Visual Detection of Structural Changes in Time-Varying Graphs Using Persistent Homology

Mustafa Hajij*
University of South Florida

Bei Wang†
University of Utah

Carlos Scheidegger‡
University of Arizona

Paul Rosen§
University of South Florida

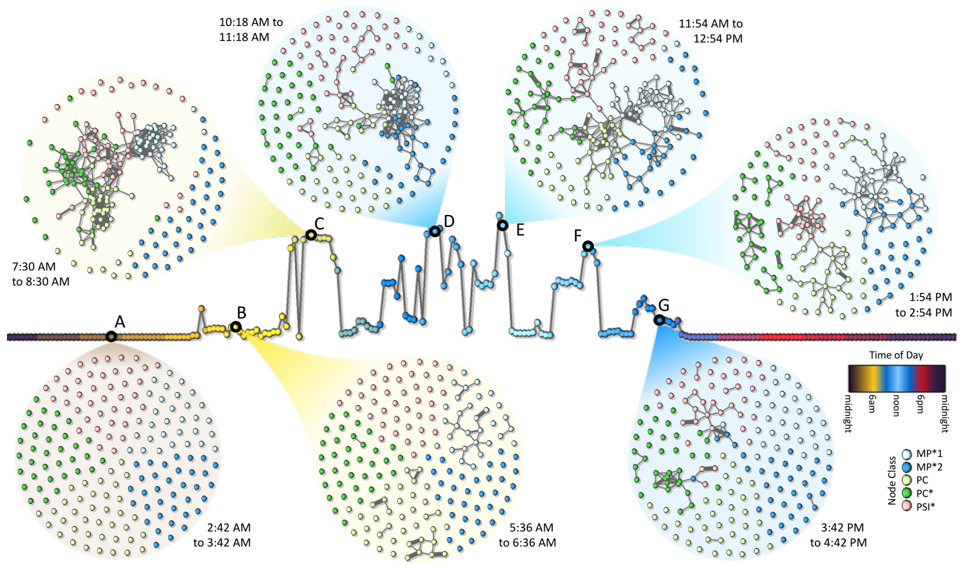


Figure 1: A timeline showing the first Monday of the High School Communication Network dataset. The timeline is generated by comparing the commute-time 0-dimensional homological features of the time-varying network using the bottleneck distance. Here, the 0-dimensional homological features capture cluster-like behaviors in the data at multiple scales. The timeline differentiates periods of highly connected behaviors, such as instances C, D, E, and F, from periods of low or no activity, such as A, B, or G.

Abstract

Topological data analysis is an emerging area in exploratory data analysis and data mining. Its main tool, persistent homology, has become a popular technique to study the structure of complex, high-dimensional data. In this paper, we propose a novel method using persistent homology to quantify structural changes in time-varying graphs. Specifically, we transform each instance of the time-varying graph into a metric space, extract topological features using persistent homology, and compare those features over time. We provide a visualization that assists in time-varying graph exploration and helps to identify patterns of behavior within the data. To validate our approach, we conduct several case studies on real-world datasets and show how our method can find cyclic patterns, deviations from those patterns, and one-time events in time-varying graphs. We also examine whether a persistence-based similarity measure satisfies a set of well-established, desirable properties for graph metrics.

Keywords: Topological data analysis, time-varying graph, persistent homology, graph visualization

1 Introduction

Time-varying graphs are ubiquitous across many disciplines, yet difficult to analyze, making them a natural target for visualization – a good visual representation of a time-varying graph will present its *structure* and *structural changes* quickly and clearly, to enable further analysis and exploration.

A major development in graph drawing has been the observation that using *derived* information can retain structure in static graph visualizations. For example, the dot layout uses *node ranks* to perform hierarchical drawings [31]; the neato algorithm employs *graph distances* within statistical multi-dimensional scaling [30]; and Noack's energy model utilizes *approximated clustering* [46].

In this paper, we take the first steps toward using topological features – captured by persistent homology – with the design goal of *detecting potentially important structural changes in time-varying graph* data. By topological features, we do not mean the configuration of nodes and edges alone, but instead the 0- and 1-dimensional homology groups of a metric space that describe its connected components and tunnels, respectively.

This definition allows us to quantify structural elements within time-varying graphs to identify behavior patterns in the data. Persistent homology quantifies individual topological features (events) in the graph according to their significance (or *persistence*). The set of all features, encoded by the *persistence diagram*, can be seen as a fingerprint for the graph. Using this fingerprint, the most topologically important structures of two graphs can be compared in a manner that is robust to small perturbations in the data.

Well-understood techniques in topological data analysis typically focus on the qualitative study of point cloud data under the metric space setting. In order to study graph data, our approach is to embed the graph in a metric space, where topological techniques can be applied. In other words, the notion of metric space acts as an organizational principle [9] in interpreting the graph data.

Our approach, as seen Figure 2, can be summarized as follows. The input of our pipeline is a time-varying graph, which is an ordered sequence of graph *instances*. First, each instance is embedded into

*e-mail: mhajij@usf.edu
†e-mail: beiwang@sci.utah.edu
‡e-mail: cscheid@email.arizona.edu
§e-mail: prosen@usf.edu

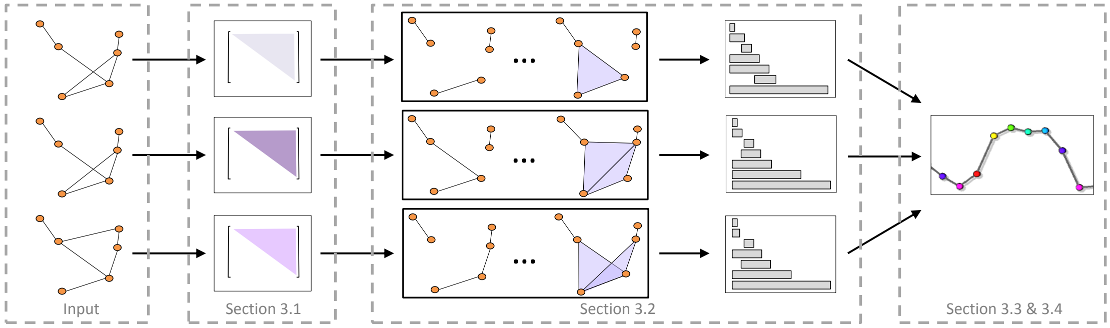


Figure 2: The pipeline of our approach. An ordered sequence of graphs representing a time-varying graph is given as an input. Each graph instance is individually embedded into a metric space (Section 3.1). The topological features of each (metric-space-embedded) graph instance are extracted by computing persistent homology of its corresponding Rips filtration; the topological features are encoded by persistence diagrams and visualized as barcodes (Section 3.2). Finally, persistence diagrams are compared and the structural changes among the graph instances are visualized (Sections 3.3 and 3.4).

a metric space. Second, topological features of each instance are extracted, using persistent homology, and encoded within persistence diagrams. Third, instances are compared by calculating the distance between persistence diagrams and projecting them using classical multi-dimensional scaling (MDS) [6].

The data is then visualized using an interactive timeline and node-link diagrams, as shown in Figure 1. The horizontal axis is used to represent time, while the vertical location is the first component of MDS; in other words, it captures the dissimilarities among instances. Graph instances from selected timeframes are drawn using a force-directed layout to demonstrate how the approach highlights different structures in the graph. The contributions of our paper are:

- A novel pipeline for detecting structural changes in time-varying graphs that uses persistent homology to summarize important structures, as opposed to directly comparing nodes and edges.
- An interface that uses conventional visualization approaches adapted to the design goal of highlighting structural changes.
- Two case studies of time-varying graphs showing how our approach can find cyclic patterns, deviations from those patterns, and unique one-time events in the graphs.
- A study of the suitability of using persistence-based similarity measure for detecting structural changes in time-varying graphs.

2 Related Work

Static Graph Analysis and Visualization. We provide a brief overview here, see [53] for a survey.

The first automated technique for node-link diagrams is Tutte's barycentric coordinate embedding [49], followed by linear programming [31], force-directed/mass-spring embeddings [28, 36], embeddings of the graph metric [30], and techniques using linear-algebraic properties of the connectivity structures (especially, the graph Laplacian and its associated eigenspace) [39, 40].

