Annotating Simplices with a Homology Basis and Its Applications[*]

Oleksiy Busaryev[†] Sergio Cabello[‡] Chao Chen[§] Tamal K. Dey[¶] Yusu Wang[||]

Abstract

Let $\mathcal{K}$ be a simplicial complex and g the rank of its p-th homology group $\mathsf{H}_p(\mathcal{K})$ defined with $\mathbb{Z}_2$ coefficients. We show that we can compute a basis H of $\mathsf{H}_p(\mathcal{K})$ and annotate each p-simplex of $\mathcal{K}$ with a binary vector of length g with the following property: the annotations, summed over all p-simplices in any p-cycle z, provide the coordinate vector of the homology class $[z]$ in the basis H. The basis and the annotations for all simplices can be computed in $O(n^\omega)$ time, where n is the size of $\mathcal{K}$ and $\omega < 2.376$ is a quantity so that two $n \times n$ matrices can be multiplied in $O(n^\omega)$ time. The pre-computation of annotations permits answering queries about the independence or the triviality of p-cycles efficiently.

Using annotations of edges in 2-complexes, we derive better algorithms for computing optimal basis and optimal homologous cycles in 1-dimensional homology. Specifically, for computing an optimal basis of $\mathsf{H}_1(\mathcal{K})$, we improve the time complexity known for the problem from $O(n^4)$ to $O(n^\omega + n^2 g^{\omega-1})$. Here n denotes the size of the 2-skeleton of $\mathcal{K}$ and g the rank of $\mathsf{H}_1(\mathcal{K})$. Computing an optimal cycle homologous to a given 1-cycle is NP-hard even for surfaces and an algorithm taking $2^{O(g)} n \log n$ time is known for surfaces. We extend this algorithm to work with arbitrary 2-complexes in $O(n^\omega) + 2^{O(g)} n^2 \log n$ time using annotations.

1 Introduction

Cycles play a fundamental role in summarizing the topological information about the underlying space that a simplicial complex represents. Homology groups are well known algebraic structures that capture topology of a space by identifying equivalence classes of cycles. Consequently, questions about homological characterizations of input cycles often come up in computations dealing with topology. For example, to compute a shortest basis of a homology group with a greedy approach, one has to test several times whether the cycles in a given set are *independent*. To determine the topological complexity of a given cycle, a first level test could be deciding if it is *null homologous*, or equivalently if it is a boundary. Recently, a number of studies have been done that concern with the computation of such topological properties of cycles [3, 2, 4, 7, 13, 15, 16, 22].

Two optimization questions about cycles have caught the attention of researchers because of their relevance in applications: (i) compute an optimal homology basis, which asks to compute a set of cycles that form a basis of the corresponding homology group and whose weight is minimum among all such basis;

---

[*] Research was partially supported by the Slovenian Research Agency, program P1-0297 and NSF grant CCF 1064416.

[†] Department of Computer Science and Engineering, The Ohio State University, Columbus, OH 43210, USA. Email: `busaryev@cse.ohio-state.edu`

[‡] Department of Mathematics, University of Ljubljana, Slovenia. Email: `sergio.cabello@fmf.uni-lj.si`

[§] Institute of Science and Technology Austria, Klosterneuburg, Austria. Email: `chao.chen@ist.ac.at`

[¶] Department of Computer Science and Engineering, The Ohio State University, Columbus, OH 43210, USA. Email: `tamaldey@cse.ohio-state.edu`

[||] Department of Computer Science and Engineering, The Ohio State University, Columbus, OH 43210, USA. Email: `yusu@cse.ohio-state.edu`

(ii) compute an optimal homologous cycle, which asks to compute a cycle with minimum weight in the homology class of a given cycle. Chen and Freedman [7] have shown that both problems for p-dimensional cycles, p-cycles in short, with $p > 1$, are NP-hard to approximate within constant factor. Thus, it is not surprising that most of the studies have focused on 1-cycles except for a special case considered in [12]. In this paper, we use *simplex annotations* which lead to better solutions to these problems. We only consider homology over the field $\mathbb{Z}_2$.

Annotation. An *annotation* for a p-simplex is a length g binary vector, where g is the rank of the p-dimensional homology group. These annotations, when summed up for simplices in a given cycle z, provide the coordinate vector of the homology class of z in a pre-determined homology basis. Such coordinates are only of length g and thus help us determine efficiently the topological characterization of z. We provide an algorithm to compute such annotations in $O(n^\omega)$ time, where n is the number of input simplices and ω is a quantity so that two $n \times n$ matrices can be multiplied in $O(n^\omega)$ time. It is known that ω is smaller than 2.376 [9].

The high-level idea for computing the annotation can be described as follows. We first compute an appropriate basis Z of the cycle space $\mathsf{Z}_p(\mathcal{K})$ with the following property: each cycle $z \in Z$ has a *sentinel*, which is a simplex σ_z that appears in the cycle z and in no other cycle from Z. We can then express any cycle z_0 efficiently in the basis Z as the addition of the cycles z from Z whose sentinels σ_z are contained in z_0. Next, we compute an arbitrary homology basis H of $\mathsf{H}_p(\mathcal{K})$. The annotation of any non-sentinel simplex is simply 0, while the annotation of a sentinel σ_z is the coordinates of the homology class $[z]$ in the basis H. Because of linearity, the sum of the annotations over the sentinels in a cycle gives the homology class of that cycle. We show how matrix decomposition algorithms can be leveraged to compute these bases and annotations efficiently. The recent works of Milosavljević et al. [20] and Chen and Kerber [8] also employ fast matrix operations in Computational Topology.

Annotating the simplices of a simplicial complex with elements of an algebraic object has a long tradition in Algebraic Topology. To work with *homotopies* in a simplicial complex one can find a non-Abelian group called fundamental group, described by certain relations, and assign to each 1-simplex an element of the group such that deciding if a path is contractible amounts to testing whether the product of the corresponding group elements is the identity. This line of work dates back to Poincaré, but testing contractibility is equivalent to the word problem for groups, and thus undecidable. When the simplicial complex is a 2-manifold, Dehn showed in 1912 that the approach leads to efficient computation which was further studied in [11]. We refer to Stillwell [23, Chapters 0, 3, 4 and 6] for a comprehensive treatment and historical account of annotations to work with homotopies. Annotations have been used extensively to work with 1-dimensional homology in surfaces, where they can actually be computed in linear time; see for example [3, 15, 16, 22]. However, we are not aware of previous works using annotations to work with homology in higher dimensions or in general simplicial complexes. For readers familiar with cohomology, it may be worth pointing out that cocycles $\{\phi_i\}_{i=1\dots g}$ whose classes generate the cohomology group provide an annotation by assigning the binary vector $(\phi_1(\sigma), \dots, \phi_g(\sigma))$ to simplex σ. From this viewpoint, annotations can be seen as exposing the classical relation between homology and cohomology groups.

