Reduction Algorithms for Persistence Diagrams of Networks: CoralTDA and PrunIT

Cuneyt G. Akcora
Department of Computer Science
University of Manitoba
cuneyt.akcora@umanitoba.ca

Yulia R. Gel
Department of Mathematical Sciences
University of Texas at Dallas
National Science Foundation
ygl@utdallas.edu

Murat Kantarcioglu
Department of Computer Science
University of Texas at Dallas
muratk@utdallas.edu

Baris Coskunuzer
Department of Mathematical Sciences
University of Texas at Dallas
coskunuz@utdallas.edu

Abstract

Topological data analysis (TDA) delivers invaluable and complementary information on the intrinsic properties of data inaccessible to conventional methods. However, high computational costs remain the primary roadblock hindering the successful application of TDA in real-world studies, particularly with machine learning on large complex networks.

Indeed, most modern networks such as citation, blockchain, and online social networks often have hundreds of thousands of vertices, making the application of existing TDA methods infeasible. We develop two new, remarkably simple but effective algorithms to compute the exact persistence diagrams of large graphs to address this major TDA limitation. First, we prove that $(k+1)$-core of a graph $\mathcal{G}$ suffices to compute its k^{th} persistence diagram, $PD_k(\mathcal{G})$. Second, we introduce a pruning algorithm for graphs to compute their persistence diagrams by removing the dominated vertices. Our experiments on large networks show that our novel approach can achieve computational gains up to 95%.

The developed framework provides the first bridge between the graph theory and TDA, with applications in machine learning of large complex networks. Our implementation is available at github.com/cakcora/PersistentHomologyWithCoralPrunit.

1 Introduction

Topological data analysis (TDA) has emerged as powerful machinery in machine learning (ML), allowing us to extract complementary information on the observed objects, especially, from graph-structured data. In particular, TDA has become quite popular in various ML tasks, ranging from bioinformatics [38, 44], finance [39, 2] material science [33], biosurveillance [51, 19], network analysis [54, 16], as well as insurance and agriculture [56, 34] (see the literature overviews [4, 17] and the TDA applications library [27]). Recently there has emerged a highly active research area that combines the PH machinery with geometric deep learning (GDL) methods [30, 57, 31].

Persistent homology (PH) is a key approach in TDA, allowing us to extract the evolution of subtler patterns in the data shape dynamics at multiple resolution scales, which are not accessible to more conventional, non-topological methods [15]. The main idea is to construct a nested sequence of topological spaces (filtration) induced from the data, and record the evolution of topological features in this sequence. In other words, the extracted patterns, or homological features, along with how

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

long such features persist throughout the considered filtration of a scale parameter, convey a critical insight into salient graph characteristics and hidden mechanisms behind system organization.

PH has been very effective in many graph machine learning tasks, such as graph and node classification [49, 14, 58, 29], link prediction [6, 55] and anomaly detection [12, 47].

Nevertheless, while PH has shown promise in various graph learning applications, prohibitive computational costs of PH constrain its wider usage. Indeed, most PH studies are limited to small graphs with a few thousand vertices at most. The problem is that the complexity of the standard PH algorithm is cubic in the number of simplices [48], so one needs to limit homology computations to 0-th and 1-th levels only. Computation of higher-level persistence for relatively large graphs can take days or weeks.

In this paper, we aim to address this fundamental bottleneck in the application of TDA to large networks by introducing two new efficient algorithms which significantly reduce the cost of computing persistence diagrams (PD) for large real-world networks: *CoralTDA* and *PrunIT*.

CoralTDA Algorithm: Based on our observation that many vertices in large real-world networks have low degrees and do not contribute to PDs in higher dimensions, we developed the CoralTDA algorithm (Theorem 2) where we prove that $(k+1)$-core $\mathcal{G}^{k+1}$ of a graph $\mathcal{G}$ is enough to compute the k^{th} PD of the graph, i.e. $PD_k(\mathcal{G}) = PD_k(\mathcal{G}^{k+1})$.

Using this property, with a much smaller core graph $\mathcal{G}^{k+1}$, we compute the exact higher persistence diagram $PD_k(\mathcal{G})$ losing no information. Our experiments show that even for lower dimensional topological features, such as $k = 1$, we reduce the graph order by up to 73% for some datasets (See Figure 4). Our findings show that many real-life data sets exhibit nontrivial second and third persistence diagrams, facilitating various classification problems. On the other hand, our reduction reaches 100% for the third or higher dimensions in several networks, implying that higher PDs are trivial for these datasets.

As a result, our reduction approach improves our understanding of the existence of higher-order dimensional holes and their role in the organization of complex networks.

PrunIT Algorithm: We further develop a topologically simple but highly efficient algorithm to facilitate computations of PDs of graphs for any dimension. In particular, for a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and filtration by clique complexes, we show that removing (pruning) a dominated vertex from the graph does not change PDs at any level, provided that the dominated vertex enters the filtration after the dominating vertex (Theorem 7).

Our experiments indicate that the new algorithm is highly efficient in PD computations of a broad category of large graphs from 100K to 1M vertices, and it can reach 95% vertex reduction (see Table 1).

Further, when we combine CoralTDA and PrunIT algorithms, we can significantly reduce the graph sizes for the computation of PDs (Figure 6).

We summarize the key novelty of our contributions as follows:

- We show that the graphs' $(k+1)$th and higher persistence diagrams only depend on their k-cores.
- We introduce a highly effective pruning algorithm that significantly reduces the graph size without changing any persistence diagram of the original graph.
- Our experiments in large datasets and large graphs show up to 95% reduction in the graph size for the computation of persistence diagrams.
- With our reduction algorithms, highly successful TDA methods can be applied to very large graphs and large datasets where previously its use was constrained by prohibitive computational costs.

2 Related Work

There are mainly two settings in practice where we use PH to obtain a topological fingerprint of a dataset. The first one is the *point cloud setting*, where the dataset comes as a point cloud in an

ambient space $\mathbb{R}^n$. Then, we define PH by constructing a sequence of simplicial complexes induced by the pair-wise distances of data points (Vietoris-Rips filtration) and keeping track of the topological changes in this sequence [59, 23]. The second one is *the network setting* where the typical PH construction uses a filtering function on the network. By construction, while the principal identifier to define PH in the point cloud setting is the pair-wise distances of points, the principal identifier in the network setting is the filtering function. Because of this, PH machinery works differently in a network setting, as explained in Section 3.

There are several works in the point cloud setting to reduce the computational costs and run-time of the persistence diagrams. Malott and Wilsey used the idea of data reduction and data partitioning [41]. Mischaikow and Nanda brought the discrete Morse Theory of geometric topology to the combinatorial setting [43]. In [46, 20, 21, 24], the authors studied the same problem with different approaches in the point cloud setting.

While several works improve the run-time of PH in the point cloud setting, only a few of them could reduce the computational costs of persistent homology in the network setting. An idea is to use discrete Morse Theory to capture the topological features occurring during the process [36] by applying the techniques developed in [43] to the network setting.

While the computational complexity of k^{th} persistence diagram (PD) is $\mathcal{O}(n^3)$ where n is the number of k-simplices [48], [43] achieves $\mathcal{O}(m^2 \times n \log n)$ where m is the number of critical k-simplices. With the additional time to find the critical k-simplices in each filtration step, the computational complexity $\mathcal{O}(m^2 \times n \log n)$ is not scalable for very large networks.