Most graph visualization systems, including Gephi [3], NodeXL [34], and Graphviz [25], use variations on node-link visualizations to display graphs. For dense graphs, edge bundling can reduce visual clutter by routing graph edges to the same portion of the screen [35]. In terms of quality, divided edge bundling [48] produces high-quality results, while hierarchical edge bundling [29] scales to millions of edges with slightly lower quality. Because these quality and runtime trade-offs are so characteristic of node-link diagram visualizations, whether or not this class of diagrams can effectively unlock the insights hidden inside the structure of large networks remains an open research question.

Other visual metaphors have been proposed to reduce clutter, ranging from relatively conservative proposals [20, 21] to variants of matrix diagrams [18] and abstract displays of graph statistics [38].

Time-Varying Graph Analysis. The problem we address is closely related to the problem of measuring similarity or dissimilarly between graphs without knowing node correspondences. Comparing between graphs up to isomorphism is hard [1]. For this reason, many notions of graph similarities have been proposed [4, 47]. These methods rely on mapping the graphs into a feature space and then defining distances on that space. Other approaches use kernel functions to build a similarity measures on graphs [43, 51]. While large portions of the literature on graph similarity focus on graph comparison with known node correspondences, there are attempts to tackle the problem where node correspondence is unknown [51, 52]. Distance functions on the space of graphs have also been studied [12].

Time-Varying Graph Visualization. Beck et al. [5] provide a detailed survey of dynamic graph visualization. They divide the techniques into two major categories, animation and timelines. Our approach falls into the latter category. Animation approaches, such as the work of Misue et al. [44], vary the graph representation over time, while making the graph as legible as possible at any given instance. Timeline approaches, such as the work of Greilich et al. [33], use a non-animated, often spatially oriented, visual channel to show the changes in the graph over time. Timeline approaches seem to provide a better overview of the data as it tries to capture the entire graph sequence in a single image. These approaches include multiple techniques such as node-link-based methods [37], matrix-based approaches [8] and feature vector-based method [50]. For more references see also [53].

Topological Data Analysis of Networks. Persistent homology is becoming an emerging tool in studying complex networks [19, 22] including collaboration [2, 10] and brain networks [11, 15]. To the best of our knowledge, our approach is the first to connect topological techniques with the visualization design of (time-varying) graphs.

3 Approach

Our approach uses persistent homology to identify and compare features in a time-varying graph. Our visual design goal is to identify high-level structural changes in the graph. To do this, consider a time-varying graph $\mathscr{G} = \{G_0, \ldots, G_n\}$ that contains an ordered sequence of static graph *instances* $G_i = (V_i, E_i)$.

We are interested in quantifying and visualizing structural changes of $\mathscr{G}$. Our analysis pipeline (see Figure 2) is described below, and we provide a detailed description of each step in the subsequent sections.

1. Associate each instance G_i with a metric space representation. This yields a symmetric distance matrix d_i, where $d_i(x,y)$ measures the (shortest-path or commute-time) distance between vertices x and y in G_i (Section 3.1).
2. Extract topological features of G_i by constructing a filtration F_i from its distance matrix d_i and computing its corresponding p-dimensional persistence diagrams $PD_p(F_i)$ for $p \in \{0,1\}$ (Section 3.2).
3. Capture the structural differences between G_i and G_j by computing the bottleneck or Wasserstein distance between their corresponding persistence diagrams $PD_p(F_i)$ and $PD_p(F_j)$ (Section 3.3).
4. Visualize the structural differences among the instances of $\mathscr{G}$ (Section 3.4).

3.1 Graphs and Metric Space Representations

Suppose an instance G_i is represented as a weighted, undirected graph with a vertex set V and an edge set E equipped with a positive edge weight w. We associate each graph instance G_i with a metric space representation, which yields a symmetric distance matrix d_i.

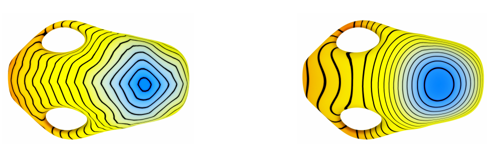
Consider the positive edge weight as the *length* of an edge. Then a natural metric d_{sp} is obtained on G_i, where for every pair of vertices x and y in G_i, the distance $d_{sp}(x,y)$ is the length of the shortest path between them. This is the classic *shortest-path distance*, which is typically computed with Dijkstra's algorithm [17] and its variations.

Alternatively, other distance metrics based on the graph Laplacian [14], such as commute-time distance, discrete biharmonic distance, and diffusion distance, can be considered. For instance, the *commute-time distance* is defined as [27]

$$d_{ct}^2(x,y) = \sum_{i=1}^{|V|-1} \frac{1}{\lambda_i} (\phi_i(x) - \phi_i(y))^2. \tag{1}$$

Here $\{\lambda_i\}_{i=0}^{|V|-1}$ and $\{\phi\}_{i=0}^{|V|-1}$ are the generalized eigenvalues and eigenvectors of the graph Laplacian of G_i, respectively [13]. In practice, we approximate the summations of Equation (1) by considering the first few nonzero eigenvectors, since the higher eigenvectors do not contribute significantly.

These distance metrics are illustrated in Figure 3, which shows the distance from a point source to all other locations on the surface. We see the commute-time distance produces a smoother gradient than the shortest-path distance.



(a) Shortest-path Distance (b) Commute-time Distance

Figure 3: The (a) shortest-path and (b) commute-time distance measured from a source point on a 2-dimensional surface embedded in $\mathbb{R}^3$. Blue indicates the regions closest to the source.

3.2 Extracting Topological Features

To extract topological features from each graph instance, we apply persistent homology to its metric space representation. To describe our process, we first briefly review persistent homology. We then describe persistence diagrams, which encode topological features of a given graph instance. For more background on persistence homology, see [23] and the references within.

Topological features. Homology deals with topological features of a space. Given a topological space $\mathbb{X}$, the 0-, 1- and 2-dimensional homology groups, denoted respectively as $\mathsf{H}_0(\mathbb{X})$, $\mathsf{H}_1(\mathbb{X})$ and $\mathsf{H}_2(\mathbb{X})$, correspond intuitively to (connected) components, tunnels and voids of $\mathbb{X}$.

In our context, we care about the 0- and 1-dimensional topological features of a graph instance G_i that, roughly speaking, correspond to *(connected) components* and *tunnels* formed by points in its metric space representation.

Persistent homology. In practice, there might not exist a unique scale that captures topological structures of the data. Instead, we adapt a multi-scale notion of homology, called *persistent homology*, a main tool in topological data analysis, to describe the topological features of a space at different spatial resolutions.

Persistent homology typically starts with a finite set of points in a metric space. In our setting, each graph instance G_i is associated with a metric space, where vertices in G_i form a finite set of points S, and d_i encodes the pairwise distance among points in S.

We then apply a geometric construction, such as a Rips complex, on the point set S, that describes the combinatorial structure among the points. For a real number $r > 0$, a *Rips complex*, denoted as $R(r)$, is formed by considering a set of balls of diameter r centered at points in S. A 1-simplex (an edge) is formed between two points in S if and only if their balls intersect (see Figure 4 left). A 2-simplex (a triangular face) is formed among three points if the balls intersect between every pair of points (see Figure 4 right).

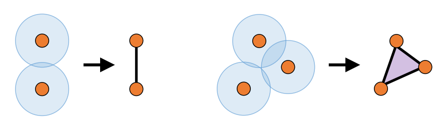


Figure 4: Edges (left) and triangles (right) in a Rips complex.

Given a finite point set S from G_i, continuously increasing the diameter forms a 1-parameter family of nested unions of balls; and correspondingly we obtain a 1-parameter family of nested Rips complexes, referred to as a *Rips filtration*. Let $0 = r_0 \leq r_1 \leq r_2 \leq \cdots \leq r_m$ denote a finite sequence of increasing diameter. The Rips filtration F_i (of G_i) is a sequence of Rips complexes connected by inclusions, $R(r_0) \to R(r_1) \to R(r_2) \to \cdots \to R(r_m)$. Figure 5 shows a Rips filtration defined on an example graph equipped with a shortest-path metric. Applying homology to a Rips filtration, the homology groups are connected from left to right by homomorphisms induced by inclusions, $\mathsf{H}(R(r_0)) \to \mathsf{H}(R(r_1)) \to \mathsf{H}(R(r_2)) \to \cdots \to \mathsf{H}(R(r_m))$.

