dimensional homology class, or destroys a $(p-1)$-dimensional homology class. The persistence algorithm pairs each destroyer $\sigma$ with a unique creator that creates a homology class destroyed by $\sigma$. This can be implemented by reducing the boundary matrix $[\partial]$ of $\mathcal{K}$ whose columns and rows are sorted according to the filtration order. The reduction considers the columns from left to right while performing column $\mathbb{Z}_2$-additions to the right as follows. Let $\text{low}(j)$ be the row index of the lowest 1 in column $j$. When we consider column $j$, we check if any column $i$ to the left of column $j$ has $\text{low}(i) = \text{low}(j)$. If so, we add column $i$ to column $j$ and thus move $\text{low}(j)$ upward. These column additions continue till either column $j$ becomes a zero column or $\text{low}(j)$ becomes a unique row index so that $\text{low}(j) \neq \text{low}(j')$ for any $j' < j$. The simplex $\sigma_j$ pairs with $\sigma_i$ if $\text{low}(j) = i$. Notice that if $\sigma_j$ is a $p$-simplex, $\sigma_i$ is necessarily a $(p-1)$-simplex.

We claim that same pairing can be obtained by two simple changes in PIVOT. First, we pivot the columns in increasing order of their indices. So, replace lines 2-3 of PIVOT with

**for** $j \leftarrow 1$ to $m$.

Second, instead of choosing *any* sentinel, choose $\text{low}(j)$ as the sentinel row and make the column additions only for columns to the right of $j$, that is, replace line 5 by

choose $i \leftarrow \text{low}(j)$

and line 7 by

**for** $k \leftarrow j + 1$ **to** $m$ **such that** $D[i, k] \neq 0$.

Actually, the change in line 7 is not required though it saves time. First, we assume the change in line 7 and later argue why it can be dropped. The following result from [7] is the key in establishing that the pairing of simplices with the proposed changes in PIVOT is same as that of the standard persistence algorithm.

PROPOSITION 6 ([7]). *Let $[\partial]$ be a boundary matrix of a simplicial complex $\mathcal{K}$ whose rows and columns are sorted according to a filtration of $\mathcal{K}$. If $[\partial]$ is reduced by column additions where the column on right is replaced by the addition result, and each non-zero column contains a unique* $\text{low}(j)$ *after reduction, then* $(\text{low}(j), j)$ *is the same as the pair obtained by the standard persistence algorithm.*

Now, observe that the changes in PIVOT precisely achieve the conditions above. Column $j$ is added to columns on its right. Also, since low($j$) disappears from all such columns because of $\mathbb{Z}_2$-additions, it is never chosen as sentinel later. It follows that low($j$) remains unique for each non-zero column. The algorithm is different from the standard persistence algorithm [7, 9] in that it considers columns on right when reducing a column $j$ rather than the columns on its left. Interestingly, this strategy runs faster in some special cases. Consider a $p$-dimensional simplicial manifold, having each of its $n$ $(p-1)$-simplices incident to at most *two* of its $m$ $p$-simplices. The algorithm computes all $p$- and $(p-1)$-simplex pairings in time $O(mn)$ instead of $O(m^2n)$ that the persistence algorithm would require. Observe that each row in this case has 1 in at most two of the columns which are on the right of the column being reduced. This invariant is true at the beginning and is maintained throughout column reductions. As a result, only one column addition is necessary per column reduction giving us an $O(mn)$ time.

Now, we argue why the change in line 7 can be dropped. After reducing with all proposed changes, the simplices pair as in the standard persistence algorithm. Think of adding a column $j$ to any column on its left that has 1 in the row low($j$). Since low($j$) is unique, these additions cannot disturb low($j$) for any column $j$, i.e. pairing remains the same. However, these additions have the same effect as adding a column $j$ to all columns that have 1 in the row low($j$). This is equivalent to keeping line 7 of Pivot unchanged.

