Topological Simplifications of Hypergraphs

Youjia Zhou, Archit Rathore, Emilie Purvine, Bei Wang

Abstract—We study hypergraph visualization via its topological simplification. We explore both vertex simplification and hyperedge simplification of hypergraphs using tools from topological data analysis. In particular, we transform a hypergraph into its graph representations, known as the line graph and clique expansion. A topological simplification of such a graph representation induces a simplification of the hypergraph. In simplifying a hypergraph, we allow vertices to be combined if they belong to almost the same set of hyperedges, and hyperedges to be merged if they share almost the same set of vertices. Our proposed approaches are general and mathematically justifiable, and put vertex simplification and hyperedge simplification in a unifying framework.

Index Terms—Hypergraph simplification, hypergraph visualization, graph simplification, topological data analysis

1 Introduction

Data that capture multiway relationships within a group of entities are ubiquitous in science and engineering. In social networks, apart from pairwise "likes" and friendships, people form multiway groups or clubs based on common interests. In computer networking, multiple IP addresses that resolve to the same domain name (e.g., `www.google.com`) form a group relationship [1]. In biological applications, collections of proteins comprise pathways that lead to a product or a change in a cell, and groups of genes make up ontology terms and contribute toward a shared molecular function, cellular component, or biological process [2], [3].

In these cases, exploration of the data can help people discover interesting patterns, subsets, and entities. Graphs (or networks) are a central way to model data that come in the form of pairwise (or binary) relationships. However, graphs cannot natively represent multiway relationships without moving to bipartite structures or employing reification strategies. Instead, hypergraphs provide a way to capture these multiway interactions. A *hypergraph*, $H = (V, E)$, consists of a set of vertices, $V = \{v_1, \cdots, v_n\}$, together with a collection of *hyperedges*, $E = \{e_1, \cdots, e_m\}$, each of which is a subset of vertices $e_i \subseteq V$.

Visualization can be a useful tool to explore data modeled as a hypergraph. There are various visual encodings for a hypergraph. We give a number of examples in Fig. 1, which includes an Euler diagram, bipartite graph, bubble sets, and rainbow box-based visualizations together with their hybrids. For instance, Fig. 1a visualizes a hypergraph with four vertices (as black nodes) and three hyperedges (as colored convex hulls).

Visualizing large graphs remains challenging as naive visualization often produces "hairballs" of little information content due to visual clutter. Such a problem is further compounded when dealing with hypergraphs, even ones with a moderate number of vertices and a small number of hyperedges. As the number of vertices grows and hyperedges become more interconnected, it becomes increasingly difficult to turn a naive visualization into

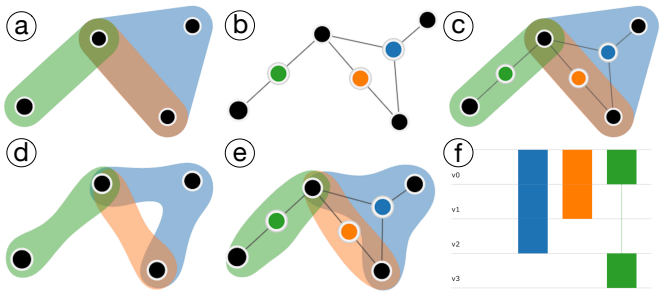


Fig. 1: Examples of visual encodings of a hypergraph. (a) Euler diagram: black nodes are vertices; colored convex hulls are hyperedges. (b) Bipartite graph: each colored node represents a hyperedge, which connects with all its vertices in black. (c) Euler-bipartite hybrid visualization obtained by superimposing the Euler diagram with the bipartite graph. (d) Bubble sets: black nodes are vertices; colored bubble sets (implicit surfaces) are hyperedges. (e) Bubble-bipartite hybrid visualization obtained by applying bubble sets to the vertices in the bipartite graph. (f) Rainbow box: each row is a vertex; each column (box) is a hyperedge.

insights. Fig. 4a illustrates a naive hypergraph visualization where vertices represent genes and hyperedges consist of pathways from the Hallmark collection within the Molecular Signatures Database (MSigDB) [4], [5]. Vertices on the periphery are shown to belong to a single hyperedge, but hyperedge memberships of those vertices closer to the center are more difficult to interpret.

To reduce visual clutter *and* to obtain compact representation for analysis, we might want to reduce the size of a hypergraph while preserving its core structure. To this end, one might want to apply *vertex collapse* and *hyperedge collapse*; a common algorithm for hypergraph simplification as part of the HyperNetX package [6]. As illustrated in Fig. 3, vertex collapse combines vertices that belong to the same set of hyperedges into a single "supervertex" (visualized by a concentric circle glyph), whereas hyperedge collapse merges hyperedges that share the same set of vertices into a "superhyperedge" (visualized by a pie chart). See Fig. 4b for the simplified biological pathway hypergraph after vertex collapse.

In this paper, we relax the notions of vertex collapse and hyperedge collapse mathematically by allowing vertices to be combined if they belong to *almost the same* set of hyperedges,

- *Youjia Zhou, Archit Rathore, Bei Wang are with Scientific Computing & Imaging (SCI) Institute, University of Utah, Salt Lake City, UT, 84112. E-mails: {zhou325, archit, beiwang}@sci.utah.edu.*
- *Emilie Purvine is with Pacific Northwest National Laboratory, Seattle, WA, 98109. E-mail: Emilie.Purvine@pnnl.gov.*

Manuscript received April 19, 2005; revised August 26, 2015.

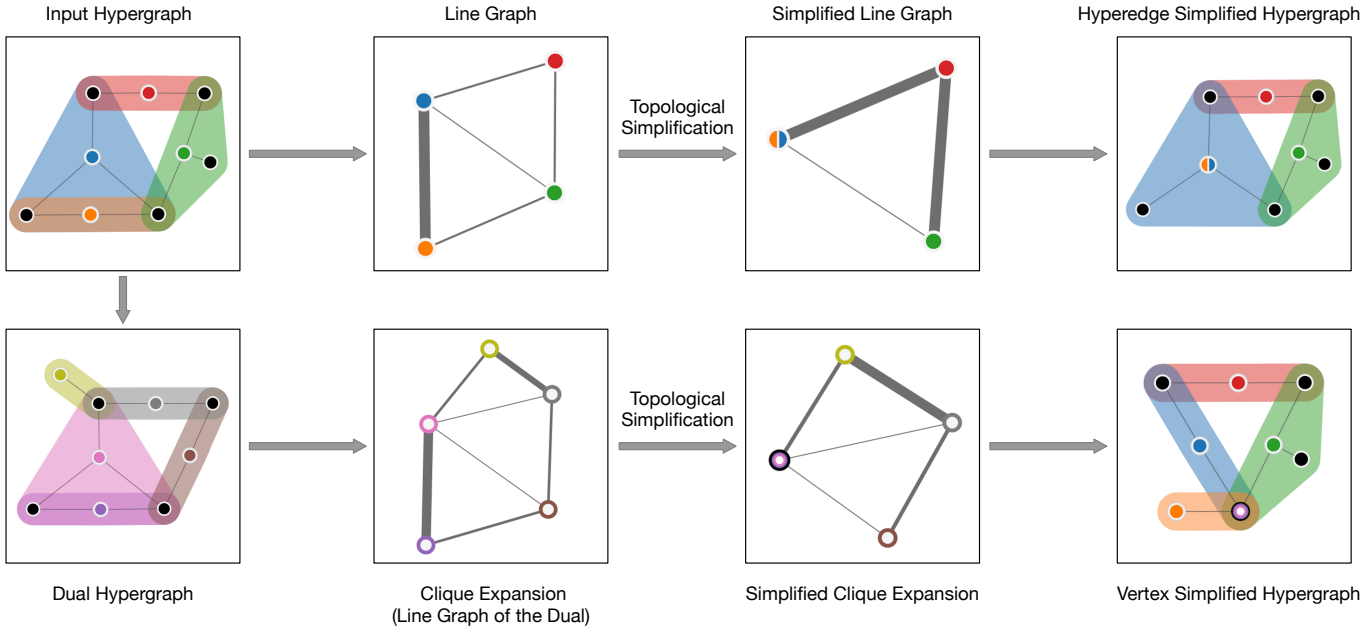


Fig. 2: An overview of topological simplification of hypergraphs. Top: hyperedge simplification. Bottom: vertex simplification.

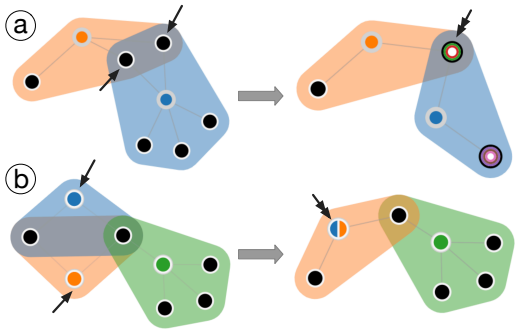


Fig. 3: A vertex collapse (a) and a hyperedge collapse (b). In (a), the concentric circle glyph (pointed by the double arrow) shows the merging of two vertices (pointed by single arrow), forming a supervertex. In (b), a pie chart (double arrow) shows the merging of the blue and orange hyperedges, forming a superhyperedge.