Topological features appear and disappear as the diameter increases: when a topological feature appears, that is, a component (i.e. a cluster) or a tunnel forms, this is called a *birth* event; when a topological feature disappears, that is, two components merge into one or a tunnel is filled, it is called a *death* event. Each topological feature is represented by a single bar, with the position of left and right sides representing the birth and death times, respectively. The *persistence* of a topological feature is the time difference between the death and the birth event. For example, at $r = 0.25$, there are six components alive, one per vertex, all of which are born at $r = 0$. At $r = 0.5$, the two components in the upper right combine into one. This causes the death of one component, represented by a barcode of length 0.5 on the right.

Persistence Diagrams. Topological features of a graph instance and their persistence are recorded by pairing their birth and the death events as a multi-set of points in the plane, called the *persistence diagram* (see [24]).

Each topological features is represented as a point (u,v), where u is the *birth time*, and v is the *death time* of the feature. Certain

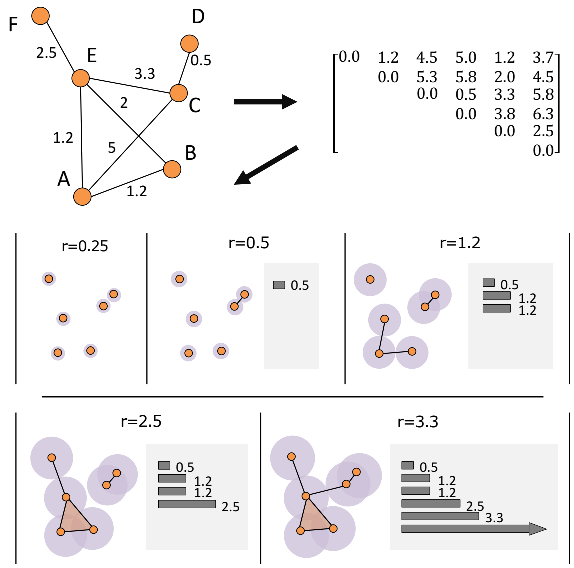


Figure 5: Constructing a Rips filtration from a distance matrix on a graph. The numbers above each Rips complex indicate the diameter at which the complex is computed. The corresponding 0-persistence diagrams are shown in the gray box to the right of each complex.

features may "live" forever; in that case, they are assigned a death time of ∞. Therefore, a persistence diagram contains a multi-set of points in the extended plane (i.e., $(\mathbb{R} \cup \pm\infty)^2$). For technical reasons, we add the points on the diagonal to the diagram, each with infinite multiplicity. The *persistence* of the pair (u, v) is simply $|v - u|$. Features with higher persistence carry more significant topological information. Features with low persistence are typically considered noise. A persistence diagram can be visualized as persistence barcodes [32] (see Figures 2 and 5), where each *bar* starts at time u and ends at time v. We are interested in 0- and 1-dimensional topological features, so we consider the 0- and 1- persistence diagrams, denoted as $PD_0(F_i)$ and $PD_1(F_i)$, respectively.

3.3 Comparing Sets of Topological Features

A persistence diagram can be thought of as a summary of topological features of a graph instance G_i. To quantify the structural difference between two instances G_i and G_j, we compute the bottleneck and Wasserstein distances between their persistence diagrams.

Given two persistence diagrams X and Y, let η be a bijection between points in the diagram. The bottleneck distance [24] is defined as

$$W_\infty(X,Y) = \inf_{\eta : X \to Y} \sup_{x \in X} \|x - \eta(x)\|_\infty . \qquad (2)$$

The Wasserstein distance is

$$W_q(X,Y) = \left[\inf_{\eta : X \to Y} \Sigma_{x \in X} \|x - \eta(x)\|_\infty^q \right]^{1/q}, \qquad (3)$$

for any positive real number q; in our setting, $q = 2$.

The set of points in the persistence diagram can be considered as a *feature vector*, where the *feature space* consists of all persistence diagrams for the time-varying graph G. Given all pairwise distances between persistence diagrams, classical MDS is then used to reduce the dimensionality of the feature vectors for visualization, and to identify the instances where topologically interesting events occur.

3.4 Visualization

The design goal of our interactive visualization tool is to provide insights about variation in the structural properties of time-varying graphs. In this way, we hope to identify time periods of uniform behavior (low variation) and outlier behavior (instances of high variation). Our visualization tool provides a number of capabilities to support this form of investigation.

Timeline. The timeline view uses the horizontal axis to represent time and the vertical axis to represent the first dimension returned by applying classical MDS to the space of persistence diagrams. This in essence highlights the dissimilarity between graph instances. Each point on the timeline represents a single instance of the time-varying graph. The points are colored using cyclic colormaps, such as the time-of-day colormap of Figure 1 or the day-of-the-week colormap of Figure 11.

Cyclic Patterns. Two techniques are available for showing repetitive patterns in the data, both being variations of the timeline. The first technique simply splits the data based upon a user-specified period length. Each period is colored uniquely. Figure 7 shows an example of this. For the second technique, the time periods are clustered based upon their ℓ^2-norm using k-means clustering with a user-specified k, see Figure 12 for an example where the points are colored by day of the week.

Graph Visualization. For investigating the behavior of specific graph instances, the instances are displayed by two visualization mechanisms. The first is a node-link diagram created using a force-directed layout. If categorical information is available (such as

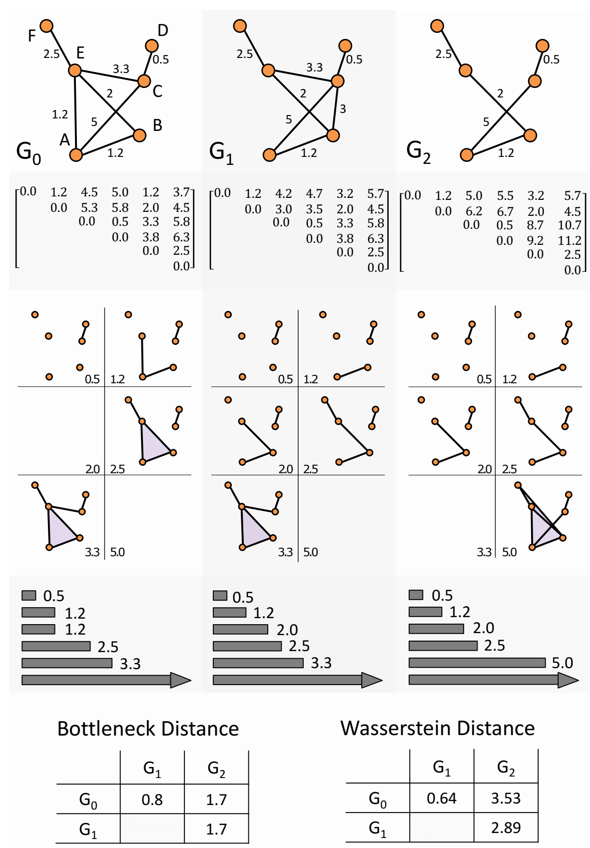


Figure 6: From left to right, 1st row: three weighted graph instances G_0, G_1 and G_2 representing a time-varying graph. 2nd row: each graph instance is embedded into a metric space, represented by a shortest-path distance matrix. 3rd row shows the filtrations in which topologically significant events occur, resulting in persistence barcodes in the 4th row. 5th row: the persistence diagrams are compared pairwise using bottleneck and Wasserstein distance.

