2022 Topological Data Analysis and Visualization (TopoInVis) | 978-1-6654-9354-3/22/$31.00 ©2022 IEEE | DOI: 10.1109/TopoInVis57755.2022.00015

Untangling Force-Directed Layouts Using Persistent Homology

Bhavana Doppalapudi, Bei Wang, and Paul Rosen

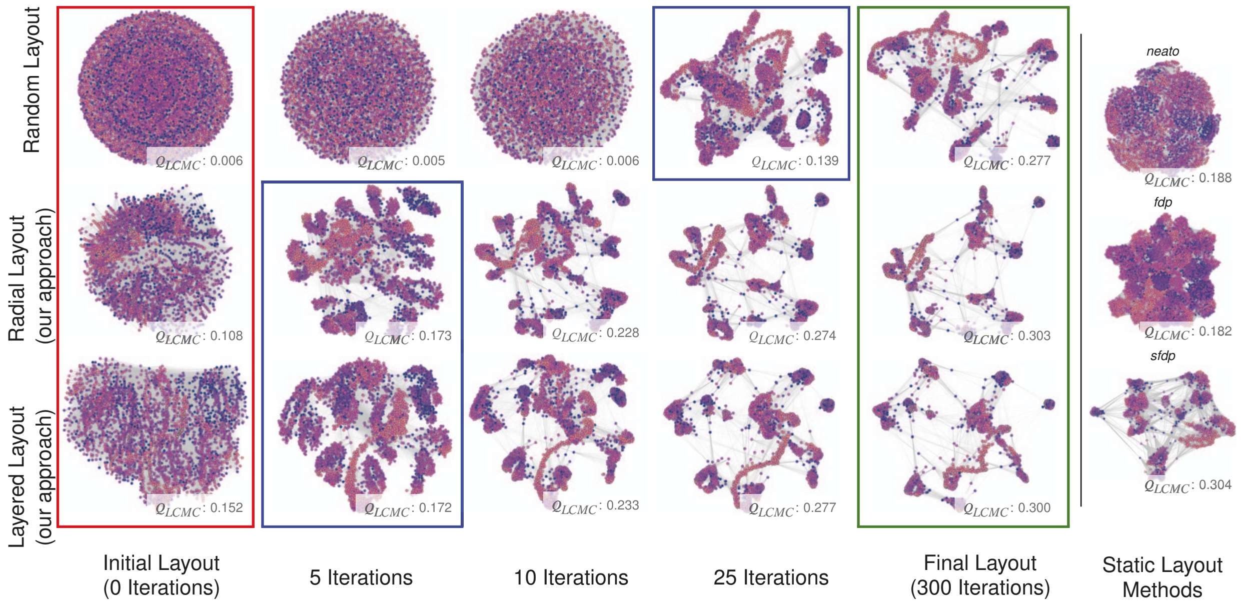


Fig. 1: Our approach in using persistent homology to untangle force-directed layouts has two main functionalities. First, as demonstrated via the HIC_1K_NET dataset, we use persistent homology to formulate an initial graph layout (column 1) that improves both the convergence rate of the graph layout and the final layout quality. Second, our approach comes with interactive capabilities, as illustrated in Fig. 2. We use the local continuity meta criterion (Q_{LCMC}) for layout evaluation (larger is better). In terms of convergence, our approach (column 2, rows 2 and 3) shows the formation of major graph structures as early as 5 iterations, whereas the standard approach (column 4, row 1) takes 25+ iterations. In terms of final layout quality, the Q_{LCMC} scores for the final layout (column 5) show that our approach significantly exceeds the standard one. Three static layout methods, *neato*, *fdp*, and *sfdp*, are also included for comparison (column 6), with only *sfdp* (column 6, row 3) showing a comparable Q_{LCMC} score.

Abstract—Force-directed layouts belong to a popular class of methods used to position nodes in a node-link diagram. However, they typically lack direct consideration of global structures, which can result in visual clutter and the overlap of unrelated structures. In this paper, we use the principles of persistent homology to untangle force-directed layouts thus mitigating these issues. First, we devise a new method to use 0-dimensional persistent homology to efficiently generate an initial graph layout. The approach results in faster convergence and better quality graph layouts. Second, we provide a new definition and an efficient algorithm for 1-dimensional persistent homology features (i.e., tunnels/cycles) on graphs. We provide users the ability to interact with the 1-dimensional features by highlighting them and adding cycle-emphasizing forces to the layout. Finally, we evaluate our approach with 32 synthetic and real-world graphs by computing various metrics, e.g., co-ranking, edge crossing, etc., to demonstrate the efficacy of our proposed method.

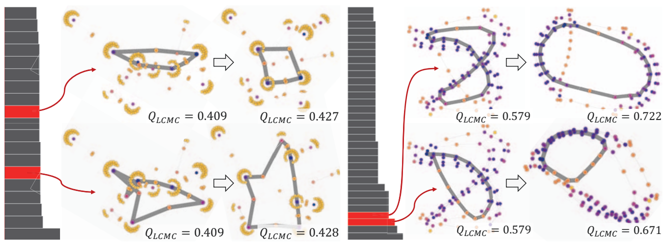
Index Terms—Force-directed layout, persistent homology, graph clustering, graph cycles

1 Introduction

Force-directed layouts remain one of the most popular methods for drawing graphs. Their popularity stems from several desirable qualities: generally, they are simple to implement, they are fast for small graphs, they produce aesthetically pleasing layouts, and their iterative algorithms make progressive visualization and interaction natural. Nevertheless, force-directed layouts also suffer from numerous limitations, including poor initialization and over-constraint, leading to poor local minima and limited robustness to noise.

This paper addresses two of these limitations by utilizing and highlighting important topological features of the graph. First, force-directed layouts are strongly influenced by the initial layout of graph nodes, which is often generated randomly. After the initialization, successive application of the forces among nodes causes the layout to settle in a locally minimal energy state, which *hopefully* shows the graph's topological structure. Unfortunately, the random initial layout does not consider the global topology, which potentially slows convergence and can lead to unrelated structures overlapping in the visualization. Second, since force-directed layouts are over-constrained systems, even without the overlap of unrelated structures, certain topological features have their shape distorted, tangled, or hidden by noise (i.e., low weight edges) making it difficult to visually identify topological features.

- *B. Doppalapudi was with the University of South Florida, Tampa, FL, 33610. E-mail: bdoppalapudi@usf.edu*
- *B. Wang was with the University of Utah, Salt Lake City, UT, 84112. E-mail: beiwang@sci.utah.edu*
- *P. Rosen was with the University of Utah, Salt Lake City, UT, 84112. E-mail: prosen@sci.utah.edu*



(a) BN-MOUSE-VISUAL-CORTEX-2 (b) BCSSTK

Fig. 2: Our second untangling functionality is to *interactively* untangle persistent homology cycle features. Each cycle is represented by a bar in the barcode (left), which is used to highlight cycles by mouse over (middle) and apply an elliptical force by click (right). In this example, the elliptical forces are used to regularize deformed cycles (a, top and bottom), untangle twisted cycles (b, top), and disentangle a cycle covered by unrelated structures (b, bottom). The Q_{LCMC} scores show that the elliptical forces also improve the overall graph layout.

We address these limitations by using techniques from persistent homology to provide better initialization and support interactive exploration of the graph topology. We are not the first to consider the use of persistent homology in force-directed layouts. In their work, Suh et al. [63] used 0-dimensional persistent homology features (i.e., components) to enable new forms of interaction with a force-directed layout. Importantly, their work did not address the problem of initial graph layouts. Furthermore, their work did not consider 1-dimension persistent homology features, i.e., tunnels/cycles, which are an important topological features frequently present in graphs. Our work advances theirs by: (1) utilizing persistent homology as a framework for efficiently generating good quality initial graph layouts (see Fig. 1); (2) extending the algorithm for extracting topological features from graphs to efficiently extract 1-dimensional features; and (3) providing new interactions with the 1-dimensional persistent homology features of the graph (see Fig. 2).

A natural question at this point would be, why persistent homology in this context? First, it has a strong mathematical foundation, making the technique technically sound and robust to noise [76]. Second, persistent homology computation, being slow for general point cloud data [53], can be made very efficient within the context we present. Finally, instead of considering only the node-link topology, persistent homology allows for a natural way of evaluating the interaction of the node-link topology with a function (i.e., a weight) defined per link. These advantages make it a valuable companion to existing graph analysis and drawing tools.

2 Prior Work

Graph visualization as a research area has received significant attention. Tamassia [65] provided a broad overview of the state of the art. Graphs are often represented as node-link diagrams, adjacency matrices, or hybrid representations. Ghoniem et al. [32] showed that matrix-based representations are generally better than node-link diagrams for node counts greater than 20, but node-link diagrams outperform for specific tasks (e.g., pathfinding) and are aesthetically pleasing. Archambault et al. [4] assessed how graph representations affected readability and showed that only clustering could be efficiently performed on larger graphs. A separate study by Saket et al. [60] concluded that node-link diagrams are superior for topology and network-related analysis compared to adjacency matrices.

2.1 Node-Link Diagrams

Node-link diagrams represent the data with nodes for entities and links for their pairwise relations. There are multiple general-purpose layout methods to position nodes with frequently used ones including force-directed, constraint-based, layered, algebraic, and multiscale layouts.

Force-Directed Layouts. Force-directed (or force-based) layouts consider graphs as mechanical systems and apply forces to the nodes. In general, repulsive forces, similar to those of electrically charged particles, exist between all the vertices, and attractive forces, similar to spring-like forces, exist between connected vertices or between neighbors. The Eades model [24] was the first to apply spring forces on the initial layout to achieve a minimal energy position. Later, Fruchterman and Reingold [28] modified the Eades model to achieve a system that distributes the vertices evenly, and has uniform edge lengths and symmetry. Kamada and Kawai [45] developed another variant on Eades' work. Instead of just applying attractive forces between neighboring vertices, they applied the concept of ideal distance, which is proportional to the length of the shortest path. Although computational costs are high for this method, speed-ups have been achieved using heuristics [27] and the GPU [33]. Meidiana et al. [50] presented a sublinear time model that pairs a radial tree drawing of a breadth first spanning tree with random sampling of repulsive forces. While their approach has technical similarities with our initial layout approach, there are important implementation differences and theoretical guarantees that come from our use of 0-dimensional persistent homology.

