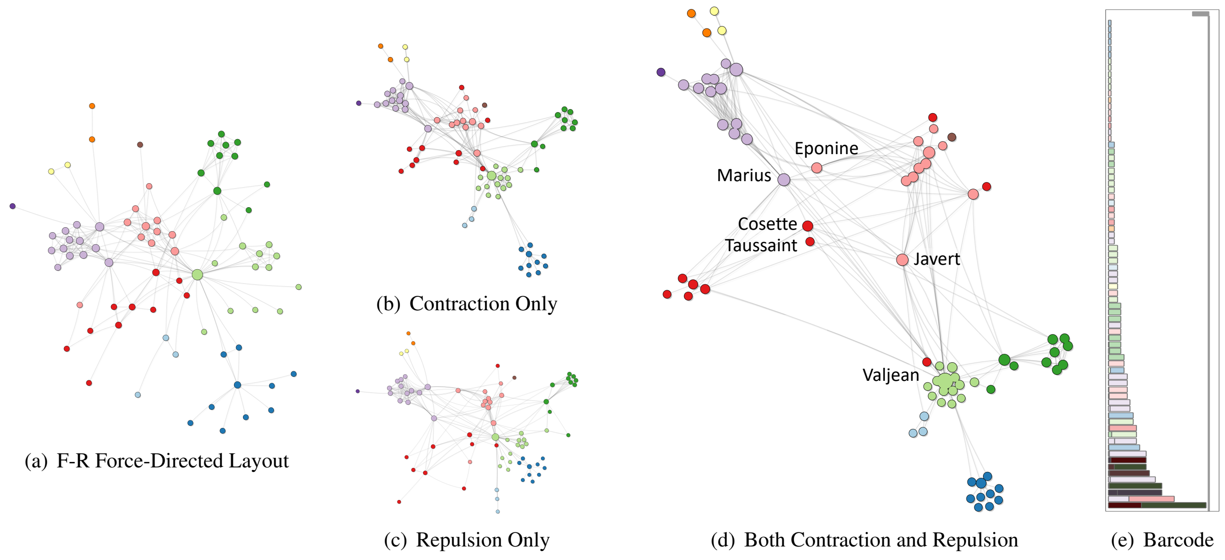
Persistent Homology Guided Force-Directed Graph Layouts

Ashley Suh, Mustafa Hajij, Bei Wang, Carlos Scheidegger, and Paul Rosen



(a) F-R Force-Directed Layout (b) Contraction Only (c) Repulsion Only (d) Both Contraction and Repulsion (e) Barcode

Fig. 1. (a) The Lés Miserables graph is drawn using a Fruchterman-Reingold (F-R) force-directed layout [31]. Our approach provides two mechanisms for interacting with the force-directed layout using (e) the persistence barcode. (b) The first mechanism contracts nodes of the graph associated with features of low significance or *persistence*. (c) The second mechanism partitions the graph using user-selected features and repulses the nodes in different partitions from one another. (d) When combined, this approach allows interactively controlling the layout to emphasize user-selected aspects of the graph using persistent homology.

Abstract—Graphs are commonly used to encode relationships among entities, yet their abstractness makes them difficult to analyze. Node-link diagrams are popular for drawing graphs, and force-directed layouts provide a flexible method for node arrangements that use local relationships in an attempt to reveal the global shape of the graph. However, clutter and overlap of unrelated structures can lead to confusing graph visualizations. This paper leverages the persistent homology features of an undirected graph as derived information for interactive manipulation of force-directed layouts. We first discuss how to efficiently extract 0-dimensional persistent homology features from both weighted and unweighted undirected graphs. We then introduce the interactive persistence barcode used to manipulate the force-directed graph layout. In particular, the user adds and removes contracting and repulsing forces generated by the persistent homology features, eventually selecting the set of persistent homology features that most improve the layout. Finally, we demonstrate the utility of our approach across a variety of synthetic and real datasets.

Index Terms—Graph drawing, force-directed layout, Topological Data Analysis, persistent homology.

1 Introduction

Graphs are ubiquitous for representing complex relationships between individuals or objects and are often used to model social interactions, energy grids, computer networks, brain connectivity, etc. The abstractness of graphs provides significant flexibility in visualization. However, dense, low-diameter subgraphs lead to confusing visualizations that appear as "hairballs". A good graph visualization should present *structure* quickly and clearly, and support further investigation of the data.

- *Ashley Suh is with the University of South Florida and Tufts University. E-mail: ashley.suh@tufts.edu.*
- *Mustafa Hajij is with the Ohio State University. E-mail: hajij.1@osu.edu.*
- *Bei Wang is with the University of Utah. E-mail: beiwang@sci.utah.edu.*
- *Carlos Scheidegger is with the University of Arizona. E-mail: cscheid@email.arizona.edu.*
- *Paul Rosen is with the University of South Florida. E-mail: prosen@usf.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

A key element of node-link diagrams is the layout algorithm that places nodes on the display (semi-)automatically. The problem of automatic graph layout has a rich literature, in which many approaches focus on finding an embedding of the graph by optimizing a readability metric [77], such as symmetry of the graph, lengths of the edges, or the number of edge crossings. A significant advancement was the realization that the use of derived information, such as node rank, graph distance, or approximate clustering, could improve graph layouts [23, 34, 68]. However, many of these techniques either lack the ability to interactively manipulate the graph layout, or lack the temporal coherency of the layout necessary to make such interactions effective.

When considering graph layouts that support interactivity, perhaps the most popular (though not necessarily the *best*) method is a force-directed or spring-mass layout [36], which converts the graph into a physical system of attractive springs and repulsive forces that iteratively minimize an energy function. These systems rely upon local relationships to reveal the overall shape in the graph. The result is a method that shows topological structures in certain graphs, particularly sparse ones. However, this approach often causes unrelated or distant topological structures to overlap or cross paths, making them difficult to differentiate. Some capacity to address this problem is provided through user interaction. Unfortunately, the interaction is most often through clicking and dragging individual nodes, which is ineffective for larger graphs and constrained by the forces applied to the graph layout.

This paper addresses the interactive manipulation of force-directed graph layouts by leveraging *persistent homology* (PH) [27,35] as derived information for the visualization of undirected graphs. PH has recently been shown to be a robust descriptor of graphs [41,78], and it has a few key qualities that make it ideal for this application. First, the PH calculation extracts *PH features*, in the form of 0-dimensional homological groups, from a graph without the need to select parameters. Second, the PH features can be quantified and ranked according to their significance, known as *persistence*. Third, they are invariant under small deformations, making them insensitive to noise and other small variations in data (e.g., removing a low-weight edge does not significantly change the graph) [41]. Finally, the set of all PH features produces a compressed description of the graph that can be represented using a *persistence barcode* [35], which our approach uses as a graphical user interface to manipulate the graph via *PH features*, instead of direct node manipulation.

In brief, our approach works as follows. We embed an undirected graph in a metric space by inducing a distance between all nodes. We extract the PH features (i.e., the 0-dimensional homological features) of the metric space structure [28] and sort them using their significance (i.e., persistence). Starting with a Fruchterman-Reingold force-directed layout [31], we employ the PH features in two user-selectable ways. First, a selected PH feature can create a strong attractive force between the nodes that created the feature, causing them to contract (see Fig. 1(b)). Second, a selected PH feature can be used to partition the graph into two subsets, which are repulsed from one another (see Fig. 1(c)). The user employs as many contractive or repulsive forces as desired, in order to emphasize graph elements of interest (see Fig. 1(d)).

Contribution. We demonstrate the usefulness of using 0-dimensional PH features for controlling force-directed layouts. In summary: 1) we discuss extracting PH features from both weighted and unweighted graphs; 2) we introduce new forces into the layout that are derived from PH features; 3) we provide an interactive interface, based upon the persistence barcode, that allows users to interactively manipulate the layout using the PH features; and 4) we evaluate the approach by comparing it to popular force-directed layouts and clustering algorithms.

2 Prior Work

Graph Visualization. Graph visualization is a broad area, as demonstrated by von Landesberger et al.'s survey [88]. Our treatment focuses on approaches for drawing node-link diagrams [36], which are used to display graphs in popular visualization systems, including Gephi [9], NodeXL [42], and Graphviz [29].

The first automated technique for laying out node-link diagrams was Tutte's barycentric coordinate embedding [86], followed by linear programming techniques [34], force-directed/mass-spring embeddings [31,47], embeddings of the graph metric [33], and techniques exploiting linear-algebraic properties of the connectivity structures [11,53,55,56]. Hybrid approaches, such as TopoLayout [4], analyzed graph topology to identify the best type of graph embedding. Recent work using stress majorization has introduced the ability to add constraints that enable those layouts to highlight certain properties, such as stars, clusters, or circles [89]. The more challenging problem of visualizing multivariate networks has been addressed through visual analytics approaches [87].

