frontiers in Artificial Intelligence

ORIGINAL RESEARCH
published: 11 October 2021
doi: 10.3389/frai.2021.681117

Minimal Cycle Representatives in Persistent Homology Using Linear Programming: An Empirical Study With User's Guide

Lu Li[1], *Connor Thompson*[2], *Gregory Henselman-Petrusek*[3], *Chad Giusti*[4] *and Lori Ziegelmeier*[1]*

[1]*Mathematics, Statistics, and Computer Science Department, Macalester College, Saint Paul, MN, United States,* [2]*Department of Mathematics, Purdue University, West Lafayette, IN, United States,* [3]*Mathematical Institute, University of Oxford, Oxford, United Kingdom,* [4]*Department of Mathematical Sciences, University of Delaware, Newark, DE, United States*

OPEN ACCESS

Edited by:
Kathryn Hess,
École Polytechnique Fédérale de Lausanne, Switzerland

Reviewed by:
Gard Spreemann,
Telenor Research, Norway
Henri Riihimäki,
University of Aberdeen, United Kingdom

***Correspondence:**
Lori Ziegelmeier
lziegel1@macalester.edu

Specialty section:
This article was submitted to Machine Learning and Artificial Intelligence, a section of the journal Frontiers in Artificial Intelligence

Received: *15 March 2021*
Accepted: *14 May 2021*
Published: *11 October 2021*

Citation:
Li L, Thompson C, Henselman-Petrusek G, Giusti C and Ziegelmeier L (2021) Minimal Cycle Representatives in Persistent Homology Using Linear Programming: An Empirical Study With User's Guide. Front. Artif. Intell. 4:681117. doi: 10.3389/frai.2021.681117

Cycle representatives of persistent homology classes can be used to provide descriptions of topological features in data. However, the non-uniqueness of these representatives creates ambiguity and can lead to many different interpretations of the same set of classes. One approach to solving this problem is to optimize the choice of representative against some measure that is meaningful in the context of the data. In this work, we provide a study of the effectiveness and computational cost of several ℓ_1 minimization optimization procedures for constructing homological cycle bases for persistent homology with rational coefficients in dimension one, including uniform-weighted and length-weighted edge-loss algorithms as well as uniform-weighted and area-weighted triangle-loss algorithms. We conduct these optimizations via standard linear programming methods, applying general-purpose solvers to optimize over column bases of simplicial boundary matrices. Our key findings are: 1) optimization is effective in reducing the size of cycle representatives, though the extent of the reduction varies according to the dimension and distribution of the underlying data, 2) the computational cost of optimizing a basis of cycle representatives exceeds the cost of computing such a basis, in most data sets we consider, 3) the choice of linear solvers matters a lot to the computation time of optimizing cycles, 4) the computation time of solving an integer program is not significantly longer than the computation time of solving a linear program for most of the cycle representatives, using the Gurobi linear solver, 5) strikingly, whether requiring integer solutions or not, we almost always obtain a solution with the same cost and almost all solutions found have entries in {-1, 0, 1} and therefore, are also solutions to a restricted ℓ_0 optimization problem, and 6) we obtain qualitatively different results for generators in Erdős-Rényi random clique complexes than in real-world and synthetic point cloud data.

Keywords: topological data analysis, computational persistent homology, minimal cycle representatives, generators, linear programming, l_1 and l_0 minimization

1 INTRODUCTION

Topological data analysis (TDA) uncovers mesoscale structure in data by quantifying its shape using methods from algebraic topology. Topological features have proven effective when characterizing complex data, as they are qualitative, independent of choice of coordinates, and robust to some choices of metrics and moderate quantities of noise (Carlsson, 2009; Ghrist, 2014). As such, topological features extracted from data have recently drawn attention from researchers in various fields including, for example, neuroscience (Bendich et al., 2016; Giusti et al., 2016; Sizemore et al., 2019), computer graphics (Singh et al., 2007; Brüel-Gabrielsson et al., 2020), robotics (Vasudevan et al., 2011; Bhattacharya et al., 2015), and computational biology (Bhaskar et al., 2019; Ulmer et al., 2019; McGuirl et al., 2020) [including the study of protein structure (Xia and Wei, 2014; Kovacev-Nikolic et al., 2016; Xia et al., 2018)].

The primary tool in TDA is persistent homology (PH) (Ghrist, 2008), which describes how topological features of data, colloquially referred to as "holes," evolve as one varies a real-valued parameter. Each hole comes with a geometric notion of dimension which describes the shape that encloses the hole: connected components in dimension zero, loops in dimension one, shells in dimension two, and so on. From a parameterized topological space $X = (X_t)_{t \in S \subset \mathbb{R}_{\geq 0}}$, for each dimension n, PH produces a collection $\text{Barcode}_n(X)$ of lifetime intervals $\mathcal{L}$ which encode for each topological feature the parameter values of its birth, when it first appears, and death, when it no longer remains.

A basic problem in the practical application of PH is interpretability: given an interval $\mathcal{L} \in \text{Barcode}_n(X)$, how do we understand it in terms of the underlying data? A reasonable approach would be to find an element of the homology class, also known as a cycle representative, that witnesses structure in the data that has meaning to the investigator. In the context of geometric data, this takes the form of an "inverse problem," constructing geometric structures corresponding to each persistent interval in the original input data. For example, a representative for an interval $\mathcal{L} \in \text{Barcode}_1(X)$ consists of a closed curve or linear combination of closed curves which enclose a set of holes across the family of spaces $(X_t)_{t \in \mathcal{L} \subset S}$. Cycle representatives are used in Emmett et al. (2015) to annotate particular loops as chromatin interactions, and Wu et al. (2017) uses cycle representatives to study and locate and reconstruct fine muscle columns in cardiac trabeculae restoration.

An important challenge, however, is that cycle representatives are not uniquely defined. For example, in the left-hand image in **Figure 1** adapted from Carlsson (2009), two curves enclose the same topological feature and thus, represent the same persistent homology class. We often want to find a cycle that captures not only the existence but also information about the location and shape of the hole that the homology class has detected. This often means optimizing an application-dependent property using the underlying data, e.g. finding a minimal length or bounding area/volume using an appropriate metric. The algorithmic problem of selecting such optimal representatives is currently an active area of research (Chen and Freedman, 2010a; Dey et al., 2011; Wu et al., 2017; Obayashi, 2018; Dey et al., 2019).

There are diverse notions of optimality we may wish to consider in a given context, and which may have significant impact on the effectiveness or suitability of optimization, including.

- weight assignment to chains (uniform vs. length or area weighted),
- choice of loss function (ℓ_0 vs. ℓ_1),
- formulation of the optimization problem (cycle size vs. bounded area or volume), and
- restrictions on allowable coefficients (rational, integral, or $\{0, 1, -1\}$).

Each has a unique set of advantages and disadvantages. For example, optimization using the ℓ_0 norm with $\{0, 1, -1\}$ coefficients is thought to yield the most interpretable results, but ℓ_0 optimization is NP-hard, in general (Chen and Freedman, 2010b). The problem of finding ℓ_1 optimal cycles with rational coefficients, can be formulated as a more tractable linear programming problem. While some literature exists to inform this choice (Dey et al., 2011; Escolar and Hiraoka, 2016; Obayashi, 2018), questions of basic importance remain, including:

Q1 How do the computational costs of the various optimization techniques compare? How much do these costs depend on the choice of a particular linear solver?

Q2 What are the statistical properties of optimal cycle representatives? For example, how often does the support of a representative form a single loop in the underlying graph? And, how much do optimized cycles coming out of an optimization pipeline differ from the representative that went in?

Q3 To what extent does choice of technique matter? For example, how often does the length of a length-weighted optimal cycle match the length of a uniform-weighted optimal cycle? And, how often are ℓ_1 optimal representatives ℓ_0 optimal?

Given the conceptual and computational complexity of these problems [see Chen and Freedman (2010b)], the authors expect that formal answers are unlikely to be available in the near future. However, even where theoretical results are available, strong empirical trends may suggest different or even contrary principles to the practitioner. For example, while the persistence calculation is known to have matrix multiplication time complexity (Milosavljević et al., 2011), in practice the computation runs almost always in linear time. Therefore, the authors believe that a careful empirical exploration of questions one to three will be of substantial value.

In this paper, we undertake such an exploration in the context of one-dimensional persistent homology over the field of rationals, $\mathbb{Q}$. We focus on linear programming (LP) and mixed-integer programming (MIP) approaches due to their ease of use, flexibility, and adaptability. In doing so, we present a new treatment of parameter-dependence (vis-a-vis selection of simplex-wise refinements) relevant to common

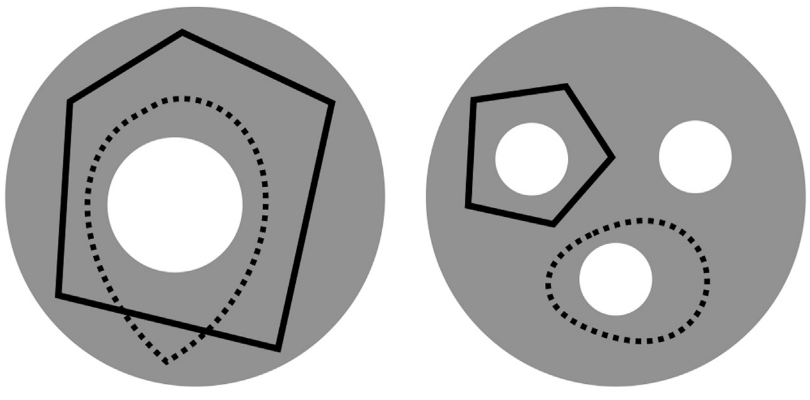

FIGURE 1 | Two disks (gray) — which we regard as 2-dimensional simplicial complexes, though the explicit decomposition into simplices is not shown—with different numbers of holes (white) and cycle representatives (black solid or dotted) adapted from (Carlsson, 2009). The disk on the left has a single 2-dimensional "hole" (β_1 = 1), and the two loops around it are cycle representatives for the same homology class. Similarly, the disk on the right has three "holes" (β_1 = 3) and the two loops shown are cycle representatives for different homology classes.

cases of rational cycle representative optimization (Escolar and Hiraoka, 2016; Obayashi, 2018), such as finding optimal cycle bases for the persistent homology of the Vietoris-Rips complex of a point cloud. We restrict our attention to one-dimensional homology to limit the number of reported statistics and data visualizations presented, although the methods discussed could be applied to any homological dimension.

The paper is organized as follows. **Section 2** provides an overview of some key concepts in TDA to inform a reader new to algebraic topology and establish notation. Then, we provide a survey of previous work on finding optimal persistent cycle representatives in **Section 3**, and formulate the methods used in this paper to find different notions of minimal cycle representatives via LP and MIP in **Section 4**. **Section 5** describes our experiments, including overviews of the data and the hardware and software we use for our analysis. In **Section 6**, we discuss the results of our experiments. We conclude and describe possible future work in **Section 7**.

2 BACKGROUND: TOPOLOGICAL DATA ANALYSIS AND PERSISTENT HOMOLOGY

In this section, we introduce key terms in algebraic and computational topology to provide minimal background and establish notation. For a more thorough introduction see, for example, Hatcher et al. (2002), Ghrist (2008), Edelsbrunner and Harer (2008), Carlsson (2009), Edelsbrunner and Harer (2010), and Topaz et al. (2015).

Given a discrete set of sample data, we approximate the topological space underlying the data by constructing a *simplicial complex*. This construction expresses the structure as a union of vertices, edges, triangles, tetrahedrons, and higher dimensional analogues (Carlsson, 2009).

2.1 Simplicial Complexes

A simplicial complex is a collection K of non-empty subsets of a finite set V. The elements of V are called vertices of K, and the elements of K are called simplices. A simplicial complex has the following properties: 1) $\{v\}$ in K for all $v \in V$, and 2) $\tau \subset \sigma$ and $\sigma \in K$ guarantees that $\tau \in K$.

Additionally, we say that a simplex has dimension n or is an n-simplex if it has cardinality $n + 1$. We use $\mathbf{S}_n(K)$ to denote the collection of n-simplices contained in K.

While there are a variety of approaches to create a simplicial complex from data, our examples use a standard construction for approximation of point clouds. Given a metric space X with metric d and real number $\varepsilon \geq 0$, the Vietoris-Rips complex for X, denoted by $\mathbf{VR}_\varepsilon(X)$, is defined as

$$\mathbf{VR}_\varepsilon(X) = \{\sigma \in \mathbf{S}_n(K) \big| d(x, y) \leq \varepsilon \text{ for all } x, y \in \sigma\}.$$

That is, given a set of discrete points X and a metric d, we build a **VR** complex at scale ε by forming an n-simplex if and only if $n + 1$ points in X are pairwise within ε distance of each other.

2.2 Chains and Chain Complexes

Given a simplicial complex K and an abelian group G, the group of n-chains in K with coefficients in G is defined as

$$\mathbf{C}_n(K; G) := G^{\mathbf{S}_n(K)}.$$

Formally, we regard $G^{\mathbf{S}_n(K)}$ as a group of functions $\mathbf{S}_n(K) \to G$ under element-wise addition. Alternatively, we may view $\mathbf{C}_n(K; G)$ as a group of formal G-linear combinations of n-simplices, i.e., $\{\sum_\sigma x_\sigma \sigma | x_\sigma \in G \text{ and } \sigma \in \mathbf{S}_n(K)\}$.

Remark 2.1. We will focus on the cases where G is $\mathbb{Q}$ (the field of rationals), $\mathbb{Z}$ (the group of integers), or $\mathbb{F}_2$ (the 2-element field). Since we are most interested in the case $G = \mathbb{Q}$, we adopt the shorthand $\mathbf{C}_n(K) = \mathbf{C}_n(K; \mathbb{Q})$.

An element $\mathbf{x} = (x_\sigma)_{\sigma \in \mathbf{S}_n(K)} \in G^{\mathbf{S}_n(K)}$ is called an n-chain of K. As in this example, see we will generally use a bold-face symbol for the tuple $\mathbf{x}$ and corresponding light-face symbols for entries x_σ. The support of an n-chain is the set of simplices on which x_σ is nonzero:

$$\mathrm{supp}(\mathbf{x}) := \{\sigma \in \mathbf{S}_n(K) \mid x_\sigma \neq 0\}.$$

The ℓ_0 norm[1] and ℓ_1 norm[2] of $\mathbf{x}$ are defined as

$$\|\mathbf{x}\|_0 := |\mathrm{supp}(\mathbf{x})| \qquad \|\mathbf{x}\|_1 := \sum_{\sigma \in \mathbf{S}_n(K)} |x_\sigma|.$$

Remark 2.2 (Indexing conventions for chains and simplices). As chains play a central role in our discussion, it will be useful to establish some special conventions to describe them. These conventions depend on the availability of certain linear orders, either on the set of vertices or the set of simplices.

Case 1: Vertex set V has a linear order $\leq$. Every vertex set V discussed in this text will be assigned a (possibly arbitrary) linear order. Without risk of ambiguity, we may therefore write

$$(v_0, \ldots, v_n)$$

for the n-chain that places a coefficient of 1 on $\sigma = \{v_0 \leq \ldots \leq v_n\}$ and 0 on all other simplices.

Case 2: Simplex set $\mathbf{S}_n(K)$ has a linear order $\leq$. We will sometimes define a linear order on $\mathbf{S}_n(K)$. This determines a unique bijection $\sigma^{(n)} : \{1, \ldots, |\mathbf{S}_n(K)|\} \to \mathbf{S}_n(K)$ such that $\sigma_i^{(n)} \leq \sigma_j^{(n)}$ iff $i \leq j$. This bijection determines an isomorphism.

$$\phi : \mathbf{C}_n(K; G) = G^{\mathbf{S}_n(K)} \to G^{|\mathbf{S}_n(K)|}.$$

such that $\phi(\mathbf{x})_i = x_{\sigma_i}$ for all i. Provided a linear order $\leq$, we will use $\mathbf{x}$ to denote both $\mathbf{x}$ and $\phi(\mathbf{x})$ and rely on context to clarify the intended meaning.

For each $n \geq 1$, the boundary map $\partial_n : \mathbf{C}_n(K) \to \mathbf{C}_{n-1}(K)$ is the linear transformation defined on a basis vector $(v_0, v_1, \ldots, v_n)$ by

$$\partial_n (v_0, v_1, \ldots, v_n) = \sum_{i=0}^{n} (-1)^i (v_0, \ldots, \widehat{v_i}, \ldots, v_n),$$

where $\widehat{v_i}$ omits v_i from the vector. This map extends linearly from the basis of n-simplices to any n-chain in $\mathbf{C}_n(K)$. By an abuse of notation, we also denote the matrix representation of this boundary map, known as the boundary matrix, as ∂_n. The boundary matrix is parametrized by the n-simplices $S_n(K)$ along the columns and n - 1 simplices $S_{n-1}(K)$ along the rows.

The collection $(\mathbf{C}_n(K))_{n \geq 0}$ along with the boundary maps $(\partial_n)_{n \geq 0}$ form a chain complex

$$\ldots \mathbf{C}_{n+1}(K) \xrightarrow{\partial_{n+1}} \mathbf{C}_n(K) \xrightarrow{\partial_n} \mathbf{C}_{n-1}(K) \xrightarrow{\partial_{n-1}} \ldots \xrightarrow{\partial_3} \mathbf{C}_2(K) \xrightarrow{\partial_2} \mathbf{C}_1(K) \xrightarrow{\partial_1} \mathbf{C}_0(K) \xrightarrow{\partial_0} 0.$$

Remark 2.3 (Indexing conventions for boundary matrices). In general, boundary matrix ∂_n is regarded as an element of $G^{\mathbf{S}_{n-1}(K) \times \mathbf{S}_n(K)}$, that is, as an array with columns labeled by n-simplices and rows labeled by n - 1 simplices. However, given linear orders on $\mathbf{S}_{n-1}(K)$ and $\mathbf{S}_n(K)$, we may naturally regard ∂_n as an element of $G^{|\mathbf{S}_{n-1}(K)| \times |\mathbf{S}_n(K)|}$, see **Remark 2.2.**

2.3 Cycles, Boundaries

The boundary of an n-chain $\mathbf{x}$ is $\partial_n(\mathbf{x})$. An n-cycle is an n-chain with zero boundary. The set of all n-cycles forms a subspace $\mathbf{Z}_n(K) := \mathbf{ker}(\partial_n)$ of $\mathbf{C}_n(K)$. An n-boundary is an n-chain that is the boundary of $(n+1)$ chains. The set of all n-boundaries forms a subspace $\mathbf{B}_n(K) := \mathbf{im}(\partial_{n+1})$ of $\mathbf{C}_n(K)$. We refer to $\mathbf{Z}_n$ and $\mathbf{B}_n$ as the space of cycles and space of boundaries, respectively.

It can be shown that $\partial_n \circ \partial_{n+1}(\mathbf{x}) = 0$ for all $\mathbf{x} \in \mathbf{C}_{n+1}(K)$; colloquially, "a boundary has no boundary." Equivalently, $\partial_n \circ \partial_{n+1}$ is the zero map. Since the boundary map takes a boundary to 0, an n-boundary must also be an n-cycle. Therefore, $\mathbf{B}_n(K) \subseteq \mathbf{Z}_n(K)$.

2.4 Homology, Cycle Representatives

The nth homology group of K is defined as the quotient.

$$\mathbf{H}_n(K) := \mathbf{Z}_n(K)/\mathbf{B}_n(K).$$

Concretely, elements of $\mathbf{H}_n(K)$ are cosets of the form $[\mathbf{z}] = \{\mathbf{z}' \in \mathbf{Z}_n(K) \mid \mathbf{z}' - \mathbf{z} \in \mathbf{B}_n(K)\}$.[3] An element $h \in \mathbf{H}_n(K)$ is called an n-dimensional homology class. We say that a cycle $\mathbf{z} \in \mathbf{Z}_n(K)$ represents h, or that $\mathbf{z}$ is a cycle representative of h if $h = [\mathbf{z}]$. We say that $\mathbf{z}$ and $\mathbf{z}'$ are homologous if $[\mathbf{z}] = [\mathbf{z}']$.

Example: Consider the example in **Figure 2A**, which illustrates two homologous 1-cycles and the example in **Figure 2B**, which illustrates two non-homologous cycles.

Remark 2.4. The term homological generator has been used differently by various authors: to refer to an arbitrary nontrivial homology class, an element in a (finite) representation of $\mathbf{H}_n(K)$, as a set of cycles which generate the homology group, or (particularly in literature surrounding optimal cycle representatives) interchangeably with cycle representative. We favor the term cycle representative, to avoid ambiguity.

2.5 Betti Numbers, Cycle Bases

A (dimension-n) homological cycle basis for $\mathbf{H}_n(K)$ is a set of cycles $\mathcal{B} = \{\mathbf{z}^1, \ldots, \mathbf{z}^m\}$ such that $[\mathbf{z}^i] \neq [\mathbf{z}^j]$ when $i \neq j$, and $\{[\mathbf{z}^1], \ldots, [\mathbf{z}^m]\}$ is a basis for $\mathbf{H}_n(K)$. Modulo boundaries, every n-cycle can be expressed as a unique linear combination in $\mathcal{B}$.

Homological cycle bases have several useful interpretations. It is common, for example, to think of a 1-cycle as a type of "loop," generalizing the intuitive notion of a loop as a simple closed curve to include more intricate structures, and to regard the operation of adding boundaries as a generalized form of "loop-deformation." Framed in this light, a homological cycle basis $\mathcal{B}$ for $\mathbf{H}_1(K)$ can be regarded as a basis for the space of

[1]The ℓ_0 "norm" is not a real norm as it does not satisfy the homogeneous requirement of a norm. For example, scaling a vector $\mathbf{x}$ by a constant factor does not change its ℓ_0 "norm."

[2]See **Remark 2.1** These choices of groups have a natural notion of absolute value.

[3]More generally, we denote the groups of cycles and boundaries with coefficients in G as $\mathbf{Z}_n(K; G)$ and $\mathbf{B}_n(K; G)$. The *(dimension-n) homology of K with coefficients in G* is $\mathbf{H}_n(K; G) = \mathbf{Z}_n(K; G)/\mathbf{B}_n(K; G)$.

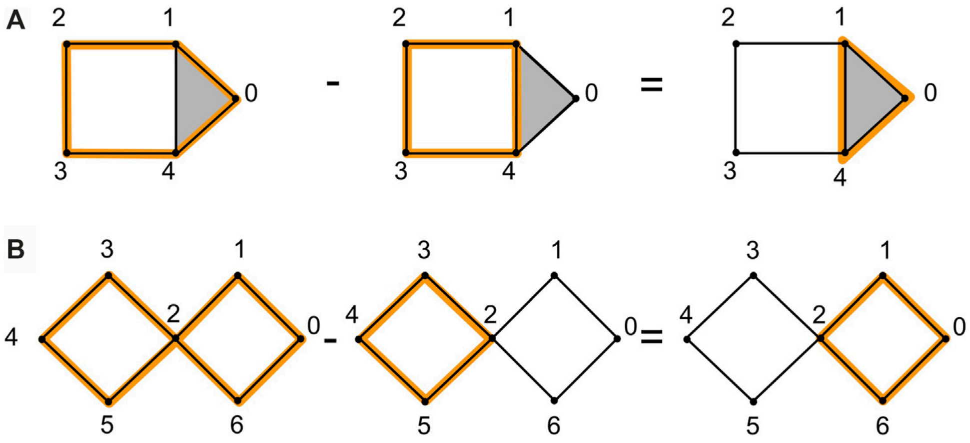

FIGURE 2 | We show an example of homologous cycles in **(A)**, adapted from (Topaz et al., 2015). The 1-cycle $(0,1)+(1,2)+(2,3)+(3,4)-(0,4)$ and the 1-cycle $(1,2)+(2,3)+(3,4)-(4,1)$ are homologous because their difference is the boundary of $(0,1,4)$. Subfigure **(B)** shows an example of non-homologous cycles. The 1-cycle $(\sum_{i=0}^{4}(i,i+1))-(5,2)+(2,6)-(0,6)$ and the 1-cycle $(2,3)+(3,4)+(4,5)-(2,5)$ are not homologous because their difference is a cycle $(0,1)+(1,2)+(2,6)-(0,6)$ which is not a linear combination of boundaries of 2-simplices.

loops-up-to-deformation in K. Higher dimensional analogs of loops involve closed "shells" made up of n-simplices.