Constraint-Based Layouts. Constraint-based layouts are a more sophisticated version of force-directed layouts. Dwyer et al. [23] archived a high-quality, topology-preserving visualization by implementing a constraint-based layout for a detailed view and a force-directed layout for the overview. Archambault et al. [3] proposed the *TopoLayout* algorithm that dynamically adapts the graph layout method based upon the topology detected within subgraphs.

Layered Layouts. Layered layouts, in general, are used for directed graph layouts. Sugiyama et al. [62] used a 4-phase approach to layout graphs: (1) removing cycles, (2) assigning nodes to layers, (3) reducing edge crossings, and (4) assigning coordinates to nodes. Bachmaier et al. [6] proposed to visualize directed cyclic graphs by skipping the cycle removal step.

Algebraic Methods. Koren et al. [46] developed the *Algebraic Multigrid Method* (ACE) algorithm that minimizes quadratic energy. Harel and Koren introduced *High-Dimensional Embedding* (HDE) [41] that projects a high-dimensional representation of a graph with PCA.

Multiscale Layouts. Multiscale layouts start with a coarse layout and refine it in phases. Hachul and Jünger [37] proposed *Fast Multipole Multilevel Method* (FM^3), a force-directed method that incorporates a multiscale approach in a system to calculate repulsive forces in rapidly evolving potential fields. By comparing various algorithms, Hachul and Jünger [38] showed that multiscale methods, including FM^3, ACE, and HDE, were significantly faster than regular force-directed layouts. They also found that FM^3 produced the best quality graphs in the group.

Node Congestion. Node co-location is a challenging problem, particularly in multiscale layouts [68]. Space-filling curves have been used to avoid node co-location [51]. However, the approach works only for datasets with clear clustering, and for dense graphs, the visualizations are not of good quality or aesthetically pleasing. Gansner and North [31] proposed to improve the force-directed layout by moving the overlapping nodes within cells of a constructed Voronoi diagram. By selecting good starting positions for nodes, Gajer et al. [30] developed a multiscale approach that improved computation time and better preserved the graph's structure. Adai et al. [2] introduced the *Large Graph Layout* (LGL) algorithm, which uses a minimum spanning tree to guide the force-directed iterative layout to visualize large protein map networks. However, they did not consider datasets with different characteristics. Dunne and Shneiderman [22] proposed to use motifs for node and edge bundling. Their technique replaced common graph patterns of nodes and links with intuitive glyphs. They showed that the approach required less screen space and effort, while preserving the underlying relations. However, the glyphs required additional learning from users, and charts with many large glyphs added clutter to the display and increased the possibility of overlap.

Edge Congestion. Node-link diagrams frequently suffer from edge crossings. Carpendale et al. [12] proposed displacing edges running through the area of interest. However, certain questions were left unanswered (e.g., the amount of edge displacement to use). For graphs without hierarchy, Holten and Van Wijk proposed a self-organizing bundling method, where edges act as flexible springs attracting each other [42]. *ASK-Graph* [1] addresses the issues for highly dense graphs with node counts approaching 200k. Bach et al. [5] proposed to use confluent drawings (CDs) for edge bundling based on network connectivity, which showed some promising results. Nevertheless, CDs worked only for sparse graphs where node counts were less than 20 and the edge density was less than 50. Such an approach also showed low participant confidence, indicating that CDs require significant learning and may be misleading. Zinsmaier et al. [74] proposed a level-of-detail technique that performs density-based nodes aggregation and edge accumulation.

Interaction. Research into interactive visualization has been done to assist with efficient data explorations. Commonly used interaction techniques for graphs include panning and zooming [67] and fisheye views [29,61,69] that focus on areas of interest.

2.2 Persistent Homology

Persistent homology studies the topological features of data that persist across multiple scales. Weinberger gave a brief explanation in his work titled "What is ... persistent homology?" [72], while Harer and Edelsbrunner [25] detailed the concept and history of persistent homology. Persistent homology has shown great promise to assist in the analysis of complex graphs due to the types of features it extracts and its ability to differentiate signal from noise [18,43,54,55,71]. For example, Rieck et al. [58] used persistent homology to track the evolution of clique communities across different edge weight thresholds. Persistent homology has also led to notable results in the study of brain networks [13,48,49] and social networks [7,40,43]. Although persistent homology has been used mainly for analysis tasks within these prior methods, it has not been used to improve the visualization of graphs.

Recently, Suh et al. [63] used 0-dimensional persistent homology features to create a persistence barcode visualization, which was then used to manipulate a force-directed graph layout. Their framework was limited to extracting 0-dimensional topological features and utilizing those features for interactive manipulation of the graph. Our work extends and complements the work of Suh et al. by utilizing the 0-dimensional features to preprocess the initial layout of graphs, thus improving the rate of convergence and quality of layouts, and by providing an algorithm to efficiently extract, interactively highlight, and manipulate the graph layout with 1-dimensional topological features of a graph.

3 Methods

We first describe how the persistent homology information is extracted from an input graph (Sect. 3.1). We then describe the use of this information for building fast initial graph layouts (Sect. 3.2) and for highlighting important structures in the visualization (Sect. 3.3).

The input is an undirected graph $G = (V, E)$ equipped with an edge weight $w : E \to \mathbb{R}$. w can be any real function that quantifies the edge importance. Recall the Jaccard index between a pair of sets A and B is defined to be $J(A,B) = \frac{|A \cap B|}{|A \cup B|}$. In this paper, if w is not known *a priori*, $w(e)$ for an edge e is assigned the Jaccard index between the 1-neighborhood of its nodes, as was also done in [63].

3.1 Persistent Homology of a Graph

We describe a novel approach to extract persistent homology features from a graph, leaving the discussions of the general theory to prior works (e.g., [25]). Previous approaches (e.g., [40]) have relied upon

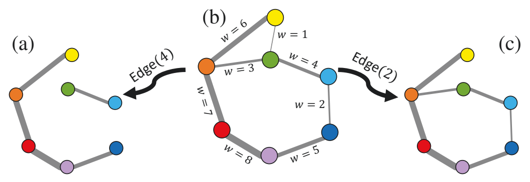


Fig. 3: Two Edge complexes computed on a weighted graph (b), capturing different homology structures. (a) Edge(4) shows two H_0 components and no H_1 cycle. (c) Meanwhile, Edge(2) shows one H_0 component and one H_1 cycle.

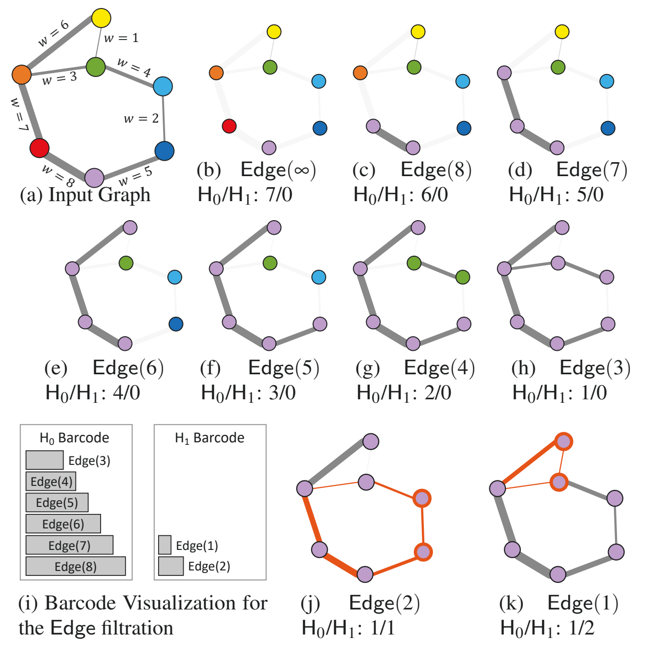


Fig. 4: (a) Illustrating an Edge filtration. (b) The filtration begins with seven components (denoted by color), one per node, and no cycles. (c-h) As the filtration continues, the H_0 components and H_1 cycles are extracted at each step. (j) When cycles form, they are extracted, shown in red. (k) By the end, one component and two cycles remain. (i) The resulting features are represented in a barcode visualization.

mapping a graph to a metric space, computing a Vietoris-Rips complex, and extracting its 0-dimensional and 1-dimensional persistent homology as topological features. However, these approaches may be costly, $O\left((|V|+|E|)^3\right)$ in the worse case, making them impractical on larger graphs. Furthermore, persistent homology identifies a class of cycles and while identifying the existence of such a class is well defined, determining which nodes specifically contribute to the cycle in the context of graphs is ambiguous [17]. In the following section, we provide an alternate strategy that resolves both of these issues.

Homology deals with the topological features of a space. In particular, given a space $\mathbb{X}$, we are interested in extracting the *0-dimensional*, $H_0(\mathbb{X})$, and *1-dimensional*, $H_1(\mathbb{X})$, homology groups of $\mathbb{X}$, which are the connected components and tunnels/cycles of the space, respectively.

To identify the homology of a graph, we begin by describing the Edge complex of a graph. Given a threshold t, for each edge e_i in G with a weight w_i, the Edge complex is $\text{Edge}(t) = \{e_i \mid w_i \geq t\}$. In other words, the Edge complex is the set of all edges whose weight is greater than or equal to the given threshold. For example, Fig. 3b shows Edge(4) and Edge(2) of the graph, in Fig. 3a and 3c, respectively.

From an Edge complex, its connected components (H_0) and cycles (H_1) can be efficiently extracted by a process that will be discussed in the forthcoming sections. However, extracting the homology of the graph from a single Edge complex may fail to capture homology visible at different thresholds (e.g., see Fig. 3) and, therefore, requires careful selection of the threshold t.

Instead of selecting a single threshold t, we extract H_0 and H_1 features of the graph across all thresholds using a multiscale notion of homology, called *persistent homology*. Persistent homology is calculated by extracting a sequence of Edge complexes, referred to as a *filtration*. We consider a finite sequence of decreasing thresholds, $\infty = t_0 \geq t_1 \geq \cdots \geq t_m = 0$. A sequence of Edge complexes, known as an Edge filtration, is calculated and connected by inclusions,

$$\text{Edge}(t_0) \to \text{Edge}(t_1) \to \cdots \to \text{Edge}(t_m).$$

In other words, the Edge complexes are subsets of one another, $\text{Edge}(t_i) \subseteq \text{Edge}(t_{i+1})$ for $0 \leq i \leq m-1$. The Edge filtration can also be described as the upper-star filtration of the graph. H_0 and H_1 features are tracked across the various Edge complexes in the filtration,

$$\mathsf{H}(\mathsf{Edge}(t_0)) \rightarrow \mathsf{H}(\mathsf{Edge}(t_1)) \rightarrow \cdots \rightarrow \mathsf{H}(\mathsf{Edge}(t_m)).$$

In the example filtration in Fig. 4, as the threshold decreases, H_0 component features merge. The merging of two H_0 features causes one feature to disappear in what is known as a *death* event while the other feature continues to live. Consider the merging of the green and purple components in Fig. 4g. In Fig. 4h, the green component has died while the purple continues to live. In contrast, as the threshold decreases, new H_1 cycle features appear in what are known as *birth* events. Note the creation of cycles at $\mathsf{Edge}(2)$ and $\mathsf{Edge}(1)$ in Fig. 4j and 4k, respectively. The birth and death events represent critical values that define the importance of a feature.