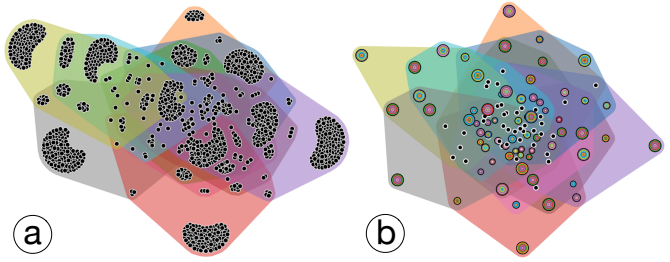


Fig. 4: (a) A biological pathway hypergraph with $|V| = 1,316$ and $|E| = 10$. Black nodes are vertices, colored convex hulls are hyperedges. (b) The simplified hypergraph after vertex collapse.

and hyperedges to be merged if they share *almost the same* set of vertices. The former is referred to as the *vertex simplification* (or approximate vertex collapse), and the latter is referred to as the *hyperedge simplification* (or approximate hyperedge collapse). Using tools from topological data analysis, in particular, barcodes [7] that capture the topology of hypergraphs, we perform topological simplification of hypergraphs. Our approach is *general* as it generalizes vertex and hyperedge collapses to their approximate versions. As we simplify a hypergraph in a way that removes topological noise as determined by its barcode, our approach is also *mathematically justified* by the stability of barcodes [8].

Pipeline overview. Our pipeline is illustrated in Fig. 2. To enable hyperedge simplification (Fig. 2 top), we first convert a hypergraph H into a graph representation called the (weighted) *line graph* [9], which captures the similarities among hyperedges. The line graph $L(H)$ of H is a graph whose vertex set corresponds to the set of hyperedges of H; two vertices are adjacent in $L(H)$ if their corresponding hyperedges have a nonempty intersection in H. We then perform a topological simplification of the line graph. The simplified line graph induces a hyperedge simplification of the input hypergraph.

On the other hand, to enable vertex simplification (Fig. 2 bottom), we first consider a (weighted) *clique expansion* [10] of an input hypergraph. The clique expansion $Q(H)$ of H constructs a graph from a hypergraph by replacing each hyperedge with a clique among its vertices. We then perform a topological simplification of the clique expansion. The line graph and clique expansion are related through the concept of a *dual hypergraph*, which swaps the roles of vertices and hyperedges. A clique expansion captures the similarities among vertices; it is known to be the line graph of the dual of a hypergraph. The simplified clique expansion in turn induces the vertex simplification of the input hypergraph. Using barcode-guided topological simplification, we formalize both vertex and hyperedge simplification in a *unifying* way.

Contributions. We summarize our contributions below:

- We introduce a topology-based method for simplifying complex hypergraph data in a mathematically principled way. Through the notion of line graph and clique expansion, our method provides a unifying framework for approximate vertex and hyperedge collapse.

- This approximate collapse not only reduces visual clutter but also provides a compact representation that retains important structural information for downstream analysis. It provides greater flexibility in hypergraph simplification in comparison to state-of-the-art strict vertex and hyperedge collapse.
- We demonstrate the utility of our simplification method with real-world examples. We compare with strict vertex and hyperedge collapse, and evaluate the effects of topological simplification on the resulting hypergraph visualization across multiple visual encodings.
- Although our simplification method is independent of the method used to visualize the hypergraph, we provide an open-source, interactive tool that implements our simplification method and visualizes how the hypergraph changes as a result of our simplification method using a choice of six visual encodings. The tool is modular and extendable, allowing a user to implement other visual encodings as desired.

Our tool allows users to explore the simplification framework and apply vertex and hyperedge simplifications to gain insights from their own datasets. The tool is available at: `https://github.com/tdavislab/Hypergraph-Vis`.

2 Related Work

We focus on visualization techniques relevant to hypergraphs. For graph visualization, see surveys on graph visualization for information visualization [11], graph representations for scientific visualization [12], visual analysis of large graphs [13], dynamic graphs [14], and graph drawing [15].

Mäkinen [16] introduced two widely used approaches for drawing hypergraphs. In an *edge-based* approach, hyperedges are drawn as smooth curves connecting their vertices. In a *subset-based* approach, they are drawn as closed curves enclosing their vertices. For the edge-based approach, by mapping a hypergraph to a graph, hypergraph visualization could be considered as an extension of graph visualization. Arafat *et al.* [17] proposed four ways to encode a hypergraph as a graph, via complete-, star-, cycle-, and wheel-associated-graphs. Paquette *et al.* [18] considered a hypergraph as a bipartite graph, where hyperedges and vertices form two disjoint and independent sets. For the subset-based approach, hypergraph visualization is closely related to set visualization (see [19] for a survey), which goes back to Euler diagrams [20] and their more restrictive form, the Venn diagrams. Kritz and Perlin [21] proposed the QUAD scheme, which resembled a matrix encoding of set relations: each hyperedge is a column represented by a rectangle and each vertex is a point along a particular row. Riche and Dwyer [22] attempted to improve the readability of the set intersections based on untangling Euler diagrams, by using compact rectangular shapes or duplicating set elements. Simonetto *et al.* [23], [24] introduced an automatic generation of Euler-like diagrams (*EulerView*) for any collection of sets and their intersections. Many recent works focused on representing sets in more efficient ways, including LineSets [25], BubbleSets [26], MapSets [27], UpSet [28], and LinearDiagrams [29]. Jacobsen *et al.* [30] proposed MetroSets, a novel tool to visualize sets automatically in the form of metro maps, which scales well for hundreds of elements and more than a dozen sets. The rainbow box-based visualization implemented in our tool is inspired by the work of Lamy [31], which visualized undirected graphs and symmetric square matrices by transforming them into overlapping sets, and visualized them with rainbow boxes. The HyperNetX Python package [6] includes hypergraph visualization using an Euler diagram approach. It also includes the ability to perform *exact* hyperedge and vertex collapses (as opposed to *approximate* collapses, which this paper explores). Collapsed vertices and hyperedges are visualized in HyperNetX as larger "supervertices" and thicker "superhyperedges".

In terms of layouts, Valdivia *et al.* [32] introduced Parallel Aggregated Ordered Hypergraph (PAOH) as a hybrid of an edge-based and a subset-based (matrix) approach, which represents vertices as parallel horizontal bars and hyperedges as vertical lines, using dots to depict the connections to one or more vertices. Kerren and Jusufi [33] introduced a radial layout, where vertices are evenly distributed on a circle, and the hyperedges are represented as arcs that enclose the circle. Hypergraphs can be represented geometrically [34], [35], starting with Zykov [36]. Gropp [37] positioned the vertices in the plane such that those that form hyperedges are collinear in the plane. Evans *et al.* [38] used polygons to represent hyperedges in 3D to gain additional flexibility. Qu *et al.* [39] encoded hyperedges as polygons and proposed a joint optimization on the layout of the hypergraph and its dual.

Evaluating hypergraph visualizations can be considered from a quantitative and a qualitative perspective. Many evaluation criteria for graph visualization are applicable for hypergraphs (e.g., [40], [41]), including aesthetic criteria such as *readability* [40] and *faithfulness* [41]. Mäkinen [16] gave a set of desirable aesthetic properties for subset-based hypergraph visualization. Arafat *et al.* [17] perfected these properties by introducing quantitative metrics such as *concavity*, *planarity*, and *coverage*. In our work, we evaluate the quality of hypergraph visualizations after simplification using four aesthetic criteria. The first criterion evaluates the Euler diagram-based visualization, which is to minimize the approximate number of contour intersections. The remaining three criteria evaluate the bipartite graph-based visualization, which aims to minimize the number of edge crossings [42], to minimize the normalized edge length variation [43], and to maximize the minimum angle between edges out from a vertex [42], respectively.

For graph simplification, a number of works focused on algorithmic developments [44], [45], [46] and visualization [47], [48], [49], [50], [51]. In particular, Suh *et al.* [52] proposed a topology-based graph simplification tool, which enables contraction of edges with weight below a user-specified threshold. Work on hypergraph simplification is much sparser. Lemonnier *et al.* [53] studied hypergraph simplification theoretically using a graphical language from quantum physics. To the best of our knowledge, we propose the first framework to use topological profiles to guide the hypergraph simplification process for visual exploration.

Finally, research efforts have also focused on visualizing large graphs with advanced hardwares such as GPUs, e.g., Graphistry (`https://github.com/graphistry/`); see [13] for surveys. Few works focus on large hypergraph visualization. We consider these scalable visualization approaches to be tangential to hypergraph simplification.

In this paper, we focus on increasing the readability while preserving as much information faithfulness of hypergraph data as possible via simplification for insight discovery. It is important to emphasize that our simplification applies to *any* hypergraph visualization technique. We primarily use subset-based approaches for hypergraph visualization, including Euler diagram, bipartite graph, bubble sets, and rainbow box-based approaches (Fig. 1).

3 Technical Background

We review two graph representations relevant to hypergraphs, namely, line graphs and clique expansions, as well the notion of a dual hypergraph. To explore graphs, one might employ network science concepts such as walks, distance, and connected components to discover entities of interest. In this paper, we work with analogous *hypernetwork science* [54] concepts for hypergraphs, namely, s-walks, s-distance, and s-connected components.

3.1 Line Graphs, and Clique Expansions

We begin with an example hypergraph H in Fig. 5a, with five black vertices and four hyperedges. It is specified by $V = \{v_1, \cdots, v_5\}$ and $E = \{e_1, e_2, e_3, e_4\} = \{\{v_1, v_2, v_5\}, \{v_2, v_3\}, \{v_3, v_4, v_5\}, \{v_1, v_5\}\}$. In most scenarios of this paper, a hypergraph is shown with an Euler-bipartite visualization.

