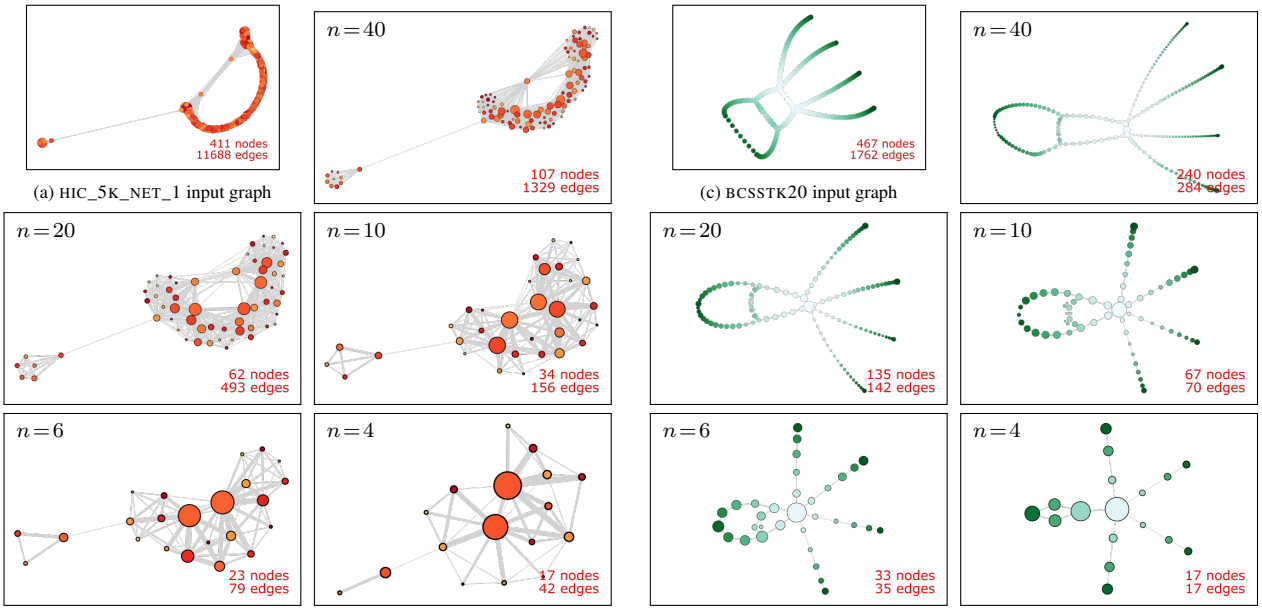
Homology-Preserving Multi-Scale Graph Skeletonization Using Mapper on Graphs

Paul Rosen, Mustafa Hajij and Bei Wang



(a) HIC_5K_NET_1 input graph

(c) BCSSTK20 input graph

(b) PageRank; modularity clustering; $n = \{40, 20, 10, 6, 4\}$; $\epsilon = 0$

(d) Eccentricity (equalized); conn. components; $n = \{40, 20, 10, 6, 4\}$; $\epsilon = 0$

Fig. 1: Our approach adapts the *mapper* construction—a popular tool from topological data analysis—to the visualization of graphs. It provides multi-scale skeletonizations of the graph from diverse perspectives. First, a topological lens is applied to the graph, based upon a property of interest from the graph, for instance, (a) node importance and (b) eccentricity. Then, by modifying a single input parameter, namely the number of cover elements n, our approach provides a multi-scale skeletonization of the input graph that emphasizes the property of interest. In the examples presented here, the visible level-of-detail is reduced as n decreases (within an appropriate range), whereas the homology of the graph—in the form of components and tunnels—remains well persevered.

Abstract—Node-link diagrams are a popular method for representing graphs that capture relationships between individuals, businesses, proteins, and telecommunication endpoints. However, node-link diagrams may fail to convey insights regarding graph structures, even for moderately sized data of a few hundred nodes, due to visual clutter. We propose to apply the mapper construction—a popular tool in topological data analysis—to graph visualization, which provides a strong theoretical basis for summarizing the data while preserving their core structures. We develop a variation of the mapper construction targeting weighted, undirected graphs, called *mapper on graphs*, which generates homology-preserving skeletons of graphs. We further show how the adjustment of a single parameter enables multi-scale skeletonization of the input graph. We provide a software tool that enables interactive explorations of such skeletons and demonstrate the effectiveness of our method for synthetic and real-world data.

Index Terms—Graph visualization, topological data analysis, mapper, skeletonization, multi-scale visualization

1 Introduction

Graphs are often used to model relationships in social, biological, and technological systems. In recent years, our ability to collect and archive such data has far outpaced our ability to understand them. The challenges for graph visualizations are two-fold: how to effectively extract features from such large and complex data; and how to design effective visualizations to communicate these features to the users?

One approach to addressing this problem has been graph aggregation, which creates a skeleton of a graph via node [94] and edge clustering [29, 43]. Common node clustering techniques include spectral methods [34, 45, 67, 106], similarity-based aggregation [100], community detection [81, 82], random walks [60, 89], and hierarchical clustering [16, 20]. There are two limitations to these existing approaches. First, they make no guarantees about the preservation of homological features (i.e., components and tunnels) in the original graph (see Fig. 1). Second, they provide only a single perspective on the aggregation of the graph, driven by the objective of the clustering algorithm.

We propose to address these challenges by adapting the *mapper* construction [98]—a tool from topological data analysis (TDA)—to develop homology-preserving skeletonizations of graphs for visualiza-

- *Paul Rosen is with the University of Utah. E-mail: paul.rosen@utah.edu.*
- *Mustafa Hajij is with the University of San Francisco. E-mail: mhajij@usfca.edu.*
- *Bei Wang is with the University of Utah. E-mail: beiwang@sci.utah.edu.*

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

tion. The original mapper algorithm has enjoyed tremendous success in data science, e.g., cancer research [84], phonemics [62, 108], and others [76, 101].

Our approach works by first applying a *topological lens* to the input graph that captures a certain property of the graph. We consider topological lenses that preserve graph-theoretic properties such as symmetries, density, and centrality. We then perform a series of operations to skeletonize the graph, producing a *topological summary*. The benefits of our approach are three-fold. First, it provides guarantees on preserving the homology of the graph it is summarizing, assuming certain sampling conditions [19, 79]. Second, the topological lens allows observing the graph from multiple perspectives, each highlighting different aspects of the graph being studied. Finally, the mapper construction connects naturally with visualization by providing a strong theoretical basis for multiscale simplifications of the input graph while preserving its core homological structures. In this paper, we develop and evaluate a variation of the mapper construction targeting weighted undirected graphs; the method is referred to as *mapper on graphs*, which produces *mapper graph* as output. Specifically:

- We describe a series of modifications required to make the mapper algorithm effective on graph data;
- We study the utility of five graph-specific topological lenses;
- We demonstrate the effectiveness of our method on synthetic and real-world data;
- We perform a human-subject study showing *mapper on graphs* produces good quality skeletons.

Finally, we provide an open-sourced implementation together with our experimental datasets via GitHub.

2 Related Work

2.1 Graph Visualization

We limit our review to node-link diagrams, which are utilized by many visualization software tools, including Gephi [8], GraphViz [42], and NodeXL [56]. For a comprehensive overview of graph visualization techniques, see [105]. One of the biggest challenges with node-link diagrams is visual clutter, which has been extensively studied in graph visualization [41]. It is addressed in several ways: improved node layouts, edge bundling, lenses, and alternative visual representations.

Tutte [102] provided the earliest graph layout method for node-link diagrams, followed by methods driven by linear programming [50], force-directed embedding [46, 59], graph metric embedding [48], and connectivity structures [17, 63, 65, 66]. TopoLayout [2] creates a hybrid layout by decomposing a graph into subgraphs based on their topological features, including trees, complete graphs, bi-connected components, and clusters, which are grouped and laid out as meta-nodes. The difference between TopoLayout and our work is that we use functions defined on the graph to automatically guide decomposition and feature extraction among subgraphs, and multiple functions induce graph skeletonizations, capturing different properties of the graph.

Edge bundling, which bundles adjacent edges together, is commonly used to reduce visual clutter on dense graphs [57]. For massive graphs, hierarchical edge bundling scales to millions of edges [47], while divided edge bundling [95] tends to produce higher-quality visual results. Nevertheless, these approaches only deal with edge clutter, not node clutter, and they only support limited types of analytic tasks [6, 78].

The concept of applying lenses to view large and complex data has also been used in graph visualization. An early example is the topologically-driven "fisheye views" for visualizing large graphs [49]. Another example is the TugGraph used to explore neighborhoods and paths extending from the foci of interest within a large graph [4, 51].

Finally, alternative visual representations have been used to remove clutter, ranging from variations on node-link diagrams, such as replacing nodes with modules [40] and motifs [39], to abstract representations, such as matrix diagrams [36] and graph statistics [61].

2.2 Graph Clustering

The objective in node clustering (or graph clustering) is to group the nodes of the graph by taking into consideration their edge structure [94]. Vehlow et al. provided a recent survey on the visualization of clustered structures [104] (see Sec. 1 for examples). Edge clustering has also been studied [29, 43]. Importantly, in their assessment of graph readability, Archambault et al. [5] concluded that only clustering could be efficiently performed on large graphs. Broadly speaking, our approach is a type of graph clustering that simultaneously preserves relationships between clusters to retain the homological features of the graph.

Several approaches have leveraged clustering and hierarchical relationships to improve exploration within graphs. For example, van Ham and van Wijk provided methods for interacting with clusters in graph visualizations [103]. Abello et al. built a system, ASK-GraphView, which enables interacting with very large graphs using clustering, among other techniques [1]. Grouse and GrouseFlocks are systems designed to ease the exploration through a feature topology-based hierarchy of the graph [3]. Batagelj et al. created the (x,y)-clustering, with the goal of generating visualizations where intercluster (x) and intracluster (y) graphs retained desired topological properties [9].

2.3 TDA in Graph Analysis and Visualization

Persistent homology (the study of topological features across multi-scales) and mapper construction are two of the most widely used tools in TDA. Persistent homology has been used to analyze graphs [37, 58, 86, 87], with applications to collaboration networks [7, 24] and brain networks [25, 30, 69–72, 88]. In terms of graph visualization, persistent homology has been used in capturing changes in time-varying graphs [55], as well as supporting interactive force-directed layouts [38, 99]. It has also been used to simplify and visualize hypergraphs [109].