3 Persistent Homology

This part provides a background on the theory of persistent homology. Homology $H_k(X)$ is an essential invariant in algebraic topology, which captures the information of the k-dimensional holes (connected components, loops, cavities) in a topological space X. For example, a connected component in a graph is a zero-dimensional hole, whereas a graph loop is a 1-dimensional hole. Persistent homology is a way to use this invariant to keep track of the changes in a controlled topological space sequence induced by the original space X. For basic background on persistent homology, see [23, 22].

There are several ways to use PH in a network setting, such as power filtration or using different complexes (e.g., Vietoris-Rips, Čech complexes) to construct the filtration for a given filtering function [3]. We focus on the most common methods to define PH for graphs: sub/superlevel filtrations obtained by a filtering function and the clique (flag) complexes. Sub/superlevel filtrations are the most common methods because one can inject domain information into the PH process if the chosen filtering function comes from the network domain (e.g., atomic number in protein networks, transaction amount for blockchain networks). Note that our results can be generalized to the persistent homology defined with a filtering function for different complexes.

Throughout the paper, we use the terms *graph* and *network* interchangeably. Let $\mathcal{G}$ be a graph with vertex set $\mathcal{V} = \{v_r\}$ and edge set $\mathcal{E} = \{e_{rs}\}$, i.e. $e_{rs} \in \mathcal{E}$ if there is an edge between the vertex v_r and v_s in $\mathcal{G}$. Let $f : \mathcal{V} \to \mathbb{R}$ be a filtering function defined on the vertices of $\mathcal{G}$. Let $\mathcal{I} = \{\alpha_i\}$ be a threshold set with $\alpha_0 = \min_{v_r \in \mathcal{V}} f(v_r) < \alpha_1 < ... < \alpha_m = \max_{v_r \in \mathcal{V}} f(v_r)$. For $\alpha_i \in \mathcal{I}$, let $\mathcal{V}_i = \{v_r \in \mathcal{V} \mid f(v_r) \leq \alpha_i\}$. Let $\mathcal{G}_i$ be the induced subgraph of $\mathcal{G}$ by $\mathcal{V}_i$, i.e. $\mathcal{G}_i = (\mathcal{V}_i, \mathcal{E}_i)$ where $\mathcal{E}_i = \{e_{rs} \in \mathcal{E} \mid v_r, v_s \in \mathcal{V}_i\}$. Let $\widehat{\mathcal{G}}_i$ be the clique complex of $\mathcal{G}_i$. A clique complex is obtained by filling in all the $(k+1)$-complete subgraphs with k-simplices. In other words, if the vertices $\{v_{r_0}, v_{r_1}, ..., v_{r_k}\} \subset \mathcal{G}_i$ are pairwise connected by an edge in $\mathcal{G}$, then the clique complex $\widehat{\mathcal{G}}_i$ contains a k-simplex $\sigma = [v_{r_0}, v_{r_1}, ..., v_{r_k}]$. This simplicial complex $\widehat{\mathcal{G}}_i$ obtained by filling in all complete subgraphs is called *the clique complex* of $\mathcal{G}_i$. This construction induces a nested sequence of high dimensional simplicial complexes:

$$\widehat{\mathcal{G}}_0 \subset \widehat{\mathcal{G}}_1 \subset \widehat{\mathcal{G}}_2 \subset ... \subset \widehat{\mathcal{G}}_m.$$

This sequence of simplicial complexes is called *the sublevel filtration* for $\mathcal{G}$. Superlevel filtrations can be defined similarly by considering the generating sets $\{f(v_r) \geq \alpha_i\}$ instead of $\{f(v_r) \leq \alpha_i\}$ above. Here, $\widehat{\mathcal{G}}_i$ can be taken as the different simplicial complexes induced by $\mathcal{G}_i$ which gives different

types of filtrations [3]. After obtaining the filtration, one considers the homology groups $H_k(\widehat{\mathcal{G}}_i)$ of each simplicial complex $\widehat{\mathcal{G}}_i$. The homology group $H_k(X)$ keeps the information of k-dimensional topological features in the simplicial complex X.

Persistent homology keeps track of the topological changes in the sequence $\{\widehat{\mathcal{G}}_i\}$ by using the homology groups $\{H_k(\widehat{\mathcal{G}}_i)\}$. When a k-dimensional hole σ (a connected component, loop or cavity) appears in $H_k(\widehat{\mathcal{G}}_i)$, we mark $b_\sigma = \alpha_i$ as its birth time. The feature σ can disappear at a later time in $H_k(\widehat{\mathcal{G}}_j)$ by merging with another feature or by being filled in. Then, we mark $d_\sigma = \alpha_j$ as its death time. Hence, we say that σ persists along the interval $[b_\sigma, d_\sigma)$, i.e. $[\alpha_i, \alpha_j)$. The longer the interval $(d_\sigma - b_\sigma)$, the more persistent the feature σ.

The multi-set $PD_k(\mathcal{G}, f) = \{(b_\sigma, d_\sigma) \mid \sigma \in H_k(\widehat{\mathcal{G}}_i) \text{ for } b_\sigma \leq i < d_\sigma\}$ is called the k^{th} *persistence diagram* of $(\mathcal{G}, f)$ which is the collection of 2-tuples marking the birth and death times of k-dimensional holes $\{\sigma\}$ in $\{\widehat{\mathcal{G}}_i\}$. In particular, $PD_k(\mathcal{G}, f)$ represents the k^{th} PD of the sublevel filtration, induced by the filtering function $f : \mathcal{V} \to \mathbb{R}$. For brevity, we suppress f and use $PD_k(\mathcal{G})$ throughout the text.

4 CoralTDA Reduction and Higher Persistence Diagrams

A *k-core* $\mathcal{G}^k$ of a graph $\mathcal{G}$ is the subgraph of $\mathcal{G}$ obtained by iteratively deleting all vertices (and edges connected to it) with degree less than k [52]. In other words, $\mathcal{G}^k$ is the largest subgraph of $\mathcal{G}$ where all the vertices have a degree of at least k.

Figure 1 shows a graph with its core structure. Here, vertex 1 belongs to 0-core as it is disconnected from the graph. Vertex colors indicate shared coreness. When we use vertex degree as the filtering function and allow graph cliques of size three at most, the only one-dimensional hole (shown with the red circle) appears at degree 4 for vertices 4, 6, 8, and 9. Vertices 3, 5, and 7 can only contribute to 0-dimensional holes because their degree is 1. Similarly, 8 can only contribute to 0 and 1-dimensional holes because its degree is 2.

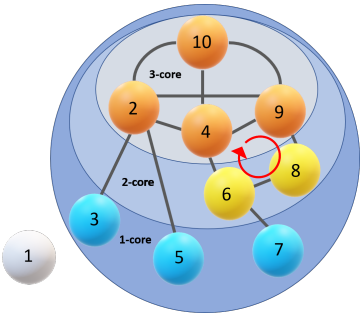


Figure 1: K-core decomposition of a graph of 10 vertices. Vertex 1 has no edges and belongs to the $0th$-core. A one-dimensional hole of vertices 4, 6, 8, and 9 is shown with a red circle.