3.1.1 Efficient Identification of H_0 Connected Components

To calculate H_0 features of a graph, Suh et al. [63] calculated the minimum spanning tree of a metric space representation of the graph by transforming edge weights into distances, e.g., $d(A,B) = 1/w_{AB}$ (i.e., larger weights having smaller distances), which is inefficient on larger graphs, taking $O(|V|^2 \log |V|)$.

Our choice of the Edge filtration is a very specific one designed to capture features and increase efficiency. Using the Edge filtration, the H_0 information for the graph can be obtained by calculating the *maximal spanning tree* of the graph, which is the spanning tree with edge weights greater than or equal to every other possible spanning tree. In calculating the maximal spanning tree of the graph, as edges are added to the tree, each edge e_i represents an H_0 death event at w_i (i.e., merging of two connected components). We calculate the maximal spanning tree using Kruskal's algorithm [47], selecting the largest weight edge instead of the smallest. The algorithm has a worst-case time complexity of $O(|E| \log |E|)$. For non-negative weights, the resulting maximal spanning tree captures exactly the same structure as the prior minimal spanning tree of the metric space approach, only more efficiently. In addition, our maximal spanning tree approach captures meaningful features for negative and zero weight edges.

3.1.2 Efficient Identification of H_1 Cycles

In general persistent homology calculations, both detecting the existence of H_1 features and extracting a representative cycle are computationally expensive, roughly $O(|V|^3)$ [17]. The use of the Edge filtration enables both detecting and extracting the H_1 features much more efficiently within the limited context of graphs. To do this, we begin with an interesting observation in Theorem 1.

Theorem 1. *Given any spanning tree S of a connected graph G, inserting any additional graph edge into S creates a cycle.*

Proof. Since S is a tree, it is acyclic, and any two nodes have a unique simple path between them. Therefore, if an edge is added between any two nodes in S, those nodes will now have two non-overlapping paths between them. Concatenating the edges of the two paths will create closed trail, i.e., a cycle, between them. □

As it turns out, this property enables efficient extraction of H_1 features with the Edge filtration. As the maximal spanning tree is calculated, an edge e_i that would be excluded from the spanning tree signifies the existence of an H_1 cycle feature with a birth at w_i[1].

To extract the H_1 cycle paths themselves, the unweighted shortest path is calculated between the endpoints of each edge, e_i, in the associated Edge complex, $\mathsf{Edge}(w_i)$. The shortest path is computed using Dijkstra's algorithm with worst-case time complexity $O((|V| + |E|) \log |V|)$. In practice, paths are short and generally fast to compute. Nevertheless, the shortest path needs to be calculated for every H_1 cycle. Therefore, in practice, we defer calculating the complete cycle paths until the visualization needs the information. To further reduce the number of H_1 features considered, cycles of length three are considered to be trivial and discarded[2].

[1]Since 2-simplices (i.e., triangles) are not used, we do not track cycle death.
[2]These are found by comparing the 1-neighborhood of the edge nodes.

While this approach will extract all H_1 features of the Edge complex, it will not extract all cycles of the graph. Our H_1 features are a specific type of cycle, where no chord within the cycle has a weight greater than or equal to the weight of all edges of the cycle. This type of cycle has a strong theoretical basis that is useful for many analysis tasks, but it may not be relevant for all such tasks. As we will show in our evaluation (see Sect. 5.2), these cycles are useful for many graphs. Nevertheless, adapting our approach to extracting other representative cycle types would further extend the utility of the approach.

3.2 Using H_0 Features to Untangle Initial Graph Layouts

Recent works (e.g., [19, 63]) have demonstrated the value of using H_0 information in generating high-quality layouts of graphs and high-dimensional data, respectively. In contrast, we focus on quickly producing a good-quality layout that reflects the most important structures of the graph, as defined by persistent homology. We then utilize a D3.js's force-directed layout capabilities [11] to optimize the final layout.

Our algorithm works by laying out the graph using the maximal spanning tree. Inspired by early works on tidy tree drawing [57, 64, 73], our approach has two main steps, with a focus on simplicity and efficiency. First, we generate an abstract layout of the maximal spanning tree that determines the distribution of space to nodes and subtrees of the graph. Then, we embed the tree into the drawing canvas using either a layered or radial scheme.

3.2.1 Abstract Layout

The first phase of the algorithm forms an abstract layout of the tree. The algorithm begins by selecting a node at random to serve as the root of the tree[3]. The children of the selected root are then laid out hierarchically. The algorithm recursively processes subtrees, subdividing the available space until all nodes have been visited. The available space is divided between children at each level based on the number of nodes in their respective subtrees. For example, in Fig. 5b, the orange node is selected as the root. The leftmost subtree is allocated more space since it contains more nodes.

3.2.2 Graph Embedding

In the second part of the algorithm, the abstract layout is used to embed nodes into the drawing canvas using either a layered or radial layout.

Layered Layout. The first version of our layout algorithm maps the abstract layout directly to the available drawing canvas in a layered tree visualization. Specifically, the horizontal space on the canvas is mapped to the width of the tree in the abstract layout, and the vertical space is mapped to the height of the tree in the abstract layout. For example, in Fig. 5c, the abstract layout from Fig. 5b is mapped to the available drawing canvas.

[3]We tested several other strategies, e.g., finding the most central node or the node with the most children. However, improvements were minimal, sometimes with a high additional cost over a randomly chosen node.

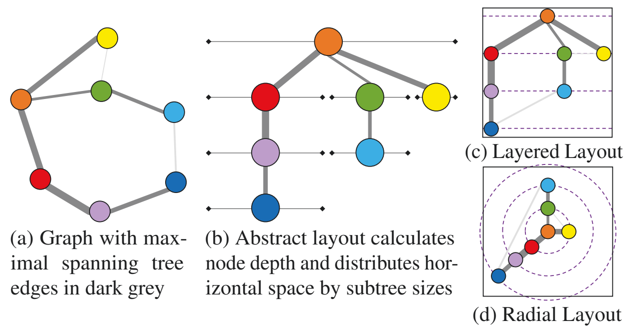

(a) Graph with maximal spanning tree edges in dark grey
(b) Abstract layout calculates node depth and distributes horizontal space by subtree sizes
(c) Layered Layout
(d) Radial Layout

Fig. 5: An illustration of the initial layout schemes. The input graph (a) first has an abstract layout formed in (b) and is then mapped into a layered layout (c) or a radial layout (d). After the initial layout, any force-directed layout can be used.

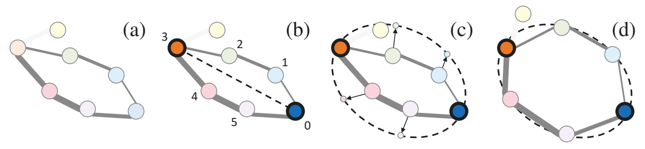


Fig. 6: Illustration of the elliptical force applied to an H_1 feature. (a) A cycle (dark gray) is selected in the barcode. (b) The two nodes in the cycle with the greatest euclidean distance in the visualization identify the major axis of the ellipse, whereas the minor axis diameter is calculated by a user-specified aspect ratio. (c) The nodes on the cycle are parameterized (i.e., ordered) and their target locations on the ellipse are identified. (d) Forces are applied to the nodes on the cycle to move them toward their target locations.

Radial Layout. The second version adopts a radial layout for the tree. The width of the abstract tree is mapped to an angle in the unit circle, and each layer of the tree occupies an increasing radius. For example, in Fig. 5d, the abstract layout from Fig. 5b is mapped to polar coordinates in the drawing canvas.

For either layered or radial layout, after the initial layout is formulated, a standard force-directed layout is applied to the entire graph.

3.3 Interactive Untangling with Persistent Homology

3.3.1 Visualization of Persistent Homology Features

We visualize the H_0 and H_1 persistent homology using a visualization based upon a persistence barcode (see Fig. 2), a standard tool of persistent homology. For this visualization, a barcode is associated with a set of H_0 or H_1 features. For each barcode, a bar represents a single topological feature. Its length is proportional to the death or birth time of the associated H_0 or H_1 features, respectively.

3.3.2 Interacting with H_0 Components

Similar to the H_0 interactions in [63], when H_0 bars are selected in the barcode, a strong attractive force is created (i.e., a spring-like force) between the nodes of the associated edge from the spanning tree. Our selection offers a filtering slider (see demo[4]) to select multiple features simultaneously.

3.3.3 Interacting with H_1 Cycles

For interacting with H_1 cycles, we offer two modalities. The first is a highlighting modality. As the user's mouse goes over the bar for a given cycle, that cycle is extracted and highlighted in the graph. Fig. 2 shows two examples, each highlighting two cycles.

The second modality, triggered when a user clicks a feature in the barcode, uses information about the cycle to add a new elliptical force to the cycle nodes in the force-directed layout. The approach (see Fig. 6 for an illustration) first takes the nodes of the cycle and identifies the two nodes with the largest Euclidean distance from one another in the visualization. Those nodes serve as the major axis of the ellipse and determine the diameter of the major axis. The minor axis diameter is determined by a user-selectable aspect ratio. Once the elliptical shape is calculated, the cycle nodes are parameterized, i.e., ordered around the ellipse, to select a target location. The ordering step is quite important, as it enables powerful modifications, e.g., untangling cycles. Finally, a strong force is added to attract the nodes to their target locations. However, due to the over-constraint of force-directed layouts, this new force does not guarantee nodes will end up on the ellipse. Fig. 2 shows examples of imposing the elliptical shape on the cycles and examples of the forces untangling cycles in the graph.