The mapper construction [98], which provides a multiscale skeletonization of point clouds has been widely utilized in TDA for a number of applications [21, 75, 76, 84, 101]. It has witnessed recent major theoretical developments (e.g., [23, 33, 79]) that further adjudicate its use in data analysis and provide mathematical foundations for its homology-preserving properties. Recently, Bodnar et al. considered learning the lens of a mapper graph using graph neural networks [13], which is based upon a preprint draft of this work on applying mapper to graphs [54].

3 Mapper on Graphs Construction

Our *mapper on graphs* method—a variation of the classic mapper construction—provides a general framework to skeletonize and visualize a graph. However, to accomplish this, *every stage of the classic mapper construction required significant modification* (see Fig. 3). This includes the selection of graph-specific topological lenses, and modification of the methods used to calculate the cover and the output *mapper graph* nodes and edges.

Whereas the input to classic mapper construction is a high-dimensional point cloud $P \subset \mathbb{R}^n$, the input to *mapper on graphs* is a weighted undirected graph $G = (V, E)$ equipped with a positive edge weight $w : E \to \mathbb{R}$ (see Fig. 3a). If a weight is not provided, several strategies are possible. We assign a uniform edge weight, as seen in other recent works [38, 99].

3.1 Topological Lens (Fig. 3a)

The first step of classic mapper construction is to apply a real-valued function to each point. The function plays the role of a *topological lens* through which we look at the properties of the data. An interesting open problem for the classic mapper construction is how to formulate topological lenses beyond the best practice or a rule of thumb [10, 11]. Nevertheless, prior work has shown that different lenses are important for providing different insights into the data [10, 98].

Table 1: Topological lens comp. scalability and properties preserved

Method	Scalability	Properties Preserved
Average Geodesic Distance (AGD)	Low	Symmetries
Density Estimation	Low	Density
Eccentricity	Low	Centrality
Eigenfunctions (*Fiedler vector)	Low (*High)	Spectral
PageRank	High	Importance

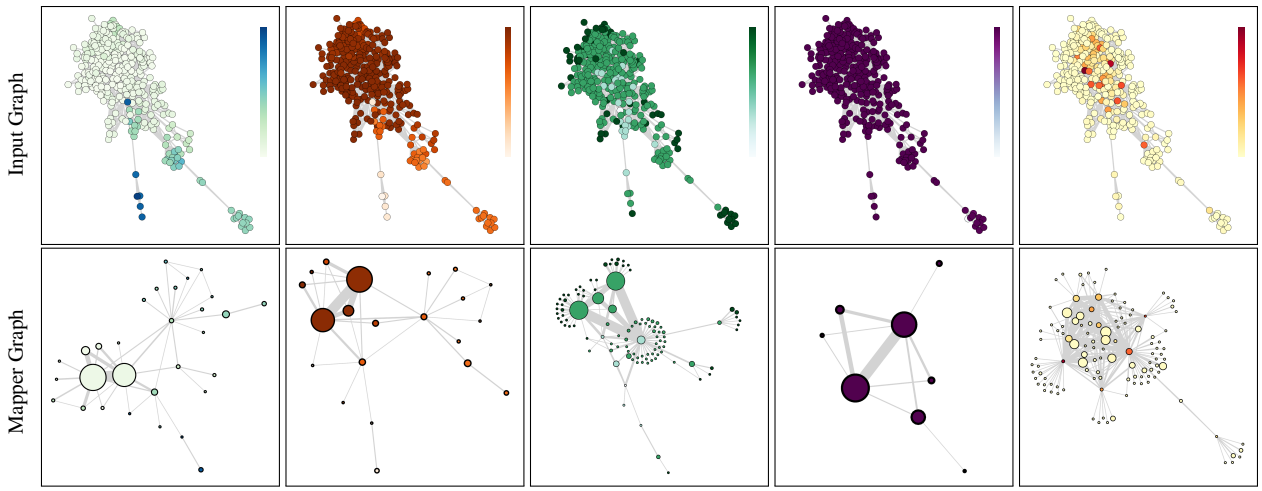


Fig. 2: Examples of the topological lenses applied to the USAIR97 input graph (top) and the resulting *mapper graph* (bottom) using $n = 8$, $\epsilon = 0$, and modularity clustering. Each of the topological lenses provides a slightly different perspective on the input graph, which is reflected in the output *mapper graph*.

For *mapper on graphs*, the *topological lens* (see Fig. 3a) is a real-valued function defined on the nodes, $f : V \to \mathbb{R}$. We focus on five graph-theoretic lenses, including average geodesic distance (AGD), density estimation, eccentricity, eigenfunctions of the graph Laplacian, and PageRank (see Fig. 2), although our framework can be easily extended to include other lenses. Each lens is chosen to reflect a specific property of interest that is intrinsic to the structure of a graph, which are outlined in Tab. 1. In addition to utilizing the function directly, we provide a non-parametric variation obtained using histogram equalization on the topological lens.

3.1.1 Average Geodesic Distance (AGD)

The average geodesic distance (AGD) [64] is a topological lens that detects symmetries in the graph while being invariant to reflection, rotation, and scaling, and it has been used extensively in shape analysis due to these properties [64]. AGD is calculated by first defining a geodesic distance $d(u, v)$ between any two nodes $u, v \in V$, in our case, utilizing Dijkstra's shortest path algorithm. The AGD is given by

$$AGD(v) = \frac{1}{|V|} \sum_{u \in V} d(v, u).$$

This definition implies that the nodes near the center of the graph will likely have low function values, while points on the periphery will have high values. See Fig. 2a for an example.

3.1.2 Density Estimation

Density estimation helps to differentiate dense regions from sparse regions and outliers. The density estimation function [97] is given by

$$D_\delta(v) = \sum_{u \in V} \exp\left(\frac{-d(u, v)^2}{\delta}\right),$$

where $d(u, v)$ is the geodesic distance between two nodes in the graph and $\delta > 0$. D_δ tends to differentiate dense regions from sparse regions and outliers, and interestingly, it tends to negatively correlate with the AGD. We set $\delta = 0.5$ for all examples. See Fig. 2b for an example.

3.1.3 Eccentricity

Eccentricity measures the maximum distance from one node to all others. The eccentricity of a graph node is given by

$$Ecc(v) = \max_{u \in V}(d(u, v)),$$

where $d(u, v)$ is the geodesic distance between two nodes in the graph. Eccentricity measures centrality on the graph, since nodes that are towards the center need to travel a shorter distance to reach all other nodes than those on the periphery. See Fig. 2c for an example.

3.1.4 Eigenfunctions of the Graph Laplacian

Eigenfunctions of the graph Laplacian L capture spectral properties of the graph [68]. In particular, the gradient of the eigenfunctions of L tends to follow the overall shape of the data [74], and these functions have been used in applications, such as graph understanding [96], segmentation [92], spectral clustering [83], and min-cut problems [77]. Let $C(G)$ be the vector space of all functions, $f : V \to \mathbb{R}$. The unnormalized graph Laplacian of G is the linear operator $L : C(G) \to C(G)$ defined by mapping $f \in C(G)$ to Lf, with weight, w, where

$$(Lf)(v) = \sum_{u \sim v} w_{u,v}(f(v) - f(u)).$$

Sorting the eigenvectors of L by increasing eigenvalues, we can use eigenvectors of the second or the third smallest eigenvalues of L as the lens, denoted as l_2, l_3, and so on. We focus in particular on l_2 (both unnormalized and normalized), commonly referred to as the Fiedler vector [74], since it has desirable geometric properties [35].

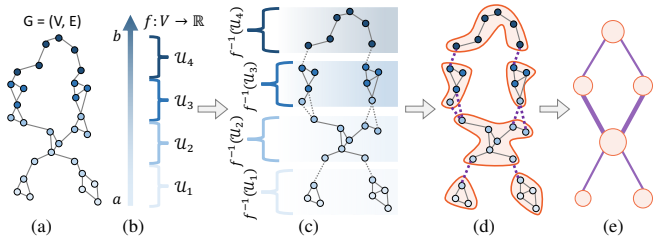


Fig. 3: An illustration of the *mapper on graphs* construction: (a) A weighted graph $G = (V, E)$ with a topological lens $f : V \to \mathbb{R}$. (b) A cover $\mathcal{U}$ of the range space is given by intervals U_1, U_2, U_3, and U_4 as cover elements. (c-d) The connected subgraphs induced by $f^{-1}(U_i)$ form a cover of G. (e) The resulting *mapper graph* is a 1-dimensional skeleton whose nodes represent the connected subgraphs (in orange), and edges represent the graph-cut between the subgraphs (in purple). Observe that the output *mapper graph* retains the main homological features of the input graph, which is represented by a large cycle and the two components attached at its bottom.

For example, the minimum and maximum of the Fiedler vector tend to occur at nodes with maximum geodesic distances [27]. Efficient computations of the Fiedler vector also exist, making it feasible to use in larger graphs. See Fig. 2d for an example.

3.1.5 PageRank

Except for the Fiedler vector, prior topological lenses are computationally expensive, making them prohibitive to use on large graphs. To address the scalability issue, we utilize *PageRank* [18] as a computationally efficient lens that measures the importance of nodes. We use a version of the PageRank algorithm applicable to undirected graphs [52]. A PageRank vector $R : V \to \mathbb{R}$ is defined for every node $v \in V$,

$$R(v) = \frac{(1-d)}{|V|} + d \sum_{u \in N(v)} \frac{R(u)}{|N(u)|},$$

where $N(v)$ is the set of neighbors of v; $0 < d < 1$ is the *damping factor*, which is set at 0.85 for our examples. Using this formation of $R(v)$, PageRank yields an iterative algorithm that can be computed efficiently in practice [18, 85]. A high PageRank score at v typically means that v is connected to many nodes that also have high PageRank scores. PageRank has been shown to be a continuous function that has many properties similar to density [90].

3.2 Cover of f (Fig. 3b)

In the second step, a *cover* $\mathcal{U}$ is formed over the range of f, $f(V) \subseteq [a, b]$ (see Fig. 3b). The cover consists of a finite number of open intervals called cover elements, $\mathcal{U} = \{U_\alpha\}_{\alpha \in A}$. To obtain the cover, we use a common strategy, *uniformly sized overlapping* intervals. Given a graph G equipped with a topological lens $f : V \to \mathbb{R}$, suppose the range of the function $f(V)$ is normalized to be within $[0, 1]$. To obtain an initial cover, we split $[0, 1]$ into n (the *resolution* parameter) intervals $[c_1, c_2], \cdots, [c_{n-1}, c_n]$ with equal length, such that $c_1 = 0$ and $c_n = 1$. Adjusting the resolution parameter increases or decreases the amount of aggregation *mapper on graphs* provides. Broadly speaking, smaller n leads to a smaller topological summary of the graph. Fig. 4a-4d show an example by varying the number of cover elements n.