Another interpretation construes each nontrivial homology class $[\mathbf{z}]\neq 0$ as a hole in K. Such holes are "witnessed" by loops or shells that are not homologous to the zero cycle. Viewed in this light, $\mathbf{H}_n(K)$ can naturally be regarded as the space of $(n+1)$ dimensional holes in K. The rank of the nth homology group

$$\beta_n(K) := \dim(\mathbf{H}_n(K)) = \dim(\mathbf{Z}_n(K)) - \dim(\mathbf{B}_n(K)),$$

therefore quantifies the "number of gray independent holes" in K. We call β_n the nth Betti number of K.

Example: Consider the gray disks in **Figure 1** [similar to Carlsson (2009)] with different numbers of holes and cycle representatives.

2.6 Filtrations of Simplicial Complexes

A filtration on a simplicial complex K is a nested sequence of simplicial complexes $K_\bullet = (K_{\varepsilon_i})_{i\in\{1,\ldots,T\}}$ such that

$$K_{\varepsilon_1}\subseteq K_{\varepsilon_2}\subseteq\cdots\subseteq K_{\varepsilon_T} = K,$$

where $\varepsilon_1<\cdots<\varepsilon_T$ are real numbers. A filtered simplicial complex is a simplicial complex equipped with a filtration $K_\bullet$.

Example Let X be a metric space with metric d, and let $\varepsilon_1<\cdots<\varepsilon_T$ be an increasing sequence of non-negative real numbers. Then the sequence $K_\bullet = (K_{\varepsilon_i})_{i\in\{1,\ldots,T\}}$ defined by $K_{\varepsilon_i} = \mathbf{VR}_{\varepsilon_i}(X)$ is a filtration on K.

The data of a filtered complex is naturally captured by the birth function on simplices, defined

$$\text{Birth} : K\to\mathbb{R},\ \sigma\mapsto \min\{\varepsilon_i : \sigma\in K_{\varepsilon_i}\}.$$

We regard the pair (K,Birth) as a simpilicial complex whose simplices are weighted by the birth function. For convenience, we will implicitly identify the sequence $K_\bullet$ with this weighted complex. Thus, for example, when we say that $\sigma\in K$ has birth parameter t, we mean that $\sigma\in K$ and $\text{Birth}(\sigma)=t$.

Definition 2.5. A filtration $K_\bullet$ is simplex-wise if one can arrange the simplices of K into a sequence $(\sigma_1,\ldots,\sigma_{|K|})$ such that $K_{\varepsilon_i}=\{\sigma_1,\ldots,\sigma_i\}$ for all i. A simplex-wise refinement of $K_\bullet$ is a simplex-wise filtration $K_\bullet'$ such that each space in $K_\bullet$ can be expressed in form $\{\sigma_1,\ldots,\sigma_j\}$ for some j.

As an immediate corollary, given a simplex-wise refinement of $K_\bullet$, we may naturally interpret each boundary matrix ∂_n as an element of $G^{|\mathbf{S}_{n-1}(K)|\times|\mathbf{S}_n(K)|}$, see **Remark 2.3** Under this interpretation, columns (respectively, rows) with larger indices correspond to simplices with later birth times; that is, birth time increases as one moves left-to-right and top-to-bottom.

2.7 Filtrations of Chain Complexes

If we regard $\mathbf{C}_n(K_{\varepsilon_i};G)$ as a family of formal linear combinations in $\mathbf{S}_n(K_{\varepsilon_i})$, then it is natural to consider $\mathbf{C}_n(K_{\varepsilon_i};G)$ as a subgroup of $\mathbf{C}_n(K_{\varepsilon_j};G)$ for all $i<j$. In particular, we have an inclusion map

$$\iota : \mathbf{C}_n(K_{\varepsilon_i};G)\to\mathbf{C}_n(K_{\varepsilon_j};G),$$
$$\sum_{\sigma\in\mathbf{S}_n(K_{\varepsilon_i})} x_\sigma\sigma \mapsto \sum_{\sigma\in\mathbf{S}_n(K_{\varepsilon_i})} x_\sigma\sigma + \sum_{\tau\notin\mathbf{S}_n(K_{\varepsilon_i})} 0\cdot\tau.$$

Given a simplex-wise refinement $K_\bullet'$, one can naturally regard $\mathbf{c}$ as an element $(c_1,c_2,\ldots)$ of $G^{|\mathbf{S}_n(K_{\varepsilon_i})|}$. From this perspective, ι has a particularly simple interpretation, namely "padding" by zeros:

$$\iota(\mathbf{c}) = (\underbrace{c_1,c_2,\ldots}_{\mathbf{c}},0,\ldots,0).$$

Similar observations hold when one replaces $\mathbf{C}_n$ with either $\mathbf{Z}_n$, the space of cycles, or $\mathbf{B}_n$, the space of boundaries.

2.8 Persistent Homology, Birth, Death

The notion of birth for simplices has a natural extension to chains, as well as a variant called death. Formally, the birth and death parameters of $\mathbf{c}\in\mathbf{C}_n(K)$ are

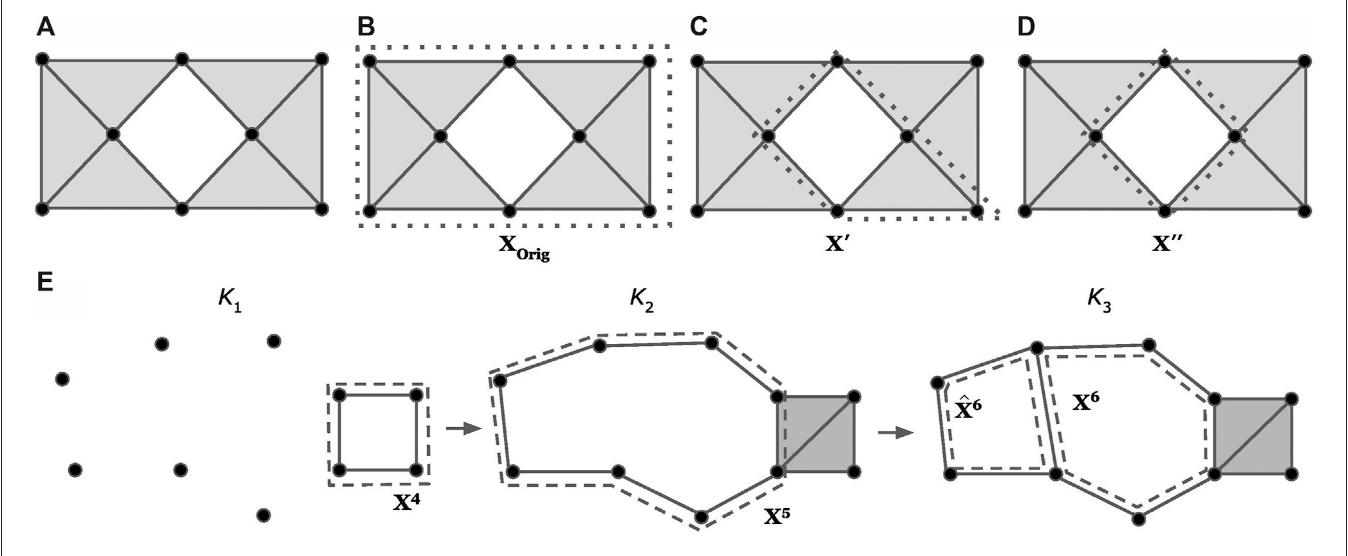

FIGURE 3 | Examples of optimizing a cycle representative (using the notion of minimizing edges) within the same homology class **(A-D)** and using a basis of cycle representatives **(E)**, modified examples adapted from (Escolar and Hiraoka, 2016; Obayashi, 2018). The dotted lines represent a cycle representative for the enclosed "hole." Intuitively, we consider $\mathbf{x}''$ in **(D)** as the optimal cycle representative since it consists of the smallest number of edges. Subfigure **(E)** shows a case where we optimize a cycle representative using a basis of cycle representatives. In **(E)**, $\{\mathbf{x}^4, \mathbf{x}^5, \mathbf{x}^6\}$ is the original basis of cycle representatives. We can substitute $\mathbf{x}^6$ with $\hat{\mathbf{x}}^6$, which we can obtain by adding $\mathbf{x}^5$ to $\mathbf{x}^6$, and thus obtain $\{\mathbf{x}^4, \mathbf{x}^5, \hat{\mathbf{x}}^6\}$ as the new basis of cycle representatives.

$$\text{Birth}(\mathbf{c}) = \min\{\varepsilon_i : \mathbf{c} \in \mathbf{C}_n(K_{\varepsilon_i})\}$$
$$\text{Death}(\mathbf{c}) = \begin{cases} \min\{\varepsilon_i : \mathbf{c} \in \mathbf{B}(K_{\varepsilon_i})\} & \mathbf{c} \in \mathbf{B}(K) \\ \infty & \text{else.} \end{cases}$$

In the special case where $\mathbf{c}$ is a cycle, Birth($\mathbf{c}$) is the first parameter value where $[\mathbf{c}]$ represents a homology class, and Death$(\mathbf{c})$ is the first parameter value where $[\mathbf{c}]$ represents the zero homology class. Thus, the half-open lifespan interval

$$\mathcal{L}(\mathbf{c}) = [\text{Birth}(\mathbf{c}), \text{Death}(\mathbf{c})),$$

is the range of parameters over which $\mathbf{c}$ represents a well-defined, nonzero homology class.

A (dimension-n) persistent homology cycle basis is a subset $\mathcal{B} \subseteq \mathbf{Z}_n(K)$ with the following two properties:

1. Each $\mathbf{z} \in \mathcal{B}$ has a nonempty lifespan interval.
2. For each $i \in \{1, \ldots, T\}$, the set

$$\mathcal{B}_{\varepsilon_i} := \{\mathbf{z} \in \mathcal{B} : \varepsilon_i \in \mathcal{L}(\mathbf{z})\},$$

is a homological cycle basis for $\mathbf{H}_n(K_{\epsilon_i})$.

Every filtration of simplicial complexes $(K_{\varepsilon_i})_{i \in \{1,\ldots,T\}}$ admits a persistent homological cycle basis $\mathcal{B}$ (Zomorodian and Carlsson, 2005). Moreover, it can be shown that the multiset of lifespan intervals (one for each basis vector), called the dimension-n barcode of $K_\bullet$,

$$\text{Barcode}_n = \{\mathcal{L}(\mathbf{z}) : \mathbf{z} \in \mathcal{B}\},$$

is invariant over all possible choices of persistent homological cycle bases $\mathcal{B}$ (Zomorodian and Carlsson, 2005).

Example: Consider the sequence of simplicial complexes (K_1, K_2, K_3) shown in **Figure 3E**. The set $\mathcal{B} = \{\mathbf{x}^4, \mathbf{x}^5, \mathbf{x}^6\}$ is a (dimension-1) persistent homological cycle basis of the filtration. The associated dimension-1 barcode is Barcode$_1$ = $\{[1,2), [2,\infty), [3,\infty)\}$ where $[2,\infty)$ and $[3,\infty)$ are the lifespans of $\mathbf{x}^5$ and $\mathbf{x}^6$, respectively.

Barcodes are among the foremost tools in topological data analysis (Ghrist, 2008; Edelsbrunner and Harer, 2008), and they contain a great deal of information about a filtration. For example, it follows immediately from the definition of persistent homological cycle bases that $\beta_n(K_{\varepsilon_i}) = |\mathcal{B}_{\varepsilon_i}|$ for all n and i. Consequently,

$$\beta_n(K_{\varepsilon_i}) = |\{J \in \text{Barcode}_n : \varepsilon_i \in J\}|.$$

2.9 Computing PH Cycle Representatives

Barcodes and persistent homology bases may be computed via the so-called $R = DV$ decomposition (Cohen-Steiner et al., 2006) of the boundary matrices ∂_n. Details are discussed in the **Supplementary Material**.

3 RELATED WORK ON MINIMIZING CYCLE REPRESENTATIVES

One important problem in TDA is interpreting homological features. In general, a lifetime interval $\mathcal{L}$ corresponding to a feature may be represented by many different cycle representatives. As discussed in Chen et al. (2008), localizing homology classes can be characterized as finding a representative cycle with the most concise geometric measure. As an illustrative example from Escolar and Hiraoka (2016), **Figure 3A** shows a simplicial complex K with $\mathbf{H}_1(K)$ isomorphic to $\mathbb{Q}$ or equivalently, $\beta_1 = 1$; it contains one hole. **Figures 3B–D** display three cycle representatives, $\mathbf{x}^{Orig}$, $\mathbf{x}'$, and $\mathbf{x}''$, each of which represents the same homology class (heuristically, they encircle the same hole). We intuitively prefer $\mathbf{x}''$ as a representative, since it involves the fewest edges and "hugs" the hole most tightly. Given a simplicial complex K and a nontrivial cycle $\mathbf{x}^{Orig}$ on it, we are interested in finding a cycle representative that is optimal with respect to some geometric criterion. In this section, we discuss previous studies on optimal cycle representatives.

Minimal cycle representatives have proven useful in many applications. Hiraoka et al. (2016) use TDA to geometrically analyze amorphous solids. Their analysis using minimal cycle representatives explicitly captures hierarchical structures of the shapes of cavities and rings. Wu et al. (2017) discuss an application of optimal cycles in Cardiac Trabeculae Restoration, which aims to reconstruct trabeculae, very complex muscle structures that are hard to detect by traditional image segmentation methods. They propose to use topological priors and cycle representatives to help segment the trabeculae. However, the original cycle representative can be complicated and noisy, causing the reconstructed surface to be messy. Optimizing the cycle representatives makes the cycle more smooth and thus, leads to more accurate segmentation results. Emmett et al. (2015) use PH to analyze chromatin interaction data to study chromatin conformation. They use loops to represent different types of chromatin interactions. To annotate particular loops as interactions, they need to first localize a cycle. Thus, they propose an algorithm to locate a minimal cycle representative for a given PH class using a breadth-first search, which finds the shortest path that contains the edge that enters the filtration at the birth time of the cycle and is homologically independent from the minimal cycles of all PH classes born before the current cycle.

There are several approaches used to define an optimal cycle representative. Dey et al. (2011) propose an algorithm to find an optimal homologous 1-cycle for a given homology class via linear programming. That is, they consider a single homology class $[\mathbf{x}]$ and search for a homologous cycle representative that minimizes some geometric measure within that class, for instance, the number of 1-simplices within the representative. Escolar and Hiraoka (2016) extend this approach to find an optimal cycle by using cycles outside of a single homology class to "factor out" redundant information. In this approach, an optimal cycle representative is no longer guaranteed to be homologous to the original representative, but the collection of cycle representatives have each been independently optimized and the collection still forms a homology basis. Further, Escolar and Hiraoka (2016) extends this approach to achieve a filtered cycle basis, although we note that it is not guaranteed to be a persistent homology basis. The two approaches in Dey et al. (2011) and Escolar and Hiraoka (2016) aim to minimize the number of 1-simplices in a cycle representative. Obayashi (2018) proposes an alternative algorithm for finding volume-optimal cycles in persistent homology, which minimize the number of 2-simplices which the cycle representative bounds, also using linear programming. These methods serve as the foundation for our present paper and are discussed in more detail in the rest of this section.

In addition to linear programming, many researchers have contributed to the problem of computing optimal cycles: Wu et al. (2017) propose an algorithm for finding shortest persistent 1-cycles. They first construct a graph based on the given simplicial complex and then compute annotation for the given complex. The annotation assigns all edges different vectors and can be used to verify if a cycle belongs to the desired group of cycles. They then find the shortest path between two vertices of the edge born at the birth time of the original cycle representative using a new A^* heuristic search strategy. Their algorithm is a polynomial time algorithm but in the worst case, the time complexity is exponential to the number of topological features. Dey et al. (2010) propose a polynomial-time algorithm that computes a set of loops from a VR complex of the given data whose lengths approximate those of a shortest basis of the one dimensional homology group $\mathbf{H}_1$. In Dey et al. (2019), show that finding optimal (minimal) persistent 1-cycles is NP-hard and then propose a polynomial time algorithm to find an alternative set of meaningful cycle representatives. This alternative set of representatives is not always optimal but still meaningful because each persistent 1-cycle is a sum of shortest cycles born at different indices. They find shortest cycles using Dijkstra's algorithm by considering the 1-skeleton as a graph. This list is by no means exhaustive, and does not touch on the wide variety of related approaches, e.g. Chen and Freedman (2010b), which attempts to fit cycle representatives within a ball of minimum radius.

In the next subsection, we briefly introduce some basic notions of linear programming, and then in the subsequent three subsections, we survey the optimization problems on which the present work is based.

3.1 Background: Linear Programming

Linear programming seeks to find a set of decision variables $\mathbf{x} = (x_1, \ldots, x_\eta)^T$ which optimize a linear cost (or objective) function $\mathbf{c}^T\mathbf{x}$ subject to a set of linear (in)equality constraints $\mathbf{a}_1^T\mathbf{x} = b_1, \ldots, \mathbf{a}_\mu^T\mathbf{x} = b_\mu$. Any linear optimization problem can be written as a Linear Program (LP) in standard form

$$\begin{aligned} \text{minimize} \quad & \mathbf{c}^T\mathbf{x} \\ \text{subject to} \quad & A\mathbf{x} = \mathbf{b} \\ & \mathbf{x} \geq 0, \end{aligned} \tag{1}$$

where A is the $\mu \times \eta$ matrix with coefficients of the constraints as rows and $\mathbf{b} = (b_1, \ldots, b_\mu)^T$. Linear programming is well-studied and discussed in many texts (Bertsimas and Tsitsiklis, 1997; Vanderbei, 2014; Boyd and Vandenberghe, 2004).

The optimal solution $\mathbf{x}^\star$ satisfies the constraints while optimizing the objective function, yielding the optimal cost $\mathbf{c}^T\mathbf{x}^\star$. The feasible set of solutions in a linear optimization problem is a polyhedron defined by the linear constraints. In general, the optimal solution of a (non-degenerate) LP will occur at a vertex of the polyhedron and can be solved with the standard simplex algorithm, which traverses through the edges of the polytope to vertices in a cost reducing manner, or interior point methods, which traverse along the inside of the polytope to reach an optimal vertex. In the worst-case, the complexity of the simplex method is exponential, yet it often runs remarkably fast, while interior point methods are polynomial time algorithms.

Standard LPs search for real-valued optimal solutions, but in some instances, a restriction of the decision variables, such as requiring integral solutions, may be necessitated. The mixed integer programming (MIP) problem is written

$$\begin{array}{ll} \text{minimize} & \mathbf{c}^T\mathbf{x} + \mathbf{d}^T\mathbf{y} \\ \text{subject to} & A\mathbf{x} + B\mathbf{y} = \mathbf{b} \\ & \mathbf{x}, \mathbf{y} \geq \mathbf{0} \\ & \mathbf{x} \text{ integer,} \end{array} \quad (2)$$

for matrices A, B and vectors $\mathbf{b}, \mathbf{c}, \mathbf{d}$. A standard LP has fewer constraints, and thus, will have optimal cost less than or equal to that of the analogous MIP. MIPs are much more challenging to solve than LPs, as they are discrete as opposed to convex optimization problems, and no efficient general algorithm is known (Bertsimas and Tsitsiklis, 1997). However, LP relaxations, (exponential-time) exact, (polynomial-time) approximation, and heuristic algorithms can be used to obtain solutions to MIPs.

In this paper, we determine optimal cycle representatives with both LP and MIP formulations.

3.2 Minimal Cycle Representatives of a Homology Class

Given a homology class $h = [\mathbf{x}^{Orig}] \in \mathbf{H}_n(K; G)$ and a function $\text{loss}: \mathbf{Z}_n(K; G) \to \mathbb{R}$, how does one find a cycle representative of h on which loss attains minimum? This problem is equivalent to solving the following program defined in Dey et al. (2011):

$$\begin{array}{ll} \text{minimize} & \text{loss}(\mathbf{x}) \\ \text{subject to} & \mathbf{x} = \mathbf{x}^{Orig} + \partial_{n+1}\mathbf{w} \\ & \mathbf{w} \in \mathbf{C}_{n+1}(K; G). \end{array} \quad (3)$$

This formulation considers all cycle representatives homologous to $\mathbf{x}^{\text{Orig}}$, i.e. that differ by a boundary, and selects the optimal representative $\mathbf{x}$ which minimizes loss. The program in **Eq. 3** is correct because the coset h can be expressed in the form

$$h = \mathbf{x}^{\text{Orig}} + \mathbf{B}_n(K; G) = \{\mathbf{x}^{Orig} + \partial_{n+1}\mathbf{w} | \mathbf{w} \in \mathbf{C}_{n+1}(K; G)\}.$$

In practice, a cycle representative $\mathbf{x}^{Orig}$ is almost always provided together with the initial problem data (which consists of K, G, loss, and h), so the central challenge lies with solving the program in **Eq. 3**.

Several variants of the program in **Eq. 3** have been studied, especially where $\text{loss}(\mathbf{x}) = \|\mathbf{x}\|_0$ or $\text{loss}(\mathbf{x}) = \|\mathbf{x}\|_1$. For a survey of results when $G = \mathbb{F}_2$, see Chen and Freedman (2010b). For a discussion of results when $G = \mathbb{Z}$, see Dey et al. (2011). Broadly speaking, minimizing against ℓ_0 tends to be hard, even when K has attractive properties such as embeddability in a low-dimensional Euclidean space (Borradaile et al., 2020). Minimizing against ℓ_1 is hard when $G = \mathbb{F}_2$ (since, in this case, $\ell_1 = \ell_0$), but tractable via linear programming when $G \in \{\mathbb{Q}, \mathbb{R}\}$.

An interesting variant of the minimal cycle representative problem is the minimal persistent cycle representative problem. This problem was described in Chen et al. (2008) and may be formulated as follows: given an interval $[a, b) \in \text{Barcode}_n(K_\bullet)$, solve

$$\begin{array}{ll} \text{minimize} & \text{loss}(\mathbf{x}) \\ \text{subject to} & \text{Birth}(\mathbf{x}) = a \\ & \text{Death}(\mathbf{x}) = b \\ & \mathbf{x} \in \mathbf{Z}_n(K_a; G), \end{array} \quad (4)$$

for $\mathbf{x}$. An advanced treatment of this problem can be found in (Chen et al., 2008) for special case where 1) $G = \mathbb{F}_2$, 2) loss is a weighted sum of incident edges, and 3) the birth function assigns distinct values to any two simplices of the same dimension, and 4) $n = 1$.

3.3 Minimal Homological Cycle Bases

The program in **Eq. 3** has a natural extension when G is a field. This extension focuses not on the smallest representative of a single homology class, but the smallest homological cycle basis. It may be formally expressed as follows:

$$\begin{array}{ll} \text{minimize} & \sum_{\mathbf{x} \in \mathcal{B}} \text{loss}(\mathbf{x}) \\ \text{subject to} & \mathcal{B} \in \text{HCB}_n(K; G), \end{array} \quad (5)$$

where $\text{HCB}_n(K, G)$ is the family of dimension-n homological cycle bases of $\mathbf{H}_n(K; G)$. Thus, the program is finding a complete generating set $\mathcal{B}$ for all of the homological cycles of dimension n where each element has been minimized in some sense.

It is natural to wonder whether a solution to the program in **Eq. 5** could be obtained by first calculating an arbitrary (possibly non-minimal) homological cycle basis $\mathcal{B} = \{\mathbf{x}^1, \ldots, \mathbf{x}^m\}$ and then selecting an optimal cycle representative $\mathbf{z}^i$ from each homology class $[\mathbf{x}^i]$. Unfortunately, the resulting basis need not be optimal. To see why, consider the simplicial complex K_3 shown in **Figure 3E**, taking G to be $\mathbb{Q}$ and loss to be the ℓ_0 norm. Complex $K_\bullet$ has several different homological cycle bases in degree 1, including $\mathcal{B}_0 := \{\hat{\mathbf{x}}^6, \mathbf{x}^6\}$, $\mathcal{B}_1 := \{\mathbf{x}^5, \mathbf{x}^6\}$, and $\mathcal{B}_2 := \{\mathbf{x}^5, \hat{\mathbf{x}}^6 + \mathbf{x}^4\}$. However, only $\mathcal{B}_0$ is ℓ_0 minimal. Moreover, each of the cycle representatives $\mathbf{x}^5, \mathbf{x}^6, \hat{\mathbf{x}}^6$ is already minimal within its homology class, so element-wise minimization will not transform $\mathcal{B}_1$ or $\mathcal{B}_2$ into optimal bases, as might have been hoped.