Edge clutter presents a challenging problem for node-link diagrams. For denser graphs, edge bundling can reduce clutter by routing graph edges to the same portion of the screen [45]. In terms of quality, divided edge bundling [83] produces high-quality results, whereas hierarchical edge bundling [32] scales to millions of edges. There are also localized versions of edge bundling [91] and filtering [48,84] that adapt the display of edges based upon a user-selected region of interest.

Other visual metaphors have been proposed to reduce overall clutter, ranging from relatively conservative proposals, such as replacing nodes with motifs [24] based on graph topology or modules [25], to more aggressive forms, such as variants of matrix diagrams [21] and abstract displays of graph statistics [50].

When displaying a large dataset, it is natural to question the hard visual limits for graphs. Popular approaches such as pixel-based visualizations [51,52] encode large amounts of data within small rectangles or display pixels. Space-filling curves have also been used to build pixel-based graph visualizations [64]. Furthermore, using visual boosting [73] tailored to network data may further reveal hidden information.

Research into interactive manipulation of force-directed node layouts includes interaction techniques [44,88] such as panning and zooming, which are used to focus on regions of interest in a graph. In addition to interacting directly with nodes in force-directed layouts [31,47], approaches have included hierarchical layout constructions [43] and exploration [3,5], fisheye lenses [81,90], interactive refinement of automatic layouts [30], or constraint-based optimization [80].

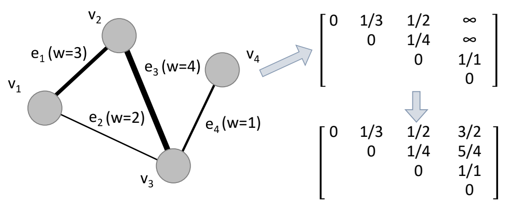
Persistent Homology and Graphs. PH is an emerging tool in studying complex graphs [22,26,46,74,75], including collaboration networks [7,12] and brain networks [13,17,58–61,76]. PH has recently been used in the visualization community for graph analysis targeting clique communities [78] and time-varying graphs [41]. Many of these techniques take similar approaches, using PH to summarily analyze, quantify, and compare graphs. In contrast, our approach focuses on using PH to enable interactive manipulation of the graph layout.

3 Persistent Homology of a Graph

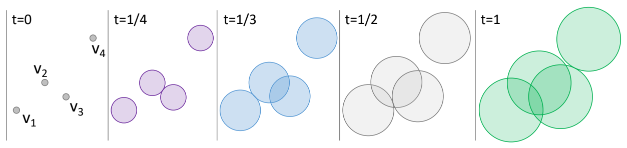
We first provide a theoretic framing for our approach that is grounded in PH. In Sect. 4.1 we will discuss how the restricted form of PH used in this paper is a special case of single-linkage hierarchical clustering.

To extract PH features from a graph, we apply PH to a metric space representation of the graph [41]. See [27] for an introductory survey and [28] for a formal treatment of PH.

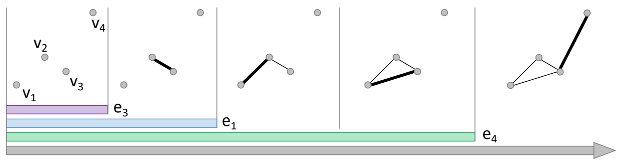
In algebraic topology, 0-dimensional homology groups of a graph describe the connected components of a metric space at a *single spatial*



(a) Conversion from Graph to Metric Space Embedding



(b) Conceptual Construction of PH Features Using Growing Balls



(c) Computational Construction of PH Features Using 1-Simplicies

Fig. 2. Example of extracting 0-dimensional PH of a graph. (a) Given an undirected graph G with edge weights w, we obtain a metric space representation by converting weights to distances by using $d = 1/w$ and completing the metric space using shortest-path distance. (b) Conceptually, components, or PH features, are formed around each point in the metric space. Balls grow around the points in the metric space to identify the diameter t at which components merge into larger components. (c) A filtration is constructed from G by adding edges when two balls intersect. When two components merge into one, the bar associated with one of the component in the persistence barcode (bottom) terminates.

resolution. In this paper, we use a multiscale notion of homology, persistent homology (PH), to describe the evolution of features of the space at different spatial resolutions.

Given a weighted graph $G = (V, E, w)$ with positive edge weights defined by $w : E \to \mathbb{R}$, our first step is to associate the graph G with a metric space representation. Considering the *inverse*[1] of the positive edge weight as the length of an edge, a classical *shortest path* metric d is defined on G, where the distance between each pair of nodes $x, y \in G$ is the shortest path between them. This metric can be computed using Dijkstra's algorithm [19]. See Fig. 2(a) for an illustration. The remainder of the algorithm operates only on the metric space, no longer considering the original graph.

Every node in G corresponds to a point in the metric space. To compute the 0-dimensional PH of G, we apply a simple geometric construction on its metric space representation. Consider the set of balls centered at every point in the metric space with a diameter t. We keep track of how the components of the union of balls evolve as t increases from $0 \to \infty$. As t increases, the unions of balls form components in a hierarchical fashion.

Considering Fig. 2(b), starting with each point as a component when $t = 0$, as t increases, the number of components decreases by one when two components merge—formally, this is referred to as a *topological event*[2]. At $t = 1/4$, the balls representing v_2 and v_3 touch, causing the merging of the components $\{v_2, v_3\}$. At $t = 1/3$, the balls representing v_1 and v_2 touch, causing the merging of sets $\{v_1\}$ and $\{v_2, v_3\}$. At $1/2$, v_1 and v_3 touch. However, they are already part of the same component $\{v_1, v_2, v_3\}$, meaning no merging occurs. Finally, at $t = 1$, v_3 and v_4 touch, merging the components $\{v_1, v_2, v_3\}$ and $\{v_4\}$.

A PH feature corresponds to the birth (appearance) and death (merging) of a component (union of balls) in the metric space. The *birth* of a component is the diameter when the component appears. In our case, all points appear simultaneously with diameter 0. The *death* is the diameter at which a component disappears; that is, when two components merge, one will disappear by joining the other (the choice of which disappears is discussed in the next section). The lifetime of a component (i.e., its death time minus its birth time) is its *persistence*.

The PH features associated with G are placed into a *persistence barcode* [35], which consists of a collection of bars, each corresponding to a single PH feature, whose starting and ending points correspond to the birth time and the death time of its associated component, with a width proportional to its persistence. See Fig. 2(c) for an illustration.

4 Computation of Persistent Homology

The restricted form of PH used in this paper is functionally equivalent to single-linkage clustering. Therefore, the PH of the graph can be calculated by finding the minimum spanning tree (MST) of the graph using Kruskal's algorithm [57].

4.1 Fast Computation of the 0-Dimensional Barcode

In using Kruskal's algorithm to compute the 0-dimensional barcode of the graph G, we suppose for simplicity that G is connected. However, if it is not, each connected component of the graph is processed independently. In a nutshell, our algorithm consists of computing the MST T of $G = (V, E, w)$ based on its metric space embedding using edge lengths $1/w$.

Let $V = \{v_1, \cdots, v_m\}$ be the node set of G and let $E = \{e_1, \cdots, e_n\}$ be its edge set sorted in increasing order with respect to $1/w$.

The algorithm starts by creating an empty spanning tree. It then creates components C_i and bars $\mathcal{B}_i$, one per graph node. Each bar in the persistence barcode is represented by a pair of real numbers $(birth, death)$, initially $birth = 0$ for each node as its own component. The second step of the algorithm looks at the edges one at a time, ordered by increasing $1/w_i$. For each e_i, we check if nodes of this edge belong to two different components. If this is the case, then we merge the components and set the *death* of one to be the $1/w_i$. The choice of which component dies is arbitrary and does not influence our result. The *persistence* of the component that dies is its death time minus its birth time. The appearance and the disappearance of such a component gives rise to a bar $(0, 1/w_i)$ in the persistence barcode. This step can be performed efficiently using the *disjoint set* data structure.

[1]The inverse of the edge weight between two nodes $1/w(x,y)$ captures the dissimilarity between them.

[2]In this context, topology refers to the homology groups of the metric space, not to be confused with graph topology.

```
Data: A weighted graph $G = (V, E, w)$
Result: Minimum spanning tree $T$ and 0-dim barcode $\mathcal{B}$
1  Create an empty spanning tree $T = \{\}$
2  foreach node $v_i$ do
3      Create a component $C_i = \{v_i\}$
4      Create a bar $\mathcal{B}_i$ with $birth = 0$ and $death = \infty$
5  end
6  foreach edge $e_i = (u, v)$ in $E$ do
7      if $C_u$ and $C_v$ are different components then
8          Merge $C_u$ and $C_v$
9          Set the death of $\mathcal{B}_u$ to $1/w(e_i)$
10         Add $e_i$ to the spanning tree $T$
11     end
12 end
```

The addition of an edge to the spanning tree coincides with the event of two components merging. Therefore, there is a one-to-one correspondence between edges of the MST T and the 0-dimensional barcode of G with finite persistence.