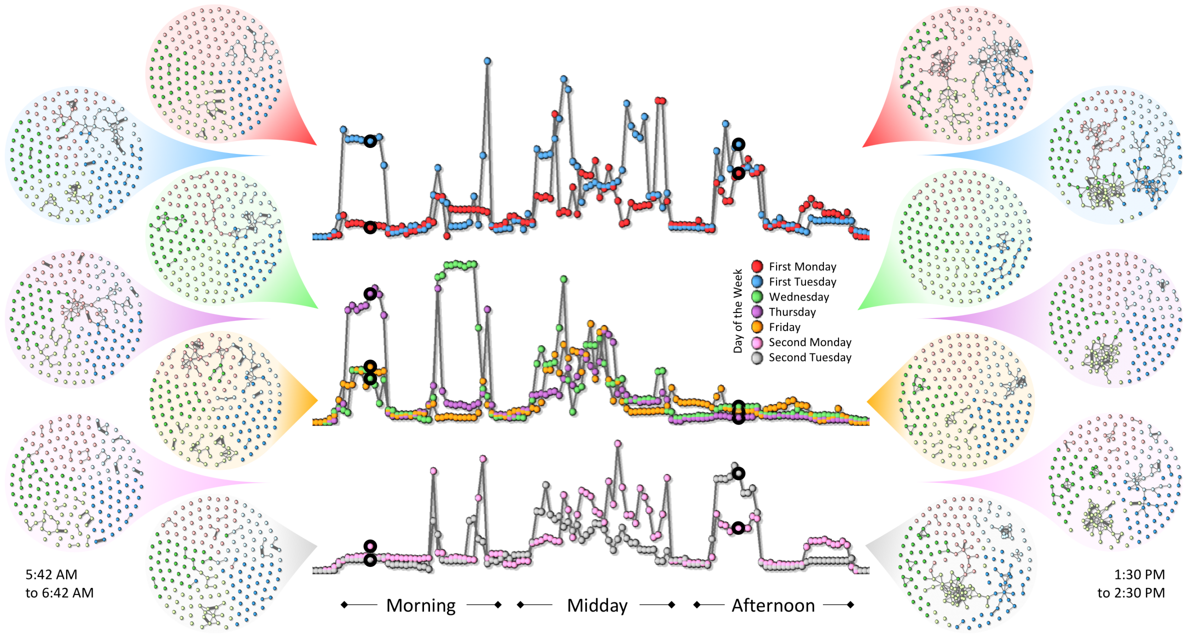


Figure 7: Timeline comparison for the seven weekdays of the High School Communications dataset together with graph instances along these timelines. These graph instances validate the different levels of communication visible using our approach.

in Figure 1), the nodes are colored by those categories. For 1-dimensional topological features, nodes can be parameterized around the tunnel using a 1-dimensional cyclic parameterization [16, 54]. An example of this is seen in Figure 9. In other cases, nodes receive a fixed color. The second mechanism visualizes the persistence diagram for a given graph instance using its barcodes (see fourth row of Figure 6). The barcode is a variation on a bar chart that represents the birth and death of all topological features in the graph.

3.5 Example

We provide an illustrative example of our pipeline in Figure 6.

In step 1 (1st row), a time-varying graph G is given as a sequence of graph instances, where each instance is a connected, weighted graph. In step 2 (2nd row), each graph instance is embedded in a metric space by calculating a distance matrix using the shortest-path metric. In step 3 (3rd row), each distance matrix is used to compute a series of filtrations. In reality, additional filtrations are created, but we show only those that produce 0-dimensional features. In step 4 (4th row), the 0-dimensional persistence diagrams of the filtrations are extracted and shown as barcodes. The final step (5th row) consists of computing the distances between these diagrams using bottleneck and Wasserstein distances.

The bottleneck or Wasserstein distance as a *persistence-based similarity measure* helps quantify topologically similarity between a pair of instances. For example, under both distances, G_0 and G_1 are much closer to one another than are the pairs (G_0, G_2) and (G_1, G_2).

4 Case Studies

To validate our approach, we look at case studies of two publicly available datasets. Both are communication networks, one involves interpersonal communication of high school students; and the other contains e-mail communications between researchers. These case studies help demonstrate how our approach can identify cyclic patterns in data, deviations from patterns, and one-time events in time-varying graphs.

Our pipeline requires a number of tools for processing. Graph processing and metric space embedding are coded using Python. Persistent homology calculations and the bottleneck and Wasserstein distances are computed using Dionysus[1]. Finally, visualizations are implemented using Processing[2].

4.1 High School Communications

The High School Communications dataset [26] is a time-varying graph that tracks the contact between high school students. The data was collected for 180 students in five classes over seven school days in November 2012 in Marseilles, France. The graph tracks Monday through Friday of the first week and Monday and Tuesday of the following week.

We compute both shortest-path and commute-time distances and both 0- and 1-dimensional persistence diagrams. Then, both the bottleneck and Wasserstein distances are used to compare persistence diagrams. We present a small set of configurations and draw a few conclusions from them. Many similar conclusions have been identified in other configurations that are not shown.

4.1.1 An Average Day

First, to examine an average day of communication, we look at the 0-dimensional features of the first Monday of the dataset in Figure 1. Commute-time is used to generate persistence diagrams and bottleneck distance is used to compare diagrams. In this figure, a number of phases can be seen. In the early and late hours, no interactions occur (e.g., time A). As the school day begins at time B, light, loosely connected communications begin. By mid-morning (time C), class MP*1, PC, PC*, and PSI* are all interacting heavily internally and externally. Midday (times D & E), shows classes heavily interacting once again. Early afternoon (time F) shows mostly internal communications for classes PC, PC*, and PSI* and internal and external communications for MP*1 and MP*2. Finally, the end of the day, time G, shows much sparser group communications.

4.1.2 Comparison with Other Days

While observing patterns within a single day is interesting, comparing Monday with other days can help to better identify regular and

[1] http://www.mrzv.org/software/dionysus/
[2] https://processing.org/

irregular daily behavior. Figure 7 shows just such a comparison; it uses commute-time to generate 0-dimensional persistence diagrams, and Wasserstein distance to compare diagrams.

The top chart of Figure 7 compares the first Monday and the first Tuesday. Ignoring outlier graph instances, two main differences can be observed. First, the early morning of Tuesday shows different levels of activity than Monday. This can be confirmed by looking at examples from those days. Figure 7 (top left) shows example graphs from Monday and Tuesday morning. Second, at the beginning and the end of midday, Tuesday shows higher activity than Monday.

The middle chart of Figure 7 compares Wednesday, Thursday, and Friday. Wednesday and Friday show more early morning activity than Monday, but Thursday shows activity levels similar to Tuesday. Individual graph instances of the time-varying graph from this timeframe can be seen in Figure 7 (middle left). Late morning shows that Wednesday is extremely active, while Thursday and Friday are mostly inactive. Midday (midrange active) and the afternoons (inactive) across all three days remains similar. Sample graphs for this timeframe are shown in Figure 7 (middle right).

The bottom of Figure 7 shows the second Monday and Tuesday. These days show almost no morning activity (also see Figure 7 (bottom left)) and normal midday activity. Early afternoon shows midrange and high activity for Monday and Tuesday, respectively. Graphs associated with these activity levels can be seen in Figure 7 (bottom right).

As a means to compare results to a more traditional analytic, Figure 8 bottom is a timeline that captures the number of interaction events for a given graph instance in the time-varying graph (i.e., the sum of the weights). Comparing this chart to our approach in Figure 8 top, it is clear that our approach captures a *different* type of behavior than edge counting alone.

4.1.3 1-Dimensional Topological Features

The High School Communications dataset ultimately contains very few 1-dimensional topological features, the majority of which have low persistence. The one-time exception, which appears on the first Monday, can be seen in Figure 9. Between 11:48 am and 12:48 pm, a high-persistence 1-dimensional pattern appears in the graph. The nodes of the graph are parameterized using that feature and visualized using a cyclic rainbow colormap. The graph shows a large tunnel (loop) toward the upper left.

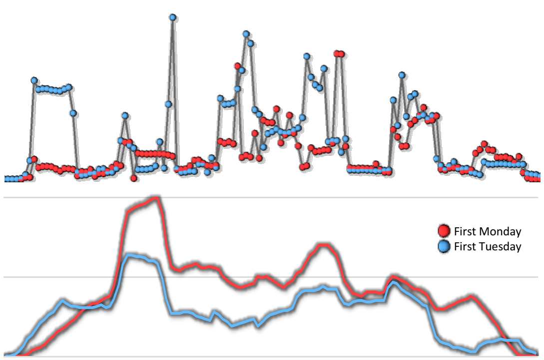


Figure 8: Top: Persistent homology timeline for the first Monday and Tuesday of the High School Communications dataset. Bottom: Timeline counting the number of events (sum of all weights) in each graph instance. The timeline shows how different features can be identified in our approach as compared to edge counts alone.