As with the minimal cycle representative problem, the minimal homological cycle basis problem has been well-studied in the special case where loss is the ℓ_0 norm and $G = \mathbb{F}_2$. In this case, the program in **Eq. 5** is NP-hard to approximate for $n > 1$, but $O(n^3)$ when $n = 1$ (Dey et al., 2018). Several interesting variants and special cases have been developed in the $n = 1$ case, as well Erickson and Whittlesey (2009), Dey et al. (2010), and Chen and Freedman (2010). We are not currently aware of a systematic treatment for the case $G \in \{\mathbb{Q}, \mathbb{R}\}$.

A natural variant of the minimal homological cycle basis program in **Eq. 5** is the minimal persistent homological cycle basis problem

$$\begin{array}{ll} \text{minimize} & \sum_{\mathbf{x} \in \mathcal{B}} \text{loss}(\mathbf{x}) \\ \text{subject to} & \mathcal{B} \in \text{PrsHCB}_n(K_\bullet; G), \end{array} \quad (6)$$

where $\text{PrsHCB}_n(K_\bullet; G)$ is the set of persistent homological cycle bases. This is a stricter condition than the program in **Eq. 5** in that not only does it require that the elements of $\mathcal{B}$ form a generating set of all cycles of dimension n, but the barcode associated to $\mathcal{B}$ must match $\text{Barcode}_n(K_\bullet)$. That is, the multisets of birth/death pairs must be identical.

The program in **Eq. 6** is much more recent than the program in **Eq. 5**, and consequently appears less in the literature. In the special case where every bar in the multiset $\text{Barcode}_n(K_\bullet)$ has multiplicity 1 (i.e. there are no duplicate bars), the program in **Eq. 6** can be solved by making one call to the minimal persistent cycle representative program in **Eq. 4** for each bar. In particular, the method of Chen et al. (2008) may be applied to obtain a minimal persistent basis when the correct hypotheses are satisfied: $G = \mathbb{F}_2$, loss is a weighted sum of incident simplices, there are distinct birth times for all simplices of the same dimension, and $n = 1$. In general, however, bars of multiplicity two are possible, and in this case repeated application of the program in **Eq. 4** will be insufficient.

3.4 Minimal Filtered Cycle Space Bases

A close cousin of the minimal homological cycle basis the program in **Eq. 5** is the minimal filtered cycle basis problem, which may be formulated as follows

$$\begin{array}{ll} \text{minimize} & \sum_{\mathbf{x}\in\mathcal{C}}\text{loss}(\mathbf{x}) \\ \text{subject to} & \mathcal{C} \in \text{FCB}(K_\bullet; G), \end{array} \tag{7}$$

where $\text{FCB}(K_\bullet)$ is the family of all bases $\mathcal{C}$ of $\mathbf{Z}_n(K_{\varepsilon_T})$ such that $\mathcal{C}$ contains a basis for each subspace $\mathbf{Z}_n(K_{\varepsilon_i})$, for $i \in \{1, \ldots, T\}$.

Escolar and Hiraoka (2016) provide a polynomial time solution via linear programming when.

1. loss is the ℓ_1 norm,
2. $G = \mathbb{Q}$, and
3. $K_\bullet$ is a simplex-wise filtration [without loss of generality, $K_\bullet = (K_1, \ldots, K_T)$].

Their key observation is that $\mathcal{C}$ is an optimal solution to the program in **Eq. 6** if and only if $\mathcal{C}$ can be expressed as a collection $\{\mathbf{z}^j : j \in J\}$ where

1. the set $J = \{j : \mathbf{Z}_n(K_{j-1}) \subsetneq \mathbf{Z}_n(K_j)\}$ that indexes the cycles is the list of filtrations at which a novel n-cycle appears, and.
2. for each $j \in J$, the cycle $\mathbf{z}^j$ first appears in K_j and is a minimizer for the loss function among all such cycles, i.e. $\mathbf{z}^j \in \text{argmin}_{\mathbf{z}\in\mathbf{Z}_n(K_j)\setminus\mathbf{Z}_n(K_{j-1})}\text{loss}(\mathbf{z})$.

The authors formulate this problem as

$$\begin{array}{ll} \text{minimize} & \|\mathbf{x}\|_1 \\ \text{subject to} & \mathbf{x} = \mathbf{x}^{Orig} + \sum\limits_{r\in R} w_r g^r + \sum\limits_{s\in S} v_s f^s \\ & \mathbf{w} \in \mathbb{Q}^R \\ & \mathbf{v} \in \mathbb{Q}^S, \end{array} \tag{8}$$

where $\mathbf{x}^{Orig} \in \mathbf{Z}_N(K_j)\setminus\mathbf{Z}_N(K_{j-1})$ is a novel cycle representative at filtration j; $\{g^r : r \in R\}$ is a basis for $\mathbf{B}_n(K_{j-1})$[4]; and $\{g^r : r \in R\} \cup \{f^s : s \in S\}$ is an extension of the given basis for $\mathbf{B}_n(K_{j-1})$ to a basis for $\mathbf{Z}_n(K_{j-1})$. That is, $\mathbf{x}^{Orig}$ is a cycle that has just appeared in the filtration. To optimize it, we are allowed to consider linear combinations of both boundaries, $\{g^r\}$, and cycles, $\{f^s\}$, born before $\mathbf{x}^{Orig}$. The cycle $\mathbf{x}$ obtained in this way cannot have a birth time before that of $\mathbf{x}^{Orig}$, but may have a different death time if $[\sum_{s\in S} v_s f^s]$ dies later than $[\mathbf{x}^{Orig}]$.

The algorithm developed in Escolar and Hiraoka (2016) is cleverly constructed to extract $\mathbf{x}^{Orig}$, $\{g^r : r \in R\}$, and $\{f^s : s \in S\}$ from matrices which are generated in the normal course of a barcode calculation.

Remark 3.1. It is important to distinguish between PrsHCB and FCB, hence between the optimization the programs in **Eqs 6**, **7**. As Escolar and Hiraoka (2016) point out, given $\mathcal{B} \in \text{PrsHCB}$ and $\mathcal{C} \in \text{FCB}$, one can always find an injective function $\phi : \mathcal{B} \to \mathcal{C}$ such that $\text{Birth}(\mathbf{z}) = \text{Birth}(\phi(\mathbf{z}))$ for all $\mathbf{z}$. However, this does not imply that $\phi(\mathcal{B}) \in \text{PrsHCB}$, as the deaths of each cycle may not coincide. Indeed, the question of whether a persistent homological cycle basis can be extracted from $\mathcal{C}$ by any means is an open question, so far as we are aware. We provide an example in **Figure 4** where the cycle basis obtained by optimizing each cycle using the program in **Eq. 7** is not a persistent homology cycle basis $\mathcal{B}$.

Though **Remark 3.1** is a bit disappointing for those interested in persistent homology, the machinery developed to study the program in **Eq. 7** is nevertheless interesting, and we will discuss an adaptation.

3.5 Volume-Optimal Cycles: Minimizing Over Bounding Chains

Schweinhart (2015) and Obayashi (2018) consider a different notion of minimization: volume[5] optimality. This approach focuses on the "size" of a bounding chain; it is specifically designed for cycle representatives in a persistent homological cycle basis.

Obayashi (2018) formalizes the approach as follows. First, assume a simplex-wise filtration $K_\bullet$; without loss of generality, $K_\bullet = (K_1, \ldots, K_T)$, and we may enumerate the simplices of K_T such that $K_i = \{\sigma_1, \ldots, \sigma_i\}$ for all i. Since each simplex has a unique birth time, each interval in $\text{Barcode}_n(K_\bullet) = \{[b_1, d_1), \ldots, [b_N, d_N)\}$ has a unique left endpoint. Fix $[b_i, d_i) \in \text{Barcode}_n(K_\bullet)$ such that $d_i < \infty$ (in the case $d_i = \infty$, volume is undefined). It can be shown that σ_{b_i} is an n-simplex and σ_{d_i} is an $(n+1)$ simplex. We use $\tau_k = \sigma_k$ below when the dimension of σ_k is equal to $n+1$.

A persistent volume $\mathbf{v}$ for $[b_i, d_i)$ is an $(n+1)$ chain $\mathbf{v} \in \mathbf{C}_{n+1}(K_{d_i})$ such that[6]

$$\mathbf{v} = \tau_{d_i} + \sum_{\tau_k \in \mathcal{F}_{n+1}} \alpha_k \tau_k \tag{9}$$

$$(\partial_{n+1}\mathbf{v})_\sigma = 0 \quad \forall \sigma \in \mathcal{F}_n \tag{10}$$

$$(\partial_{n+1}\mathbf{v})_{\sigma_{b_i}} \neq 0, \tag{11}$$

[4]Because of the assumption that $K_\bullet$ is a simplex-wise filtration, if there is a new n-cycle in K_j then there cannot also be a new $(n+1)$ simplex, so this is also a basis for $\mathbf{B}_n(K_j)$.

[5]This notion of volume differs from that of Chen and Freedman (2010b). The latter refers to volume as the ℓ_0 norm of a chain, while the former (which we discuss in this section) refers to the ℓ_0 norm of a *bounding* chain.

[6]If we regard $\partial_{n+1}\mathbf{v}$ as a function $\mathbf{S}_n(K_{d_i}) \to \mathbb{Q}$, then $(\partial_{n+1}\mathbf{v})_\sigma$ is the value taken by $\partial_{n+1}\mathbf{v}$ on simplex σ. Alternatively, if we regard $\partial_{n+1}\mathbf{v}$ as a linear combination of n-simplices, then $(\partial_{n+1}\mathbf{v})_\sigma$ is the coefficient placed by $\partial_{n+1}\mathbf{v}$ on σ.

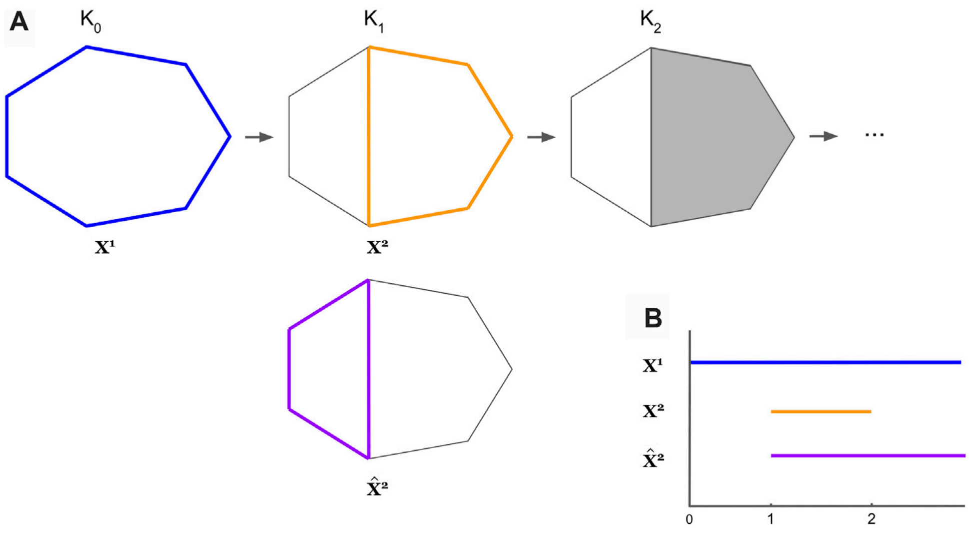

FIGURE 4 | An example where the optimal cycles obtained from **Eq. 8** do not form a persistent homological cycle basis. The thickened colored cycles in Subfigure **(A)** represent a cycle representative for the hole it encloses, and the bar with the corresponding color in Subfigure **(B)** records the lifespan of the cycle. In Subfigure **(A)**, we see $\mathcal{L}(\mathbf{x}^1) = [0,\infty)$, $\mathcal{L}(\mathbf{x}^2) = [1,2)$ Then, $\{\mathbf{x}^1,\mathbf{x}^2\}$ forms a basis for the persistent homological cycles. The cycle representative $\hat{\mathbf{x}}^2$ is an optimal cycle representative obtained by solving **Eq. 7** for the filtered simplicial complex K_2. However, $\mathcal{L}(\hat{\mathbf{x}}^2) = [1,\infty)$, and thus $\{\mathbf{x}^1,\hat{\mathbf{x}}^2\}$ is no longer a persistent homological cycle basis.

where $\mathcal{F}_n = \{\sigma_k \in \mathbf{S}_n(K) : b_i < k < d_i\}$ denotes the n-simplices alive in the window between the birth and death time of the interval under consideration.

We interpret these equations as follows: Given a persistence interval $[b_i, d_i)$, condition **Eq. 9** implies that $\mathbf{v}$ only contains $n+1$ simplices born between b_i and d_i and must contain the $n+1$ simplex born at d_i. Condition (**Eq. 10**) ensures that the boundary of $\mathbf{v}$ contains no n-simplex born after b_i, and condition (**Eq. 11**) ensures that the boundary of $\mathbf{v}$ contains the n-simplex born at b_i. This guarantees that $\partial_{n+1}\mathbf{v}$ exists at step b_i, does not exist before step b_i, and dies at step d_i.

Theorem 3.2. (Obayashi, 2018). Suppose that $[b_i, d_i) \in \text{Barcode}_n(K_\bullet)$ and $d_i < \infty$.

1. Interval $[b_i, d_i)$ has a persistent volume.
2. If $\mathbf{v}$ is a persistent volume for $[b_i, d_i)$ then $\mathcal{L}(\partial_{n+1}\mathbf{v}) = [b_i, d_i)$.
3. Suppose that $\mathcal{B}$ is an n-dimensional persistent homological cycle basis for $K_\bullet$, that $\mathbf{x}^{Orig} \in \mathcal{B}$ is the basis vector corresponding to $[b_i, d_i)$, and that $\mathbf{v}$ is a persistent volume for $[b_i, d_i)$. Then, $(\mathcal{B}\backslash\{\mathbf{x}^{Orig}\}) \cup \{\partial_{n+1}\mathbf{v}\}$ is also a persistent homological cycle basis.

By **Theorem 3.2**, for any barcode composed of finite intervals, one can construct a persistent homological cycle basis from nothing but (boundaries of) persistent volumes! Were we to build such a basis, it would be natural to ask for volumes that are optimal with respect to some loss function; that is, we might like to solve

$$\begin{array}{ll} \text{minimize} & \text{loss}(\mathbf{v}) \\ \text{subject to} & (9), (10), (11) \\ & \mathbf{v} \in \mathbf{C}_{n+1}(K_{d_i}), \end{array} \tag{12}$$

for each barcode interval $[b_i, d_i)$. A solution $\mathbf{v}$ to the program in **Eq. 12** is called an optimal volume; its boundary, $\mathbf{x} = \partial_{n+1}\mathbf{v}$ is called a volume-optimal cycle.

It is interesting to contrast ℓ_0 minimal cycle representatives for an interval[7] $[b_i, d_i)$ with ℓ_0 volume-optimal cycle for the same interval. Consider, for example, **Figure 5**. For the persistence interval $[b_i, d_i)$, the cycle with minimal number of edges is $(a,b) + (b,c) + (c,d) + (d,a)$. However, the volume-optimal cycle would be found as follows: considering K_{d_i}, we must find the fewest 2-simplices whose boundary captures the persistence interval. In this case, we would have an optimal volume $(a,b,e) + (b,c,e) + (a,d,e)$ and volume-optimal cycle $(a,b) + (b,c) + (c,e) + (e,d) + (d,a)$.

3.6 ℓ_0 vs. ℓ_1 Optimization

As mentioned above, it is common to choose $\text{loss}(\mathbf{x}) = \|\mathbf{x}\|_0$ or $\text{loss}(\mathbf{x}) = \|\mathbf{x}\|_1$.[8] A linear program (LP) with ℓ_1 objective function is polynomial time solvable. However, an objective function with the ℓ_0 norm restricted to $\{0, 1, -1\}$ coefficients is often preferred as the output of such a problem is highly interpretable: a cycle representative with minimal number of edges or enclosing the minimal number of triangles. Yet, ℓ_0

[7]Technically, this notion is not well-defined; to be formal, we should fix a persistent homology cycle basis $\mathcal{B}$, fix a cycle representative $\mathbf{z} \in \mathcal{B}$ with lifespan interval $[b_i, d_i)$, and ask for an ℓ_0 cycle representative in the same homology class, $[\mathbf{z}] \in \mathbf{H}_n(K_{b_i})$, as per the program in **Eq. (3)**. However, in simple cases the intended meaning is clear.

[8]Other choices of loss function, e.g., the ℓ_p norm, are common throughout mathematical optimization. While we focus on ℓ_0 and ℓ_1 due to their tendency to produce sparse solutions, other choices may be better or worse suited, depending on the intended application. For example, since ℓ_2 loss imposes lighter penalties on small errors and heavier penalties on large ones (as compared to ℓ_1), it is especially sensitive to outliers; this makes it useful for tasks such as function estimation. On the other hand, by imposing relatively heavy penalties on small errors, ℓ_1 loss encourages sparsity (Tahbaz-Salehi and Jadbabaie, 2008; Tahbaz-Salehi and Jadbabaie, 2010).

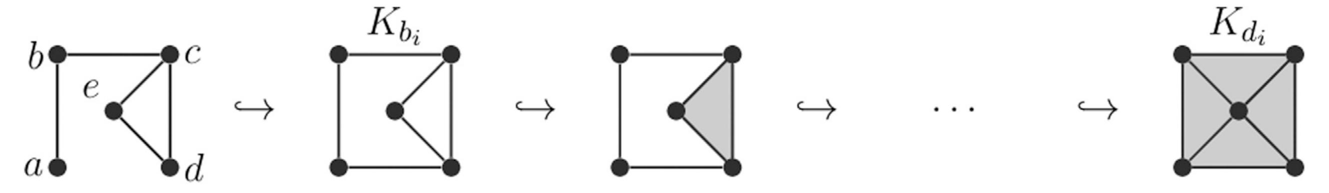

FIGURE 5 | A situation in which a volume-optimal cycle is different from the uniform minimal cycle. Consider the filtered simplicial complex pictured. For the persistence interval $[b_i, d_i)$, the cycle with minimal 0-norm (fewest number of edges) is $(a,b) + (b,c) + (c,d) + (d,a)$. However, the volume-optimal cycle would be found as follows: considering K_{d_i}, we must find the fewest 2-simplices whose boundary captures the persistence interval. In this case, we would have an optimal volume $(a,b,e) + (b,c,e) + (a,d,e)$ and volume-optimal cycle $(a,b) + (b,c) + (c,e) + (e,d) + (d,a)$.

optimization is known to be NP-hard (Tahbaz-Salehi and Jadbabaie, 2010).

The ℓ_1 norm promotes sparsity and often gives a good approximation of ℓ_0 optimization (Tahbaz-Salehi and Jadbabaie, 2008; Tahbaz-Salehi and Jadbabaie, 2010), but the solution may not be exact. Yet, if all of the coefficients of the solution x are restricted to 0 or ± 1 in the optimization problem, then the ℓ_0 and ℓ_1 norms are identical. A looser restriction, as proposed in Escolar and Hiraoka (2016), would be to solve an optimization with ℓ_1 objective function with integer constraints on the solution.

Requiring the solution to be integral also allows us to understand the optimal solution more intuitively than having fractional coefficients. Such an optimization problem is called a mixed integer program (MIP), which is known to be slower than linear programming and is NP-hard (Obayashi, 2018). Many variants of integer programming special to optimal homologous cycles, in particular, have been shown to be hard as well (Borradaile et al., 2020). In **Section 4**, we discuss the optimization problems we implement, where each is solved both as an LP with an ℓ_1 norm in the objective function and an MIP by adding the constraint that $\mathbf{x}$ is integral.

Dey et al. (2011) gives the totally unimodularity sufficient condition which guarantees that an LP and MIP give the same optimal solution. A matrix is totally unimodular if the determinant of each square submatrix is -1, 0, or 1. Dey et al. (2011) give conditions for when the ∂_{n+1} matrix is totally unimodular. If the totally unimodularity condition is not satisfied, then an LP may not give the desired result. As totally unimodularity is not guaranteed for all boundary matrices (Henselman and Dłotko, 2014), we cannot rely on this condition.

3.7 Software Implementations

Edge-minimal cycles: Software implementing the edge-loss method introduced in Escolar and Hiraoka (2016) can be found at Escolar and Hiraoka (2021). This is a C++ library specialized for 3-dimensional point clouds.

Triangle-loss optimal cycles: The volume optimization technique introduced in Obayashi (2018) is available through the software platform HomCloud, available at Obayashi et al. (2021). The code can be accessed by unarchiving the HomCloud package (for example, https://homcloud.dev/download/homcloud-3.1.0.tar.gz) and picking the file homcloud-x.y.z/homcloud/optvol.py.

4 PROGRAMS AND SOLUTION METHODS

The present work focuses on linear programming (LP) and mixed integer programming (MIP) optimization of 1-dimensional persistent homology cycle representatives with $\mathbb{Q}$ − coefficients. While the methods discussed below can be applied to any homological dimension, we limit the scope of the present work to dimension one. As described in **Section 3**, we follow two general approaches: those that measure loss as a function of n-simplices, and those that measure loss as a function of $n+1$ simplices. Motivated by the $n = 1$ case, we refer to the former as edge-loss methods and the latter as triangle-loss methods. For our empirical analysis, four variations (corresponding to two binary parameters) are chosen from each approach, yielding a total of eight distinct optimization problems.

Concerning implementation, we find that triangle-loss methods [namely, Obayashi (2018)] can be applied essentially as discussed in that paper. The greatest challenge to implementing this approach is the assumption of an underlying simplex-wise filtration. This necessitates parameter choices and preprocessing steps not included in the optimization itself; we discuss how to execute these steps below.

Implementation of edge-loss methods is slightly more complex. For binary coefficients ($G = \mathbb{F}_2$) a variety of combinatorial techniques have been implemented in dimension 1 (Chen et al., 2008; Zhang and Wu, 2019). Escolar and Hiraoka (2016) provide an approach for $\mathbb{Q}$ − coefficients, but in general this may not yield a persistent homology cycle basis, see **Remark 3.1**. In addition to the triangle-loss method mentioned in **Section 3.5**, Obayashi (2018) introduces a modified form of this edge-loss method which *does* guarantee a persistent homology basis, but assumes a simplex-wise filtration. We show that this approach can be modified to remove the simplex-wise filtered constraint.

Neither of the approaches presented here is guaranteed to solve the minimal persistent homology cycle basis problem, the program in **Eq. 6**. In the case of triangle-loss methods, this is due to the (arbitrary) choice of a total order on simplices. In the case of edge-loss methods, it is due to the choice of an initial persistent homology cycle basis.

In the remainder of this section, we present the eight programs studied, including any modifications from existing work.

4.1 Structural Parameters

Each program addressed in our empirical study may be expressed in the following form

$$\begin{aligned} \text{minimize} \quad & \|W\mathbf{x}\|_1 = \sum_\sigma W[\sigma,\sigma]\left(x^+_\sigma + x^-_\sigma\right) \\ \text{subject to} \quad & \mathbf{x} = x^+ - x^- \\ & \mathbf{x}^+, \mathbf{x}^- \geq 0 \\ & \mathbf{x} \in \mathcal{X}, \end{aligned} \tag{13}$$

where $\mathcal{X}$ is a space of feasible solutions and W is a diagonal matrix with nonnegative entries. These programs vary along 3 parameters:

1. *Chain dimension of* $\boldsymbol{x}$. If $\mathcal{X}$ is a family of 1-chains, then we say that the program in **Eq. 13** is an edge-loss program. If $\mathcal{X}$ is a family of 2-chains, we say that the program in **Eq. 13** is a triangle-loss program.
2. *Integrality*. The program is integral if each $\boldsymbol{x} \in \mathcal{X}$ has integer coefficients; otherwise we call the problem non-integral.
3. *Weighting*. For each loss type (edge vs. triangle) we consider two possible values for W: identity and non-identity. In the identity case, all edges (or triangles) are weighted equally; we call this a uniform-weighted problem. In the non-identity case we weigh each entry according to some measurement of "size" of the underlying simplex (length, in the case of edges, and area, in the case of triangles).[9] There is precedent for such weighting schemes in existing literature (Chen et al., 2008; Dey et al., 2011).