Applications. Our annotation technique has the following applications.

1. Using the annotations for edges, we can compute an optimal basis for 1-dimensional homology group $\mathsf{H}_1(\mathcal{K})$ in $O(n^\omega + n^2 g^{\omega-1})$ time, where g is the first Betti number of a simplicial complex $\mathcal{K}$. This improves the previous $O(n^4)$ best known algorithm for computing an optimal homology basis in simplicial complexes [13].

2. Since it is known that computing an optimal homologous cycle is NP-hard even for 1-cycles [5, 6, 4]

in surfaces, Chambers et al. [4] designed an algorithm taking near-linear time when g is constant. Erickson and Nayyeri [15] improved the running time to $2^{O(g)}n\log n$, and Italiano et al. [18] provide an algorithm using $g^{O(g)}n\,\mathrm{loglog}\,n$ time. Using our annotations together with the approach of Erickson and Nayyeri we obtain an algorithm for finding an optimal homologous cycle in simplicial complexes in $O(n^{\omega}) + 2^{O(g)}n^2\log n$ time.

3. Using annotations for p-simplices, we can determine if a given p-cycle is null homologous or if two p-cycles are homologous in time $O(tg)$ time where t is the number of p-simplices in the given p-cycles. Given a set of p-cycles, we can also answer queries about their homology independence. A set of p-cycles is called *homology independent* if they represent a set of linearly independent homology classes. A maximal subset of homology independent cycles from a given set of k cycles with t simplices can be computed in $O(tg + (g+k)g^{\omega-1})$ time after computing the annotations.

In many applications, g, the dimension of the concerned homology group is small and can be taken as a constant. In such cases, the applications listed above benefit considerably, e.g., applications in 1 and 2 run in $O(n^{\omega})$ time.

Structure of the paper. We introduce appropriate background in Section 2, and describe the matrix operations we use in Section 3. In Section 4 we explain how the annotations for edges can be computed to work with 1-cycles. In Section 5 we show results on computing an optimal homology basis and an optimal homologous cycle using edge-annotations. In Section 6 we explain how to extend the annotation algorithm for edges to compute annotations for p-simplices. We next describe some applications of this general annotation result in Section 7. We conclude in Section 8.

2 Background

Homology. In this paper, we focus on simplicial homology over the field $\mathbb{Z}_2$; see comments in the conclusion section for extension to other finite fields. We briefly introduce the notations for chains, cycles, boundaries, and homology groups of a simplicial complex, adapted to $\mathbb{Z}_2$. The details and general case appear in any standard book on algebraic topology such as [21].

Let $\mathcal{K}$ be a simplicial complex. Henceforth, we assume that $\mathcal{K}$ is connected and use $\mathcal{K}_p$ to denote the set of simplices in $\mathcal{K}$ of dimension at most p. To work with the 1-skeleton we use $V = \mathcal{K}_0$, $E = \mathcal{K}_1$, and borrow standard notation from graph theory.

A *p-chain* in $\mathcal{K}$ is a formal sum of p-simplices, $c = \sum_{\sigma\in\mathcal{K}_p} \alpha_\sigma\sigma$, $\alpha_\sigma \in \mathbb{Z}_2$. The set of p-chains forms a vector space $\mathsf{C}_p(\mathcal{K})$ under $\mathbb{Z}_2$-addition where the empty chain plays the role of identity 0. The chain group C_p is in one-to-one correspondence to the family of subsets of $\mathcal{K}_p$. Hence C_p is isomorphic to the n_p-dimensional binary vector space $(\mathbb{Z}_2)^{n_p}$, where n_p is the number of p-simplices in $\mathcal{K}$. A natural basis of C_p consists of the p-simplices in $\mathcal{K}$. In this basis, the coordinate vector of a p-chain is the incidence vector telling which p-simplices appear in the corresponding subset.

The boundary of a p-simplex is a $(p-1)$-chain consisting of the set of its $(p-1)$-faces. This can be linearly extended to a *boundary map* $\partial_p\colon \mathsf{C}_p \to \mathsf{C}_{p-1}$, where the boundary of a chain is defined as the sum of the boundaries of its elements. Using the natural bases of C_p and C_{p-1}, computing the boundary of a p-chain corresponds to multiplying the chain vector with a boundary matrix $[b_1\ b_2 \cdots b_{n_p}]$ whose column vectors are boundaries of p-simplices. We slightly abuse the notation and denote the *boundary matrix* also with ∂_p.

We define the *group of p-cycles* as the kernel of ∂_p, $\mathsf{Z}_p := \ker\partial_p$, and define the *group of p-boundaries* as the image of ∂_{p+1}, $\mathsf{B}_p := \operatorname{im}\partial_{p+1}$. The latter is a subgroup of the former. The *p-th homology group* H_p is the quotient $\mathsf{Z}_p/\mathsf{B}_p$. Each element in H_p, called a *homology class*, is an equivalence class of p-cycles whose differences are p-boundaries. For any p-cycle z, we use $[z]$ to denote the corresponding homology class. Two cycles are *homologous* when they belong to the same homology class. Note that Z_p, B_p, and H_p are also vector spaces. We call their bases a *cycle basis*, a *boundary basis*, and a *homology basis* respectively. The dimension of the p-th homology group is called the *p-th Betti number*. We will denote it by g. A set of p-cycles $\{z_1, \cdots, z_g\}$ is a *homology cycle basis* if the set of classes $\{[z_1], \cdots, [z_g]\}$ forms a homology basis.

Optimization problems. Given a simplicial complex $\mathcal{K}$, exponentially many cycles may belong to a homology class $[z]$. We consider an optimization problem over such a set with all p-cycles assigned well-defined weights. Given a non-negative real weight $w(\sigma)$ for each p-simplex σ, we define the weight of a cycle as the sum of the weights of its simplices, $w(z) = \sum_{\sigma \in z} w(\sigma)$. For example, when $p = 1$ and the weights are the lengths of the edges, the weight of a cycle is its length and the optimization problem seeks for the shortest cycle in a given class. Formally, we state:

Problem 2.1. *Given a simplicial complex and a cycle z, find the cycle* $\operatorname{argmin}_{z_0 \in [z]} w(z_0)$.