4 Evaluation

To demonstrate the efficacy of our approach, we provide a two-phase evaluation. In Sect. 5.1, we first evaluate our graph initialization approach using H_0 persistent homology in terms of layout quality and rate of convergence. Second, in Sect. 5.2, we evaluate the layout quality of using H_1 persistent homology to modify the forces of a force-directed layout. For all comparisons, we primarily compare to the state-of-the-practice force-directed layout provided by D3.js [11], which uses a random initialization. Furthermore, in Sect. 5.1.4, we compare to static graph visualizations, including the *neato* [45], *fdp* [24], and *sfpd* [44] algorithms coming from Graphviz [26].

4.1 Implementation

We have implemented our approach in JavaScript and D3.js v5 using the base implementation of D3.js force-directed layout with all standard settings. To initialize the layout, our code provides xy-coordinates to all nodes before the D3.js force-directed layout simulation takes control of the data. Modifications to the graph forces are done by adding new forces to the D3.js layout simulation. All experiments use the default D3.js stopping criteria for computation. A demo version of our approach is at *https://usfdatavisualization.github.io/UntangleFDL/*, and our source code is available at *https://github.com/USFDataVisualization/UntangleFDL/*.

4.2 Datasets

We have tested 32 datasets that include a mix of synthetic and real-world datasets, acquired from sources including the Network Repository [59], NetworkX [39], BioSNAP [75], and the UF Sparse Matrix Collection [15]. The graphs are evenly divided into 16 dense and 16 sparse graphs, based upon their average node eccentricity (ECC)[5]. A summary of graphs found in the paper can be seen in Table 1 and Table 2. Further, as a practical matter, interactivity of graph visualizations begins to degrade at ~ 1000 nodes in D3.js. Therefore, we differentiate larger graphs as those where $|N| > 1000$. Graphs not in the paper can be found in a comprehensive table of results included in our supplemental materials and in our demo.

All graphs are colored using the D3.js plasma color map (0 ▬ 1) of their normalized node valence. The only exception is the MAP OF SCIENCE dataset (see Fig. 9e), which is colored using a categorical color map.

4.3 Evaluation Metrics

Layout algorithms are often optimized considering aesthetic criteria. Purchase [56] worked on various aesthetic criteria of importance and priority and showed that minimizing the number of edge crossings serves as critical aesthetic quality. Beck et al. [9] defined several aesthetic criteria that ease designing, comparing, and evaluating different dynamic visualizations, including general aesthetic criteria, dynamic aesthetic criteria, and aesthetic scalability criteria. We use several criteria, including time (T_*), convergence (C_*), and layout quality (Q_*).

Our main goal is to measure whether global structures overlap with one another. To identify those overlaps, the primary measure we consider is that of co-ranking. Co-ranking compares the k-neighborhoods of a high-dimensional space, in our case defined by the unweighted shortest path distance in the graph, with a low-dimensional embedding, the Euclidean distance between nodes on the image. We use several meta-criteria on the co-ranking.

Local Continuity Meta Criterion (Q_{LCMC}/C_{LCMC}) measures the ranked order overlap of k-neighborhoods for a range $[1,k]$ and averages them [14]. To ease comparisons, we fix $k = 20$. Q_{LCMC} is normalized such that $Q_{LCMC} \in [-1,1]$, where larger is better and negative values imply opposite ordering. We also utilize the convergence C_{LCMC}, which is the iteration number when Q_{LCMC} is within 0.01 of the final value (after 300 iterations of force calculations).

Trustworthiness (Q_{trust}) and **Continuity** (Q_{cont}) co-ranking meta criteria [66], whose conclusions parallel LCMC, are also provided.

We next consider three measures of performance that quantify the processing time.

[4]Demo at *https://usfdatavisualization.github.io/UntangleFDL/*.

[5]Eccentricity is the maximum shortest path distance from a given node.

Table 1: Table of *dense* datasets. See Sect. 4.3 for details about metrics. Our discussion focuses on LCMC metrics (in blue). In these cells, bold indicates a smaller time for T_{LCMC}, a lower iteration count to convergence for C_{LCMC}, or a value 0.005 larger for Q_{LCMC}.

Dataset	$\|V\|$	$\|E\|$	Avg ECC	Layout	T_{IT}	T_{AIT}	T_{LCMC}	C_{LCMC}	Q_{LCMC}	Q_{trust}	Q_{conv}	Q_{EC}	Q_{CA}	Q_{MAR}	Source	Description
AVES SPARROW SOCIAL	31	211	3.5	Random	13.5 ms	10.9 ms	1.19 s	108	0.350	0.885	0.880	0.764	0.757	0.081	Network Repository	Nodes represents individual free range birds, and edges are a proximity-based association index.
				Layered	14.1 ms	11.3 ms	796 ms	**69**	**0.416**	0.928	0.926	0.792	0.766	0.140		
BARBASI-ALBERT (50,40)	50	400	2.0	Random	18.7 ms	10.8 ms	1.53 s	140	0.145	0.657	0.677	0.546	0.714	0.077	NetworkX	A graph that satisifies Barbasi-Albert preferential attachment model [8].
				Radial	17.8 ms	11.8 ms	29.6 ms	**1**	**0.186**	0.691	0.702	0.575	0.725	0.029		
BIO-CELEGANS	453	2025	5.2	Random	110 ms	47.1 ms	3.08 s	63	0.219	0.867	0.761	0.919	0.719	0.196	Network Repository	A metabolic network where substrates are nodes and edges are metabolic reactions between them.
				Layered	102 ms	47.3 ms	1.09 s	**21**	**0.229**	0.867	0.776	0.915	0.720	0.181		
BN-MOUSE-VISUAL-CORTEX-2	193	214	6.4	Random	16.6 ms	14.0 ms	282 ms	19	0.409	0.958	0.960	0.997	0.746	0.931	Network Repository	Mouse brain network where nodes are locations and edges are unweighted fiber tracks between them.
				Layered	17.2 ms	14.2 ms	201 ms	**13**	**0.425**	0.967	0.963	0.999	0.837	0.930		
CHORDAL CYCLE (90)	90	180	5.0	Random	11.9 ms	13.5 ms	1.19 s	87	0.335	0.897	0.818	0.938	0.738	0.467	NetworkX	Graph where all cylces contain a chord.
				Radial	14.6 ms	13.6 ms	1.02 s	**74**	**0.362**	0.905	0.840	0.940	0.735	0.477		
DAVIS SOUTHERN WOMEN	32	89	3.7	Random	6.3 ms	14.8 ms	1.53 s	103	**0.382**	0.921	0.903	0.880	0.736	0.244	NetworkX	A graph of observered attendance at 14 social events by 18 southern women in 1930's.
				Radial	7.3 ms	14.5 ms	528 ms	**36**	0.371	0.917	0.903	0.879	0.749	0.180		
DOLPHIN SOCIAL	62	159	6.5	Random	10.4 ms	12.8 ms	1.38 s	107	0.474	0.948	0.947	0.955	0.748	0.397	Network Repository	A social interaction network between dolphins.
				Radial	10.4 ms	13.3 ms	1.02 s	**76**	**0.486**	0.948	0.944	0.947	0.767	0.347		
DOROGOVTSEV-GOLTSEV-MENDES (5)	123	243	4.3	Random	14.4 ms	12.5 ms	1.31 s	104	0.377	0.930	0.860	0.986	0.722	0.534	NetworkX	A graph that satisifies Dorogovtsev and Mendes algorithm [20].
				Radial	31.2 ms	12.7 ms	730 ms	**55**	**0.446**	0.960	0.912	0.996	0.734	0.563		
DUPLICATE DIVERGENCE	50	99	4.6	Random	7.7 ms	14.1 ms	1.29 s	91	0.415	0.925	0.913	0.907	0.747	0.483	NetworkX	Starting with a small graph, nodes and some edges repeatedly duplicated.
				Layered	8.2 ms	14.1 ms	318 ms	**22**	0.414	0.919	0.916	0.931	0.733	0.488		
ENRON EMAIL	143	623	6.1	Random	28.1 ms	17.4 ms	1.02 s	57	**0.390**	0.921	0.888	0.948	0.725	0.234	Network Repository	An email network collected from the Enron energy accounting scandal of 2001.
				Layered	27.7 ms	17.9 ms	726 ms	**39**	0.385	0.928	0.873	0.949	0.725	0.244		
HIC 1K NET 6	4581	284924	4.2	Random	12.4 s	10.9 s	12.3 min	112	0.277	0.991	0.970	–	–	–	BioSNAP	Nodes are gnomic regions and edges are normalized contacts between regions.
				Radial	16.0 s	10.9 s	**6.2 min**	**55**	**0.303**	0.993	0.985	–	–	–		
LES MISERABLES	77	254	4.1	Random	12.9 ms	11.0 ms	770 ms	69	**0.444**	0.930	0.888	0.948	0.750	0.407	NetworkX	The co-appearance of charecters in chapters of Les Miserables.
				Layered	12.8 ms	11.7 ms	340 ms	**28**	0.424	0.920	0.873	0.945	0.759	0.417		
MOVIES	101	192	7.0	Random	27.0 ms	12.8 ms	1.14 s	87	**0.298**	0.862	0.819	0.932	0.732	0.483	[16]	Collabaration between 40 Hollywood composers and 61 producers between 1964 and 1976.
				Radial	13.1 ms	12.9 ms	685 ms	**52**	0.280	0.859	0.804	0.939	0.754	0.471		
SMITH	2970	97133	4.8	Random	4.98 s	3.91 s	114 s	28	0.045	0.882	0.705	–	–	–	Facebook 100	Nodes are students from Smith college and edges are the friendships between them.
				Radial	5.62 s	3.73 s	**20.5 s**	**4**	0.045	0.887	0.698	–	–	–		
TRAIN BOMBING	64	243	4.6	Random	31.2 ms	11.9 ms	1.84 s	152	0.393	0.912	0.887	0.898	0.746	0.311	NetworkX	Nodes are individuals involved in 2004 madrid train bombing and edge are for prior known relations.
				Radial	12.8 ms	11.6 ms	174 ms	**14**	**0.486**	0.954	0.930	0.928	0.751	0.304		
USAIR 97	332	2126	5.1	Random	101 ms	41.3 ms	2.21 s	**51**	**0.323**	0.938	0.851	0.876	0.711	0.301	Network Repository	A weighted graph of the air traffic between airports in the US in 1997.
				Layered	108 ms	41.2 ms	3.69 s	87	0.307	0.926	0.820	0.864	0.712	0.301		

Table 2: Table of *sparse* datasets. See Table 1 for a description.