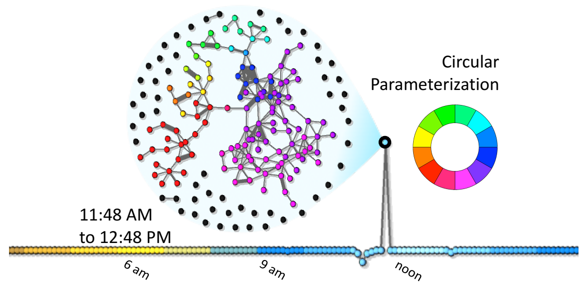


Figure 9: Timeline of the High School Communications dataset for 1-dimensional features. The timeline was generated by comparing the commute-time features using bottleneck distance. The single outlier is a graph with a high persistence cycle. To highlight that feature, the graph is parameterized and visualized with a cyclic rainbow colormap [54].

4.2 EU Research Institution E-Mail

The EU Research Institution E-Mail [42][3] dataset is an anonymized time-varying graph tracking e-mails between members of *a large European research institution*. We have used the smaller of the available networks containing 986 nodes and 332,334 temporal edges. The graph tracks the activity for 803 days. A period of about 200 days is missing toward the end of the dataset, so we have analyzed the first 500 days. A single graph instance is created per day and shared 45% overlap with neighboring days. Once again, edge weight is chosen by counting the number of communications between vertices in a graph instance.

4.2.1 Bottleneck vs. Wasserstein Distance

The bottleneck and Wasserstein distances both capture important but distinct differences among sets of topological features. Intuitively, the bottleneck distance ($p = \infty$) captures the most perturbed topological feature (or the extreme behavior); while the Wasserstein distance ($p = 2$) captures the perturbation across all features (or the average behavior). Figure 10 shows how this difference impacts the analysis of the EU E-Mail dataset. For 0-dimensional (Figure 10(a)) and 1-dimensional (Figure 10(b)) bottleneck distances, the result is noisy, as the value captured has the most variation. For 0-dimensional (Figure 10(c)) and 1-dimensional (Figure 10(d)) Wasserstein distances,

[3] http://snap.stanford.edu/data/email-Eu-core.html

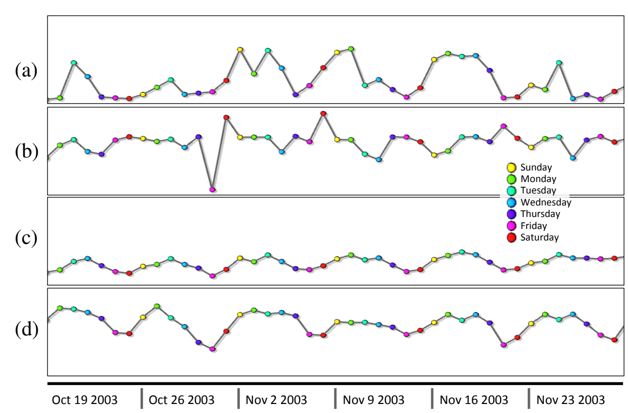


Figure 10: Comparing shortest-path bottleneck ((a) and (b)) and Wasserstein ((c) and (d)) distance on 0-dimensional ((a) and (c)) and 1-dimensional ((b) and (d)) features in the EU E-Mail dataset. Since bottleneck distance captures the most perturbed feature, the result may be noisy. Wasserstein distance captures variation across all features in the graph, resulting in a smoother pattern.

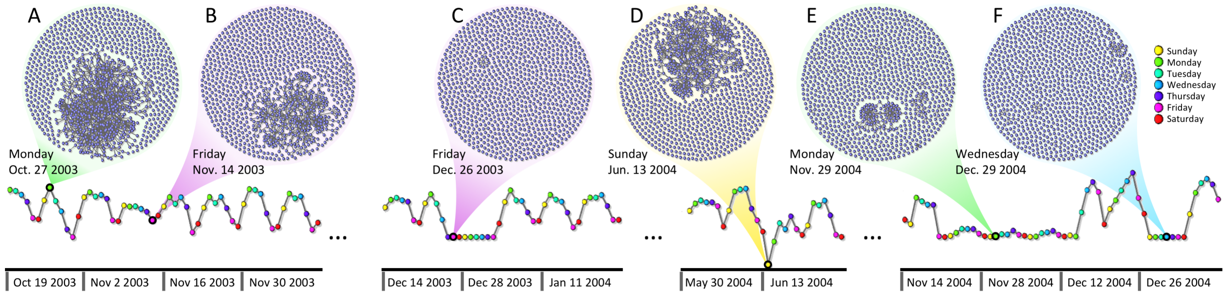


Figure 11: Highlights from the EU E-Mail dataset using the Wasserstein distance on 1-dimensional persistence diagrams based on shortest-path metric. A & B show graphs from a timeframe of normal weekly cyclic activity. C & F show timeframes of limited activity from December of 2003 and 2004 during the Christmas and New Years holidays. D shows an unexpected boost in activity on June 13, 2004 that is correlated with the release of results for the EU Parliamentary Election. E shows a 3- to 4-week period of low activity in November and December of 2004. We could not identify any externally correlated event to explain this occurrence.

the result is smoother, since it encodes the perturbations across all features. For our analysis of the EU E-Mail data, this property is more desirable.

4.2.2 Revealing Cyclic Patterns

Upon investigating the data, cyclic patterns were immediately apparent with all configurations of the Wasserstein distance (0- & 1-dimensional features and shortest-path & commute-time). Figure 11 A & B show the 1-dimensional shortest-path version, where the cyclic patterns are most prominent (also see supplemental material for the complete 1-dimensional feature timeline).

It is notable that this pattern is related to the natural cycle of the week. To identify the pattern of the "standard" week, we divided the data into seven-day segments and used k-means clustering to group similar weeks. Figure 12 shows the result with five clusters. Each cluster shows a version of the typical week for this institution.

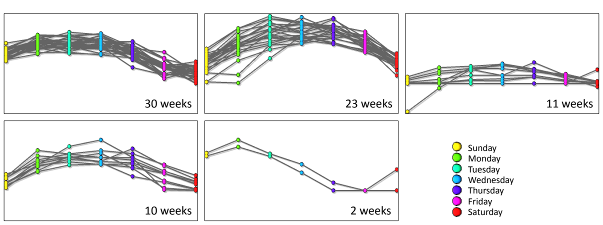


Figure 12: Clustering of the weekly behavior in the EU E-Mail dataset using the Wasserstein distance on 1-dimensional features based on the shortest-path metric. The clusters shows four primary patterns and one outlier pattern (bottom). The number of weeks in each cluster is listed in the lower right.

4.2.3 One-time Events

When looking at the entire timeline (see supplemental material), a number of one-time events are easily discovered. Figure 11 C & F are two such events. During these time periods, very little activity is present in the graphs. These times happen to be the last week of December and first few days of January, during the Christmas and New Year's holidays. Figure 11 D is a one-time event that shows an extreme increase in activity for a 1- to 2-day period. After entering the date, June 13, 2004, into Google, we discovered that this day corresponds to the release day of the results for the EU Parliamentary Election. Finally, Figure 11 E shows a 3- to 4-week period of significantly decreased activity. Despite our best efforts, we could not identify a major external event that would have caused such a reduction, and since the data is anonymized, we could not identify the institution to investigate a local or internal cause.

5 Discussion

In the previous section, we construct a similarity measure between two graph instances of a time-varying graph by utilizing the bottleneck or Wasserstein distance between their persistence diagrams, which encode the topological features associated with each instance.

However, one might ask: why persistent homology? We argue that using topological data analysis and in particular, persistent homology, to study graphs, has complementary benefits and offers new insights. In this section, we conduct several experiments to justify our approach. In addition, we describe some intuition behind the information encoded by the persistence diagram of a graph, and the distance metric defined on them.