Next, we consider an optimization problem over the set of all homology cycle bases. The weight of a homology cycle basis H is defined as the sum of the weights of its elements, $w(H) = \sum_{z \in H} w(z)$. Note that a simplex may contribute to the weight multiple times if it belongs to multiple cycles in the basis H. Formally, we have the following problem.

Problem 2.2. *Given a simplicial complex, find a homology cycle basis H with minimal $w(H)$.*

3 Efficient Matrix Operations

Under $\mathbb{Z}_2$ coefficients, the groups C_p, Z_p, B_p, and H_p are all vector spaces. Linear maps among such spaces or change of bases within the same space can be represented by matrices and operations on them. Our algorithm computes simplex annotations via manipulations of such matrices and bases. Several of our computations use the following concept.

Definition 3.1 (Earliest Basis)**.** *Given a matrix A with rank r, the set of columns $B_{opt} = \{a_{i_1}, \cdots, a_{i_r}\}$ is called the* earliest basis *if the column indices $\{i_1, \cdots, i_r\}$ are the lexicographically smallest index set such that the corresponding columns of A have full rank.*

For convenience, we often use the same symbol to denote both a set of vectors and the matrix they form and denote by B_{opt} also the matrix $[a_{i_1}\ a_{i_2} \cdots a_{i_r}]$. It is convenient to consider the following alternative view of the earliest basis: a column vector of A is in the earliest basis if and only if it does not belong to the subspace generated by column vectors to its left.

We next summarize the operations on matrices that we need. For simplicity, we assume that the matrix multiplication exponent $\omega > 2$; otherwise, some additional logarithmic terms appear in the running times.

Proposition 3.2. *Let A be an $m \times n$ matrix of rank r with entries over $\mathbb{Z}_2$ where $m \leq n$.*

(*a*) *If A is square and has full rank, one can compute its inverse A^{-1} in $O(n^\omega)$ time.*

(*b*) *There is an $O(n^\omega)$ time algorithm to compute the earliest basis B_{opt} of A.*

(*c*) *In $O(n^\omega)$ time, one can compute the coordinates of all columns of A in the earliest basis B_{opt}. Formally, one can compute $AP = B_{opt}[I_r \mid R]$, where P is a permutation matrix, I_r is an $r \times r$ identity matrix, and R is an $r \times (n - r)$ matrix.*

Proof. Item (a) appears in Bunch and Hopcroft [1]; alternatively, see [10]. Item (b) and (c) follows from the *LSP-decomposition* [17, 19], which can be computed in $O(n^\omega)$ time. We restate the following result for a column version, where we transpose both sides of the standard LSP-decomposition and rename the matrices.

LSP-decomposition [19]. *Given an $m \times n$ matrix A, one can compute in $O(n^\omega)$ time a decomposition $A = QSU$, where Q is an $m \times m$ permutation matrix, U is an $n \times n$ upper unitriangular matrix, S is an $m \times n$ matrix with $r = \operatorname{rank}(A)$ non-zero columns which are linearly independent. Furthermore, S is lower unitriangular if we remove all zero columns and the lowest $m - r$ rows.*

Notice that the permutation matrix Q only permutes rows and thus does not affect the computation of the earliest basis of the column rank. By definition, the indices of non-zero columns of S (called the *column rank profile*) are the indices of the earliest basis, and thus can be computed by applying LSP-decomposition once. Therefore item (b) follows.

Next, we prove item (c). Due to item (b), we know the indices of the earliest basis B_{opt} in A and we can compute a column permutation matrix P so that AP has the first r columns as this earliest basis. Next, we compute the LSP-decomposition of the matrix $AP = \widehat{Q}\widehat{S}\widehat{U}$. Since indices of the non-zero columns in $\widehat{S}$ correspond to those of the earliest basis of AP, the last $n - r$ columns of $\widehat{S}$ are necessarily zero. Hence we can rewrite the LSP decomposition as

$$AP \;=\; \widehat{Q}\widehat{S}\widehat{U} \;=\; \widehat{Q}[\widehat{S}_1 \mid 0]\begin{bmatrix} \widehat{U}_{11} & \widehat{U}_{12} \\ 0 & \widehat{U}_{22} \end{bmatrix}$$

where $\widehat{S}_1$ has size $m \times r$, $\widehat{U}_{11}$ has size $r \times r$ and is upper unitriangular, $\widehat{U}_{12}$ has size $r \times (n - r)$, and $\widehat{U}_{22}$ has size $(n - r) \times (n - r)$. Evaluating the right side, we have

$$AP \;=\; \widehat{Q}\widehat{S}_1\widehat{U}_{11}[I_r \mid \widehat{U}_{11}^{-1}\widehat{U}_{12}].$$

Since the first r columns of the matrix on right hand side equal to $\widehat{Q}\widehat{S}_1\widehat{U}_{11}$, and B_{opt} consists of the first r columns of AP by definition of P, we see that $\widehat{Q}\widehat{S}_1\widehat{U}_{11} = B_{opt}$. Setting $R := \widehat{U}_{11}^{-1}\widehat{U}_{12}$ we obtain the desired decomposition $AP = B_{opt}[I_r \mid R]$. The algorithm involves one computation of an earliest basis, one LSP-decomposition, and constant number of matrix inversions and multiplications. Since each of these operations takes $O(n^\omega)$ time, item (c) follows. ☐

4 Annotating Edges

Let $\mathcal{K}$ be a given simplicial complex. First, we define annotations in general terms using g for the dimension of $\mathsf{H}_p(\mathcal{K})$.

Definition 4.1 (Annotations). *An* annotation *for p-simplices is a function $\mathrm{a}\colon \mathcal{K}_p \to (\mathbb{Z}_2)^g$ with the following property: any two p-cycles z and z' belong to the same homology class if and only*

$$\sum_{\sigma\in z} \mathrm{a}(\sigma) \;=\; \sum_{\sigma\in z'} \mathrm{a}(\sigma).$$

Given an annotation a, *the* annotation *of any p-cycle z is defined by* $\mathrm{a}(z) = \sum_{\sigma\in z} \mathrm{a}(\sigma)$.

We will construct annotations using coordinate vectors of cycles in a homology basis. Let $H = (h_1, h_2, \ldots, h_g)$ be a basis of the vector space $\mathsf{H}_p(\mathcal{K})$. For a p-cycle z, if $[z] = \sum_{i=1}^{g} \lambda_i h_i$ where each $\lambda_i \in \mathbb{Z}_2$, then the coordinate vector of $[z]$ in H is $(\lambda_1, \ldots, \lambda_g) \in (\mathbb{Z}_2)^g$. The question is how to annotate the p-simplices so that the sum of annotations in the simplices of z gives $(\lambda_1, \ldots, \lambda_g)$.

