Fast Algorithms for Minimum Homology Basis

Amritendu Dhar[1*†], Vijay Natarajan[1] and Abhishek Rathod[2†]

[1*]Department of Computer Science and Automation, Indian Institute of Science, Bangalore, 560012, Karnataka, India.
[2]Department of Computer Science, Ben Gurion University, Be'er Sheva, 8410501, Israel.

*Corresponding author(s). E-mail(s): amritendud@iisc.ac.in;
Contributing authors: vijayn@iisc.ac.in; arathod@post.bgu.ac.il;
†These authors contributed equally to this work.

Abstract

We study the problem of finding a minimum homology basis, that is, a lightest set of cycles that generates the $\mathbf{1}$-dimensional homology classes with $\mathbb{Z}_{\mathbf{2}}$ coefficients in a given simplicial complex $\boldsymbol{K}$. This problem has been extensively studied in the last few years. For general complexes, the current best deterministic algorithm, by Dey et al. [13], runs in $\boldsymbol{O(Nm^{\omega-1}+nmg)}$ time, where $\boldsymbol{N}$ denotes the total number of simplices in $\boldsymbol{K}$, $\boldsymbol{m}$ denotes the number of edges in $\boldsymbol{K}$, $\boldsymbol{n}$ denotes the number of vertices in $\boldsymbol{K}$, $\boldsymbol{g}$ denotes the rank of the $\mathbf{1}$-homology group of $\boldsymbol{K}$, and $\boldsymbol{\omega}$ denotes the exponent of matrix multiplication. In this paper, we present three conceptually simple randomized algorithms that compute a minimum homology basis of a general simplicial complex $\boldsymbol{K}$. The first algorithm runs in $\boldsymbol{\tilde{O}(m^{\omega})}$ time, the second algorithm runs in $\boldsymbol{O(Nm^{\omega-1})}$ time and the third algorithm runs in $\boldsymbol{\tilde{O}(N^2g+Nmg^2+mg^3)}$ time which is nearly quadratic time when $\boldsymbol{g=O(1)}$. We also study the problem of finding a minimum cycle basis in an undirected graph $\boldsymbol{G}$ with $\boldsymbol{n}$ vertices and $\boldsymbol{m}$ edges. The best known algorithm for this problem runs in $\boldsymbol{O(m^{\omega})}$ time. Our algorithm, which has a simpler high-level description, but is slightly more expensive, runs in $\boldsymbol{\tilde{O}(m^{\omega})}$ time.

We also provide a practical implementation of computing the minimum homology basis for general weighted complexes. The implementation is broadly based on the algorithmic ideas described in this paper, differing in its use of practical subroutines. Of these subroutines, the more costly step makes use of a parallel implementation, thus potentially addressing the issue of scale. We compare results against the currently known state of the art implementation (ShortLoop).

Keywords: Computational topology, Minimum homology basis, Minimum cycle basis, Matrix computations, Randomized algorithms

1 Introduction

Minimum cycle bases in graphs have several applications, for instance, in analysis of electrical networks, analysis of chemical and biological pathways, periodic scheduling, surface reconstruction and graph drawing. Also, algorithms from diverse application domains like electrical circuit theory and structural engineering require cycle basis computation as a preprocessing step. Cycle bases of small size offer a compact description of representatives that is advantageous from a mathematical as well as from an application viewpoint. For this reason, the problem of computing a minimum cycle basis has received a lot of attention, both in its general setting as well as in special classes of graphs such as planar graphs, sparse graphs, dense graphs, network graphs, and so on. We refer the reader to [21] for a comprehensive survey.

In topological data analysis, "holes" of different dimensions in a geometric dataset constitute "features" of the data. Algebraic topology offers a rigorous language to formalize our intuitive picture of holes in these geometric objects. More precisely, a basis for the first homology group H_1 can be taken as a representative of the one-dimensional holes in the geometric object. The advantages of using minimum homology bases are twofold: firstly, one can bring geometry in picture by assigning appropriate weights to edges, and secondly, smaller cycles are easier to understand and analyze, especially visually. We focus solely on the bases of the first homology group since the problem of computing a lightest basis for higher homology groups with $\mathbb{Z}_2$ coefficients was shown to be NP-hard by Chen and Freedman [9].

Outline and Contributions

In Section 2, we discuss the necessary preliminaries for cycle basis and homology basis computation. In Section 3, we describe a simple algorithm for computing a minimum cycle basis of a weighted graph.

In Section 4, we prove a structural result relating minimum homology bases to minimum cycle bases. Specifically, we show that every minimum cycle basis of the 1-skeleton of a complex contains a minimum homology basis. In Section 5, we describe two randomized algorithms (Algorithms 4 and 5) for computing a minimum homology basis of a complex. In Section 6, we describe a third randomized algorithm (Algorithm 6) for the same problem that runs in nearly quadratic time when the first Betti number of the complex is a constant. All three algorithms use state-of-the-art black box matrix operations, and only the second one (Algorithm 5) uses the structural result proved in Section 4. Algorithm 4 computes the column rank profile of a matrix consisting of tight cycles appended to the boundary matrix of the complex, whereas Algorithm 5 computes the column rank profile of a matrix consisting of a minimum cycle basis of the 1-skeleton of the complex appended to the boundary matrix of the complex. Algorithm 6 builds a matrix $\mathbf{B}$ containing the minimum homology basis by iteratively finding a lexicographically smallest cycle from the set of tight cyles that is linearly independent of the current set of cycles stored in $\mathbf{B}$ by using a randomized binary search.

To demonstrate that the ideas in this work have practical relevance, we provide an implementation of an algorithm based on Algorithm 5. The implemented algorithm for computing minimum homology basis differs from Algorithm 5 in the use of

matrix operations for computational efficiency. In particular, unlike Algorithm 5, it uses matrix reduction algorithm from the PHAT library [4]. The details of the implementation can be found in Section 8. Finally, in Section 9, we describe experiments on real-world datasets as well as random complexes. We show that our implementation FastLoop[1] consistently outperforms the state-of-the-art homology basis computation, ShortLoop [12, 26]).

Finally, we remark that this paper is the full and extended version of the conference version published in [28].

2 Background and Preliminaries

2.1 Cycle Basis

Let $G = (V, E)$ be a connected graph. Throughout this paper, in context of graphs, we use n to denote the number of vertices $|V|$ and m to denote the number of edges $|E|$. A subgraph of G that has even degree for each vertex is called a *cycle* of G. A cycle is called *elementary* if the set of edges form a connected subgraph in which each vertex has degree 2. We associate an incidence vector C, indexed on E, to each cycle, so that $C_e = 1$ if e is an edge of the cycle, and $C_e = 0$ otherwise. The set of incidence vectors of cycles forms a vector space over $\mathbb{Z}_2$, called the *cycle space* of G. It is a well-known fact that for a connected graph G, the cycle space is of dimension $m - n + 1$. Throughout, we use ν to denote the dimension of the cycle space of a graph. A basis of the cycle space, that is, a maximal linearly independent set of cycles is called a *cycle basis*.

Suppose that the edges of G have non-negative weights. Then, the weight of a cycle is the sum of the weights of its edges, and the weight of a cycle basis is the sum of the weights of the basis elements. The problem of computing a cycle basis of minimum weight (or lightest cycle basis) is called the *minimum cycle basis* problem. Since we assume all edge weights to be non-negative, there always exists a minimum cycle basis of elementary cycles, allowing us to focus on minimum cycle basis comprising entirely of elementary cycles.

Moreover, we define the *length of a cycle* to be the number of edges in the cycle, and the *length of a cycle basis* to be the sum of lengths of the cycles of the basis elements.

An elementary cycle C is *tight* if it contains a lightest path between every pair of points in C. We denote the set of all tight cycles in the graph by $\mathcal{T}$. Tight cycles are sometimes also referred to as *isometric* cycles [2, 21]. Tight cycles play an important role in designing algorithms for minimum cycle bases, owing to the following theorem by Horton.

Theorem 1 (Horton [18])**.** *A minimum cycle basis $\mathcal{M}$ consists only of tight cycles.*

A key structural property about minimum cycle bases was proved by de Pina.

Theorem 2 (de Pina [27])**.** *Cycles $C_1 \ldots, C_\nu$ form a minimum cycle basis if there exists m-dimensional vectors $S_1, \ldots, S_\nu$ that satisfy the following three conditions for all i, $1 \leq i \leq \nu$.*

Prefix Orthogonality: $\langle C_k, S_i \rangle = 0$ for all $1 \leq k < i$.

[1] https://bitbucket.org/vgl_iisc/fastloop/

Non-Orthogonality: $\langle C_i, S_i \rangle = 1$.
Shortness: C_i *is a minimum weight cycle in* $\mathcal{T}$ *with* $\langle C_i, S_i \rangle = 1$.

The vectors $S_1, \ldots, S_\nu$ in Theorem 2 are called *support vectors*. The recent line of algorithmic work [2, 20, 22, 24, 27] on the minimum cycle basis problem rely on Theorem 2. In fact, these algorithms may all be seen as refinements of the algorithm by de Pina, see Algorithm 1.

Algorithm 1 De Pina's Algorithm for computing a minimum cycle basis

1: Initialize S_i to the i-th unit vector e_i for $1 \leq i \leq \nu$
2: **for** $i \leftarrow 1, \ldots, \nu$ **do**
3: Compute a minimum weight cycle C_i with $\langle C_i, S_i \rangle = 1$.
4: **for** $j \leftarrow i+1, \ldots, \nu$ **do**
5: $S_j = S_j + \langle C_i, S_j \rangle S_i$
6: RETURN $\{C_1, \ldots, C_\nu\}$.

Algorithm 1 works by inductively maintaining a set of support vectors $\{S_i\}$ so that the conditions of Theorem 2 are satisfied when the algorithm terminates. In particular, Lines 4 and 5 of the algorithm ensure that the set of vectors S_j for $j > i$ are orthogonal to vectors $C_1, \ldots, C_i$. Updating the vectors S_j as outlined in Lines 4 and 5 of Algorithm 1 takes time $O(m^3)$ time in total. Using a divide and conquer procedure for maintaining S_j, Kavitha et al. [20] improved the cost of maintaining the support vectors to $O(m^\omega)$ effectively improving the cost of computing minimum cycle basis from $O(m^\omega n)$ to $O(m^2 n + mn^2 \log n)$, see Algorithm 2.

Algorithm 2 Divide and conquer procedure for fast computation of support vectors by Kavitha et al. [20]

1: Initialize S_i to the i-th unit vector e_i for $1 \leq i \leq \nu$.
2: MinCycleBasis$(1, \nu)$.

3: **procedure** MINCYCLEBASIS(ℓ, u)
4: **if** $\ell = u$ **then**
5: Compute a minimum weight cycle C_ℓ with $\langle C_\ell, S_\ell \rangle = 1$.
6: **else**
7: $q \leftarrow \lfloor \frac{(\ell+u)}{2} \rfloor$.
8: MinCycleBasis(ℓ, q).
9: $\mathbf{C} \leftarrow [C_\ell, \ldots, C_q]$.
10: $\mathbf{W} \leftarrow (\mathbf{C}^T[S_\ell, \ldots, S_q])^{-1}\mathbf{C}^T[S_{q+1}, \ldots, S_u]$.
11: $[S_{q+1}, \ldots, S_u] \leftarrow [S_{q+1}, \ldots, S_u] + [S_\ell, \ldots, S_q]\mathbf{W}$.
12: MinCycleBasis$(q+1, u)$.
13: RETURN $\{C_1, \ldots, C_\nu\}$.

Lemma 3 (Lemma 5.6 in [21]). *The total number of arithmetic operations performed in lines 9 to 11 of Algorithm 2 is $O(m^\omega)$. That is, the support vectors satisfying conditions of Theorem 2 can be maintained in $O(m^\omega)$ time.*

Finally, in [2], Amaldi et al. designed an $O(m^\omega)$ time algorithm for computing a minimum cycle basis by improving the complexity of Line 5 of Algorithm 2 to $o(m^\omega)$ (from $O(m^2n)$ in [20]), while using the $O(m^\omega)$ time divide-and-conquer template for maintaining the support vectors as presented in Algorithm 2. The $o(m^\omega)$ procedure for Line 5 is achieved by performing a Monte Carlo binary search on the set of tight cycles (sorted by weight) to find a minimum weight cycle C_i that satisfies $\langle C_i, S_i \rangle = 1$. An efficient binary search is made possible on account of the following key structural property about tight cycles.

Theorem 4 (Amaldi et al. [2]). *The sum of lengths of the tight cycles in a graph is at most $n\nu$.*

Amaldi et al. [2] also show that there exists an $O(nm)$ algorithm to compute the set of all the tight cycles of an undirected graph G. See Sections 2 and 3 of [2] for details about Amaldi et al.'s algorithm.