The k-core decomposition is a fundamental operation in many areas such as graph similarity matching [45], graph clustering [25], network visualization [26], anomaly detection [53] and robustness analysis [13].

A naïve implementation of k-core iteratively deletes vertices whose degree falls below a k, until it deletes all vertices from the graph. The implementation has a computational complexity of $\mathcal{O}(m \log n)$, where m and n are the number of edges and vertices in the network, respectively. Batagelj and Zaversnik reduce the complexity to $\mathcal{O}(m + n)$ "by keeping an in-memory array of all possible degree values and keeping track of bin boundaries" [5].

4.1 Relation between $\widehat{\mathcal{G}}_i$ and $\widehat{\mathcal{G}}_i^k$

Our main idea is to compute high-dimensional persistence features on their associated graph cores. Note that a k-clique in graph theory corresponds to a $(k-1)$-simplex in PH; a k-clique (complete subgraph of order k) in $\mathcal{G}$ induces a $(k-1)$-simplex in $\widehat{\mathcal{G}}$.

The clique complex $\widehat{\mathcal{G}}$ is a simplicial complex of dimension $K-1$ where K denotes the degeneracy of $\mathcal{G}$, i.e. $K = \max\{k \mid \mathcal{G}^k \neq \emptyset\}$. That is, $\widehat{\mathcal{G}}$ contains a $(k-1)$-simplex if and only if its k-core $\mathcal{G}^k$ is not empty.

For any i, k, we have the k-core of $\mathcal{G}_i$ contained in $\mathcal{G}_i$ by construction, i.e. $\mathcal{G}_i^k \subset \mathcal{G}_i$. This implies that the same holds for their clique complexes, i.e. $\widehat{\mathcal{G}}_i^k \subset \widehat{\mathcal{G}}_i$. On the other hand, if one restricts the original filtering function $f : \mathcal{V} \to \mathbb{R}$ to the vertices $\mathcal{V}^k$ of the k-core of $\mathcal{G}$, we have $f : \mathcal{V}^k \to \mathbb{R}$. By

using the same thresholds for $f : \mathcal{V}^k \to \mathbb{R}$, we obtain the filtration $\widehat{\mathcal{G}}_0^k \subset \widehat{\mathcal{G}}_1^k \subset \widehat{\mathcal{G}}_2^k \subset ... \subset \widehat{\mathcal{G}}_m^k$. This will induce the persistence diagram $PD_r(\mathcal{G}^k)$ for any dimension r.

Since for any i, k, $\widehat{\mathcal{G}}_i^k \subset \widehat{\mathcal{G}}_i$, we have the following diagram.

$$\begin{array}{ccccccc} \widehat{\mathcal{G}}_0^k & \subset & \widehat{\mathcal{G}}_1^k & \subset & ... & \subset & \widehat{\mathcal{G}}_m^k \\ \cap & & \cap & & & & \cap \\ \widehat{\mathcal{G}}_0 & \subset & \widehat{\mathcal{G}}_1 & \subset & ... & \subset & \widehat{\mathcal{G}}_m \end{array} \tag{1}$$

Notice that for any $j \geq k-1$, if there is a j-cycle σ living in $C_j(\widehat{\mathcal{G}}_i^k)$, then we have $\sigma \subset C_j(\widehat{\mathcal{G}}_i)$ as $\widehat{\mathcal{G}}_i^k \subset \widehat{\mathcal{G}}_i$. In the following, we show that for these cycles, the converse is also true, and we show the equivalence in the homology level.

Remark 1. [Restriction of f to $\mathcal{V}^k$] Notice that the filtering function $f : \mathcal{V}^k \to \mathbb{R}$ on $\mathcal{V}^k$, the vertices of $\mathcal{G}^k$, is defined directly by restricting values of $f : \mathcal{V} \to \mathbb{R}$ to the subset $\mathcal{V}^k \subset \mathcal{V}$. In particular, if $f : \mathcal{V} \to \mathbb{R}$ is a function coming from the graph attributes (such as vertex degree), then $f : \mathcal{V}^k \to \mathbb{R}$ may not be the same function coming from the graph attributes induced by the graph $\mathcal{G}^k$. For example, let f be the degree function on $\mathcal{V}$, the vertices of $\mathcal{G}$. Then, for any $w \in \mathcal{V}^k$, $f(w)$ is the degree of w in $\mathcal{G}$, *not its degree in* $\mathcal{G}^k$. While the k-core graph $\mathcal{G}^k$ changes, we *do not update* the values of f on $\mathcal{V}^k$ according to its attribute definition in $\mathcal{G}^k$, but we keep the same values in the original function $f : \mathcal{V} \to \mathbb{R}$ for the remaining vertices in $\mathcal{V}^k \subset \mathcal{V}$. In graph terms, this corresponds to computing vertex filtering (activation) values on the original graph but using the edges of the reduced graph to extract simplices.

4.2 CoralTDA Reduction

Our CoralTDA technique shows that lower degree vertices do not affect higher persistence diagrams, i.e., CoralTDA yields exact results. Note that in the following result, even though the graph size changes, we keep the same filtering function $f : \mathcal{V} \to \mathbb{R}$ with the original values. See Remark 1 for further details. We give the proof of the following theorem in Appendix.

Theorem 2. *Let $\mathcal{G}$ be an unweighted connected graph. Let $f : \mathcal{V} \to \mathbb{R}$ be a filtering function on $\mathcal{G}$. Let $PD_k(\mathcal{G}, f)$ represent the k^{th} persistence diagram for the sublevel filtration of the clique complexes. Let $\widehat{\mathcal{G}}^k$ be the k-core of $\mathcal{G}$. Then, for any $j \geq k$*

$$PD_j(\mathcal{G}, f) = PD_j(\mathcal{G}^{k+1}, f).$$

Outline of the proof: We show that for any nontrivial k-homology class σ in the original clique complex $\widehat{\mathcal{G}}$, a generating k-cycle S in this homology class also lives in a much smaller subcomplex: the clique complex of the $(k+1)$-core ($\widehat{\mathcal{G}}^{k+1}$). That is, we prove that any vertex in the k-cycle S must have a degree at least $k+1$ where this degree count comes only from the k-simplices of S, and removing the lower degree vertices from $\mathcal{G}$ has no effect on the existence of such S. We give the proof of the theorem in Appendix.

The above result indicates that k^{th} persistence diagram information can be obtained by only considering the $(k+1)$-core of a graph. CoralTDA is an effective tool for reducing computational costs to compute higher persistence diagrams. See Figure 4 for reduction results for various datasets.

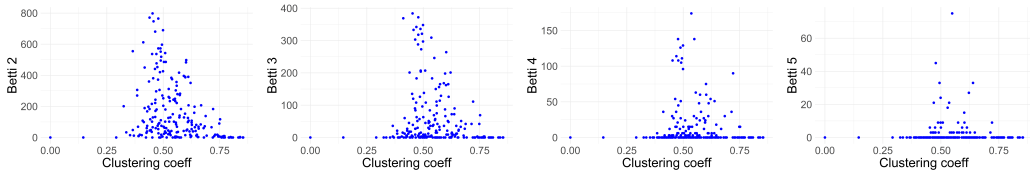


Figure 2: Clustering coefficients vs. number of topological features in Facebook and Twitter datasets. Each data point is a graph instance. We observe hundreds of higher topological features in these datasets which can be highly useful for various graph learning tasks.