Edge-loss and triangle-loss programs will be denoted Edge and Tri, respectively. Integrality will be indicated by a superscript I (integer) or NI (non-integer). Uniform weighting will be denoted by a subscript *Unif* (uniform); non-uniform weighting will be indicated by subscript *Len* (for edge-loss programs) or *Area* (for triangle-loss programs). Thus, for example, Edge^I_{Len} denotes a length-weighted edge-loss program with integer constraints.

4.2 Edge-Loss Methods

Our approach to edge-loss minimization, based on work by Escolar and Hiraoka (2016), is summarized in **Algorithm 1**. As in Escolar and Hiraoka (2016), we obtain $\mathbf{x}$ by taking a linear combination of $\mathbf{x}^{Orig}$ with not only boundaries but cycles as well; consequently $\mathbf{x}$ need not be homologous to $\mathbf{x}^{Orig}$.

Our pipeline differs from Escolar and Hiraoka (2016) in three respects. First, we perform all optimizations after the persistence calculation has run. On the one hand, this means that our persistence calculations fail to benefit from the memory advantages offered by optimized cycles; on the other hand, separating the calculations allows one to "mix and match" one's favorite persistence solver with one's favorite linear solver, and we anticipate that this will be increasingly important as new, more efficient solvers of each kind are developed. Second, we introduce additional constraints which guarantee that $\mathcal{B}^* \in$ PrsHCB [and, moreover, $\mathcal{L}(\mathbf{x}) = \mathcal{L}(\mathbf{x}^{Orig})$ for each $\mathbf{x}^{Orig} \in \mathcal{B}$]. Third, we remove the hypothesis

Algorithm 1 | Edge-loss persistent cycle minimization

```
1: Compute a persistent homology basis $\mathcal{B}$ for homology in dimension 1, with
   coefficients in $\mathbb{Q}$, using the standard matrix decomposition procedure
   described in the **Supplementary Material**. Arrange the elements of $\mathcal{B}$ into
   an ordered sequence $Z^0 = (\mathbf{z}^{0,1}, \ldots, \mathbf{z}^{0,m})$.
2: for $j = 0, \ldots, m-1$ do
3:    Solve the program in **Eq. 14** to optimize the $j+1$th element of $Z^j$. Let $\mathbf{x}$ denote the
      solution to this problem, and define $Z^{j+1}$ by replacing the $j+1$th element of $Z^j$ with
      $\mathbf{x}$. Concretely, $\mathbf{z}^{j+1,j+1} = \mathbf{x}$, and $\mathbf{z}^{j+1,k} = \mathbf{z}^{j,k}$ for $k \neq j$.
4: end for
5: Return $\mathcal{B}^* := \{\mathbf{z}^{m,1}, \ldots, \mathbf{z}^{m,m}\}$, the set of elements in $Z^m$.
```

of a simplex-wise filtration; this requires some technical modifications, whose motivation is explained in the **Supplementary Material**. The crux of this modification lies with the for loop, which replaces cycles that have been optimized in the cycle basis for later cycle optimization.

The program in **Eq. 14** optimizes the jth element of an ordered sequence of cycle representatives $Z = (\mathbf{z}^1, \ldots, \mathbf{z}^m)$. In particular, it seeks to minimize $\mathbf{x}^{Orig} := \mathbf{z}^j$. To define this program, we first construct a matrix A such that $A[:, i] = \mathbf{z}^i$ for $i = 1, \ldots, m$. We then define three index sets, $\mathcal{P}$, $\mathcal{Q}$, $\mathcal{R}$ such that

$$\begin{aligned} \mathcal{P} &= \{i : \text{Birth}(\mathbf{z}^i) \leq \text{Birth}(\mathbf{x}^{Orig}),\ \text{Death}(\mathbf{z}^i) \leq \text{Death}(\mathbf{x}^{Orig}),\ i \neq j\} \\ \mathcal{Q} &= \{\tau \in \mathbf{S}_{n+1}(K) : \text{Birth}(\tau) \leq \text{Birth}(\mathbf{x}^{Orig})\} \\ \mathcal{R} &= \{\sigma \in \mathbf{S}_n(K) : \text{Birth}(\sigma) \leq \text{Birth}(\mathbf{x}^{Orig})\}, \end{aligned}$$

That is, $\mathcal{P}$ indexes the set of cycles $\mathbf{z}^i$ such that $\mathbf{z}^i$ is born (respectively, dies) by the time that $\mathbf{z}^j$ is born (respectively, dies), excluding the original cycle $\mathbf{z}^j$ itself. Set $\mathcal{Q}$ is the family of triangles born by Birth $(\mathbf{x}^{Orig})$, and set $\mathcal{R}$ is the family of edges born by Birth $(\mathbf{x}^{Orig})$.

With these definitions in place, we now formalize the general edge-loss problem as the program in **Eq. 14**, where $\partial_{n+1}[\mathcal{R}, \mathcal{Q}]$ denotes the submatrix of ∂_{n+1} indexed by triangles born by Birth $(\mathbf{x}^{Orig})$ (along columns) and edges indexed by edges born by Birth $(\mathbf{x}^{Orig})$. Likewise $A[\mathcal{R}, \mathcal{P}]$ is the column submatrix of A corresponding to cycles that are born before the birth time of $\mathbf{x}^{Orig}$ (and which die before the death time of $\mathbf{x}^{Orig}$), excluding $\mathbf{x}^{Orig}$ itself.

$$\begin{aligned} \text{minimize} \quad & \|W\mathbf{x}\|_1 = \sum_{\sigma \in R} \mathrm{W}[\sigma,\sigma]\left(x^+_\sigma + x^-_\sigma\right) \\ \text{subject to} \quad & (\mathbf{x}^+ - \mathbf{x}^-) = \mathbf{x}^{Orig}[\mathcal{R}] + \partial_{n+1}[\mathcal{R}, \mathcal{Q}]\mathbf{q} + A[\mathcal{R}, \mathcal{P}]\mathbf{p} \\ & \mathbf{p} \in \mathbb{Q}^{\mathcal{P}} \\ & \mathbf{q} \in \mathbb{Q}^{\mathcal{Q}} \\ & \mathbf{x} \in G^{\mathcal{R}} \\ & \mathbf{x}^+, \mathbf{x}^- \geq 0. \end{aligned} \tag{14}$$

Recall from **Section 4.1** that this program varies along two parameters (integrality and weighting). In integral programs $G = \mathbb{Z}$, whereas in nonintegral programs $G = \mathbb{Q}$. The weight matrix W is always diagonal, but in uniform-weighted programs $W[\sigma, \sigma] = 1$ for all $\sigma = R$, whereas in length-weighted programs $W[\sigma, \sigma]$ is the length of edge σ. The program in **Eq. 14** thus results in four variants:

Edge^{NI}_{Unif}: Nonintegral edge-loss with uniform weights.
Edge^{I}_{Unif}: Integral edge-loss with uniform weights.

[9]These notions make sense due to our use of coefficient field $\mathbb{Q}$. The distance used to form a simplicial complex can be used to define length. We restrict our attention of area to points in Euclidean space.

Edge^{NI}_{Len}: Nonintegral edge-loss with edges weighted by length.
Edge^{I}_{Len}: Integral edge-loss with edges weighted by length.

The program in **Eq. 14** may have many more variables than needed, because ∂_{n+1} is often highly singular. Indeed, in applications, ∂_{n+1} can have hundreds or thousands of times as many columns as rows!

A simple means to reduce the size of the program in **Eq. 14**, therefore, is to replace $\mathcal{Q}$ with a subset $\widehat{\mathcal{Q}} \subseteq \mathcal{Q}$ such that $\partial_{n+1}[\mathcal{R}, \widehat{\mathcal{Q}}]$ is a column basis for $\partial_{n+1}[\mathcal{R}, \mathcal{Q}]$. Replacing $\mathcal{Q}$ with $\widehat{\mathcal{Q}}$ will not change the space of feasible values for $\mathbf{x}$ in the program in **Eq. 14**, but it can cut the number of decision variables significantly. In particular, one may take $\widehat{\mathcal{Q}} := \{\sigma : R[:, \sigma] \neq 0\}$ in the $R = \partial_{n+1} V$ decomposition of ∂_{n+1} described in the **Supplementary Material**. We also show correctness of this choice of $\widehat{\mathcal{Q}}$ there.

4.3 Triangle-Loss Methods

Our approach to triangle-loss optimization is essentially that of Obayashi (2018), plus a preprocessing step that converts more general problem data into the simplex-wise filtration format assumed in Obayashi (2018). There are several noteworthy methods for time and memory performance enhancement developed in Obayashi (2018), which we do not implement (e.g., using restricted neighborhoods $\mathcal{F}^{(r)}_q$ to reduce problem size), but which may substantially improve runtime and memory performance.

The original method makes the critical assumption that $K_\bullet$ is a simplex-wise filtration, more precisely, that there exists a linear order $\sigma_1 \leq \cdots \leq \sigma_{|K|}$ such that $K_i = \{\sigma_1, \ldots, \sigma_i\}$. This hypothesis allows one to map each finite-length interval $[i, j) \in \mathrm{Barcode}_n(K_\bullet)$ to a unique pair of simplices (σ_i, σ_j), called a *birth/death pair*, where $\sigma_i \in \mathbf{S}_n(K)$ and $\sigma_j \in \mathbf{S}_{n+1}(K)$. This mapping makes it possible to formulate the program in **Eq. 12**. Unlike the general edge-loss the program in **Eq. 13**, one can formulate the program in **Eq. 12** without ever needing to choose an initial (non-optimal) cycle. Thus, for simplex-wise filtrations, the method of Obayashi (2018) has the substantial advantage of being "parameter free."

However, in many applied settings the filtration $K_\bullet$ is not simplex-wise. Indeed, even accessing information about the filtration can be difficult in modern workflows. Such is the case, for example, for the filtered Vietoris-Rips (VR) construction. In many VR applications, the user presents raw data in the form of a point cloud or distance matrix to a "black box" solver; the solver returns the barcode without ever exposing information about the filtered complex to the user. Thus, the problem of mapping intervals back to pairs of simplices has practical challenges in common applied settings.

To accommodate this more general form of problem data, we employ **Algorithm 2**. This procedure works by (implicitly) defining a simplex-wise refinement $K'_\bullet$ of $K_\bullet$, applying the method of Obayashi (2018) to this refinement, then extracting a persistent homology cycle basis for the subspace of finite intervals from the resulting data. More details, including recovery of a complete persistent homology cycle basis with infinite intervals,[10] and a proof of correctness can be found in the **Supplementary Material**.

Algorithm 2 | Triangle-loss persistent cycle minimization.

```
1: Place a filtration-preserving linear order $\leq^{(l)}$ on $\mathbf{S}_l(K)$ for each $l$.
2: Compute an $R = \partial_{n+1}V$ decomposition as described in (Cohen-Steiner et al., 2006) and the Supplementary Material. We then obtain a set $\Gamma$ of birth/death pairs $(\sigma, \tau)$.
3: For each $(\sigma, \tau) \in \Gamma$ such that $\mathrm{Birth}(\sigma) < \mathrm{Birth}(\tau)$, put
       $\mathcal{F}_n := \{\sigma' \in \mathbf{S}_n(K) : \mathrm{Birth}(\sigma') \leq \mathrm{Birth}(\tau),\ \sigma \leq^{(n)} \sigma'\}$
       $\mathcal{F}_{n+1} := \{\tau' \in \mathbf{S}_{n+1}(K) : \mathrm{Birth}(\sigma) \leq \mathrm{Birth}(\tau'),\ \tau' \leq^{(n+1)} \tau\}$,
   and $\widehat{\mathcal{F}}_{n+1} := \mathcal{F}_{n+1} \cup \{\tau\}$. Compute a solution to the corresponding program in Eq. 15, and denote this solution by $\mathbf{v}^{\sigma,\tau}$
4: Put $\widehat{\mathcal{D}} := \{\partial_{n+1}(\mathbf{v}^{\sigma,\tau}) : (\sigma, \tau) \in \Gamma \text{ and } \mathrm{Birth}(\sigma) < \mathrm{Birth}(\tau)\}$ and let
   $\widehat{\mathcal{D}}' := \{\mathbf{z} \in \mathcal{M} : \mathrm{Death}(\mathbf{z}) = \infty\}$, where $\mathcal{M}$ is a persistent homology cycle basis calculated by the standard $R = DV$ method.
5: Return $\mathcal{D} := \widehat{\mathcal{D}} \cup \widehat{\mathcal{D}}'$.
```

A key component of **Algorithm 2** is the program in **Eq. 15**, which we refer to as the triangle-loss program.

$$
\begin{aligned}
\text{minimize} \quad & \|W\mathbf{v}\|_1 = \sum_{i=1}^{n} W[\gamma, \gamma]\left(\mathbf{v}^+_\gamma + \mathbf{v}^-_\gamma\right) \\
\text{subject to} \quad & \partial_{n+1}[\sigma, \widehat{\mathcal{F}}_{n+1}]\mathbf{v} \neq 0 \\
& \partial_{n+1}[\mathcal{F}_n, \widehat{\mathcal{F}}_{n+1}]\mathbf{v} = 0 \\
& \mathbf{v}_\tau = 1 \\
& \mathbf{v}^+, \mathbf{v}^- \geq 0 \\
& \mathbf{v}^+, \mathbf{v}^- \in G^{\widehat{\mathcal{F}}_{n+1}}.
\end{aligned}
\tag{15}
$$

This terminology is motivated by the special case $n = 1$, which is our focus for empirical studies. As with the general edge-loss program in **Eq. 15** varies along two parameters (integrality and weighting). In integral programs $G = \mathbb{Z}$, whereas in nonintegral programs $G = \mathbb{Q}$. The weight matrix W is always diagonal, but in uniform-weighted programs $W[\gamma, \gamma] = 1$ for all γ, whereas in *area*-weighted programs $W[\gamma, \gamma]$ is the area of triangle γ.[11] The program in **Eq. 15** thus results in four variants:

Tri^{NI}_{Unif}: Nonintegral triangle-loss with uniform weights.
Tri^{I}_{Unif}: Integral triangle-loss with uniform weights.
Tri^{NI}_{Area}: Nonintegral triangle-loss with edges weighted by area.
Tri^{I}_{Area}: Integral triangle-loss with edges weighted by area.

Remark 4.1. Algorithm 2 offers an effective means to apply the methods of Obayashi (2018) to some of the most common data sets in TDA. However, this is done at the cost of parameter-dependence; in particular, outputs depend on the choice of linear orders $\leq^{(l)}$. A brief discussion on how the choice of a total order $\leq$ in **Algorithm 2** may impact the difficulty of the linear programs one must solve is

[10]Recall volume is undefined for infinite intervals.

[11]We compute the area of a 2-simplex using Heron's Formula. We calculate area only for VR complexes whose vertices are points in Euclidean space, though more general metrics could also be considered.

discussed in the **Supplementary Material**. In particular, we explain why the total order implicitly chosen in **Algorithm 2** is reasonable, from a computational/performance standpoint.

4.4 Acceleration Techniques

We consider acceleration techniques to reduce the computational costs of the programs in **Eqs 14**, **15**.

4.4.1 Edge-Loss Methods

The technique used for edge-loss problems aims to reduce the number of decision variables in the program in **Eq. 14**. It does so by replacing a (large) set of decision variables indexed by $\mathcal{Q}$ with a much smaller set, $\widehat{\mathcal{Q}}$. See **Section 4.2** for details.

4.4.2 Triangle-Loss Methods

When ∂_n is large, the memory and computation time needed to construct the constraint matrix $\partial_{n+1}[\mathcal{F}_n, \widehat{\mathcal{F}}_{n+1}]$ can be nontrivial. In applications that require an optimal representative for every interval in the barcode, these costs can be incurred for hundreds or even thousands of programs. We consider two ways to generate the constraint matrices $\partial_{n+1}[\mathcal{F}_n, \widehat{\mathcal{F}}_{n+1}]$ for each of the intervals in a barcode: build $\partial_{n+1}[\mathcal{F}_n, \widehat{\mathcal{F}}_{n+1}]$ from scratch for each program, or build the complete boundary matrix ∂_{n+1} in advance; rather than recompute block submatrices for each program, we pass a slice of the complete matrix stored in memory.

The difference between these two techniques can be seen as a speed/memory tradeoff. As we will see in **Section 6.2**, the first approach is generally faster to optimize the entire basis of homology cycle representatives, but when the data set is large, the full boundary matrix ∂_{n+1} may be too large to store in memory.

5 EXPERIMENTS

In order to address the questions raised in **Section 1**, we conduct an empirical study of minimal homological cycle representatives in dimension one—as defined by the optimization problems detailed in **Section 4** — on a collection of point clouds, which includes both real world data sets and point samples drawn from four common probability distributions of varying dimension.

5.1 Real-World Data Sets

We consider 11 real world data sets from Otter et al. (2017), a widely used reference for benchmark statistics concerning persistent homology computations. There are 13 data sets considered by Otter et al. (2017), however, one of them (gray-scale image) is not available, and one of them is a randomly generated data set similar to our own synthetic data. We summarize information about the dimension, number of points, persistence computation time of each point cloud in **Table 1**. Below we provide brief descriptions of each data set, but we refer the interested reader to Otter et al. (2017) for further details.[12]

[12]We use the distance matrices found on the associated github page (Otter et al., 2017b), except in two cases. For the **Vicsek** data, we use a distance to account for the intended periodic boundary conditions of the model, and for the **genome** data, we use Euclidean distance as the distance matrix in Otter et al. (2017b) resulted in an integer overflow error.

1. Vicsek biological aggregation model. The Vicsek model is a dynamical system describing the motion of particles. It was first introduced in Vicsek et al. (1995) and was analyzed using PH in Topaz et al. (2015). We consider a snapshot in time of a single realization of the model with each point specified by its (x, y) position and heading. To compute distances, the positions and headings are scaled to be between 0 and 1, and then distance is calculated on the unit cube with periodic boundary conditions. The distance between a and b is computed as $\min\{d(a, q) : q\text{-}b \in \{0, 1, \text{-}1\}^3\}$. We denote this data by **Vicsek**.
2. Fractal networks. These networks are self-similar and are used to explore the connection patterns of the cerebral cortex (Sporns, 2006). The distances between nodes in this data set are defined uniformly at random by Otter et al. (2017). In another data set, the authors of Otter et al. (2017) define distances between nodes by using linear weight-degree correlations. We consider both data sets and found the results to be similar. Therefore, we opt to use the one with distances defined uniformly at random. We denote this data set by **Fract R**.
3. C.elegans neuronal network. This is an undirected network in which each node is a neuron, and edges represent synapses. It was studied using PH in Petri et al. (2013). Each nonzero edge weight is converted to a distance equal to its inverse by Otter et al. (2017). We denote this data by ***C.elegans***.
4. Genomic sequences of the HIV virus. This data set is constructed by taking $1,088$ different genomic sequences of dimension 673. The aligned sequences were studied using PH in Chan et al. (2013) with sequences retrieved from (Los Alamos National Laboratory, 2021). Distances are defined using the Hamming distance, which is equal to the number of entries that are different between two genomic sequences. We denote this data by **HIV**.
5. Genomic sequences of H3N2. This data set contains $1,173$ genomic sequences of H3N2 influenza in dimension $2,722$. Distances are defined using the Hamming distance. We denote this data set as **H3N2**.
6. Human genome. This is a network representing a sample of the human genome studied using PH in Petri et al. (2013), which was created using data retrieved from Davis and Hu (2011). Distances are measured using Euclidean distance. We denote this data set by **Genome**.
7. U.S. Congress roll-call voting networks. In the two networks below, each node represents a legislator, and the edge weight is a number in $[0, 1]$ representing the similarity of the two legislators' past voting decisions. Distance between two nodes i, j are defined to be $1\text{-}w_{i,j}$.
    1. **House**. This is a weighted network of the House of Representatives from the 104th United States Congress.
    2. **Senate**. This is a weighted network of the Senate from the 104th United States Congress.
8. Network of network scientists. This data set represents the largest connected component of a collaboration network of network scientists (Newman, 2006). The edge weights indicate the number of joint papers between two authors. Distances are defined as the inverse of edge weight. We denote this data set by **Network**.

9. Klein. The Klein bottle is a non-orientable surface with one side. This data set was created in Otter et al. (2017a) by linearly sampling 400 points from the Klein bottle using its "figure 8" immersion in $\mathbb{R}^3$. This data set originally contains (Borradaile et al., 2020) duplicate points, which we remove. Distances are measured using the Euclidean distance. We denote this data set by **Klein**.
10. Stanford Dragon graphic. This data set contains 1,000 points sampled uniformly at random by Otter et al. (2017) from 3-dimensional scans of the dragon (Stanford University Computer Graphics Laboratory, 1999). Distances are measured using the Euclidean distance. We denote this data set **Drag**.

5.2 Randomly Generated Point Clouds

We also generate a large corpus of synthetic point clouds, each containing 100 points in $\mathbb{R}^q$ with $q = 2, \ldots, 10$, drawn from normal, exponential, gamma, and logistic distributions. We produce 10 realizations for each distribution and dimension combination, for a total of 360 randomly generated point clouds. We use Euclidean distance to measure similarity between points and the Vietoris- Rips filtered simplicial complex to compute persistent homology.

5.3 Erdős-Rényi Random Complexes

To investigate which properties of homological cycle representatives could arise as the result of the underlying geometry of the point clouds, we also consider a common non-geometric model for random complexes: Erdős-Rényi random clique complexes. Here, we construct 100 symmetric dissimilarity matrices of size 100 × 100 by drawing entries i. i.d. from the uniform distribution on $[0, 1]$ for each pair of distinct points. As these dissimilarities are fully independent, they are in particular not subject to geometric constraints like the triangle inequality. A natural filtration is placed on these dissimilarity matrices by forming filtered simplicial complex $K_{\bullet} = (K_{\varepsilon_i})_{i\in\{1,\ldots,T\}}$ where $0 = \varepsilon_1 < \cdots < \varepsilon_T = 1$ to compute persistent homology.

5.4 Computations

For each of the data sets, we perform **Algorithms 1**, **2** (using Vietoris-Rips complexes with $\mathbb{Q}$ − coefficients) to find optimal bases $\mathcal{B}^*, \mathcal{D} \in$ PrsHCB. For comparison to the edge-loss problem in **Algorithm 1**, we also apply the program in **Eq. 8** to each representative in the persistent homology cycle basis to find a basis $\mathcal{C} \in$ FCB.

5.5 Hardware and Software

We test our programs on an iMac (Retina 5K, 27-inch, 2019) with a 3.6 GHz Intel Core i9 processor and 40 GB 2667 MHz DDR4 memory.

Software for our experiments is implemented in the programming language Julia; source code is available at Li and Thompson (2021). This code specifically implements **Algorithms 1**, **2** and the program in **Eq. 8**.[13]

Since our interest lies not only with the outputs of these algorithms but with the structure of the linear programs themselves Li and Thompson (2021), implements a standalone workflow that exposes the objects built internally within each pipeline. This library is simple by design, and does not implement the performance-enhancing techniques developed in Escolar and Hiraoka (2016) and Obayashi (2018). Users wishing to work with optimal cycle representatives for applications may consider these approaches discussed in **Section 3.7**.

To implement **Algorithms 1**, **2** in homological dimension one, the test library (Li and Thompson, 2021) provides three key functions: *A novel solver for persistence with $\mathbb{Q}$ − coefficients*. To compute cycle representatives for persistent homology with $\mathbb{Q}$ − coefficients, we implement a new persistent homology solver adapted from Eirene (Henselman-Petrusek, 2016). The adapted version uses native Eirene code as a subroutine to reduce the number of columns in the top dimensional boundary matrix in a way that is guaranteed not to alter the outcome of the persistence computation (Henselman and Ghrist, 2016).

Formatting of Inputs to Linear Programs. Having computed barcodes and persistent homology cycle representatives, library (Li and Thompson, 2021) provides built-in functionality to format the linear the program in **Eqs 14**, **15** for input to a linear solver. This "connecting" step is executed in pure Julia.

Wrappers for Linear Solvers. We use the Gurobi linear solver (Gurobi Optimization, 2020) and the GLPK solver (GNU Project, 2012). Both solvers can optimize both LPs and MIPs. Experiments indicate that Gurobi executes much faster than GLPK on this class of problems, and thus, we use it in the bulk of our computations. Both solvers are free for academic users.

6 RESULTS AND DISCUSSION

In this section, we investigate each of the questions raised in **Section 1** with the following analyses.