2.2 Matrix operations

The *column rank profile* (respectively *row rank profile*) of an $m \times n$ matrix $\mathbf{A}$ with rank r, is the lexicographically smallest sequence of r indices $[i_1, i_2, \ldots, i_r]$ (respectively $[j_1, j_2, \ldots, j_r]$) of linearly independent columns (respectively rows) of $\mathbf{A}$. Suppose that $\{a_1, a_2, \ldots, a_n\}$ represent the columns of $\mathbf{A}$. Then, following Busaryev et al. [7], we define the *earliest basis* of $\mathbf{A}$ as the set of columns $\mathcal{E}(\mathbf{A}) = \{a_{i_1}, a_{i_2}, \ldots, a_{i_r}\}$. Throughout, we use $\mathsf{nnz}(\mathbf{A})$ to denote the number of nonzero entries in matrix $\mathbf{A}$.

It is well-known that classical Gaussian elimination can be used to compute rank profile in $O(nmr)$ time. The current state-of-the-art deterministic matrix rank profile algorithms run in $O(mnr^{\omega-2})$ time.

Theorem 5 ([15, 19]). *There is a deterministic $O(mnr^{\omega-2})$ time algorithm to compute the column rank profile of an $m \times n$ matrix $\mathbf{A}$.*

In case of randomized algorithms, Cheung, Kwok and Lau [10] presented a breakthrough Monte Carlo algorithm for rank computation that runs in $(\mathsf{nnz}(\mathbf{A}) + r^\omega)^{1+o(1)}$ time, where $o(1)$ in the exponent captures some missing multiplicative $\log n$ and $\log m$ factors. Equivalently, the complexity for Cheung et al.'s algorithm can also be written as $\tilde{O}(\mathsf{nnz}(\mathbf{A}) + r^\omega)$. The notation $\tilde{O}(\cdot)$ is often used in literature to hide small poly logarithmic factors in time bounds. While the algorithm by Cheung et al. also computes r linearly independent columns of $\mathbf{A}$, the columns may not correspond to the column rank profile. However, building upon Cheung et al.'s work, Storjohann and Yang established the following result.

Theorem 6 (Storjohann and Yang [29, 30, 33]). *There exists a Monte Carlo algorithm for computing row (resp. column) rank profile of a matrix $\mathbf{A}$ that runs in $\tilde{O}(\mathsf{nnz}(\mathbf{A}) + r^\omega)$ time. The failure probability of this algorithm is $1/2$.*

In Section 6, we use Wiedemann's algorithms as subroutines to design an output sensitive algorithm to compute the minimum homology basis of a complex.

Remark 1. *Wiedemann [32] presented randomized algorithms to compute the rank of an $m \times n$ matrix $\mathbf{A}$ over a finite field and for computing a solution to a sparse system*

of linear equations $\mathbf{A}x = b$ *(if one exists). Both algorithms run in* $\tilde{O}(n_1(\mathsf{nnz}(\mathbf{A}) + n_1))$ *time, where* $n_1 = \max(m, n)$ *is the maximal dimension of the matrix* $\mathbf{A}$*.*

2.3 Homology

In this work, we restrict our attention to simplicial homology with $\mathbb{Z}_2$ coefficients. For a general introduction to algebraic topology, we refer the reader to [17]. Below we give a brief description of homology over $\mathbb{Z}_2$.

Let K be a connected simplicial complex. We use $K^{(p)}$ to denote the set of p-dimensional simplices of K, and n_p the number of p-dimensional simplices of K. Also, the p-dimensional skeleton of K is denoted by K_p. In particular, the 1-skeleton of K (which is an undirected graph) is denoted by K_1. Throughout this paper, in context of simplicial complexes, we use n to denote $|K^{(0)}|$, m to denote $|K^{(1)}|$, and N to denote $|K|$.

We consider formal sums of simplices with $\mathbb{Z}_2$ coefficients, that is, sums of the form $\sum_{\sigma \in K^{(p)}} a_\sigma \sigma$, where each $a_\sigma \in \{0, 1\}$. The expression $\sum_{\sigma \in K^{(p)}} a_\sigma \sigma$ is called a *p-chain*. Since chains can be added to each other, they form an abelian group, denoted by $\mathsf{C}_p(K)$. Since we consider formal sums with coefficients coming from $\mathbb{Z}_2$, which is a field, $\mathsf{C}_p(K)$, in this case, is a vector space of dimension n_p over $\mathbb{Z}_2$. The p-simplices in K give rise to the standard basis for $\mathsf{C}_p(K)$. This establishes a one-to-one correspondence between elements of $\mathsf{C}_p(K)$ and subsets of $K^{(p)}$. Thus, associated with each chain is an incidence vector v, indexed on $K^{(p)}$, where $v_\sigma = 1$ if σ is a simplex of v, and $v_\sigma = 0$ otherwise. The *boundary* of a p-simplex is a $(p-1)$-chain that corresponds to the set of its $(p-1)$-faces. This map can be linearly extended from p-simplices to p-chains, where the boundary of a chain is the $\mathbb{Z}_2$-sum of the boundaries of its elements. Such an extension is known as the *boundary homomorphism*, and denoted by $\partial_p : \mathsf{C}_p(K) \to \mathsf{C}_{p-1}(K)$. A chain $\zeta \in \mathsf{C}_p(K)$ is called a *p-cycle* if $\partial_p \zeta = 0$, that is, $\zeta \in \ker \partial_p$. The group of p-dimensional cycles is denoted by $\mathsf{Z}_p(K)$. As before, since we are working with $\mathbb{Z}_2$ coefficients, $\mathsf{Z}_p(K)$ is a vector space over $\mathbb{Z}_2$. A chain $\eta \in \mathsf{C}_p(K)$ is said to be a p-boundary if $\eta = \partial_{p+1} c$ for some chain $c \in \mathsf{C}_{p+1}(K)$, that is, $\eta \in \operatorname{im} \partial_{p+1}$. All p-boundaries are also p-cycles and sometimes referred to as *trivial cycles*. The p-cycles that are not p-boundaries are referred to as *nontrivial cycles*. The group of p-boundaries is denoted by $\mathsf{B}_p(K)$. In our case, $\mathsf{B}_p(K)$ is also a vector space, and in fact a subspace of $\mathsf{Z}_p(K)$.

Thus, we can consider the quotient space $\mathsf{H}_p(K) = \mathsf{Z}_p(K)/\mathsf{B}_p(K)$. The elements of the vector space $\mathsf{H}_p(K)$, known as the p-th *homology group* of K, are equivalence classes of p-cycles, where p-cycles are equivalent if their $\mathbb{Z}_2$-difference is a p-boundary. Equivalent cycles are said to be *homologous*. For a p-cycle ζ, its corresponding homology class is denoted by $[\zeta]$. Bases of $\mathsf{B}_p(K)$, $\mathsf{Z}_p(K)$ and $\mathsf{H}_p(K)$ are called *boundary bases*, *cycle bases*, and *homology bases* respectively. The dimension of the p-th homology group of K is called the p-th *Betti number* of K, denoted by $\beta_p(K)$. We are primarily interested in the first Betti number $\beta_1(K)$. For notational convenience, let $g = \beta_1(K)$, and denote the dimension of $\mathsf{B}_1(K)$ by b.

Using the standard bases for $\mathsf{C}_p(K)$ and $\mathsf{C}_{p-1}(K)$, the matrix $[\partial_p \sigma_1 \, \partial_p \sigma_2 \cdots \partial_p \sigma_{n_p}]$ whose column vectors are boundaries of p-simplices is called the p-th boundary matrix. Abusing notation, we denote the p-th boundary matrix by ∂_p. For the rest of the

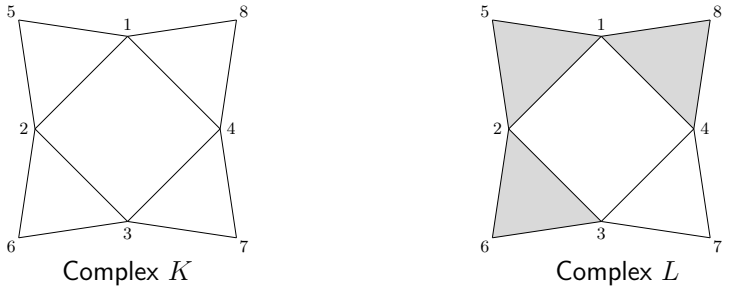


Fig. 1 Consider complexes K and L in the figure above with unit weights on the edges. Since K has no 2-simplices, its 1-skeleton K_1 is identical to K itself. The set of cycles $\mathcal{C} = \{\{1,2,5\},\{1,4,8\},\{3,4,7\},\{2,3,6\},\{1,2,3,4\}\}$ constitutes a minimum cycle basis for the respective 1-skeletons K_1 and L_1 (viewed as graphs). The set $\mathcal{C}$ also constitutes a minimum homology basis for K. The set $\mathcal{C}' = \{\{1,2,3,4\},\{3,4,7\}\}$ constitutes a minimum homology basis for L.

paper, we use n, m and N to denote the number of vertices, edges and simplices in the complex respectively.

This paper focuses on 1-dimensional cycles. A set of 1-cycles $\{\zeta_1, \ldots, \zeta_g\}$ is called a *homology cycle basis* if the set of classes $\{[\zeta_1], \ldots, [\zeta_g]\}$ forms a homology basis. For brevity, we abuse notation by using the term "homology basis" for $\{\zeta_1, \ldots, \zeta_g\}$. Assigning non-negative weights to the edges of K, the *weight of a cycle* is the sum of the weights of its edges, and the *weight of a homology basis* is the sum of the weights of the basis elements. The problem of computing a minimum weight basis of $\mathsf{H}_1(K)$ (that is, a *lightest basis* of $\mathsf{H}_1(K)$) is called the *minimum homology basis* problem. Note that, when the input simplicial complex is a graph, the notions of homology basis and cycle basis coincide. Please refer to Figure 1 for an example.

Moreover, we define the *length of a* 1*-cycle* to be the number of edges in the 1-cycle, and the *length of a homology basis* to be the sum of lengths of the 1-cycles of the basis elements.

For the special case when the input complex is a surface, Erickson and Whittlesey [16] gave a $O(n^2 \log n + gn^2 + g^3 n)$-time algorithm. Recently, Borradaile et al. [6] gave an improved deterministic algorithm that runs in $O(g^3 n \log n + m)$ assuming the lightest paths are unique. For small values of g, the algorithm in [6] runs in nearly linear time.

Furthermore, Dey et al. [12] and Chen and Freedman [8] generalized the results by Erickson and Whittlesey [16] to arbitrary complexes. Subsequently, introducing the technique of annotations, Busaryev et al. [7] improved the complexity to $O(N^\omega + N^2 g^{\omega-1})$. More recently, Dey et al. [13] designed an $O(N^\omega + N^2 g)$ time algorithm by adapting the divide and conquer algorithm for computing a minimum cycle basis of Kavitha et al. [20] for the purpose of computing a minimum homology basis. Dey et al. also designed a randomized 2-approximation algorithm for the same problem that runs in $O(N^\omega \sqrt{N \log N})$ expected time. It is possible to etablish a tighter bound for the algorithm by Dey et al. [13] (See Section 7).

2.4 Matroids

A matroid $\mathscr{M}$ consists of a pair $(S, \mathscr{I})$, where S is a finite ground set and $\mathscr{I}$ is a family of subsets of S satisfying the following axioms:

1. $\emptyset \in \mathscr{I}$;
2. if $I \in \mathscr{I}$ and $J \subseteq I$, then $J \in \mathscr{I}$; and
3. if $I, K \in \mathscr{I}$ and $|I| < |K|$, then there is an element $e \in K \setminus I$ such that $I \bigcup \{e\} \in \mathscr{I}$.

If a set $I \subset S$ belongs to $\mathscr{I}$, then it is called an *independent set*; otherwise it is called a *dependent set*. A *circuit* in a matroid $\mathscr{M}$ is a minimal dependent subset of S. All proper subsets of circuits are independent sets. A maximal independent set is called a *basis* of the matroid.

Matroids admit a very useful property that goes by the name of *basis exchange property*: If A and B are distinct bases of a matroid and $a \in A \setminus B$, then there exists an element $b \in B \setminus A$ such that $A(\setminus \{a\}) \bigcup \{b\}$ is again a basis. A *weighted matroid* is a matroid $\mathscr{M}$ equipped with a weight function $w : S \to \mathbb{R}^+$ that additively extends to all subsets of S. From an algorithmic standpoint, the most important property of weighted matroids is that there is a greedy algorithm for these matroids that computes the maximum (minimum) weight basis. It can be easily checked that B is a minimum weight basis of a matroid if and only if none of the elements of B can be exchanged for a lighter element while still preserving linear independence. As an immediate consequence, if the elements in two distinct minimum bases are sorted by weight, then the ordered sets of sorted weights coincide.