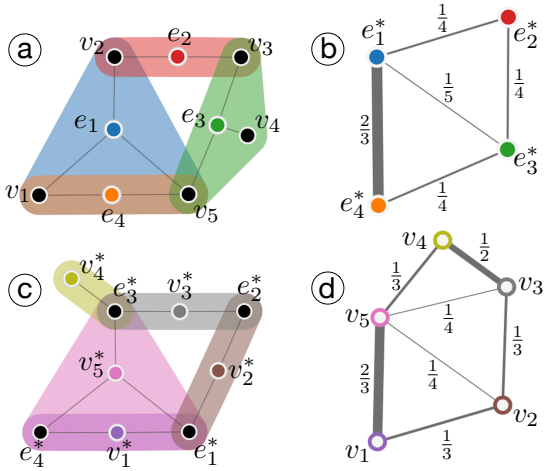


Fig. 5: (a) A hypergraph H; (b) the Jaccard weighted line graph $L_J(H)$; (c) the dual hypergraph H^*; and (d) the Jaccard weighted clique expansion $Q_J(H)$. In (b) the hyperedges e_1 and e_2 in H turn into vertices e_1^* and e_2^* in $L_J(H)$ with weight on (e_1^*, e_2^*) equal to $|e_1 \cap e_2|/|e_1 \cup e_2| = |\{v_2\}|/|\{v_1, v_2, v_3, v_5\}| = 1/4$. In (d) vertices v_1 and v_2 in H belong to sets of hyperedges $\{e_1, e_4\}$ and $\{e_1, e_2\}$, respectively, so the edge (v_1, v_2) in $Q_J(H)$ has a weight equal to $|\{e_1, e_4\} \cap \{e_1, e_2\}|/|\{e_1, e_4\} \cup \{e_1, e_2\}| = 1/3$.

As part of the pipeline for hypergraph simplification, we convert the hypergraph into a graph. There are two candidate graph representations of a hypergraph: the *line graph* [9] and the *clique expansion* [10]. We formalize these concepts below.

Definition 3.1. The *line graph* $L(H)$ of a hypergraph H consists of a vertex set $\{e_1^*, \cdots, e_m^*\}$, and an edge set $\{(e_i^*, e_j^*) \mid e_i \cap e_j \neq \emptyset, i \neq j\}$.

Definition 3.2. The *clique expansion* $Q(H)$ of a hypergraph $H = (V, E)$ consists of vertex set V (the same vertex set as H), and there is an edge (v_i, v_j) in $Q(H)$ if there exists some hyperedge $e \in E$ such that $v_i, v_j \in e$.

The line graph and clique expansion are related through the concept of duality.

Definition 3.3. The *dual hypergraph* $H^* = (E^*, V^*)$ of $H = (V, E)$ has vertex set $E^* = \{e_1^*, \cdots, e_m^*\}$ and hyperedge set $V^* = \{v_1^*, \cdots, v_n^*\}$, where $v_i^* = \{e_j^* \mid v_i \in e_j \text{ in } H\}$.

As shown in Fig. 5c, H^* swaps the roles of vertices and hyperedges. For instance, hyperedge e_1 in H becomes vertex e_1^* in H^*, and vertex v_1 in H becomes hyperedge v_1^* in H^*. It is not difficult to show that the clique expansion is the line graph of the dual, i.e., $Q(H) = L(H^*)$.

For our hypergraph simplification, we work primarily with a weighted line graph or clique complex, using intersection sizes or Jaccard indices as weights. In general the *Jaccard index* (also known as the *Jaccard similarity*) of two sets, X and Y, is a measure of similarity between them. Formally, it is the size of their intersection divided by the size of their union,

$$jaccard(X, Y) = \frac{|X \cap Y|}{|X \cup Y|}.$$

In $L(H)$ the intersection weight of edge (e_i^*, e_j^*) is $|e_i \cap e_j|$ and the Jaccard weight is $|e_i \cap e_j|/|e_i \cup e_j|$. By duality, in $Q(H)$ the intersection weight of edge (v_i, v_j) is $|v_i^* \cap v_j^*|$ and the Jaccard weight is $|v_i^* \cap v_j^*|/|v_i^* \cup v_j^*|$. We denote the Jaccard (resp. intersection) weighted line graph as $L_J(H)$ (resp. $L_I(H)$) and clique expansion as $Q_J(H)$ (resp. $Q_I(H)$). The Jaccard weighted line graph of our example H is shown in Fig. 5b, and the Jaccard weighted clique expansion is shown in Fig. 5d.

Intuitively, the Jaccard and intersection weighted line graphs of a hypergraph capture the similarities between hyperedges; the higher the weights, the more similar they are. On the other hand, the Jaccard and intersection weighted clique expansions capture similarities between vertices.

3.2 s-Walks and s-Connected Components

For a graph $G = (V, E)$, a *walk* of length k is a sequence of vertices connected by edges. It can also be described as a sequence of successively incident edges. We include a similar notion of a walk on a hypergraph, introduced by [54], using a hyperedge perspective.

Definition 3.4. An *s-walk of length k* between hyperedges f and g in a hypergraph H is a sequence of hyperedges, $f = e_{i_0}, e_{i_1}, \cdots, e_{i_k} = g$, where for each $1 \leq j \leq k$, $i_{j-1} \neq i_j$ and $|e_{i_{j-1}} \cap e_{i_j}| \geq s$.

Definition 3.5. For a hypergraph $H = (V, E)$, a subset of hyperedges $C \subseteq E$ is *s-connected* if there exists an s-walk between all pairs of hyperedges in C. C is an *s-connected component* if it is maximal, that is, there is no s-connected set $C' \subseteq E$ such that $C \subsetneq C'$.

Definition 3.6. An *s-line graph* of H (for $s \geq 1$), denoted as $L^s(H)$, is a filtered line graph where edge (e_i^*, e_j^*) is present only if $|e_i \cap e_j| \geq s$. Similarly, an *s-clique expansion* of H, denoted as $Q^s(H)$, is a filtered clique expansion where edge (v_i, v_j) is present only if v_i and v_j share at least s hyperedges in H.

The notion of s-line graphs and s-clique expansions allows us to *filter* a hypergraph by its connectivity, as illustrated in Fig. 6. For example, the orange and the green hyperedges are 3-connected, whereas the orange and the blue hyperedges are 1-connected. There is a 3-walk (dotted black arrow) between the the orange and the purple nodes via the green node in Fig. 6d, so the orange and the purple hyperedges are in the same 3-connected component.

For the remainder of this paper, we work primarily with the Jaccard weighted s-line graph of H, denoted as $L_J^s(H)$, and the Jaccard weighted s-clique expansion of H, denoted as $Q_J^s(H)$. Unless otherwise stated, $s = 1$. In one of our examples, we will

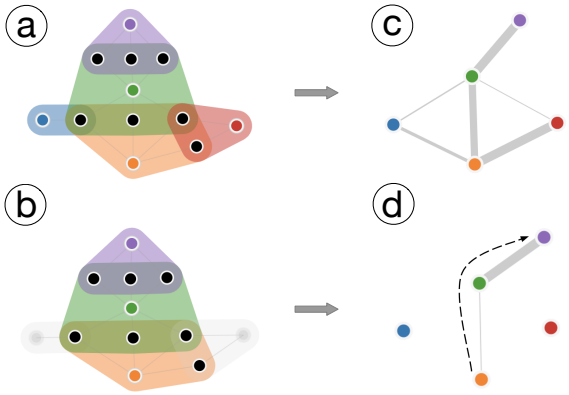


Fig. 6: An example of filtering an s-line graph using the s parameter, for $s = 1$ (c) and $s = 3$ (d). Such a filtering leads to a filtering of the original hypergraph (a), where hyperedges that are in singleton s-components (i.e., hyperedges that are not s-connected to any other hyperedge) are grayed out in (b).

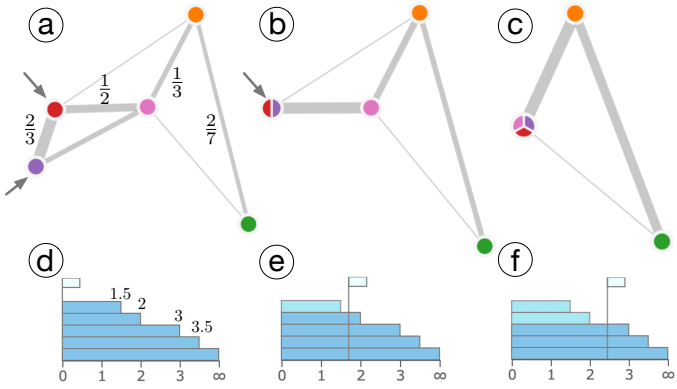


Fig. 7: A barcode-guided topological simplification of a graph. Note that the weights shown on the graph in (a) are inverted to get the bar lengths in (d).

provide a comparison between edge weights based on the Jaccard indices and the intersection size.

3.3 Topological Simplifications of Graphs

In this section, we first introduce the topological profile of a weighted graph, formally known as its *barcode* [7], [55], which is grounded in persistent homology [56]. We then use the barcode to guide the topological simplification of the graph. Later, in Sect. 4, we show how simplification of graphs $L^s(H)$ and $Q^s(H)$ is used to perform our hypergraph simplification.

To obtain the barcode of a weighted graph G, we apply persistent homology to a metric space representation of the graph [57]. See [58] for an introduction and [7] for an algebraic treatment of persistent homology. Persistent homology can be used to capture topological features (e.g., connected components, loops, and higher dimensional voids) in any dimension, d. In this paper we will focus on $d = 0$, allowing us to simplify the definition of a barcode since this restricted version can be computed using the notion of a minimum spanning tree (MST) of a graph. In other words, a merger of two components corresponds to an edge of the MST. Recall an MST is a spanning tree with minimum possible total edge weight. As an MST can also be used to derive the single linkage clustering (SLC) dendrogram [59], the barcode-guided simplification process is also equivalent to applying the SLC with a threshold.