6.1 Computation Time Comparisons

We summarize results for Programs Edge^{NI}_{Unif}, Edge^{I}_{Unif}, Edge^{NI}_{Len}, Edge^{I}_{Len}, Tri^{NI}_{Unif}, and Tri^{I}_{Unif} in **Table 1** for data described in **Section 5.1** and **Table 2** for data described in **Section 5.2** and **Section 5.3**. Further, we summarize results for Programs Tri^{NI}_{Area} and Tri^{I}_{Area} in **Table 2** for data described in **Section 5.2**.[14] We use $T_{\textbf{persistence}}$ to denote the time taken to compute all original cycle representatives and their lifespans $\mathcal{L}$. We use $T^*_{\bullet}$ to denote the computation time for optimizing all generators found by the persistence algorithm, where the subscript denotes the cost function e.g. *E-Unif* or *T-Unif*, and the superscript denotes the nonintegral ^{NI} or integral ^{I} constraint. The $T^*_{\bullet}$ computations include the time required to construct the inputs to the solver for the edge-loss methods, and exclude the time required to construct the inputs to the triangle-loss methods, whose computation time is separately recorded in

[13]The program in **Eq. 8** is implemented analogously.

[14]We compute the area of a 2-simplex using Heron's Formula for data whose distances are measured using the Euclidean distance. For data with non-Euclidean distances, we find that there are triangles that do not obey the triangle inequality, thus, we only compute area-weighted triangle-loss cycles for data described in **Section 5.2**. As such, Tri^{NI}_{Area}, Tri^{NI}_{Area} do not appear in **Table 1** and the Erős-Rényi column of **Table 2**.

TABLE 1 | Summary of the experimental results of the data sets from Otter et al. (2017) as described in **Section 5.1**. The rows include the ambient dimension, number of points, the number of cycle representatives in $\mathbf{H}_1$, and the time (measured in seconds) it took to compute persistent homology for each data set. We also include the computation time taken to optimize the set of cycle representatives under six different optimization problems, and computation time of two different implementation choices for the triangle-loss optimal cycles: building the full ∂_2 boundary matrix once and extracting the part needed, or constructing part of the ∂_2 boundary matrix for each cycle representative. In this table, T stands for computation time measured in seconds with subscripts indicating the type of the optimal cycle and superscripts indicating whether the program was solved using linear programming (NI) or integer programming (I). The time taken to construct the input to the optimization problem is included in the optimization time for edge-loss minimal cycle representatives, but is excluded and separately listed in the last two rows for the triangle-loss minimal cycle representatives. For triangle-loss cycles, we were able to compute 115 out of the 117 cycle representatives for the **Genome** data set and 52 out of 57 cycle representatives for the **H3N2** data set due to memory constraints. The numbers in the parenthesis represent the other optimization statistics corresponding to the triangle-loss optimal cycles we were actually able to compute. The last two rows compare two ways of building the input $\partial_2[:,\widehat{\mathcal{F}}_2]$ matrix to the triangle-loss optimal cycle program. The penultimate row records the time of building the entire ∂_2 matrix once and then extracting columns born in the interval $[b_i, d_i]$ for each representative. The last row records the total time to iteratively build the part of the boundary matrix $\partial_2[:,\widehat{\mathcal{F}}_2]$ for each cycle representative.

	Klein	Vicsek	C.*elegans*	HIV	Genome	Fractal R	Network	House	Senate	Drag	H3N2
Ambient dimension	3	3	202	673	688	259	300	261	60	3	1,173
# Points	400	300	297	1,088	1,397	512	379	445	103	1,000	2,722
# Representatives	257	149	107	174	117 (115)	438	7	126	12	311	28 (26)
$T_{\text{persistence}}$	100.97	129.39	5.14	728.51	967.61	143.07	12.18	9.62	0.10	1,053.53	71,081.77
Edge-loss persistent homological cycle representatives (Eq. 14)											
$T^{I}_{E\text{-}Len}$	16.01	8.20	19.64	466.85	656.05	150.46	0.17	63.93	0.31	45.14	4,732.59
$T^{NI}_{E\text{-}Len}$	11.28	6.61	16.07	403.63	491.69	86.95	0.13	48.65	0.22	34.73	4,540.55
$T^{I}_{E\text{-}Unif}$	14.59	9.09	19.22	473.82	689.51	119.94	0.23	63.34	0.33	45.51	4,714.90
$T^{NI}_{E\text{-}Unif}$	11.38	5.55	15.63	404.95	492.66	83.40	0.12	48.88	0.22	33.88	4,547.37
Edge-loss filtered homological cycle represnetatives (Eq. 8)											
$T^{I}_{E\text{-}Len}$	16.93	8.64	20.41	468.22	1,144.17	155.08	0.17	62.20	0.30	67.77	2,999.24
$T^{NI}_{E\text{-}Len}$	10.29	5.51	16.15	403.74	973.15	88.66	0.13	48.24	0.22	50.25	2,829.12
$T^{I}_{E\text{-}Unif}$	15.14	8.32	19.76	476.84	1,191.44	142.4	0.24	61.82	0.31	68.63	2,937.16
$T^{NI}_{E\text{-}Unif}$	11.07	5.63	16.23	406.97	981.72	87.59	0.12	48.11	0.22	54.05	2,833.06
Triangle-loss persistent homological cycle representatives (Eq. 15)											
$T^{I}_{T\text{-}Unif}$	316.33	24.52	657.53	25,402.56	16,379.86	20,440.33	2.91	234.05	0.29	384.91	39,140.67
$T^{NI}_{T\text{-}Unif}$	154.36	19.18	540.06	23,260.12	14,535.42	18,279.82	2.47	206.63	0.18	277.93	36,401.50
T Build all	2.16	0.32	4.88	268.57	—	138.46	0.06	6.23	0.03	5.94	—
Total T build part	9.18	3.51	28.47	1688.10	415.79	917.42	0.28	45.02	0.05	106.64	1,236.80

order to compare two ways of constructing the input matrix, as discussed in **Section 4.4**. In each table, rows 1–3 provide information about the data by specifying ambient dimension, number of points, and number of cycle representatives. Row 4, labeled as $T_{\mathbf{persistence}}$, gives the total time to compute persistent homology for the data, measured in seconds. Rows 5–12 (**Table 1**) and rows 5–14 (**Table 2**) give the total time to optimize all cycle representatives that are feasible to compute using each optimization technique. In the last two rows of each table, we provide the time of constructing the input to the triangle-loss methods using two different approaches described in **Section 4.4**. The penultimate row records the time of building the entire ∂_2 matrix once and then extracting $\partial_2[\mathcal{F}_1, \widehat{\mathcal{F}}_2]$ for each representative. The last row records the total time to iteratively build the part of the boundary matrix $\partial_2[\mathcal{F}_1, \widehat{\mathcal{F}}_2]$ for each cycle representative. In **Table 2**, the computation times displayed average all random samples from each dimension for each distribution.

The two numbers in parenthesis in the third row of **Table 1** indicate the actual number of representatives we were able to optimize using the triangle-loss methods (all edge-loss representatives were optimized). For the **Genome** and **H3N2** data sets, we are not able to compute all triangle-loss cycle representatives due to the large number of 2-simplices born between the birth and death interval of some cycles. For instance, for a particular cycle representative in the **Genome** data set, there were 10,522,991 2-simplices born in this cycle's lifespan. Also, given the large number of 2-simplices in the simplicial complex, we are not able to build the full ∂_2 matrix due to memory constraints, denoted by - in the penultimate row of **Table 1**.

Below we describe some insights on computation time drawn from the two tables.

6.1.1 Persistence and Optimization $T_{\text{persistence}}$ vs. $T^*_\bullet$

We observe that $T^*_\bullet$[15] $> T_{\mathbf{persistence}}$ e.g. for 5 out of the 11 real-world data sets described in **Section 5.1** when using the four edge-loss methods. The same inequality holds in seven out of the 11 data sets when using the two uniform-weighted triangle-loss methods. For all of the synthetic data described in **Sections 5.2** and **5.3**, we have $T^*_\bullet > T_{\mathbf{persistense}}$ when using all eight optimization programs. Therefore, the computational cost of optimizing a basis of cycle representatives generally exceeds the cost of computing such a basis.

This somewhat surprising result highlights the computational complexity of the algorithms used both to compute persistence and to optimize generators. A common feature of both the persistence computation and linear optimization is that empirical performance typically outstrips asymptotic complexity by a wide margin; the persistence computation, for example, has cubic complexity in the size of the complex, but usually runs in linear time. Thus, worst-case

[15]Including the time of constructing the input to the optimization programs.

TABLE 2 | Summary of the experimental results for the synthetic, randomly generated data sets described in **Section 5.2** and **Section 5.3**. For each distribution, we sample 10 data sets each containing 100 points in ambient dimensions from 2–10. The computation time in this table averages the 10 random samples for each dimension and distribution combination. The number of cycle representatives is totaled over the 90 samples for each distribution. The rows of this table are analogous to those of **Table 1**, excluding the penultimate row of that table, as the time comparison is only done for the large real-world data sets.

	Normal	**Gamma**	**Logistic**	**Exponential**	**Erdős-Rényi**
Ambient dimension	2–10	2–10	2–10	2–10	NA
# Points	100	100	100	100	100
Total # representatives	4,815	3,706	4,456	3,788	34,214
Average $T_{\text{persistence}}$ (seconds)	2.80	2.12	2.01	2.63	2.20
Edge-loss persistent homological cycle representatives (Eq. 14)					
Average total $T^{I}_{E\text{-}Len}$	5.52	6.01	5.65	5.91	5.99
Average total $T^{NI}_{E\text{-}Len}$	4.37	4.55	4.32	4.47	4.99
Average total $T^{I}_{E\text{-}Unif}$	5.31	5.97	5.45	5.90	6.16
Average total $T^{NI}_{E\text{-}Unif}$	4.08	4.58	4.23	4.51	4.87
Edge-loss filtered homological cycle representatives (Eq. 8)					
Average total $T^{I}_{E\text{-}Len}$	5.32	6.46	6.27	6.88	7.44
Average total $T^{NI}_{E\text{-}Len}$	4.07	5.05	4.78	5.11	4.69
Average total $T^{I}_{E\text{-}Unif}$	5.23	6.46	6.25	6.66	6.25
Average total $T^{NI}_{E\text{-}Unif}$	4.17	4.94	4.61	5.29	4.64
Triangle-loss persistent homological cycle representatives (Eq. 15)					
Average total $T^{I}_{T\text{-}Unif}$	6.56	9.91	7.06	9.68	4.64
Average total $T^{NI}_{T\text{-}Unif}$	5.24	7.99	5.79	7.75	4.49
Average total $T^{I}_{T\text{-Area}}$	6.59	10.20	7.30	9.99	—
Average total $T^{I}_{T\text{-Area}}$	5.19	7.89	5.80	7.57	—
Average total T build all	1.40	1.71	1.56	1.07	1.24
Average total T build part	3.51	1.54	1.61	1.56	0.85

complexity paints an incomplete picture. Moreover, naive "back of the envelope" calculations are often hindered by lack of information. For example, the persistence computation (which essentially reduces to Gaussian elimination) typically processes each of the m columns of a boundary matrix ∂_n in sequence. The polytope of feasible solutions for an associated linear program (edge-loss or triangle-loss) may have many fewer or many more vertices than m, depending on the program; moreover, even if the number of vertices is very high, the number of *visited* vertices (e.g., by the simplex algorithm) can be much lower. Without knowing these numbers *a priori*, run times can be quite challenging to estimate. Empirical studies, such as the present one, give a picture of how these algorithms perform in practice.

6.1.2 Integral and Nonintegral Programs $(T^{I}$ vs. $T^{NI})$

In **Tables 1** and **2**, we observe that $(T^{I} > T^{NI})$, i.e., the total computation time of optimizing a basis of cycle representatives using an integer program exceeds the computation time using a non-integer constrained program. Yet, T^{I} and T^{NI} are on the same order of magnitude, for both edge-loss methods and triangle-loss method.

Let $r_{E\text{-}Unif} = \frac{t^{I}_{E\text{-}Unif}}{t^{NI}_{E\text{-}Unif}}$, where $t^{*}_{\bullet}$ represents the computation time for optimizing a single cycle representative. We define $r_{E\text{-}Len}$ and $r_{T\text{-}Unif}$ similarly. We compute each for every cycle representative for data described in **Tables 1**, **2**. Let $\bar{r}_{\bullet}$ denote the average of $r_{\bullet}$ and $\sigma_{r_{\bullet}}$ denote the standard deviation of $r_{\bullet}$. We have $\bar{r}_{E\text{-}Unif} = 1.49, \sigma_{r_{E\text{-}Unif}} = 1.34,$ $\bar{r}_{E\text{-}Len} = 1.55, \sigma_{r_{E\text{-}Len}} = 1.38,$ $\bar{r}_{T\text{-}Unif} = 1.28, \sigma_{r_{T\text{-}Unif}} = 1.14.$ **Figures 6A,C,E** plots $r_{\bullet}$ using scatter plots and **Figures 6B,D,F** displays the same data using box plots. The vertical axis represents the ratio between the MIP time and LP time of optimizing a cycle representative. The horizontal axis in the scatter plots represents the computation time to solve the LP. The red line in each subfigure represents the horizontal line $y = 1$. As we can see from the box plots, the ratio between the computation time of MIP and LP for most of the cycle representatives (>50%) is around 1 and less than 2. Although there are cases where the computation time of solving an MIP is 108.70 times the computation time of solving an LP, such cases happen only for cycle representatives with a very short LP computation time.

6.1.3 Triangle-Loss Vs. Edge-Loss Programs $T_{T\text{-}\bullet}$ vs. $T_{E\text{-}\bullet}$

We observe that the edge-loss optimal cycles are more efficient to compute than the triangle-loss cycles for more than 60.11% of the cycle representatives[16]. This aligns with our intuition because for representatives with a longer persistence, the number of columns in the boundary matrix $\partial_2[\mathcal{F}_1, \widehat{\mathcal{F}}_2]$ grows faster than that of $\partial_1[:, \mathcal{Q}]$. Consequently, the edge-loss programs are feasible for all cycle representatives we experiment with, whereas the triangle-loss technique fails for six representatives due to the large problem size (with greater than twenty million triangles born between the life span of those cycle representatives).

6.1.4 Different Linear Solvers

The choice of linear solver can significantly impact the computational cost of the optimization problems. We perform experiments on length/uniform-minimal cycle representatives using the GLPK (GNU Project, 2012; Gurobi Optimization,

[16]Obayashi (2018) proposes a few techniques for accelerating the triangle-loss methods which we did not implement.

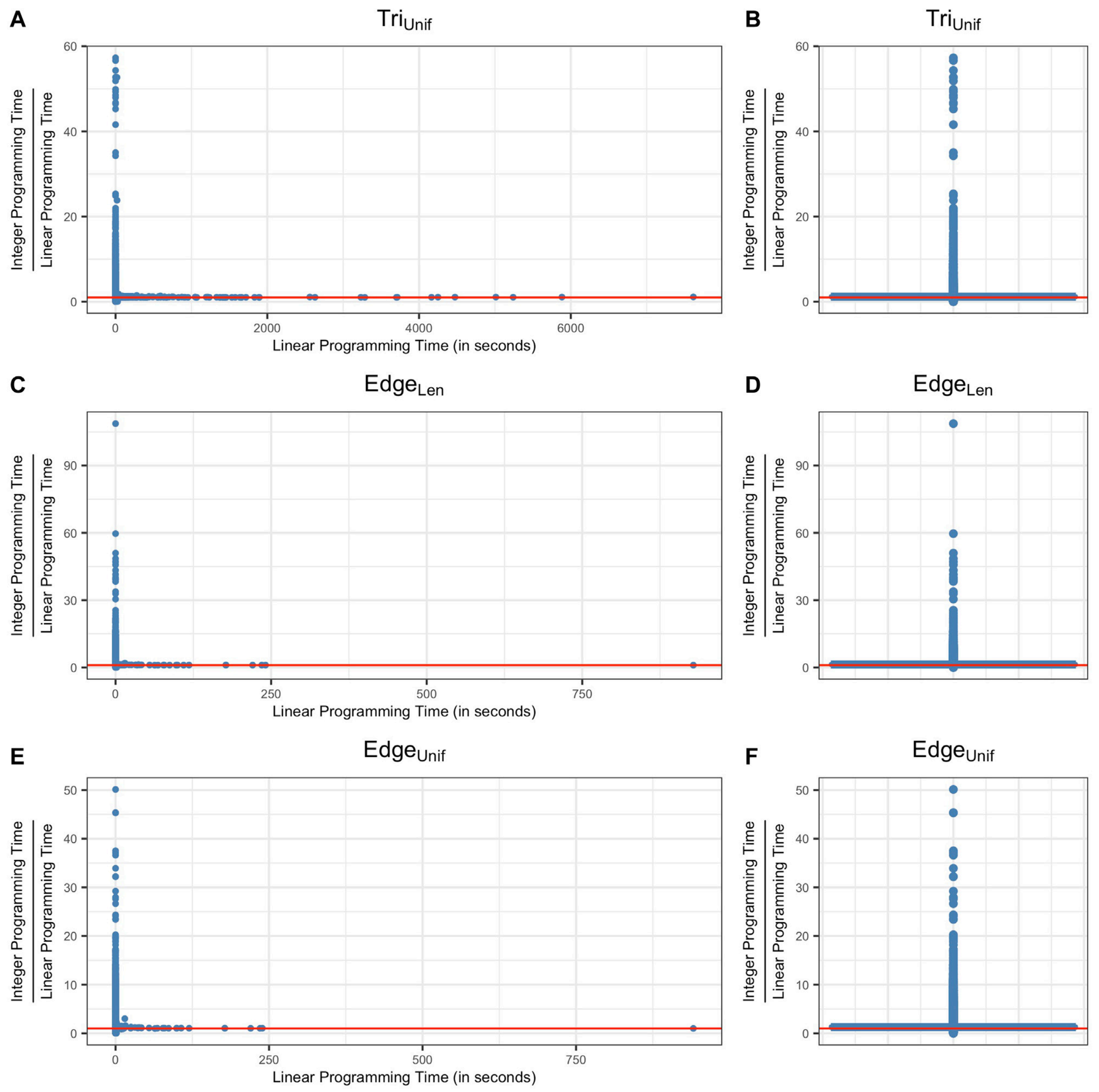

FIGURE 6 | Ratio between the computation time of solving an integer programming problem (Programs Tri^{I}_{Unif}, $Edge^{I}_{Len}$, $Edge^{NI}_{Unif}$) and the computation time of solving a linear programming problem (Programs Tri^{NI}_{Unif}, $Edge^{NI}_{Len}$, $Edge^{NI}_{Unif}$) for all of the cycle representatives from data sets described in **Sections 5.1**, **5.2** and **5.3**. Subfigures **(A) (C) (E)** plot the data using scatter plots and subfigures **(B) (D) (F)** show the same data using box plots. The vertical axis represents the ratio between the integer programming time and linear programming time of optimizing a cycle representative and the horizontal axis represents the computation time to solve the linear program. The red line in each subfigure represents the horizontal line $y = 1$, where the time for each optimization is equivalent. As we can see from the box plots, the ratio between the computation time of integer programming and linear programming for most of the cycle representatives (>50%) center around 1.

2020) linear solvers on 90 data sets drawn from the normal distribution with dimensions from 2 to 10 with a total of 4, 815 cycle representatives. The median of the computation time ratio between using the GLPK solver and Gurobi solver is 2.22 for Program $Edge^{NI}_{Unif}$, 1.68 for Program $Edge^{I}_{Unif}$, 2.28 for Program $Edge^{NI}_{Len}$, and 1.73 for Program $Edge^{I}_{Len}$, and the computation time using the GLPK solver can be 30 times larger than the computation time using the Gurobi solver for some cycles, see figure in the **Supplementary Material**. Therefore, we use the Gurobi solver in all other analyses in this paper.

TABLE 3 | Computation time of three differently sized input boundary matrices to edge-loss and triangle-loss methods. The superscripts denote whether the program requires an integral solution or not, and the subscripts indicate the type of optimal cycle. All time is measured in seconds. We perform experiments on a small-sized data set (**Senate**) that consists of 103 points in dimension 60 and a medium-sized data set (**House**) that contains 445 points in dimension 261. For edge-loss methods, we consider three implementations to solve these optimization problems: using the full boundary matrix ∂_2, using the basis columns and all rows $\partial_2[:,\hat{\mathcal{Q}}]$, and using the basis columns and deleting rows corresponding to edges born after the birth time of the cycle $\partial_2[\mathcal{R},\hat{\mathcal{Q}}]$. For triangle-loss methods, we consider three approaches to solve these optimization problems: zeroing out the columns in the boundary matrix outside of $[b_i, d_i]$ denoted as $\partial_{2_{zero}}$, deleting columns outside of this range $\partial_2[:,\hat{\mathcal{F}}_2]$, and deleting both columns outside of $[b_i, d_i]$ and rows born after d_i denoted $\partial_2[\mathcal{F}_1,\hat{\mathcal{F}}_2]$. The **House** data set was too large to implement the first method.

	Edge-loss Optimal Cycles (Eq. 14)			
	T	∂_2	$\partial_2[:,\hat{\mathcal{Q}}]$	$\partial_2[\mathcal{R},\hat{\mathcal{Q}}]$
Small Data Set (Senate)	$T^{NI}_{E\text{-}Unif}$	1.06	1.03	0.41
	$T^{I}_{E\text{-}Unif}$	1.25	1.23	0.60
	$T^{NI}_{E\text{-}Len}$	1.05	1.05	0.41
	$T^{I}_{E\text{-}Len}$	1.23	1.19	0.65
Medium Data Set (House)	$T^{NI}_{E\text{-}Unif}$	184.70	122.72	47.10
	$T^{I}_{E\text{-}Unif}$	188.88	147.27	64.64
	$T^{NI}_{E\text{-}Len}$	184.41	121.80	46.02
	$T^{I}_{E\text{-}Len}$	193.01	146.46	63.87

	Triangle-loss Optimal Cycles (Eq. 15)			
	T	$\partial_{2_{zero}}$	$\partial_2[:,\hat{\mathcal{F}}_2]$	$\partial_2[\mathcal{F}_1,\hat{\mathcal{F}}_2]$
Small Data Set (Senate)	$T^{NI}_{T\text{-}Unif}$	21.37	0.64	0.18
	$T^{I}_{T\text{-}Unif}$	24.51	0.86	0.29
Medium Data Set (House)	$T^{NI}_{T\text{-}Unif}$	—	297.34	203.63
	$T^{I}_{T\text{-}Unif}$	—	321.31	234.05

6.2 Performance of Acceleration Techniques

6.2.1 Edge-Loss Optimal Cycles

As discussed in **Section 4.4**, we accelerate edge-loss problems by replacing $\partial_2[:,\mathcal{Q}]$ with the column basis submatrix of $\partial_2[:,\hat{\mathcal{Q}}]$. We further reduce the size of $\partial_2[:,\hat{\mathcal{Q}}]$ by only including the rows corresponding to 1-simplices born before the birth time of the cycle, denoted as $\partial_2[\mathcal{R},\hat{\mathcal{Q}}]$. We perform experiments on a small-sized data set (**Senate**) that consists of 103 points in dimension 60 and a medium-sized data set (**House**) that contains 445 points in dimension 261. In **Table 3**, we report the computation time of solving the optimization problems in Programs Edge^{NI}_{Unif}, Edge^{I}_{Unif}, Edge^{NI}_{Len}, and Edge^{I}_{Len} using these three techniques of varying the size of the input boundary matrix. The results align with intuition that the optimizations are faster with fewer input variables, and thus, the third implementation is the most efficient among the three.

6.2.2 Triangle-Loss Optimal Cycles

As discussed in **Section 4.4**, there are also multiple approaches to creating the input to the triangle-loss problems. To recap, we restrict the boundary matrix ∂_2 to $\partial_2[\mathcal{F}_1,\hat{\mathcal{F}}_2]$ for a particular cycle representative $\mathbf{x}^i$. We can do so in various ways: 1) zeroing out the columns of ∂_2 not in $\hat{\mathcal{F}}_2$ but maintaining the original size of the boundary matrix, 2a) building the entire boundary matrix ∂_2 once and then deleting the columns not in $\hat{\mathcal{F}}_2$ for each representative, 2b) building the columns in $\hat{\mathcal{F}}_2$ iteratively for each representative, and 3a/b) in conjunction with 2a) or 2b) respectively, reducing the rows of the boundary matrix of ∂_2 to only include the rows born before the death time of the cycle $\mathcal{F}_1$.