4.2 Node Relationships with the Spanning Tree

We relate information encoded by the MST to the graph G, which will later define modifications to the graph layout. Denote the MST as $T(V, E)$, where E denote the edges in the tree. Deleting an edge $e = (u, v)$ from E splits the tree T into two sets, V_u and V_v.

Each PH feature (i.e., a bar in the barcode) is associated with the following information, as illustrated in Fig. 3:

- For our purpose, we visualize each bar in the persistence barcode as an interval $(0, w)$ instead of $(0, 1/w)$. Such a visualization emphasizes high weight edges as long bars[3]. Under an abuse of notation, the *persistence measure* of such a bar is assigned w.
- The *cause of death*, u and v, are the nodes of the edge that cause the components to merge. These nodes will be used to modify the graph layout to the reflect PH feature selection.
- The *subsets of nodes*, V_u and V_v, represent the sets of connected nodes after the removal of the edge from the MST. These sets will also be important when updating the graph layout.
- The *subset ratio* is a measure of the number of nodes, $|V_u| : |V_v|$ in the two subsets of nodes. It is a measure of centrality within the MST. For example, in Fig. 4, an example MST is augmented with the subset ratios. Edges in the fan-like areas to the left and right have low ratios, 1:7. The 2 central edges have a more balanced ratio, 4:4 and 3:5, indicating that they are more central to the MST. The distribution of subset ratios is entirely data dependent.

[3]This visualization is different from the conventional PH approach; however it is justified as edges with higher weights w are considered more important in our setting.

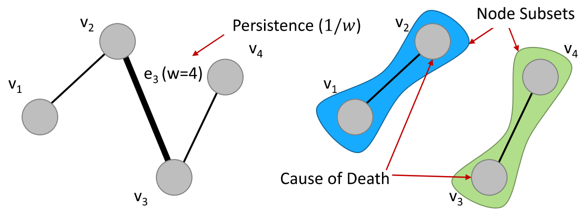


Fig. 3. Example of information extracted from a spanning tree (left) for edge e_3 from Fig. 2(c) (the purple bar). The right shows the clusters created when a selected edge is removed from the spanning tree.

However, we observed in our examples that low ratios are far more common than balanced ratios.

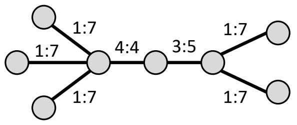


Fig. 4. Example MST with associated subset ratios. Nodes toward the periphery have low ratios, whereas more central nodes have balanced ratios.

4.3 Application to Unweighted Graphs

The calculation described above requires a weight on the edges of the graph. For unweighted graphs, a similarity measure, such as the Jaccard index [49], can take the place of edge weight.

Our procedure first gathers for each node a neighborhood or ego graph [66] within a user-selectable number of hops. For example, a 1-hop ego graph will contain adjacent neighbors, whereas a 2-hop ego graph will contain both 1-hop neighbors and nodes adjacent to the 1-hop neighbors. The ideal number of hops is data dependent. We used 1-hop for denser graphs and 2- or 3-hops for sparser graphs. Then, given an edge $e = (v_i, v_j)$, with neighborhood graphs N_i and N_j, the Jaccard index between those nodes is $J(v_i, v_j) = \frac{|N_i \cap N_j|}{|N_i \cup N_j|}$. The edge weight becomes $w(e) = J(v_i, v_j)$. In this way, the Jaccard index provides the similarity between the neighbors of 2 connected nodes. Finally, we proceed with the approach described in Sect. 3.

Other measures, such as edge centrality [37], can be used in place of the Jaccard index, to highlight different aspects of the graph. The only requirement is that the measure must provide a weight on the edges.

5 Visual Design and Interaction Design

Our design goal is to use the PH of a graph as a control to manipulate its force-directed layout. We use a simple interactive interface to provide contextual information and enable fast manipulation of the layout.

5.1 Graph Drawing

Our design uses a force-directed layout, with optional edge bundling.

Force-Directed Layout. A graph is initially drawn with a Fruchterman-Reingold (F-R) force-directed layout [31]. The layout starts with three types of forces. The first force enables all nodes to repulse one another (Fig. 5(c)). The second force is a spring attraction for nodes connected by an edge (Fig. 5(d)). Finally, a weak attracting force draws all nodes toward the middle of the display, essentially centering the layout (Fig. 5(e)). The parameters for these forces, such as mass, force strength, and spring resting length, require manual tuning.

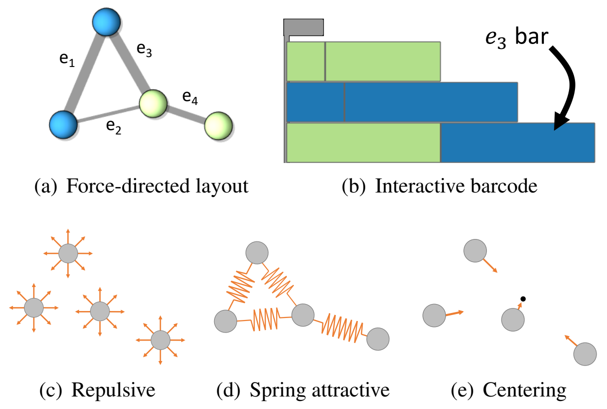


Fig. 5. The force-directed layout (a) for the graph in Fig. 2 is constructed using (c) repulsive forces, (d) spring attractive forces, and (e) a centering force. The interactive barcode (b) is used to manipulate the display.

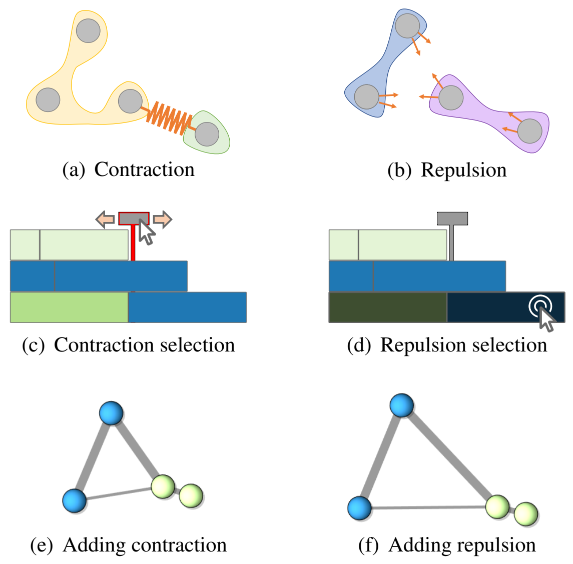


Fig. 6. Illustration of forces applied to the graph in Fig. 5(a). (a, c, and e): A contracting force applied to the layout. (b, d, and f): A repulsive force is then added to the layout from (e).

Edge Bundling. For analysis tasks that benefit from further clutter reduction [6, 63], edge bundling can be optionally used to reduce the distractive impact of overlapping edges. Since the layout is actively manipulated, we require an edge bundling implementation that is temporally coherent. To accomplish this, we implemented force-directed edge bundling [45]. This technique subdivides edges and uses a variant of a force-directed layout to attract edges with similar proximity and direction. Fig. 1 shows an example with edge bundling enabled.

Node Coloring. Some of our experimental graphs have categorical data attached to the nodes and are colored using a categorical color map. Other graphs contain nodes colored using node degrees.

5.2 Persistence Barcode

As mentioned in Sect. 4.2, we visualize the persistence barcode in a unconventional way. A PH feature that is born at 0 and dies at $1/w$ is visualized by a bar $(0, w)$ whose length indicates its persistence measure w. That is, the larger the bar, the more “important” the PH feature it represents. We augment the barcode with additional visual encodings (Fig. 5(b)) to further guide interaction.

Subset Ratio. Each bar is augmented with a vertical line, splitting it into two based upon the *Subset Ratio*. For example, in Fig. 5(b) the bottom bar, representing e_3 from Fig. 2 and 3, has a 50/50 split, because two nodes exist on either side of the edge in the MST (see Fig. 3).

Color. For graphs with categorical data, each side of the split is colored based on the category of the *Cause of Death* nodes for the PH feature.

Bar Sorting. Bars are sorted based upon two criteria. The first is *persistence* from low to high, which helps to differentiate high persistence features from those with low persistence. If the persistence values of two bars are equal, then they are sorted by their *Subset Ratios*, with bars having 50/50 ratios appearing lower in the barcode, since those are more central to the MST.

5.3 Interaction Using the Persistence Barcode

PH Feature Contraction. In certain scenarios, it is desirable to shrink space allocated to nodes in the layout, making more room for other parts of the graph to be displayed. We provide a *persistence simplification* tool that enables contraction of all PH features whose persistence is below a user-selected threshold. This interaction is done by dragging a filter bar at the top of the barcode; see the filter bar in red in Fig. 6(c). As the threshold is dragged left to right, PH features with persistence below that threshold will have their color washed out and their graph nodes contracted. A contraction is accomplished by adding a strong spring force between the *cause of death* nodes for the event. The scale of this force is user selectable. Fig. 6(a) illustrates this force and Fig. 6(e) shows an example of the contraction.