Next, we examine changing only lines 2-3 to choose the columns from left to right for pivot. By Proposition 3, the classification of simplices as creators/destroyers is exactly the same as that obtained by persistence algorithm. However, since we choose sentinels arbitrarily, we have a different pairing between simplices. Each set of such simplex pairings actually captures a persistent homology that is not necessarily defined by inclusions implied by a filtration of the input complex $\mathcal{K}$. To elaborate, consider a filtration and the homomorphisms it induces between the homology groups:

$$\mathsf{H}_p(\mathcal{K}_1) \xrightarrow{\iota_1} \mathsf{H}_p(\mathcal{K}_2) \xrightarrow{\iota_2} \cdots \xrightarrow{\iota_{n-1}} \mathsf{H}_p(\mathcal{K}_n),$$

where $\iota_j$s are induced by inclusions $\mathcal{K}_j \subset \mathcal{K}_{j+1}$. For $j \geq i$, let $\beta_{i,j} = \text{rank}\,(\mathsf{H}_p(\mathcal{K}_i) \xrightarrow{\iota} \mathsf{H}_p(\mathcal{K}_j))$ where $\iota = \iota_i \circ \cdots \circ \iota_j$. Following [6] define the persistence number $\mu_{i,j} = \beta_{i,j} - \beta_{i-1,j} + \beta_{i,j+1} - \beta_{i-1,j+1}$. The persistence algorithm pairs $(\sigma_i, \sigma_j)$ iff $\mu_{i,j} = 1$. If we replace $\iota_i \colon \mathsf{H}_p(\mathcal{K}_i) \to \mathsf{H}_p(\mathcal{K}_{i+1})$ by another homomorphism $h_i \colon \mathsf{H}_p(\mathcal{K}_i) \to \mathsf{H}_p(\mathcal{K}_{i+1})$ such that rank $\iota_i$ = rank $h_i$ then we get different values for $\mu_{i,j}$ which dictates a different pairing. The algorithm Pivot with only lines 2-3 changed, implements different homomorphisms $h_i$s with different choices of sentinels. In particular, when low($\cdot$) is chosen, $h_i$ equates $\iota_i$ and hence we get the standard persistence pairs. In general, it spews out pairs that are consistent with the persistent homology connecting homology groups with homomorphisms $h_i$ which need not be $\iota_i$.

To summarize, the pivoting algorithm we presented bears similarity with the persistence algorithm and the Smith normal form reduction algorithm. However, unlike the other two, our algorithm obtains a meaningful annotation for each simplex and our analysis gives insights into the reduction procedure. By different choice of sentinels, the algorithm outputs a persistence pairing with respect to a different sequence of homomorphisms for the filtered space.

## 6. ANNOTATION EXAMPLE

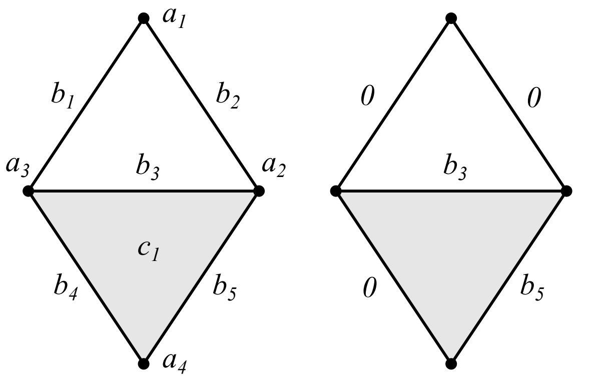

**Figure 1: Left: a simplicial complex. Right: the same complex with annotations assigned.**

Figure 1 illustrates the annotation process for a simplicial complex with a single nontrivial 1-homology class. Below, we elaborate on both reduction sequences. The sentinel simplices are shown in bold.

$$\begin{array}{l} d_1 : \partial(b_1) = \mathbf{a_1} + a_3 \\ d_2 : \partial(b_2) = a_1 + a_2 \\ d_3 : \partial(b_3) = a_2 + a_3 \\ d_4 : \partial(b_4) = a_3 + a_4 \\ d_5 : \partial(b_5) = a_2 + a_4 \end{array} \rightarrow \begin{array}{l} \partial(b_1) = a_1 + a_3 \\ \partial(b_1 + b_2) = \mathbf{a_2} + a_3 \\ \partial(b_3) = a_2 + a_3 \\ \partial(b_4) = a_3 + a_4 \\ \partial(b_5) = a_2 + a_4 \end{array} \rightarrow$$