In **Table 3**, we summarize the computation time of solving Programs Tri^{NI}_{Unif} and Tri^{I}_{Unif} to find triangle-loss optimal cycles with three different sized boundary matrices as input: 1) zeroing out, 2b) deleting partial columns, and 3b) deleting partial rows and columns. Note that 2a) and 2b) both result in the same boundary matrix $\partial_2[:,\hat{\mathcal{F}}_2]$. We again use the **Senate** and **House** data sets for analysis. We see that deleting partial rows and columns is the most efficient among the three implementations, which again matches intuition that reducing the number of variables accelerates the optimization problem.

We also ran experiments on the real-world data sets to compare the timing of building $\partial_2[\mathcal{F}_1,\hat{\mathcal{F}}_2]$ via methods 3a) and 3b) and summarize the results in the last two rows of **Table 1**. We find that approach 3a), where we build the entire matrix ∂_2 and then delete columns for each cycle representative, is in general faster than approach 3b), where the boundary matrix $\partial_2[\mathcal{F}_1,\hat{\mathcal{F}}_2]$ is iteratively built for each representative. However, this latter approach can be more useful for large data sets, whose full boundary matrix ∂_2 might be too large to construct. For example, building the full boundary matrix for the **Genome** data set caused Julia to crash due to the large number of 2-simplices (453, 424, 290 triangles for the **Genome** data set and 3, 357, 641, 440 triangles for the **H3N2** data set). Whereas, by implenting 3b) where we rebuild a part of the boundary matrix for each representative, we were able to optimize 115 out of the 117 cycle representatives for the **Genome** data set and 52 of 57 cycle representatives for the **H3N2** data set.

6.3 Coefficients of Optimal Cycle Representatives in Data Sets From Section 5.1 and Section 5.2

As discussed in **Section 3.6**, the problem of solving an ℓ_0 optimization is desirable for its interpretability but doing so is NP-hard (Tahbaz-Salehi and Jadbabaie, 2010). Often, ℓ_0 optimization is approximated by an ℓ_1 optimization problem, which is solvable in polynomial time. If the coefficients of a solution of the ℓ_1 problem are in {-1, 0, 1}, then it is in fact an ℓ_0 solution to the restricted optimization problem where we require solutions to have entries in {-1, 0, 1} (Escolar and Hiraoka, 2016; Obayashi, 2018).

We find that 99.50% of the original, unoptimized cycle representatives obtained from data sets described in **Section 5.1** and 99.91% of the unoptimized cycle representatives obtained from data sets described in **Section 5.2** have coefficients in {-1, 0, 1}. All unoptimized cycle representatives turned out to have integral entries.

We then systematically check each solution of the eight programs Edge^{NI}_{Unif}, Edge^{I}_{Unif}, Edge^{NI}_{Len}, Edge^{I}_{Len}, Tri^{NI}_{Unif}, Tri^{I}_{Unif}, and Tri^{NI}_{Area}, Tri^{I}_{Area} across all data sets and all optimal cycle representatives from data discussed in **Sections 5.1** and **5.2**,[17] found by **Algorithms 1**, **2** and the program in **Eq. 8** to see if the coefficients are integral or in {-1, 0, 1}. We analyze the 18, 163 optimal cycle representatives and find the following consistent results.

All optimal solutions to the program in **Eq. 8** (edge-loss minimization of filtered cycle bases) and all but one of the solutions returned by **Algorithm 1** (edge-loss minimization of persistent cycle bases) had coefficients in {-1, 0, 1}; see the table in the **Supplementary Material** for details. The exceptional representative $\mathbf{x}^{NI}_{E\text{-}Unif}$ occurred in the **C.elegans** data set, with coefficients in {-0.5, 0, 0.5}. It corresponds to one of only a few cases where two intervals with equal birth and death time occur within the same data set; see **Section 6.6**. An interesting consequence of these fractional coefficients is that here, unlike all other cycle representatives from data discussed in **Section 5.1** and **Section 5.2**, the ℓ_0 norm and ℓ_1 norm differ. This accounts for the sole point that lies below the $y = 1$ line in the first column of row (B) in **Figure 8**.

On the one hand, this exceptional behavior could bear some connection to **Algorithm 1**. Recall that **Algorithm 1** operates by removing a sequence of cycles from a cycle basis, replacing each cycle with a new, optimized cycle on each iteration (that is, we swap the $j+1^{th}$ element of Z^j with an optimized cycle $\mathbf{x}$ in order to produce Z^{j+1}). Replacing optimized cycles in the basis is key, since without replacement it would be possible in theory to get a set of optimized cycles that no longer form a basis. We verified that if we modify **Algorithm 1** to skip the replacement step, we achieve {-1, 0, 1} solutions for the exceptional **C.elegans** cycle representative (for the other repeated intervals we obtain the same optimal cycle with and without the replacement). On the other hand, we find that even with replacement the GLPK solver obtains a solution with coefficients in {-1, 0, 1}. Thus, every one of the cases considered produced {-1, 0, 1} coefficients for at least one of the two solvers, and the appearance of fractional coefficients may be naturally tied to the specific implementation of the solver used.

When solving the integral triangle-loss problem by **Algorithm 2**, we obtain two solutions whose boundaries $x = \partial_2 \mathbf{v}$ have coefficients in {-1,0,1,2} for two different cycle representatives from the logistic distribution data set. However, the corresponding solutions **v** of these cycle representatives do have coefficients in {-1,0,1}.

The surprising predominance of solutions in {-1, 0, 1} suggests that in most cases, the modeler can reap both the computational advantage of ℓ_1 solutions and the theoretical and interpretability advantages of ℓ_0 solutions[18] by solving an ℓ_1 optimization problem. Further, we find that the optimum cost is the same whether we require an integer solution or not for more than 99.97% of solutions to Program Edge_{Len}, 100% of solutions to Edge_{Unif}, and 100% of solutions to Tri_{Unif}. Thus, the modeler can drop the integral constraint to save computation time while still being able to achieve an integral solution in most cases.

6.4 Comparing Optimal Cycle Representatives Against Different Loss Functions

We compare the optimal cycle representatives against different loss functions to study the extent to which the solutions produced by each technique vary. We consider two loss functions on an H_1 cycle representative $\mathbf{x} \in \mathbf{Z}_1(K)$:

$$L_{E\text{-}Len}(\mathbf{x}) = \sum_{\sigma \in \text{supp}(\mathbf{x})} \text{Length}(\sigma),$$

where $\text{Length}(\sigma)$ is the distance—as designated by the metric d used to define the VR complex—between the two vertices of a 1-simplex σ, and

$$L_{E\text{-}Unif}(\mathbf{x}) = \|\mathbf{x}\|_0 = |\text{supp}(\mathbf{x})|,$$

the number of 1-simplices (edges) in a representative.

We also consider two loss functions on 2-chains $\mathbf{v} \in \mathbf{C}_2(K)$, namely area-weighted loss:

$$L_{T\text{-}Area}(\mathbf{v}) = \sum_{\tau \in \text{supp}(\mathbf{v})} \text{Area}(\tau),$$

where $\text{Area}(\tau)$ is the area of a 2-simplex as computed by Heron's Formula, and uniform-weighted loss

$$L_{T\text{-}Unif}(\mathbf{v}) = \|\mathbf{v}\|_0 = |\text{supp}(\mathbf{v})|.$$

Remark 6.1. These weighted ℓ_0 loss functions differ from the objective functions used in the optimization problems presented in **Section 4**, which measure weighted ℓ_1 norm. However, weighted ℓ_0 norm and weighted ℓ_1 norm agree on solutions with {-1, 0, 1} coefficients, and (as reported in **Section 6.3**) nearly all cycle representatives for **Sections 5.1** and **5.2** data satisfy this condition, both pre- and post-optimization.In the special case where $\text{supp}(\mathbf{x})$ determines a simple closed polygonal curve c with vertices $(p^1, q^1), \ldots, (p^n, q^n) \in \mathbb{R}^2$, we also use the Surveyor's Area Formula (Braden, 1986) to quantify area of **X** as

$$L_{Sur\text{-}Area}(c) = \frac{1}{2}\left|\sum_{i=1}^{n} p^i q^{i+1} - p^{i+1} q^i\right|,$$

where, by convention, $p^{i+1} = p^1$ and $q^{i+1} = q^1$. We evaluate this function only when (i) the ambient point cloud of the VR complex is a subset of $\mathbb{R}^2$, b) $\text{supp}(\mathbf{x})$ forms a graph-theoretic cycle when regarded as a subset of edges in the combinatorial graph formed by 1-skeleton of K, and 3) no pair of distinct closed line segments intersect one another.

In the case when we compute the loss function of a corresponding optimal solution, we use the notation for the cost $C^*_\bullet := L_\bullet(\mathbf{x}^*_\bullet)$ to an edge-loss problem that finds optimal solution $\mathbf{x}^*_\bullet$, and $C^*_\bullet := L_\bullet(\mathbf{v}^*_\bullet)$ to a triangle-loss problem that finds optimal solution $\mathbf{v}^*_\bullet$. For instance, $C^{NI}_{E\text{-}Unif} = L_{E\text{-}Unif}(\mathbf{x}^{NI}_{E\text{-}Unif})$. We will also compute the loss functions of optimal solutions from differing optimizations. For instance, $L_{E\text{-}Len}(\mathbf{x}^{NI}_{E\text{-}Unif})$, and in that case, we do not use the $C^*_\bullet$ notation.

[17]We discuss the coefficients of the Erdős-Rényi complexes of **Section 5.3** in **Section 6.7**.

[18]Recall that, in the current discussion, ℓ_0 optimality refers to the *restricted* integer problem where coefficients are constrained to lie in {-1, 0, 1}. The unrestricted problem (about which we have nothing to say) may have quite different properties.

Figure 7 shows an example of various optimal cycle representatives obtained from Programs Edge^{NI}_{Unif}, Edge^{NI}_{Len}, Tri^{NI}_{Unif}, and Tri^{NI}_{Area} on an example point cloud drawn from the normal distribution in $\mathbb{R}^2$. In this example, solutions obtained from **Algorithm 1** and the program in **Eq. 8** are the same. Each subfigure is labeled by program in the upper left corner. The values of different loss functions evaluated on each optimal representative appear in the upper right corner. We do not compute $L_{T\text{-}Unif}$ or $L_{T\text{-}Area}$ of the optimal edge-loss minimal cycle representatives, as no bounding 2-chain for this 1-cycle is specified in the optimization.[19] We observe that various notions of optimality lead to differing cycle representatives, yet each solution to an optimization problem minimizes the loss function it is intended to optimize. This will not always be the case, as we will see momentarily.

Figure 8 reports ratios on the losses $L_{E\text{-}Unif}$, $L_{E\text{-}Len}$, and $L_{Sur\text{-}Area}$[20] for the eight PrsHCB optimization problems detailed in **Section 4** as well as the four edge-loss FCB problems from the program in **Eq. 8**, evaluated on the data from **Sections 5.1** and **5.2**. These ratios suggest that the uniform-weighted and length-weighted edge-loss cycles do minimize what they set out to minimize, namely, the number of edges and the total length, respectively. We also observe that intuitively the less-constrained solutions to the FCB program in **Eq. 8** can have a lower cost than the more-constrained solutions to the PrsHCB program in **Eq. 14**.

We also see that the edge-loss-minimal cycles have similar loss in terms of length and number of edges ($L_{E\text{-}Len}$ and $L_{E\text{-}Unif}$) whereas the triangle-loss-minimal cycles can have larger losses ($L_{E\text{-}Len}(\mathbf{x}_{T\text{-}Unif})$ and $L_{E\text{-}Unif}(\mathbf{x}_{T\text{-}Unif})$). We find that 63.28% of the $L_{E\text{-}Unif}$ minimal cycle representatives are also $L_{E\text{-}Len}$ minimal while 99.66% of the $L_{E\text{-}Len}$ minimal cycle representatives are also $L_{E\text{-}Unif}$ minimal across all cycle representatives from all data sets for PrsHCB cycles. Similarly, we find that 61.31% of the $L_{E\text{-}Unif}$ minimal cycle representatives are also $L_{E\text{-}Len}$ minimal while 99.32% of the $L_{E\text{-}Len}$ minimal cycle representatives are also $L_{E\text{-}Unif}$ minimal across all cycle representatives from all data sets for FCB cycles. This suggests that modelers can often use the length-weighted minimal cycle to substitute the uniform-weighted minimal cycle. However, the triangle-loss cycles can potentially provide very different results.

Counterintuitively, the $L_{T\text{-}Area}$ optimal cycle representative might not be the representative that encloses the smallest surveyor's area. As shown in **Figure 8**, we observe that 15.55% of $\mathbf{x}^{NI}_{E\text{-}Unif}$, 13.14% of $\mathbf{x}^{I}_{E\text{-}Unif}$, 23.59% of $\mathbf{x}^{NI}_{E\text{-}Len}$, and 23.59% of $\mathbf{x}^{I}_{E\text{-}Len}$ for the cycles from PrsHCB using the program in **Eq. 14** have an area smaller than that of the triangle-loss area-weighted optimal cycle $\mathbf{x}^{NI}_{T\text{-}Area}$. Similarly, 15.55% of $\mathbf{x}^{NI}_{E\text{-}Unif}$, 12.87% of $\mathbf{x}^{I}_{E\text{-}Unif}$, 24.53% of $\mathbf{x}^{NI}_{E\text{-}Len}$, and 24.53% of $\mathbf{x}^{I}_{E\text{-}Len}$ for the cycles from FCB using the program in **Eq. 8** have an area smaller than that of the triangle-loss area-weighted optimal cycle $\mathbf{x}^{NI}_{T\text{-}Area}$. Lastly, 3.22% of $\mathbf{x}^{I}_{T\text{-}Unif}$, 2.81% of $\mathbf{x}^{NI}_{T\text{-}Unif}$, and 2.95% of $\mathbf{x}^{I}_{T\text{-}Area}$ for the cycles found using the program in **Eq. 15** have an area smaller than that of the triangle-loss area-weighted optimal cycle $\mathbf{x}^{NI}_{T\text{-}Area}$.

In **Figure 9**, we provide an example illustrating why the triangle-loss area-weighted optimal cycle, solving Programs Tri^{NI}_{Area}, or Tri^{I}_{Area}, might not be the cycle that encloses the smallest surveyor's area. Another reason why the area-weighted triangle-loss cycles could have a larger enclosed area is that in the optimization problems, the loss function is the sum of the triangles the cycle bounds, not the real enclosed area. Therefore, the area-weighted triangle-loss cycle will have the optimal area-weighted optimal cost, but not necessarily the smallest enclosed area.

[19]We formulated an Obayashi-style linear program similar to Program in **Eq. 15** to compute the volume of edge-loss optimal cycles but in many cases it had no feasible solution.

[20]Recall, we only compute L_{Area} on the 2-dimensional distribution data.

6.5 Comparative Performance and Precision of LP Solvers

Our experiments demonstrate that the choice of linear solver may impact speed, frequency of obtaining integer solutions, and frequency of obtaining ℓ_0 optimal solutions. While these particular results are subject to change due to regular updates to each platform, they illustrate the degree to which these factors can vary.

As discussed in **Section 6.1**, the GLPK solver performs much slower than the Gurobi solver in an initial set of experiments. The GLPK solver also finds non-integral solutions when solving a linear programming problem in Programs Edge^{NI}_{Unif}, and Edge^{NI}_{Len} more often than the Gurobi solver. On the same set of experiments as in **Section 6.1**, when finding the FCB using the program in **Eq. 8**, 9.74% of the edge-loss length-weighted minimal cycle representatives have non-integral entries, and 8.32% of the edge-loss uniform-weighted minimal cycle representatives have non-integral entries when using the GLPK solver, whereas when using the Gurobi solver, 0.12% of the length-weighted minimal cycle representatives have non-integral entries, and 0.04% of the uniform-weighted minimal cycle representatives have non-integral entries. For the length-weighted minimal cycle representatives, the non-integral solutions differ from an ℓ_0 optimal solution by a margin of machine error with both solvers. However, for the uniform-weighted minimal cycle representatives, the GLPK solver has 1.83% of its non-integral solutions differing from an ℓ_0 optimal solution by a margin not of machine epsilon, and the Gurobi solver has 0.02% of its non-integral solutions differing from an ℓ_0 optimal solution by a margin greater than machine epsilon. For the GLPK solver, when solving Program Edge^{NI}_{Unif}, instead of finding an integral solution, it occasionally finds a solution with fractional entries that sum to 1. For example, instead of assigning an edge a coefficient of 1, it sometimes assigns two edges each with a coefficient of 0.5. In that way, the solution is still ℓ_1 optimal, but no longer ℓ_0 optimal. Thus, the choice of linear solver may affect the optimization results.

6.6 Statistical Properties of Optimal Cycle Representatives With Regard to Various Other Quantities of Interest

6.6.1 Support of a Representative Forming a Single Loop in the Underlying Graph

The support of the original cycle, $\text{supp}(\mathbf{x}^{Orig}) \subseteq \mathbf{S}_1(K)$, need not be a cycle in the graph-theoretic sense. Concretely, this means that the nullity, p, of column submatrix $\partial_1[:, \mathbf{x}^{Orig}]$ may be strictly greater than 1. We refer to p informally as the "number of loops" in $\mathbf{x}^{Orig}$.

We are interested in exploring how often the support of an original cycle representative forms a single loop in

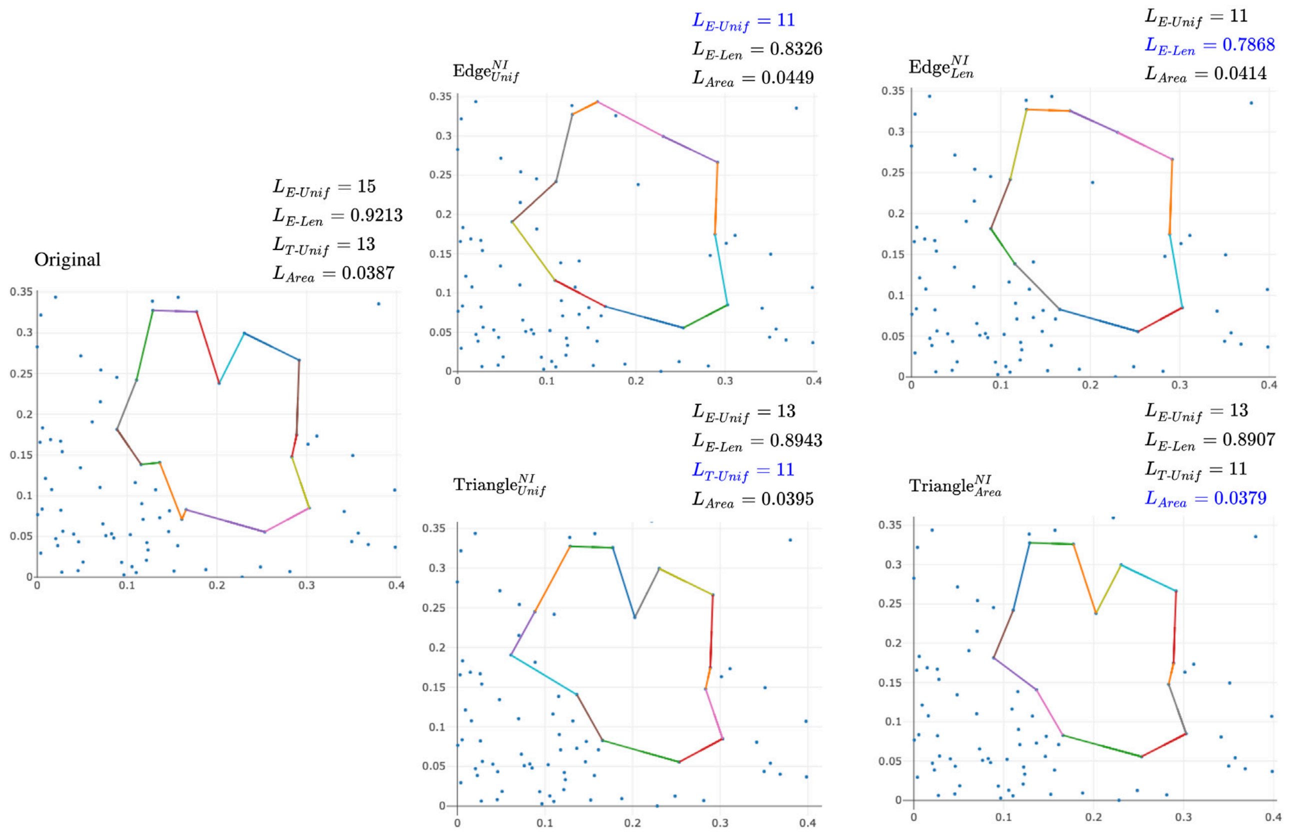

FIGURE 7 | Examples of different optimal cycles and cost against different loss functions using a point cloud of 100 points with ambient dimension two randomly drawn from a normal distribution. The upper left corner of each subfigure labels the optimization algorithm used to optimize the original cycle representative. The upper right corner of each subfigure records the different measures of the size of the optimal representative. Blue text represents the measure an algorithm sets out to optimize.

the underlying graph. We analyze each of the 360 synthetic data sets of various dimensions and distributions discussed in **Section 5.2** as well as the 100 Erdős-Rényi random complexes discussed in **Section 5.3** and display the results in **Figure 10**. We find that the majority of the original cycle representatives have one loop. After optimizing these cycle representatives with the edge-loss methods, we verify that all FCB and PrsHCB optimal cycles only have one loop, whereas 0.13% of the triangle-loss cycles have two loops. However, we observe that 91.93% of the optimal cycle representatives for Erdős-Rényi complexes have 1 loop, 5.81% have two loops, and 2.14% have more than two loops, with 15 as the maximum number of loops.

As shown in **Figure 11** the reduction in size of the original cycle, formalized as $\frac{C^*_\bullet}{L_\bullet(\mathbf{x}^{Orig})}$, correlates closely with the reduction in the number of loops by the optimization.

6.6.2 Overall Effectiveness of Optimization ($L_\bullet(\mathbf{x}^*_\bullet)$ vs. $L_\bullet(\mathbf{x}^{Orig})$)

We compare the optimal representatives against the original cycle representatives[21] with respect to edge-loss functions $L_{E\text{-}Unif}$ and $L_{E\text{-}Len}$. As shown in **Figure 12**, we find that the optimizations are in general effective in reducing the size of the cycle representative, especially for representatives with larger size. On each of the subfigures, the horizontal axis is the size of the original cycle representative and the vertical axis is the ratio between the loss of each optimal representative and the loss of the original representative:

$$\frac{C^*_\bullet}{L_\bullet(\mathbf{x}^{Orig})}.$$

The average ratio $\frac{C^{NI}_{E-Unif}}{L_{E-Unif}(\mathbf{x}^{Orig})}$ is 83.17%, aggregated over cycle representatives obtained from data described in **Section 5.1** and 90.35% aggregated over cycle representatives obtained from data described in **Section 5.2** for cycles obtained from the program in **Eq. 14**. The average ratio $\frac{C^{NI}_{E-Len}}{L_{E-Len}(\mathbf{x}^{Orig})}$ is 87.02% over cycle representatives obtained from data described in **Section 5.1** and 90.41% over cycle representatives obtained from data describedra **Section 5.2** for cycles obtained from the program in **Eq. 14**. The average ratio $\frac{C^{NI}_{T-Unif}}{L_{T-Unif}(\mathbf{x}^{Orig})}$ is 88.34% over cycle representatives obtained from data described in **Section 5.1** and 95.54% over cycle representatives obtained from data described in **Section 5.2** for cycles obtained from the program in **Eq. 15**.

[21]The remainder of this subsection excludes the Erdős-Rényi cycles.