Let the cycle space of G be the ground set Ω, and let $\mathscr{I}$ be defined as follows.

$$\mathscr{I} = \{I \mid I \text{ is a linearly independent set of cycles of } G\}$$

Then, the pair $\mathscr{M} = (\Omega, \mathscr{I})$ forms a matroid. When combined with a weight function on edges, it becomes a weighted matroid. Cycle bases of G correspond to the bases of $\mathscr{M}$.

Analogously, let the nontrivial 1-cycles of a 2-complex K be the ground set Ω', and let $\mathscr{I}'$ be defined as follows.

$$\mathscr{I}' = \{I \mid I \text{ is a linearly independent set of nontrivial cycles of } \mathsf{K}\}$$

Then, the pair $\mathscr{M}' = (\Omega', \mathscr{I}')$ together with a weight function on the edges forms a weighted matroid. Sets of cycles whose classes form 1-homology bases of K are the bases of matroid $\mathscr{M}'$.

Key to our algorithms is the property that the cycles in a cycle (homology) basis can be exchanged with other cycles while preserving independence.

Remark 2. *For any subset Ω_1 of the ground set Ω (resp. any subset Ω'_1 of the ground set Ω') with the property that Ω_1 (resp. Ω'_1) contains a minimum weight basis, the column rank profile of a matrix whose columns consist of cycles from Ω_1 (resp. cycles from Ω'_1) is* the *greedy algorithm that returns a minimum weight basis.*

3 An algorithm for computing a minimum cycle basis

Given a graph $G = (V, E)$, let $\{C_1, \ldots, C_{|\mathcal{T}|}\}$ be the list of tight cycles in G sorted by weight, and let $\mathbf{T}(G) = [C_1\, C_2 \ldots\, C_{|\mathcal{T}|}]$ be the matrix formed with cycles C_i as its columns. Using Theorem 4, since the total length of tight cycles is at most $n\nu$, and since each tight cycle consists of at least three edges, we have that $|\mathcal{T}| \leq \frac{n\nu}{3}$. Also, the rank of $\mathbf{T}(G)$ is ν and $\mathbf{T}(G)$ is a sparse matrix with $\mathsf{nnz}(\mathbf{T}(G))$ bounded by $n\nu$. This sparsity is implicitly used in the design of the Monte Carlo binary search algorithm for computing a minimum cycle basis, as described in [2]. We now present a simple and fast algorithm for minimum cycle basis that exploits the sparsity and the low rank of $\mathbf{T}(G)$ more directly.

Algorithm 3 Algorithm for minimum cycle basis

1: Compute the sorted list of tight cycles in G, and assemble the matrix $\mathbf{T}(G)$.
2: Compute the column rank profile $[i_1, i_2, \ldots, i_\nu]$ of $\mathbf{T}(G)$ using Storjohann and Yang's algorithm described in [30].
3: RETURN $\mathcal{E}(\mathbf{T}(G))$.

Theorem 7. *There is a Monte Carlo algorithm that computes the minimum cycle basis in $\tilde{O}(m^\omega)$ time, with failure probability at most $1/2$.*

Proof. The correctness of the algorithm follows immediately from Theorem 1. As noted in Section 2.4, the cycles of a graph form a weighted matroid. Since the cycles in $\mathbf{T}(G)$ are sorted by weight, and since the tight cycles of a graph are guaranteed to contain a minimum cycle basis, from Remark 2 the column rank profile of $\mathbf{T}(G)$ is the greedy algorithm, and the earliest basis $\mathcal{E}(\mathbf{T}(G))$ is a minimum cycle basis.

The list of tight cycles in G can be computed in $O(nm)$ time using the algorithm described in Section 2 of [2]. Hence, Step 1 of Algorithm 3 takes $O(nm \log(nm))$ time (which in turn is same as $O(nm \log n)$ time). Moreover, using Theorem 6, the complexity of Step 2 is bounded by $\tilde{O}(n\nu + \nu^\omega)$. Since $n, \nu < m$, the complexity of Algorithm 3 is bounded by $\tilde{O}(m^\omega)$. Using Theorem 6, the failure probability of the algorithm is at most $1/2$. □

4 Minimum homology basis, minimum cycle basis and tight cycles

To begin with, note that since every graph is a 1-dimensional simplicial complex, the minimum cycle basis problem is a restriction of the minimum homology basis problem to instances (simplicial complexes) that have no 2-simplices. In this section, we refine this observation by deriving a closer relation between the two problems.

We will now define some notation that we will use in Lemma 8 and Theorem 9.

Notation 1. *We assume that we are provided a complex K in which all edges are assigned non-negative weights. Let $w : E \to \mathbb{R}^+$ be a non-negative weight function*

on the edges of a complex K, and let $\mathcal{B} = \{\eta_1, \ldots, \eta_b\}$ be a basis for the boundary vector space $\mathsf{B}_1(K)$ indexed so that $w(\eta_i) \leq w(\eta_{i+1})$, $1 \leq i < b$ (with ties broken arbitrarily). Also, let $\mathcal{H} = \{\zeta_1, \ldots, \zeta_g\}$ be a minimum homology basis of K indexed so that $w(\zeta_i) \leq w(\zeta_{i+1})$, $1 \leq i < g$ (with ties broken arbitrarily). Then, the set $\mathcal{C} = \{\eta_1, \ldots, \eta_b, \zeta_1, \ldots, \zeta_g\}$ is a cycle basis for K_1. Let $\mathcal{M}$ be a minimum cycle basis of K_1. Every element $C \in \mathcal{M}$ is homologous to a cycle $\sum_{i=1}^{g} a_i\zeta_i$ where $a_i \in \{0, 1\}$ for each i. Then, for some fixed integers p and q, $\mathcal{M} = \{B_1, \ldots, B_q, C_1, \ldots, C_p\}$ is indexed so that the elements $B_1, \ldots, B_q$ are trivial and the elements $C_1, \ldots, C_p$ are nontrivial cycles. Also, we have $w(B_j) \leq w(B_{j+1})$ for $1 \leq j < q$ (with ties broken arbitrarily), and $w(C_j) \leq w(C_{j+1})$ for $1 \leq j < p$ (with ties broken arbitrarily).

Remark 3. *We note that $p \geq g$. This is because a trivial cycle can be obtained as a linear combination of nontrivial cycles, but a nontrivial cycle cannot be obtained as a linear combination of trivial cycles. Thus, there have to be at least g linearly nontrivial cycles in $\mathcal{M}$ to be able to obtain all cycles in any homology basis from linear combinations of cycles in $\mathcal{M}$.*

Lemma 8. *Given a simplicial complex K, suppose that $\mathcal{H}$, $\mathcal{C}$ and $\mathcal{M}$ are defined as in Notation 1. The following two statements are true*

1. *$w(\zeta_1) = w(C_1)$, and*
2. *there exists a minimum homology basis $\overline{\mathcal{H}}$ with ζ_1 homologous to C_1.*

Proof. Targeting a contradiction, suppose there exists a minimum homology basis with $w(\zeta_1) < w(C_1)$. Let $\zeta_1 = \sum_{i=1}^{p} a_iC_i + \sum_{j=1}^{q} b_jB_j$, where $a_i \in \{0, 1\}$ for each i and $b_j \in \{0, 1\}$ for each j. Since ζ_1 is a nontrivial cycle, there exists at least one i with $a_i = 1$. Let $\ell \in [1, p]$ be the largest index in the above equation with $a_\ell = 1$. Rewriting the equation, we obtain $C_\ell = \sum_{i=1}^{\ell-1} a_iC_i + \sum_{j=1}^{q} b_jB_j + \zeta_1$. Since $w(\zeta_1) < w(C_1)$ by assumption, we have $w(\zeta_1) < w(C_\ell)$ because $w(C_\ell) \geq w(C_1)$ by indexing of $\mathcal{M}$. It follows that the basis obtained by exchanging C_ℓ for ζ_1, that is, $\{B_1, \ldots, B_q, \zeta_1, C_1, \ldots, C_{\ell-1}, C_{\ell+1}, \ldots, C_p\}$ gives a smaller cycle basis than the minimum one, a contradiction.

Once again, targeting a contradiction, suppose there exists a minimum homology basis with $w(\zeta_1) > w(C_1)$. Let $C_1 = \sum_{i=1}^{g} a_i\zeta_i + \sum_{j=1}^{b} b_j\eta_j$. As before, since C_1 is not trivial, there exists at least one i with $a_i = 1$. Let $\ell \in [1, g]$ be the largest index in the above equation with $a_\ell = 1$. Then, $\zeta_\ell = \sum_{i=1}^{\ell-1} a_i\zeta_i + \sum_{j=1}^{b} b_j\eta_j + C_1$. Note that $w(\zeta_\ell) \geq w(\zeta_1)$ because of the indexing, and $w(\zeta_1) > w(C_1)$ by assumption. Therefore, the set $\{C_1, \zeta_1, \ldots, \zeta_{\ell-1}, \zeta_{\ell+1}, \ldots, \zeta_p\}$ obtained by exchanging ζ_ℓ for C_1 gives a smaller homology basis than the minimum one, a contradiction. This proves the first part of the lemma.

From the first part of the lemma, we have $w(\zeta_1) = w(C_1)$ for every minimum homology basis. Let $\mathcal{H}$ be an arbitrary minimum homology basis. Then, if C_1 is not homologous to $\zeta_1 \in \mathcal{H}$, by using basis exchange we can obtain $\overline{\mathcal{H}} = \{C_1, \zeta_1, \ldots, \zeta_{\ell-1}, \zeta_{\ell+1}, \ldots, \zeta_p\}$, which is the minimum homology basis with its first element homologous to C_1, and having the same weight as $w(C_1)$, proving the claim. □

We now prove a theorem which allows us to harness fast algorithms for minimum cycle basis in service of improving time complexity of algorithms for minimum homology basis.

Theorem 9. *Given a simplicial complex K, suppose that $\mathcal{H}$, $\mathcal{C}$ and $\mathcal{M}$ are defined as in Notation 1. Then, there exists a minimum homology basis $\overline{\mathcal{H}}$ of K, and a set $\{C_{i_1}, \ldots, C_{i_g}\} \subset \{C_1, \ldots, C_p\} \subset \mathcal{M}$ such that, for every $k \in [1, g]$, we have C_{i_k} homologous to a cycle spanned by $\zeta_1, \ldots, \zeta_k$, and $w(C_{i_k}) = w(\zeta_k)$. Moreover, $i_1 = 1$, and i_k for $k > 1$ is the smallest index for which C_{i_k} is not homologous to any cycle spanned by $\{C_{i_1}, \ldots, C_{i_{k-1}}\}$. In particular, the set $\{C_{i_1}, \ldots, C_{i_g}\} \subset \mathcal{M}$ constitutes a minimum homology basis of K.*

Proof. The key argument is essentially the same as for the proof of Theorem 8. Nonetheless, we present it here for the sake of completeness. We shall prove the claim by induction. Theorem 8 covers the base case. By induction hypothesis, there is an integer $k < g$, and a minimum homology basis $\mathcal{H} = \{\zeta_1, \ldots, \zeta_g\}$, for which, vectors $\{C_{i_1}, \ldots, C_{i_k}\} \subseteq \{C_1, \ldots, C_p\}$ are such that, for every $j \in [1, k]$, we have C_{i_j} homologous to a cycle spanned by $\zeta_1, \ldots, \zeta_j$, and $w(C_{i_j}) = w(\zeta_j)$. Let i_{k+1} be the smallest index for which $C_{i_{k+1}} \in \{C_1, \ldots, C_p\}$ is not homologous to any cycle spanned by $\{C_{i_1}, \ldots, C_{i_k}\}$. Such an index $i_{k+1} \leq p$ exists assuming every nontrivial cycle in $\mathcal{C}$ can be obtained as a linear combination of cycles in $\mathcal{M}$.

Suppose that $w(\zeta_{k+1}) < w(C_{i_{k+1}})$. Let $\zeta_{k+1} = \sum_{i=1}^{p} a_i C_i + \sum_{j=1}^{q} b_j B_j$. Let $\ell \in [1, p]$ be the largest index in the above equation with $a_\ell = 1$. Then, $C_\ell = \sum_{i=1}^{\ell-1} a_i C_i + \sum_{j=1}^{q} b_j B_j + \zeta_{k+1}$. From the induction hypothesis, we can infer that $\ell \geq i_{k+1}$, and hence $w(C_\ell) \geq w(C_{i_{k+1}})$ by indexing of $\mathcal{M}$. Thus, if $w(\zeta_{k+1}) < w(C_{i_{k+1}})$, then we have $w(\zeta_{k+1}) < w(C_\ell)$. It follows that, $\{B_1, \ldots, B_q, \zeta_{k+1}, C_1, \ldots, C_{\ell-1}, C_{\ell+1}, \ldots, C_p\}$ obtained by exchanging C_ℓ for ζ_{k+1} gives a smaller cycle basis than the minimum one, contradicting the minimality of $\mathcal{H}$.