The *overlap* parameter ϵ is used to obtain the cover $\mathcal{U}$ consisting of cover elements $(c_i - \epsilon, c_{i+1} + \epsilon)$ for $1 \leq i \leq n - 1$. Fig. 4e-4h show an example of increasing ϵ. Increasing the overlap causes the number of output nodes and density of edges in the *mapper graph* to increase, which is caused by the duplication of data in the overlapped cover elements. The necessity for the overlapping cover elements in the classic mapper is partially attributable to the lack of connectivity in point clouds. Importantly, *in a significant departure from classic mapper, we utilize the edges of the graph for the connectivity*, as will be discussed shortly. *This modification eliminates the need for overlap.* Therefore, we use $\epsilon = 0$ in all of our experiments.

3.3 Inverse Image (Fig. 3c)

In the third step, the *inverse image of the cover* is calculated, as shown in Fig. 3c. For each cover element $\mathcal{U}_\alpha$, the inverse image $f^{-1}(\mathcal{U}_\alpha)$ is the subset of nodes $V_\alpha \subseteq V$, where $f(V_\alpha) \in \mathcal{U}_\alpha$. In other words, a node is associated with a cover element whose interval cover the node's function value. In contrast with the classic mapper, for each element of the cover $\mathcal{U}_\alpha$, we find the nodes $V_\alpha = f^{-1}(\mathcal{U}_\alpha)$ and extract its induced subgraph $H_\alpha \subseteq G$, where $f(V_\alpha) \in \mathcal{U}_\alpha$. It is important to note that when the overlap is used in cover elements (i.e., $\epsilon \neq 0$), some nodes from the input graph will be replicated across multiple inverse images.

3.4 Mapper Graph Nodes and Edges (Fig. 3d)

The final step of the process is *forming nodes and edges of the output mapper graph*, as shown in Fig. 3d.

3.4.1 Forming Mapper Graph Nodes

To form the nodes of the output, classic mapper construction applies a clustering algorithm to a set of points in the inverse image of a cover element (e.g., agglomerative clustering or DBSCAN [44]). In our setting, the subgraphs H_α extracted with *mapper on graphs* have additional structures coming from the connectivity of the graph. Therefore, to form the nodes of the *mapper graph*, we apply a graph clustering algorithm to each subgraph H_α. Every cluster within H_α is returned as a node of the output *mapper graph*.

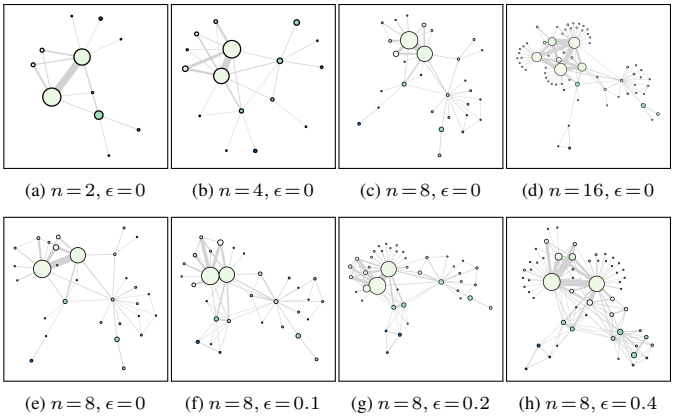


Fig. 4: An example using AGD while varying n and ϵ in a cover on the USAIR97 data: (a-d) fixing $\epsilon = 0$, $n = \{2, 4, 8, 16\}$, respectively; and (e-h) fixing $n = 8$, $\epsilon = \{0, 0.1, 0.2, 0.4\}$, respectively.

We apply three clustering algorithms that preserve different aspects of the subgraph. The first identifies connected components in each subgraph (see Fig. 5a), that is, groups of nodes in H_α that are reachable to each other. The second algorithm is modularity clustering [15] (see Fig. 5b), which identifies communities with dense connections between nodes and sparse connections between communities. Finally, we consider label propagation (see Fig. 5c), which identifies communities by iteratively assigning node labels and propagating the labels through the graph. We tested two label propagation algorithms (semi-synchronous [28] and asynchronous [91]) and obtained similar results (all presented results use asynchronous label propagation).

Fig. 5 show the results of these clustering algorithms, which preserve similar structures. However, the graph details they retain vary based on the properties of the underlying algorithm.

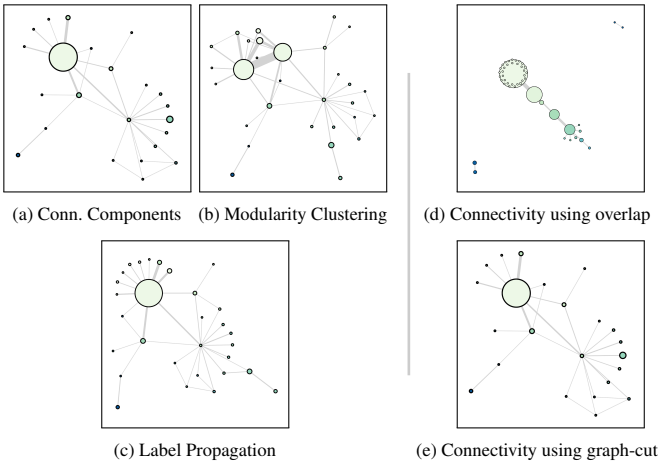


Fig. 5: Left: An example of using (a) connected components, (b) modularity clustering, and (c) asynchronous label propagation as node clustering methods on the USAIR97 data with $n = 8$ and $\epsilon = 0$. Right: An example of using different edge forming methods on the USAIR97 data with $n = 8$, $\epsilon = 0$, and connected components.

3.4.2 Forming Mapper Graph Edges

Finally, we form the edges of the *mapper graph*. Initially, we test a classic (mapper) approach that identifies the overlapping node sets between clusters. For all mapper graph nodes created in the prior step, if they share any node from the original graph, they are connected with a weight equal to the overlap between them. In other words, consider two mapper nodes N_i and N_j ($i \neq j$), they are connected if $N_i \cap N_j \neq \emptyset$ with $w_{ij} = |N_i \cap N_j|$. Unfortunately, this method does not work as well as we have hoped. As seen in Fig. 5d, the method has two undesirable problems. First, the required overlap creates a few dense connections. Second, the method does not guarantee connectivity.

Instead, we use the connectivity of the input graph to determine the connectivity of the output graph. We utilize the number of edges in the input graph that go between any two *mapper graph* nodes N_i and N_j ($i \neq j$). We count the number of edges between the nodes of N_i and N_j, which can be efficiently computed using a graph-cut. The nodes are connected if $graph\_cut(N_i, N_j) \neq \emptyset$, with $w_{ij} = |graph\_cut(N_i, N_j)|$. Fig. 5e shows an example. Alternative approaches could sum ($w_{ij} = \sum_{q \in graph\_cut(N_i,N_j)} w_q$) or apply a statistic, such as the mean, to the weights of the edges of the graph-cut. The utility of such approaches depends upon the analytical goals of the visualization. In comparison with the initial approach, the graph-cut-based method better preserves graph details and eliminates the need for enforcing overlaps among cover elements. Unless otherwise specified, all examples in the paper use the graph-cut method to form edges in the *mapper graph*.

3.5 Visualization (Fig. 3e)

The final output *mapper graph* (Fig. 3e) provides an overview of the homological structure of the input graph (Fig. 3a) through multiple perspectives using different topological lenses. The levels of details captured by the *mapper graph* may be adjusted by modifying the cover. Its visualization consists of two components (see Fig. 6): a visualization of the *mapper graph*, and a visualization of the cover with a histogram useful for parameter selection.

3.5.1 Output Mapper Graph Visualization

The output of *mapper on graphs* is a *mapper graph* with additional metadata, which is the primary visualization target, with flexibility in its layout and visual encodings; see Fig. 6d. For the purpose of this paper, we have chosen a simple approach. The *mapper graph* is laid out using a standard force-directed layout provided by D3.js. Our approach is ultimately agnostic of the graph layout algorithm, and different layouts (e.g., layered approaches) may improve the presentation under certain scenarios. For illustrative purposes, we also add several (optional) visual encodings to better understand the information captured by *mapper on graphs*. First, a *mapper graph* node N_i (i.e., the cluster $N_i \subseteq G$) is colored using the average function value across its members using a topological lens specific colormap (see Fig. 2). Next, the sizes of the nodes are proportional to $|N_i|$. Finally, edge thickness is drawn proportional to the edge weight (i.e., $|graph\_cut(N_i, N_j)|$).

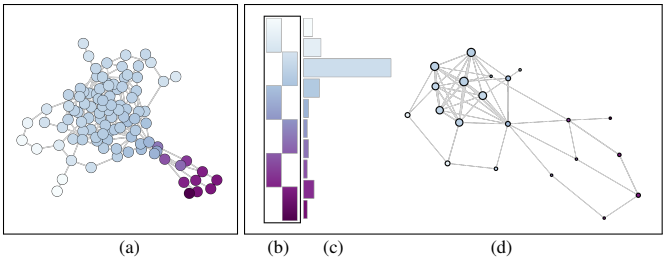


Fig. 6: Visualization of the (a) MOVIES graph includes a visualization of (b) the cover elements $\mathcal{U}$, (c) the histogram of the topological lens f, and (d) the *mapper graph* ($n=6$, $\epsilon=0$, with modularity clustering).

3.5.2 Histogram and Cover Visualization

To assist with parameter selection, we provide additional information about the input graph, which is particularly useful when the input graph is too large to draw.

To understand the distribution of a topological lens f, a histogram of the function values is laid out vertically, with the minimum value a at the bottom and the maximum value b at the top. Fig. 6c shows an example of a histogram for the Fiedler lens. The histogram is split into an adjustable number of bins.

The cover (see Fig. 6b) is visualized using a series of boxes, one per cover element, displayed next to the histogram of the topological lens. Each box is placed and colored based on its corresponding interval. The boxes are laid out horizontally automatically using a greedy packing approach. Observing patterns in the histogram and its alignment with the cover elements can help inform choices for n, ϵ, and the equalization of the topological lens.