In this section, we explain the technique for annotating edges. An extension to p-simplices is explained in Section 6. We compute edge annotations in three steps. First, we construct a cycle basis Z in which any cycle can be expressed in simple and efficient terms. Second, we find a homology cycle basis H. Last, we compute the homology of each cycle in Z in the homology cycle basis H. From this information, one can compute the homology class of any other cycle using vector sums in the coordinate system provided by H. The approach is based on using a spanning tree of the 1-skeleton to generate the space of cycles. The approach of using a spanning tree to generate the fundamental group and then obtain the homology group, is well known in topology and has been used extensively; see for example [14, 16, 23]

Step 1: Computing a cycle basis $\boldsymbol{Z}$. Let us fix throughout this section a spanning tree T in the 1-skeleton of $\mathcal{K}$; it contains $n_0 - 1$ edges. Let $k = n_1 - n_0 + 1$ be the number of edges in $E \setminus E(T)$. We fix an enumeration $e_1, \ldots, e_{n_1}$ of the edges of E with the property that the edges $e_1, \ldots, e_k$ are precisely the edges of $E \setminus E(T)$. Thus, $e_{k+1}, \ldots, e_{n_1}$ are the edges of T. The edges of $E \setminus E(T)$ are called *sentinel edges*, while the edges of $E(T)$ are *non-sentinel edges*.

For any sentinel edge $e \in E \setminus E(T)$, denote by $\gamma(T, e)$ the cycle corresponding to the unique simple path that connects the endpoints of e in T plus the edge e. We call it a *sentinel cycle*. Let Z be the set of such sentinel cycles $\{\gamma(T, e_1), \gamma(T, e_2), \ldots, \gamma(T, e_k)\}$. We have the following property: a sentinel edge e_i belongs to a sentinel cycle $\gamma(T, e_j)$ if and only if $i = j$. For completeness, we set $\gamma(T, e) = 0$ when e belongs to T. The following result is probably folklore.

Proposition 4.2 (Cycle basis). *Z is a cycle basis and for any cycle $z \in \mathsf{Z}_1$ we have $z = \sum_{e \in z} \gamma(T, e)$.*

Proof. Since the edge $e_1 \in E \setminus E(T)$ does not appear in $\gamma(T, e_2), \ldots, \gamma(T, e_k)$, the cycle $\gamma(T, e_1)$ is linearly independent of $\gamma(T, e_2), \ldots, \gamma(T, e_k)$. The same argument applies to any cycle $\gamma(T, e_i) \in Z$, and thus the cycles of Z are linearly independent. Since $k = \dim(\mathsf{Z}_1)$, Z is indeed a basis for Z_1.

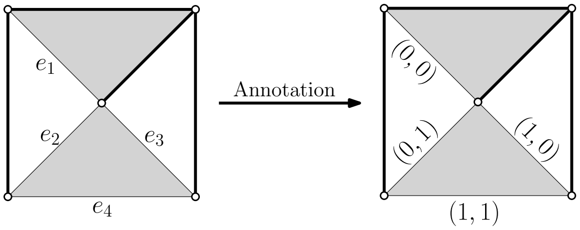
To show the second half of the claim, fix an arbitrary cycle z. Fix a vertex s of the tree T and for any vertex u of T, let $T[s, u]$ denote the unique simple path in T connecting u to s. We then have:

$$z \;=\; \sum_{e = uv \in z} e \;=\; \sum_{e = uv \in z} (T[s, u] + e + T[s, v]) \;=\; \sum_{e \in z} \gamma(T, e).$$

The second equality holds as every vertex in the cycle z is the endpoint of an even number of edges of z. $\square$

Step 2: Computing a homology cycle basis $\boldsymbol{H}$. In this step, we compute a homology cycle basis H from Z with the help of Proposition 3.2(b). Specifically, we construct a new matrix $[\partial_2 \mid Z]$ with the submatrix Z being formed by the chain vectors of cycles in Z. We compute the earliest basis $\widetilde{Z} = [B \mid H]$ of $[\partial_2 \mid Z]$ where B contains the first $r = \operatorname{rank}(\partial_2)$ columns of $\widetilde{Z}$. Since the set of columns of ∂_2 generates the boundary group, by the definition of earliest basis, it is necessary that the columns in B come from ∂_2 and form a boundary cycle basis. Since Z and hence $\partial_2 \cup Z$ generates the cycle group, the remaining columns of $\widetilde{Z}$, namely H, form a homology cycle basis.

Step 3: Computing annotations. Finally, for elements of Z we compute their coordinates in the cycle basis $\widetilde{Z}$. For each sentinel cycle $z = \gamma(T, e)$, we compute its coordinate vector in $\widetilde{Z}$ by solving the linear system $\widetilde{Z}x = z$. The last g entries of x give its coordinates in the basis H. We use this length g vector as the annotation of the sentinel edge e. We can compute annotations for all sentinel edges together by solving $\widetilde{Z}X = Z$ and taking the last g rows of the solution X. For a non-sentinel edge, we simply set its annotation to be the zero vector.



An example of annotation for a 2-complex is shown on right. The edges of a spanning tree are shown with thicker edges. The edges e_1, e_2, e_3, e_4 are sentinel edges. The cycles given by sentinel edges e_2 and e_3 form the homology cycle basis H computed by the algorithm. We show the annotations for the sentinel edges; all other edges get annotation $(0,0)$. The annotation of $(1,1)$ for e_4 makes it possible to evaluate the cycle $e_2e_3e_4$ to $(0,0)$ as it is null-homologous and also evaluate the outer boundary to $(1,1)$ as it is homologous to the sum of the two holes.