Now, suppose that $w(\zeta_{k+1}) > w(C_{i_{k+1}})$. Let $C_{i_{k+1}} = \sum_{i=1}^{g} a_i \zeta_i + \sum_{j=1}^{b} b_j \eta_j$. Let $\ell \in [1, g]$ be the largest index in the above equation with $a_\ell = 1$. Rewriting the equation, we obtain $\zeta_\ell = \sum_{i=1}^{\ell-1} a_i \zeta_i + \sum_{j=1}^{b} b_j \eta_j + C_{i_{k+1}}$. Again, using the induction hypothesis, $\ell \geq k + 1$, and hence, $w(\zeta_\ell) \geq w(\zeta_{k+1})$ because of the indexing. Since we have assumed $w(\zeta_{k+1}) > w(C_{i_{k+1}})$, this gives us $w(\zeta_\ell) > w(C_{i_{k+1}})$. Hence, the set $\{C_{i_{k+1}}, \zeta_1, \ldots, \zeta_{\ell-1}, \zeta_{\ell+1}, \ldots, \zeta_p\}$ obtained by exchanging ζ_ℓ for $C_{i_{k+1}}$ gives a smaller homology basis than the minimum one, contradicting the minimality of $\mathcal{H}$.

From the first part of the proof, we have established that $w(C_{i_{k+1}}) = w(\zeta_{k+1})$. So, if $C_{i_{k+1}}$ is not homologous to $\zeta_{k+1} \in \mathcal{H}$ and $w(\zeta_{k+1}) = w(C_{i_{k+1}})$, then $\overline{\mathcal{H}} = \{C_{i_{k+1}}, \zeta_1, \ldots, \zeta_{\ell-1}, \zeta_{\ell+1}, \ldots, \zeta_p\}$ obtained by exchanging ζ_ℓ for $C_{i_{k+1}}$ is the desired minimum homology basis, proving the induction claim. □

Previously, it was known from Erickson and Whittlesey [16] that $\mathcal{H}$ is contained in $\mathcal{T}$.

Theorem 10 (Erickson and Whittlesey [16])**.** *With non-negative weights, every cycle in a lightest basis of $\mathsf{H}_1(K)$ is tight. That is, if $\mathcal{H}$ is any minimum homology basis of K, then $\mathcal{H} \subset \mathcal{T}$.*

Using Theorems 1 and 9, we can refine the above observation.

Corollary 11. *Let $\mathcal{T}$ denote the set of tight cycles of K_1, and let $\mathcal{M}$ be a minimum cycle basis of K_1. Then, there exists a minimum homology basis $\mathcal{H}$ of K such that $\mathcal{H} \subset \mathcal{M} \subset \mathcal{T}$.*

5 Algorithms for minimum homology basis

To begin with, note that since $\mathsf{C}_p(K), \mathsf{Z}_p(K), \mathsf{B}_p(K)$ and $\mathsf{H}_p(K)$ are vector spaces, the problem of computing a minimum homology basis can be couched in terms of matrix operations.

Given a complex K, let $\{C_1, \ldots, C_{|\mathcal{T}|}\}$ be the list of tight cycles in K_1 sorted by weight, and let $\mathbf{T} = [C_1\, C_2 \ldots C_{|\mathcal{T}|}]$ be the matrix formed with cycles C_i as its columns. Then, the matrix $\mathbf{Y} = [\partial_2 \mid \mathbf{T}]$ has $O(N + n\nu)$ columns and $O(N + n\nu)$ non-zero entries since $\mathbf{T}$ has $O(n\nu)$ columns and $O(n\nu)$ non-zero entries by Theorem 4, and ∂_2 has $O(N)$ columns and $O(N)$ non-zero entries. Since $\mathbf{Y}$ has m rows, the rank of $\mathbf{Y}$ is bounded by m. This immediately suggests an algorithm for computing minimum homology basis analogous to Algorithm 3.

Algorithm 4 Algorithm for minimum homology basis

1: Compute the sorted list of tight cycles in $\mathbf{T}$, and assemble matrix $\mathbf{Y}$.
2: Compute the column rank profile $[j_1, j_2, \ldots, j_b, i_1, i_2, \ldots, i_g]$ of $\mathbf{Y}$ using Storjohann and Yang's algorithm [30], where columns $\{\mathbf{Y}_{j_k}\}$ and $\{\mathbf{Y}_{i_\ell}\}$ are linearly independent columns of ∂_2 and $\mathbf{T}$ respectively.
3: Return Columns $\{\mathbf{Y}_{i_1}, \mathbf{Y}_{i_2}, \ldots, \mathbf{Y}_{i_g}\}$.

Theorem 12. *Algorithm 4 is a Monte Carlo algorithm for computing a minimum homology basis that runs in $\tilde{O}(m^\omega)$ time with failure probability at most $\frac{1}{2}$.*

Proof. The correctness of the algorithm is an immediate consequence of Corollary 11. As noted in Section 2.4, the nontrivial cycles of a complex form a weighted matroid. Since the cycles in $\mathbf{T}$ are sorted by weight, and since the tight cycles of the 1-skeleton of the complex is guaranteed to contain a minimum homology basis, from Remark 2, the column rank profile of $\mathbf{T}$ is the greedy algorithm that returns a minimum homology basis.

The list of tight cycles in G can be computed in $O(nm)$ time using the algorithm described in Section 2 of [2]. Hence, Step 1 of Algorithm 4 takes $O(nm \log n)$ time. Moreover, using Theorem 6, the complexity of Step 2 is bounded by $\tilde{O}(N + n\nu + m^\omega)$, which is the same as $\tilde{O}(m^\omega)$ since N and $n\nu$ are both in $\tilde{O}(m^\omega)$, and the failure probability is at most $1/2$. □

When the number of 2-simplices in complex K is significantly smaller than the number of edges, the complexity for minimum homology can be slightly improved by decoupling the minimum homology basis computation from the minimum cycle basis computation, as illustrated in Algorithm 5.

Algorithm 5 Algorithm for minimum homology basis

1: Compute a minimum cycle basis $\mathcal{M}$ of K_1 using the Monte Carlo algorithm by Amaldi et al. [2]. Let $\mathbf{M}$ be the matrix whose columns are cycle vectors in $\mathcal{M}$ sorted by weight.
2: Assemble the matrix $\mathbf{Z} = [\partial_2 \mid \mathbf{M}]$.
3: Compute the column rank profile $[j_1, j_2, \ldots, j_b, i_1, i_2, \ldots, i_g]$ of $\mathbf{Z}$ using the deterministic algorithm by Jeannerod et al. [19], where columns $\{\mathbf{Z}_{j_k}\}$ and $\{\mathbf{Z}_{i_\ell}\}$ are linearly independent columns of ∂_2 and $\mathbf{M}$ respectively.
4: RETURN Columns $\{\mathbf{Z}_{i_1}, \mathbf{Z}_{i_2}, \ldots, \mathbf{Z}_{i_g}\}$.

Theorem 13. *A minimum homology basis can be computed in $O(Nm^{\omega-1})$ time using the Monte Carlo algorithm described in Algorithm 5. The algorithm fails with probability at most $\nu \log(nm)\, 2^{-k}$, where $k = m^{0.1}$.*

Proof. As in Theorem 12, the correctness of the algorithm is an immediate consequence of Corollary 11 and Remark 2. The algorithm fails only when Step 1 returns an incorrect answer, the probability of which is as low as $\nu \log(nm)\, 2^{-k}$, where $k = m^{0.1}$, see Theorem 3.2 of [2].

The minimum cycle basis algorithm by Amaldi et al. [2] runs in $O(m^\omega)$ time (assuming the current exponent of matrix multiplication $\omega > 2$). Furthermore, using Theorem 5, the complexity of Line 3 is bounded by $O(Nm^{\omega-1})$. So, the overall complexity of the algorithm is $O(m^\omega + Nm^{\omega-1}) = O(Nm^{\omega-1})$. □

Note that in Line 3 of Algorithm 5, it is possible to replace the deterministic algorithm by Jeannerod et al. [19] with the Monte Carlo algorithm by Storjohann and Yang's algorithm [30]. In that case, the complexity of the algorithm will once again be $\tilde{O}(m^\omega)$, and the failure probability will be at most $1 - \frac{1}{2}(1 - \nu \log(nm) 2^{-k})$.

6 A g-sensitive algorithm for minimum homology basis

In this section, we describe a randomized g-sensitive algorithm for computing minimum homology basis whose complexity depends on the value of g. Specifically, when $g = O(1)$, Algorithm 6 computes the minimum homology basis of a complex correctly in nearly quadratic time. To begin with, note that using Corollary 11, we know that the tight cycles of the 1-skeleton K_1 of a complex K contains a minimum homology basis of K. In this algorithm, the tight cycles of K_1 are maintained in a matrix denoted by $\mathbf{T}$. Specifically, the tight cycles are maintained in the columns of $\mathbf{T}$ and are sorted by weight. Essentially, Algorithm 6 builds a matrix $\mathbf{B}$ containing the minimum homology basis by iteratively finding a lexicographically smallest cycle in $\mathbf{T}$ that is linearly independent of the current set of cycles stored in $\mathbf{B}$. It uses binary search each time to locate the lexicographically smallest linearly independent cycle. To make the search procedure efficient, randomization is used. Wiedemann's black box algorithms are used twice in Algorithm 6, first, in Line 2 to estimate g, and then in Line 13 to check if

a randomly selected cycle $\mathbf{w}$ is linearly independent of the basis cycles assembled in matrix $\mathbf{B}$.

Throughout this section, for some indices i, j, $\mathbf{T}[i]$ represents the i-th column of $\mathbf{T}$, and $\mathbf{T}[i \dots j]$ represents the submatrix of $\mathbf{T}$ formed by choosing columns i through j.

Recall that there are g cycles in any homology basis. In Algorithm 6, the outer **for** loop in Lines 6-30 runs g' times, finding one linearly independent cycle in each iteration. Here, $g' = g$ with high probability since Wiedemann's algorithm computes the rank of a matrix correctly with high probability. The **while** loop in Algorithm 6 uses a modification of binary search to find the lexicographically smallest cycles that are linearly independent of cycles in the matrix $\mathbf{B}$. Suppose that we have some probabilistic guarantee that the first k cycles in the minimum homology basis are correctly computed. Now, if $\mathbf{T}[\ell \dots p]$ has a cycle that is linearly independent of the cycles in $\mathbf{B}$, then in the probability amplification for loop of Lines 10-17 such a cycle is identified correctly with probability at least $\left(1 - \frac{1}{m^2}\right)$ (See Lemma 15). On the other hand, if $\mathbf{T}[\ell \dots p]$ does not have a cycle that is linearly independent of the cycles in $\mathbf{B}$, then in the **if** condition of Line 14, a cycle that is linearly dependent on cycles in $\mathbf{B}$ is misidentified as linearly independent with probability at most $\frac{1}{m^2}$ (See Lemma 16). If a linearly independent cycle is successfully identified in $\mathbf{T}[\ell \dots p]$, then the search interval is halved by setting $r \leftarrow \left\lfloor \frac{\ell + r}{2} \right\rfloor$ (See Line 15). On the other hand, if the algorithm fails to find a linearly independent cycle in $\mathbf{T}[\ell \dots p]$, then in the next iteration of the **while** loop, the search interval is halved by setting $\ell \leftarrow \left\lfloor \frac{\ell + r}{2} \right\rfloor + 1$ (See Line 23).

If we have narrowed down the search of the next cycle in the basis to $\mathbf{T}[\ell]$ and $\ell = r$ but $\mathbf{T}[\ell]$ is linearly dependent on cycles in $\mathbf{B}$, then the algorithm has clearly failed and is therefore terminated (See the **if** condition in Line 19). On the other hand, if we have narrowed down the search of the next cycle in the basis to $\mathbf{T}[\ell]$ and $\ell = r$ where $\mathbf{T}[\ell]$ is linearly independent of cycles in $\mathbf{B}$, then we add $\mathbf{T}[\ell]$ to $\mathbf{B}$ and initiate the search for the next cycle in the basis (See the **if** condition in Line 25).