PH Feature Repulsion. In other situations, stretching out the graph can clear space for nodes and edges that might otherwise be overlapping and difficult to track. When a bar is individually selected, the bar color is darkened, and a strong repulsive force is added between the *subsets of nodes* associated with that PH feature. The scale of the force is user selectable. The repulsion of the nodes from these two groups allows the layout to cluster. For example, Fig. 6(b) illustrates the force when the bottom bar in Fig. 6(d) is selected for repulsion, causing the subsets to push apart. In other words, in Fig. 6(b), all the points in the blue region have an additional repulsion from all of the points in the purple region and vice versa. Fig. 6(f) shows the result of adding this force to the graph.

Selecting Multiple Bars. Multiple bars may be selected for contraction and repulsion, since the forces do not directly depend on each other. Whenever a new bar is selected, the force is simply added to the layout.

Preview Hovering. To help users preview the impact of a bar's selection when the mouse hovers over a bar in the barcode, set visualization can be employed, such as bubble sets [15] or kelp diagrams [20]. We employ bubble sets on the *subsets of nodes* to differentiate which nodes belong to which subset. Fig. 7 shows examples of bubble sets employed on a dataset. The bubble sets demonstrate two examples of before and after the PH feature is selected for repulsion, respectively.

Hyperbolic Zoom for a Large Barcode. The number of bars is equal to the number of nodes in the graph minus one. In order to scale the barcode appropriately for a large graph, a scrollbar with hyperbolic zoom is placed to the right of the bars. As the scrollbar moves, the focus of the hyperbolic zoom is modified to emphasize the associated bars. An example of this can be seen in our accompanying software.

5.4 Typical Usage Session

After a graph is loaded and any adjustments are made to standard force strengths, the user begins to explore the PH features of the graph. To start, the contraction forces are explored by slowly adjusting the threshold higher (to the right), which enables finding when the compactness reaches a desirable level.

Next, the PH features are explored for repulsive forces. PH defines the elements with the highest persistence to be the "most important". Therefore, we typically start by looking for high persistence bars with a higher subset ratio and/or bars that split between different categories for labeled data. Hovering over a bar first informs the user if the partitioning may be interesting. Examples of interesting partitions include, but are not limited to, partitions that are mixed together (see Fig. 7(a)) or a cluster of nodes already spatially co-located that the user would like to move away from the rest of the graph (see Fig. 7(b)). If the user finds the partition interesting, the repulsive force is enabled by clicking. Typically, we found a good layout was achieved by selecting 5-10 bars for repulsion, although this is by no means a limit. Usually, the entire process took no more than a few minutes to sufficiently investigate and fine-tune the graph layout.

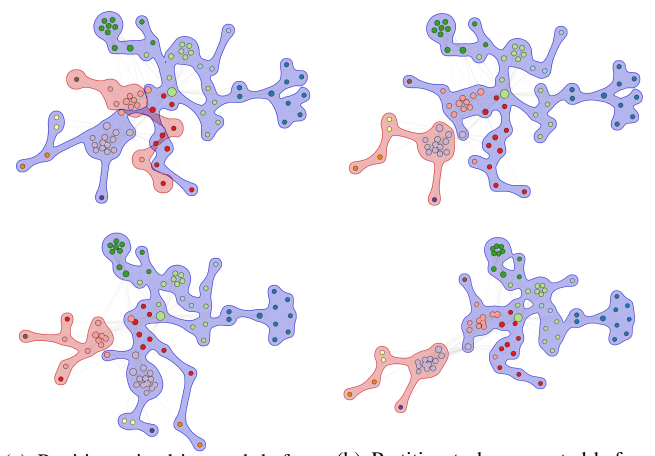

(a) Partition mixed in graph before (top) and after (bottom) repulsion
(b) Partition to be separated before (top) and after (bottom) repulsion

Fig. 7. Two scenarios for the Lés Miserables graph are shown where repulsion may be considered by a user (top) and the result of it (bottom).

Table 1. Quantitative Analysis of Results

Dataset	/V/	/E/	Figure	Layout	Figure	Computation	Figure	Computation	Layout	Contraction Eff.	Repulsion Eff.
			F-R		sfdp		Our Approach				
Les Mis.	77	254	1(a)	< 1 ms	n/s	52 ms	1(d)	< 1 ms	< 1 ms	-23%	134%
Bcsstk	110	364	8(a)	3 ms	8(b)	65 ms	8(d)	46 ms	3 ms	46%	201%
6-ary	9,331	9,330	8(a)	49 ms	8(b)	3,514 ms	8(d)	1.97 s	81 ms	63%	1986%
Barbell	150	2,501	8(a)	3 ms	8(b)	95 ms	8(e)	29 ms	4 ms	30%	411%
Lobster	300	299	8(a)	4 ms	8(b)	116 ms	8(e)	62 ms	4 ms	50%	363%
Senate	101	5,048	n/s	4 ms	n/s	110 ms	12(c)	36 ms	5 ms	42%	94%
Madrid	70	243	n/s	4 ms	n/s	88 ms	11(b)	46 ms	3 ms	-24%	212%
			F-R		Modular Cluster		Our Approach				
Airport	2,896	15,645	9(a)	15 ms	9(c)	128 ms	9(d)	3.81 s	35 ms	-12%	231%
Science	554	2,276	9(a)	7 ms	9(c)	846 ms	9(e)	281 ms	7 ms	-7%	706%
Collab.	379	914	9(a)	7 ms	9(c)	26 ms	9(d)	121 ms	5 ms	14%	544%
CalTech	762	16,651	n/s	7 ms	10(d)	110 ms	10(c)	314 ms	14 ms	29%	399%
Smith	2,970	97,133	9(a)	86 ms	9(c)	647 ms	9(e)	2.10 s	38 ms	-239%	426%

n/s: Not shown

6 Results

To evaluate using PH to guide force-directed layouts, we examine 12 datasets using hand-tuned layouts (see Table 1). Our evaluation considers the following:

- We examine the scalability of our approach to quickly calculate PH and update the graph layout fast enough to support interactive visualization, using graphs up to 10,000 nodes or 100,000 edges.
- We measure the quality of the layouts produced by our approach.
- We compare the layouts produced by our approach to popular force-directed layout methods and clustering techniques.
- We present three brief case studies on real-world data.

6.1 Performance

The accompanying software is implemented using Processing [2]. All calculations are performed on the CPU. The force calculations are multithreaded. The live footage from the accompanying video was produced on a 2017 MacBook Pro with a 3.1 Ghz i5 processor to demonstrate the interactivity of the interface.

Once the data is loaded, the 0-dimensional PH features are extracted. The MST calculation is a variation of Kruskal's algorithm performed on the metric space. It is implemented using disjoint sets taking $O(|E|\alpha(|V|))$, where α is the inverse Ackermann function [16], an extremely slow growing function. After the MST is found, determining the node subsets (see Sect. 4.2) takes $O(|V|)$.

For the F-R force-directed layout, we use the Barnes-Hut approximation [8] for repulsive forces and standard pairwise springs. The total cost is $O(|V|\log|V|+|E|)$ per iteration. In addition, contraction forces are single pairwise springs. The worst case total cost is $O(|V|)$, if all PH features are selected for contraction. The repulsion force can be costly. Each force that is applied requires an additional run of the Barnes-Hut algorithm. Since the number of repulsive forces is usually small, the expected run time is $O(|V|\log|V|)$ with the worst case $O(|V|^2\log|V|)$, when all PH features are set to repulse.

To improve interactivity of the visualization on larger datasets, certain noncritical features are disabled. For example, edge bundling is automatically disabled when the number of edges is greater than 500. In addition, bubble sets are disabled when the number of nodes is greater than 100. Instead, the graph nodes are surrounded by a halo, which is colored according to the set they belong to.