5.1 Persistent Diagram As a Graph Fingerprint

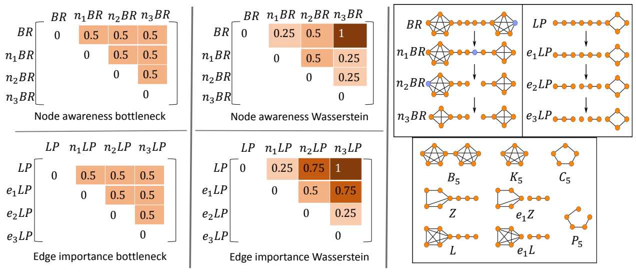
Conventional graph-theoretical approaches typically utilize the statistical properties of the vertices and edges, for instance, degrees, connectivity and path lengths, to describe the short range and pairwise interactions in the system. On the other hand, topological summaries, such as the persistence diagrams, are compressed feature representation of the underlying data, that can capture long-range and higher order interactions.

We test our persistence-based similarity measure against a set of desirable properties for a similarity measure on a graph (the first four conditions are introduced in [41]):

1. **Edge importance:** An edge whose insertion or deletion changes the number of connected components is more important than an edge that does not.
2. **Weight awareness:** In weighted graphs, the bigger the weight of the removed edge is, the greater the impact on the similarity measure should be.
3. **Edge submodularity:** Changing an edge in a dense graph is less important than changing an edge in an equally sized, sparse graph.
4. **Focus awareness:** Random changes in graphs are less important than targeted changes of the same extent.
5. **Node awareness:** We add an extra condition in this paper, i.e., deleting a large number of nodes in a graph has a larger impact than deleting a small number of nodes from the same graph.

We conduct several experiments on synthetic and real-world datasets to test the above conditions.

For the node awareness (property 5), we consider the graphs BR shown in Figure 13 (c) top left. Each graph n_iBR is obtained from the original graph BR by deleting i number of nodes (in blue). The bottleneck and Wasserstein distance matrices of PD_0 between these graphs are shown in the top of Figure 13 (a)-(b). The PD_1



(a) Bottleneck dim-0 (b) Wasserstein dim-0 (c) Small graphs

Figure 13: Given synthetic, small exemplar graphs in (c), we study the node awareness (property 5, a-b, top) and the edge importance (property 1, a-b, bottom) on these graphs by computing the bottleneck (a) and Wasserstein distances (b) matrix between PD_0 of the corresponding graphs. All edge weights are assumed to be 1.

distance matrices are omitted since their entries are all zeros. From the matrices in Figure 13 (a)-(b) top, we observe that the persistence-based similarity measure is sensitive to node deletion, that is, it satisfies *node awareness*, in particular, the Wasserstein distance, is more node aware than the bottleneck distance in these examples.

Similarly, to test edge importance (property 1) against our similarity measure, we delete a set of edges from a graph LP, shown in Figure 13 (c) top right. The graph e_iLP is obtained from LP by deleting i edges (in blue). The bottleneck and Wasserstein distance matrices of PD_0 among these graphs are shown in Figure 13 (a)-(b) bottom. We observe that our persistence-based similarity measure is sensitive to edge deletions that change the connectivity of the graph, that is, it satisfies *edge importance*. Notice how the Wasserstein distance is more aware of the level of (dis)connectedness between the graphs than the bottleneck distance.

To test weight awareness (property 2), we run our test on three randomly generated, weighted graphs $A_1 = (V_1, E_1, w_1)$, $A_2 = (V_2, E_2, w_2)$ and $A_3 = (V_3, E_3, w_3)$, where $|V_1| = 50$, $|V_2| = 60$, $|V_3| = 70$, $|E_1| = 200$, $|E_2| = 250$ and $|E_3| = 300$ respectively. Each is generated from the $G_{n,m}$ random graph model, where a graph is chosen uniformly at random from the set of all graphs with n nodes and m edges (by setting $n = |V_i|$ and $m = |E_i|$ for $1 \leq i \leq 3$). The weights on the edges are drawn uniformly from $(0.1, 1)$. For each graph $A_i = (V_i, E_i, w_i)$, we obtain a set of $|E_i|$ modified graphs $B_i^e = (V_i, E_i, u_i)$ by modifying only the weight of an edge e (for all edges) in A_i such that $u_i(e) = w_i(e) + \delta$, where δ is drawn uniformly randomly from $(4,5)$; similarly, we obtain a set of modified graphs $C_i^e = (V_i, E_i, v_i)$ from A_i by modifying only the weight of edge e such that $v_i(e) = w_i(e) + \delta'$, where δ' is drawn uniformly randomly from $(2,3)$. Let graph eA_i denote the graph obtained from A_i by deleting an edge e. Property 2 holds when $W(eA_i, B_i^e) - W(eA_i, C_i^e) \geq 0$ for all e in A_i.

In Figure 14, we represent the difference $W(eA_i, B_i^e) - W(eA_i, C_i^e)$ by plotting the points $(W(eA_i, C_i^e), W(eA_i, B_i^e))$. Hence, property 2 holds for a point (x, y) on and above the diagonal (i.e. $y \geq x$). Note that our similarity measure satisfies *weight awareness* for dimension 0 but violates the condition for dimension 1. This is because $W_{q,1}(eA_i, B_i^e) - W_{q,1}(eA_i, C_i^e)$ for some e captures the creation or the destruction of a cycle.

To test edge submodularity (property 3), we consider a set of four graphs A, B, C and D. These graphs share the same number of nodes. Graph A is denser than graph C; while graph B and D are obtained from A and C, respectively, by deleting an edge. We test property 3 against four sets of small synthetic graphs in Figure 13 (c) bottom; the results are shown in Table 1. We see that both Wasserstein and bottleneck on PD_0 better capture the changes that occur in a sparser graph than they do on an equally sized denser graph; i.e., they satisfy *edge submodularity* in dimension 0. However, these distances behave differently on PD_1. Table 1 shows some negative entries; this is because between C and D, a cycle is either created or destroyed; while no cycle appears/disappears between A and B (that is, $W(A,B) = 0$).

For focus awareness (property 4), we generate three random weighted graphs A_1, A_2 and A_3 following the same $G_{n,m}$ model as before, with 35, 100, and 120 vertices, and 70, 500, and 300 edges, respectively; and all edge weights are chosen uniformly random from $(0.1, 1)$. We generate a collection of so-called *corrupted* (i.e., modified) graphs from the original graph with two types of corruptions: (1) by deleting 10% to 70% of random edges (with 10% increment) of the original graph; and (2) by deleting the same number of edges from the original graph in a targeted way, specifically, among the edges with the largest weights. For each graph A_i we plot the difference between the targeted corruption $T_k(A_i)$ and the random corruption $R_k(A_i)$, for some percentage k: $\Delta(W_{q,j}) := \{W_{q,j}(A_i, T_k(A_i)) - W_{q,j}(A_i, R_k(A_i))\}_{k=10}^{70}$ against the percentage of deleted edges i.

We obtain similar observations shown in Figure 15 as the property

Graphs				$W_{2,0}$	$W_{2,1}$	$W_{\infty,0}$	$W_{\infty,1}$
A	B	C	D	$\Delta(W) = W(A,B) - W(C,D)$			
K_5	e_1K_5	C_5	e_1C_5	0	0.25	0	0.5
C_5	e_1C_5	P_5	e_1P_5	0.25	-0.25	0.5	-0.5
K_9	e_1K_9	C_9	e_1C_9	0	1	0	1
C_9	e_1C_9	P_9	e_1P_9	0.25	-1	0.5	-1
L	e_1L	Z	e_1Z	0.25	0	0.5	0

Table 1: Testing edge submodularity (property 3) using the graphs from Figure 13 (c).