Notation 2. *Given an $m \times n$ matrix $\mathbf{A}$ and a n-dimensional column vector $\mathbf{x}$, the matrix vector product $\mathbf{A} \cdot \mathbf{x}$ to be equal to the vector $\sum_{i=1}^{n} \mathbf{A}_i \mathbf{x}_i$.*

Lemma 14. *If there exists a column vector in $\mathbf{T}[\ell \dots p]$ that is linearly independent of column vectors of $[\mathbf{B} \mid \partial_2]$, then the probability that the vector $\mathbf{w}$ chosen in Line 12 is such that the system of equations $[\mathbf{B} \mid \partial_2] \cdot \mathbf{x} = \mathbf{w}$ in Line 13 does not have a solution is at least $\frac{1}{2}$.*

Proof. We begin with noting that $\mathbf{w}$ is set to $\mathbf{T}[\ell \dots p] \cdot \mathbf{v}$ in Line 12.

We will prove the claim in the lemma by induction. Let i_1 be the smallest index for which the column $\mathbf{T}[i_1]$ is not in the column space of $[\mathbf{B} \mid \partial_2]$. Then, setting $\mathbf{v}[i_1]$ to 1 and $\mathbf{v}[j]$ to either 0 or 1 for $j \in \{\ell, \dots, i_1 - 1\}$, we obtain a set of linear combinations of columns in $\mathbf{T}[\ell \dots i_1]$, denoted by S_{i_1}, which do not lie in the column space of $[\mathbf{B} \mid \partial_2]$. Since $|S_{i_1}| = 2^{i_1 - \ell}$, half of the linear combinations of the first $i_1 - \ell$ columns of $\mathbf{T}$ generate a column that does not belong to the column space of $[\mathbf{B} \mid \partial_2]$. This completes the base case of the induction.

For the inductive hypothesis, assume that for some $i > i_1$, at least half of the $2^{i - \ell + 1}$ linear combinations of columns of $\mathbf{T}[\ell \dots i]$ using the first $i - \ell + 1$ indices generate a column that is not in the column space of $[\mathbf{B} \mid \partial_2]$. Denote this set of linear

Algorithm 6 Algorithm for minimum homology basis

1: Compute the set of tight cycles of K_1 in matrix $\mathbf{T}$ sorted by weight.
2: Use Wiedemann's algorithm to compute $\mathrm{rk}(\partial_1)$. Let $z_1 = m - \mathrm{rk}(\partial_1)$. Next, use Wiedemann's algorithm to compute $b_1 = \mathrm{rk}(\partial_2)$. Then, $g' \leftarrow z_1 - b_1$.
3: ▷ Wiedemann's algorithm computes rank correctly with high probability. Hence, $g' = g$ with high probability.
4: Initialize $\mathbf{B}$ with the empty matrix.
5: $\ell \leftarrow 1$; $r \leftarrow m - n + 1$;
6: **for** $i = 1$ to g' **do** ▷ Outer for loop
7: $k \leftarrow 0$; ▷ The variable k is not used in the algorithm; but is used in the analysis.
8: **while** $\ell \leq r$ **do** ▷ The while loop is used for binary search.
9: $p \leftarrow \lfloor \frac{\ell+r}{2} \rfloor$; found $\leftarrow$ `false`;
10: **for** $2 \log m$ times **do** ▷ Probability amplification for loop
11: Let $\mathbf{v}$ be a uniformly random 0-1 vector of size $(p - \ell + 1)$
12: $\mathbf{w} \leftarrow \mathbf{T}[\ell \ldots p] \cdot \mathbf{v}$
13: Use $2 \log m$ independent runs of Wiedemann's algorithm to determine whether $[\mathbf{B} \mid \partial_2] \cdot \mathbf{x} = \mathbf{w}$ has a solution
14: **if** a solution to $[\mathbf{B} \mid \partial_2] \cdot \mathbf{x} = \mathbf{w}$ is not found in any of the runs **then**
15: $r \leftarrow p$; found $\leftarrow$ `true`
16: foundIndex $\leftarrow p$
17: Exit the **for** loop in Lines 10–17
18: **if** found = `false` **then**
19: **if** $\ell = r$ **then**
20: Print "`Algorithm failed.`" ▷ Binary search fails. Cycle not found.
21: **return**;
22: **else**
23: $\ell \leftarrow p + 1$
24: **else** ▷ Binary search successful. Linearly independent cycle found.
25: **if** $\ell = r$ **then**
26: $\ell \leftarrow$ foundIndex $+ 1$
27: $r \leftarrow m - n + 1$
28: $\mathbf{B} \leftarrow [\mathbf{B} \mid \mathbf{w}]$
29: Exit the **while** loop
30: $k \leftarrow k + 1$

combinations by S_i. Denote the complementary set of linear combinations by $\overline{S}_i$. Said differently, $|S_i| \geq 2^{i-\ell}$ and $|S_i \cup \overline{S}_i| = 2^{i-\ell+1}$.

Then, we have two cases: either $\mathbf{T}[i+1]$ is in the column space of $[\mathbf{B} \mid \partial_2]$ or not.

(1.) If it is in the column space of $[\mathbf{B} \mid \partial_2]$, then S_{i+1} is obtained by extending the combinations in S_i by setting $\mathbf{v}[i+1]$ to be either 0 or 1.

(2.) If it is not in the column space of $[\mathbf{B} \mid \partial_2]$, then the combinations in S_{i+1} that generate a column that is not in the column space of $[\mathbf{B} \mid \partial_2]$ are obtained by (a.) extending combinations in $\overline{S}_i$ by setting $\mathbf{v}[i+1] = 1$, (b.) extending all

combinations in S_i by setting $\mathbf{v}[i+1] = 0$, and some combinations in S_i by setting $\mathbf{v}[i+1] = 1$ (if linear independence is preserved).

For every combination in S_{i+1}, the vector $\mathbf{v}[\ell \dots i+1]$ picks a column from $\mathbf{T}$ that is not in the column space of $[\mathbf{B} \mid \partial_2]$. Note that in both cases (1.) and (2.), $|S_{i+1}| \geq 2^{i-\ell+1}$ assuming $|S_i| \geq 2^{i-\ell}$. The claim follows by noting that $\mathbf{v}$ is selected uniformly at random. □

Lemma 15. *If there exists a column vector in $\mathbf{T}[\ell \dots p]$ that is linearly independent of column vectors of $[\mathbf{B} \mid \partial_2]$, then the **if** condition in Line 14 fails to satisfy in each of the $2 \log m$ iterations of the probability amplification **for** loop (of Lines 10-17) with probability at most $\frac{1}{m^2}$.*

Proof. By Lemma 14, the failure probability of one iteration of the probability amplification **for** loop is at most $\frac{1}{2}$. Since the **for** loop of Lines 10-17 is executed $2 \log m$ times, and each time the vector $\mathbf{w}$ is chosen independently, the **if** condition fails to satisfy with probability at most $2^{-2 \log m} = \frac{1}{m^2}$. □

Lemma 16. *If there does not exist a column vector in $\mathbf{T}[\ell \dots p]$ that is linearly independent of column vectors of $[\mathbf{B} \mid \partial_2]$, then the **if** condition in Line 14 is satisfied with probability at most $\frac{1}{m^2}$.*

Proof. By assumption, there does not exist a column vector in $\mathbf{T}[\ell \dots p]$ that is linearly independent of column vectors of $[\mathbf{B} \mid \partial_2]$. Suppose that in one of the iteration of the probability amplification **for** loop (of Lines 10-17) all $2 \log m$ runs of the Wiedemann's algorithm fail to find a solution to $[\mathbf{B} \mid \partial_2] \cdot \mathbf{x} = \mathbf{w}$. The failure probability of one run of Wiedemann's algorithm (Line 13) is at most $\frac{1}{2}$. Hence, the probability that $2 \log m$ runs fail to find a solution even when one exists is at most $2^{-2 \log m} = \frac{1}{m^2}$. □

Let ζ_1 be the lexicographically smallest nontrivial cycle in $\mathbf{T}$, and for every $i \in \{2, \dots, g\}$ let ζ_i be the lexicographically smallest cycle in $\mathbf{T}$ that is linearly independent of cycles ζ_j for $j \in [i-1]$. Let $\mathcal{E}_i$ be the event that $\mathbf{B}[i] = \zeta_i$. Also, let $\mathcal{E}_0$ be the event that $g' = g$.

Lemma 17. $\boldsymbol{Pr}[\mathcal{E}_1 \mid \mathcal{E}_0] \geq (1 - \frac{1}{m^2})^{\lceil \log \nu \rceil}$.

Proof. For $i = 1$, in the outer **for** loop of Lines 6-30, we denote the values of ℓ, p and r in the k-th iteration of the **while** loop by ℓ_k, p_k and r_k, respectively. Note that $\ell_0 = 1$ and $r_0 = m - n + 1$. For $k \in \{2, \dots, \lceil \log \nu \rceil\}$, when $i = 1$ and $k > 1$, if the **if** condition in Line 14 is successful, then $\ell_k = \ell_{k-1}$ and $r_k = p_{k-1}$ (see Line 15), else $\ell_k = p_{k-1} + 1$ (see Line 23) and $r_k = r_{k-1}$. The algorithm uses a modification of binary search to find the lexicographically smallest indexed column of $\mathbf{T}$ that does not lie in the column space of $[\mathbf{B} \mid \partial_2]$ (which is the same as $[\partial_2]$ when $i = 1$).

Assuming $\mathcal{E}_0$ is satisfied, we have $g = g' \geq 1$. Clearly, $\zeta_1 \in \mathbf{T}[\ell_0, r_0]$. Suppose that after $k-1$ iterations, the probability that $\zeta_1 \in \mathbf{T}[\ell_{k-1}, r_{k-1}]$ is at least $(1 - \frac{1}{m^2})^{k-1}$. If $\zeta_1 \in \mathbf{T}[\ell_{k-1}, p_{k-1}]$, then using Lemma 15, $r_k = p_{k-1}$ and $\ell_k = \ell_{k-1}$ with probability at least $1 - \frac{1}{m^2}$. On the other hand, if $\zeta_1 \in \mathbf{T}[p_{k-1} + 1, r_{k-1}]$, then using Lemma 16, $\ell_k = p_{k-1} + 1$ and $r_k = r_{k-1}$ with probability at least $1 - \frac{1}{m^2}$. In either case, the probability that $\zeta_1 \in \mathbf{T}[\ell_k, r_k]$ is at least $(1 - \frac{1}{m^2})^k$.

In every iteration of the **while** loop, the size of the search interval reduces by half. Since the total number of columns in $\mathbf{T}$ is ν, $\mathbf{Pr}[\mathcal{E}_1 \mid \mathcal{E}_0]$ is at least $(1 - \frac{1}{m^2})^{\lceil \log \nu \rceil}$. □

Lemma 18. ***Pr***$[\mathcal{E}_i \mid \cap_{j=0}^{i-1} \mathcal{E}_j] \geq (1 - \frac{1}{m^2})^{\lceil \log \nu \rceil}$.

Proof. The proof is nearly identical to Lemma 17. □

We now recall a useful inequality from Motwani and Raghavan's book [25].

Proposition 19 (Proposition B.3 [25])**.** *For all $t, n \in \mathbb{R}$ such that $n \geq 1$ and $t \leq n$,*

$$e^t \left(1 - \frac{t^2}{n}\right) \leq \left(1 + \frac{t}{n}\right)^n.$$

Theorem 20. *Algorithm 6 correctly computes the minimum homology basis with probability at least $\frac{1}{4} e^{-1} \left(1 - \frac{1}{m^2}\right)$.*

Proof. To begin with, note that Wiedemann's algorithm [32] for computing the rank of a matrix has success probability at least $\frac{1}{2}$. Hence, $\mathbf{Pr}[\mathcal{E}_0] \geq \frac{1}{4}$.

Using $\mathbf{Pr}[\cap_{i=0}^{g'} \mathcal{E}_i] = \mathbf{Pr}[\mathcal{E}_0] \times \mathbf{Pr}[\mathcal{E}_1 \mid \mathcal{E}_0] \times \mathbf{Pr}[\mathcal{E}_2 \mid \mathcal{E}_1 \cap \mathcal{E}_0] \times \cdots \times \mathbf{Pr}[\mathcal{E}_{g'} \mid \cap_{j=0}^{g'-1} \mathcal{E}_j]$ and Lemmas 17 and 18, we deduce that

$$\mathbf{Pr}[\cap_{i=0}^{g'} \mathcal{E}_i] \geq \frac{1}{4} \left(1 - \frac{1}{m^2}\right)^{\lceil \log \nu \rceil g'}.$$

From Proposition 19,

$$e^{-1} \left(1 - \frac{1}{m^2}\right) \leq \left(1 + \frac{(-1)}{m^2}\right)^{m^2} \leq \left(1 - \frac{1}{m^2}\right)^{\lceil \log \nu \rceil g'}$$

Hence, the probability that Algorithm 6 correctly computes the minimum homology basis is given by $\frac{1}{4} \cdot e^{-1} \left(1 - \frac{1}{m^2}\right)$. □

Theorem 21. *Algorithm 6 runs in $\tilde{O}(N^2 g + N m g^2 + m g^3)$ time.*

Proof. The list of tight cycles in G can be computed in $O(nm)$ time using the algorithm described in Section 2 of [2]. Hence, Line 1 of Algorithm 6 takes $O(nm \log(nm)) = O(nm \log n)$ time.