$$\begin{array}{l} \partial(b_1) = a_1 + a_3 \\ \partial(b_1 + b_2) = a_2 + a_3 \\ \partial(b_1 + b_2 + b_3) = 0 \\ \partial(b_4) = \mathbf{a_3} + a_4 \\ \partial(b_1 + e_2 + e_5) = a_3 + a_4 \end{array} \rightarrow \begin{array}{l} \partial(b_1 + b_4) = a_1 + a_4 \\ \partial(b_1 + b_2 + b_4) = a_2 + a_4 \\ \partial(b_1 + b_2 + b_3) = 0 \\ \partial(b_4) = a_3 + a_4 \\ \partial(b_1 + b_2 + b_4 + b_5) = 0 \end{array}$$

R1 reductions: After all pivots, $b_1^* = b_1 + b_4$, $b_2^* = b_1 + b_2 + b_4$, $b_3^* = b_1 + b_2 + b_3$, $b_4^* = b_4$, and $b_5^* = b_1 + b_2 + b_4 + b_5$. We have $U = \{b_3^*, b_5^*\}$ and $V = \{b_1^*, b_2^*, b_4^*\}$ (note that, we haven't re-indexed which was done for convenience in the text).

$$e_1 : \partial(c_1) = \mathbf{b_3} + b_4 + b_5 \rightarrow \partial(c_1) = b_3 + b_4 + b_5$$

R2 reductions: There is only one chain $e_1$ which generates the boundary group $\mathsf{B}_1$. To pivot $e_1$, either $b_3$ or $b_5$ can be chosen as the sentinel since both $b_3^*$ and $b_5^*$ are in $U$ after R1 reductions. Here $T = \{e_1\}$. Notice that $e_1 = b_3^* + b_5^*$ as suggested by our argument in the text after Proposition 4. We choose $b_3$ as the sentinel and thus $P = \{b_3^*\}$ and $W = \{b_5^*\}$. The annotations in this case assigns the edge $b_5$ the chain $b_5$, the edge $b_3$ the chain $b_5$, and all other edges the chain 0. The class $[b_5^*] = [b_1 + b_2 + b_4 + b_5]$ constitutes a basis for $\mathsf{H}_1$. The expression (coordinates) of the class of any cycle in this basis can be obtained by adding the annotations of its edges. For example, the sum of annotations for the cycle $b_3 + b_4 + b_5$ gives 0 which confirms that it is a trivial cycle in $\mathsf{H}_1$. Similarly, the sum of annotations for cycle $b_1 + b_2 + b_4 + b_5$ gives $b_5$ which should be the case as the expression of its class in the basis $[b_5^*]$ is indeed $[b_5^*]$.

## 7. APPLICATIONS

Our annotation algorithm can be used to address some of the computational problems involving cycles.

## 7.1 Null homology

A $p$-cycle $z$ in a complex $\mathcal{K}$ is called *null homologous* if $[z] = 0$. A cycle is null homologous if and only if it has zero coordinates in any basis of $\mathsf{H}_p(\mathcal{K})$ (Corollary 1). Consider the problem:

QUESTION 1. *Given a $p$-cycle $z$ in a simplicial complex $\mathcal{K}$, decide if $z$ is null homologous.*

This problem can be solved by the persistence algorithm as follows. First, run the persistence algorithm on any filtration of $\mathcal{K}$. Then, think of inserting an open $(p+1)$-cell whose boundary is $z$ into $\mathcal{K}$ treating it as a cell complex. If $z$ is already null homologous in $\mathcal{K}$, the inserted $(p+1)$-cell creates a $(p+1)$-cycle in the cellular homology of $\mathcal{K}$. This can be detected by inserting a column, say $d$, for the dummy $(p+1)$-cell at the right end of the reduced boundary matrix $[\partial_{p+1}]$ of $\mathcal{K}$. We set $[\partial_{p+1}][i, d] \leftarrow \langle z, \sigma_i \rangle$ for each $p$-simplex $\sigma_i \in \mathcal{K}$ and reduce the new column $d$ by column additions as in the persistence algorithm. The cycle $z$ is null homologous if and only if column $d$ is reduced to a zero column. After the initial persistence algorithm which runs in $O(n^3)$ time, this algorithm takes $O(m^2)$ query time where $\mathcal{K}$ contains $n$ simplices and $m$ $p$-simplices. This is because reduction of a column involves $O(m^2)$ additions ($O(m)$ column additions of $O(m)$ $\mathbb{Z}_2$-additions each). Hence, Question 1 can be answered in $O(m^2)$ query time with $O(n^3)$ preprocessing time.