Theorem 4.3. *The algorithm described above computes an annotation of length* $\dim(\mathsf{H}_1(\mathcal{K}))$ *for the edges of a* 2*-complex* $\mathcal{K}$ *in* $O(n^\omega)$ *time, where* n *is the size of* $\mathcal{K}$*.*

Proof. From Step 3, the annotation of a sentinel edge e is exactly the coordinate vector of the homology class $[\gamma(T,e)]$. It then follows from Proposition 4.2 that, for any cycle z, the coordinate vector of the homology class $[z]$ is simply the summation of annotations of all edges in z. For the time complexity, notice that Step 1 requires computing a spanning tree and the cycle basis Z, which takes $O(n^2)$ time. Steps 2 and 3 take $O(n^\omega)$ time because of Proposition 3.2(b) and (a) respectively. ☐

5 Optimality for 1-cycles

5.1 Shortest homology basis

In this section we discuss the problem of computing an optimal homology basis for one dimensional homology H_1. The optimal homology cycle basis here is the *shortest homology basis* since we minimize the weights / lengths. We present an efficient algorithm that combines the approach of Erickson and Whittlesey [16] and our annotation technique. The approach restricts the search to a well-structured family of cycles, represents each cycle in this family with a length-g binary vector, and then reduces the computation to the problem of finding an earliest basis in a matrix of size $g \times n^2$.

For each vertex $s \in V$, let T_s be the *shortest path tree* from s with respect to the weight function. Denote by Z_s the set of sentinel cycles corresponding to this tree T_s and Π the union of Z_s for all $s \in V$, that is,

$$\Pi = \bigcup_{s\in V} Z_s = \bigcup_{s\in V} \{\gamma(T_s,e) \mid e \in E \setminus E(T_s)\}.$$

The following property was noted by Erickson and Whittlesey [16]. See also Dey, Sun, and Wang [13] for an extension.

Proposition 5.1. *If we sort the cycles of* Π *in non-decreasing order of their weights, the earliest basis of* Π *is a shortest homology basis.*

Theorem 5.2. *Let* $\mathcal{K}$ *be a simplicial complex of size* n*. We can find a shortest homology basis in time* $O(n^\omega + n^2 g^{\omega-1})$ *where* $g = \mathrm{rank}(\mathsf{H}_1(\mathcal{K}))$*.*

Proof. By Theorem 4.3 we compute annotations for all edges in $O(n^\omega)$ time. Let $\mathrm{a}(e)$ denote such annotation for any edge e, and $\mathrm{a}(z) = \sum_{e\in z} \mathrm{a}(e)$ for any 1-cycle z.

Next we compute annotations for all cycles $z \in \Pi$. Instead of computing them one by one, we annotate all cycles in Z_s at once for each s. Given a fixed s, we first compute T_s in $O(n \log n)$ time. We assign a g-long label $\ell(x)$ to each vertex $x \in V$. The label $\ell(x)$ is the label $\ell(x')$ of its parent x' plus the annotation of the edge xx', $\mathrm{a}(xx')$. We compute labels for all vertices in $O(ng)$ time by a breadth-first traversal of

T_s. Afterward, the annotation of any sentinel cycle $\gamma(T_s, xy) \in Z_s$ is computed in $O(g)$ time as $\ell(x) + \ell(y) + \mathrm{a}(xy)$. Thus, we can compute the annotations for all cycles in Z_s in $O(ng)$ time given T_s and edge annotations. To annotate all cycles of Π, we repeat the procedure for all source vertices s. Computing annotations for all cycles thus takes $O(n^2 g + n^2 \log n)$ time.

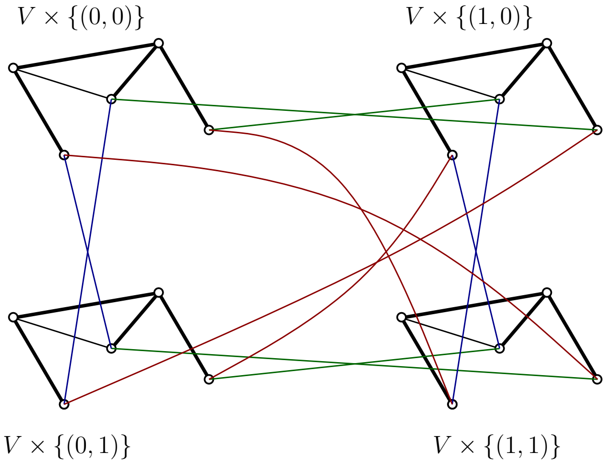
Since annotations of cycles give us the homology classes they belong to, we can use them to find a shortest homology basis. We sort cycles in Π in non-decreasing order of their weights in $O(n^2 \log n)$ time. Let $z_1, z_2, z_3, \ldots$ be the resulting ordering. We construct a matrix A whose ith column is the vector $\mathrm{a}(z_i)$, and compute its earliest basis. By Proposition 5.1, the cycles defining the earliest basis of A form a shortest homology basis. Since there are up to n^2 elements in Π, the matrix A has size $g \times n^2$, and thus it is inefficient to compute its earliest basis using Proposition 3.2 directly. Instead, we use the following iterative method to compute the set J of indices of columns that define the earliest basis.

We partition A from left to right into submatrices $A = [A_1|A_2|\cdots]$ where each submatrix A_i contains g columns with the possible exception of the last submatrix which contains at most g columns. Initially, we set J to be the empty set. We then iterate over the submatrices A_i by increasing index. At each iteration we compute the earliest basis for the matrix $[A_J|A_i]$ where A_J is the submatrix whose column indices are in J. We then set J to be the indices from the resulting earliest basis, increment i, and proceed to the next iteration. At each iteration we need to compute the earliest basis in a matrix with g rows and at most $|J| + g \leq 2g$ columns. There are at most $O(n^2/g)$ iterations each taking $O(g^\omega)$ time.

We obtain the claimed time bound by adding up the time to annotate edges, annotate cycles in Π, and compute the earliest basis. $\square$

5.2 Shortest homologous cycle

In this section, we show how to compute the shortest cycle in a given one-dimensional homology class. In fact, within the same running time, we can compute a shortest cycle in each homology class. The idea is to use covering graphs, and it closely resembles the approach of Erickson and Nayyeri [15]. We skip most of the details because of this similarity. Nevertheless, our main contribution is the use of the annotations from Section 4.



Let $G = (V, E)$ be the 1-skeleton of $\mathcal{K}$. We first compute an annotation $\mathrm{a}\colon E \to (\mathbb{Z}_2)^g$, as given by Theorem 4.3. A walk in G is a sequence of vertices $x_0 x_1 \ldots x_t$ connected by edges in E. It is closed if $x_0 = x_t$. In this section we keep using the term cycle for elements of Z_1. Each closed walk in G defines a cycle, where only edges appearing an odd number of times in the walk are kept. The annotation $\mathrm{a}(w)$ of a walk $w = x_0 x_1 \ldots x_t$ is defined as the sum of the annotations of its edges $x_{i-1}x_i$ for $i = 1, \ldots, t$. Notice that the annotation $\mathrm{a}(w)$ of a closed walk w is the annotation of the cycle defined by w, as annotations in edges appearing an even number of times in the walk cancel out.