Remark 3. [Higher PDs in Random Networks vs. Real-Life Networks] Note that by Kahle's seminal result [35], to observe nontrivial Betti numbers for higher dimensions in Erdós-Rényi graphs $G(n, p)$,

the average degree must be very high. In particular, for a graph $G(n,p)$, in order to have nontrivial k^{th}-homology in its clique complex, Kahle proved that for $p = n^{\alpha}$, α should be between $-1/k$ and $-1/(k+1)$. In terms of average degree $n \times p$, this means the average degree should be between $n^{(k-1)/k}$ and $n^{k/(k+1)}$. For instance, for dimension $k = 2$, the average degree should be between $\sqrt{n}$ and $\sqrt[3]{n^2}$. For a graph order of $n = 1000$, this implies that the average degree should be between 31 and 100 to have a nontrivial second homology in random networks. However, in real-life networks, our results show that higher Betti numbers are prevalent in much sparser graphs (see fig. 2 and appendix fig. 10). These findings can be further used to derive error bounds and the associated loss of topological information when $G(n,p)$ is employed to approximate real-world network phenomena, for instance, in the case of synthetic power grid networks and other cyber-physical systems.

In the following, we give another effective method to reduce the size of a graph $\mathcal{G}$ without affecting the persistence diagrams $PD_r(\mathcal{G})$ *for any dimension* $r \geq 0$.

5 PrunIT Algorithm

This section introduces another effective reduction technique for computing persistence diagrams of graphs induced by a filtering function. In particular, we show that for a graph $\mathcal{G}$, and filtering function $f : \mathcal{V} \to \mathbb{R}$, removing (pruning) specific vertices from the graph does not change the persistent homology at any level. The result is valuable because the algorithm may reduce the vertex set considerably (Table 1). Furthermore, as our experiments show, the reduced vertex set can significantly lower the simplex count, leading to much shorter computational times for persistent homology (see Figure 4 and appendix Figure 7).

In algebraic topology, homotopy is a very effective tool to compute topological invariants like homology, and fundamental group [28]. These topological invariants are homotopy invariant, meaning that if two spaces are homotopy equivalent, then their corresponding topological invariants are the same, e.g., $X \sim Y \Rightarrow H_i(X) = H_i(Y)$. We give a very natural homotopy construction to simplify a graph in the following.

For a given filtering function $f : \mathcal{V} \to \mathbb{R}$, let $\widehat{\mathcal{G}}_i$ be the clique complex of $\mathcal{G}_i$ which induces the sublevel filtration $\widehat{\mathcal{G}}_0 \subset \widehat{\mathcal{G}}_1 \subset \widehat{\mathcal{G}}_2 \subset ... \subset \widehat{\mathcal{G}}_m$. Let $PD_k(\mathcal{G}, f)$ represent the k^{th} persistence diagram for the sublevel filtration $\{\widehat{\mathcal{G}}_i\}$ as described above.

Now, we define *dominated vertices* in $\mathcal{G}$. Define the neighborhood of u_0 as $N(u_0) = \{u_0\} \cup \{v \in \mathcal{V} \mid e_{u_0 v} \in \mathcal{E}\}$. In particular, $N(u_0) \subset \mathcal{V}$ is the set of all vertices adjacent to u_0, and u_0 itself.

Definition 4. *A vertex u is dominated by the vertex v in $\mathcal{G}$ if $N(u) \subset N(v)$. If there is such a vertex v, we call u a dominated vertex of $\mathcal{G}$ (see Figure 3).*

Removing a vertex u from a graph $\mathcal{G}$ creates the natural subgraph of $\mathcal{G}$ obtained by removing the vertex u and all adjacent edges from $\mathcal{G}$, i.e. $\mathcal{G} - \{u\} = \mathcal{G}' = (\mathcal{V}', \mathcal{E}')$ where $\mathcal{V}' = \mathcal{V} - \{u\}$, and $\mathcal{E}' = \mathcal{E} - \{e_{uw} \in \mathcal{E}\}$ for any w.

We can alternatively express these via the star notion. The *star* $\mathbf{St}(u)$ of a vertex u is the union of all simplices which contains u. Then, u is dominated by v if $\mathbf{St}(u) \subset \mathbf{St}(v)$. Similarly, removing a vertex u from $\mathcal{G}$ corresponds to removing $\mathbf{St}(u)$ from the clique complex $\widehat{\mathcal{G}}$, i.e. $\widehat{\mathcal{G}} - \mathbf{St}(u) = \widehat{\mathcal{G}}'$. A useful result is that removing a dominated vertex does not affect the homotopy type of the corresponding clique complexes.

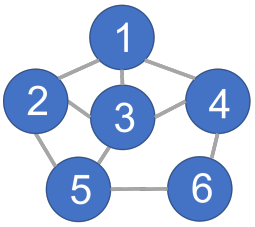


Figure 3: Vertex 3 dominates vertices 1 and 2 because all neighbors of 1 or 3 are neighbors of 3. There are no other dominated vertices.

Lemma 5. *Let u be a dominated vertex in $\mathcal{G}$. Let $\mathcal{G}' = \mathcal{G} - \{u\}$. Then the clique complexes $\widehat{\mathcal{G}}$ and $\widehat{\mathcal{G}}'$ are homotopy equivalent, i.e. $\widehat{\mathcal{G}} \sim \widehat{\mathcal{G}}'$.*

Proof: Notice that $\mathcal{G}'$ is a subgraph of $\mathcal{G}$, and hence $\widehat{\mathcal{G}}'$ is a subcomplex in $\widehat{\mathcal{G}}$. Let u be dominated by v in $\mathcal{G}$. Then, we can write a deformation retract from $\widehat{\mathcal{G}}$ to $\widehat{\mathcal{G}}'$ by pushing the edge e_{uv} starting from u toward v. In other words, by using the simplicial coordinates, one can define a homotopy $F : \widehat{\mathcal{G}} \times I \to \widehat{\mathcal{G}}$ which is identity on $\widehat{\mathcal{G}}'$ and pushing all the faces in $\widehat{\mathcal{G}} - \widehat{\mathcal{G}}'$ to the corresponding faces

in $\widehat{\mathcal{G}}'$. This gives a homotopy equivalence $\widehat{\mathcal{G}} \sim \widehat{\mathcal{G}}'$. To visualize, in Figure 3, one can push vertex 1 in the clique complex $\widehat{\mathcal{G}}$ towards vertex 3 along the edge between them. After the push, the 2-simplices [1,2,3] and [1,3,4] are pushed to the edges [2,3] and [3,4] respectively. See [1, 7] and [11, Lemma 2.2] for details. □

Remark 6. [Collapsing] Note that this collapsing operation is adaptation of a well-known notion called *deformation retract* in algebraic topology in a simplicial complex setting [28]. This operation keeps the homotopy type the same, and hence the homology does not change with this reduction. In [1, 7, 11], this is called *folding* ($\mathcal{G}$ folds onto $\mathcal{G} - \{u\}$) or a *strong collapse*. In these papers, the algorithm reduces simplicial complexes in the filtration one by one so that its associated clique complex keeps the same homotopy type. Our contribution here is to adapt this operation to the graph filtrations and define a smaller subgraph before the filtration step so that the induced simplicial complexes are homotopy equivalent. Since we prune the graph at the beginning of the process, our algorithm significantly reduces the computational costs for the induced persistence diagrams.