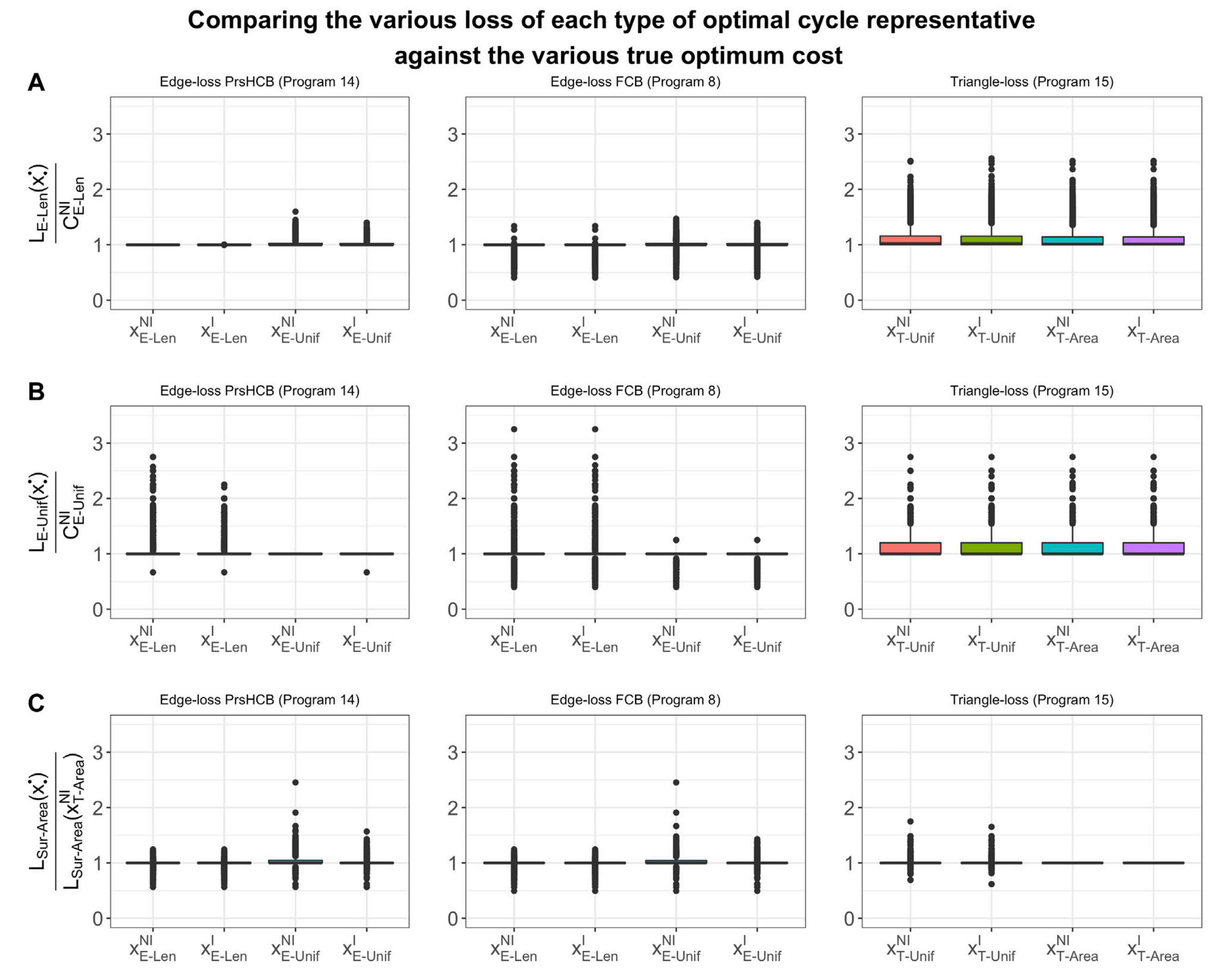

FIGURE 8 | Box plots of the ratios between **(A)** $L_{E\text{-}Len}(\mathbf{x}^*_\bullet)$ and $C^{NI}_{E\text{-}Len}$ **(B)** $L_{E\text{-}Unif}(\mathbf{x}^*_\bullet)$ and $C^{NI}_{E\text{-}Unif}$, and **(C)** $L_{Sur\text{-}Area}(\mathbf{x}^*_\bullet)$ and $L_{Sur\text{-}Area}(x^{NI}_{T\text{-}Area})$. Within each row, the denominator is fixed across all three columns, and corresponds to the PrsHCB cycles which are solutions to Programs Edge^{NI}_{Unif} row **(A)**, Edge^{NI}_{Len} row **(B)**, or Tri^{NI}_{Area} row **(C)**. The horizontal axis of each subplot is the type of optimal representative. The cost of the optimal solutions to the programs in **Eqs 8**, **14**, and Program Tri^{NI}_{Area} was equal regardless of the presence of an integer constraint in nearly all examples (as discussed in **Section 6.3**), resulting in two columns in each row having ratio 1. The data used in **(A)** and **(B)** aggregate over all cycle representatives from data described in **Sections 5.1** and **5.2**. The data used in **(C)** aggregate the 746 cycle representatives from 40 point clouds with ambient dimension of two from data described in **Section 5.2**. We observe that some edge-loss and uniform-weighted-triangle-loss optimal cycles have a surveyor's area strictly smaller than the denominator in row **(C)**; refer to **Figure 8** and **Section 6.4** to see why this may happen. It is possible for $L_{E\text{-}Unif}(\mathbf{x}^*_\bullet)$ to be strictly smaller than $C^{NI}_{E\text{-}Unif}$ because the cycle $\mathbf{x}^{NI}_{E\text{-}Unif}$ is calculated to be optimal relative to ℓ_1 loss, not $L_{E\text{-}Unif}$, which is a measure of ℓ_0 loss. We observe this behavior in the first plot on the second row, discussed in detail in **Section 6.3**.

6.6.3 Comparing Solutions to Integral Programs and Non-Integral Programs ($\mathbf{x}^{NI}_\bullet$ vs. $\mathbf{x}^{I}_\bullet$)

Among all cycle representatives found by solving the program in **Eq. 14**, 66.38% of them have $\mathbf{x}^{NI}_{E\text{-}Unif} = \mathbf{x}^{I}_{E\text{-}Unif}$, and 92.52% of them have $\mathbf{x}^{NI}_{E\text{-}Len} = \mathbf{x}^{I}_{E\text{-}Len}$. We find $\mathbf{x}^{NI}_{T\text{-}Unif} = \mathbf{x}^{I}_{T\text{-}Unif}$ for 80.44% of the cycle representatives and $\mathbf{x}^{NI}_{T\text{-}Area} = \mathbf{x}^{I}_{T\text{-}Area}$ for 100% of the cycle representatives when using the triangle-loss program in **Eq. 15**. Thus, the presence or absence of integer constraints rarely impacts the result of an area- or length-weighted program, but often impacts solutions of a uniform-weighted program. We saw in **Section 6.3** that essentially all solutions had coefficients in $\{-1, 0, 1\}$ regardless of integer or non-integer constraints. As such, we conjecture that the higher rate of different solutions in the uniform-weighted problems could result from a larger number of distinct optimal solutions and be a feature of particular choice of solution selected by the linear solvers, rather than the non-existence of a particular integer solution.

6.6.4 Cycle Representative Size for Different Distributions and Dimensions

Figure 13 provides a summary of the size and number of cycle representatives found for each distribution data set described in

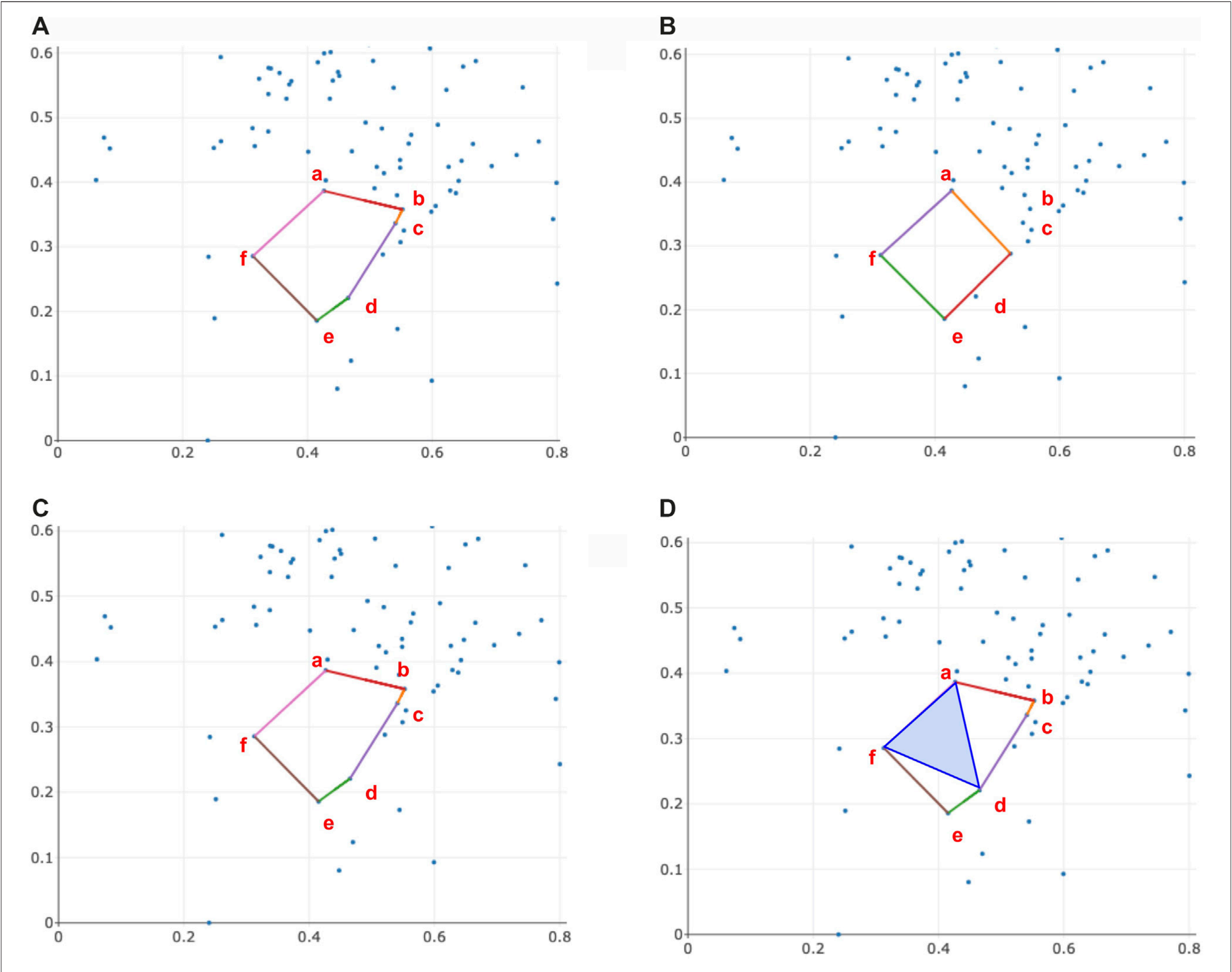

FIGURE 9 | An example illustrating when the area enclosed by the triangle-loss area-weighted optimal cycle, solution to Program Tri^{NI}_{Area}, can be larger than the area enclosed by the edge-loss length-weighted minimal cycle, solution to Program Edge^{NI}_{Len} **(A)** is the original cycle of a representative point cloud in $\mathbb{R}^2$ drawn from the normal distribution **(B)** is the length-weighted edge-loss optimal cycle **(C)** is the area-weighted triangle-loss optimal cycle, in this example, it is the same cycle as the original cycle **(D)** is the area-weighted minimal cycle where the blue shaded area marks the triangle born at the death time of the cycle. Constraint **Eq. 9** specifies that the area-weighted optimal cycle must contain the 2-simplex born at the death time of the cycle. Therefore, this cycle must contain (a, d, f) because it was born at the death time. The length-weighted minimal cycle does not have this constraint, and as such, can result in a smaller area.

Section 5.2. We observe that there tend to be more and larger (with respect to ℓ_0 norm) representatives in higher dimensions.

6.6.5 Duplicate Intervals in the Barcode

Of all data sets analyzed, only **Klein** and **C.elegans** have barcodes in which two or more intervals had equal birth and death times (that is, bars with multiplicity ≥ 2). Among the 107 total intervals of the **C.elegans** data set, there are 75 unique intervals, 10 intervals with multiplicity two, and one interval each with multiplicity three, four, and five. The duplicate bars in the **C.elegans** data set are noteworthy for having produced the sole example of an optimized cycle representative $\mathbf{x}^{NI}_{E\text{-}Unif}$ with coefficients outside {-1, 0, 1} (in particular, it had coefficients in {-0.5, 0.5}).

Among the 257 total intervals of the **Klein** data set, there are 179 unique intervals, 1 interval that is repeated twice, and two intervals that are repeated 38 times. For the **Klein** data set, if we replace the distance matrix provided by Otter et al. (2017) with the Euclidean distance matrix calculated using Julia (the maximum difference between the two matrices is on the scale of 10^{-5}), we obtain only one interval that is repeated twice. This indicates that duplicate intervals are rare in practice, at least in dimension 1.

6.6.6 Edge-Loss Cycle Representatives FCB vs. PrsHCB

We find that for 84.52% of Edge^{NI}_{Unif}, 90.84% of Edge^{I}_{Unif}, 93.49% of Edge^{NI}_{Len}, and 93.49% of Edge^{I}_{Len}, the FCB edge-loss cycle representatives found by the program in **Eq. 8** and the PrsHCB edge-loss cycles from the program in **Eq. 14** are the same, i.e. the ℓ_1 norm of their difference is 0. As mentioned in **Remark 3.1**, the FCB cycles may not have the same death time as $\mathbf{x}^{Orig}$. For the real-world data sets, 6.72% of the (Edge^{NI}_{Len}) and (Edge^{I}_{Len}), 7.65% of the (Edge^{NI}_{Unif}) and

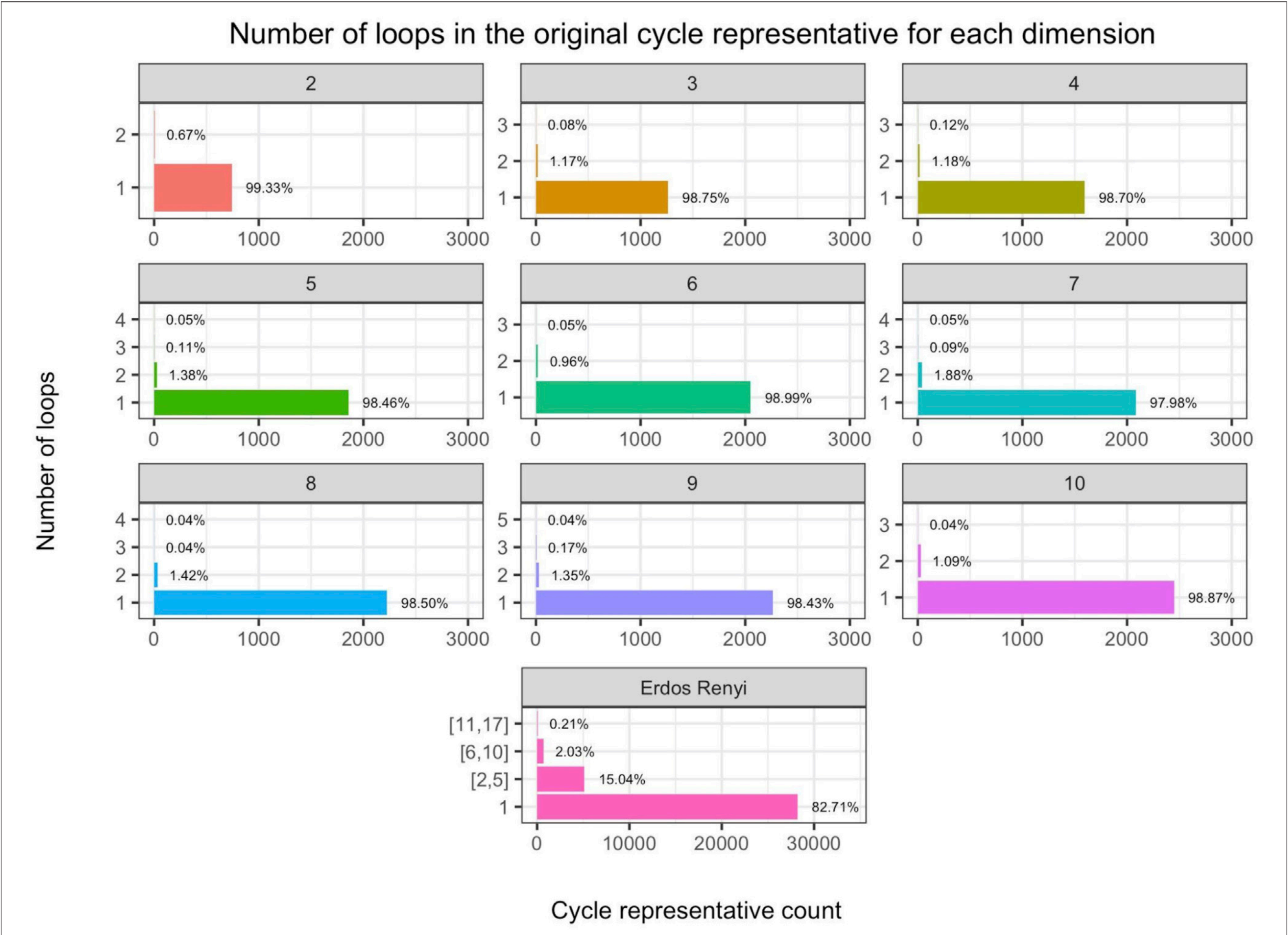

FIGURE 10 | (Rows 1–3) Number of loops in the original cycle representative aggregated by dimension (labeled by subfigure title) in the 360 randomly generated distribution data sets discussed in **Section 5.2** and (Row 4) same for the Erdős-Rényi random complexes discussed in **Section 5.3**, where we bin cycle representatives that have two to five loops, 6–10, loops, or more than 10 loops. The horizontal axis is the number of cycle representatives and the vertical axis is the number of loops in the original representative. We observe that for the distribution data, an original cycle representative can have up to 5 loops in higher dimensions, and in general, it is uncommon to find an original representative with multiple loops. However, we observe that 17.47% of the cycle representatives for Erdős-Rényi complexes have more than 1 loop, with a maximum number of 17 loops in a cycle representative.

4.48% of (Edge^{I}_{Unif}) have lifetimes different than $\mathbf{x}^{Orig}$. For the randomly generated distribution data sets, 7.11% of the (Edge^{NI}_{Len}) and (Edge^{NI}_{Len}), 8.06% of the (Edge^{NI}_{Unif}) and 4.25% of (Edge^{I}_{Len}) have lifetimes different than $\mathbf{x}^{Orig}$. All cycle representatives with lifetimes different than $\mathbf{x}^{Orig}$ have death time beyond that of $\mathbf{x}^{Orig}$.

6.7 Optimal Cycle Representatives for Erdős-Rényi Random Clique Complexes

We observe qualitatively different behavior in cycle representatives from the Erdős-Rényi random clique complexes. Among the 34, 214 cycle representatives from the 100 dissimilarity matrices found by solving the programs in **Eqs 14**, **15**, we find that 91.04% of the original cycle representatives have entries in {-1, 0, 1} and 99.75% of the original cycle representatives have integral entries. We have 3.89% of the length-weighted edge-loss representatives, 4.49% of the uniform-weighted edge-loss representatives, and 3.52% of the uniform-weighted triangle-loss representatives with entries not in {-1, 0, 1}. We find 2.66% of the length-weighted edge-loss representatives, 3.57% of the uniform-weighted edge-loss representatives, and 1.58% of the uniform-weighted triangle-loss representatives with non-integral entries when not requiring integral solutions.

We find $\frac{L_{E-Unif}(\mathbf{x}^{NI}_{E-Unif})}{L_{E-Unif}(\mathbf{x}^{I}_{E-Unif})} > 1$ for 1.07% of the cycle representatives and $\frac{L_{E-Len}(\mathbf{x}^{NI}_{E-Len})}{L_{E-Len}(\mathbf{x}^{I}_{E-Len})} > 1$ for 1.09% of the representatives. All such representatives have entries outside of {-1, 0, 1} and involve some fractional entries. An average of 96.75% of the nonzero entries in the reduced boundary matrices are in {-1, 1}, 2.15% in {-2, 2}, and 0.27% with an absolute value greater than or equal to 3.

Because of the non-integrality of some original cycle representatives found by the persistence algorithm, we fail to find an integral solution for 0.27% of the edge-loss representatives and 0.11% of the triangle-loss representatives.

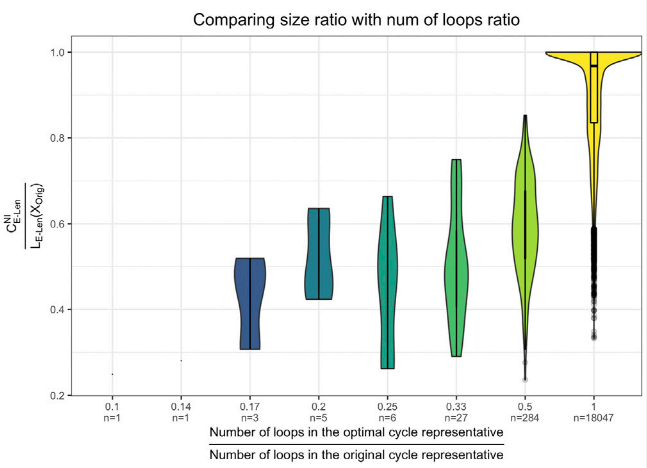

FIGURE 11 | Violin plot of the effectiveness of the optimization as a function of the number of loops in the original cycle representative. Results are aggregated over the data sets from **Section 5.1** and **Section 5.2**. The *x*-axis shows the size reduction in terms of number of loops, and the *y*-axis shows the size reduction in terms of the length of the cycle. We see that in general, the reduction in size of the original cycle mostly comes from the reduction in the number of loops by the optimization.

A partial explanation for this behavior can be found in the work of (Costa et al., 2015). Here, the authors show that a connected two-dimensional simplicial complex for which every subcomplex has fewer than three times as many edges as vertices must have the homotopy type of a wedge of circles, 2-spheres, and real projective planes. With high probability, certain ranges of thresholds for the i.i.d. dissimilarity matrices on which the Erdős-Rényi random complexes are built produces random complexes with approximately such density patterns at each vertex. Thus, some of the persistent cycles are highly likely to correspond to projective planes. Because of their non-orientability, the corresponding minimal generators could be expected to have entries outside of the range {-1, 0, 1}.

7 CONCLUSION

In this work, we provide a theoretical, computational, and empirical user's guide to optimizing cycle representatives against four criteria of optimality: total length, number of edges, internal volume, and area-weighted internal volume. Utilizing this framework, we undertook a study on statistical properties of minimal cycle representatives for $\mathbf{H}_1$ homology found via linear programming. In doing so, we made the following four main contributions.