We construct a covering graph $\widetilde{G}$ of the 1-skeleton of $\mathcal{K}$, defined as follows:

- $V(\widetilde{G}) = V \times (\mathbb{Z}_2)^g$.
- vertex $(v, h) \in V \times (\mathbb{Z}_2)^g$ is adjacent to $(v', h') \in V \times (\mathbb{Z}_2)^g$ if and only if $e = vv'$ is an edge of E and $h' = h + \mathrm{a}(e)$. The weight of an edge $(v, h)(v', h')$ is the weight of vv'.

The graph $\widetilde{G}$ has $n_0 \cdot 2^g$ vertices and $n_1 \cdot 2^g$ edges. The covering graph for the example shown previously in section 4 for annotation is depicted above. The second coordinate of a vertex $(v, h) \in V(\widetilde{G})$ is used to encode the homology of cycles, as we will see. Similar covering graphs are used, for example, in [3, 15, 22].

Proposition 5.3. *For all $h \in (\mathbb{Z}_2)^g$, we can compute a shortest walk w_h in G among all closed walks with annotation h in $O(2^g n^2(g + \log n))$ time.*

We say that a cycle is *elementary* if it is connected and each vertex is adjacent to at most two edges of the cycle. Each cycle is the union of edge-disjoint elementary cycles. First, we bound the number of elementary cycles in optimal solutions and then use dynamic programming across annotations and the number of elementary cycles to obtain the following

Proposition 5.4. *The shortest cycle in any given homology class consists of at most g elementary cycles.*

Theorem 5.5. *In $O(n^\omega + 4^g g + 2^g n^2(g + \log n)) = O(n^\omega) + 2^{O(g)} n^2 \log n$ time we can compute the shortest homology cycle for all homology classes in H_1.*

6 Annotating p-simplices

In this section, we show how to compute annotations for p-simplices. Notice that the only thing we need to generalize is the first step: find a set Σ of p-simplices (*sentinel simplices*) with cardinality $\dim(\mathsf{Z}_p)$ and a cycle basis $Z = \{z_\sigma\}_{\sigma \in \Sigma}$ (*sentinel cycles*) for the p-cycle group Z_p with the property that z_σ contains $\sigma' \in \Sigma$ if and only if $\sigma = \sigma'$. With this property, any p-cycle z can then be written as $z = \sum_{\sigma \in z \cap \Sigma} z_\sigma$. Taking $z_\sigma = 0$ for all $\sigma \notin \Sigma$, we have $z = \sum_{\sigma \in z} z_\sigma$. With such a basis, we proceed with Step 2 and 3 just like in the case for edges to annotate p-simplices. Below, we explain how to compute such a cycle basis Z.

In the case for annotating edges, we first fix a spanning tree. The boundaries of its edges form a 0-dimensional boundary basis. Any of the remaining edges when added to the tree creates a unique 1-cycle which is kept associated with this edge as a sentinel cycle. For p-simplices, $p > 1$, we do not have a spanning tree, but Proposition 3.2(c) provides us an algebraic tool that serves the same purpose.

Specifically, consider the $n_{p-1} \times n_p$ boundary matrix ∂_p of rank r, where the i-th column in ∂_p corresponds to the $(p-1)$-boundary of p-simplex σ_i. Using Proposition 3.2(c) we can obtain an $n_p \times n_p$ matrix P, an $n_{p-1} \times r$ matrix B_{opt}, and an $r \times (n_p - r)$ matrix R so that

$$\partial_p P = B_{opt}[I_r \mid R].$$

Notice that P permutes the p-simplices so that the first r columns of $\partial_p P$ form the earliest basis B_{opt}. By reordering the columns of ∂_p, we may assume that P is the identity, giving $\partial_p = B_{opt}[I_r \mid R]$. In this scenario, the columns of B_{opt} form a basis of the column-space of ∂_p, and contains the first $r = \operatorname{rank}(\partial_p)$ columns of ∂_p. Note that the i-th column in $[I_r \mid R]$ gives the coordinate vector of the boundary cycle for σ_i in the boundary basis B_{opt}.

Take the first r p-simplices $\{\sigma_1, \ldots, \sigma_r\}$. Their boundaries are linearly independent. Therefore, no subset of them can form a p-cycle. In analogy to Section 4, we use T for this collection of p-simplices and call them non-sentinel simplices. The set $\Sigma = \mathcal{K}_p \setminus T$ of p-simplices are the *sentinel simplices*.

Now consider any sentinel p-simplex, say $\sigma_{r+i} \in \Sigma$. Its boundary is the $(r+i)$-th column in ∂_p and is equal to $B_{opt}R[i]$, where $R[i]$ is the i-th column of R. This means that $\partial_p \sigma_{r+i} = \sum_{j=1}^{r} R[j,i](\partial_p \sigma_j)$ where $R[j,i]$ is the j-th entry in the i-th column of R. Hence taking the set of p-simplices σ_j, $j \in [1, r]$, whose corresponding entries $R[j,i]$ are 1, plus σ_{r+i} itself, we obtain a p-cycle $\gamma(T, \sigma_{r+i})$. We call this p-cycle a *sentinel cycle*. Similar to Section 4, we set $\gamma(T, \sigma) = 0$ for each non-sentinel simplex $\sigma \in T$. Clearly, $\gamma(T, \sigma_{r+i})$ can only contain one simplex from Σ which is σ_{r+i}. We have the desired property: a sentinel simplex $\sigma \in \Sigma$ belongs to a sentinel cycle $\gamma(T, \sigma')$ if and only if $\sigma = \sigma'$. Finally, observe that the columns of $\begin{bmatrix} R \\ I_{n_p - r} \end{bmatrix}$ give the set of sentinel cycles Z. The $(n_p - r) \times (n_p - r)$ identity matrix I_{n_p-r} associates each sentinel cycle $\gamma(T, \sigma)$ in Z to its sentinel p-simplex σ. Similar to Proposition 4.2, we have:

Proposition 6.1. *$Z = \{\gamma(T, \sigma_{r+1}), \ldots, \gamma(T, \sigma_{n_p})\}$ is a cycle basis, and for any p-cycle z we have $z = \sum_{\sigma \in z} \gamma(T, \sigma)$.*

Proof. Z is linearly independent since each cycle contains a unique sentinel simplex. Since Z has $n_p - r = n_p - \mathrm{rank}(\partial_p) = \dim(\mathsf{Z}_p)$ elements, Z forms a basis for Z_p. An arbitrary p-cycle z has a unique coordinate in the cycle basis Z. Since each sentinel simplex $\sigma \in \Sigma$ belongs to one and only one cycle in Z, the corresponding coordinate is one for a cycle $\gamma(T, \sigma)$ if and only if $\sigma \in z$. $\square$

Combining this proposition with Step 2 and 3 from Section 4, we obtain the following theorem.