In the following, we introduce the *PrunIT Algorithm* by showing that removing a dominated vertex does not change the persistence diagrams of the graph. We give the proof in Appendix C.

Theorem 7. *Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be an unweighted graph, and $f : \mathcal{V} \to \mathbb{R}$ be a filtering function. Let $u \in \mathcal{V}$ be dominated by $v \in \mathcal{V}$ and $f(u) \geq f(v)$. Then, removing u from $\mathcal{G}$ does not change the persistence diagrams for sublevel filtration, i.e. for any $k \geq 0$*

$$PD_k(\mathcal{G}, f) = PD_k(\mathcal{G} - \{u\}, f).$$

Outline of the proof: The main idea is to employ the collapsing idea in the simplicial complexes of the filtration $\widehat{\mathcal{G}}_0 \subset \widehat{\mathcal{G}}_1 \subset \widehat{\mathcal{G}}_2 \subset \cdots \subset \widehat{\mathcal{G}}_m$ in a suitable way. In particular, the lemma above shows that if a vertex u is dominated by a vertex v in $\widehat{\mathcal{G}}_i$, then removing $\mathbf{St}(u)$ from $\widehat{\mathcal{G}}_i$ does not change the homotopy type. Hence, if we ensure that when u first appears in the filtration $\{\widehat{\mathcal{G}}_i\}$, the dominated vertex v is already there, then u can be removed from all the simplicial complexes in the filtration; removing u from the original graph before building the simplicial complexes does not affect the homotopy type of complexes in the filtration. The condition $f(u) \geq f(v)$ makes sure that whenever u exists in $\{\widehat{\mathcal{G}}_i\}$, the dominant vertex is already there, and u can be removed from all simplicial complexes, and hence from the graph $\mathcal{G}$. We give the proof of the theorem in Appendix C.

Notice that the primary condition to remove dominated vertices from the graph ensures that the dominated vertex enters the filtration after its dominating counterpart. With the PrunIT Algorithm, we show that removing the dominated vertex does not change the homotopy type of the simplicial complexes in the filtration. As homotopy equivalence implies the equivalences of homology groups at all levels, the reduction with this algorithm works in all dimensions. Furthermore, while coral reduction works above the corresponding dimension ($j > k$), the PrunIT algorithm works in any dimension.

Remark 8. [Superlevel Filtration] The same proof applies to the superlevel filtration by changing the condition $f(u) \geq f(v)$ to $f(u) \leq f(v)$ in the theorem. In particular, if $PD_k^{\mathrm{v}}(\mathcal{G}, f)$ represents the k^{th} PD for superlevel filtration, then with the condition $f(u) \leq f(v)$, we would have $PD_k^{\mathrm{v}}(\mathcal{G}, f) = PD_k^{\mathrm{v}}(\mathcal{G} - \{u\}, f)$ for any $k \geq 0$. Notice that if one takes f to be the degree function and uses the superlevel filtration, then the theorem automatically holds for any dominated vertex as $deg(u) \leq deg(v)$ when u is dominated by v.

Remark 9. [Detecting Dominating Vertices] The dominating vertices can be computed by using the following approach (Algorithm is given in appendix Section B). Let $\mathcal{A} = (a_{ij})$ be the adjacency matrix for a graph $\mathcal{G}$. Given $v_{i_0} \in \mathcal{V}$, consider all j's with $a_{i_0 j} = 1$. Check if v_{i_0} is dominated by v_j by comparing the rows R_{i_0} and R_j, i.e. for any $k \neq j$ with $a_{i_0 k} = 1$, check whether $a_{j_k} = 1$. If this holds, v_j dominates v_{i_0}. Removing i_0^{th} row R_{i_0} and i_0^{th} column C_{i_0} from $\mathcal{A}$ corresponds to removing v_{i_0} from $\mathcal{G}$. Essentially, vertex v_{i_0} is compared to each neighbor v_j by checking whether v_{i_0} is already a neighbor of each of v_j's neighbors. These checks require iterating over each vertex, searching vertex neighbors in the graph and getting the neighbors of each neighbor. The computational complexity is therefore $\mathcal{O}(|\mathcal{V}| \times d^2)$ where d is the average degree in the graph.

While our main focus is the most common method, sublevel/superlevel filtration, in the application of PH in graph setting, our PrunIt algorithm works perfectly well with another common method, power filtration [3], as well, i.e. removing a dominated vertex does not change persistence diagrams.

Theorem 10. *[PrunIt for Power Filtration] Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be an unweighted connected graph. Let $\widehat{PD}_k(\mathcal{G})$ represent k^{th} persistence diagram of $\mathcal{G}$ with power filtration. Let $u \in \mathcal{V}$ be dominated by any other vertex in $\mathcal{V}$. Then, for any $k \geq 1$,*

$$\widehat{PD}_k(\mathcal{G}) = \widehat{PD}_k(\mathcal{G} - \{u\}).$$

The proof of this result is given in appendix Section C.2.

Combining the CoralTDA and PrunIT Algorithms

Even though both algorithms are quite effective by themselves, we significantly reduce the computational costs (Figure 6) by combining them as follows. For a given graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and filtering function $f : \mathcal{V} \to \mathbb{R}$, one can start by trimming all dominated vertices with respect to f, and get a smaller graph $\mathcal{G}'$. We have already proven that $PD_k(\mathcal{G}) = PD_k(\mathcal{G}')$. Then, one can take the k-core of this smaller graph $\mathcal{G}'$ to compute higher persistence diagrams of the original graph $\mathcal{G}$ as before. In particular, by applying both reduction algorithms, for any $k \geq 0$, we obtain

$$PD_k(\mathcal{G}) = PD_k(\mathcal{G}') = PD_k((\mathcal{G}')^{k+1}).$$

6 Experiments and Discussion

We apply our new approaches to three types of datasets. The details of datasets are provided in Table 1 and appendix Table 2.

Graph classification datasets consists of biological kernel [37] and ego networks from TWITTER and FACEBOOK [42]. *Node classification datasets* includes CITESEER and CORA [50] and Open Graph Benchmark citation (paper cites paper) OGB-ARXIV and OGB-MAG [32] networks. *Large networks dataset* contains 11 large networks of 100K-1M vertices from the Stanford Repository [40].

We used an AMD Ryzen 5 2100 MHZ 4 core computer in our R, Python and Java experiments.

We evaluate both algorithms by comparing vertex and edge sets and the total run time for the reduced graph with respect to the original graph. In the rest of this manuscript, we compute the vertex set reduction as $100 \times (|\mathcal{V}| - |\mathcal{V}'|)/|\mathcal{V}|$ where $\mathcal{V}'$ is the vertex count in the reduced graph. Edge and time reductions are computed similarly.

6.1 Reduction on Graph Classification Datasets

In this task, our goal is to evaluate the reduction of computational costs when we use the CoralTDA and PrunIT algorithms on datasets chosen from different graph classification tasks. We used one of the most commonly used functions in these experiments, the degree function with sublevel filtration.