Dataset	$\|V\|$	$\|E\|$	Avg ECC	Layout	T_{IT}	T_{AIT}	T_{LCMC}	C_{LCMC}	Q_{LCMC}	Q_{trust}	Q_{conv}	Q_{EC}	Q_{CA}	Q_{MAR}	Source	Description
AIRPORT	2896	15641	10.2	Random	908 ms	902 ms	69.5 s	76	0.245	0.948	0.849	–	–	–	Openflights.org	Airports are represented as nodes with edges counting the number of routes between airports.
				Layered	1.1 s	893 ms	**34.2 s**	**37**	**0.253**	0.962	0.870	–	–	–		
BALANCED TREE (3,6)	1093	1092	11.5	Random	92.9 ms	63.8 ms	7.55 s	117	0.221	0.808	0.676	0.995	0.702	0.752	NetworkX	This dataset represents a balanced tree with 3 children and a height of 6.
				Radial	109 ms	74.5 ms	6.07 s	**80**	**0.374**	0.982	0.940	1.000	0.671	0.779		
BARBELL	150	2501	48.3	Random	107 ms	35.7 ms	3.04 s	82	0.257	0.893	0.886	0.779	0.741	0.302	NetworkX	Two non-overlapping 50 node complete subgraphs connected by a 50 node path.
				Layered	108 ms	35.1 ms	529 ms	**12**	**0.350**	0.913	0.914	0.780	0.741	0.234		
BCSSTK	110	254	13.4	Random	32.6 ms	12.0 ms	1.41 s	115	**0.582**	0.972	0.897	0.984	0.685	0.318	UF Sparse Matrix Collection	The stiffness matrix used in structural simulation.
				Radial	16.8 ms	11.9 ms	1.27 s	**106**	0.575	0.973	0.900	0.990	0.705	0.386		
BIO-DISEASOME	516	1188	11.6	Random	76.1 ms	42.4 ms	3.64 s	84	0.407	0.909	0.845	0.991	0.738	0.440	Network Repository	Graph of links for scientifically known disorder-gene association [34].
				Radial	82.7 ms	42.8 ms	1.24 s	**27**	**0.478**	0.964	0.957	0.992	0.744	0.436		
CIRCULAR LADDER GRAPH (100)	200	300	51.0	Random	33.7 ms	16.6 ms	1.77 s	105	0.455	0.956	0.805	0.995	0.750	0.426	NetworkX	Pairs of nodes are connected in a ladder like pattern and the ladder forms a large cycle.
				Radial	20.2 ms	16.4 ms	1.11 s	**66**	**0.814**	0.999	0.998	1.000	0.427	0.557		
CONNECTED CAVEMEN (10,20)	200	1900	11.0	Random	85.3 ms	31.9 ms	1.33 s	39	0.493	0.985	0.986	0.961	0.766	0.050	NetworkX	A graph of 10 cliques of 20 nodes each.
				Radial	93.6 ms	32.9 ms	915 ms	**25**	0.496	0.986	0.985	0.961	0.768	0.045		
ENGYMES-G123	90	127	10.3	Random	13.9 ms	13.9 ms	1.7 s	121	0.374	0.891	0.808	0.979	0.788	0.611	Network Repository	Graph of cheminformatics.
				Layered	15.5 ms	14.3 ms	1.2 s	**83**	**0.436**	0.937	0.792	0.986	0.733	0.592		
LADDER	20	28	8.0	Random	3.6 ms	15.3 ms	1.71 s	111	0.377	0.932	0.929	0.975	0.834	0.507	NetworkX	Pairs of nodes connected in a ladder like pattern.
				Radial	2.5 ms	15.1 ms	1.21 s	**80**	**0.402**	0.950	0.947	1.000	1.000	0.602		
LOBSTER	300	299	77.5	Random	21.2 ms	20.9 ms	3.79 s	180	0.210	0.916	0.869	0.992	0.723	0.857	NetworkX	A tree that forms a caterpillar graph with the removal of a leaf [35].
				Layered	21.1 ms	19.9 ms	1.12 s	**55**	**0.688**	0.996	0.997	1.000	1.000	0.807		
LOLLIPOP (10,50)	60	95	40.2	Random	6.9 ms	14.3 ms	1.37 s	95	0.444	0.894	0.833	0.909	0.803	0.770	NetworkX	The shape of a lollipop with a clique of 10 nodes connected to a thread of 50 nodes.
				Layered	7.4 ms	14.4 ms	482 ms	**33**	**0.755**	0.996	0.995	0.921	0.813	0.551		
MAP OF SCIENCE	554	2276	12.6	Random	129 ms	56.4 ms	5.77 s	100	0.361	0.956	0.930	0.986	0.666	0.135	[10]	Graph of science sub-disciplines and cross-disciplinary co-authorships.
				Radial	153 ms	58.6 ms	4.14 s	**68**	**0.402**	0.977	0.957	0.990	0.687	0.145		
RANDOM GEOMETRIC (400,0,1)	400	2263	13.5	Random	107 ms	51.8 ms	4.56 s	86	0.546	0.983	0.925	0.990	0.665	0.071	NetworkX	Nodes randomly placed in a cube and connectted if their distance is less than 0.1.
				Radial	113 ms	53.2 ms	2.83 s	**51**	**0.622**	0.987	0.963	0.993	0.660	0.076		
RETWEET	96	117	7.3	Random	9.7 ms	14.0 ms	1.44 s	102	0.440	0.936	0.905	0.985	0.678	0.781	Network Repository	Network of twitter users as nodes and retweets as edges.
				Radial	9.8 ms	14.0 ms	977 ms	**69**	**0.466**	0.935	0.900	0.989	0.736	0.778		
SCIENCE COLLABORATION NETWORK	379	914	12.1	Random	61.6 ms	31.1 ms	2.52 s	79	0.433	0.947	0.887	0.992	0.741	0.352	[52]	Nodes are network theory publishing scientists and edges are collaborations between them.
				Layered	50.6 ms	36.2 ms	2.4 s	**65**	**0.500**	0.972	0.960	0.994	0.749	0.356		
WATTS STROGATZ (100,5,0,0.5)	100	200	12.3	Random	24.7 ms	13.1 ms	1.75 s	131	0.479	0.927	0.882	0.989	0.687	0.199	NetworkX	A small world graph that satisifies the Watts-Strogatz model [70].
				Layered	13.9 ms	13.8 ms	470 ms	**33**	**0.611**	0.976	0.941	0.991	0.646	0.196		

Initialization Time (T_{IT}) is the time taken to initialize the force-directed layout simulation. For our approach, the timing includes the overhead to calculate the spanning tree and position nodes.

Average Iteration Time (T_{AIT}) is the average time required to calculate one iteration of the force-directed layout.

Total Time to LCMC Convergence (T_{LCMC}) is the total time ($T_{IT} + T_{AIT} * C_{LCMC}$) required to reach the LCMC convergence criteria.

Finally, we produce a set of well-established graph readability metrics [21,36]. Our evaluation does not discuss them, but they are included for completeness.

Edge Crossings (Q_{EC}) measures the ratio of non-intersecting edges to total possible intersections. Graphs are generally considered more readable with fewer crossings. Q_{EC} is normalized, such that $Q_{EC} \in [0,1]$, where larger is better.

Crossing Angle (Q_{CA}) is the average deviation from the ideal crossing angle. If edges cross, it is preferable they cross at an ideal crossing angle of 70 degrees that makes their individual paths most visible. Q_{CA} is normalized, such that $Q_{CA} \in [0,1]$, where larger is better.

Minimum Angular Resolution (Q_{MAR}) measures the average deviation of adjacent edge angles from the ideal angle ($360°/degree(v_i)$ for any $v_i \in V$). For nodes with multiple edges, it is preferable to their egress be distributed around the node as much as possible. Q_{MAR} is normalized, such that $Q_{MAR} \in [0,1]$, where larger is better.

5 Results

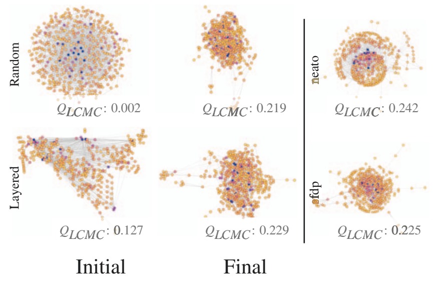
We evaluate our method's ability to untangle initial graph layouts, followed by untangling cycle structures.

5.1 Untangling Initial Graph Layouts

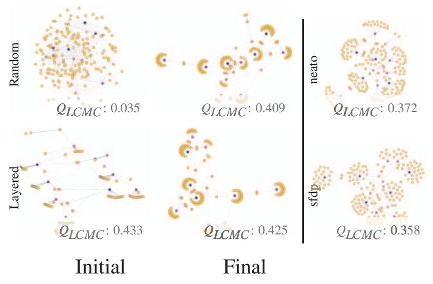
We evaluate our initial graph layout approach in terms of graph quality, convergence, and time. For the experiments, we initialized the graphs with either the standard D3.js random layout or our approach and let them run until D3.js stopped force calculations (after 300 iterations using the default settings).

5.1.1 Layout Quality

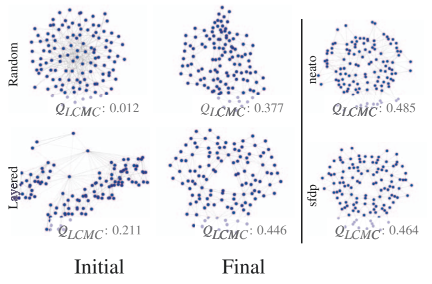
Table 1 and 2 show the results for all quality metrics from Sect. 4.3, except for the readability measures for the three largest graphs, which were skipped due to very high computational costs. Although all metrics are available, we discuss only Q_{LCMC}.