1. We developed a publicly available code library (Li and Thompson, 2021) to compute persistent homology with rational coefficients, building on the software package Eirene (Henselman and Ghrist, 2016) and implemented and extended algorithms from (Escolar and Hiraoka, 2016; Obayashi, 2018) for computing minimal cycle representatives. The library employs standard linear solvers (GLPK and Gurobi) and implements various acceleration techniques described in **Section 4.4** to make the computations more efficient.
2. We formulated specific recommendations concerning procedural factors that lie beyond the scope of the optimization problems per se (for example, the process used to generate inputs to a solver) but which bear directly on the overall cost of computation, and of which practitioners should be aware.
3. We used this library to compute optimal cycle representatives for a variety of real-world data sets and randomly generated point clouds. Somewhat surprisingly, these experiments demonstrate that computationally advantageous properties are typical for persistent cycle representatives in data. Indeed, we find that we are able to compute uniform/length-weighted optimal cycles for all data sets we considered, and that we are able to compute triangle-loss optimal cycles for all but six cycle representatives, which fail due to the large number of triangles (more than 20 million) used in the optimization problem. Computation time information is summarized in **Table 1** and **Table 2**.
   Consequently, heuristic techniques may provide efficient means to extract solutions to cycle representative optimization problems across a broad range of contexts. For example, we find that edge-loss optimal cycles are faster to compute than triangle-loss optimal cycles for cycle representatives with a longer persistence interval, whereas for cycles with shorter persistence intervals, the triangle-loss cycle can be less computationally expensive to compute.
4. We provided statistics on various minimal cycle representatives found in these data, such as their effectiveness in reducing the size of the original cycle representative found by the persistence algorithm and their effectiveness evaluated against different loss functions. In doing so, we identified consistent trends across samples that address the questions raised in **Section 1**.
   a. Optimal cycle representatives are often significant improvements in terms of a given loss function over the initial cycle representatives provided by persistent homology computations (typically, by a factor of 0.3–1.0). Interestingly, we find that area-weighted triangle-loss optimal cycle representatives can enclose a greater area than length- or uniform-weighted optimal cycle representatives.
   b. We find that length-weighted edge-loss optimal cycles are also optimal with respect to a uniform-weighted edge-loss function upwards of 99% of the time in the data we studied. This suggests that one can often find a solution that is both length-weighted minimal and uniform-weighted minimal by solving only the length-weighted minimal optimization problem. However, the triangle-loss optimal cycles can have a relatively higher length-weighted edge-loss or uniform-weighted edge-loss than the length/uniform-weighted minimal cycles. Thus, computing triangle-loss optimal cycles might provide distinct information and insights.
   c. Strikingly, all but one ℓ_1 optimal representatives were also ℓ_0 optimal (that is, ℓ_0 optimal among cycles taking $\{0, 1, \text{-}1\}$ coefficients; ℓ_0 optimality among cycles taking $\mathbb{Z}$ coefficients was not tested) in the real-world and synthetic point cloud data. Thus, it appears that solutions to the NP-hard problem of finding ℓ_0 optimal cycle representatives can often be solved using linear programming in real data. In the Erdős-Rényi random complexes, qualitatively different behavior was found; this may relate to the fact that spaces in this random

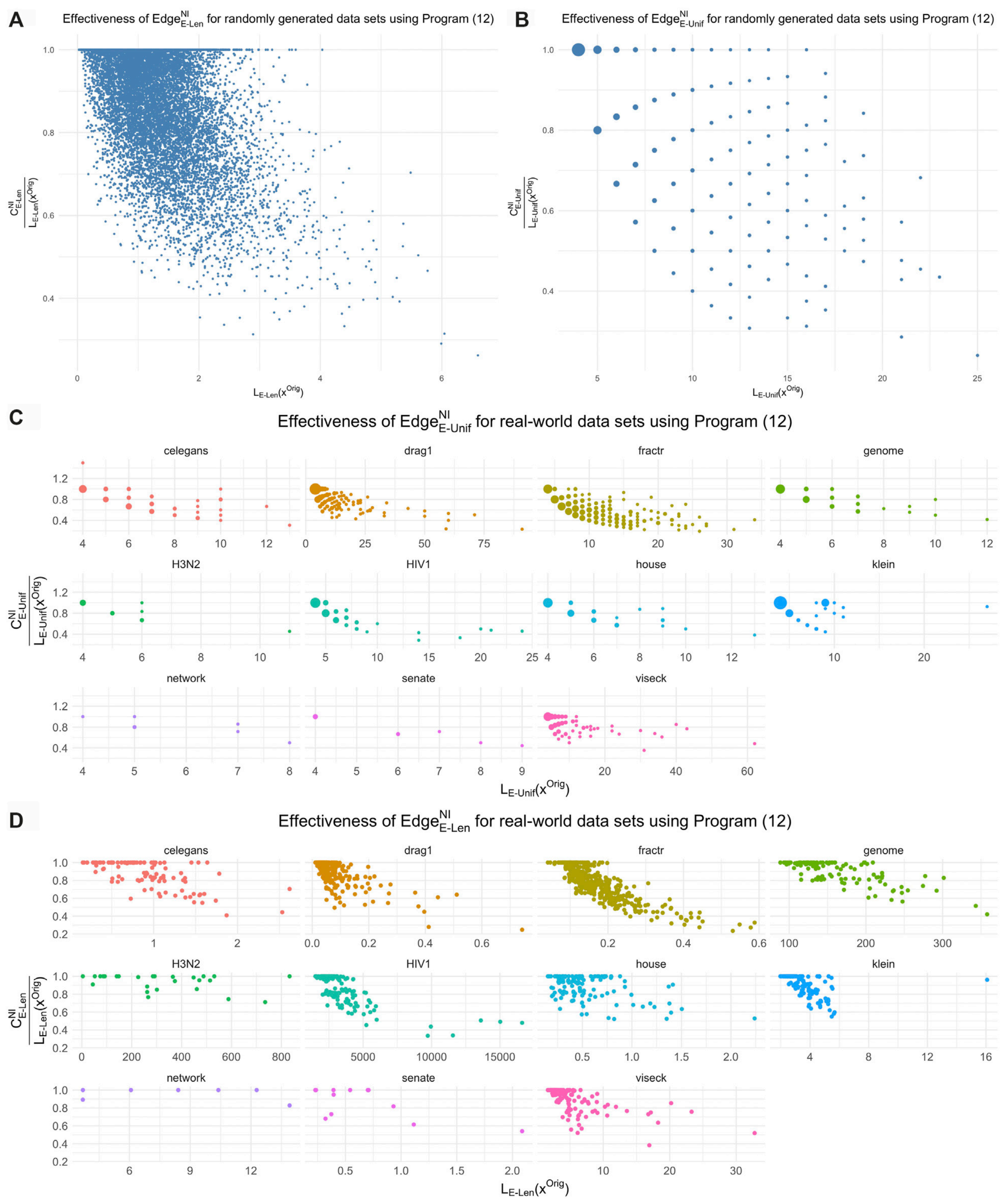

FIGURE 12 | The effectiveness of length-weighted and uniform-weighted optimization for the data sets in **Sections 5.1** and **5.2** in reducing the size of the original cycle representative found by the persistence algorithm. In each subfigure, the horizontal axis is the size of the original representative and the vertical axis is the ratio between the size of the optimal representative and the size of the original representative. The uniform-weighted graphs appear more sparse because reductions in the cost $L_{T\text{-}Unif}$ ($\mathbf{x}^{Orig}$) can only come in multiples of the reciprocal of the original length. The node size in the uniform-weighted graphs corresponds to the number of overlapping points.

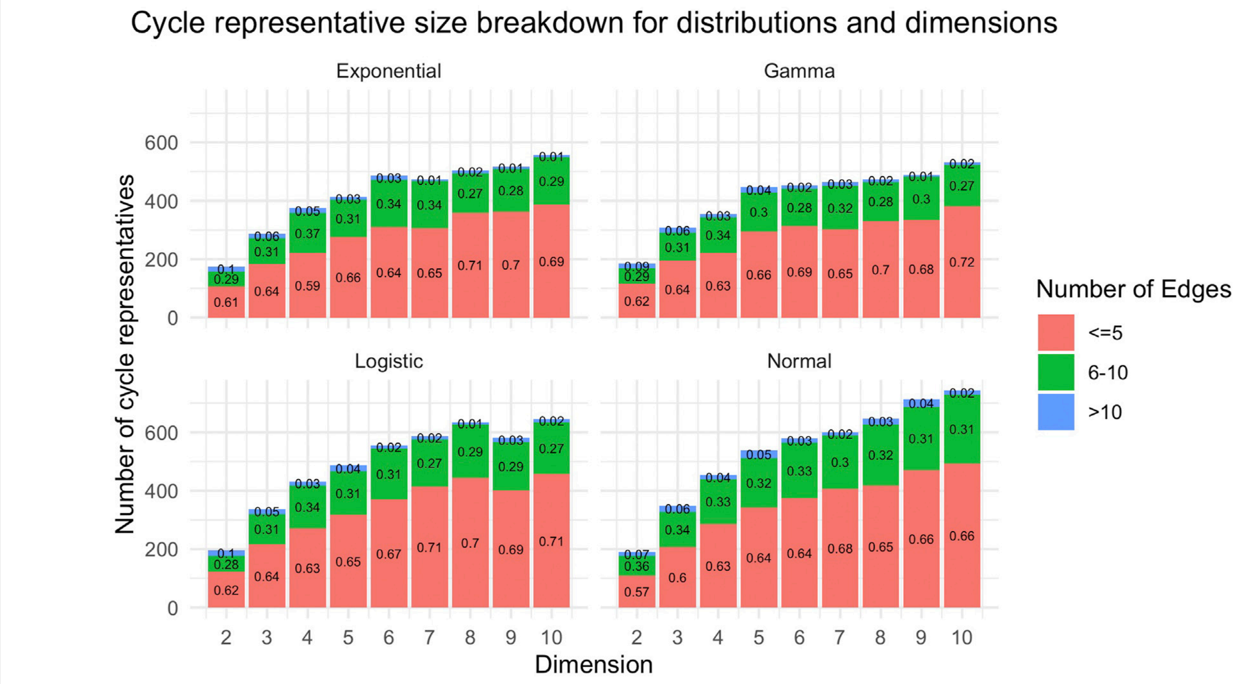

FIGURE 13 | The number of original cycle representatives and the number of edges within each original representative for data described in **Section 5.2**. These plots aggregate all cycle representatives for each dimension of a particular distribution. The horizontal axis for each subplot is the dimension of the data set, and the vertical axis is the number of cycle representatives found in each dimension. In general, we see there are more cycle representatives in higher dimensional data sets. Each bar is partitioned by the number of edges of the representative. We observe that as dimension increases, there tend to be more cycle representatives with more edges.

family contain non-orientable subcomplexes with high probability.

Several questions lie beyond the scope of this text and merit future investigation. For example, while the methods discussed in **Section 4** apply equally to homology in any dimension, we have focused our empirical investigation exclusively in dimension one; it would be useful and interesting to compare these results with homology in higher dimensions. It would likewise be interesting to compare with different weighting strategies on simplices, and loss functions other than ℓ_0 and ℓ_1, e.g. ℓ_2. Future work may also consider whether the modified approach to the edge-loss minimization program in **Eq. 14** could be incorporated into persistence solvers themselves, as pioneered in (Escolar and Hiraoka, 2016). Unlike the programs formulated in this earlier work, the program in **Eq. 14** requires information about the death times of cycles in addition to their births; typically this information is not available until after the persistence computation has already finished, so new innovations would probably be needed to make progress in this direction.

DATA AVAILABILITY STATEMENT

The datasets presented in this study can be found in online repositories. The names of the repository/repositories and accession number(s) can be found below: https://github.com/TDAMinimalGeneratorResearch/minimal-generator.

AUTHOR CONTRIBUTIONS

GH-P wrote the Eirene code. LL and CT wrote the rest of the code and performed all experiments. LL created all figures and tables. GH-P and LZ designed, directed, and supervised the project. GH-P developed the theory in the **Supplementary Material**. All authors contributed to the analysis of the results and to the writing of the manuscript.

FUNDING

This material is based upon work supported by the National Science Foundation under grants no. DMS-1854683, DMS-1854703, and DMS-1854748. GH-P is a member of the Centre for Topological Data Analysis, supported by the EPSRC grant New Approaches to Data Science: Application Driven Topological Data Analysis EP/R018472/1.

ACKNOWLEDGMENTS

The authors are grateful to David Turner for helpful advice on selection of linear solvers. They also thank and acknowledge the initial work done by Robert Angarone and Sophia Wiedemann on this study.

SUPPLEMENTARY MATERIAL

The Supplementary Material for this article can be found online at: https://www.frontiersin.org/articles/10.3389/frai.2021.681117/full#supplementary-material

REFERENCES

Bendich, P., Marron, J. S., Miller, E., Pieloch, A., and Skwerer, S. (2016). Persistent Homology Analysis of Brain Artery Trees. *Ann. Appl. Stat.* 10, 198–218. doi:10.1214/15-AOAS886

Bertsimas, D., and Tsitsiklis, J. (1997). *Introduction to Linear Optimization*. (Belmont, MA: Athena Scientific).

Bhaskar, D., Manhart, A., Milzman, J., Nardini, J. T., Storey, K. M., Topaz, C. M., et al. (2019). Analyzing Collective Motion with Machine Learning and Topology. *Chaos: Interdiscip. J. Nonlinear Sci.* 29, 123125. doi:10.1063/1.5125493

Bhattacharya, S., Ghrist, R., and Kumar, V. (2015). Persistent Homology for Path Planning in Uncertain Environments. *IEEE Trans. Robotics* 31, 578–590. doi:10.1109/TRO.2015.2412051

Borradaile, G., Maxwell, W., and Nayyeri, A. (2020). "Minimum Bounded Chains and Minimum Homologous Chains in Embedded Simplicial Complexes," in *36th International Symposium on Computational Geometry (SoCG 2020). Leibniz Int. Proc. Informatics* 21, 21:1–21:15.

Boyd, S., and Vandenberghe, L. (2004). *Convex Optimization*. Belmont, MA: Cambridge University Press.

Braden, B. (1986). The Surveyor's Area Formula. *Coll. Math. J.* 17, 326–337. doi:10.1080/07468342.1986.11972974

Brüel-Gabrielsson, R., Ganapathi-Subramanian, V., Skraba, P., and Guibas, L. J. (2020). Topology-aware Surface Reconstruction for point Clouds. *Comput. Graphics Forum* 39, 197–207. doi:10.1111/cgf.14079

Carlsson, G. (2009). Topology and Data. *Bull. Am. Math. Soc.* 46, 255–308.

Chan, J. M., Carlsson, G., and Rabadan, R. (2013). Topology of Viral Evolution. *Proc. Natl. Acad. Sci.* 110, 18566–18571. doi:10.1073/pnas.1313480110

Chen, C., and Freedman, D. (2010a). Measuring and Computing Natural Generators for Homology Groups. *Comput. Geometry* 43, 169–181. doi:10.1016/j.comgeo.2009.06.004

Chen, C., and Freedman, D. (2010b). "Hardness Results for Homology Localization," in *SODA '10: Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms*. USA: Society for Industrial and Applied Mathematics).

Chen, C., and Freedman, D. (2008). "Quantifying Homology Classes. Albers S," in *25th International Symposium On Theoretical Aspects Of Computer Science*. (Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum Fuer Informatik). Editor P. Weil, 1, 169–180. doi:10.4230/LIPIcs.STACS.2008.1343

Cohen-Steiner, D., Edelsbrunner, H., and Morozov, D. (2006). "Vines and Vineyards by Updating Persistence in Linear Time," in Proceedings of the Twenty-Second Annual Symposium on Computational Geometry, 119–126.

Costa, A., Farber, M., and Horak, D. (2015). Fundamental Groups of Clique Complexes of Random Graphs. *Trans. Lond. Math. Soc.* 2, 1–32. doi:10.1112/tlms/tlv001

Davis, T. A., and Hu, Y. (2011). The university of florida Sparse Matrix Collection. *ACM Trans. Math. Softw.* 38, 1–25. doi:10.1145/2049662.2049663

Dey, T. K., Sun, J., and Wang, Y. (2010). "Approximating Loops in a Shortest Homology Basis from point Data," in *SoCG '10: Proceedings of the Twenty-Sixth Annual Symposium on Computational Geometry*. (New York, NY, USA: Association for Computing Machinery), 166–175. doi:10.1145/1810959.1810989

Dey, T. K., Hirani, A. N., and Krishnamoorthy, B. (2011). Optimal Homologous Cycles, Total Unimodularity, and Linear Programming. *SIAM J. Comput.* 40, 1026–1044. doi:10.1137/100800245

Dey, T. K., Li, T., and Wang, Y. (2018). "Efficient Algorithms for Computing a Minimal Homology Basis," in *Latin American Symposium on Theoretical Informatics*. Springer, 376–398.

Dey, T. K., Hou, T., and Mandal, S. (2019). "Persistent 1-cycles: Definition, Computation, and its Application," in *International Workshop on Computational Topology in Image Context*. Málaga, Spain: Springer, 123–136.

Edelsbrunner, H., and Harer, J. (2008). Persistent Homology–A Survey. *Discrete Comput. Geometry - DCG* 453, 257–282. doi:10.1090/conm/453/08802

Edelsbrunner, H., and Harer, J. (2010). *Computational Topology: An Introduction*. Providence, RI: Applied Mathematics (American Mathematical Society).

Emmett, K., Schweinhart, B., and Rabadan, R. (2015). *Multiscale Topology of Chromatin Folding*. arXiv preprint arXiv:1511.01426.

Erickson, J., and Whittlesey, K. (2009). "Greedy optimal homotopy and homology generators. *SODA* 5, 1038–1046.

Escolar, E. G., and Hiraoka, Y. (2016). "Optimal Cycles for Persistent Homology via Linear Programming," in *Optimization in the Real World*. Tokyo: Springer, 79–96.

Escolar, E., and Hiraoka, Y. (2021). *OptiPersLP - optimal cycles in persistence via linear programming*. Available at: https://bitbucket.org/remere/optiperslp/src/master/.

Ghrist, R. (2008). Barcodes: the Persistent Topology of Data. *Bull. Am. Math. Soc.* 45, 61–75. doi:10.1090/s0273-0979-07-01191-3

Ghrist, R. (2014). *Elementary Applied Topology*. (Seattle: CreateSpace Independent Publishing Platform).

Giusti, C., Ghrist, R., and Bassett, D. S. (2016). Two's Company, Three (Or More) Is a Simplex. *J. Comput. Neurosci.* 41, 1–14.

GNU Project (2012). *GLPK (GNU Linear Programming Kit)*. Available at:.

Gurobi Optimization, L. (2020). *Gurobi Optimizer Reference Manual*.

Hatcher, A., and Press, C. U. of Mathematics CUD (2002). Algebraic Topology. *Algebraic Topology*. Cambridge, UK: Cambridge University Press.

Henselman, G., and Dłotko, P. (2014). "Combinatorial Invariants of Multidimensional Topological Network Data," In IEEE Global Conference on Signal and Information Processing (GlobalSIP). IEEE, 828–832.

Henselman, G., and Ghrist, R. (2016). *Matroid Filtrations and Computational Persistent Homology*. Available at: http://adsabs.harvard.edu/abs/2016arXiv160600199H (Accessed May, 2021).

Henselman-Petrusek, G. (2016). Eirene (Julia Library for Computational Persistent Homology). Available at: https://github.com/Eetion/Eirene.jl (Accessed May, 2021). Commit 4f57d6a0e4c030202a07a60bc1bb1ed1544bf679.

Hiraoka, Y., Nakamura, T., Hirata, A., Escolar, E. G., Matsue, K., and Nishiura, Y. (2016). Hierarchical Structures of Amorphous Solids Characterized by Persistent Homology. *Proc. Natl. Acad. Sci.* 113, 7035–7040. doi:10.1073/pnas.1520877113

Kovacev-Nikolic, V., Bubenik, P., Nikolić, D., and Heo, G. (2016). Using Persistent Homology and Dynamical Distances to Analyze Protein Binding. *Stat. App. Genet. Mol Biol.* 15, 19–38. doi:10.1515/sagmb-2015-0057

Li, L., and Thompson, C. (2021). *Source Code,* Minimal Cycle Representatives In Persistent Homology Using Linear Programming: An Empirical Study With User's Guide. Available at: https://github.com/TDAMinimalGeneratorResearch/minimal-generator (Accessed May, 2021).

Los Alamos National Laboratory (2021). *HIV Database*. Available at: http://www.hiv.lanl.gov/content/index.

McGuirl, M. R., Volkening, A., and Sandstede, B. (2020). Topological Data Analysis of Zebrafish Patterns. *Proc. Natl. Acad. Sci.* 117, 5113–5124. doi:10.1073/pnas.1917763117

Milosavljević, N., Morozov, D., and Skraba, P. (2011). "Zigzag Persistent Homology in Matrix Multiplication Time," in *SoCG '11: Proceedings of the Twenty-Seventh Annual Symposium on Computational Geometry*. New York, NY, USA: Association for Computing Machinery), 216–225. doi:10.1145/1998196.1998229

Newman, M. E. (2006). Finding Community Structure in Networks Using the Eigenvectors of Matrices. *Phys. Rev. E* 74, 036104. doi:10.1103/physreve.74.036104

Obayashi, I., Wada, T., Tunhua, T., Miyanaga, J., and Hiraoka, Y. (2021). *HomCloud: Data Analysis Software Based on Persistent Homology)*. Available at: https://homcloud.dev/.

Obayashi, I. (2018). Volume-optimal Cycle: Tightest Representative Cycle of a Generator in Persistent Homology. *SIAM J. Appl. Algebra Geometry* 2, 508–534.

Otter, N., Porter, M. A., Tillmann, U., Grindrod, P., and Harrington, H. A. (2017a). A Roadmap for the Computation of Persistent Homology. *EPJ Data Sci.* 6, 17. doi:10.1140/epjds/s13688-017-0109-5

Otter, N., Porter, M. A., Tillmann, U., Grindrod, P., and Harrington, H. A. (2017b). A Roadmap for the Computation of Persistent Homology. Available at: https://github.com/n-otter/PH-roadmap

Petri, G., Scolamiero, M., Donato, I., and Vaccarino, F. (2013). Topological Strata of Weighted Complex Networks. *PloS one* 8, e66506. doi:10.1371/journal.pone.0066506

Schweinhart, B. (2015). *Statistical Topology of Embedded Graphs*. Ph.D. thesis. Princeton, NJ: Princeton University.

Singh, G., Mémoli, F., and Carlsson, G. E. (2007). Topological Methods for the Analysis of High Dimensional Data Sets and 3d Object Recognition. *SPBG* 91, 100. doi:10.2312/SPBG/SPBG07/091-100

Sizemore, A. E., Phillips-Cremins, J. E., Ghrist, R., and Bassett, D. S. (2019). The Importance of the Whole: Topological Data Analysis for the Network Neuroscientist. *Netw. Neurosci.* 3, 656–673. doi:10.1162/netn_a_00073

Sporns, O. (2006). Small-world Connectivity, Motif Composition, and Complexity of Fractal Neuronal Connections. *Bio. Syst.* 85, 55–64. doi:10.1016/j.biosystems.2006.02.008

Stanford University Computer Graphics Laboratory (1999). *The Stanford 3D Scanning Repository*. (Accessed May, 2021).

Tahbaz-Salehi, A., and Jadbabaie, A. (20082008). Distributed Coverage Verification in Sensor Networks without Location Information. *IEEE Conf. Decis. Control.*, 4170–4176. doi:10.1109/CDC.2008.4738751

Tahbaz-Salehi, A., and Jadbabaie, A. (2010). Distributed Coverage Verification in Sensor Networks without Location Information. *IEEE Trans. Automatic Control.* 55, 1837–1849. doi:10.1109/TAC.2010.2047541

Topaz, C. M., Ziegelmeier, L., and Halverson, T. (2015). Topological Data Analysis of Biological Aggregation Models. *PloS one* 10, e0126383. doi:10.1371/journal.pone.0126383

Ulmer, M., Ziegelmeier, L., and Topaz, C. M. (2019). A Topological Approach to Selecting Models of Biological Experiments. *PLOS ONE* 14, 1–18. doi:10.1371/journal.pone.0213679

Vanderbei, R. J. (2014). "Linear Programming Foundations and Extensions," in *International Series in Operations Research Management Science*. 4th ed. edn.v.2014 New York: Springer.

Vasudevan, R., Ames, A. D., and Bajcsy, R. (2011). Human Based Cost from Persistent Homology for Bipedal Walking," in *18th IFAC World Congress*. *IFAC Proc. Volumes* 44, 3292–3297. doi:10.3182/20110828-6-IT-1002.03807

Vicsek, T., Czirók, A., Ben-Jacob, E., Cohen, I., and Shochet, O. (1995). Novel Type of Phase Transition in a System of Self-Driven Particles. *Phys. Rev. Lett.* 75, 1226–1229. doi:10.1103/PhysRevLett.75.1226

Wu, P., Chen, C., Wang, Y., Zhang, S., Yuan, C., Qian, Z., et al. (2017). "Optimal Topological Cycles and Their Application in Cardiac Trabeculae Restoration," in *Proceedings of the Twenty- IJCAI-19*, International Joint Conferences on 1136 Artificial Intelligence Organization, 80–92.

Xia, K., and Wei, G. W. (2014). Persistent Homology Analysis of Protein Structure, Flexibility, and Folding. *Int. J. Numer. Methods Biomed. Eng.* 30, 814–844. doi:10.1002/cnm.2655

Xia, K., Li, Z., and Mu, L. (2018). Multiscale Persistent Functions for Biomolecular Structure Characterization. *Bull. Math. Biol.* 80, 1–31. doi:10.1007/s11538-017-0362-6

Zhang, X., Wu, P., Yuan, C., Wang, Y., Metaxas, D., and Chen, C. (2019). Heuristic Search for Homology Localization Problem and its Application in Cardiac Trabeculae Reconstruction. *Proceedings of the Twenty-Eighth1135International Joint Conference on Artificial Intelligence, IJCAI-19*, 1312–1318. doi:10.24963/ijcai.2019/182

Zomorodian, A., and Carlsson, G. (2005). Computing Persistent Homology. *Discrete Comput. Geometry* 33, 249–274. doi:10.1007/s00454-004-1146-y

Conflict of Interest: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.