4 Evaluation

We evaluate the computational performance of our approach (see Sec. 4.1) and its ability to provide multiscale homology-preserving skeletonizations (see Sec. 4.2, 4.3, and 4.4).

Implementation. Our approach is implemented using Python for the calculation of the *mapper graph* and D3.js for visualization. The code and datasets are available on GitHub <https://github.com/shape-vis/MapperOnGraphs>. A demo version of the software is available at <https://shape-vis.github.io/MapperOnGraphsDemo/>.

Datasets. We evaluate our approach using over 45 graphs (see Tab. 2). We focus on a subset of small ($|V| \in [100, 1\,000)$, $|E| \in [200, 16\,651]$), medium ($|V| \in [1\,000, 5\,000)$, $|E| \in [1\,092, 284\,924]$), and large ($|V| \in [5\,000, 1\,134\,890]$, $|E| \in [91\,342, 2\,987\,624]$) graphs. The datasets are a mix of synthetic and real-world datasets. Graphs not in the paper are available in the demo.

Table 2: Listing of Datasets Used in Evaluation

Dataset	$\|V\|$	$\|E\|$	Source
AIRPORT	2,896	15,641	Openflights.org
AMAZON0302	262,111	899,792	SNAP [73]
BALANCED TREE(3,6)	1,093	1,092	NetworkX [53]
BARBELL(50,20)	120	2,471	NetworkX [53]
BCSSTK	110	254	UF Sparse Matrix Repo. [31]
BCSSTK20	467	1,762	UF Sparse Matrix Repo. [31]
BCSSTK22	110	364	UF Sparse Matrix Repo. [31]
BIO-CELEGANS	453	2,025	Network Repository [93]
BIO-DISEASOME	516	1,188	Network Repository [93]
BN-MOUSE-VISUAL-CORTEX-2	193	214	Network Repository [93]
CA-CONDMAT	21,363	91,342	SNAP [73]
CALTECH	762	16,651	Facebook 100
CHCH-MINER	1,510	48,512	BioSNAP [110]
CIRCULAR LADDER(100)	200	300	NetworkX [53]
COLLABORATION NETWORK	379	914	[80]
COM-AMAZON	334,863	925,872	SNAP [73]
COM-YOUTUBE	1,134,890	2,987,624	SNAP [73]
CONNECTED CAVEMAN(15,30)	450	6,525	NetworkX [53]
DCH-MINER	7,197	466,656	BioSNAP [110]
DF-MINER	20,549	802,760	BioSNAP [110]
DOROGOVTSEV GOLTSEV MENDES(5)	123	243	NetworkX [53]
ENRON-EMAIL	143	623	SNAP [73]
FF-MINER	46,007	106,499	BioSNAP [110]
HIC 1K NET 1	5,420	315,852	
HIC 1K NET 2	5,096	333,129	
HIC 1K NET 3	5,345	320,659	BioSNAP [110]
HIC 1K NET 4	5,472	378,343	
HIC 1K NET 5	7,015	434,977	
HIC 1K NET 6	4,581	284,924	
HIC 5K NET 1	411	11,688	
HIC 5K NET 2	308	9,599	
HIC 5K NET 3	632	19,469	
HIC 5K NET 4	265	8,402	BioSNAP [110]
HIC 5K NET 5	1,032	32,329	
HIC 5K NET 6	192	6,242	
HIC 5K NET 7	435	13,428	
HIC 5K NET 8	1,194	37,569	
MAP OF SCIENCE	554	2,276	[14]
MOVIES	101	192	[32]
RANDOM LOBSTER(100,0 6,0 4)	596	595	NetworkX [53]
RING OF CLIQUES(6,70)	420	14,496	NetworkX [53]
SMITH	2,970	97,133	Facebook 100
SOC-EPINIONS1	75,877	405,739	SNAP [73]
USAIR97	332	2,126	Network Repository [93]
WATTS STROGATZ(100,5,0.05)	100	200	NetworkX [53]

4.1 Performance

4.1.1 Calculating Topological Lenses

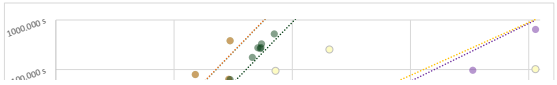
Calculating the topological lens is oftentimes the most expensive part of generating a *mapper graph*. We recorded the compute time for all lenses on all graphs in our test set. In some cases, certain lenses can not be computed in a reasonable amount of time. We terminated those computations after 900 seconds.

Fig. 7a shows the (completed) compute time for all lenses. They are laid out by the number of input graph nodes ($|V|$) against the compute time in seconds using a log-log scale. A power regression is plotted for each lens. The experimental results show that the methods from fastest to slowest are: (1) PageRank: $O(|V|^{1.07})$, (2) Fiedler: $O(|V|^{1.17})$, (3) eccentricity: $O(|V|^{2.45})$, (4) density: $O(|V|^{2.45})$, and (5) AGD: $O(|V|^{2.46})$. One conclusion is that Fiedler and PageRank perform only slightly worse than linear, making them good candidates for large graphs. The other methods—AGD, density, and eccentricity—are worse than quadratic, making them impractical on large graphs.

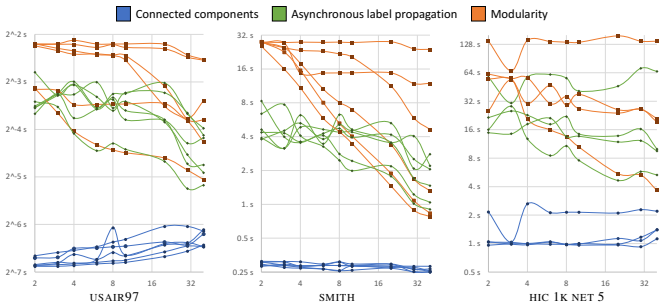
4.1.2 Calculating Mapper Graphs

As we investigated the performance of the *mapper on graphs* framework, it became clear that the performance was highly dependent upon the clustering algorithm. Fig. 7b shows the time to calculate *mapper graph* for three typical datasets, one small, one medium, and one large, across several lens functions and various numbers of cover elements. In addition, it differentiates the clustering algorithm by color. The connected components algorithm (in blue) is the most efficient, often by several orders of magnitude. We also observed that asynchronous label propagation (in green) is often faster than modularity clustering. In addition, for both asynchronous label propagation and modularity clustering, the time required decreases as the number of cover elements increases. This occurs because as more cover elements are added, the number of input graph nodes that fall into each inverse image decreases.

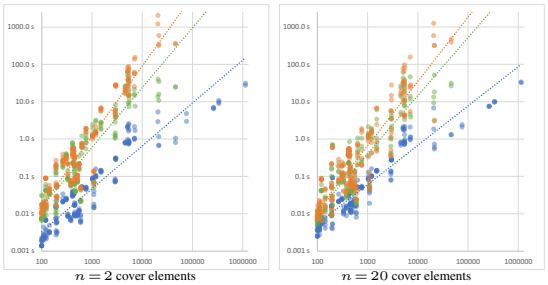
Fig. 7c shows the time to calculate a *mapper graph* (excluding lens calculation) for all datasets and all lenses with $n=2$ and $n=20$ cover elements. The plots show the performance trend of different clustering algorithms across a variety of datasets. The power regressions show that from fastest to slowest are: (1) connected components: $O(|V|^{1.15})$ and $O(|V|^{1.05})$; (2) asynchronous label propagation: $O(|V|^{1.60})$ and $O(|V|^{1.55})$; and (3) modularity clustering: $O(|V|^{1.88})$ and $O(|V|^{1.72})$, for $n=2$ and $n=20$, respectively. In summary, connected component clustering is significantly more scalable than label propagation, which is somewhat more scalable than modularity clustering.



re
e)
).



(b) Compute time for small (USAIR97), medium (SMITH), and large (HIC 1K NET 5) graphs across all topological lenses. The plots show time in seconds vertically, and the number of cover elements horizontally. Instability is associated with variability in clustering time.



(c) Compute time for all graphs and lens functions for (left) $n=2$ and (right) $n=20$ cover elements. The plot shows the time in seconds vertically, and the number of input graph nodes ($|V|$) horizontally. The power regressions (dashed lines) show that the size of the input graph and the clustering algorithm have an influence on the compute time.

Fig. 7: Log-log plots of the compute time of (a) each topological lens, and *mapper on graphs* for (b) different sized datasets across all topological lenses and (c) all graphs and lens functions across different numbers of cover elements. The data are colored by (a) their associate topological lens and (b-c) the chosen clustering algorithm.

4.2 Examples

A major challenge in evaluation is that we were unable to identify appropriate quantitative metrics for comparing skeltonizations in our context [107]. As such, we relied on qualitative evaluations in this section and Sec. 4.3 and a human-subject evaluation in Sec. 4.4.

4.2.1 Multi-scale Homology Preservation

To demonstrate the homology preservation properties of our approach, we show a variety of examples. The examples were selected such that the homology of the input graph was clear, and a corresponding *mapper graph* was selected that reflected that homology. Fig. 1, Fig. 8, and Fig. 9 show two, two, and four examples, respectively. These examples were chosen to show diversity among topological lenses and clustering algorithms. In all examples, as the number of cover elements n increases, the *mapper graph* becomes more similar to the input graph. As n decreases, the *mapper graph* contains fewer nodes while still retaining important homological structures. These structures include the preservation of tunnels, which are particularly visible in both graphs of Fig. 1 and can also be observed in Fig. 9a, 9b, 9c, 9d, and others.

4.2.2 Large Graphs

We demonstrate the effectiveness of our approach to summarize large graphs as well. Fig. 10 shows results for three larges graphs, AMAZON0302, COM-AMAZON, and COM-YOUTUBE, with different numbers of cover elements n. As n increases, the topological structure remains visible with more details added. In the case of COM-AMAZON, the graph eventually becomes a hairball. The same would eventually happen for the other graphs as n and the level-of-detail increase.

4.2.3 Case Study: USAIR97

The USAIR97 dataset is an undirected graph whose nodes are 332 airports in the US, and edges encode direct flights between the airports in 1997. We are interested in studying the properties of the network in terms of identifying which airports have high or low access to other airports in the network. Since US airlines tend to work on the hub-and-spoke model, we chose PageRank as the topological lens, since it identifies nodes of high importance (i.e., hubs). The input graph and three levels of *mapper graph* skeletonizations are shown in Fig. 12a.