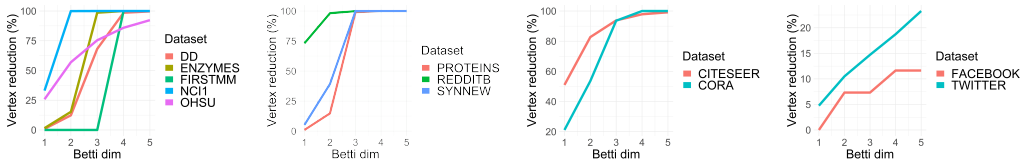
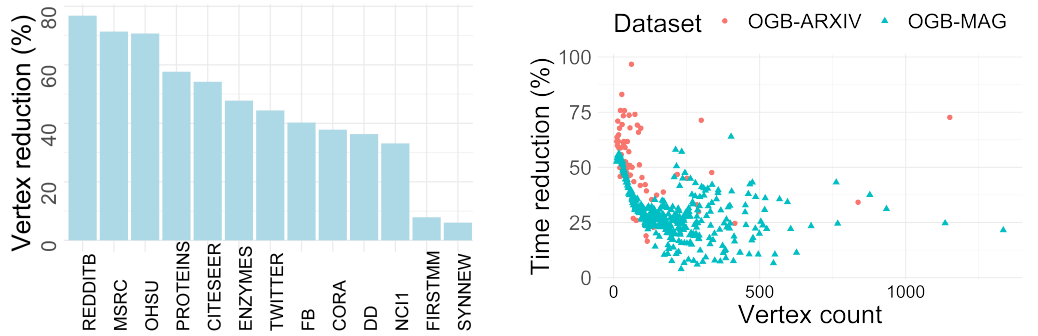


Figure 4: CoralTDA vertex reduction in graph and node classification datasets (higher is better). Reduction values are averages from graph instances of the datasets (CORA and CITESEER node classification datasets contain a single graph instance only). FACEBOOK and TWITTER datasets are reduced by 20% for $k > 4$, whereas in other datasets graphs are reduced to empty sets.

In Figure 4, we show the vertex reduction when using CoralTDA for computations of $PD_k(\mathcal{G})$ for dimensions (Betti) $k = 1$ to $k = 5$. At dimension $k = 4$ and $k = 5$, CoralTDA reduces 10 datasets by 100%, i.e., these datasets have trivial $PD_k(\mathcal{G})$ for $k \geq 4$. Even at smaller dimensions, CoralTDA can reduce the vertex set by 25%-75%.

Figure 5a shows reduction percentages by the PrunIT algorithm. FIRSTMM and SYNNEW datasets are reduced by less than 10%; however the other 11 datasets are reduced by at least 35%. The lower reduction on FIRSTMM and SYNNEW are due to stronger cores on the networks. SYNNEW is



(a) Vertex reduction by PrunIT algorithm in the super-level filtration. Results are averages of graph instances from the datasets.

(b) PrunIT reduction in OGB node classification dataset. Each data point is an ego network. Even for large networks, time reduction rates can reach 75%.

Figure 5: PrunIt vertex and time reduction in graph datasets.

synthetically created, but FIRSTMM is created from 3d point cloud data and categories of various household objects. We believe that the physical proximity of similar objects (e.g., chairs are close to each other) in a household creates a denser community structure in the FIRSTMM dataset, which in turn results in strong cores.

We further report reductions in computational time (Figure 8), edge set (Figure 9), and simplex count (Figure 7) in the Appendix.

6.2 Reduction on Node Classification Datasets

In this task, our goal is to compute the reduction of computational costs by using CoralTDA and PrunIT algorithms on datasets chosen from node classification tasks. The CoralTDA results are computed over CITESEER and CORA networks and shown in Figure 4 with more than 20% reduction for the first and higher dimensional persistence.

In node classification, we can also analyze the k-hop ($k \geq 1$) neighborhood of a vertex with topological features (such as Betti-0) and use the computed persistence diagram to classify the vertex. Such an approach has yielded SOTA results with significant improvement in accuracy by using 0-dimensional persistence [18]. However, the computational costs of persistent homology are non-negligible in large graphs, even for 0-dimensional features. For example, in Open Graph Benchmark datasets [32], one must compute persistence diagrams for each vertex in 100k to 111M vertex graphs.

We apply the PrunIT algorithm to two graphs, Arxiv and MAG, from the Open Graph Benchmark to compute time reduction in persistence diagram computations. We follow the approach in [18] and extract the 1-hop neighborhood of each ego vertex. We use the degree function as filtering function as before. In Figure 5b, we show the reduction in computational time for 0-dimensional persistence. We compute the time costs of PrunIT by considering all the algorithm steps: finding and removing the dominated vertices, creating an induced graph with the vertices, and running 0-dimensional persistent homology on the graph by using vertex degrees as the filtering function. As Figure 5b shows, we see more than 25% reduction in computation time in most graphs. Specifically, on average, computation times of 0-dimensional persistence on OGB-ARXIV networks are reduced by 37%, and those of OGB-MAG networks are reduced by 23%. The results show that we can mitigate the computational costs of persistence homology by using the PrunIT algorithms.

6.3 Reduction on Large Networks

Our goal is to combine PrunIt and CoralTDA algorithms to achieve the maximum vertex and edge reduction in large networks in these experiments.

Table 1 shows that on the biggest network of com-youtube, we eliminate 59% of the vertices when we only apply the PrunIt (on average 62% in all datasets). The reduction is as high as 95% (in emailEuAll). Similarly, PrunIt creates significant edge reduction; 40% of all edges are removed on

Table 1: PrunIt reductions in the number of vertices and edges.

Dataset	$\|V\|$	$\|V\|$ Reduction (↑)	$\|E\|$	$\|E\|$ Reduction (↑)
com-youtube	1134890	59%	2987624	25%
com-amazon	334863	37%	925872	40%
com-dblp	317080	72%	1049866	65%
web-Stanford	281903	67%	1992636	76%
emailEuAll	265214	95%	364481	94%
soc-Epinions1	75879	57%	405740	14%
p2pGnutella31	62586	46%	147892	20%
Brightkite_edges	58228	48%	214078	21%
Email-Enron	36692	76%	183831	38%
CA-CondMat	23133	69%	93439	65%
oregon1_010526	11174	62%	23409	48%

average. Figure 6 shows the reduction when we apply both CoralTDA and PrunIt on large networks. Even for low cores of 2 and 3, the combined algorithms reach a vertex reduction rate of 78%. These results show that our algorithms can effectively reduce large networks to more manageable sizes.

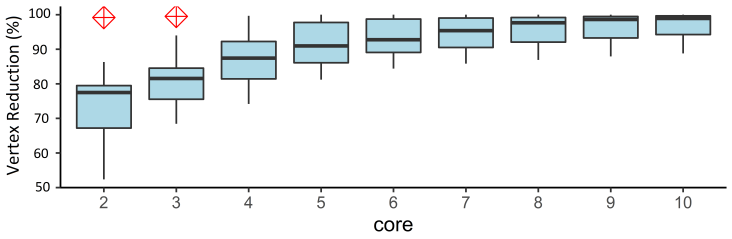


Figure 6: Vertex reduction results for 11 large datasets after the application of PrunIt and CoralTDA algorithms. `emailEuAll` is the outlier for the 2nd and 3rd cores (shown with a crossed square).