A weighted graph, $G = (V, E, w)$, consists of vertices, V, edges, E, and a weight function $w : E \to \mathbb{R}^+$. Constructing an MST will tend to keep edges with smaller weight and remove those with higher weight. Typically, this is done because the weights represent distances (or similarities), where a small weight means two vertices are close (or similar), and a large weight means two vertices are far (or dissimilar). In the case that weights are similarities, not distances, two vertices with high weight are more similar than two with low weight. To simplify a similarity-weighted graph, we wish to merge vertices in G into supervertices based on a decreasing order of their similarities. Therefore, before computing the barcode of G, we first invert each edge weight $w(e)$ to be $1/w(e)$ and then compute the corresponding MST. Alternatively, one could set the edge weights to similarities and compute a maximum spanning tree to achieve the same goal. However, in keeping with the topological interpretation as 0-dimensional persistent homology, we choose to invert weights and use the MST formulation, which allows for flexibility in the future to employ other distance metrics or d-dimensional persistent homology for $d > 0$; see Sect. 8.

The *barcode* $\mathcal{B}(G)$ is a visual representation of the MST that consists of a collection of sorted horizontal line segments (*bars*) in a plane, where each line segment (excluding the longest one) corresponds to an edge in the MST with length proportional to its weight. As illustrated in Fig. 7a, four of the thickest edges (shown with edge weights) form the MST; each of these edges gives rise to a bar in the barcode in Fig. 7d. For instance, the edge connecting the most similar vertices – the red and the purple vertices (pointed by arrows) – in Fig. 7a with a weight of $2/3$ gives rise to the shortest bar of length 1.5 in Fig. 7d.

Suh *et al.* [52] used the barcode to control the contraction and repulsion of edges in the force-directed layout of a graph. Instead, in this paper, we use the barcode to guide the merging of vertices into supervertices as part of the simplification pipeline. As illustrated in Fig. 7e, if we choose a threshold that passes the first bar, we combine the purple and red vertices together into a supervertex in (b). Similarly, choosing a threshold that passes the second bar in Fig. 7f results in the combination of purple, red, and pink vertices into a supervertex in (c). The number of bars with length larger than a chosen threshold indicates the number of vertices remaining after simplification. The last bar corresponds to a single supervertex after all vertices are combined together. Since this supervertex cannot be simplified further, the last bar has a length of infinity.

Stability. We consider a barcode-guided simplification of a graph to be mathematically justified in the following sense. We call a graph G' an *ε-simplification* of another graph G, if G' is obtained from G via *vertex contractions* (that is, merging subsets of vertices in G, thus contracting the induced edges), and the barcode $\mathcal{B}(G')$ is the same as the barcode $\mathcal{B}(G)$ except all bars with lengths at most ε have been removed. Based on the stability of barcodes [8], by performing a simplification up to a threshold of ε, the distance between the barcodes of G' and G is upper bounded by ε. The distance we use to compare barcodes is called the bottleneck distance.

Definition 3.7. Let γ be a matching between the intervals (bars) I and I' of $\mathcal{B}(G)$ and $\mathcal{B}(G')$ respectively. The *bottleneck distance*

between $\mathcal{B}(G)$ and $\mathcal{B}(G')$ is defined as

$$d_\infty(\mathcal{B}(G),\mathcal{B}(G')) = \inf_{\gamma} \sup_{I\in\mathcal{B}(G)} ||I-\gamma(I)||_\infty,$$

where L_∞ distance between two bars $I=(b,d)$ and $I'=(b,d')$ is defined as $||I-I'||_\infty = \max(|b-b'|,|d-d'|)$. In our setting, the L_∞ distance between two bars that start at zero, $I=(0,d)$ and $I'=(0,d')$, is the absolute difference of their end points, $||I-I'||_\infty = |d-d'|$.

Intuitively, the bottleneck distance measures the smallest value x such that there is a matching γ between the bars of one barcode, and the bars of the other barcode where all pairs of γ-matched bars have L_∞ distance at most x.

Given this formulation of bottleneck distance, we can now formally state the stability of barcodes result as

$$d_\infty(\mathcal{B}(G),\mathcal{B}(G')) \leq \varepsilon.$$

By construction, the MST of G' is generated from the MST of G by contracting edges with lengths at most ε, therefore merging vertices connected by these edges that are at most ε apart. Therefore, $\mathcal{B}(G)$ and $\mathcal{B}(G')$ differ only by the bars that are removed via the simplification, which have lengths at most ε.

4 Methods

We now describe multiscale topological simplifications of hypergraphs. We use the term vertex (resp. hyperedge) simplification to mean a sequence of operations that reduce the size of a hypergraph by merging vertices (resp. hyperedges) in decreasing levels of similarity. Our framework is as follows:

1. Map a hypergraph H to a graph representation G;
2. Generate the barcode $\mathcal{B}(G)$ of G and use $\mathcal{B}(G)$ to guide its simplification;
3. Use a simplified G to induce a simplification of H.

At the core of our approach is the idea that a simplified clique expansion induces a vertex simplification of the hypergraph, whereas a simplified line graph induces a hyperedge simplification. Using barcodes of these weighted graph representations, we allow vertices to be combined if they belong to almost the same set of hyperedges, and hyperedges to be merged if they share almost the same set of vertices, both in a mathematically justifiable way.

Two parameters guide the simplification process. First, the parameter s gives rise to a filtered version of the line graph or the clique expansion. Fig. 6 illustrates a multiscale filtering of hyperedges using the s parameter, for $s=1$ and 3, respectively.

Second, the parameter ε used in the ε-simplification of a hypergraph H by merging vertices (or hyperedges) whose distances are upper bounded by ε (correspondingly, whose similarities are lower bounded by $1/\varepsilon$). Fig. 8 illustrates multiscale hyperedge simplifications. At ε_1, the brown and purple hyperedges merge into one (pointed by a black arrow); and at ε_2, three hyperedges in red, green, and blue merge into one at the same time (pointed by the hollow arrow). Fig. 9 illustrates multiscale vertex simplifications. At ε'_1, the gray and pink vertices in the clique expansion (that correspond to the two vertices in the orange hyperedge) merge into a supervertex (pointed by a black arrow); and at ε'_2, two vertices in purple and teal in the clique expansion merge into a supervertex (pointed by a hollow arrow).

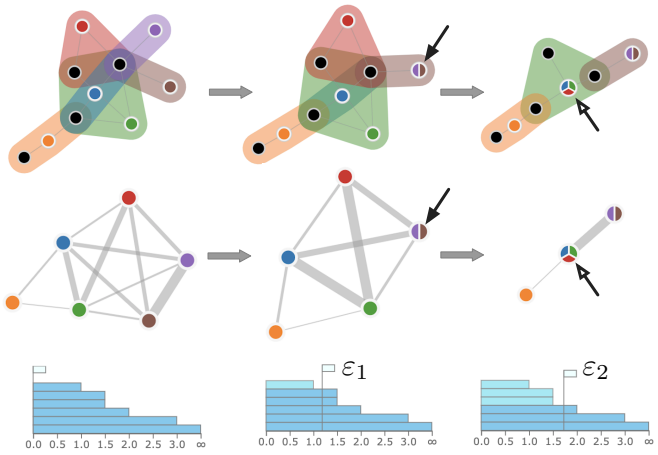


Fig. 8: Multiscale hyperedge simplifications of a hypergraph. Top row includes from left to right: the original hypergraph and its hyperedge simplifications across two scales. Middle row shows its corresponding Jaccard weighted line graphs. Botton row shows the simplification thresholds *w.r.t.* the barcodes.

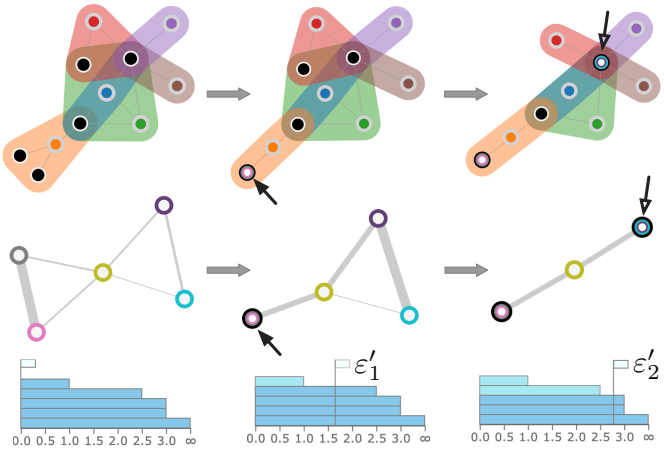


Fig. 9: Multiscale vertex simplifications of a hypergraph. Top row includes from left to right: the original hypergraph and its vertex simplifications across two scales. Middle row shows its corresponding Jaccard weighted clique expansions. Botton row shows the simplification thresholds *w.r.t.* the barcodes.

5 Interactive Visualization System

We provide an open-source, interactive visualization system that supports both vertex and hyperedge simplification of an input hypergraph. We describe its visual interface in Sect. 5.1 followed by a discussion of the design decisions in Sect. 5.2.

5.1 Visual Interface

The user interface is shown in Fig. 10; see the supplementary video for a demo. We describe the interface based on hyperedge simplification; the interface for vertex simplification is similar with minor modifications.

In Fig. 10, the **graph visualization panel** in the middle visualizes the original hypergraph (a), the weighted line graph representation (b), the simplified line graph (d), and the induced simplified hypergraph (c). The vertices and hyperedges across (a-d) are connected via linked views based upon their correspondences.

When we switch from hyperedge simplification to vertex simplification, weighted line graph representations (b & d) become weighted clique expansions accordingly.