With annotations whose computations take $O(n^3)$ time, we can improve the query time for Question 1 to $O(mg)$ where $g$ is the rank of $\mathsf{H}_p(\mathcal{K})$. For this we simply add the annotations of the $p$-simplices in $z$ and check if the result is zero (Corollary 1). The annotations of the $p$-simplices in $z$ can be added in time linear in the total length of all annotations in $z$ which is $O(mg)$. For adding the annotations, one can simply use an array of length $n$ and index the annotations which are integer values between 1 and $n$ while toggling an entry between 0 and 1 each time it is hit to simulate $\mathbb{Z}_2$-additions. At the end we check if all entries that have been toggled are reduced to 0 or not, which again takes at most $O(mg)$ time.

Given two cycles in $\mathcal{K}$, the question of whether they belong to the same equivalence class motivates the following decision problem:

QUESTION 2. *Given two cycles $p$-cycle $z_1$ and $z_2$ in a simplicial complex $\mathcal{K}$, decide if $z_1$ and $z_2$ are homologous.*

Question 2 reduces to Question 1 because $z_1$ and $z_2$ are homologous if and only if $z_1 + z_2$ is null homologous. Therefore, Question 2 can be answered in $O((m_1 + m_2)g)$ time after $O(n^3)$ time preprocessing where $m_1$ and $m_2$ are the number of $p$-simplices in $z_1$ and $z_2$ respectively.

## 7.2 Independence

An analogous problem to testing null homology is the problem of testing independence. A set of $p$-cycles $z_1, \ldots, z_k$ is called independent if $\Sigma_{i=1}^{k} \alpha_i [z_i] = 0$ implies that $\alpha_i = 0$ for each $i \in [1, k]$. Consider the following problem of independence;

QUESTION 3. *Find a maximally independent subset of a given set of $p$-cycles $z_1, \ldots, z_k$ in a simplicial complex $\mathcal{K}$.*

Since we are considering $\mathbb{Z}_2$-homology, the homology group $\mathsf{H}_p(\mathcal{K})$ is a vector space. As a result, a maximal independent subset of a given set of $p$-cycles can be found with a greedy approach [11]. Assume that $\{z'_1, \ldots, z'_i\} \subseteq \{z_1, \ldots, z_{j-1}\}, j \leq k$, have been determined to be a maximal independent subset of the first $j-1$ cycles. Next, we check if $z_j$ is independent of the chosen subset so far. If so, it is included into the subset. It is easy to show that the new subset is a maximally independent subset of $z_1, \ldots, z_j$.

This greedy approach can be implemented by maintaining a reduced boundary matrix $[\partial_{p+1}]$ as is done by the persistence algorithm when run on a filtration of $\mathcal{K}$. As in the case of null homology testing, we modify (implicitly) $\mathcal{K}$ with the test of each cycle $z_j$. Insert an open 'dummy' $(p+1)$-cell with the boundary $z_j$ into $\mathcal{K}$. The cycle $z_j$ is independent of the cycles $\{z'_1, \ldots, z'_i\}$ if and only if the $(p+1)$-cell of $z_j$ does not make a $(p+1)$-cycle in the modified complex $\mathcal{K}$ that already have the dummy $(p+1)$-cells of the independent cycles $\{z'_1, \ldots, z'_i\}$. We carry out the implicit modification of $\mathcal{K}$ with each test cycle by running the persistence algorithm. First, we run the persistence algorithm on the boundary matrix $[\partial_{p+1}]$ of $\mathcal{K}$. Then, we insert a column $d_j$ for each cycle $z_j \in \{z_1, \ldots, z_k\}$ where $[\partial_{p+1}][i, d_j] \leftarrow \langle z_j, \sigma_i \rangle$ where $\sigma_i$ is a $p$-simplex. We reduce the column $d_j$ till it becomes a zero column or $\mathrm{low}(d_j)$ becomes unique. In the former case, we delete the column $d_j$, and in the latter we add $z_j$ into the independent set.