Since ∂_1 and ∂_2 have $O(N)$ nonzero entries, rank computations, from Remark 1, using Wiedemann's algorithm [32] in Line 2 take $\tilde{O}(N^2)$ time.

We note that if $g' > g$, then in the outer **for** loop of Lines 6-30, for $i = g + 1$, the binary search will fail and the **if** condition in Line 19 will be satisfied, and the algorithm will terminate. Therefore, we can state the complexity analysis in terms of g instead of g'.

By Theorem 4, the total length of tight cycles of K_1 is at most $n\nu = O(nm)$. Using a sparse matrix representation of $\mathbf{M}$, Line 12 takes $O(nm)$ time. From Remark 1, we know that a single run of Wiedemann's algorithm takes $\tilde{O}((N + mg) \cdot (N + g)) = \tilde{O}(N^2 + Nmg + mg^2)$ time since $Ng = O(N^2)$. So, $\log m$ runs of Wiedemann's algorithm in Line 13 takes $\tilde{O}((N^2 + Nmg + mg^2) 2 \log m) = \tilde{O}(N^2 + Nmg + mg^2)$ time. The outer **for** loop in Lines 6-30 runs at most $g + 1$ times. The **while** loop in Lines 8-30

runs at most $\log m$ times. The probability amplification **for** loop in Lines 10-17 runs $2\log m$ times. Line 13 is executed at most $O(2g\log^2 m)$ times. Since Line 13 is the most expensive step in the probability amplification **for** loop from Lines 10-17, the complexity of the algorithm is $\tilde{O}(N^2g + Nmg^2 + mg^3)$. □

Note that when $g = O(1)$, Theorem 4 runs in nearly quadratic time.

7 Runtime comparison

Recall that Algorithm 4 runs in $\tilde{O}(m^\omega)$, Algorithm 5 runs in $O(Nm^{\omega-1})$ time and Algorithm 6 runs in $\tilde{O}(N^2g + Nmg^2 + mg^3)$ time, where n is the number of vertices, m is the number of edges and N is the total number of simplices.

Note that Dey et al. prove a bound of $O(N^\omega + N^2g)$ on the running time of their algorithm [13, Section 3.2]. However, a more refined analysis shows that the algorithm described in Dey et al. [13] runs in $O(nmg+Nm^{\omega-1})$ time. This is because the annotation algorithm takes $O(Nm^{\omega-1})$ time in the worst case, whereas the ShortestCycle procedure in [13] takes $O(nm)$ time. Using the recurrence relation in [13, Section 3.2] we obtain a time complexity bound of $O(Nm^{\omega-1} + nmg)$.

Comparison of Algorithm 4 with Algorithm 5

For families of complexes with $N^{1-\epsilon} = \omega(m)$ for some $\epsilon > 0$, Algorithm 4 is faster than Algorithm 5. However, for families of complexes such as triangulations of surfaces with $N = \Theta(m)$, Algorithm 5 is faster than Algorithm 4.

Comparison of Algorithm 4 and Algorithm 5 with Dey et al.'s algorithm

For surfaces with $g = \Theta(m)$, Algorithms 4 and 5 are faster than Dey et al.'s algorithm since Dey et al.'s algorithm takes at least $\Theta(m^{2.5})$ time since $n = \Omega(m^{0.5})$, while Algorithms 4 and 5 run in $\tilde{O}(m^\omega)$ and $O(m^\omega)$ time, respectively.

For dense simplicial complexes with n vertices, $\Theta(n^2)$ edges and $\Theta(n^3)$ 2-simplices, Algorithm 4 is faster than Dey et al.'s algorithm since Algorithm 4 runs in $\tilde{O}(n^{2\omega})$ time, whereas Dey et al.'s algorithm runs in $O(n^{2\omega+1})$ time.

For surfaces with bounded g, Dey et al.'s algorithm is slightly faster than Algorithm 4 since Dey et al.'s algorithm runs in $O(m^\omega)$ time, whereas Algorithm 4 runs in $\tilde{O}(m^\omega)$ time.

Finally, it is easy to check that Algorithm 5 is asymptotically always at least as fast Dey et al.'s algorithm, whereas in some important cases (such as surfaces with $g = \Theta(m)$ as discussed above), it is indeed much faster.

Comparison of Algorithm 6 with Algorithm 4 and Algorithm 5

When $N = O(m)$ and g is bounded, Algorithm 6 is faster than Algorithms 4 and 5. On the other hand when $g = \Theta(m)$, Algorithms 4 and 5 are faster than Algorithm 6.

Comparison of Algorithm 6 with Dey et al.'s algorithm

For surfaces with bounded g, Algorithm 6 is faster than Dey et al's algorithm since Algorithm 6 runs in $O(m^2)$ time whereas Dey et al's algorithm runs in $O(m^\omega)$ time.

On the other hand, for surfaces with $g = \Theta(m)$, Dey et al.'s algorithm is faster than Algorithm 6 since Algorithm 6 runs in $O(m^4)$ time whereas Dey et al.'s algorithm runs in $O(m^\omega)$ time.

8 Implementation

While the algorithms described in the previous section have good worst case runtime bounds, they are not amenable to a simple and efficient implementation. We use ideas from Algorithm 5 to implement an algorithm that exhibits good performance in practice by leveraging existing state-of-the-art software for matrix reduction and minimum cycle basis computation. We begin by describing the data representations for input/output and intermediate storage.

8.1 Input/output format

Geometric simplicial complex input are expected to be in the `OFF` [1] file format. The current implementation assumes that the complex is embedded in $\mathbb{R}^3$ but this may be extended to high dimensional Euclidean spaces. The weight of an edge in the complex is assumed to be given by the Euclidean distance between its end point vertices. A general simplicial complex input may be specified in a simple text file that stores the 2-skeleton of the complex. The text file contains the number of vertices, edges, and triangles, followed by a list of weighted edges and finally a list of triangles of the complex, all in ASCII format. An edge is represented as a 3-tuple $\{i, j, w\}$ in a separate line where i, j are the indices of its end point vertices and w is the edge weight. A triangle is represented as a 3-tuple $\{i, j, k\}$ in a separate line where i, j, k are the indices of the vertices of the triangle.

The output is available in a single text file consisting of the betti number (β_1) followed by a list of cycles that represent a minimum homology basis. Each cycle is represented as a sequence of vertex indices.

8.2 Internal data representation

We maintain the following in-memory data structures for querying the input simplicial complex and storing the results of the intermediate steps of the algorithm.

1. *vertexList:* A list of vertices and their location in $\mathbb{R}^3$ for geometric simplicial complexes.
2. *triangles:* A list of triangles of the complex. Each triangle is stored as a 3-tuple representing the index of its three vertices in *vertexList*.
3. *vPairToE:* Edges of the complex are enumerated. The index *edgeNo* of an edge ranges from 1 to m. *vPairToE* is a map where each key is a pair of vertex indices corresponding to the end points of an edge and the associated value is the *edgeNo* of that edge.
4. *eToVPair:* The inverse map of *vPairToE*. A map that specifies the pair of vertex indices corresponding to an edge.
5. *eToWeightMap:* A map from an edge index to the edge weight.

6. *graph:* The 1-skeleton of the input simplicial complex, stored as a Boost adjacency list [5].

8.3 Algorithm

We now describe FastLoop, a practical implementation of Algorithm 4. FastLoop employs alternate algorithms for computing the minimum cycle basis and column rank profile. These algorithms are efficient in practice, amenable to parallel computation, and their implementation is made available within reliable software libraries. Below, we provide an overview of the main steps of FastLoop.

1. Load the input simplicial complex and populate the key data structures described in the previous section.
2. Compute a minimum cycle basis (see Section 8.3.1).
3. Assemble the matrix $\mathbf{Y}$, the boundary matrix prepended to the minimum cycle basis (Algorithm 5)
4. Employ a column reduction algorithm that adds columns from left to right to compute the column rank profile (see Section 8.3.2).

8.3.1 Computing a minimum cycle basis

We use parmcb, the Parallel Minimum Cycle Basis library [14], to compute a minimum cycle basis of the 1-skeleton of the input simplicial complex. The parmcb library implements a suite of algorithms that are broadly based on De Pina's algorithm Algorithm 1 but differ in the step that computes a minimum weight cycle that is required in Step 3 of the algorithm. It supports parallel execution using MPI and Intel TBB. The input graph is represented using the graph data structure from the Boost library. For our purpose, we build a Boost library adjacency list representation of the 1-skeleton of the 2-complex (*graph*). The minimum cycle basis is reported as a collection of cycles, where each cycle is represented as a list of edges.

8.3.2 Computing the minimum homology basis

Any matrix reduction algorithm that adds columns from left to right can be used to compute the column rank profile. Specifically, the indices of the non-zero columns at the end of such a reduction procedure gives the column rank profile. We use the standard reduction algorithm in the PHAT library [4]. PHAT is a library of matrix reduction algorithms [3] implemented in C++ for computing barcodes in persistent homology. We recall the standard reduction algorithm in Algorithm 7. Theorem 4 implies that the matrix $\mathbf{Y}$ is sparse. This motivates the use of PHAT, which is optimized to exploit the sparsity of the underlying matrix. PHAT supports multiple sparse representations of the matrix. We choose the bit_tree representation [3, Section 4], where each column is represented as a balanced binary tree of row indices that contain a 1 in that column. The matrix is maintained as a list (*vector*) of such balanced binary trees.

Algorithm 7 Standard reduction algorithm for matrix reduction [3]

```
Input: A 0-1 matrix ∂ with m columns.
low(j) denotes the row index of the lowest 1 in ∂, it is undefined if column j contains only zeros.
for j = 1 to m do
    while there exists j_0 < j with low(j_0) = low(j) do
        Add column j_0 to j
```

9 Experimental results

We now describe results from our computational experiments on various real world and synthetic datasets. The experiments serve two primary purposes. First, they validate the correctness of FastLoop. Second, they reveal the efficiency of the algorithm when measured against the size and type of the input complex and the number of CPU cores deployed. All experiments were performed on an Intel workstation powered by a Xeon(R) Gold 6230 CPU with 20 cores at 2.10 GHz and 384 GB RAM running Ubuntu Linux. Parallelization in computing the minimum cycle basis was achieved using Intel Thread Building Blocks (TBB).

9.1 Cycle representatives

The real world datasets include 2D meshes and 3D volume meshes, both manifold and non-manifold. Table 1 lists all datasets and Figure 3 shows results on a subset of the datasets. The top two rows in Figure 3 are polygonal meshes from the Visionair shape repository [31]. These meshes are medium sized datasets consisting of 30000–70000 simplices. The accompanying video (Online Resource 1) shows the computed cycle representatives from different view points.

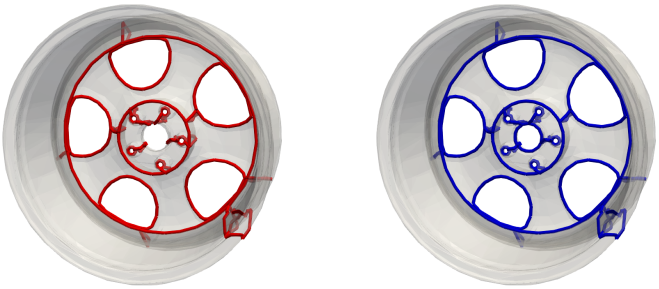
Computational experiments on the real world datasets help demonstrate the practical utility of the algorithm via direct visualization of the loops and tunnels. Our algorithm computes optimal cycle representatives for holes of different sizes. The third and fourth rows in Figure 3 show data from the PCOD hypothetical zeolite database [11]. Zeolite structures are known to contain pores. Hypothetical structures are generated computationally and their properties are studied to determine if they are similar to existing zeolites with desirable properties. A distance field that captures distance from a point to its nearest atom is computed for each hypothetical zeolite structure using the Zeo++ software [23]. The structure of three hypothetical zeolite materials are visualized by rendering the zero isosurface, the preimage of distance value 0. Zeolites are known to contain pores. Identifying the pores and quantifying their size is an important problem because the pore size determines the chemical properties of the zeolite. Zeolites also have a spatially repeating structure that contributes to a larger value of β_1.