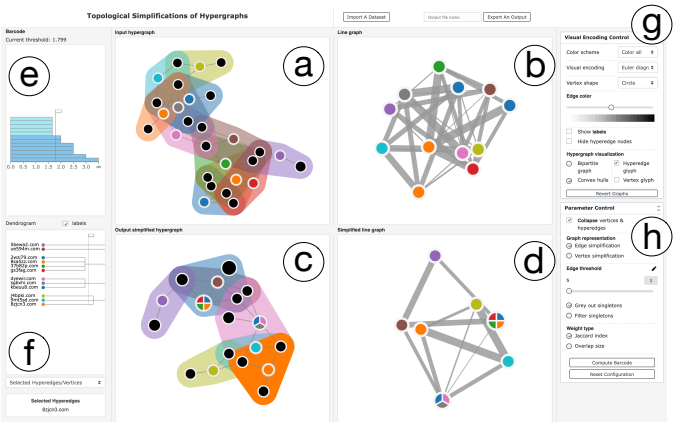


Fig. 10: Interactive user interface.

On the left, the **barcode panel** (e) controls the level of simplification (parameter ε) for the line graph where vertices connected by edges $\{e_i^*, e_j^*\}$ with weight below the selected threshold in (b) are combined into supervertices in (d). Correspondingly, hyperedges e_i, e_j in (c) are merged into one hyperedge. Panel (e) also shows the hierarchical merging of hyperedges (or vertices) via a dendrogram. Further refinement of a simplified hypergraph can be performed via the *bar expansion*: clicking on a bar that has already been simplified will undo the simplification step, which means, if hyperedges e_1 and e_2 have been merged into a single hyperedge e under the current simplification level, clicking the bar that corresponds to this operation will separate e back into e_1 and e_2. To choose the right simplification level, the lower part (f) of the panel displays the labels of hyperedges (or vertices) when hovering on a simplified hyperedge (or vertex). Alternatively, it can also display the persistence graph (not shown here) that shows the number of features (connected components) as a function of the persistence level ε. An appropriate value of ε is typically obtained at the plateau of the persistence graph (see [60] for details).

On the right, the **visual encoding control panel** (g) provides various visual encodings for the hypergraph; see Fig. 1 for an example. The panel also provides options to display vertex and hyperedge labels. The **parameter control panel** (h) deals with parameter configuration. A large input hypergraph can be pre-processed to allow vertex collapse and hyperedge collapse. Collapsing the hyperedges here affects the weights in the line graph or clique expansion, which in turn affects the barcode. If two hyperedges intersect in k vertices, and those k vertices are collapsed into $\ell < k$ supervertices, then the weight between those hyperedges will be ℓ, not k. Choosing hyperedge simplification employs the line graph representation, whereas vertex simplification uses the clique expansion. The s parameter controls the edges present in the s-line graph or s-clique expansion. Singletons in the s-line graph or s-clique expansion are either grayed out (used in barcode computation but visualized as light gray) or filtered (removed from the hypergraph visualization and not used in barcode computation). In computing the barcodes, either the Jaccard index or overlap size can be used. All the parameters set in panel (h) contribute to barcode computation. Any change made to the parameters requires a re-computation of the barcode by clicking *"compute barcode"*.

5.2 Design Decisions

Feedback from a set visualization expert was used to inform a number of design decisions that we justify below.

1. Support user-defined color assignment. Since a categorical color palette provides a limited number (10) of colors, hyperedges lose their distinctiveness within a hypergraph with more than 10 hyperedges. In order to accommodate user preference in hyperedge color management, our tool supports user-driven manual color assignment to hyperedges. During the initialization, users may choose to automatically color all hyperedges (color repeats may occur), or they may choose to color only the top five largest hyperedges (the remaining hyperedges are colored gray); this preserves the distinctiveness among a small number of hyperedges (Fig. 11a, Fig. 14a). Our system also allows a user to manually assign colors to hyperedges of interest for subsequent exploration (Fig. 11b). In addition, a user can adjust the darkness value of edges in the bipartite graph (for the Euler-bipartite and bubble-bipartite hybrid visualizations) to emphasize or de-emphasize the relations between vertices and hyperedges; see Fig. 11b.

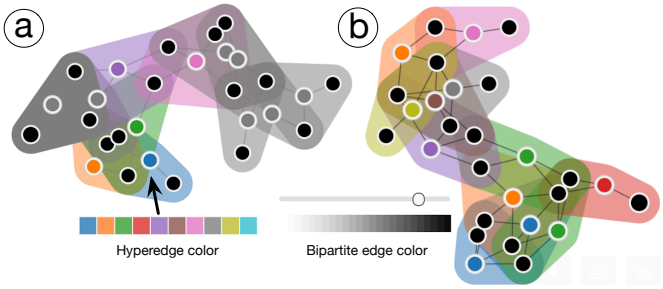


Fig. 11: User-defined color assignment.

2. Explicitly visualize vertices and hyperedges. While other visual encodings are possible for hypergraphs, we choose to support visualizations based on Euler diagram, bipartite, bubble sets, and rainbow box; the first three approaches capture spatial relations among the hyperedges, whereas the fourth approach highlights overlaps among the hyperedges. Since our framework applies topological simplifications to both vertices and hyperedges, our hypergraph visualization aims to explicitly visualize both vertices and hyperedges during the simplification process. To that end, we choose to support Euler-bipartite and bubble-bipartite hypergraph visualizations, where each hyperedge is explicitly visualized as a node that connects to all vertices it contains. Such a visual encoding supports linked views between the original hypergraph and its line graph and clique expansion.

3. Diversify glyph representations. For the Euler diagram (resp. bubble sets) hypergraph visualization, vertices are visualized as black filled circles, whereas hyperedges are double-encoded as colored convex hulls (resp. colored bubble sets) and as colored filled circles. Supervertices and superhyperedges after simplification are represented by concentric circle glyphs and pie charts, respectively. Based on the expert's feedback, our system allows a user to further diversify glyph representations using black filled squares for vertices, and colored concentric square glyphs for supervertices; see Fig. 12 (*cf.* Fig. 1).

4. Highlight relations during hypergraph simplification. Our simplification framework relies on exploiting the relations between a hypergraph, its dual hypergraph, line graph, and clique expansion.

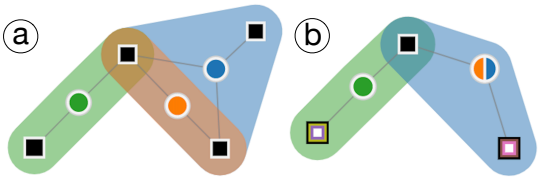


Fig. 12: Vertices as black filled squares and supervertices as colored concentric square glyphs.

Our system highlights their relations via linked views between hyperedges and vertices during the simplification process to increase interpretability.

5. Explicit labels for the dendrogram. The dendrogram panel shows a labeled tree depicting the hierarchical merging of hyperedges (or vertices), as shown in Fig. 13 right and Fig. 21. The dendrogram is also connected with the graph visualization panel via linked views for increased interpretability.

Implementation. We implemented our framework as an interactive web application with a *Python* back-end using a *Flask*-based server. We used the standard *HTML/CSS/Javascript* stack in tandem with *D3.js* and *JQuery JavaScript* libraries for designing the user interface and visualization panels. The front-end handles data upload and graph and hypergraph visualization. The *Python* back-end handles data parsing, creates the hypergraph data structures using the *HyperNetX* [6] library (including its hyperedge and vertex collapse functions), constructs bipartite graphs, computes the persistence barcodes, and performs parameter updates.

6 Results

We provide five example use cases to show how our barcode-guided hypergraph simplification provides an interpretation of the underlying data and helps with insight discovery.

6.1 Southern Women

Our first example considers a small social network. In the 1930s, a group of ethnographers collected data on a group of 18 women in Natchez, Mississippi [61]. They recorded attendance at 14 informal social events over the course of a nine-month period; see Fig. 13a. This dataset has been studied by many other researchers in sociology, information theory, and mathematics; see [62] for a meta-analysis of previous studies.

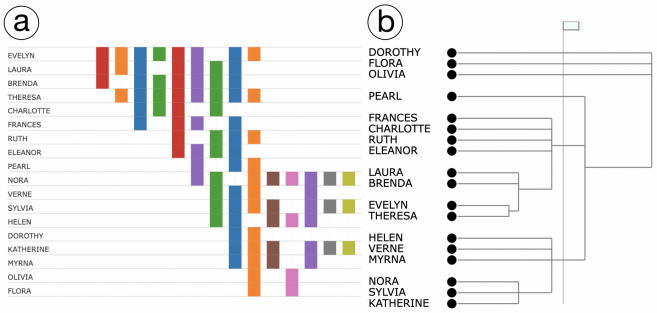


Fig. 13: (a): A table reproduced from [61] using rainbow boxes that records which social events (columns) each woman (rows) attended. (b): A hierarchical representation of our simplification result (at $s = 1$, $\varepsilon = 0.28$) that highlights intrinsic group structure.

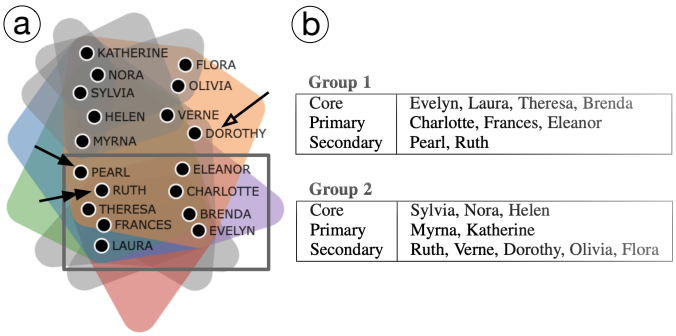


Fig. 14: (a) A hypergraph showing women as vertices (black nodes) grouped by events they attended as hyperedges (top five largest hyperedges are visualized as colored convex hulls). (b) Groups identified from interviews in the original study. In (a), vertices in the rectangular box belong to Group 1, others below to Group 2.