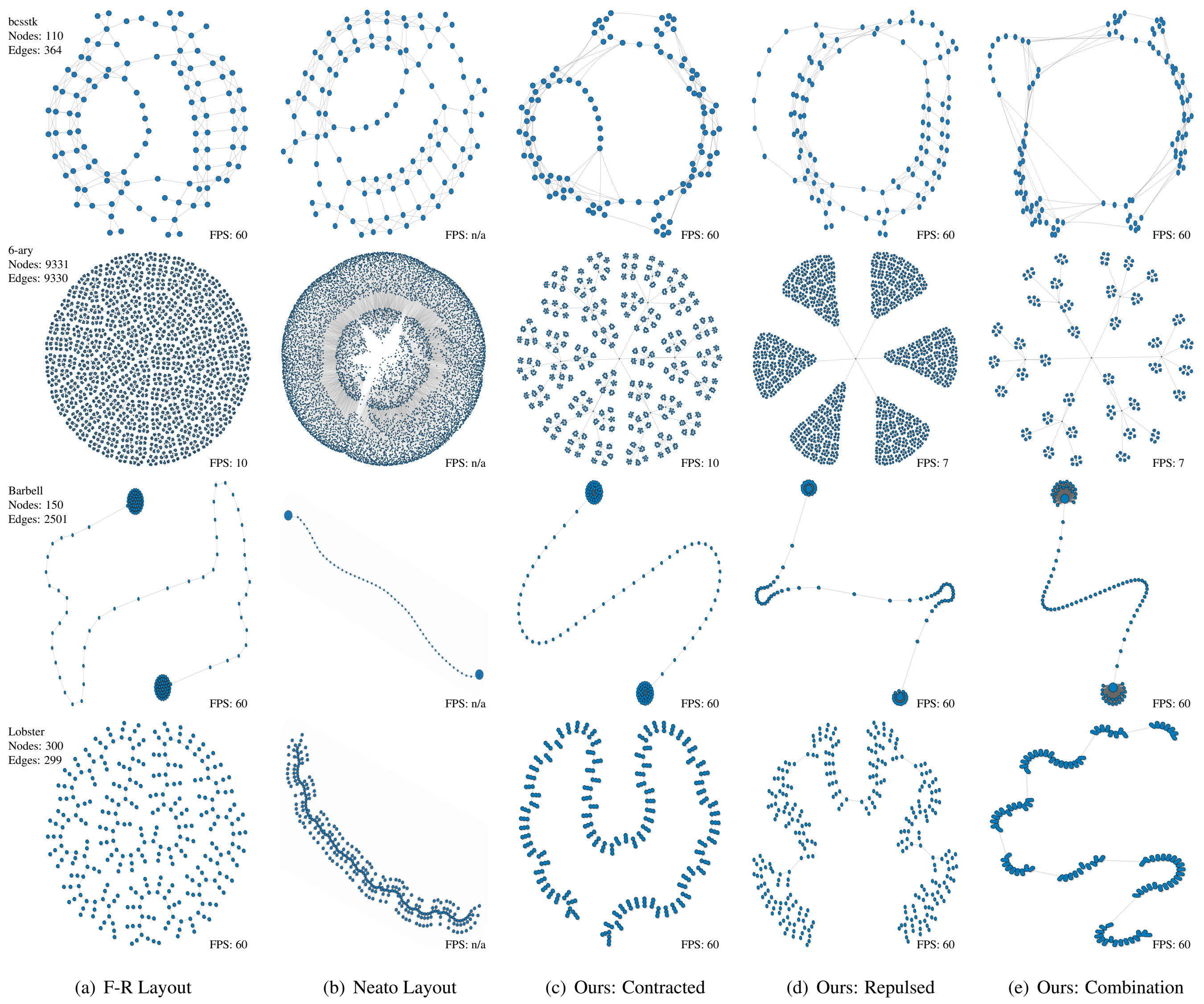


Fig. 8. Illustration of our approach on the synthetic graph examples: (a) a Fruchterman-Reingold (F-R) force-directed layout, hand-tuned; (b) a Neato layout [72] generated by Graphviz [29] using Jaccard index for edge weights; (c) our approach: only contraction is applied; (d) our approach: only repulsion is applied; and (e) our approach: both contraction and repulsion are applied.

To show that the performance of our approach is *comparable* to other techniques, we report computation times in Table 1. For our approach, the category *Computation* is the time needed to calculate the PH and determine the node subsets. For Neato [72], it is the time to converge. For hierarchical clustering, it is the time to compute the entire hierarchy. These computations occur only once, when the data is loaded. The category *Layout* is the time in milliseconds (ms) needed for one iteration of the force-directed layout calculation. Our implementation runs one iteration per rendering frame. In addition, many of our examples include frame rate, reported as frames per second (FPS). This number is generally less relevant, as it includes the extra costs of rendering, edge bundling, etc., which are mostly fixed costs.

The results demonstrate that, for all datasets, our approach has the scalability necessary to be utilized in interactive visualization. The PH calculations take at most a few seconds, and the time required for most layouts is less than 10 ms; for the larger graphs tested, it is always less than 100 ms.

6.2 Layout Quality

To assess the responsiveness of our approach, we introduce a measure specifically designed to quantify how well the resulting layouts reflect user intentions in emphasizing or de-emphasizing selected PH features. The approach works by comparing the PH of the embedded graph nodes of the layout before and after user modification. The approach starts by calculating the PH of the nodes of the *source* F-R force-directed layout (without any contraction or repulsion) using Euclidean distance between nodes[4]. Then, the PH of the user-selected *target* layout is calculated similarly.

Given a source and target, we extract the set of bars from each that are selected by the user for contraction (C) and repulsion (R). The persistence of those bars in the source and target is P_S and P_R, respectively. The effect of contraction (E_C) and repulsion (E_R) are calculated as:

$$E_C = \frac{1}{|C|}\sum_{x\in C}\frac{P_S(x)-P_T(x)}{P_S(x)} \quad , \quad E_R = \frac{1}{|R|}\sum_{x\in R}\frac{P_T(x)-P_S(x)}{P_S(x)}$$

The results appear in the final two columns of Table 1, comparing the source layout (F-R layout listed in the table) to the target layout (our approach). Intuitively, this measure quantifies how much on average the features of a layout have been contracted or repulsed in the target layout

[4]Note, this calculation does *not* consider the connectivity of the graph.

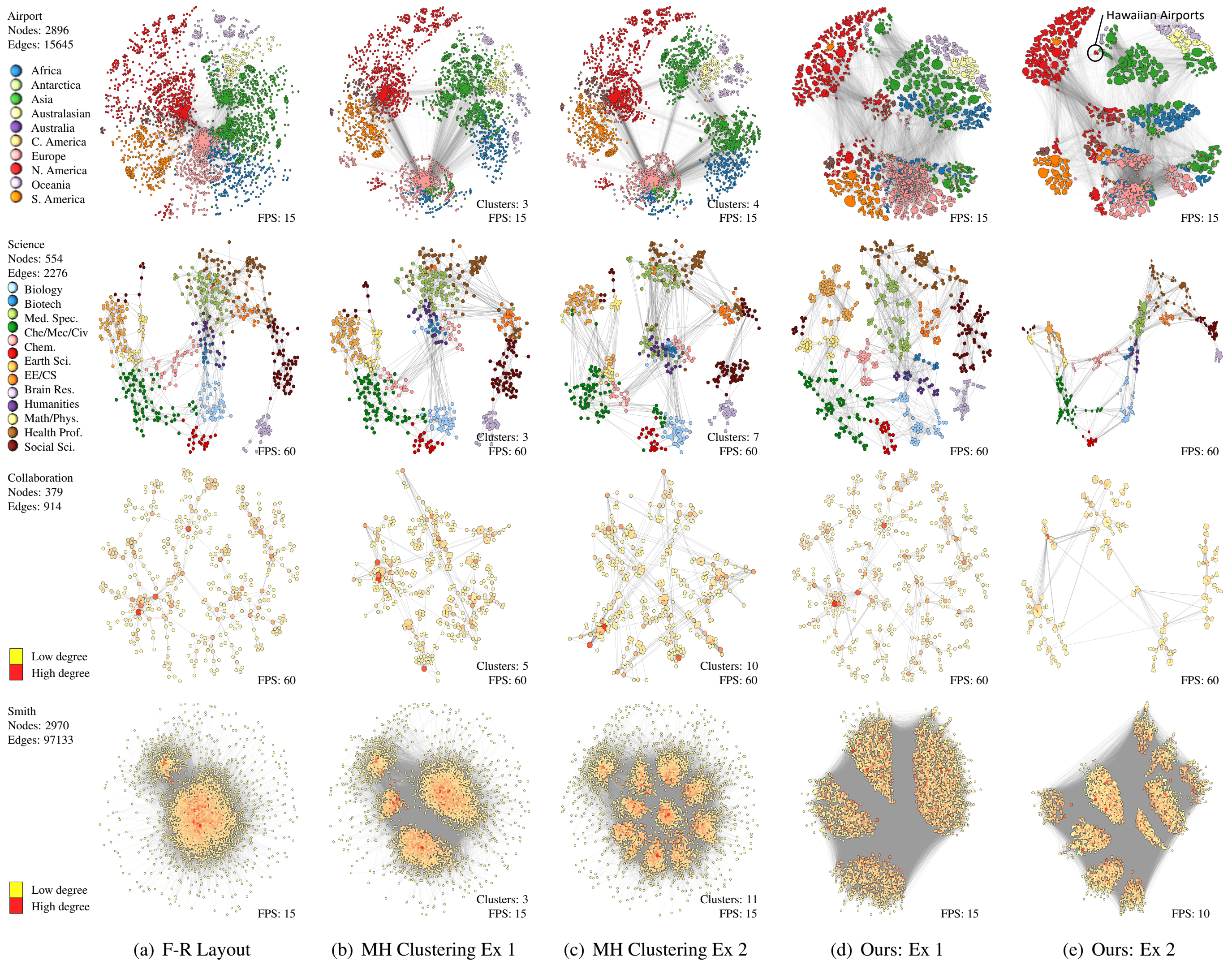


Fig. 9. Illustration of our approach on a series of graph examples: (a) Fruchterman-Reingold force-directed layout, hand-tuned; (b-c) 2 examples of modularity hierarchical clustering with cluster number selected manually; (d-e) 2 examples using our approach.

relative to the source layout—negative is undesirable; zero means no impact; positive is desirable and the larger the better.