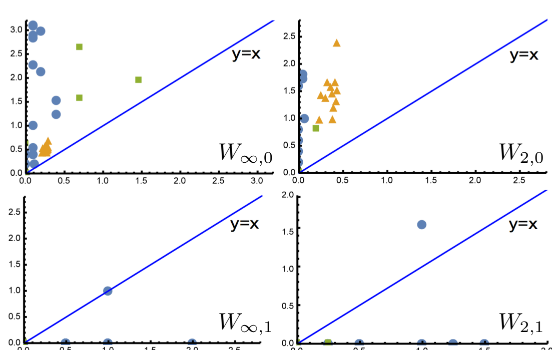


Figure 14: Testing weight awareness (property 2). Points $(W(eA, C^e), W(eA, B^e))$ on and above the diagonal correspond to instances where property 2 is satisfied. Three sets of graphs are represented by blue, orange, and green points respectively.

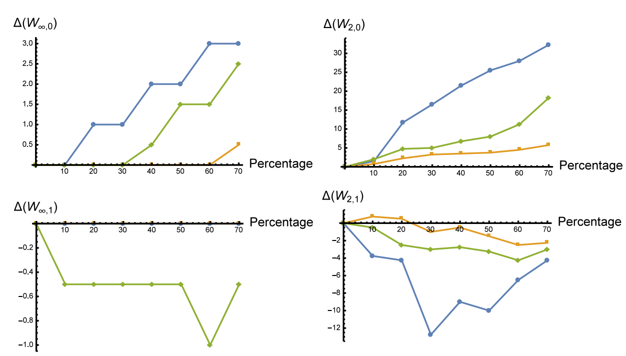
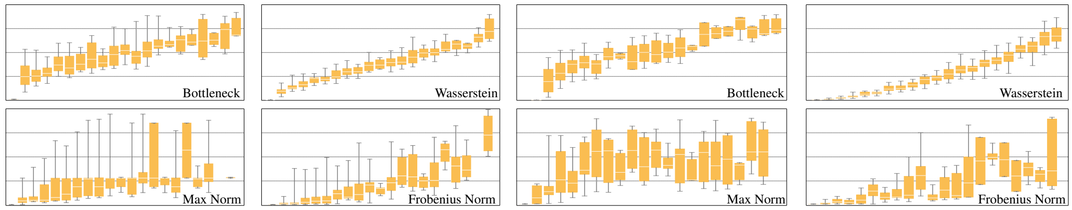


Figure 15: Testing focus awareness (property 4). Each colored curve represents a graph among three randomly generated graphs. The difference between the targeted corruption and the random corruption is plotted against the percentage of the deleted edges.



(a) Shortest-path Distance Metric

(b) Commute-time Distance Metric

Figure 16: Study of the stability for different similarity measures under small perturbations. The x-axis of each plot shows the percentage of edges deleted from the graph. The y-axis represents the difference between the perturbed graph and the original graph. The y-axes are normalized to $[0,1]$ based upon the maximum observed values.

3 test. Our persistence-based similarity measure satisfies the *focus awareness* property in dimension 0 but not in dimension 1. This is because the deletion of an edge might create a cycle in the corrupted graph (see the negative values in Figure 15 bottom).

5.2 Stability Under Perturbation

The persistence diagram computation depends on the distance matrix we impose on a graph. A natural question is: What are the advantages of using the persistence diagram on a graph over the distance matrix itself as a topological fingerprint of the graph? We would like to give some experimental evidence in this section to justify our choice of a persistence-based similarity measure.

To simplify the analysis, we perturb a small percentage of edges on a simple example, the "map of science" graph [7], and we focus only on edge deletion. The experiments we show here use only PD_0. PD_1 is omitted because the results are similar. The map of science graph consists of 554 nodes and 2276 edges; we refer to it as the *baseline* graph, denoted as G_0.

Edge Deletion Model. Our edge deletion is designed as follows: For the i-th perturbation step, $i\%$ of edges are deleted from the baseline G_0 uniformly at random; and such a perturbation is repeated 20 times to obtain (almost) unbiased results. We perform a total of 20 perturbation steps, that is, up to 20% of edges can be deleted from the baseline.

Similarity Measures. We compare variations among various similarity measures. Recall G_0 is the baseline graph, and d_0 is the distance matrix of its metric space representation. Let G_i be an instance of a perturbed graph at the i-the perturbation step and d_i be its distance matrix of its metric space representation. The first set of similarity measures is based on bottleneck and Wasserstein distances. We examine the bottleneck distance W_∞ and the Wasserstein distance W_2 between the 0- and 1-dimensional persistence diagrams associated with G_0 and G_i, respectively. The second set of similarity measures is based upon matrix norms on the distance matrices. We measure the *matrix max norm*, that is, $\|d_i - d_0\|_{\max}$, where $\|A\|_{max} := \max_{ij} |a_{ij}|$ for a matrix A. We also measure the *matrix Frobenius norm*, that is, $\|d_i - d_0\|_F$, where $\|A\|_F := \sqrt{\sum_i \sum_j (a_{ij})^2}$.

Experimental Results. Figure 16 shows our experimental results. Figure 16(a) uses the shortest-path distance metric in the computation of various similarity measures, and Figure 16(b) uses the commute-time distance metric.

Each sub-figure is a box-plot whose y-axis corresponds to a particular similarity measure. Since these similarity measures are not directly comparable, the range of y-axis for each plot has been normalized to $[0,1]$ according to the maximum similarity measure across all experimental instances.

In Figure 16(a), under the shortest-path distance metric, there appears to be a linear relationship between perturbation and the bottleneck distance (and Wasserstein distance). Furthermore, the Wasserstein distance has a smaller variance than the bottleneck distance, making it suitable to study global perturbation in the data. On the other hand, similarity measures based on matrix norms are relatively unstable. Both max norm and Frobenius norm show large fluctuations and variance, making them less suitable for analysis. Moreover, these measures completely fail when the perturbed graph becomes disconnected, which is not an issue for our approach.

In Figure 16(b), under the commute-time distance, we observe that the persistence-based measure appears to be less noisy and more stable than the shortest-path distance metric.

6 Conclusion

Time-varying graphs are becoming increasingly important in data analysis and visualization. In this paper, we address the problem of capturing and visualizing structural changes in time-varying graphs using techniques originated from topological data analysis, in particular, persistent homology. We provide a simple and intuitive visual interface for investigating structural changes in the graph using persistence-based similarity measures.

There are many on-going and future research avenues based upon our approach. For example, in our work, we restrict topological feature extraction to Rips filtrations. Other types of filtrations, such as clique filtration [55], can be used to analyze and understand time-varying graphs.

One interesting question that arises in our approach is how best to convert edge weights into distances. The conventional wisdom is that the stronger the communication between nodes (i.e., higher edge weight), the closer together they should be. However, we have some evidence that such a conversion may not always capture the underlying structural changes, and sometimes, an inverse weighting scheme may be more effective.

It would also be interesting to perform a systematic comparison of a wide range of similarity measures in the study of time-varying graphs [45], in particular, to see how these different measures can complement one another in enriching our current visual analytic framework. A final note is that we hope the work described here could inspire more graph visualization research to move beyond graph-theoretical measures and venture into techniques from topological data analysis.

Acknowledgements

This work was supported in part by NSF IIS-1513616.

References

[1] L. Babai. Graph isomorphism in quasipolynomial time. *arxiv.org/abs/1512.03547*, 2016.

[2] M. Bampasidou and T. Gentimis. Modeling collaborations with persistent homology. *CoRR*, abs/1403.5346, 2014.

[3] M. Bastian, S. Heymann, and M. Jacomy. Gephi: an open source software for exploring and manipulating networks. In *ICWSM*, pages 361–362, 2009.

[4] M. Baur and M. Benkert. Network comparison. In *Network analysis*, pages 318–340. Springer, 2005.

[5] F. Beck, M. Burch, S. Diehl, and D. Weiskopf. The state of the art in visualizing dynamic graphs. *EuroVis STAR*, 2, 2014.

[6] I. Borg and P. J. Groenen. *Modern multidimensional scaling: Theory and applications*. Springer Science & Business Media, 2005.

[7] K. Börner, R. Klavans, M. Patek, A. M. Zoss, J. R. Biberstine, R. P. Light, V. Larivière, and K. W. Boyack. Design and update of a classification system: The ucsd map of science. *PloS one*, 7(7):e39464, 2012.