The hypergraph is shown with an Euler-bipartite visualization in Fig. 14a (hyperedge glyphs are hidden). Through interviews with these women, Davis *et al.* [61] identified two largely distinct groups and determined core, primary, and secondary members of each group based on how they were involved with the events, as summarized in Fig. 14b. Such groupings are considered as the *ground truth* in our exploration. The original layout of the hypergraph in Fig. 14a gives the illusion of a left-right split of the groups, whereas the ground truth indicates a top-bottom split, where **Pearl** (filled arrow) belongs to Group 1, **Dorothy** (hollow arrow) belongs to Group 2, and **Ruth** (double filled arrow) is identified as secondary in both groups.

Using this *Southern Women* dataset, we will demonstrate how varying parameter choices using our simplification framework highlights the group relations within the ground truth. We will observe which parameter choices agree with the reported ground truth and which do not. In all cases, we will perform vertex simplification as the goal is to see how the women are grouped based on the events they attend.

Simplification using Jaccard weights. We first consider vertex simplification of the Jaccard weighted clique expansion $Q_J^1(H)$ derived from the original hypergraph H with $s = 1$. As we increase our simplification parameter ε, guided by a priori knowledge of the group structure, we observe that at $\varepsilon = 1.6$ (Fig. 15a & d), our simplification recovers *almost perfectly* the core members.

As shown in Fig. 15a, supervertices are formed mostly according to the core groups from the ground truth. All the core members of Group 1 (**Evelyn**, **Theresa**, **Laura**, **Brenda**) are merged together into a supervertex (filled arrow on the bottom). Almost all core and primary members of Group 2 (**Nora, Sylvia, Katherine, Myrna**, minus **Helen**) form a second supervertex (filled double arrow on the top). Here, all supervertices consist of members of the same group, with the exception of **Pearl** and **Dorothy** (Group 1 and Group 2 secondary, respectively, hollow arrow). **Flora** and **Olivia** form a supervertex, since they attended the same set of two events (*cf.* Fig. 13, double hollow arrow).

We further hypothesize that after simplifying $Q_J^1(H)$ down to two supervertices, each supervertex would correspond to a group in the ground truth. However, this is not the case as shown in Fig. 15b. In the simplified hypergraph, one supervertex contains only **Flora** and **Olivia**, whereas the other supervertex contains everyone else.

Finally, we increase the s value to $s = 4$, filter out singletons,

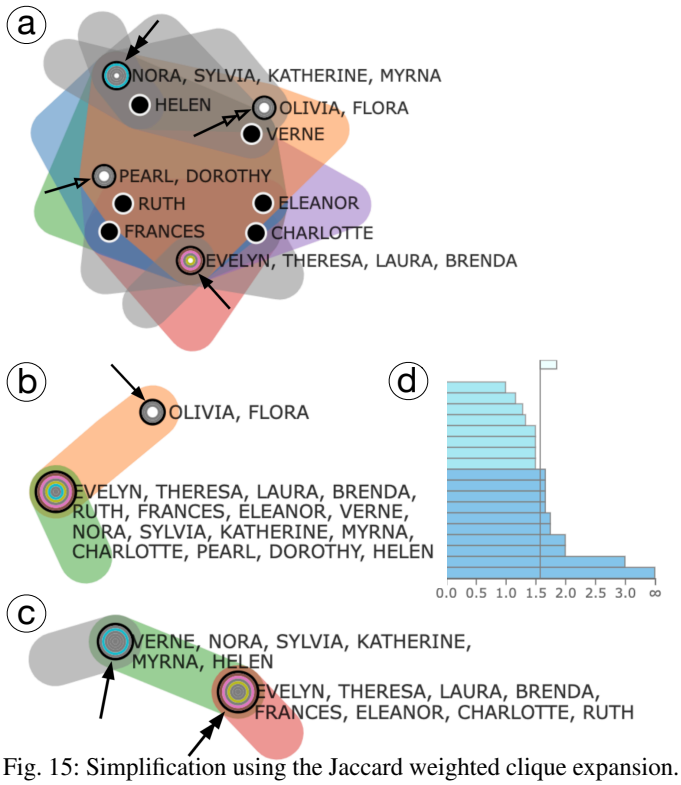


Fig. 15: Simplification using the Jaccard weighted clique expansion. (a) $s = 1$, $\varepsilon = 1.6$. (b) $s = 1$, $\varepsilon = 3.2$, down to two supervertices. (c) $s = 4$, filtering out singletons, down to two supervertices. (d) Corresponding barcode for (a).

and simplify $Q_J^4(H)$ to two remaining supervertices. The result is shown in Fig. 15c. This filtering process leaves out **Dorothy**, **Olivia**, **Flora**, and **Pearl** (all of whom are secondary members), but otherwise splits the women into the correct two groups. The left supervertex (filled arrow) consists of Group 2 core and primary members and **Verne** (Group 2 secondary). The right supervertex (filled double arrow) similarly consists of Group 1 core and primary members and **Ruth** (Group 1 & 2 secondary).

In summary, simplification using the Jaccard weights and $s = 1$ is unable to identify the two groups in the ground truth without using the a priori knowledge. Using $s = 4$, we could identify two subgroups appropriately; however, certain secondary members are filtered out unintentionally. Next we compare with simplification results using overlap weights.

Simplification using overlap weights. We first simplify the overlap weighted clique expansion $Q_w^1(H)$ by setting $s = 1$. As we increase ε, the simplification process clearly identifies the two subgroups in the ground truth; see Fig. 16a with the corresponding merging hierarchy in Fig. 13 (right). It identifies the same split as in Fig. 15c without filtering **Pearl, Dorothy, Flora**, and **Olivia**. In particular, the top supervertex (filled arrow) contains all core and primary members of Group 2 plus its secondary member **Verne**; and the bottom supervertex (filled double arrows) contains all core and primary members of Group 1 plus **Ruth**. **Flora** and **Olivia** are combined as usual.

As we increase ε further, we simplify $Q_w^1(H)$ down to three groups; see Fig. 16b. There is no threshold with two supervertices as the final simplification merges all three into one supervertex. As in the case of Jaccard weights, this naive simplification does not achieve the desired split into the correct groups. **Flora** and **Olivia**

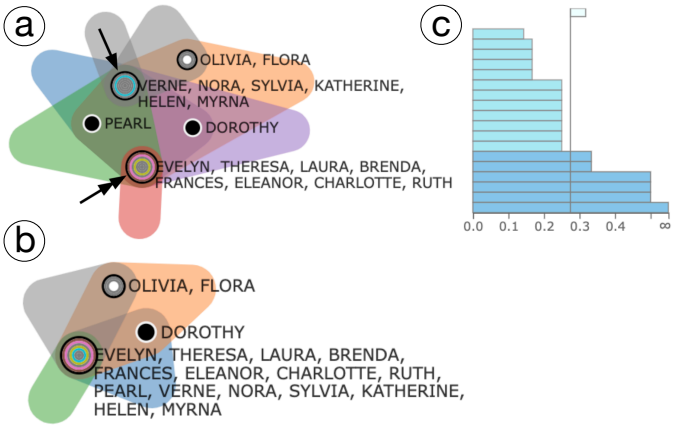


Fig. 16: Simplification using overlap weighted clique expansion $Q_w^1(H)$ with $s = 1$: (a) $\varepsilon = 0.28$, with five groups; (b) filtering out singletons, down to three groups.

are again grouped together, **Dorothy** is on her own, and everyone else is grouped together.

Finally, we increase the s value again to $s = 4$. It shows the same split as Fig. 16a, while again filtering **Pearl, Dorothy, Flora**, and **Olivia**.

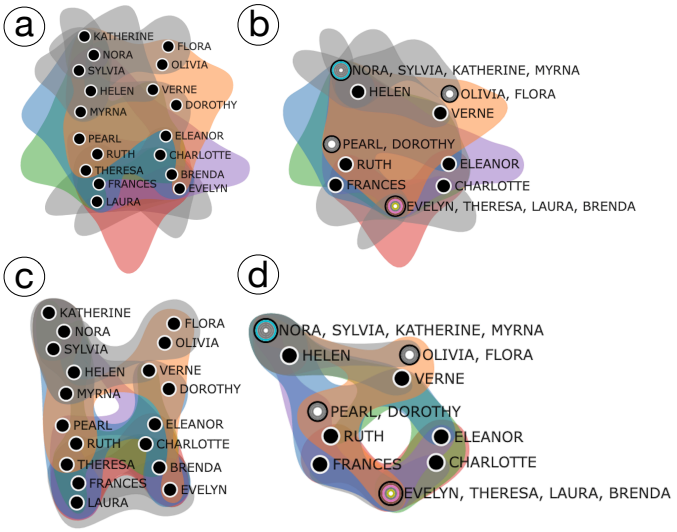


Fig. 17: *Southern Women* hypergraph visualized with bubble-bipartite visualization before (a) and after (b) simplification (hyperedge glyphs not showing), and with bubble sets before (c) and after (d) simplification.

Final remarks. In this small example with ground truth, we are able to see how Jaccard and overlap weights perform slightly differently. The fact that it is easier to identify the two groups using overlap weights may be an artifact of how the two groups were identified in the first place by Davis *et al.* through interviews and sociological observations. Hence, we cannot expect such an observation to be generalized to other datasets. It is clear from this example that Jaccard and overlap weights may help provide different insights into the same dataset. In crafting the examples in the following subsections, we explored each dataset using a variety of parameter and weight choices and will show the choice that provided the most insight. It happens that Jaccard weights are used in the remaining examples.

For a qualitative comparison *w.r.t.* the effects of topological simplification on visualization, we show in Fig. 17 the hypergraph before and after simplification using the bubble-hybrid (a-b) and bubble sets (c-d) visualization (*cf.* the Euler-bipartite visualization in Fig. 14a and Fig. 15a). See Sect. 7 for a quantitive evaluation.