Theorem 6.2. *We can annotate the p-simplices in a simplicial complex with n simplices in $O(n^\omega)$ time.*

7 Null Homology and Independence

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

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

> Q1: Given a p-cycle z in a simplicial complex $\mathcal{K}$, decide if z is null homologous.

A p-cycle z is null homologous if and only if it is linearly dependent to columns of ∂_{p+1}. This could be determined by checking whether z belongs to the earliest basis of the matrix $[\partial_{p+1} \mid z]$. The complexity of such computation is $O(n^\omega)$ (Proposition 3.2(b)).

However, with annotations whose computations take $O(n^\omega)$ time, we can improve the query time for Q1 to $O(tg)$ where $g = \dim \mathsf{H}_p(\mathcal{K})$ and t is the number of p-simplices in z. For this we simply add the annotations of the p-simplices in z and check if the result is zero, which takes $O(tg)$ time. Now consider the following decision problem:

> Q2: Given two p-cycles z_1 and z_2 in a simplicial complex $\mathcal{K}$, decide if z_1 and z_2 are homologous.

The problem Q2 reduces to Q1 because z_1 and z_2 are homologous if and only if $z_1 + z_2$ is null homologous. Therefore, Q2 can be answered in $O((t_1 + t_2)g)$ time after $O(n^\omega)$ time preprocessing where t_1 and t_2 are the number of p-simplices in z_1 and z_2 respectively.

Independence. An analogous problem to testing null homology is the problem of testing independence.

> Q3: Find a maximally independent subset of a given set of p-dimensional homology classes $[z_1], \ldots, [z_k]$ in a simplicial complex $\mathcal{K}$.

Without our annotation technique, for each such query, we could apply Proposition 3.2(b) to the $n \times (n+k)$ matrix $[\partial_{p+1} \mid z_1 \; z_2 \cdots z_k]$. The set of cycles z_is belonging to the earliest basis would represent a linearly independent set of classes. The overall complexity is $O((n+k)^\omega)$. Using the iterative technique delineated in Theorem 5.2, we can improve the query time to $O(n^\omega(n+k)/n) = O(n^\omega + kn^{\omega-1})$. However, with our annotation technique, we could improve the query time.

Compute the annotations for all edges in $O(n^\omega)$ time. By Theorem 6.2, we can then compute an annotation $\mathrm{a}(z_i)$ in $O(tg)$ time for all cycles z_is, where t is the number of simplices altogether in all cycles and each $\mathrm{a}(z_i)$ is a length-g vector. Now construct a matrix A whose ith column is the vector $\mathrm{a}(z_i)$. Notice that the earliest basis of A form a maximally independent subset of column vectors in A. Thus the set of cycles z_is corresponding to columns in this earliest basis form a maximally independent subset of the input set of p-cycles. Since A is of size $g \times k$, we have an $O(g^\omega + k^\omega)$ query time, after an $O(n^\omega)$ preprocessing time. We can improve the query time to $O((g+k)g^{\omega-1})$ by using the iterative technique in Theorem 5.2. Therefore, total time for computing a maximally independent set takes $O(tg + (g+k)g^{\omega-1})$ time after annotations.

8 Conclusions

In this paper, we present an algorithm to annotate p-simplices in a complex so that computations about the homology groups can be done faster. We have shown its applications to some problems that concern with the optimality of 1-cycles and topological characterizations of the p-cycles. The algorithm uses operations such as matrix inversion and matrix multiplication, and thus can take advantage of the best known algorithms for these operations, which take $o(n^3)$ time.

For defining the weights of a cycle we used 1-norm to combine the weights of the individual edges. For defining the weight of a basis we also used 1-norm to combine the weights of the basis cycles. In these problems we can use any other norm to define these weights.

One may wonder why we cannot extend our annotation approach to compute the optimal homology basis or the optimal homologous cycle for higher dimensional cycles. The main bottleneck for finding an optimal basis is that the Proposition 5.1 does not generalize to higher dimensional cycles. Given that the problems are NP-hard in high dimensions even for $g = 1$ [7], such extensions cannot exist unless P=NP.

Instead of computing the shortest homology basis, one may want to compute a set of edges with minimal total weight which supports a homology cycle basis. In such case, the algorithm in Theorem 5.2 cannot be used. The annotations and the covering graph might help.

Finally, we point out that one can use any finite field instead of Z_2 for annotations. Since annotations mainly utilize matrix multiplications which remain valid under any field, the annotation algorithm in section 6 remains applicable without any change. However, the optimal cycles in sections 5 and 5.2 require computations of cycles associated with shortest paths which we do not know how to generalize for general fields. Specifically, it is not clear how to obtain results for applying Propositions 5.1 and 5.3. This could be a topic of further research.

References

[1] J. Bunch and J. Hopcroft. Triangular factorization and inversion by fast matrix multiplication. *Math. Comp.*, 28:231–236, 1974.

[2] S. Cabello, É. Colin de Verdière, and F. Lazarus. Finding shortest non-trivial cycles in directed graphs on surfaces. In *Proc. ACM Symp. on Computational Geometry (SOCG)*, pages 156–165, 2010.

[3] S. Cabello, É. Colin de Verdière, and F. Lazarus. Finding cycles with topological properties in embedded graphs. *SIAM J. Disc. Math.*, 25(4):1600–1614, 2011.

[4] E. Chambers, J. Erickson, and A. Nayyeri. Minimum cuts and shortest homologous cycles. In *Proc. ACM Symp. on Computational Geometry (SOCG)*, pages 377–385, 2009.

[5] C. Chen and D. Freedman. Quantifying homology classes II: Localization and stability. *CoRR*, abs/0709.2512, 2007. Available at `http://arxiv.org/abs/0709.2512`.

[6] C. Chen and D. Freedman. Quantifying homology classes. In *Proc. Symp. on Theoretical Aspects of Computer Science (STACS)*, pages 169–180, 2008.

[7] C. Chen and D. Freedman. Hardness results for homology localization. *Discrete and Computational Geometry*, 45(3):425–448, 2011.