Figure 4 shows the surface of two protein molecules, 3EAM and 1OED. Both proteins contain a central tunnel and multiple long pores. A subset of the cycles of the minimum 1-homology basis, including the central tunnel, are highlighted in middle

column. All the cycles of the minimum 1-homology basis are highlighted in the right column.

9.2 Verifying correctness

Notwithstanding the theoretical correctness of the algorithm, we undertake several measures to ensure correctness of the implementation in FastLoop. The two major components of FastLoop are based on highly optimized, well maintained, stable, and well tested software. Specifically, the library parmcb [14] is used for computing the minimum cycle basis (MCB), and PHAT [4] is used for the reduction step to compute the minimum homology basis (MHB) from the MCB. We also compare the output of FastLoop and ShortLoop [12, 26] on the real world datasets. The Betti numbers reported by both software match each other, and for a majority of the datasets the MHB reported by both software also match. However, we found a few examples where the weight of the MHB computed by FastLoop differed from that reported by ShortLoop. In order to explain this discrepancy, we performed a few sanity checks on the outputs of Shortloop and FastLoop. In particular, we checked if the loops reported by the software are indeed non-bounding and independent. We observed that a few cycles reported by ShortLoop fail the independence check, specifically in the cases when ShortLoop reports a smaller weight basis when compared to FastLoop. This discrepancy can also be detected visually in some instances. Figure 2 shows an example where the collection of basis cycles reported by ShortLoop (red) misses the central hole of the wheel. The developers of ShortLoop have been informed about this issue.



ShortLoop FastLoop

Fig. 2 ShortLoop misses the central hole (left) whereas FastLoop correctly identifies the cycle representing the hole.

9.3 Synthetic data

We also present results of experiments on synthetic datasets, consisting of two classes of random complexes, with the aim of studying the scaling behavior of our algorithm. The first class is random clique complexes, a well-studied class of random complexes. Let n be the number of vertices in the complex and let p be a probability parameter.

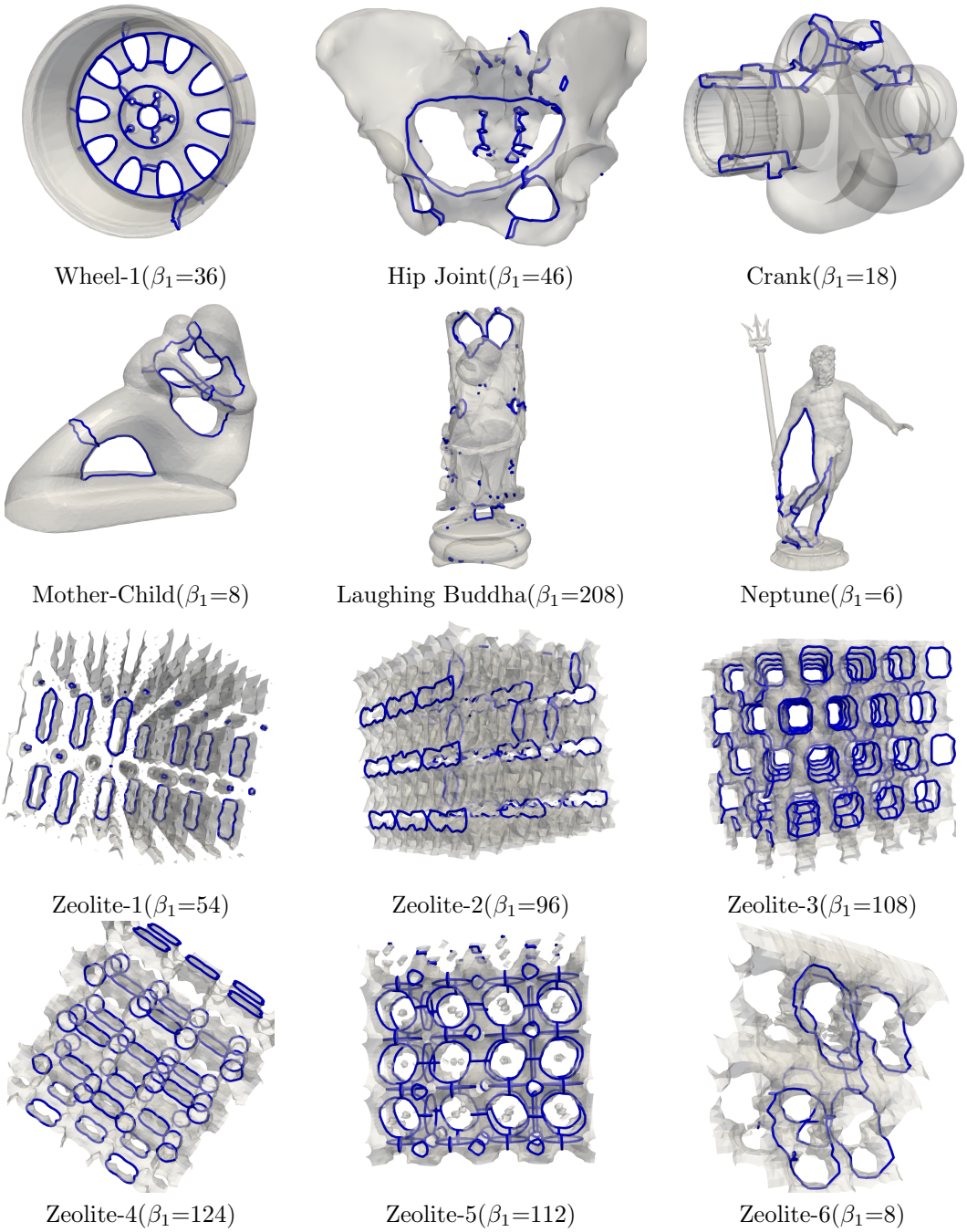


Fig. 3 Minimum 1-homology basis (blue) computed on various 2D and 3D datasets.

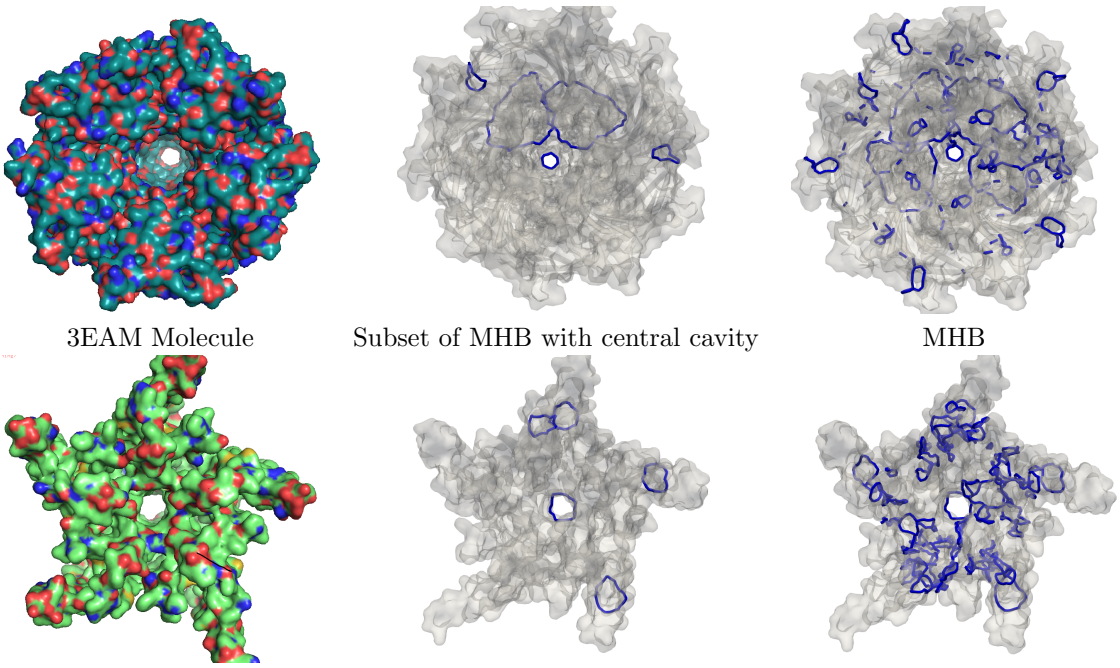


Fig. 4 Minimum homology basis computed for two membrane proteins. Left: A rendering of the molecular surface. Right: The minimum homology basis. Middle: A subset of cycles in the basis that represent the central tunnel and some of the larger pores in the protein.

A two dimensional clique complex $C(n,p)$ is constructed as follows: an edge $\{i,j\}$ is chosen in $C(n,p)$ with probability p, and a 2-simplex is included in $C(n,p)$ if all of its edges belong to $C(n,p)$.

A second class of 2-complexes are random triangle complexes $R(n,p)$, which are constructed as follows. Let n be the number of vertices and p be a probability parameter. A triangle $\{i,j,k\}$ is included in $R(n,p)$ with probability p. When the triangle $\{i,j,k\}$ is included, all edges on its boundary are also included in $R(n,p)$.

9.4 Runtime and scaling

Table 1 reports runtimes for both real world and synthetic data. We note that, in general, the runtimes are short for medium sized datasets. The table also presents a comparison against ShortLoop. We observe a significant improvement in execution time over ShortLoop, with a speedup of 1-2 orders of magnitude in some cases. ShortLoop does not terminate within a reasonable time frame (1 day) for the synthetic data. So, we omit a comparison on these random complexes.

The synthetic datasets help study the scaling behavior of FastLoop. We choose three probability parameters for each class of random complexes, $(0.025, 0.05, 0.075)$ for random clique complexes and $(5 \times 10^{-}5, 10^{-4}, 2 \times 10^{-4})$ for random triangle complexes. We fix value of the probability parameter p and study the runtime performance

against increasing size of the complex. Figure 5 shows a log-log plot of the total running time against number of simplices in the input complex. Figure 6 shows a log-log plot of the total running time against the number of edges in the complex. Both plots are linear indicating that the runtime is a power function of the form ax^b. We fit a straight line to the points on the plot and compute its slope in order to estimate the exponent b of the power function.

The value of the exponent for the plot of runtime vs. total simplices in Figure 5 is at most 1.8 for clique complexes and at most 2 for random triangles complexes. The exponent is at most 2 for both classes of complexes in the plot of runtime vs. number of edges. The empirically observed complexity is better than the theoretical worst case runtime complexity of Algorithm 5. Figure 8 shows a plot of MCB size, the cardinality of the multi-set of edges in the minimum cycle basis. The MCB is the output of the first step of the algorithm. The scaling behavior of the MCB size is indicative of the rate determining step of the algorithm. Indeed, a comparison between Figure 7 and Figure 5 shows that the time taken for computing the MCB is several orders of magnitude greater than the subsequent steps, and that it constitutes a large fraction of the overall runtime.

dataset	#Vertices	#Edges	#Triangles	Time (FastLoop)	Time (ShortLoop)
Wheel-1	6970	21000	14000	35s	245s
Wheel-2	2476	7500	5000	7s	80s
Genus3	2462	7500	5000	7s	84s
Mother-Child	6494	19500	13000	34s	40s
Neptune	6246	18750	12500	30s	30s
Zeolite-1	17275	46084	30000	125s	80s
Zeolite-2	13410	38200	25000	90s	$>$ 2 hr
Zeolite-3	12962	38116	24999	100s	$>$ 2 hr
Zeolite-4	7629	21741	14128	300s	$>$ 2 hr
Zeolite-5	24643	72075	47435	360s	$>$ 2 hr
Zeolite-6	24573	71587	46963	300s	$>$ 2 hr
Protein-1(3EAM)	30374	90163	60000	480s	$>$ 2 hr
Protein-2(1OED)	29846	89949	59999	540s	$>$ 2 hr

Table 1 Runtime analysis. FastLoop outperforms ShortLoop in terms of total runtime with a speedup of 10× or more in many instances.