For the most part, our approach shows a substantial positive impact from the applied contraction and repulsion, indicating the user's intentions have been reflected in the layout. The notable exceptions are that some datasets have a negative contraction effect. For these examples, negativity does not mean that the contraction is entirely ineffective—in reality, the repulsive effect just *overpowers* the contractive effects on some parts of the layout, leading to an average effect that is negative.

6.3 Comparison to Other Techniques

6.3.1 Comparison to Popular Force-Directed Layout Methods

We test our method on four synthetic unweighted graphs in Fig. 8 to compare the layouts of F-R force-directed layout, Neato [72] from Graphviz [29], and our approach. *Bcsstk* is taken from the UF Sparse Matrix Collection [18]; *6-ary*, *Barbell*, and *Lobster* are generated using NetworkX [40]. *Bcsstk* is a symmetric stiffness graph containing 110 nodes and 364 edges. *6-ary* is a balanced tree of depth five containing 9331 nodes and 9330 edges. *Barbell* is a simple graph connecting two complete subgraphs of 50 nodes each with a bridge of 50 nodes, totaling 150 nodes and 2501 edges. *Lobster* is a tree with the property that the removal of leaves results in a caterpillar graph [38]. For all layout methods, each graph has weights applied by using the method described in Sect. 4.3 with the following neighborhood size: Bcsstk: 1-hop; *6-ary*: 2-hop; *Barbell*: 1-hop; and *Lobster*: 1-hop. Fig. 8 shows three examples of our approach using contraction only, repulsion only, and a combination of both.

6.3.2 Comparison to Hierarchical Clustering

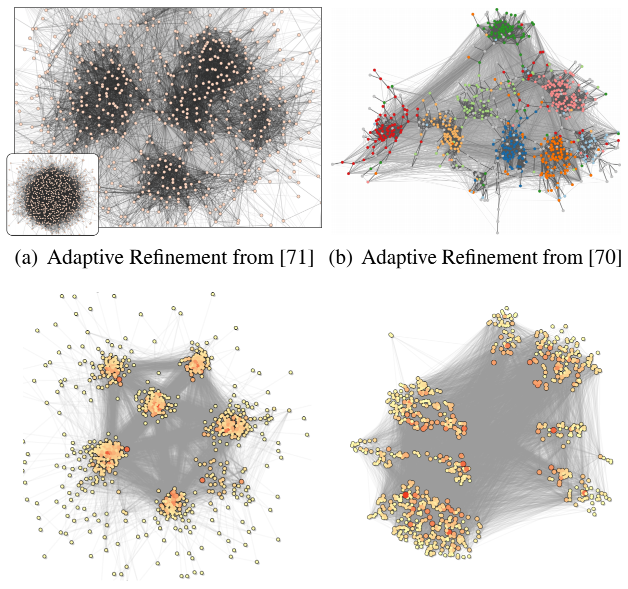
Next, we compare our approach to modularity-based hierarchical clustering. For a survey of graph clustering, including modularity, see [82].

We use a greedy approach to form the clusters [67, 69] available in Graphviz [29], since the optimal version is NP-hard [14]. The algorithm begins by initializing each node in the graph into its own cluster. The 2 clusters whose merging will cause the largest increase in modularity are then combined. The weighted modularity is calculated as [62]:

$$Q_w = \frac{1}{2w_s} \sum_{ij} \left[w_{ij} - \frac{w_i w_j}{2w_s} \right] \delta_{c_i, c_j},$$

where $2w_s = \sum_{ij} w_{ij}$, $w_i = \sum_j w_{ij}$, and Kronecker delta δ_{c_i,c_j} is 1 if both i and j are in the same community, 0 otherwise. This process is repeated until a single cluster remains.

To reflect the clustering in the graph layout, we increase the spring resting lengths between nodes of different clusters. Examples can be seen in Fig. 9. Table 1 shows the time to compute the clustering.



(a) Adaptive Refinement from [71] (b) Adaptive Refinement from [70]

(c) Modularity Hierarchical Clustering (d) Our Approach

Fig. 10. Caltech datasets containing 762 nodes and 16,651 edges are compared using (a,b) 2 adaptive refinement techniques, (c) modularity hierarchical clustering, and (d) our approach.

2011 International Airports (Airport) (Fig. 9 row 1) is taken from Openflights.org, where each node is an international airport, labeled by continental region, and edges are weighted by the number of routes between two airports. The largest connected component from this dataset contains 2,896 nodes and 15,645 edges. Using a combination of contraction and repulsion forces, Fig. 9 columns 4 and 5 reveal a split among Western, Eastern, and Central airports. One notable cluster is formed at the split of North America and Asia, containing several Hawaiian airports. The results in Fig. 9 columns 2 and 3 show similar clustering insights, but Hawaiian airports are not directly visible.

UCSD Map of Science (Science) (Fig. 9 row 2) [10] is a map of 554 subdisciplines in science, represented as nodes, and cross-disciplinary coauthorship as the 2,276 edges. Our method, when applied to this particular dataset, results in a graph that retains the overall shape of the F-R layout with the addition of clustering communities that share similar disciplines. On the other hand, the clustered versions of the graph (columns 2 and 3) highlight the clustering structure but lose the context (ground truth labels).

The Collaboration Science Network (Collaboration) (Fig. 9 row 3) [65] is a coauthorship network with 379 nodes–publishing scientists in network theory–and 914 edges–a connection of two authors appearing on the same paper. The emphasis of the graph is to identify communities of collaborators. With the majority of bars selected for contraction (column 4 and 5), subcommittees are brought more tightly together, better revealing the overall graph shape. Clustering on the other hand (columns 2 and 3) produces results that we found difficult to interpret.

Smith College (Smith) (Fig. 9 row 4) is from the Facebook100 dataset [85] and shows the social relations of students at Smith College. The graph has 2,970 nodes and 97,133 edges. The original graph contained attributes, including dormitory, gender, etc., but we were unable to locate this data. For this example, the key emphasis is on the scalability of our approach (see Table 1).

Caltech (Fig. 10) is a dataset of the social links at the California Institute of Technology, also from the Facebook100 dataset [85]. The graph has 762 nodes and 16,651 edges. For this example, we compare against modularity hierarchical clustering and adaptive refinement techniques [70, 71]. Each of the techniques presents similar looking communities. Unfortunately, without the label data, other direct comparisons are difficult.

6.4 Case Studies

6.4.1 Lés Miserables

The *Lés Miserables Co-occurrence* network has 77 nodes and 254 edges, where a node represents a character, and an edge is weighted by the number of scenes two characters share during any chapter of Victor Hugo's novel "Lés Miserables" [54]. The node classification comes from the primary group affiliation of the characters in the novel; the groups are named based upon our knowledge of those characters.

In Fig. 1(a), we show the F-R force-directed layout for this dataset. When selecting a combination of high persistence bars and contracting the remaining ones (see Fig. 1(e)), we reveal some of the key characters featured in the book, seen in Fig. 1(d). On two opposing sides of the layout are nodes Marius and a cluster around Valjean, two main characters, along with Éponine—a woman in love with Marius; Javert—the primary antagonist to Valjean; Cosette—Valjean's daughter and Marius' lover; and Toussaint—a motherly-figure assisting in raising Cosette through her childhood.

6.4.2 Madrid Train Bombing

The Madrid Train Bombing dataset contains 70 nodes and 243 edges, where a node represents "individuals involved in the bombing of commuter trains in Madrid on March 11, 2004" [79]. Each group has been identified and colored based on whether the person was involved in previous terrorist acts and whether this person was a member of the Field Operations Group. A link is connected if two individuals were related prior to or during the bombing. Weight is calculated on an index between 1-4, where each of the following four parameters are summed per pair: trust–friendship (contact, kinship, links in the telephone center); ties to Al Qaeda and Osama Bin Laden; co-participation in training camps and/or wars; and co-participation in previous terrorist Attacks (September 11, Casablanca, etc.).

We begin by examining the layout in Fig. 11(a), which is drawn to replicate the original graph in Rogriguez's paper [79][5]. This is

[5]We suspect some form of force-directed layout was used originally, but that information is not documented.