Using the most aggregated *mapper graph* version (see Fig. 12b), we identified several groups of airports with varying levels of access. At the highest level are hub nodes in orange (type 1), which contain airline hubs (e.g., LAX/Los Angeles, ATL/Atlanta, etc.) and popular vacation

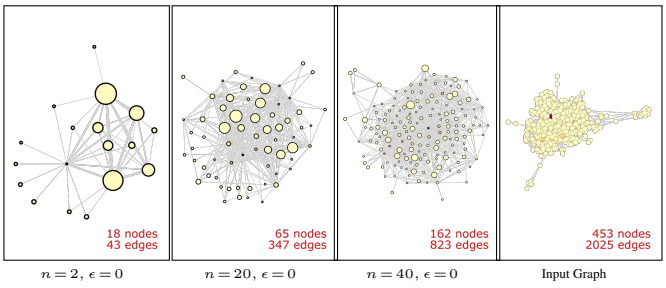
destinations (e.g., LAS/Las Vegas and MCO/Orlando). Unsurprisingly, flights from these airports can directly access many other airports, as well as connecting limited access airports. Another group with high access are the yellow nodes in the center (type 2), connected to all three hub groups. These airports can reach most destinations directly or through a single layover. The next group with medium access are the few airports that are connectors between two hub groups (type 3). These airports connect to multiple hubs and mostly require a single layover. Finally, the limited access airports are the fans that connect to hub nodes (type 4). These airports require going through at least one hub to reach any destination (besides the hub itself).

4.3 Comparison With Community-based Skeletonization

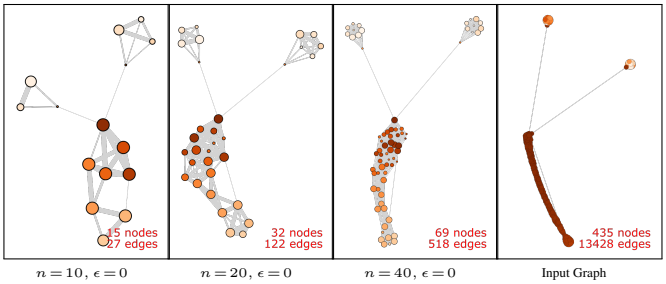
To compare with existing approaches for skeletonization via clustering, we compare our approach with two community finding algorithms provided by NetworkX [53] that offer a resolution parameter, namely, modularity-based communities [15] and Louvain community detection [12]. To ease comparison, graphs are constructed using the method described in Sec. 3.4. We show several examples in Fig. 11. For each example, the mapper graph and community-based graph were chosen to have approximately the same number of nodes with the exception of Fig. 11b, which had five large fans. In many cases, the community-based methods produce graphs that are similar to the *mapper on graphs* approach (e.g., see Fig. 11a). However, there are some distinctions.

Multiple Perspectives. Community-driven skeletonization provides only a single perspective on the graph. The ability to use different topological lenses provides *mapper on graphs* with the ability to provide multiple context-specific visualizations of the homological structure of an input graph. Fig. 11b shows the most dramatic difference. However, for each example in Fig. 11, the *mapper graph* reveals some variation in the structure.

Maintenance of Homological Features. Community-drive skeletonization often preserves some homological structures, tunnels in particular. However, these methods make no guarantees of the preservation of these features. In particular, community-based methods struggle with dense graphs where extremities are lost, and no structure is visible in the core of the graph. Fig. 11d and 11e are the best examples of this, where the output community-based graphs are hairballs, and the *mapper on graphs* graphs provide insights into the graph structure.



(a) BIO-CELEGANS: PageRank, modularity clustering



(b) HIC 5K NET 7: density (equalized), connected components

Fig. 8: A series of small input graphs demonstrating how *mapper on graphs* retains the prominent homological structure of the input graph as the number of cover elements, n, is varied.

4.4 Human-Subject Evaluation

To test if our approach captures the overall homology of the graph, we ran an IRB-approved human-subject evaluation on Mechanical Turk.

4.4.1 Experiment Design

We generated 1140 stimuli (i.e., *mapper graphs*) by computing all permutations of the following parameters:

- Small graphs from Tab. 2 / Fig. 13b (19 total)
- Number of cover elements: $n = \{6, 10, 20, 30\}$
- Clustering methods: {connected components, modularity clustering, asynchronous label propagation}
- Topological lens: {AGD, density, eccentricity, Fiedler vector, PageRank}
- Graphs were laid out using the approach in [38] to ensure consistency.

For each trial, subjects were exposed to one input graph and four *mapper graphs* (see Fig. 13a): two of the *mapper graphs* were generated from the input graph using different topological lenses selected at random; the other two *mapper graphs* were generated from randomly selected graphs and parameters. The subject was asked: "Which picture has the most similar structure to the reference picture?" The idea was that we were trying to capture if their concept of the graph shape was preserved by our approach. They could select one of the four *mapper graphs* or an "Unsure" option.

4.4.2 Subjects and Trials

The study was conducted using 25 subjects. The experiment took 5-10 minutes, and subjects were compensated $1.50 USD. Two subjects failed to complete the experiment and were excluded for a total of 23 subjects. In terms of age, subjects reported: $7 \times [25, 34]$; $11 \times [35, 44]$; $1 \times [45, 54]$; and $4 \times [55, 64]$. In terms of gender, 10 reported female, and 13 reported male. In terms of visualization experience, 10 reported minimal or none, 8 reported casual, and 5 reported regular or extensive experience. The 23 subjects were each exposed to one warm-up question and 37 trials (amounting to each input graph seen twice) for a total of 851 trials. The 15-second timer expired for 11 trials, leaving a total of 840 trials for analysis.

4.4.3 Analysis

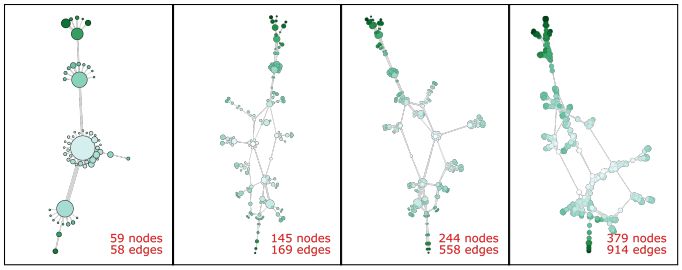
We predicted that the subjects would select *mapper graphs* associated with the input graph with high accuracy. Since each subject was exposed to two stimuli from the input graph, the null hypothesis was that they would select one of the correct graphs 50% of the time.

Fig. 13b summarizes the results of the survey. Accuracy is defined as $N_{correct}/(N_{correct} + N_{incorrect} + N_{unsure})$. In the final overall column, the results show that subjects had a 79% accuracy at picking one of the two correct answers, well above the 50% null hypothesis. A binomial test, which is appropriate for testing whether the expected distribution of a binary outcome (correct vs. incorrect) matches the observed distribution, was run to determine if the difference was statistically significant with a p-value $p < 0.001$, thus confirming our hypothesis.

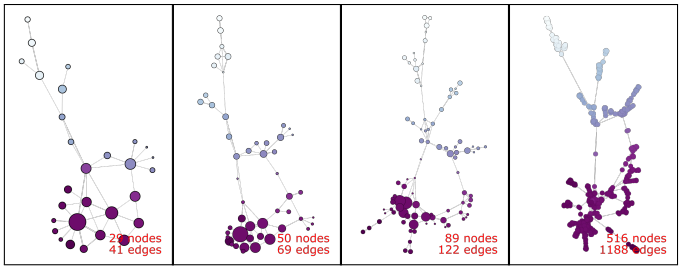
Fig. 13b also reports on the results for individual graphs. Participants performed well with most of the input graphs, except the CIRCULAR LADDER and the CONNECTED CAVEMAN. Upon further inspection, we discovered that several challenging examples were generated for these graphs (see Fig. 14). In these cases, the combination of the topological lens, the clustering algorithm, and the level of aggregation (n) made the relationship difficult to identify, leading to lower scores.

4.4.4 Ecological Validity

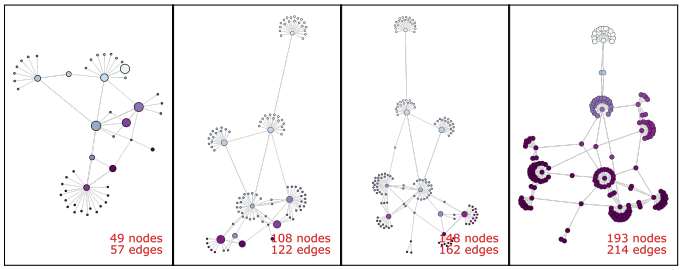
Our experimental question was intentionally vague in order to elicit a response that captured subjects' instincts about what makes two graph visualizations similar. Within this narrow context, our experiment has high ecological validity. In other words, if the visual analytics task is to determine if two graphs have the same "structure", our experiment showed that participants largely found our approach to satisfy that



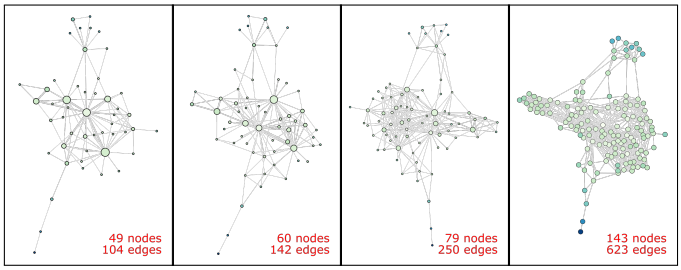
(a) COLLABORATION NETWORK: eccentricity, connected components



(b) BIO-DISEASOME: Fiedler vector, modularity clustering

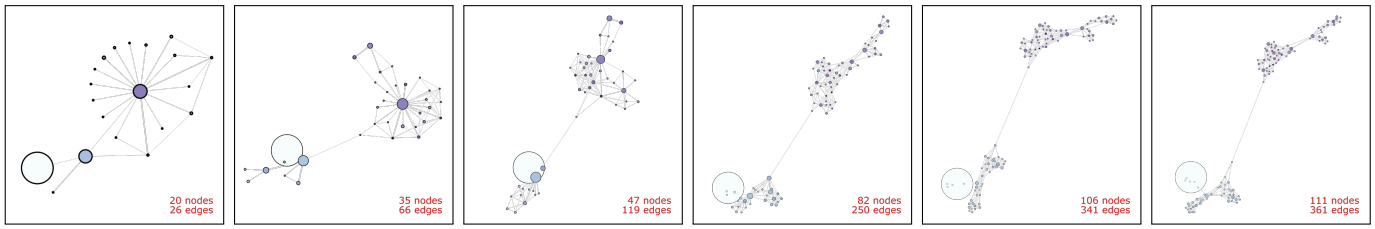


(c) BN-MOUSE-VISUAL-CORTEX-2: Fiedler vec. (equalized), async. label prop.