6.2 *Les Misérables*

Our second example considers Victor Hugo's novel Les Misérables, as broken down in the file `jean.dat` from the Stanford Graph Base [63]. This dataset lists the set of characters found within each volume, book, chapter, and scene of the story. To form our hypergraph, we consider each character to be a vertex and each (volume, book)-pair to be a hyperedge containing those characters that appear within. We use the following parameter settings for the simplification: collapse vertices and hyperedges; vertex simplification; $s = 1$; filter singletons; and Jaccard weighted edges. Fig. 18 shows the simplification results.

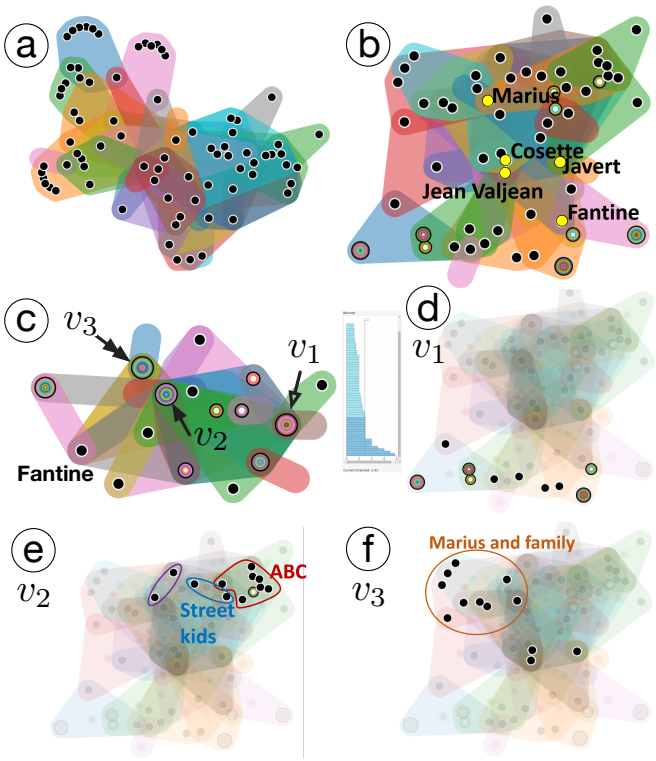


Fig. 18: *Les Misérables*: hypergraph simplification via vertex simplification, (a) original, (b) collapsed, and (c) simplified hypergraphs with the barcode thresholded at $\varepsilon = 2.93$. (d-f): Correspondences between three main supervertices of the simplified hypergraph (c) with their vertices in the original hypergraph (a).

It is not possible to go through the entire plot of this very long novel here, but the story of *Les Misérables* revolves around the characters of **Jean Valjean, Javert, Cosette**, and **Fantine** (labeled yellow nodes in Fig. 18b). Other characters of interest include Cosette's love interest **Marius** (also labeled in Fig. 18b), a revolutionary student club, and some others who get mixed up in an uprising. Our vertex simplification groups many characters together in interesting ways that reflect the narrative of this story.

In the simplified hypergraph Fig. 18c, a vertex (character) of interest is the fact that **Fantine**, one of the characters many consider a main character in the novel, does not group with her daughter **Cosette** or any other main character. In hindsight, this makes sense, since **Fantine** appears only in the first volume (of five) and acts as a bridge from the first volume to the rest of the story, losing her daughter **Cosette** early on. **Fantine** remains an unsimplified vertex (not grouped with any other characters) since she interacts with many groups that do not interact with each other. This makes her Jaccard similarity to all of these groups low.

Fig. 18(d-f) show and describe correspondences between three of the supervertices in the simplification (v_1, v_2, and v_3, pointed by arrows in Fig. 18c) with the vertices of the original hypergraph (Fig. 18a). The supervertex v_1 from Fig. 18c contains many peripheral characters in the first volume in Fig. 18d, those that interact with **Jean Valjean** and **Fantine**. The supervertex v_2 contains all of the "Friends of the ABC" revolutionary student group (circled in red) plus two street kids (circled in blue) who get mixed up in the uprising and two additional prominent uprising characters (circled in purple); see Fig. 18e. The supervertex v_3 from Fig. 18c contains **Jean Valjean**, **Cosette, Javert, Marius**, and all of Marius' family (circled in orange); see Fig. 18f.

Finally, we can use the *bar expansion* capability to explore the two bars that are merged just before our chosen threshold ($\varepsilon = 2.93$). Fig. 19a shows that expanding one bar splits the supervertices containing **Marius** and his family from the one containing **Jean Valjean, Cosette**, and **Javert**. In Fig. 19b, we see that expanding the second bar further splits **Valjean** and **Cosette** from **Javert**. Both operations make sense in the context of the story. All of these characters interact frequently, but **Valjean** and **Cosette** are certainly the closest and most central pair in the story. Moreover, **Javert** is always chasing **Valjean** so they do not interact as much, instead they each interact with the same intermediate characters.

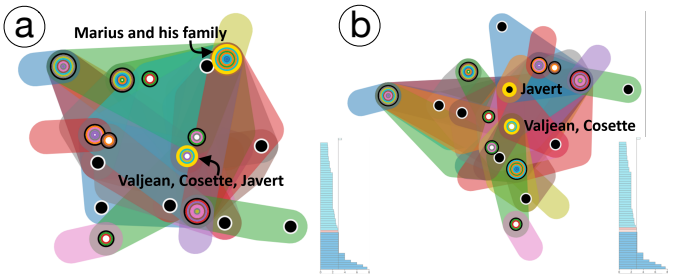


Fig. 19: *Les Misérables*: expanding the last two bars before the simplification threshold.

This barcode-guided simplification does not capture the complete narrative flow of the story from beginning to end, but it does group characters in ways that align with the story. It also shows how various smaller groups interact with each other as smaller groups merge to form larger groups, and how some characters act as bridges between others.

6.3 Hallmark Biological Pathways

Our third example explores the *Hallmark Biological Pathways* hypergraph [4] as shown in Fig. 4. The full Hallmark dataset contains 50 pathways (hyperedges) and 4,386 genes (vertices). We first use HyperNetX to break this large hypergraph into 2-connected components in order to focus on a smaller subset for visual exploration. We chose a component that contains 10 hyperedges. Throughout this example, biological information about pathways is obtained online from MSigDB [4], [5].

For this example, we use the following parameter settings: do not collapse vertices or hyperedges; hyperedge simplification; $s = 1$;

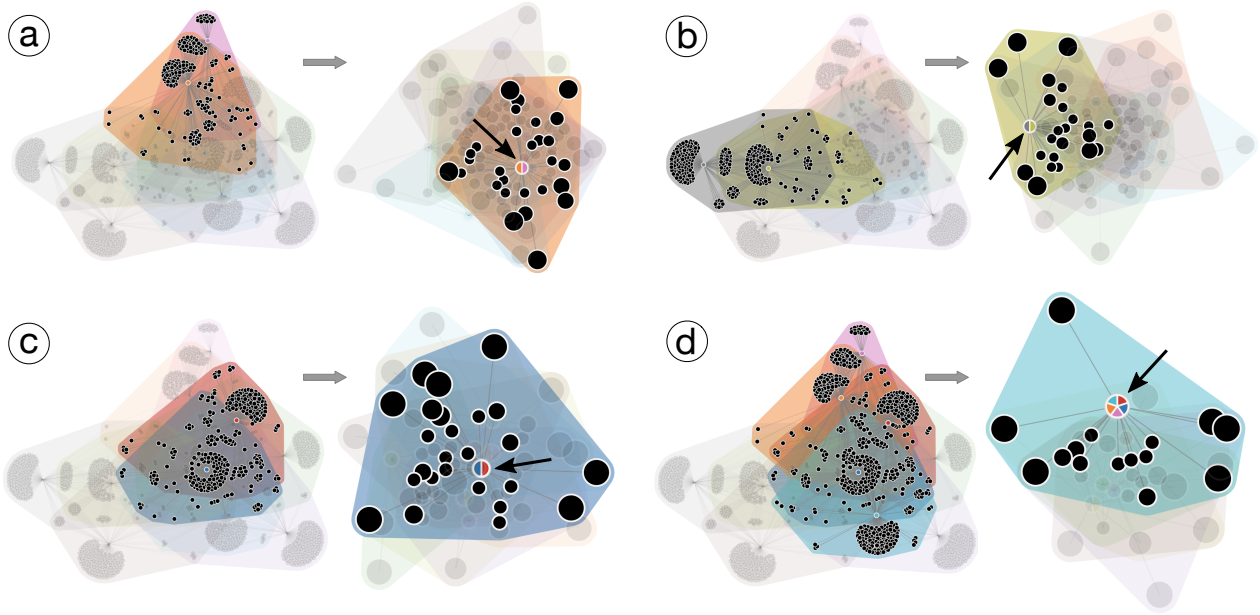


Fig. 20: (a-d): Steps 1 to 4 of *Hallmark Biological Pathways* hypergraph simplification. For each subfigure, the right image shows the new merged hyperedge in the simplified hypergraph, and the left image highlights the corresponding hyperedges in the original hypergraph. (a) Step 1 merges hyperedges (pathways) INTERFERON_ALPHA_RESPONSE and INTERFERON_GAMMA_RESPONSE. (b) Step 2 merges pathways HYPOXIA and GLYCOLYSIS. (c) Step 3 merges pathways INFLAMMATORY_RESPONSE and TNFA_SIGNALING_VIA_NKFB. (d) Step 4 merges hyperedges formed in steps 1 and 3 with ALLOGRAFT_REJECTION.

gray out singletons; and Jaccard weights. Rather than choosing one threshold for simplification as in the previous example, we will show insights gathered by walking through the various ϵ thresholds to see the order in which the hyperedges (pathways) merge. The original Hallmark hypergraph and the vertex collapsed hypergraph is shown in Fig. 4.