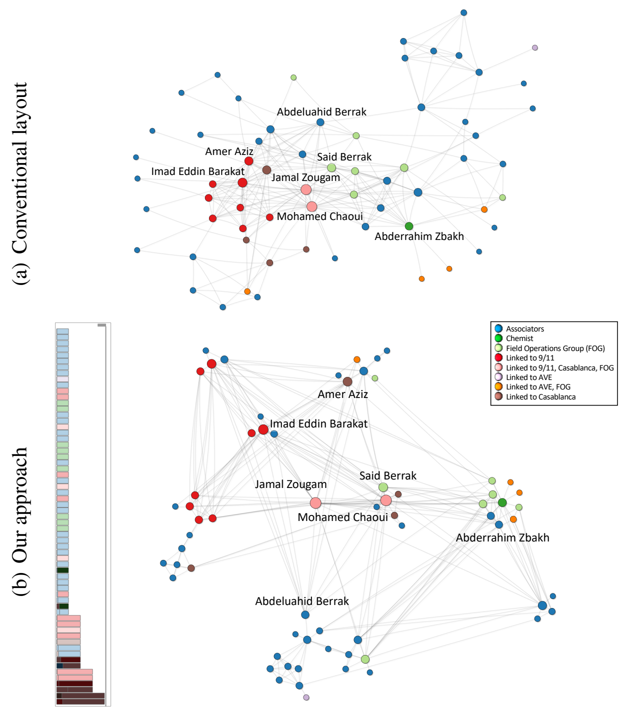
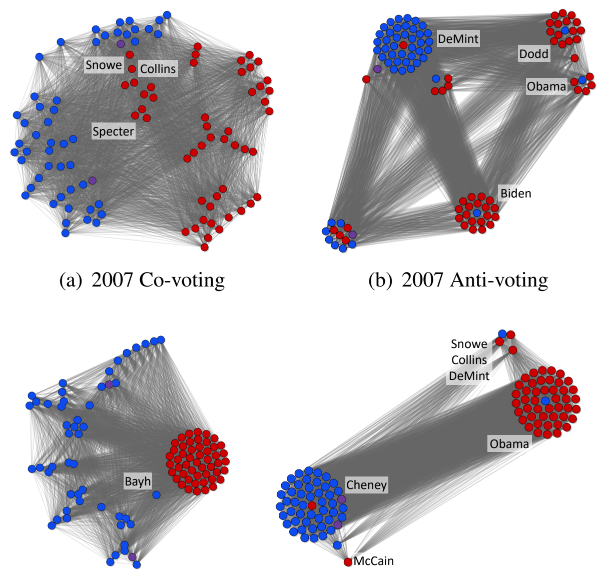


Fig. 11. Examples from the Madrid Train Bombing dataset: (a) the conventional layout, as recreated from the original paper [79]; (b) the final visualization using our layout highlighting key players in the network.



(a) 2007 Co-voting (b) 2007 Anti-voting

(c) 2008 Co-voting (d) 2008 Anti-voting

Fig. 12. Co- and anti-voting graphs for the US Senate in 2007 and 2008. Using a mixture of contracting and repulsive forces, these graphs show the role of major political figures during these timeframes. Color are Democrats: blue; Republicans: red; and independents: purple.

the complete network, incorporating all players with their respective binding ties. Although Rodriguez created this graph layout to highlight the central core of the network, it is difficult to identify whom he calls the "three most central players" involved in the Madrid bombing.

When selecting a number of bars for contraction and repulsion, shown in Fig. 11(b), the graph closely corresponds with the analysis provided in the paper. Jamal Zougam, Mohamed Chaoui, and Said Berrak are the three most central players in the Field Operations Group network, with all three central to the graph. Below Zougam is Abdeluahid Berrak, who was suspected to be responsible for recruiting new members to the Field Operations Group. Another central player highlighted in Rodriguez's analysis is Abderrahim Zbakh, known as "The Chemist". In the original layout, his node is not remarkable; yet, Rodriguez described Zbakh as "cementing the network" by uniting unacquainted members of various terrorist groups. Other significant players include Amer Aziz and Imad Eddin Barrakat: al Qaeda, operatives closely linked as facilitators in the September 11 attacks.

6.4.3 US Senate 2007 and 2008 Co- and Anti-voting

The *US Senate 2007 and 2008* datasets are complete co- and anti-voting graphs, both with 101 nodes (100 senators plus the Vice President) and 5,048 edges. This dataset is created using voting records provided by GovTrack [1] with weights corresponding to how frequently two senators vote together (or against each other).

The 2007 co-voting graph in Fig. 12(a) shows the normal partisan divide—Democrats voting with Democrats and Republicans voting with Republicans. Three figures stand out among the "centerist" group: Snowe, Collins, and Specter. Snowe and Collins are well known for voting across partisan lines. Specter switched from the Republican to Democrat party in 2009. The 2007 anti-voting graph in Fig. 12(b) highlights whom each person was most likely to vote against. The graph shows 4 clusters that isolate individuals of the opposite party: DeMint, Dodd, Biden, and Obama.

In 2008, the presidential election was in full swing, and the co-voting graph in Fig. 12(c) shows that partisanship reigned. The Republicans in particular voted together. Bayh is the Democrat standout who appeared most aligned with Republicans. The 2008 anti-voting graph in Fig. 12(d) highlights the politics of the election. On one side, the Republicans were running against Obama. On the Democrat side, Democrats focused on running against McCain and the Vice President, Cheney. The vote against Cheney is likely an artifact of how Congress works—the Vice President votes needs to only when there is a tie. The vote against McCain fits, since he was the Republican nominee.

7 Discussion

Transferability to Other Force-Directed Layout Algorithms. F-R force-directed layout was used as our reference layout. Since our modification uses only additional repulsive and spring forces, by and large, other force-directed layout variants, such as sfdp [47] or FM^3 [39], should be able to adapt our approach to their algorithms and see benefits similar to those we have demonstrated.

Relationship to Hierarchical Clustering. Calculating 0-dimensional PH features has a strong relationship to finding hierarchical clusters. However, the main differentiating factor is the treatment of features in PH. For example, PH sees low-weight edges as "noise", and hence collapsing them makes sense. At the same time, it sees high-weight edges as signals, and hence separating such features is meaningful.

(Semi-)Automatically Selecting Features. Given the procedure defined in Sect. 5.4, it is possible to imagine that a (semi-)automatic heuristic-based approach could be used to initialize the contraction and repulsion. We did not pursue such an approach for two reasons. First, the amount of interaction saved by such an approach would not be that significant. In our experience, exploring the contraction and repulsion options takes only a few minutes. Second and more importantly, the interaction process provides intuition about the graphs. This intuition is critical in selecting a final graph layout.

Disconnected Graphs. In the case of a disconnected graph, PH calculations work without any modification by recognizing that disconnected components never merge. After that, our approach considers each connected component in the graph separately.

Limitations. Our approach is not without limitations. The first problem is that force-directed layouts are, by their nature, over-constrained. Adding additional forces can exacerbate this problem. We see this in a number of examples that have negative results for contraction effectiveness (see Sect. 6.2). Second, there is the possibility for selective-engineering of the layout. Users can choose to ignore or select any PH feature to repulse, which could lead to intentionally ignoring important PH features in the final layout. Next, the worst case number of interactions needed can be quite large. To test every possible contraction and repulsion requires $2|N|$ interactions, not to mention the cost of testing combinations. Fortunately, we find the actual interaction process to be fairly quick for high-quality results (see Sect. 5.4). Finally, our approach assumes that a unique MST exists for an input graph. If such an assumption does not hold (for example in Sect. 6.4.2), an arbitrary tree from the set of MSTs is selected. Selecting an *optimal* MST using various constraints deserves future investigation.

8 Conclusion

We have presented a new approach to graph drawing that uses PH to interactively modify a force-directed layout. The approach provides a flexible interface for selecting layouts that highlight features of the graph, as defined by PH. In the future, we would like to look at the potential of using higher-dimensional PH features to control graph drawing. For example, 1-dimensional PH features encode tunnels within metric spaces, which would be useful in constraining similar structures in the graph. However, with higher-dimensional PH features, finding the set of points that generate these features is a nontrivial problem.

Acknowledgments

We thank the reviewers for their valuable feedback. This work was supported in part by National Science Foundation grants IIS-1513616 and DBI-1661375, CRA-W Collaborative Research Experiences for Undergraduates (CREU) program, DARPA CHESS FA8750-19-C-0002, and an NVIDIA Academic Hardware Grant.

References

[1] GovTrack. `https://www.govtrack.us`. Accessed: 2019-03-01.

[2] Processing. `https://processing.org/`. Accessed: 2019-03-01.

[3] D. Archambault, T. Munzner, and D. Auber. Grouse: Feature-based, steerable graph hierarchy exploration. In *Proceedings of Eurographics / IEEE VGTC Symposium on Visualization*, vol. 2007, pp. 67–74, 2007.

[4] D. Archambault, T. Munzner, and D. Auber. Topolayout: Multilevel graph layout by topological features. *IEEE transactions on visualization and computer graphics*, 13(2), 2007.