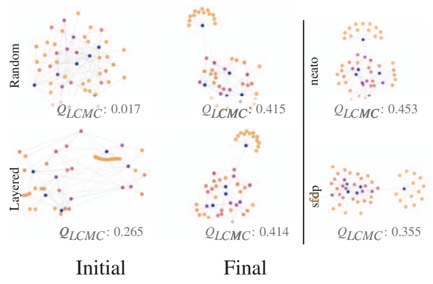
(a) BIO-CELEGANS



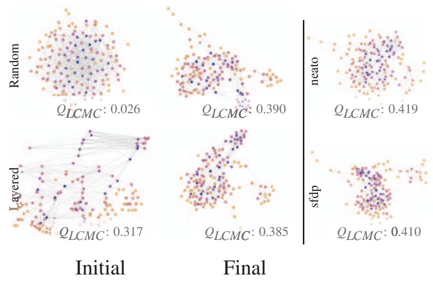
(b) BN-MOUSE-VISUAL-CORTEX-2



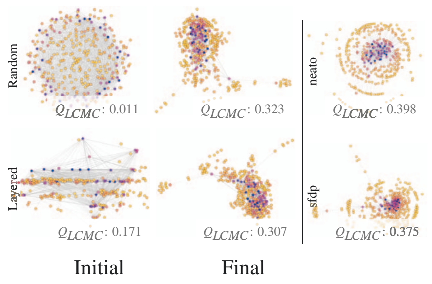
(c) DOROGOVTSEV-GOLTSEV-MENDES



(d) DUPLICATE DIVERGENCE



(e) ENRON EMAIL



(f) USAIR 97

Fig. 7: Example of *dense* graph initial layouts (left), final layouts (middle), and the result of running neato and sfdp (right). Graphs using our technique (bottom) converge more quickly and show similar, occasionally better, results compared to the standard random approach (top).

Layered vs. Radial. The results in Table 1 and 2 show only the layout method, layered or radial, which produced higher Q_{LCMC}. In many cases, the result between both methods is effectively identical. The results show that neither method is universally better and seemed to be graph dependent. Nevertheless, the results for both layout methods are available in the supplemental material.

Dense Graphs. With dense graphs (see Table 1 and Fig. 7), our approach generally produced higher Q_{LCMC} scores. However, for several graphs, our scores were similar or slightly lower, e.g., Fig. 7e and 7f. In these cases, the results are still quite similar visually. In general though for dense graphs, it seems that no matter the initial position of nodes, the layout will end in a more or less similar configuration. Importantly, even when our score is lower, our approach converges much more quickly, e.g., for the HIC 1K NET 6 dataset (see Fig. 11b), our method produced a similar Q_{LCMC} in about one fifth the time of the random layout. This property is discussed more in Sect. 5.1.2.

Sparse Graphs. With sparse graphs (see Table 2 and Fig. 9), the story is a bit different, as these graphs are not so overconstrained. Using random initial layouts, quite often, their topological structures are overlapping or hidden altogether. On the other hand, our approach untangles these topological structures, leading to better final graph layouts, e.g., with ENGYMES-G123 dataset (see Fig. 9c) our approach ($Q_{LCMC} = 0.436$) produces higher co-ranking scores than the random layout ($Q_{LCMC} = 0.374$), and the cycle structures of the graph are more clearly visible. There was one sparse case, BCSSTK, where our approach performed slightly worse than random because it was unable to untangle the long cycle in the graph (see supplement). However, the interactive functionality discussed in Sect. 5.2 resolved that issue.

Larger Graphs. Improving layouts is particularly important for larger graphs (i.e., graphs of 1000 nodes, see Sect. 4.2). Fig. 1 and 11 show examples of larger graphs. Our approach shows better clustering structures for the HIC 1K NET and AIRPORT datasets, supported by the improved Q_{LCMC} scores. SMITH, on the other hand, being a dense graph, shows similar clustering and identical Q_{LCMC} scores.

Overall, we observe that although our approach could improve the layout quality of dense graphs, sparse graphs almost always benefited from utilizing our approach.

5.1.2 Rate of Convergence

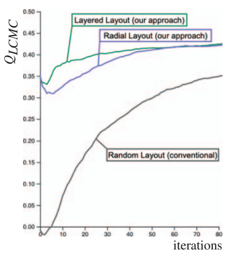
We compute the convergence metrics (see Sect. 4.3) on all of the datasets listed in Table 1 and 2, and we focus on the convergence of the LCMC (C_{LCMC}). Our results show that for all except one dataset, USAIR 97, using our approach converged faster than random layouts, often significantly so. Fig. 8 shows plots of Q_{LCMC} against iterations for three example datasets, including USAIR 97. In all cases, our approach starts with a much higher Q_{LCMC} score and fine-tunes the results. One interesting observation is that the Q_{LCMC} sometimes starts high and dips, e.g., in Fig. 8c. This results from our good initial layout being in a high energy state which is distorted by the force-directed layout before settling in a good quality low energy state.

The rate of convergence is particularly important for larger graphs, where the average time per iteration is higher. For the large datasets, AIRPORT, HIC 1K NET 6, and SMITH, our approach converged faster than random layouts, with 37 vs. 76, 55 vs. 112, and 4 vs. 28 iterations, respectively. This phenomenon is also visible in Fig. 1 and 11, where graphs layouts are shown at several intervals—initial, 5 iterations, 10 iterations, etc. In all cases, the structures shown in the final graph are visible much earlier with our approach (iteration 5 or 10) than with a random layout (iteration 25 or more).

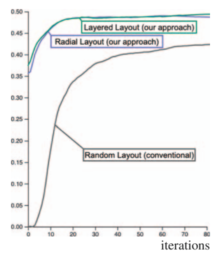
Therefore, we conclude that our approach significantly improves convergence in most cases.

5.1.3 Compute Time

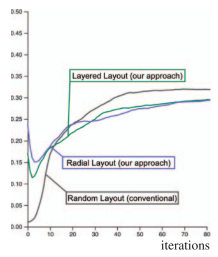
The improved rate of convergence our method offers does not come for free. An initialization time penalty (i.e., T_{IT}), albeit small, must be paid. Due to the imprecise time measurements offered by the web browser,



(a) ENGYMES-G123



(b) SCIENCE COLLAB.



(c) USAIR 97

Fig. 8: Plots of Q_{LCMC} against the number of iterations show that compared to the random initial layout, our approach begins with high co-ranking scores and spends most of the iterations fine-tuning.

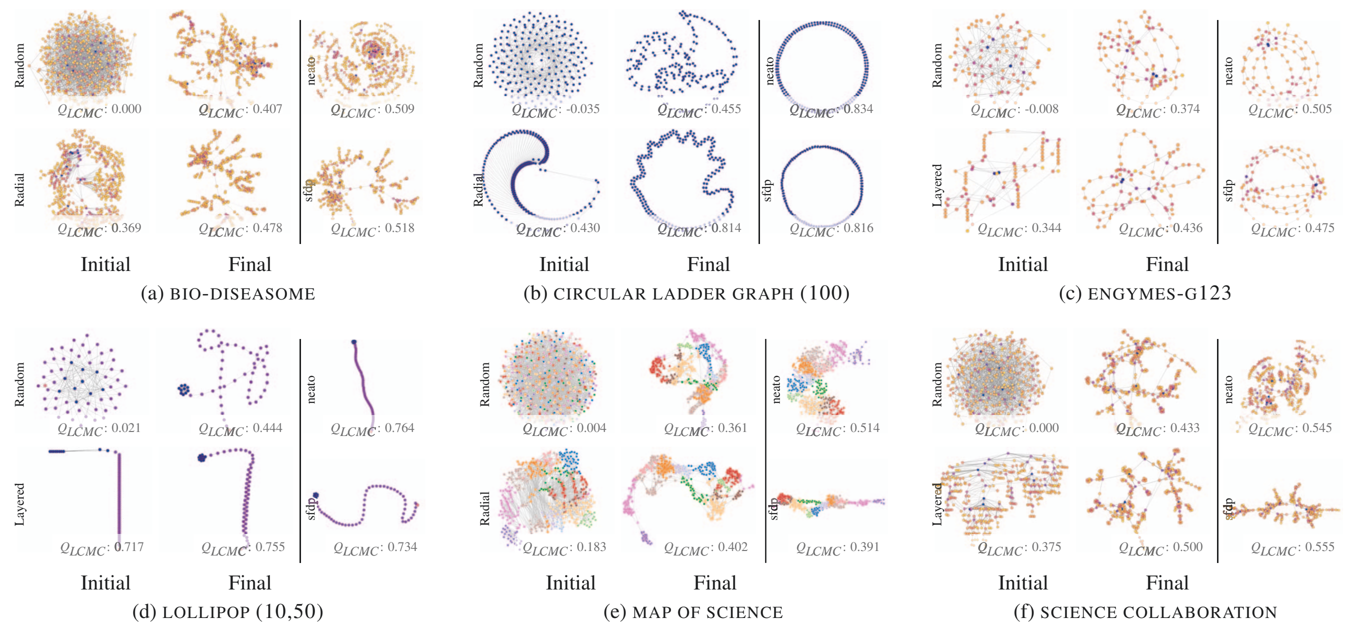


Fig. 9: Example of *sparse* graph initial layouts (left), final layouts (middle), and the result of running neato and sfdp (right). Graphs using our technique (bottom) converge more quickly and produce better layouts than the standard random approach (top).

we forgo discussing small graphs and focus on larger graphs instead, namely AIRPORT, HIC 1K NET 6, and SMITH. For these graphs, there was an additional initialization cost (T_{IT}) of $15-20\%$ over a random initialization. Since we made no modifications to the force calculations, the average time per iteration, T_{AIT}, was virtually identical.

However, the time benefit of our approach is placed in context when considering the time to convergence, T_{LCMC}. Due to the low additional overhead and significant reduction in number of iterations, our approach offers a speed-up of $\sim 2\times$ for AIRPORT and HIC 1K NET 6, and a speed-up of $\sim 5.6\times$ for SMITH.

Given these observations, we conclude the benefits of fast convergence far outweigh the additional initialization time required for the spanning tree calculation, particularly for larger graphs.