[8] M. Burch, B. Schmidt, and D. Weiskopf. A matrix-based visualization for exploring dynamic compound digraphs. In *Information Visualisation International Conference*, pages 66–73. IEEE, 2013.

[9] G. Carlsson. Topological pattern recognition for point cloud data. *Acta Numerica*, 23:289–368, 2014.

[10] C. J. Carstens and K. J. Horadam. Persistent homology of collaboration networks. *Mathematical Problems in Engineering*, 2013, 2013.

[11] B. Cassidy, C. Rae, and V. Solo. Brain activity: Conditional dissimilarity and persistent homology. *IEEE 12th International Symposium on Biomedical Imaging (ISBI)*, pages 1356 – 1359, 2015.

[12] G. Chartrand, F. Saba, and H. B. Zou. Edge rotations and distance between graphs. *Časopis pro pěstování matematiky*, 110(1):87–91, 1985.

[13] F. R. Chung. *Spectral graph theory*, volume 92. American Mathematical Soc., 1997.

[14] D. M. Cvetković, M. Doob, and H. Sachs. *Spectra of graphs: theory and application*, volume 87. Academic Pr, 1980.

[15] Y. Dabaghian, F. Mémoli, L. Frank, and G. Carlsson. A topological paradigm for hippocampal spatial map formation using persistent homology. *PLoS Computational Biology*, 8(8):e1002581, 2012.

[16] V. de Silva, D. Morozov, and M. Vejdemo-Johansson. Persistent cohomology and circular coordinates. *Proceedings 25th Annual Symposium on Computational Geometry*, pages 227–236, 2009.

[17] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[18] K. Dinkla, M. A. Westenberg, and J. J. van Wijk. Compressed adjacency matrices: untangling gene regulatory networks. *IEEE Transactions on Vis. and CG*, 18(12):2457–2466, 2012.

[19] I. Donato, G. Petri, M. Scolamiero, L. Rondoni, and F. Vaccarino. Decimation of fast states and weak nodes. *Proceedings of the European Conference on Complex Systems*, pages 295–301, 2012.

[20] C. Dunne and B. Shneiderman. Motif simplification. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2013.

[21] T. Dwyer, N. H. Riche, K. Marriott, and C. Mears. Edge compression techniques for visualization of dense directed graphs. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2596–2605, 2013.

[22] W. E, J. Lu, and Y. Yao. The landscape of complex networks. *CoRR*, abs/1204.6376, 2012.

[23] H. Edelsbrunner and J. Harer. Persistent homology - a survey. *Contemporary Mathematics*, 453:257–282, 2008.

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

[25] J. Ellson, E. Gansner, L. Koutsofios, S. C. North, and G. Woodhull. In *Graph Drawing*, pages 483–484. Springer, 2002.

[26] J. Fournet and A. Barrat. Contact patterns among high school students. *PloS one*, 9(9):e107878, 2014.

[27] F. Fouss, A. Pirotte, J.-M. Renders, and M. Saerens. Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. *IEEE transactions on Knowledge and data engineering*, 19(3):355–369, 2007.

[28] T. M. Fruchterman and E. M. Reingold. Graph drawing by force-directed placement. *Software: Practice and experience*, 21(11):1129–1164, 1991.

[29] E. R. Gansner, Y. Hu, S. North, and C. Scheidegger. Multilevel agglomerative edge bundling for visualizing large graphs. In *IEEE Pacific Visualization Symposium*, pages 187–194, 2011.

[30] E. R. Gansner, Y. Koren, and S. North. Graph drawing by stress majorization. In *Graph Drawing*, pages 239–250. Springer, 2005.

[31] E. R. Gansner, E. Koutsofios, S. C. North, and K.-P. Vo. A technique for drawing directed graphs. *IEEE Transactions on Software Engineering*, 19(3):214–230, 1993.

[32] R. Ghrist. Barcodes: The persistent topology of data. *Bullentin of the American Mathematical Society*, 45:61–75, 2008.

[33] M. Greilich, M. Burch, and S. Diehl. Visualizing the evolution of compound digraphs with timearctrees. In *Computer Graphics Forum*, volume 28, pages 975–982. Wiley Online Library, 2009.

[34] D. Hansen, B. Shneiderman, and M. A. Smith. *Analyzing social media networks with NodeXL: Insights from a connected world*. Morgan Kaufmann, 2010.

[35] D. Holten and J. J. Van Wijk. Force-directed edge bundling for graph visualization. In *Computer Graphics Forum*, volume 28, pages 983–990. Wiley Online Library, 2009.

[36] Y. Hu. Efficient, high-quality force-directed graph drawing. *Mathematica Journal*, 10(1):37–71, 2005.

[37] W. Javed and N. Elmqvist. Exploring the design space of composite visualization. In *PacificVis 2012*, pages 1–8. IEEE, 2012.

[38] S. Kairam, D. MacLean, M. Savva, and J. Heer. Graphprism: Compact visualization of network structure. In *Advanced Visual Interfaces*, 2012.

[39] M. Khoury, Y. Hu, S. Krishnan, and C. Scheidegger. Drawing large graphs by low-rank stress majorization. In *Computer Graphics Forum*, volume 31, pages 975–984. Wiley Online Library, 2012.

[40] Y. Koren, L. Carmel, and D. Harel. Ace: A fast multiscale eigenvectors computation for drawing huge graphs. In *IEEE Symposium on Information Visualization*, pages 137–144, 2002.

[41] D. Koutra, J. T. Vogelstein, and C. Faloutsos. Deltacon: A principled massive-graph similarity function. In *Proceedings of the 2013 SIAM International Conference on Data Mining*, pages 162–170. SIAM, 2013.

[42] J. Leskovec, J. Kleinberg, and C. Faloutsos. Graph evolution: Densification and shrinking diameters. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1):2, 2007.

[43] G. Li, M. Semerci, B. Yener, and M. J. Zaki. Graph classification via topological and label attributes. In *Proceedings of the 9th international workshop on MLG*, volume 2, 2011.

[44] K. Misue, P. Eades, W. Lai, and K. Sugiyama. Layout adjustment and the mental map. *Journal of Visual Languages & Computing*, 6(2):183–210, 1995.

[45] N. D. Monnig and F. G. Meyer. The resistance perturbation distance. *arXiv preprint arXiv:1605.01091*, 2016.

[46] A. Noack. Energy models for graph clustering. *Journal of Graph Algorithms and Applications*, 11(2):453–480, 2007.

[47] P. Papadimitriou, A. Dasdan, and H. Garcia-Molina. Web graph similarity for anomaly detection. *Journal of Internet Services and Applications*, 1(1):19–30, 2010.

[48] D. Selassie, B. Heller, and J. Heer. Divided edge bundling for directional network data. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2354–2363, 2011.

[49] W. T. Tutte. How to draw a graph. *Proceedings of the London Mathematical Society*, s3-13(1):743–767, Jan 1963.

[50] S. van den Elzen, D. Holten, J. Blaas, and J. J. van Wijk. Reducing snapshots to points: A visual analytics approach to dynamic network exploration. *IEEE transactions on visualization and computer graphics*, 22(1):1–10, 2016.

[51] S. V. N. Vishwanathan, N. N. Schraudolph, R. Kondor, and K. M. Borgwardt. Graph kernels. *Journal of Machine Learning Research*, 11(Apr):1201–1242, 2010.

[52] J. T. Vogelstein and C. E. Priebe. Shuffled graph classification: Theory and connectome applications. *arXiv preprint arXiv:1112.5506*, 2011.

[53] T. Von Landesberger, A. Kuijper, T. Schreck, J. Kohlhammer, J. J. van Wijk, J.-D. Fekete, and D. W. Fellner. Visual analysis of large graphs: State-of-the-art and future research challenges. In *Computer graphics forum*, volume 30, pages 1719–1749. Wiley Online Library, 2011.

[54] B. Wang, B. Summa, V. Pascucci, and M. Vejdemo-Johansson. Branching and circular features in high dimensional data. *IEEE Transactions on Visualization and Computer Graphics*, 17:1902–1911, 2011.

[55] A. Zomorodian. The tidy set: A minimal simplicial set for computing homology of clique complexes. *Proceedings 26th ACM Symposium on Computational Geometry*, pages 257–266, 2010.