10 Discussion

In this paper, we show that questions about minimum cycle basis and minimum homology basis can be naturally recast into the problem of computing rank profiles of matrices, leading to fast algorithms with simple and elegant high-level descriptions. The column rank profile (or the earliest basis) of a matrix has previously been used to compute the minimum homology basis of a simplicial complex [7, 13]. Such a greedy approach that picks, at each step, an independent cycle of the smallest index, works because of the matroid structure of homology bases and cycle bases. The novelty of

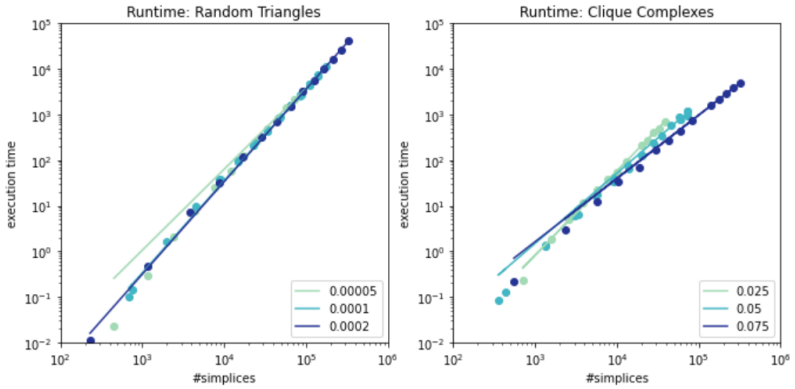


Fig. 5 Scaling study. A log-log plot of running time vs. total number of simplices in the input complex. The running time scales at most quadratically with the number of simplices.

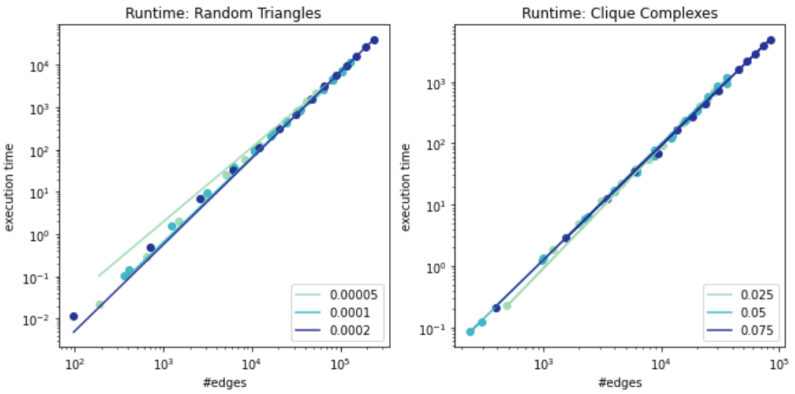


Fig. 6 Scaling study. A log-log plot of running time vs. number of edges in the input complex. The running time scales quadratically.

our approach is the observation that independence can be efficiently checked owing to the sparsity of the matrices comprising of candidate cycles.

In Section 6, we describe a randomized g-sensitive algorithm for computing minimum homology basis that runs in nearly quadratic time when $g = O(1)$. We believe this is the first such algorithm for this problem for general complexes.

Experiments on real-world data sets reveal how FastLoop captures the one dimensional "holes" that may be useful in a variety of practical applications. FastLoop computes the minimum homology basis of a variety of medium to large sized real-world data sets within a few minutes and consistently outperforms the state of the art implementation, ShortLoop. The algorithm as well as the software consist of two major components, namely computing a minimum cycle basis followed by a reduction step. The two components are based on independent algorithms, which may be replaced in the future with alternate methods to achieve better theoretical complexity or practical running times.

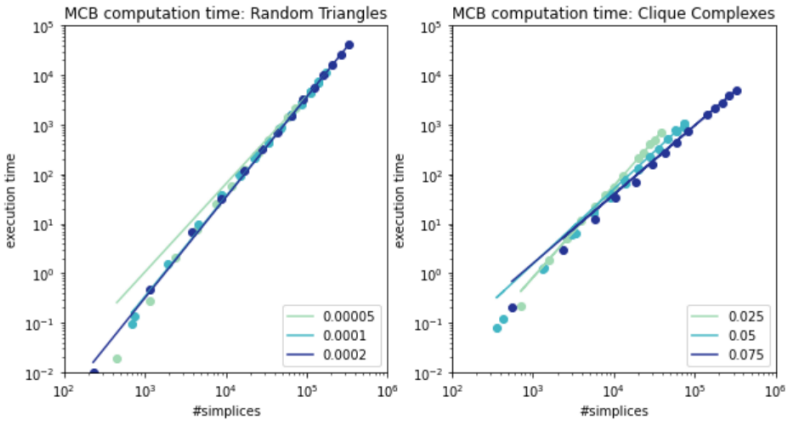


Fig. 7 A log-log plot of the time taken to compute the MCB vs. number of simplices in the input indicates that this step scales quadratically with the input.

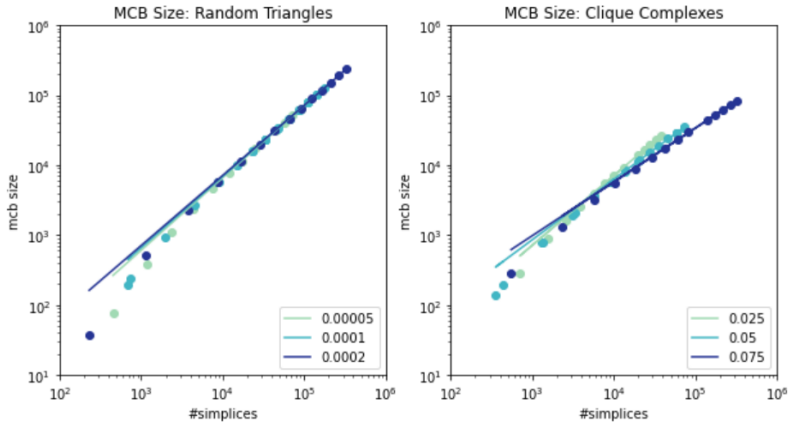


Fig. 8 The number of edges in the MCB increases quadratically with the size of the input, which explains why computing the MCB is the rate determining step.

Supplementary information. The accompanying video (Online Resource 1) shows the computed cycle representatives from different view points.

Acknowledgements. This work is partially supported by the PMRF, MoE Govt. of India, a DFG Project SFB/TRR 109 "Discretization in Geometry and Dynamics", an NSF grant CCF 2049010, and a SERB grant CRG/2021/005278. VN acknowledges support from the Alexander von Humboldt Foundation, and Berlin MATH+ under the Visiting Scholar program. Part of this work was completed when VN was a guest Professor at the Zuse Institute Berlin.

References

[1] (2024) OFF format. http://www.geomview.org/docs/html/OFF.html, [Online; accessed 14-May-2024]

[2] Amaldi E, Iuliano C, Jurkiewicz T, et al (2009) Breaking the $O(m^2n)$ barrier for minimum cycle bases. In: Fiat A, Sanders P (eds) Algorithms - ESA 2009. Springer Berlin Heidelberg, Berlin, Heidelberg, pp 301–312

[3] Bauer U, Kerber M, Reininghaus J, et al (2017) PHAT – persistent homology algorithms toolbox. Journal of Symbolic Computation 78:76–90. https://doi.org/10.1016/j.jsc.2016.03.008, URL http://dx.doi.org/10.1016/j.jsc.2016.03.008

[4] Bauer U, Kerber M, Reininghaus J (2024) Persistent homology algorithm toolbox. URL https://github.com/blazs/phat, [Online; accessed 14-May-2024]

[5] Boost (2023) Boost library. https://boost.org, [Online; accessed 18-April-2023]

[6] Borradaile G, Chambers EW, Fox K, et al (2017) Minimum cycle and homology bases of surface-embedded graphs. JoCG 8(2):58–79

[7] Busaryev O, Cabello S, Chen C, et al (2012) Annotating simplices with a homology basis and its applications. In: Fomin FV, Kaski P (eds) Algorithm Theory – SWAT 2012. Springer Berlin Heidelberg, Berlin, Heidelberg, pp 189–200

[8] Chen C, Freedman D (2010) Measuring and computing natural generators for homology groups. Comput Geom Theory Appl 43(2):169–181. https://doi.org/10.1016/j.comgeo.2009.06.004

[9] Chen C, Freedman D (2011) Hardness results for homology localization. Discrete & Computational Geometry 45(3):425–448. https://doi.org/10.1007/s00454-010-9322-8

[10] Cheung HY, Kwok TC, Lau LC (2013) Fast matrix rank algorithms and applications. J ACM 60(5):31:1–31:25. https://doi.org/10.1145/2528404

[11] Deem M (2020) Michael Deem's PCOD and PCOD2 databases of zeolitic structures. https://doi.org/10.5281/zenodo.4030232

[12] Dey TK, Sun J, Wang Y (2010) Approximating loops in a shortest homology basis from point data. In: Proc. Twenty-Sixth Annual Symposium on Computational Geometry. Association for Computing Machinery, New York, NY, USA, SoCG '10, p 166–175, https://doi.org/10.1145/1810959.1810989

[13] Dey TK, Li T, Wang Y (2018) Efficient algorithms for computing a minimal homology basis. In: Bender MA, Farach-Colton M, Mosteiro MA (eds) LATIN 2018: Theoretical Informatics. Springer International Publishing, Cham, pp 376–398

[14] Dimitrios M (2024) Parallel minimum cycle basis library. URL https://github.com/d-michail/parmcb, [Online; accessed 14-May-2024]

[15] Dumas JG, Pernet C, Sultan Z (2013) Simultaneous computation of the row and column rank profiles. In: Proc. 38th International Symposium on Symbolic and Algebraic Computation. ACM, New York, NY, USA, ISSAC '13, pp 181–188, https://doi.org/10.1145/2465506.2465517

[16] Erickson J, Whittlesey K (2005) Greedy optimal homotopy and homology generators. In: Proc. Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, SODA '05, pp 1038–1046

[17] Hatcher A (2002) Algebraic topology. Cambridge University Press, Cambridge

[18] Horton JD (1987) A polynomial-time algorithm to find the shortest cycle basis of a graph. SIAM J Comput 16(2):358–366. https://doi.org/10.1137/0216026

[19] Jeannerod CP, Pernet C, Storjohann A (2013) Rank-profile revealing gaussian elimination and the cup matrix decomposition. Journal of Symbolic Computation 56:46 – 68. https://doi.org/https://doi.org/10.1016/j.jsc.2013.04.004

[20] Kavitha T, Mehlhorn K, Michail D, et al (2004) A faster algorithm for minimum cycle basis of graphs. In: Díaz J, Karhumäki J, Lepistö A, et al (eds) Automata, Languages and Programming. Springer Berlin Heidelberg, Berlin, Heidelberg, pp 846–857

[21] Kavitha T, Liebchen C, Mehlhorn K, et al (2009) Cycle bases in graphs characterization, algorithms, complexity, and applications. Computer Science Review 3(4):199 – 243. https://doi.org/https://doi.org/10.1016/j.cosrev.2009.08.001

[22] Kavitha T, Mehlhorn K, Michail D (2011) New approximation algorithms for minimum cycle bases of graphs. Algorithmica 59(4):471–488. https://doi.org/10.1007/s00453-009-9313-4

[23] Martin RL, Smit B, Haranczyk M (2012) Addressing challenges of identifying geometrically diverse sets of crystalline porous materials. Journal of Chemical Information and Modeling 52(2):308–318. https://doi.org/10.1021/ci200386x, pMID: 22098053

[24] Mehlhorn K, Michail D (2009) Minimum cycle bases: Faster and simpler. ACM Trans Algorithms 6(1):8:1–8:13. https://doi.org/10.1145/1644015.1644023

[25] Motwani R, Raghavan P (1995) Randomized Algorithms. Cambridge University Press, Cambridge

[26] Oleksiy B, Tamal DK, Jian S, et al (2024) Shortloop software for computing loops in a shortest homology basis. https://web.cse.ohio-state.edu/~dey.8/shortloop.html, [Online; accessed 14-May-2024]

[27] de Pina JC (1995) Applications of shortest path methods. PhD thesis, Universiteit van Amsterdam

[28] Rathod A (2020) Fast algorithms for minimum cycle basis and minimum homology basis. In: Proc. International Symposium on Computational Geometry (SoCG 2020), pp 64:1–64:11

[29] Storjohann A, Yang S (2014) Linear independence oracles and applications to rectangular and low rank linear systems. In: Proc. 39th International Symposium on Symbolic and Algebraic Computation. ACM, New York, NY, USA, ISSAC '14, pp 381–388, https://doi.org/10.1145/2608628.2608673

[30] Storjohann A, Yang S (2015) A relaxed algorithm for online matrix inversion. In: Proc. 2015 ACM on International Symposium on Symbolic and Algebraic Computation. ACM, New York, NY, USA, ISSAC '15, pp 339–346, https://doi.org/10.1145/2755996.2756672

[31] Visionair (2024) URL http://visionair.ge.imati.cnr.it/, [Online; accessed 14-May-2024]

[32] Wiedemann D (1986) Solving sparse linear equations over finite fields. IEEE Transactions on Information Theory 32(1):54–62. https://doi.org/10.1109/TIT.1986.1057137

[33] Yang S (2014) Algorithms for fast linear system solving and rank profile computation. Master's thesis, University of Waterloo