7 Conclusion

We have proposed two new highly effective algorithms to significantly reduce the computational costs of TDA methods on graphs. While coral reduction is very effective for higher persistence diagrams, PrunIt is highly efficient, in general. Our experiments have showed that even for lower dimensional topological features, such as $k = 1$, for some datasets our methods can reduce graph order by up to 95%, which alleviates computational costs substantially. Furthermore, in most graph datasets we reduce graph sizes by 100% for 3rd or higher dimensions. Our methods provides a novel solution for efficient application of the powerful TDA methods on large networks and build a bridge between the graph theory and TDA, opening a pathway for broader applicability of topological graph learning in practice.

8 Acknowledgments

This material is based upon work sponsored by the Canadian NSERC Discovery Grant RGPIN-2020-05665, NSF of USA under award number ECCS 2039701, OAC-1828467, DMS-1925346, CNS-2029661, OAC-2115094, ARO award W911NF-17-1-0356, and Simons Collaboration Grant # 579977. Part of this material is also based upon work supported by (while serving at) the National Science Foundation. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

[1] Michał Adamaszek. Clique complexes and graph powers. *Israel Journal of Mathematics*, 196(1):295–319, 2013.

[2] Cuneyt Gurcan Akcora, Yitao Li, Yulia R. Gel, and Murat Kantarcioglu. Bitcoinheist: Topological data analysis for ransomware prediction on the bitcoin blockchain. In Christian Bessiere,

editor, *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, pages 4439–4445. Ijcai.org, 2020.

[3] Mehmet E Aktas, Esra Akbas, and Ahmed El Fatmaoui. Persistence homology of networks: methods and applications. *Applied Network Science*, 4(1):1–28, 2019.

[4] Erik J Amézquita, Michelle Y Quigley, Tim Ophelders, Elizabeth Munch, and Daniel H Chitwood. The shape of things to come: Topological data analysis and biology, from molecules to organisms. *Developmental Dynamics*, 249(7):816–833, 2020.

[5] Vladimir Batagelj and Matjaz Zaversnik. An o(m) algorithm for cores decomposition of networks. *CoRR*, cs.DS/0310049, 2003.

[6] Austin R Benson, Rediet Abebe, Michael T Schaub, Ali Jadbabaie, and Jon Kleinberg. Simplicial closure and higher-order link prediction. *Proceedings of the National Academy of Sciences*, 115(48):E11221–E11230, 2018.

[7] Jean-Daniel Boissonnat and Siddharth Pritam. Computing persistent homology of flag complexes via strong collapses. In Gill Barequet and Yusu Wang, editors, *35th International Symposium on Computational Geometry, SoCG 2019, June 18-21, 2019, Portland, Oregon, USA*, volume 129 of *LIPIcs*, pages 55:1–55:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

[8] Jean-Daniel Boissonnat and Siddharth Pritam. Edge collapse and persistence of flag complexes. In Sergio Cabello and Danny Z. Chen, editors, *36th International Symposium on Computational Geometry, SoCG 2020, June 23-26, 2020, Zürich, Switzerland*, volume 164 of *LIPIcs*, pages 19:1–19:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[9] Jean-Daniel Boissonnat, Siddharth Pritam, and Divyansh Pareek. Strong collapse and persistent homology. *Journal of Topology and Analysis*, pages 1–29, 2021.

[10] Coen Boot. Algorithms for determining the clustering coefficient in large graphs. B.S. thesis, Utrecht University, 2016.

[11] Romain Boulet, Etienne Fieux, and Bertrand Jouve. Simplicial simple-homotopy of flag complexes in terms of graphs. *European Journal of Combinatorics*, 31(1):161–176, 2010.

[12] Paul Bruillard, Kathleen Nowak, and Emilie Purvine. Anomaly detection using persistent homology. In *2016 Cybersecurity Symposium (CYBERSEC)*, pages 7–12. IEEE, 2016.

[13] Kate Burleson-Lesser, Flaviano Morone, Maria S Tomassone, and Hernán A Makse. K-core robustness in ecological and financial networks. *Scientific Reports*, 10(1):1–14, 2020.

[14] Chen Cai and Yusu Wang. Understanding the power of persistence pairing via permutation test. *CoRR*, abs/2001.06058, 2020.

[15] Gunnar Carlsson. Topology and data. *Bulletin of the American Mathematical Society*, 46(2):255–308, 2009.

[16] Corrie J Carstens and Kathy J Horadam. Persistent homology of collaboration networks. *Mathematical Problems in Engineering*, 2013, 2013.

[17] Frédéric Chazal and Bertrand Michel. An introduction to topological data analysis: Fundamental and practical aspects for data scientists. *Frontiers in Artificial Intelligence*, 4:667963, 2021.

[18] Yuzhou Chen, Baris Coskunuzer, and Yulia R. Gel. Topological relational learning on graphs. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 27029–27042, 2021.

[19] Yuzhou Chen, Ignacio Segovia-Dominguez, Baris Coskunuzer, and Yulia R. Gel. Tamps2gcnets: Coupling time-aware multipersistence knowledge representation with spatio-supra graph convolutional networks for time-series forecasting. In *the Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

[20] Matija Čufar and Žiga Virk. Fast computation of persistent homology representatives with involuted persistent homology. *arXiv preprint arXiv:2105.03629*, 2021.

[21] Tamal K. Dey, Tao Hou, and Sayan Mandal. Persistent 1-cycles: Definition, computation, and its application. In Rebeca Marfil, Mariletty Calderón, Fernando Díaz del Río, Pedro Real, and Antonio Bandera, editors, *Computational Topology in Image Context - 7th International Workshop, CTIC 2019, Málaga, Spain, January 24-25, 2019, Proceedings*, volume 11382 of *Lecture Notes in Computer Science*, pages 123–136. Springer, 2019.

[22] Tamal Krishna Dey and Yusu Wang. *Computational Topology for Data Analysis*. Cambridge University Press, 2022.

[23] Herbert Edelsbrunner and John Harer. *Computational Topology - an Introduction*. American Mathematical Society, 2010.

[24] Emerson G Escolar and Yasuaki Hiraoka. Optimal cycles for persistent homology via linear programming. In *Optimization in the Real World*, pages 79–96. Springer, 2016.

[25] Christos Giatsidis, Fragkiskos D. Malliaros, Dimitrios M. Thilikos, and Michalis Vazirgiannis. Corecluster: A degeneracy based graph clustering framework. In Carla E. Brodley and Peter Stone, editors, *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada*, pages 44–50. AAAI Press, 2014.

[26] Christos Giatsidis, Dimitrios M. Thilikos, and Michalis Vazirgiannis. Evaluating cooperation in communities with the k-core structure. In *Proceedings of the International Conference on Advances in Social Networks Analysis and Mining, ASONAM 2011, Kaohsiung, Taiwan, 25-27 July 2011*, pages 87–93. IEEE Computer Society, 2011.

[27] Barbara Giunti. Tda applications library, 2022. https://www.zotero.org/groups/2425412/tda-applications/library.

[28] Allen Hatcher. *Algebraic Topology*. Cambridge University Press, 2002.

[29] Christoph D. Hofer, Florian Graf, Bastian Rieck, Marc Niethammer, and Roland Kwitt. Graph filtration learning. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 4314–4323. PMLR, 2020.