5.1.4 Comparison to Other Algorithms

We also compared our results to other graph layout algorithms, namely, *neato*, *fdp*, and *sfdp*. Due to space consideration, the majority of results are presented in our supplemental document. However, results on larger

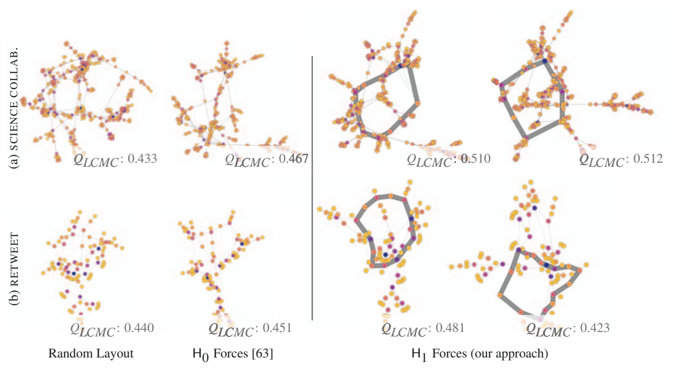


Fig. 10: A comparison of graphs using a random initialization and random initialization + H_0 forces, and examples of our approach on different cycles, random initialization + H_1 forces. The results show that our approach reveals cycles otherwise difficult to observe in the data, and in most cases, our approach improves the overall graph layout.

graphs can be seen in Fig. 1 and 11, and on smaller graphs in Fig. 7 and 9. Generally speaking, one or more of these methods produced graph layouts with similar or slightly better Q_{LCMC} scores than our approach. Therefore, our method can be viewed as closing the gap between random initialization force-directed layout and these more advanced methods. Ultimately, we are still limited by the capacity of the D3.js force-directed layout to produce high-quality final layouts. Our method provides only a boost. Finally, an important aspect of our approach is that the layouts are intended to be interactive. Users are supposed to explore H_0 and H_1 features to learn the graph's structure, which is a capability not necessarily offered by these other methods.

5.2 Interactive Untangling with H_1 Features

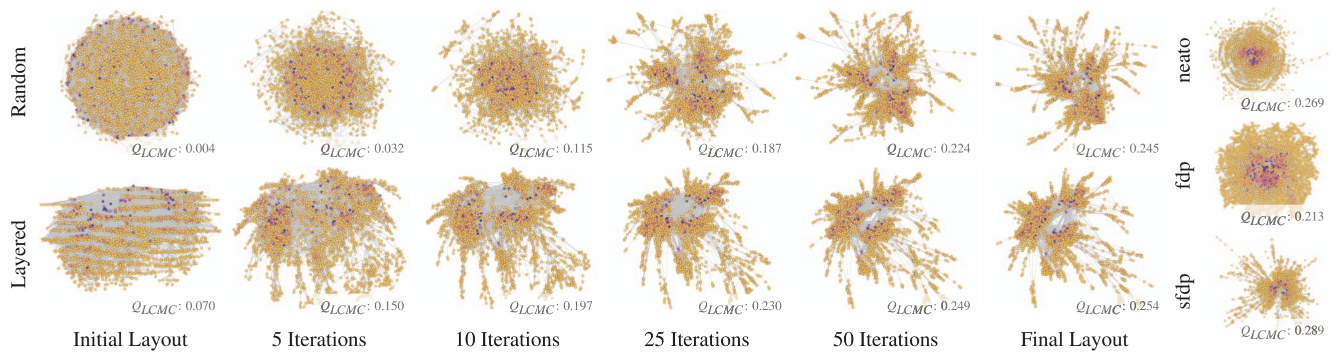
We evaluate whether the interaction with H_1 features reveals anything about the graph structure previously available using a random initialization force-directed layout or by using the H_0 forces introduced in [63].

Generating H_0 Examples. To compare to H_0 persistent homology feature forces, we start with the random initial layout (i.e., the D3.js default) and allow the graph to converge to a stable configuration. In other words, they look like the random final graph layouts in the upper middle of Fig. 7 and 9. We then apply a force to *all* H_0 features and again allow the graph to converge.

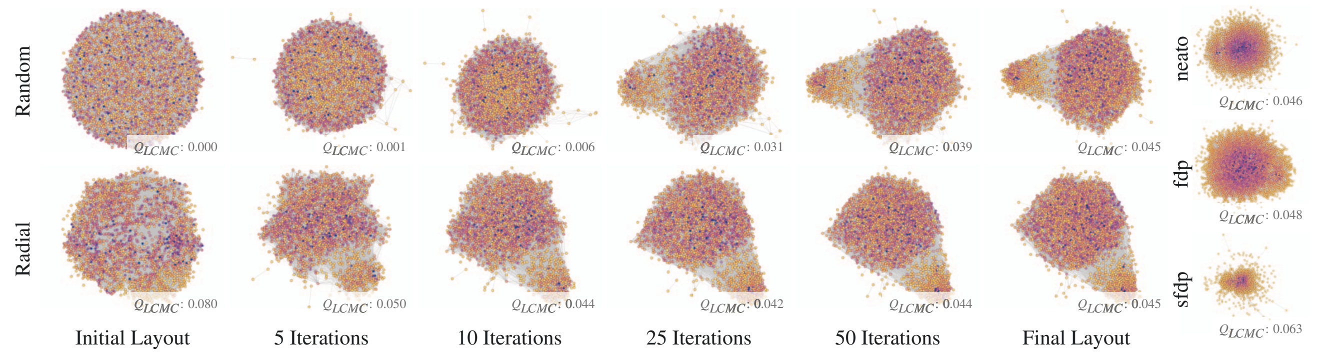
Generating H_1 Examples. To determine the efficacy of H_1 persistent homology feature forces, we configure them similarly. Starting with the random initial layout, we allow the graph to converge to a stable configuration. We then apply a force to a single H_1 feature, which is selected by considering H_1 features with longer cycle lengths, and again the graph is allowed to converge.

Results. The results are visible primarily in Fig. 10 and 12, and additionally in Fig. 2. First, we can see that our approach reveals cycle structures that are often hidden in standard graph layout and only sometimes revealed using H_0 features. In other words, our approach reveals topology of the graph that is otherwise hidden or difficult to see. The second observation we make is that whereas H_0 features tend to improve the overall presentation of the graph, i.e., higher Q_{LCMC} scores, our approach of applying forces to H_1 features has a much stronger impact, resulting in even better graph layouts, sometimes dramatically so, e.g., in CIRCULAR LADDER (Fig. 12b) or WATTS STROGATZ (Fig. 12d) graphs.

An important aspect of interacting with H_1 features is that highlight-



(a) AIRPORT dataset comparing random initial layout to layered over 300 iterations



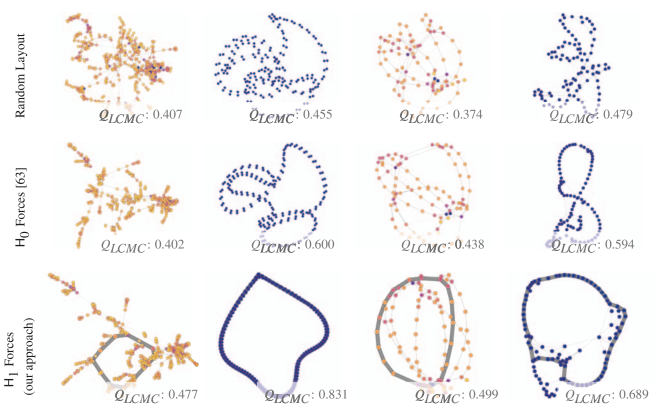
(b) SMITH dataset comparing random initial layout to radial over 300 iterations

Fig. 11: A comparison of random initial layout to our approach for large graphs shows the layouts at various stages of processing. Importantly, our approach shows the graph structure much earlier in the process, and the final layout is similar or better quality. We also compare to the results of non-interactive methods of neato, fdp, and sfdp (right).

ing different features may lead to very different graph layouts, e.g., see Fig. 10b. Taken in isolation, it may be difficult to visually relate the structures of each. However, when interacting, animation provides the context for relating different structures. Our demo[6] includes the ability to interactively evaluate additional cycles from each dataset.

Given these observations, we see that our approach has the ability to capture and highlight important cycle structures, and it can use those structures to further untangle a force-directed layout.

[6]Demo at *https://usfdatavisualization.github.io/UntangleFDL/*.



(a) BIO-DISEASOME (b) CIRCULAR LADDER (c) ENGYMES-G123 (d) WATTS STROGATZ

Fig. 12: Additional examples of graphs using a random initialization, random initialization + H_0 forces, or our approach, random initialization + H_1 forces.

6 Discussion and Conclusions

In this paper, we have evaluated two new uses of persistent homology on force-directed layouts. We first investigate using H_0 persistent homology for initializing graph layouts. Although the implementation itself relies on maximal spanning trees, persistent homology provides a theoretical foundation for justifying its use. At the same time, our experimental results show that it indeed improves the convergence rate and quality of force-directed layouts.

Second, we investigate using H_1 features for highlighting and modifying the forces of a force-directed layout. Here again, in addition to the algorithmic contribution of efficiently extracting the H_1 features, we observe that using these features reveals hidden features and improves graph layouts in many situations.

Beyond our current work, there is potentially room for developing additional initial layout schemes or perhaps automatically identifying which scheme would work best for a given dataset. Still, a good balance between performance and final quality remains of the utmost importance. In addition, our scheme for utilizing H_1 features could be utilized in a more elaborate manner. Additionally, it would be interesting to study whether other simplicial complexes could be used with persistent homology to capture topological information about other graph structures, e.g., cliques, stars and trees. Finally, our approach is implemented using the D3.js force-directed layout, but we believe the approach would work with other state-of-the-art techniques and frameworks. However, the exact implementation and the ultimate performance and quality gain require additional study.

Demo: *https://usfdatavisualization.github.io/UntangleFDL/*
Source: *https://github.com/USFDataVisualization/UntangleFDL/*

Acknowledgments

The authors wish to thank Ashley Suh, Mustafa Hajij, and Carlos Scheidegger for initial discussions. This work was supported in part by grants from the National Science Foundation (NSF) IIS-1513616 and IIS-1845204, and Department of Energy (DOE) DE-SC0021015.

References

[1] J. Abello, F. Van Ham, and N. Krishnan. Ask-graphview: A large scale graph visualization system. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):669–676, 2006.

[2] A. T. Adai, S. V. Date, S. Wieland, and E. M. Marcotte. Lgl: creating a map of protein function with an algorithm for visualizing very large biological networks. *Journal of molecular biology*, 340(1):179–190, 2004.

[3] D. Archambault, T. Munzner, and D. Auber. Topolayout: Multilevel graph layout by topological features. *IEEE Transactions on Visualization and Computer Graphics*, 13(2):305–317, 2007.

[4] D. Archambault, H. C. Purchase, and B. Pinaud. The readability of path-preserving clusterings of graphs. *Computer Graphics Forum*, 29(3):1173–1182, 2010.

[5] B. Bach, N. H. Riche, C. Hurter, K. Marriott, and T. Dwyer. Towards unambiguous edge bundling: Investigating confluent drawings for network visualization. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):541–550, 2016.

[6] C. Bachmaier, F. J. Brandenburg, W. Brunner, and G. Lovász. Cyclic leveling of directed graphs. In *International Symposium on Graph Drawing*, pages 348–359, 2008.