Each column reduction takes $O(m^2)$ time and thus each test in the above greedy algorithm takes $O(m^2)$ time. Assuming a preprocessing time $O(n^3)$ for building the initial reduction of $[\partial_{p+1}]$, the above algorithm answers Question 3 with $O(km^2)$ query time. We can improve this to $O(tg + kg^2)$ using annotations, where $t$ is the total number of $p$-simplices in the given $k$ cycles and $g$ is the rank of $\mathsf{H}_p(\mathcal{K})$. The preprocessing time for our algorithm remains $O(n^3)$. First, we compute the expression of each $z_i$ in a chosen basis by summing annotations of all $p$-simplices in $z_i$. This takes $O(t_i g)$ time if $z_i$ has $t_i$ $p$-simplices giving a total of $O(tg)$ time over all given cycles. Next, to test if a cycle $z_j$ is independent of the previously chosen independent cycles, we test its annotations of $O(g)$ length against others. This means we carry out the previous column reduction algorithm, but with a matrix $M$ of size $g \times O(g)$. When a cycle $z_j$ is tested, the column $d_j$ of $M$ is set so that $M[i, d_j] \leftarrow 1$ if $z_j$ has 1 for the basis $i$ and 0 otherwise. We delete the column $d_j$ if $z_j$ is tested to be dependent on the previously determined independent cycles. Since there can be at most $g$ independent cycles, the total number of columns of $M$ remains at most $g$ giving an $O(g^2)$ testing time.

### 7.3 Shortest 1-homology basis

Given a simplicial complex $\mathcal{K}$, we assume that each 1-cycle (a loop) is assigned a non-negative weight. We are to find a set of $g = \operatorname{rank} \mathsf{H}_1(\mathcal{K})$ loops with minimal total weight whose homology classes form a basis of $\mathsf{H}_1(\mathcal{K})$.

QUESTION 4. *Compute a set of $g$ independent loops whose total weight is minimal among all such sets.*

Dey, Sun, and Wang [8] gave an $O(n^4)$ time algorithm for this problem, where $n$ is the size of the 2-skeleton of $\mathcal{K}$. The algorithm runs as follows. For every vertex $v \in K$, a shortest path tree $T_v$ rooted at this vertex is computed. For every edge $e \notin T_v$, we form a loop $c_e$ by concatenating $e$ with two shortest paths from $v$ to the endpoints of $e$. Let the set of such loops be denoted $L_v$. The algorithm runs a persistence algorithm to determine a shortest basis $G_v$ out of all loops in $L_v$. Dey et al. show that $\cup_v G_v$ contains a shortest basis of $\mathsf{H}_1(\mathcal{K})$ and thus a greedy algorithm can find such a basis from the loops in $\cup_v G_v$ sorted in non-decreasing sequence of their weights. This greedy approach uses the persistence algorithm to test cycles for independence. Since there are $O(ng)$ loops in $\cup_v G_v$, the greedy algorithm runs in $O(n^3 g)$ time. The complexity of $O(n^4)$ is incurred due to running the persistence algorithm on $\mathcal{K}$ with different filtration for each of $O(n)$ vertices.

Our algorithm eliminates the need for running the persistence algorithm for each vertex. Instead, $O(n^3)$ time is spent to obtain annotations for simplices in $\mathcal{K}$ first and then independence checks are carried out as we described. We collect loop sets $L_v$ over all vertices. We run the greedy algorithm on the set $\cup_v L_v$. Using annotations, the greedy algorithm takes $O(tg + kg^2)$ time. Here $k = O(n^2)$ since each $L_v$ may contain at most $O(n)$ loops for any $v$, thus giving $|\cup_v L_v| = O(n^2)$. Here, $t$, the total number of edges in $\cup_v L_v$ is $O(n^3)$ since each of the $O(n^2)$ loops may contain $O(n)$ edges. Therefore, with this approach we run the greedy algorithm in $O(n^3 g + n^2 g^2)$ time. It turns out that we can improve this time to $O(n^2 g^2)$ as follows.