[5] D. Archambault, T. Munzner, and D. Auber. Grouseflocks: Steerable exploration of graph hierarchy space. *IEEE Transactions on Visualization and Computer Graphics*, 14(4):900–913, 2008.

[6] 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, 2017.

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

[8] J. Barnes and P. Hut. A hierarchical $O(N \log N)$ force-calculation algorithm. *Nature*, 324(6096):446, 1986.

[9] M. Bastian, S. Heymann, and M. Jacomy. Gephi: an open source software for exploring and manipulating networks. In *International Conference on Web and Social Media*, pp. 361–362, 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] U. Brandes and C. Pich. Eigensolver methods for progressive multidimensional scaling of large data. In *Graph Drawing*, pp. 42–53. Springer, 2007.

[12] C. J. Carstens and K. J. Horadam. Persistent homology of collaboration networks. *Mathematical problems in engineering*, 2013, 2013.

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

[14] S. Clémençon, H. De Arazoza, F. Rossi, and V. C. Tran. Hierarchical clustering for graph visualization. *arXiv e-prints*, arXiv:1210.5693, 2012.

[15] C. Collins, G. Penn, and S. Carpendale. Bubble sets: Revealing set relations with isocontours over existing visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1009–1016, 2009.

[16] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to algorithms*. MIT press, 2009.

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

[18] T. A. Davis and Y. Hu. The University of Florida sparse matrix collection. *ACM Transactions on Mathematical Software*, 38(1):1, 2011.

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

[20] K. Dinkla, M. J. Van Kreveld, B. Speckmann, and M. A. Westenberg. Kelp diagrams: Point set membership visualization. *Computer Graphics Forum*, 31(3pt1):875–884, 2012.

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

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

[23] C. Dunne and B. Shneiderman. Improving graph drawing readability by incorporating readability metrics: A software tool for network analysts. Technical report, University of Maryland, HCIL Tech Report HCIL-2009-13, 2009.

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

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

[26] W. E, J. Lu, and Y. Yao. The landscape of complex networks. *arXiv e-prints*, arXiv:1204.6376, 2012.

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

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

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

[30] M. Fröhlich and M. Werner. Demonstration of the interactive graph visualization system da vinci. In *International Symposium on Graph Drawing*, pp. 266–269. Springer, 1994.

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

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

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

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

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

[36] H. Gibson, J. Faith, and P. Vickers. A survey of two-dimensional graph layout techniques for information visualisation. *Information Visualization*, 12:324–357, 07 2012.

[37] M. Girvan and M. E. Newman. Community structure in social and biological networks. *Proceedings of the national academy of sciences*, 99(12):7821–7826, 2002.

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

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

[40] 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.

[41] 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*, 2018.

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

[43] T. R. Henry and S. E. Hudson. Interactive graph layout. In *Proceedings of the 4th annual ACM symposium on User interface software and technology*, pp. 55–64. ACM, 1991.

[44] I. Herman, G. Melançon, and M. S. Marshall. Graph visualization and navigation in information visualization: A survey. *IEEE Transactions on Visualization and Computer Graphics*, 6(1):24–43, 2000.

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

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

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

[48] C. Hurter, A. Telea, and O. Ersoy. Moleview: An attribute and structure-based semantic lens for large element-based plots. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2600–2609, 2011.

[49] P. Jaccard. Étude comparative de la distribution florale dans une portion des alpes et des jura. *Bull Soc Vaudoise Sci Nat*, 37:547–579, 1901.

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

[51] D. A. Keim. Designing pixel-oriented visualization techniques: Theory and applications. *IEEE Transactions on Visualization and Computer Graphics*, 6(1):59–78, 2000.

[52] D. A. Keim, J. Schneidewind, and M. Sips. Scalable pixel based visual data exploration. *Pixelization Paradigm, Lecture Notes in Computer Science*, 4370:12–24, 2007.

[53] M. Khoury, Y. Hu, S. Krishnan, and C. Scheidegger. Drawing large graphs by low-rank stress majorization. *Computer Graphics Forum*, 31(3pt1):975–984, 2012.

[54] D. E. Knuth. *The Stanford GraphBase: a platform for combinatorial computing*, vol. 37. Addison-Wesley Reading, 1993.

[55] Y. Koren. On spectral graph drawing. In *Computing and Combinatorics*, pp. 496–508. Springer, 2003.

[56] Y. Koren, L. Carmel, and D. Harel. Ace: A fast multiscale eigenvectors

computation for drawing huge graphs. In *IEEE Symposium on Information Visualization*, pp. 137–144, 2002.

[57] 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.

[58] 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. *International Conference on Medical Image Computing and Computer Assisted Intervention*, pp. 302–309, 2011.

[59] H. Lee, M. K. Chung, H. Kang, B.-N. Kim, and D. S. Lee. Discriminative persistent homology of brain networks. *IEEE International Symposium on Biomedical Imaging: From Nano to Macro*, pp. 841–844, 2011.

[60] 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.

[61] H. Lee, H. Kang, M. K. 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.

[62] X. Lou and J. A. Suykens. Finding communities in weighted networks through synchronization. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 21(4):043116, 2011.

[63] F. McGee and J. Dingliana. An empirical study on the impact of edge bundling on user comprehension of graphs. In *Proceedings of the International Working Conference on Advanced Visual Interfaces*, pp. 620–627. ACM, 2012.

[64] 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.

[65] M. Newman. Collaborations Between Network Scientists. `http://www-personal.umich.edu/~mejn/centrality/poster.pdf`. Accessed: 2019-03-01.

[66] M. E. Newman. Ego-centered networks and the ripple effect. *Social Networks*, 25(1):83–95, 2003.

[67] M. E. Newman. The structure and function of complex networks. *SIAM review*, 45(2):167–256, 2003.

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

[69] A. Noack and R. Rotta. Multi-level algorithms for modularity clustering. In *International Symposium on Experimental Algorithms*, pp. 257–268. Springer, 2009.

[70] A. Nocaj, M. Ortmann, and U. Brandes. Untangling hairballs. In *Graph Drawing*, pp. 101–112. Springer, 2014.

[71] A. Nocaj, M. Ortmann, and U. Brandes. Adaptive disentanglement based on local clustering in small-world network visualization. *IEEE Transactions on Visualization and Computer Graphics*, 22(6):1662–1671, 2016.

[72] S. C. North. Drawing graphs with neato. *NEATO User manual*, 11:1, 2004.

[73] D. Oelke, H. Janetzko, S. Simon, K. Neuhaus, and D. A. Keim. Visual boosting in pixel-based visualizations. *Computer Graphics Forum*, 30(3):871–880, 2011.

[74] G. Petri, M. Scolamiero, I. Donato, and F. Vaccarino. Networks and cycles: A persistent homology approach to complex networks. *Proceedings of the European Conference on Complex Systems 2012, Springer Proceedings in Complexity*, pp. 93–99, 2013.

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

[76] 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.

[77] H. C. Purchase. Metrics for graph drawing aesthetics. *Journal of Visual Languages & Computing*, 13(5):501–516, 2002.

[78] 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*, 2017.

[79] J. A. Rodríguez. The march 11 th terrorist network: In its weakness lies its strength, 2005. Data available at: `http://moreno.ss.uci.edu/data`.

[80] K. Ryall, J. Marks, and S. Shieber. An interactive constraint-based system for drawing graphs. In *Proceedings of the 10th annual ACM symposium on User interface software and technology*, pp. 97–104. ACM, 1997.

[81] M. Sarkar and M. H. Brown. Graphical fisheye views of graphs. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 83–91. ACM, 1992.

[82] S. E. Schaeffer. Graph clustering. *Computer Science Review*, 1(1):27–64, 2007.

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

[84] C. Tominski, J. Abello, F. Van Ham, and H. Schumann. Fisheye tree views and lenses for graph visualization. In *Proceedings of the 10th IEEE International Conference on Information Visualisation*, pp. 17–24, 2006.

[85] A. L. Traud, E. D. Kelsic, P. J. Mucha, and M. A. Porter. Comparing community structure to characteristics in online collegiate social networks. *SIAM review*, 53(3):526–543, 2011.

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

[87] S. Van den Elzen and J. J. Van Wijk. Multivariate network exploration and presentation: From detail to overview via selections and aggregations. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2310–2319, 2014.

[88] 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.

[89] Y. Wang, Y. Wang, Y. Sun, L. Zhu, K. Lu, C.-W. Fu, M. Sedlmair, O. Deussen, and B. Chen. Revisiting stress majorization as a unified framework for interactive constrained graph visualization. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):489–499, 2018.

[90] 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 Transaction on Visualization and Computer Graphics*, 25(1):566–575, 2019.

[91] N. Wong, S. Carpendale, and S. Greenberg. Edgelens: An interactive method for managing edge congestion in graphs. In *IEEE Symposium on Information Visualization 2003 (IEEE Cat. No. 03TH8714)*, pp. 51–58. IEEE, 2003.