[7] M. Bampasidou and T. Gentimis. Modeling collaborations with persistent homology. *arXiv preprint arXiv:1403.5346*, 2014.

[8] A.-L. Barabási and R. Albert. Emergence of scaling in random networks. *science*, 286(5439):509–512, 1999.

[9] F. Beck, M. Burch, and S. Diehl. Towards an aesthetic dimensions framework for dynamic graph visualisations. In *International Conference Information Visualisation*, pages 592–597, 2009.

[10] 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.

[11] M. Bostock, V. Ogievetsky, and J. Heer. D^3 data-driven documents. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2301–2309, 2011.

[12] M. S. T. Carpendale and X. Rong. Examining edge congestion. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 115–116, 2001.

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

[14] L. Chen and A. Buja. Local multidimensional scaling for nonlinear dimension reduction, graph drawing, and proximity analysis. *Journal of the American Statistical Association*, 104(485):209–219, 2009.

[15] T. A. Davis and Y. Hu. The university of florida sparse matrix collection. *ACM Transactions on Mathematical Software (TOMS)*, 38(1):1–25, 2011.

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

[17] T. K. Dey, T. Li, and Y. Wang. Efficient algorithms for computing a minimal homology basis. In *Latin American Symposium on Theoretical Informatics*, pages 376–398. Springer, 2018.

[18] I. Donato, G. Petri, M. Scolamiero, L. Rondoni, and F. Vaccarino. Decimation of fast states and weak nodes: topological variation via persistent homology. In *Proceedings of the European Conference on Complex Systems 2012*, pages 295–301, 2013.

[19] H. Doraiswamy, J. Tierny, P. J. Silva, L. G. Nonato, and C. Silva. Topomap: A 0-dimensional homology preserving projection of high-dimensional data. *IEEE Transactions on Visualization and Computer Graphics*, 2020.

[20] S. N. Dorogovtsev, A. V. Goltsev, J. F. Mendes, and A. N. Samukhin. Spectra of complex networks. *Physical Review E*, 68(4):046109, 2003.

[21] C. Dunne, S. I. Ross, B. Shneiderman, and M. Martino. Readability metric feedback for aiding node-link visualization designers. *IBM Journal of Research and Development*, 59(2/3):14–1, 2015.

[22] C. Dunne and B. Shneiderman. Motif simplification: improving network visualization readability with fan, connector, and clique glyphs. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 3247–3256, 2013.

[23] T. Dwyer, K. Marriott, F. Schreiber, P. Stuckey, M. Woodward, and M. Wybrow. Exploration of networks using overview+ detail with constraint-based cooperative layout. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1293–1300, 2008.

[24] P. Eades. A heuristic for graph drawing. *Congressus numerantium*, 42:149–160, 1984.

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

[26] J. Ellson, E. Gansner, L. Koutsofios, S. C. North, and G. Woodhull. Graphviz—open source graph drawing tools. In *International Symposium on Graph Drawing*, pages 483–484. Springer, 2001.

[27] A. Frick, A. Ludwig, and H. Mehldau. A fast adaptive layout algorithm for undirected graphs (extended abstract and system demonstration). In *International Symposium on Graph Drawing*, pages 388–403, 1994.

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

[29] G. W. Furnas. Generalized fisheye views. *Acm Sigchi Bulletin*, 17(4):16–23, 1986.

[30] P. Gajer, M. T. Goodrich, and S. G. Kobourov. A multi-dimensional approach to force-directed layouts of large graphs. In *International Symposium on Graph Drawing*, pages 211–221. Springer, 2000.

[31] E. R. Gansner and S. C. North. Improved force-directed layouts. In *International Symposium on Graph Drawing*, pages 364–373, 1998.

[32] M. Ghoniem, J.-D. Fekete, and P. Castagliola. A comparison of the readability of graphs using node-link and matrix-based representations. In *IEEE symposium on information visualization*, pages 17–24, 2004.

[33] A. Godiyal, J. Hoberock, M. Garland, and J. C. Hart. Rapid multipole graph drawing on the gpu. In *International Symposium on Graph Drawing*, pages 90–101, 2008.

[34] K.-I. Goh, M. E. Cusick, D. Valle, B. Childs, M. Vidal, and A.-L. Barabási. The human disease network. *Proceedings of the National Academy of Sciences*, 104(21):8685–8690, 2007.

[35] S. W. Golomb. *Polyominoes: puzzles, patterns, problems, and packings*, volume 111. Princeton University Press, 1996.

[36] R. Gove. It pays to be lazy: Reusing force approximations to compute better graph layouts faster. In *Proceedings of the 11th Forum Media Technology*, pages 43–51, 2018.

[37] S. Hachul and M. Jünger. Drawing large graphs with a potential-field-based multilevel algorithm. In *International Symposium on Graph Drawing*, pages 285–295, 2004.

[38] S. Hachul and M. Jünger. Large-graph layout algorithms at work: An experimental study. *Journal of Graph Algorithms and Applications*, 11(21):345–369, 2007.

[39] 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.

[40] M. Hajij, B. Wang, C. Scheidegger, and P. Rosen. Visual detection of structural changes in time-varying graphs using persistent homology. In *IEEE Pacific Visualization Symposium (PacificVis)*, pages 125–134, 2018.

[41] D. Harel and Y. Koren. Graph drawing by high-dimensional embedding. In *International symposium on graph drawing*, pages 207–219. Springer, 2002.

[42] D. Holten and J. J. Van Wijk. Force-directed edge bundling for graph visualization. *Computer Graphics Forum*, 28(3):983–990, 2009.

[43] D. Horak, S. Maletić, and M. Rajković. Persistent homology of complex networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2009(03):P03034, 2009.

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

[45] T. Kamada, S. Kawai, et al. An algorithm for drawing general undirected graphs. *Information processing letters*, 31(1):7–15, 1989.

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

[47] J. B. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical society*, 7(1):48–50, 1956.

[48] H. Lee, M. K. Chung, H. Kang, B.-N. Kim, and D. S. Lee. Computing the shape of brain networks using graph filtration and gromov-hausdorff metric. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 302–309, 2011.

[49] H. Lee, H. Kang, M. K. Chung, B.-N. Kim, and D. S. Lee. Persistent brain network homology from the perspective of dendrogram. *IEEE transactions on medical imaging*, 31(12):2267–2277, 2012.

[50] A. Meidiana, S.-H. Hong, M. Torkel, S. Cai, and P. Eades. Sublinear time force computation for big complex network visualization. *Computer Graphics Forum*, 39(3):579–591, 2020.

[51] C. Muelder and K.-L. Ma. Rapid graph layout using space filling curves. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1301–1308, 2008.

[52] M. E. Newman. The structure of scientific collaboration networks. *Proceedings of the national academy of sciences*, 98(2):404–409, 2001.

[53] N. Otter, M. A. Porter, U. Tillmann, P. Grindrod, and H. A. Harrington. A roadmap for the computation of persistent homology. *EPJ Data Science*, 6:1–38, 2017.

[54] G. Petri, M. Scolamiero, I. Donato, and F. Vaccarino. Networks and cycles: a persistent homology approach to complex networks. In *Proceedings of the European Conference on Complex Systems*, pages 93–99, 2013.

[55] G. Petri, M. Scolamiero, I. Donato, and F. Vaccarino. Topological strata of weighted complex networks. *PloS one*, 8(6):e66506, 2013.

[56] H. Purchase. Which aesthetic has the greatest effect on human understanding? In *International Symposium on Graph Drawing*, pages 248–261, 1997.

[57] E. M. Reingold and J. S. Tilford. Tidier drawings of trees. *IEEE Transactions on Software Engineering*, SE-7(2):223–228, 1981.

[58] B. Rieck, U. Fugacci, J. Lukasczyk, and H. Leitte. Clique community persistence: A topological visual analysis approach for complex networks. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):822–831, 2017.

[59] R. Rossi and N. Ahmed. The network data repository with interactive graph analytics and visualization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29, 2015.

[60] B. Saket, P. Simonetto, S. Kobourov, and K. Börner. Node, node-link, and node-link-group diagrams: An evaluation. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2231–2240, 2014.

[61] M. Sarkar and M. H. Brown. Graphical fisheye views. *Communications of the ACM*, 37(12):73–83, 1994.

[62] K. Sugiyama, S. Tagawa, and M. Toda. Methods for visual understanding of hierarchical system structures. *IEEE Transactions on Systems, Man, and Cybernetics*, 11(2):109–125, 1981.

[63] A. Suh, M. Hajij, B. Wang, C. Scheidegger, and P. Rosen. Persistent homology guided force-directed graph layouts. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):697–707, 2019.

[64] K. J. Supowit and E. M. Reingold. The complexity of drawing trees nicely. *Acta Informatica*, 18(4):377–392, 1983.

[65] R. Tamassia. *Handbook of graph drawing and visualization*. CRC press, 2013.

[66] J. Venna and S. Kaski. Local multidimensional scaling. *Neural Networks*, 19(6-7):889–899, 2006.

[67] F. B. Viegas, M. Wattenberg, F. Van Ham, J. Kriss, and M. McKeon. Manyeyes: a site for visualization at internet scale. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1121–1128, 2007.

[68] 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. *Computer Graphics Forum*, 30(6):1719–1749, 2011.

[69] Y. Wang, Y. Wang, H. Zhang, Y. Sun, C.-W. Fu, M. Sedlmair, B. Chen, and O. Deussen. Structure-aware fisheye views for efficient large graph exploration. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):566–575, 2018.

[70] D. J. Watts and S. H. Strogatz. Collective dynamics of 'small-world' networks. *nature*, 393(6684):440–442, 1998.

[71] E. Weinan, J. Lu, and Y. Yao. The landscape of complex networks. *arXiv preprint arXiv:1204.6376*, 2012.

[72] S. Weinberger. What is... persistent homology? *Notices of the AMS*, 58(1):36–39, 2011.

[73] C. Wetherell and A. Shannon. Tidy drawings of trees. *IEEE Transactions on Software Engineering*, SE-5(5):514–520, 1979.

[74] M. Zinsmaier, U. Brandes, O. Deussen, and H. Strobelt. Interactive level-of-detail rendering of large graphs. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2486–2495, 2012.

[75] M. Zitnik, R. Sosič, S. Maheshwari, and J. Leskovec. BioSNAP Datasets: Stanford biomedical network dataset collection. `http://snap.stanford.edu/biodata`, 2018.

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