The loops in $L_v$ are constructed out of the shortest path tree $T_v$. We traverse $T_v$ in a depth first order and each time we reach a new vertex $w$, we augment it with the sum of annotations on the unique path from $v$ to $w$ in $T_v$. This can be computed incrementally. Assume that we have reached $w$ from the vertex $u$ through the edge $uw$. At this point, the annotation $a_u$ for the vertex $u$ is already available. We add the annotation for the edge $uw$ with $a_u$ to obtain $a_w$. Completing the annotations for all vertices takes $O(ng)$ time. The expression for a loop $c_e$ can be computed by adding the annotation of $e$ with those of its endpoints. Therefore, coordinates for each cycle can be obtained in $O(g)$ time after $O(ng)$ preprocessing. Summing over all loops in $L_v$, we get an $O(ng)$ time complexity. Summing over all vertices, we get $O(n^2 g)$ time complexity to complete the expressions for all loops in $\cup_v L_v$. After obtaining the coordinates, the greedy algorithm runs in $O(n^2 g^2)$ time giving an overall $O(n^2 g^2)$ time complexity for the greedy algorithm. Since computing shortest path trees, and all other computations are dominated by the $O(n^3)$ preprocessing time, we have $O(n^3 + n^2 g^2)$ time for the shortest basis computation as claimed.

## 8. CONCLUSIONS

In this work, we present an algorithm to annotate simplices with sub-bases of the homology groups. We have shown its applications to some problems that concern with the topological characterizations of the cycles. Actually, the improvement in running time for the shortest homology basis computation with our approach has been quite phenomenal in practice. We have released the SHORTLOOP software based on this new algorithm. It would be interesting to find other applications of simplex annotations.

The connection to persistence raises some interesting questions. Since our approach allows different pairings among simplices, it associates different persistence diagrams [6] for these pairings. It might be interesting to explore the connections between these various persistence diagrams.

## 9. REFERENCES

[1] O. Busaryev, S. Cabello, C. Chen, T. K. Dey, and Y. Wang. Annotating simplices with a homology basis and its applications. In *Proc. 13th Scandinavian Symp. and Workshop on Algorithm Theory, to appear*, 2012.

[2] S. Cabello, E. Colin de Verdière, and F. Lazarus. Finding shortest non-trivial cycles in directed graphs on surfaces. In *Proceedings of the 2010 Annual Symp. Comp. Geom.*, SoCG '10, pages 156–165, 2010.

[3] S. Cabello, Éric Colin de Verdière, and F. Lazarus. Finding cycles with topological properties in embedded graphs, 2010.

[4] E. W. Chambers, J. Erickson, and A. Nayyeri. Minimum cuts and shortest homologous cycles. In *Proceedings of the 25th Annual Symp. Comp. Geom.*, SoCG '09, pages 377–385, 2009.

[5] C. Chen and D. Freedman. Quantifying homology classes. In *STACS'08*, pages 169–180, 2008.

[6] D. Cohen-Steiner, H. Edelsbrunner, and J. Harer. Stability of persistence diagrams. *Discrete Comput. Geom.*, 37:103–120, January 2007.

[7] D. Cohen-Steiner, H. Edelsbrunner, and D. Morozov. Vines and vineyards by updating persistence in linear time. In *Proceedings of the twenty-second annual symposium on Computational geometry*, SoCG '06, pages 119–126, New York, NY, USA, 2006. ACM.

[8] T. K. Dey, J. Sun, and Y. Wang. Approximating loops in a shortest homology basis from point data. In *Proceedings of the 2010 Annual Symp. Comp. Geom.*, SoCG '10, pages 166–175, 2010.

[9] H. Edelsbrunner and J. L. Harer. *Computational Topology, An Introduction.* American Mathematical Society, Providence, RI, 2010.

[10] H. Edelsbrunner, D. Letscher, and A. Zomorodian. Topological persistence and simplification. In *Proceedings of the 41st Annual Symposium on Foundations of Computer Science*, FOCS '00, pages 454–, Washington, DC, USA, 2000.

[11] J. Erickson and K. Whittlesey. Greedy optimal homotopy and homology generators. In *Proc. 16th Ann. ACM-SIAM Symp. Discrete Algorithms*, pages 1038–1046, 2005.

[12] J. R. Munkres. *Elements of Algebraic Topology.* Addison-Wesley Publishing Company, 1984.