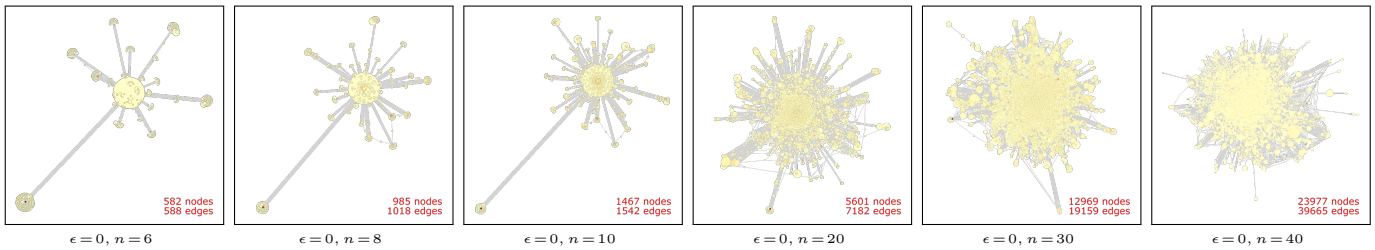


(d) ENRON-EMAIL: AGD, asynchronous label propagation

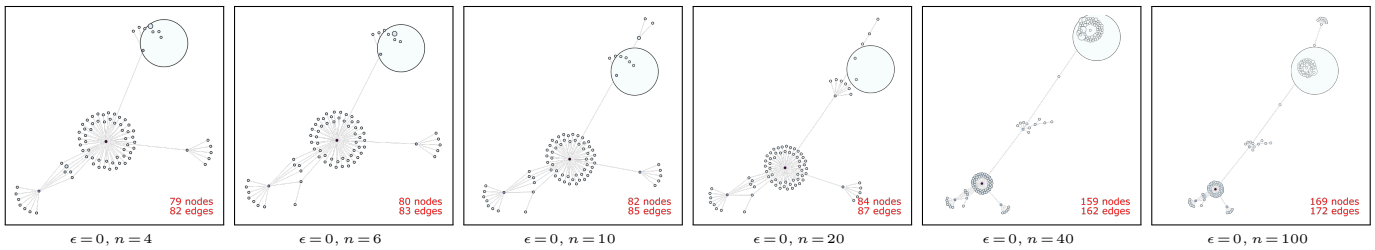
Fig. 9: Continuation of Fig. 8.



(a) AMAZON0302 ($|N| = 262, 111, |E| = 899, 792$), normalized Fiedler vector, connected components



(b) COM-AMAZON ($|N| = 334, 863, |E| = 925, 872$), PageRank, connected components



(c) COM-YOUTUBE ($|N| = 1, 134, 890, |E| = 2, 987, 624$), normalized Fiedler vector, connected components

Fig. 10: Examples of large input graphs demonstrating how a *mapper on graphs* can reveal certain prominent topological structure of graphs so large that meaningful node-link diagrams are difficult to draw.

requirement. Furthermore, the layout itself, which was randomly initialized, did not significantly impact the results. Therefore, *our experiment validates that our approach preserves homology and that homology-preservation is important*. However, we note that this is not the overall use case for our approach, which is multi-scale homology-preserving skeletonizations of graphs.

5 Conclusion and Discussion

In this paper, we present a TDA-based approach for graph visualization using a construction called *mapper on graphs*. The approach is effec-

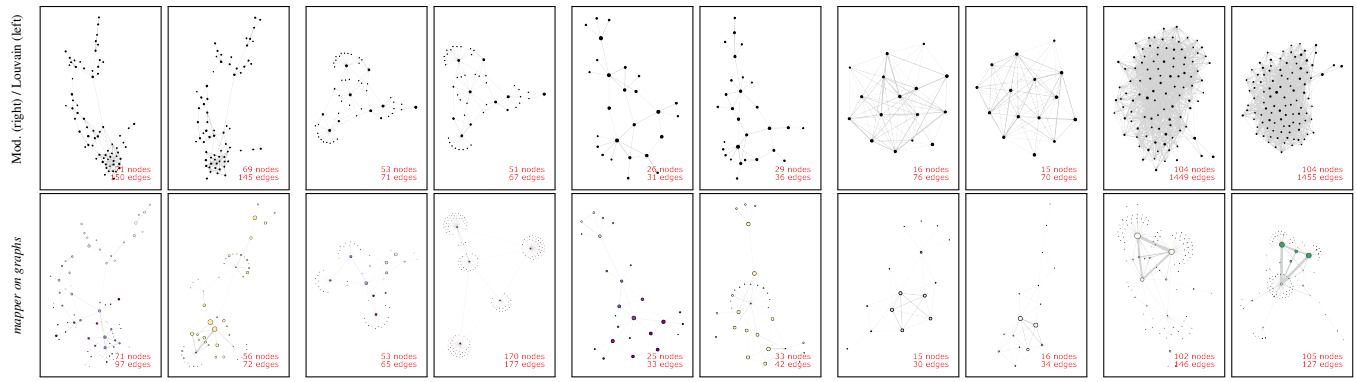
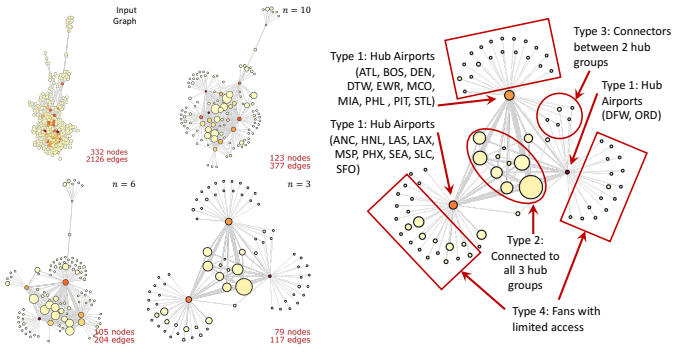


Fig. 11: Community-based skeletonization (top) compared to *mapper on graphs* skeletonization (bottom). Parameters were chosen so that graphs generally had a similar number of nodes. We observe that while community-based techniques sometimes preserve homology, (a-e) *mapper on graphs* provides multiple perspectives on the graph, and in some cases, (d-e) community-based approaches do not capture any structure.

tive at creating skeletonizations of graphs that capture the structure across multiple scales and from multiple perspectives using different topological lenses.

Parameter Selection. The flexibility of the *mapper on graphs* framework comes at a cost. It appears to have a number of parameters (e.g., the lens f, the cover parameters n, and ϵ) that can lead to different skeletonizations of the data. The selection of these parameters is largely data-dependent to the point where an exploration of the parameter space may be required. In our case, we accomplished this using a small-multiple interface that allowed us to examine different topological lenses and numbers of cover elements. This interface is part of the web demo linked previously.

For cover parameter selection, there are recent works on automatic parameter selection for mapper based on statistics [22] and information criteria [26]. For topological lens selection, the process is not necessarily obvious or intuitive. A general guideline is that the chosen lens reflects the geometric or topological property of interest to a user. For instance, the PageRank lens captures node importance, where input nodes that are close to each other *and* those of similar importance are grouped together to form a *mapper graph* node. As a consequence, the corresponding skeletonization encodes the distribution of input node importance. We plan to explore (semi-)automatic parameter selection methods in the future.



(a) Input graph and three levels of aggregation (b) Annotation of the airport group types

Fig. 12: Case study of the USAIR97 dataset using the PageRank topological lens and modularity clustering. (a) The input graph is shown along with three *mapper graphs* that have $\epsilon = 0$ across different the number of cover elements. (b) An annotated version of the $n = 3$ *mapper graph* highlights the different airport groups identified.

needed to identify the best graph layout methods and create coherency in the layout for different parameter configurations. One possible approach would be to apply our previous method [38].

Flexibility vs. Scalability. To achieve scalability, we have shown that limiting topological lenses to the Fiedler vector or PageRank and limiting clustering to connected components lead to highly scalable algorithms that are practical for large graphs. We will expand upon scalable topological lenses and clustering algorithms in the future.

Relationship to Spectral Clustering. There is a connection between *mapper on graphs*, spectral clustering, and graph min-cut. The Fiedler vector l_2 can be used to bi-partition the graph G (i.e., based on $l_2(v) > 0$ or $l_2(v) \leq 0$). Such a partition could also be realized by computing the *mapper graph* with l_2 as the lens and setting $n = 2$ and $\epsilon = 0$. Such an output not only provides a generalization of spectral clustering but also preserves the connections (which form the min-cut) between the clusters (for appropriately chosen $\epsilon > 0$). Exploring such a connection in detail would be quite interesting.

ACKNOWLEDGMENTS

This work was supported in part by grants from the NSF (IIS-1513616, DBI-1661375, IIS-2316496) and DOE (DE-SC0021015).