[8] C. Chen and M. Kerber. An output-sensitive algorithm for persistent homology. In *Proc. ACM Symp. on Computational Geometry (SOCG)*, pages 207–216, 2011.

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

[10] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, Cambridge, MA, second edition, 2001.

[11] T. K. Dey and S. Guha. Transforming curves on surfaces. *Journal of Computer and System Sciences*, 58:297–325, 1999.

[12] T. K. Dey, A. Hirani, and B. Krishnamoorthy. Optimal homologous cycles, total unimodularity, and linear programming. In *Proc. ACM Symp. on Theory of Computing (STOC)*, pages 221–230, 2010.

[13] T. K. Dey, J. Sun, and Y. Wang. Approximating loops in a shortest homology basis from point data. In *Proc. ACM Symp. on Computational Geometry (SOCG)*, pages 166–175, 2010.

[14] D. Eppstein. Dynamic generators of topologically embedded graphs. In *Proc. ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 599–608, 2003.

[15] J. Erickson and A. Nayyeri. Minimum cuts and shortest non-separating cycles via homology covers. In *Proc. ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 1166–1176, 2011.

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

[17] O. Ibarra, S. Moran, and R. Hui. A generalization of the fast LUP matrix decomposition algorithm and applications. *Journal of Algorithms*, 3(1):45–56, 1982.

[18] G. F. Italiano, Y. Nussbaum, P. Sankowski, and C. Wulff-Nilsen. Improved algorithms for min cut and max flow in undirected planar graphs. In *Proc. ACM Symp. on Theory of Computing (STOC)*, pages 313–322, 2011.

[19] C. Jeannerod. LSP matrix decomposition revisited, 2006. Available at `http://www.ens-lyon.fr/LIP/Pub/Rapports/RR/RR2006/RR2006-28.pdf`.

[20] N. Milosavljević, D. Morozov, and P. Škraba. Zigzag persistent homology in matrix multiplication time. In *Proc. ACM Symp. on Computational Geometry (SOCG)*, pages 216–225, 2011.

[21] J. R. Munkres. *Elements of Algebraic Topology*. Addision-Wesley Publishing Company, Menlo Park, 1984.

[22] V. Patel. Determining edge expansion and other connectivity measures of graphs of bounded genus. In *ESA 2010*, volume 6346 of *Lecture Notes in Computer Science*, pages 561–572. Springer, 2010.

[23] J. Stillwell. *Classical topology and combinatorial group theory*. Springer, 2nd edition, 1993.

A Omitted proofs

Proof of Proposition 5.3. We first argue that there is a bijection between closed walks in G through a vertex v with annotation h and walks in $\widetilde{G}$ from vertex $(v,0)$ to vertex (v,h). Indeed, assume first that there is a closed walk $w = x_0x_1\cdots x_t$ in G with $x_t = x_0 = v$. By the definition of annotation we have

$$\mathrm{a}(w) \;=\; \sum_{i=1,\dots,t} \mathrm{a}(x_{i-1}x_i).$$

Define $y_0 = (x_0, 0)$ and define, for each vertex x_i in w, the vertex $y_i = (x_i, \sum_{j\le i} \mathrm{a}(x_{j-1}x_j))$. By construction, there is an edge in $\widetilde{G}$ between y_i and y_{i+1}, and therefore $y_0y_1\cdots y_t$ is a path in $\widetilde{G}$ from $(v,0)$ to $(v,\mathrm{a}(w))$. Conversely, for any walk in $\widetilde{G}$ from vertex $(v,0)$ to vertex (v,h), the projection into the first coordinate provides a closed walk through v whose annotation is h.

For all $v \in V$, we compute the shortest paths from $(v,0)$ to all vertices in $\widetilde{G}$, and record the lengths. This is $|V|$ computations of shortest path trees in $\widetilde{G}$ and thus takes $O(n(2^g n\log(2^g n))=O(2^g n^2(g+\log n))$ time (Erickson and Nayyeri [15] can speed up this step because in their case $\widetilde{G}$ is embedded in a surface.) The closed walk w_h is then obtained by considering the shortest path in $\widetilde{G}$ between $(v,0)$ and (v,h), over all vertices $v \in V$, and then taking its projection onto G. □

Proof of Proposition 5.4. We can assume that the homology class is nonzero, as otherwise the result is trivial. Assume for contradiction that the shortest cycle z in a given homology class contains more than g elementary cycles $z_1,\cdots,z_t$, where $t > g$. Each elementary cycle is a cycle by itself. Since the homology classes of these cycles are not linearly independent, there exists a set of indices $I \subset \{1,\dots,t\}$ such that $0 = \sum_{i\in I}[z_i]$. Notice that $I \ne \{1,\dots,t\}$ as otherwise $[z] = \sum_{i\in\{1,\dots,t\}}[z_i] = 0$. The cycle $z' = \sum_{i\in\{1,\dots,t\}\setminus I} z_i$ is strictly shorter than z, and represents the same homology class as z. This contradicts the assumption that z is shortest in its class. □

Proof of Theorem 5.5. We compute annotations for the edges in $O(n^\omega)$ time using Theorem 4.3. For each $h \in (\mathbb{Z}_2)^g$ we compute the closed walk w_h given in Proposition 5.3. This takes $O(2^g n^2(g+\log n))$ time.

For any $h \in (\mathbb{Z}_2)^g$ and any integer $k \in [1,g]$, we define $C(h,k)$ as follows

$$C(h,k) = \begin{cases} \text{length of } w_h & \text{if } k = 1;\\ \min\{C(h_1,k-1) + C(h_2,1) \mid h = h_1 + h_2\} & \text{if } k > 1. \end{cases}$$

It is straightforward to see by induction that $C(h,k)$ is an upper bound on the length of the shortest cycle with annotation h. Most interestingly, $C(h,g)$ is the length of the shortest cycle with annotation h. Indeed, if the shortest cycle with annotation h consists of the elementary cycles $z_1,\dots,z_t$ with $t \le g$, then it follows by induction that, for any $I \subset \{1,\dots t\}$, $C\left(\sum_{i\in I}\mathrm{a}(z_i), |I|\right)$ is the length of $\left(\sum_{i\in I} z_i\right)$.

A standard dynamic programming algorithm to compute $C(h,k)$ takes $O(2^g)$ time per element, for a total of $O(4^g g)$ time. We then return for each $h \in (\mathbb{Z}_2)^g$ the cycle defining $C(h,g)$. (A shortest cycle homologous to a given cycle z is obtained as the cycle defining $C(\mathrm{a}(z),g)$.) □