[30] Christoph D. Hofer, Roland Kwitt, Marc Niethammer, and Andreas Uhl. Deep learning with topological signatures. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 1634–1644, 2017.

[31] Max Horn, Edward De Brouwer, Michael Moor, Yves Moreau, Bastian Rieck, and Karsten M. Borgwardt. Topological graph neural networks. In *Proceedings of the Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

[32] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[33] Takashi Ichinomiya, Ippei Obayashi, and Yasuaki Hiraoka. Persistent homology analysis of craze formation. *Physical Review E*, 95(1):012504, 2017.

[34] Tian Jiang, Meichen Huang, Ignacio Segovia-Dominguez, Nathaniel K. Newlands, and Yulia R. Gel. Learning space-time crop yield patterns with zigzag persistence-based LSTM: toward more reliable digital agriculture insurance. In *Proceedings of the Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, Virtual Event, February 22 - March 1, 2022*, pages 12538–12544. AAAI Press, 2022.

[35] Matthew Kahle. Topology of random clique complexes. *Discrete mathematics*, 309(6):1658–1671, 2009.

[36] Harish Kannan, Emil Saucan, Indrava Roy, and Areejit Samal. Persistent homology of unweighted complex networks via discrete morse theory. *Scientific Reports*, 9(1):1–18, 2019.

[37] Kristian Kersting, Nils M. Kriege, Christopher Morris, Petra Mutzel, and Marion Neumann. Benchmark data sets for graph kernels, 2016. http://graphkernels.cs.tu-dortmund.de.

[38] Violeta Kovacev-Nikolic, Peter Bubenik, Dragan Nikolić, and Giseon Heo. Using persistent homology and dynamical distances to analyze protein binding. *Statistical Applications in Genetics and Molecular Biology*, 15(1):19–38, 2016.

[39] Gregory Leibon, Scott Pauls, Daniel Rockmore, and Robert Savell. Topological structures in the equities market network. *Proceedings of the National Academy of Sciences*, 105(52):20589–20594, 2008.

[40] Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data, June 2014.

[41] Nicholas O Malott and Philip A Wilsey. Fast computation of persistent homology with data reduction and data partitioning. In *Big Data*, pages 880–889. IEEE, 2019.

[42] Julian J. McAuley and Jure Leskovec. Learning to discover social circles in ego networks. In Peter L. Bartlett, Fernando C. N. Pereira, Christopher J. C. Burges, Léon Bottou, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States*, pages 548–556, 2012.

[43] Konstantin Mischaikow and Vidit Nanda. Morse theory for filtrations and efficient computation of persistent homology. *Discrete & Computational Geometry*, 50(2):330–353, 2013.

[44] Jessica L Nielson, Jesse Paquette, Aiwen W Liu, Cristian F Guandique, C Amy Tovar, Tomoo Inoue, Karen-Amanda Irvine, John C Gensel, Jennifer Kloke, Tanya C Petrossian, et al. Topological data analysis for discovery in preclinical spinal cord injury and traumatic brain injury. *Nature Communications*, 6(1):1–12, 2015.

[45] Giannis Nikolentzos, Polykarpos Meladianos, Stratis Limnios, and Michalis Vazirgiannis. A degeneracy framework for graph similarity. In Jérôme Lang, editor, *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, pages 2595–2601. ijcai.org, 2018.

[46] Ippei Obayashi. Volume-optimal cycle: Tightest representative cycle of a generator in persistent homology. *SIAM Journal on Applied Algebra and Geometry*, 2(4):508–534, 2018.

[47] Dorcas Ofori-Boateng, Ignacio Segovia-Dominguez, Cuneyt Gurcan Akcora, Murat Kantarcioglu, and Yulia R. Gel. Topological anomaly detection in dynamic multilayer blockchain networks. In Nuria Oliver, Fernando Pérez-Cruz, Stefan Kramer, Jesse Read, and José Antonio Lozano, editors, *Machine Learning and Knowledge Discovery in Databases. Research Track - European Conference, ECML PKDD 2021, Bilbao, Spain, September 13-17, 2021, Proceedings, Part I*, volume 12975 of *Lecture Notes in Computer Science*, pages 788–804. Springer, 2021.

[48] Nina Otter, Mason A. Porter, Ulrike Tillmann, Peter Grindrod, and Heather A. Harrington. A roadmap for the computation of persistent homology. *EPJ Data Sci.*, 6(1):17, 2017.

[49] Bastian Rieck, Christian Bock, and Karsten M. Borgwardt. A persistent weisfeiler-lehman procedure for graph classification. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 5448–5458. PMLR, 2019.

[50] Ryan A. Rossi and Nesreen K. Ahmed. The network data repository with interactive graph analytics and visualization. In Blai Bonet and Sven Koenig, editors, *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA*, pages 4292–4293. AAAI Press, 2015.

[51] Ignacio Segovia-Dominguez, Zhiwei Zhen, Rishabh Wagh, Huikyo Lee, and Yulia R. Gel. Tlife-lstm: Forecasting future COVID-19 progression with topological signatures of atmospheric conditions. In Kamal Karlapalem, Hong Cheng, Naren Ramakrishnan, R. K. Agrawal, P. Krishna Reddy, Jaideep Srivastava, and Tanmoy Chakraborty, editors, *Proceedings of the Advances in Knowledge Discovery and Data Mining - 25th Pacific-Asia Conference, PAKDD 2021, Virtual Event, May 11-14, 2021, Proceedings, Part I*, volume 12712 of *Lecture Notes in Computer Science*, pages 201–212. Springer, 2021.

[52] Stephen B Seidman. Network structure and minimum degree. *Social Networks*, 5(3):269–287, 1983.

[53] M. Shanahan, V. P. Bingman, T. Shimizu, M. Wild, and O. Güntürkün. Large-scale network organization in the avian forebrain: a connectivity matrix and theoretical analysis. *Frontiers in Computational Neuroscience*, 7:89, 2013.

[54] Ann Sizemore, Chad Giusti, and Danielle S Bassett. Classification of weighted networks through mesoscale homological features. *Journal of Complex Networks*, 5(2):245–273, 2017.

[55] Zuoyu Yan, Tengfei Ma, Liangcai Gao, Zhi Tang, and Chao Chen. Link prediction with persistent homology: An interactive view. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 11659–11669. PMLR, 2021.

[56] Monisha Yuvaraj, Asim K Dey, Vyacheslav Lyubchich, Yulia R Gel, and H Vincent Poor. Topological clustering of multilayer networks. *Proceedings of the National Academy of Sciences*, 118(21):e2019994118, 2021.

[57] Qi Zhao and Yusu Wang. Learning metrics for persistence-based summaries and applications for graph classification. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 9855–9866, 2019.

[58] Qi Zhao, Ze Ye, Chao Chen, and Yusu Wang. Persistence enhanced graph neural network. In Silvia Chiappa and Roberto Calandra, editors, *The 23rd International Conference on Artificial Intelligence and Statistics, AISTATS 2020, 26-28 August 2020, Online [Palermo, Sicily, Italy]*, volume 108 of *Proceedings of Machine Learning Research*, pages 2896–2906. PMLR, 2020.

[59] Afra Zomorodian and Gunnar Carlsson. Computing persistent homology. *Discrete & Computational Geometry*, 33(2):249–274, 2005.