REFERENCES

[1] J. Abello, F. van Ham, and N. Krishnan. ASK-GraphView: A large scale graph visualization system. *IEEE Trans. on Visual. and Comp. Graph.*, 12(5), 2006. 2

[2] D. Archambault, T. Munzner, and D. Auber. Topolayout: Multilevel graph layout by topological features. *IEEE Trans. on Visual. and Comp. Graph.*, 13(2), 2007. 2

[3] D. Archambault, T. Munzner, and D. Auber. GrouseFlocks: Steerable exploration of graph hierarchy space. *IEEE Trans. on Visual. and Comp. Graph.*, 14(4), 2008. 2

[4] D. Archambault, T. Munzner, and D. Auber. Tugging graphs faster: Efficiently modifying path-preserving hierarchies for browsing paths. *IEEE Trans. on Visual. and Comp. Graph.*, 17(3), 2010. 2

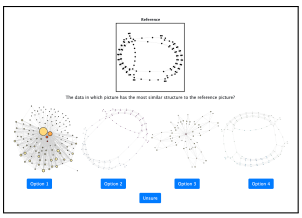
[5] D. Archambault, H. Purchase, and B. Pinaud. The readability of path-preserving clusterings of graphs. *Comp. Graph. Forum*, 29(3), 2010. 2

[6] B. Bach, N. Riche, C. Hurter, K. Marriott, and T. Dwyer. Towards unambiguous edge bundling: Investigating confluent drawings for network visualization. *IEEE Trans. on Visual. and Comp. Graph.*, 23(1), 2017. 2

[7] M. Bampasidou and T. Gentimis. Modeling collaborations with persistent homology. *arXiv preprint*, 2014. 2

[8] M. Bastian, S. Heymann, and M. Jacomy. Gephi: an open source software for exploring and manipulating networks. In *AAAI Web and Social Media*, 2009. 2

[9] V. Batagelj, F. Brandenburg, W. Didimo, G. Liotta, P. Palladino, and M. Patrignani. Visual analysis of large graphs using (x, y)-clustering and



(a) Example trial

	BARBELL	BCSSTK	BCSSTK20	BCSSTK22	BIO-CELEGANS	BIO-DISEASOME	BN-MOUSE	CALTECH	CIRCULAR LADDER	CONN. CAVEMAN	DOROGOVTSEV	ENRON-EMAIL	HIC 5K NET 6	MAP OF SCIENCE	MOVIES	RANDOM LOBSTER	RING OF CLIQUES	USAIR97	WATTS STROGATZ	**Overall**
Correct	42	37	29	42	36	38	39	40	9	20	39	36	40	42	37	37	31	34	33	661
Incorrect	2	4	11	2	5	6	4	2	26	20	5	7	2	2	6	5	8	9	10	136
Unsure	0	2	4	1	4	0	0	1	7	5	2	2	2	1	1	1	6	2	2	43
Accuracy	95%	86%	66%	93%	80%	86%	91%	93%	21%	44%	85%	80%	91%	93%	84%	86%	69%	76%	73%	**79%**
p	$< .001$	$< .001$	.024	$< .001$	$< .001$	$< .001$	$< .001$	$< .001$	1.000	.814	$< .001$	$< .001$	$< .001$	$< .001$	$< .001$	$< .001$	.008	$< .001$	.001	$< .001$

(b) Table summarizing results

Fig. 13: (a) an example trial for the human-subject evaluation and (b) the results are shown.

hybrid visualizations. *IEEE Trans. on Visual. and Comp. Graph.*, 17(11), 2010. 2

[10] S. Biasotti, D. Giorgi, M. Spagnuolo, and B. Falcidieno. Reeb graphs for shape analysis and applications. *Theoretical Comp. Sci.*, 392, 2008. 2

[11] S. Biasotti, S. Marini, M. Mortara, and G. Patane. An overview on properties and efficacy of topological skeletons in shape modelling. *Shape Modeling International*, 2003. 2

[12] V. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *J. of Statistical Mechanics*, 2008. 7

[13] C. Bodnar, C. Cangea, and P. Liò. Deep graph mapper: Seeing graphs through the neural lens. *Frontiers in Big Data*, 4, 2021. 2

[14] K. Börner, R. Klavans, M. Patek, A. Zoss, J. Biberstine, R. Light, V. Larivière, and K. Boyack. Design and update of a classification system: The ucsd map of science. *PloS One*, 7(7), 2012. 5

[15] U. Brandes, D. Delling, M. Gaertler, R. Gorke, M. Hoefer, Z. Nikoloski, and D. Wagner. On modularity clustering. *IEEE Trans. on Knowledge and Data Engineering*, 20(2), 2007. 4, 7

[16] U. Brandes, M. Gaertler, and D. Wagner. Experiments on graph clustering algorithms. In *European Symp. on Algorithms*, 2003. 1

[17] U. Brandes and C. Pich. Eigensolver methods for progressive multidimensional scaling of large data. In *Graph Drawing*, 2007. 2

[18] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems*, 30(1-7), 1998. 4

[19] A. Brown, O. Bobrowski, E. Munch, and B. Wang. Probabilistic convergence and stability of random mapper graphs. *Journal of Applied and Computational Topology (APCT)*, 5:99–140, 2021. 2

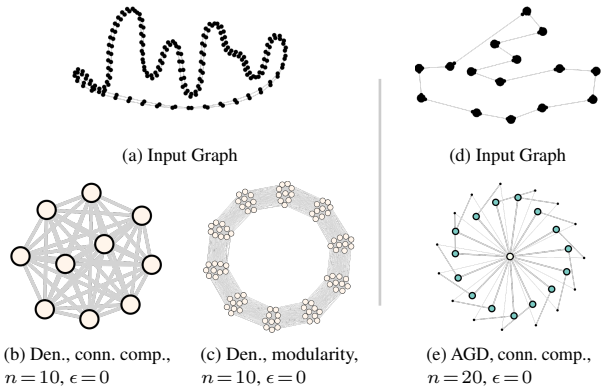
[20] T. N. Bui, S. Chaudhuri, F. T. Leighton, and M. Sipser. Graph bisection algorithms with good average case behavior. *Combinatorica*, 7(2), 1987. 1

[21] G. Carlsson. Topological pattern recognition for point cloud data. *Acta Numerica*, 23, 2014. 2

[22] M. Carriére, B. Michel, and S. Oudot. Statistical analysis and parameter selection for mapper. *J. of Machine Learning Research*, 19(12), 2018. 9

[23] M. Carriére and S. Oudot. Structure and stability of the one-dimensional mapper. *Foundations of Computational Mathematics*, 18(6), 2018. 2

[24] C. Carstens and K. Horadam. Persistent homology of collaboration networks. *Mathematical Problems in Engineering*, 2013. 2



(a) Input Graph (d) Input Graph

(b) Den., conn. comp., $n=10, \epsilon=0$ (c) Den., modularity, $n=10, \epsilon=0$ (e) AGD, conn. comp., $n=20, \epsilon=0$

Fig. 14: Challenging examples from our human-subject study: (a-c) CIRCULAR LADDER and (d-e) CONNECTED CAVEMAN.

[25] B. Cassidy, C. Rae, and V. Solo. Brain activity: Conditional dissimilarity and persistent homology. *IEEE Symp. on Biomedical Imaging*, 2015. 2

[26] N. Chalapathi, Y. Zhou, and B. Wang. Adaptive covers for mapper graphs using information criteria. *IEEE Big Data*, 2021. 9

[27] M. Chung, S. Seo, N. Adluru, and H. Vorperian. Hot spots conjecture and its application to modeling tubular structures. In *Workshop on Machine Learning in Medical Imaging*, 2011. 4

[28] G. Cordasco and L. Gargano. Community detection via semi-synchronous label propagation algorithms. In *IEEE Workshop on Business Applications of Social Network Analysis*, 2010. 4

[29] W. Cui, H. Zhou, H. Qu, P. C. Wong, and X. Li. Geometry-based edge clustering for graph visualization. *IEEE Trans. on Visual. and Comp. Graph.*, 14(6), 2008. 1, 2

[30] 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), 2012. 2

[31] T. Davis and Y. Hu. The University of Florida sparse matrix collection. *ACM Trans. on Mathematical Software*, 38(1), 2011. 5

[32] W. De Nooy, A. Mrvar, and V. Batagelj. *Exploratory social network analysis with Pajek: Revised and expanded edition for updated software*, vol. 46. Cambridge University Press, 2018. 5

[33] T. Dey, F. Memoli, and Y. Wang. Topological analysis of nerves, reeb spaces, mappers, and multiscale mappers. *arXiv preprint*, 2017. 2

[34] I. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *ACM SIGKDD Knowledge Discovery and Data Mining*, 2001. 1

[35] C. Ding, X. He, H. Zha, M. Gu, and H. Simon. A min-max cut algorithm for graph partitioning and data clustering. *IEEE Data Mining*, 2001. 3

[36] K. Dinkla, M. Westenberg, and J. van Wijk. Compressed adjacency matrices: untangling gene regulatory networks. *IEEE Trans. on Visual. and Comp. Graph.*, 18(12), 2012. 2

[37] I. Donato, G. Petri, M. Scolamiero, L. Rondoni, and F. Vaccarino. Decimation of fast states and weak nodes: topological variation via persistent homology. *European Conf. on Complex Systems*, 2012. 2

[38] B. Doppalapudi, B. Wang, and P. Rosen. Untangling force-directed layouts using persistent homology. In *Topological Data Analysis and Visualization (TopoInVis)*, 2022. 2, 7, 9

[39] C. Dunne and B. Shneiderman. Motif simplification. *ACM SIGCHI Human Factors in Computing Systems*, 2013. 2

[40] T. Dwyer, N. Riche, K. Marriott, and C. Mears. Edge compression techniques for visualization of dense directed graphs. *IEEE Trans. on Visual. and Comp. Graph.*, 19(12), 2013. 2

[41] G. Ellis and A. Dix. A taxonomy of clutter reduction for information visualisation. *IEEE Trans. on Visual. and Comp. Graph.*, 13(6), 2007. 2

[42] J. Ellson, E. Gansner, L. Koutsofios, S. North, and G. Woodhull. Graphviz— open source graph drawing tools. *Graph Drawing*, 2002. 2

[43] O. Ersoy, C. Hurter, F. Paulovich, G. Cantareiro, and A. Telea. Skeleton-based edge bundling for graph visualization. *IEEE Trans. on Visual. and Comp. Graph.*, 17(12), 2011. 1, 2

[44] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *ACM SIGKDD Knowledge Discovery and Data Mining*, 1996. 4

[45] M. Fiedler. Algebraic connectivity of graphs. *Czechoslovak Mathematical Lournal*, 23(2), 1973. 1

[46] T. Fruchterman and E. Reingold. Graph drawing by force-directed placement. *Software: Practice and Exp.*, 21(11), 1991. 2

[47] E. Gansner, Y. Hu, S. North, and C. Scheidegger. Multilevel agglomera-

tive edge bundling for visualizing large graphs. In *IEEE PacificVis*, 2011. 2

[48] E. Gansner, Y. Koren, and S. North. Graph drawing by stress majorization. In *Graph Drawing*, 2005. 2

[49] E. Gansner, Y. Koren, and S. North. Topological fisheye views for visualizing large graphs. *IEEE Trans. on Visual. and Comp. Graph.*, 11(4), 2005. 2

[50] E. Gansner, E. Koutsofios, S. North, and K.-P. Vo. A technique for drawing directed graphs. *IEEE Trans. on Software Eng.*, 19(3), 1993. 2

[51] S. Golodetz, A. Arnab, I. Voiculescu, and S. Cameron. Simplifying tuggraph using zipping algorithms. *Pattern Recognition*, 103, 2020. 2

[52] V. Grolmusz. A note on the pagerank of undirected graphs. *arXiv preprint*, 2012. 4

[53] A. Hagberg, P. Swart, and D. S Chult. Exploring network structure, dynamics, and function using networkx. Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008. 5, 7

[54] M. Hajij, B. Wang, and P. Rosen. MOG: Mapper on graphs for relationship preserving clustering. *arXiv preprint*, 2018. 2

[55] M. Hajij, B. Wang, C. Scheidegger, and P. Rosen. Visual detection of structural changes in time-varying graphs using persistent homology. *IEEE PacificVis*, 2018. 2

[56] D. Hansen, B. Shneiderman, and M. A. Smith. *Analyzing social media networks with NodeXL: Insights from a connected world*. Morgan Kaufmann, 2010. 2

[57] D. Holten and J. van Wijk. Force-directed edge bundling for graph visualization. *Comp. Graph. Forum*, 28(3), 2009. 2

[58] D. Horak, S. Maletić, and M. Rajković. Persistent homology of complex networks. *J. of Statistical Mechanics: Theory and Experiment*, 2009. 2

[59] Y. Hu. Efficient, high-quality force-directed graph drawing. *Mathematica Journal*, 10(1), 2005. 2

[60] G. Jeh and J. Widom. SimRank: a measure of structural-context similarity. In *ACM SIGKDD Knowledge Discovery and Data Mining*, 2002. 1

[61] S. Kairam, D. MacLean, M. Savva, and J. Heer. Graphprism: Compact visualization of network structure. In *Adv. Visual Interfaces*, 2012. 2

[62] M. Kamruzzaman, A. Kalyanaraman, and B. Krishnamoorthy. Detecting divergent subpopulations in phenomics data using interesting flares. In *ACM Bioinfo., Computational Biology, and Health Informatics*, 2018. 2

[63] M. Khoury, Y. Hu, S. Krishnan, and C. Scheidegger. Drawing large graphs by low-rank stress majorization. *Comp. Graph. Forum*, 31, 2012. 2

[64] V. G. Kim, Y. Lipman, X. Chen, and T. Funkhouser. Möbius transformations for global intrinsic symmetry analysis. *Comp. Graph. Forum*, 29(5), 2010. 3

[65] Y. Koren. On spectral graph drawing. In *Computing and Combinatorics*. Springer, 2003. 2

[66] Y. Koren, L. Carmel, and D. Harel. Ace: A fast multiscale eigenvectors computation for drawing huge graphs. *IEEE Symp. on Information Visualization*, 2002. 2

[67] B. Kulis, S. Basu, I. Dhillon, and R. Mooney. Semi-supervised graph clustering: a kernel approach. *Machine Learning*, 74(1), 2009. 1

[68] S. Lafon and A. B. Lee. Diffusion maps and coarse-graining: A unified framework for dimensionality reduction, graph partitioning, and data set parameterization. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 28(9), 2006. 3

[69] H. Lee, M. Chung, H. Kang, B.-N. Kim, and D. S. Lee. Computing the shape of brain networks using graph filtration and gromov-hausdorff metric. *Conference on Medical Image Computing and Computer Assisted Intervention*, 2011. 2

[70] H. Lee, M. Chung, H. Kang, B.-N. Kim, and D. S. Lee. Discriminative persistent homology of brain networks. *IEEE Symp. on Biomedical Imaging*, 2011. 2

[71] H. Lee, H. Kang, M. Chung, B.-N. Kim, and D. S. Lee. Persistent brain network homology from the perspective of dendrogram. *IEEE Trans. on Medical Imaging*, 31(12), 2012. 2

[72] H. Lee, H. Kang, M. Chung, B.-N. Kim, and D. S. Lee. Weighted functional brain network modeling via network filtration. *NIPS Workshop on Algebraic Topology and Machine Learning*, 2012. 2

[73] J. Leskovec and A. Krevl. SNAP Datasets: Stanford large network dataset collection, 2014. 5

[74] B. Lévy. Laplace-Beltrami eigenfunctions towards an algorithm that understands geometry. In *IEEE Shape Modeling and App.*, 2006. 3

[75] S. Liu, D. Maljovec, B. Wang, P.-T. Bremer, and V. Pascucci. Visualizing high-dimensional data: Advances in the past decade. *IEEE Trans. on Visual. and Comp. Graph.*, 23(3), 2017. 2

[76] P. Lum, G. Singh, A. Lehman, T. Ishkanov, M. Vejdemo-Johansson, M. Alagappan, J. Carlsson, and G. Carlsson. Extracting insights from the shape of complex data using topology. *Scientific Reports*, 3, 2013. 2

[77] U. V. Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4), 2007. 3

[78] F. McGee and J. Dingliana. An empirical study on the impact of edge bundling on user comprehension of graphs. In *Advanced Visual Interfaces*, 2012. 2

[79] E. Munch and B. Wang. Convergence between categorical representations of Reeb space and mapper. *Symp. on Comp. Geometry*, 51, 2016. 2

[80] M. Newman. The structure of scientific collaboration networks. *Nat. Academy of Sci.*, 98(2), 2001. 5

[81] M. Newman. Properties of highly clustered networks. *Physical Review E*, 68(2), 2003. 1

[82] M. Newman. Fast algorithm for detecting community structure in networks. *Physical Review E*, 69(6), 2004. 1

[83] A. Ng, M. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Info. Processing Systems*, 2002. 3

[84] M. Nicolau, A. Levine, and G. Carlsson. Topology based data analysis identifies a subgroup of breast cancers with a unique mutational profile and excellent survival. *Nat. Academy of Sci.*, 108(17), 2011. 2

[85] L. Page, S. Brin, R. Motwani, and T. Winograd. Pagerank citation ranking: Bringing order to the web. Technical report, Stanford, 1999. 4

[86] G. Petri, M. Scolamiero, I. Donato, and F. Vaccarino. Networks and cycles: A persistent homology approach to complex networks. *European Conf. on Complex Systems*, 2013. 2

[87] G. Petri, M. Scolamiero, I. Donato, and F. Vaccarino. Topological strata of weighted complex networks. *PloS One*, 8(6), 2013. 2

[88] V. Pirino, E. Riccomagno, S. Martinoia, and P. Massobrio. A topological study of repetitive co-activation networks in in vitro cortical assemblies. *Physical Biology*, 12(1), 2015. 2

[89] P. Pons and M. Latapy. Computing communities in large networks using random walks. In *Symp. on Computer and Info. Sciences*, 2005. 1

[90] L. Pretto. Analysis of web link analysis algorithms: The mathematics of ranking. In *Information Access through Search Engines and Digital Libraries*. Springer, 2008. 4

[91] U. Raghavan, R. Albert, and S. Kumara. Near linear time algorithm to detect community structures in large-scale networks. *Physical Review E*, 76(3), 2007. 4

[92] M. Reuter, S. Biasotti, D. Giorgi, G. Patané, and M. Spagnuolo. Discrete laplace–beltrami operators for shape analysis and segmentation. *Computers & Graphics*, 33(3), 2009. 3

[93] R. Rossi and N. Ahmed. The network data repository with interactive graph analytics and visualization. In *AAAI Artificial Intelligence*, vol. 29, 2015. 5

[94] S. Schaeffer. Graph clustering. *Computer Sci. Review*, 1(1), 2007. 1, 2

[95] D. Selassie, B. Heller, and J. Heer. Divided edge bundling for directional network data. *IEEE Trans. on Visual. and Comp. Graph.*, 17(12), 2011. 2

[96] D. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE Signal Processing Magazine*, 30(3), 2013. 3

[97] B. Silverman. *Density estimation for statistics and data analysis*, vol. 26. CRC press, 1986. 3

[98] G. Singh, F. Mémoli, and G. Carlsson. Topological methods for the analysis of high dimensional data sets and 3D object recognition. *Eurographics Symp. on Point-Based Graphics*, 22, 2007. 1, 2

[99] A. Suh, M. Hajij, B. Wang, C. Scheidegger, and P. Rosen. Persistent homology guided force-directed graph layouts. *IEEE Trans. on Visual. and Comp. Graph.*, 26(1), 2019. 2

[100] Y. Tian, R. Hankins, and J. Patel. Efficient aggregation for graph summarization. In *ACM SIGKDD Knowledge Discovery and Data Mining*, 2008. 1

[101] B. Y. Torres, J. Oliveira, A. T. Tate, P. Rath, K. Cumnock, and D. Schneider. Tracking resilience to infections by mapping disease space. *PLoS Biology*, 14(4), 2016. 2

[102] W. Tutte. How to draw a graph. *London Mathematical Society*, s3-13(1), 1963. 2

[103] F. van Ham and J. van Wijk. Interactive visualization of small world graphs. In *IEEE Symp. on Information Visualization*, 2004. 2

[104] C. Vehlow, F. Beck, and D. Weiskopf. Visualizing group structures in

graphs: A survey. *Comp. Graph. Forum*, 36(6), 2017. 2

[105] T. von Landesberger, A. Kuijper, T. Schreck, J. Kohlhammer, J. van Wijk, J.-D. Fekete, and D. Fellner. Visual analysis of large graphs: State-of-the-art and future research challenges. *Comp. Graph. Forum*, 30(6), 2011. 2

[106] S. White and P. Smyth. A spectral clustering approach to finding communities in graphs. In *SIAM Data Mining*, 2005. 1

[107] P. Wills and F. Meyer. Metrics for graph comparison: a practitioner's guide. *PloS One*, 15(2), 2020. 6

[108] Y. Zhou, M. Kamruzzaman, P. Schnable, B. Krishnamoorthy, A. Kalyanaraman, and B. Wang. Pheno-Mapper: an interactive toolbox for the visual exploration of phenomics data. *ACM Bioinformatics, Computational Biology, and Health Informatics*, 2021. 2

[109] Y. Zhou, A. Rathore, E. Purvine, and B. Wang. Topological simplifications of hypergraphs. *IEEE Trans. on Visual. and Comp. Graph.*, 2022. 2

[110] M. Zitnik, R. Sosič, S. Maheshwari, and J. Leskovec. BioSNAP Datasets: Stanford biomedical network dataset collection, 2018. 5