Fig. 20(a-d) shows the first four steps of the simplification process from left to right. The first two hyperedges (pathways) to merge are **INTERFERON_ALPHA_RESPONSE** and **INTERFERON_GAMMA_RESPONSE** (Fig. 20a). Both are pathways that contain genes up-regulated in response to interferon (resp. alpha or gamma) proteins. The second pair (Fig. 20b) to merge are **HYPOXIA** (genes up-regulated in response to low oxygen) and **GLYCOLYSIS** (genes involved in breaking down glucose). Unlike the first pair of hyperedges, the processes of hypoxia and glycolysis are not very related, but they are merged early in the process so they must have many genes in common. An observation that may be useful for biology researchers. The third bar guides the merging of **INFLAMMATORY_RESPONSE** with **TNFA_SIGNALING_VIA_NFKB** (genes regulated by NF-kB in response to TNF); see Fig. 20c. The NF-kB protein complex and TNF protein are both known to play a role in regulation of immune response so a merge with an inflammatory response is reasonable. The fourth step merges the **INTERFERON** pair in (a) with the **INFLAMMATORY** and **TNFA** pair in (c) and the **ALLOGRAFT_REJECTION** pathway, which is involved with transplant rejection. This new superhyperedgen (containing inflammation/immune pathways) seems to encompass the inflammatory response hyperedges present within the original set of 10.

At this point, we have five superhyperedges in our simplified hypergraph: inflammation/immune pathways; **HYPOXIA** and **GLYCOLYSIS**; **APOPTOSIS** (programmed cell death); **MTORC1_SIGNALING** (related to mTORC1 complex involved in protein synthesis); and **P53_PATHWAY** (related to cancer suppression). In the next four steps of the simplification (1) **APOPTOSIS** merges with the inflammation/immune hyperedge, (2) **MTORC1** merges with **HYPOXIA** and **GLYCOLYSIS**, (3) those two just created superhyperedges merge together, and (4), **P53_PATHWAY** merges in last.

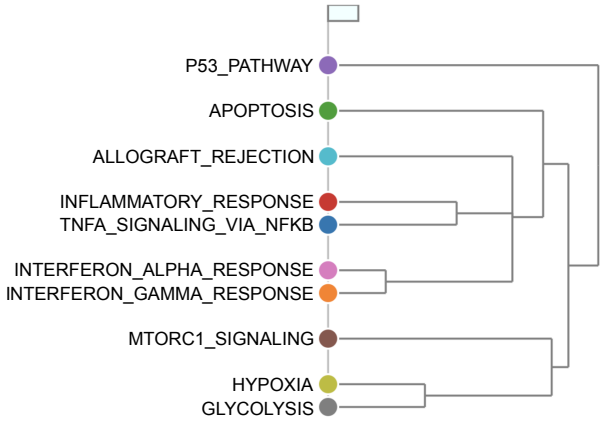


Fig. 21: Similarity hierarchy for 10 Hallmark pathways.

This simplification process provides a merging hierarchy among these 10 pathways; see Fig. 21. Whereas MSigDB provides pairwise overlap sizes for all of these pathways, the most important insight using our framework is that our simplification process goes beyond pairwise associations and hierarchically merges pathways to discover groups of related pathways.

6.4 ActiveDNS

Our fourth example is from computer networking, specifically the Domain Name System (DNS). DNS focuses on

how computers translate from the human interpretable domains (e.g., `www.google.com`) to computer-readable IP addresses (e.g., 192.168.1.1). It might seem like these should be one to one, that is, each domain has an IP address, and each IP maps to a domain, but this is not always the case. Domain aliasing means that sometimes multiple domains map to the same IP (misspellings like `www.gogle.com` still get people to the right place) and website hosting services mean that multiple domains can be served up from the same IP address. The way that IPs are allocated to domains can show interesting patterns. Using a hypergraph representation of the data, where vertices are IP addresses and hyperedges are domains, we ask the question of whether or not domains that share many common IP addresses have any common properties.

The dataset for this exploration comes from ActiveDNS [64]. This project out of Georgia Tech does daily active DNS lookups for millions of IP addresses and records the query results in a database. We work with one day of DNS records, from April 26, 2018, as our test case. These data were also explored as a hypergraph in [1]. The entire hypergraph from this day has millions of vertices and hyperedges so we first used the Chapel Hypergraph Library (CHGL) [65], [66] to break the hypergraph into 2-connected components and then chose one of these components that has 30 hyperedges (domain names) to explore visually using our framework. The original and the simplified hypergraphs are shown in Fig. 22a and b, respectively.

For the simplification, we use the following parameters: collapse vertices and hyperedges; hyperedge simplification; $s = 3$; gray out singletons; and Jaccard weight. We chose $s = 3$ to get more to the core of the interactions within this component.

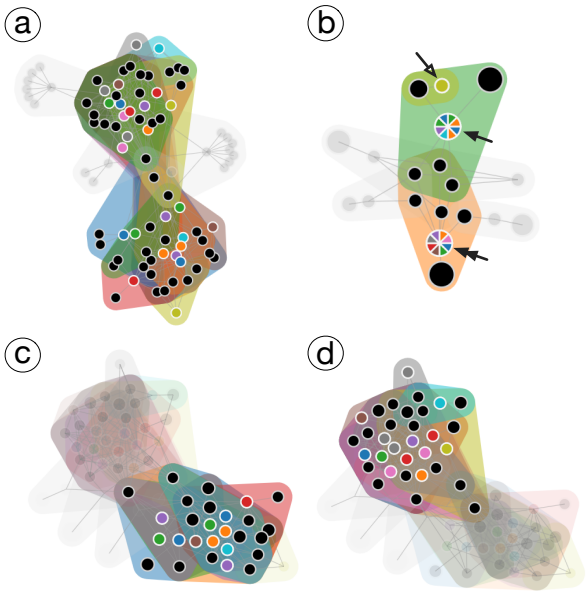


Fig. 22: *ActiveDNS* hypergraph before (a) and after (b) simplification. (c-d) A detailed analysis of simplified hyperedges: (c) elements from the green hyperedge in (b); (d) elements from the orange hyperedge in (b).

As shown in Fig. 22b, the simplified hypergraph shows a clear separation into two main superhyperedges, green (filled arrow) and orange (filled double arrow), with one additional yellow hyperedge (hollow arrow) that is fully contained within the green superhyperedge. There are also four greyed out hyperedges that are not 3-connected to the rest of the component. The separation is also evident in the original hypergraph but the groupings become more obvious in the simplified version. After making this observation, we use the WHOIS lookup from Hurricane Electric BGP Toolkit [67] to answer the question of what the domains that are grouped have in common. WHOIS gives publicly available information about the organization that registered the domain, their contact information, and a variety of other metadata.

Fig. 22d highlights the 14 hyperedges in the original hypergraph that are collapsed to form the orange hyperedge in the simplified version. The domain names (hyperedges) that are collapsed include `worldsleadingcruiselines.com`, `worldsleadingcruiselines.net`, `wlcl.com`, and `worldleadingcruiseline.com`. All 14 are some play on "World's Leading Cruise Lines" and all domains are registered to Carnival Corporation that is one of the world's largest cruise lines. Interestingly, one of the grayed out hyperedges has the domain `comebacktothesea.com`. "Come Back to the Sea" is an older advertising campaign for Carnival and perhaps that is why it does not have as much overlap with the other domains with the more current slogan.

The 11 hyperedges highlighted in Fig. 22c, which are collapsed to form the green superhyperedge in the simplified version, have less in common on the surface. These domains include `bbgdirect.com`, `azattykplus.kg`, and `globalnewsdashboard.com`. Within these 11 domains, 7 of them have "Radio Free Europe" listed as the registered organization in the publicly available WHOIS information, two have no registered organization, and two have completely different organizations. Finally, the yellow hyperedge (hollow arrow) that is contained within the green in Fig. 22b has domain `radiosvobodakrim.mobi` and is also registered to "Radio Free Europe." We make no claims of associations between these domains, especially those registered to other organizations. We only observe that our hypergraph simplification method grouped domains together that, for the most part, were registered by the same organization.

For a qualitative comparison *w.r.t.* the effects of topological simplification on hypergraph visualization, we show in Fig. 23 the hypergraph before and after simplification using the bubble-bipartite visualization (*cf.* the Euler-bipartite visualization in Fig. 22). See Sect. 7 for a quantitive evaluation.

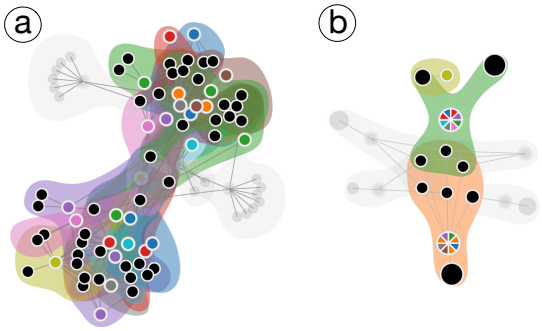


Fig. 23: *ActiveDNS* hypergraph visualized with bubble-bipartite before (a) and after (b) simplification.

6.5 Coauthor Network

Our final example explores a coauthorship network in academic publications. We use the topic-coauthor dataset from ArnetMiner [68] and focus specifically on the information-retrieval field of study

for authors. Each hyperedge represents an author and vertices in the corresponding hyperedge are researchers who have coauthored at least one paper with the author. There are 491 hyperedges and 506 vertices in this hypergraph. For this exploration, we use the following parameter setting: do not collapse hyperedges or vertices, hyperedge simplification, $s = 1$, gray out singletons, Jaccard weights, and $\varepsilon = 5.5$. Fig. 24a shows the unsimplified hypergraph, and although there seems to be a central clustering affinity, it is very hard to parse the information in the original hypergraph of this size.

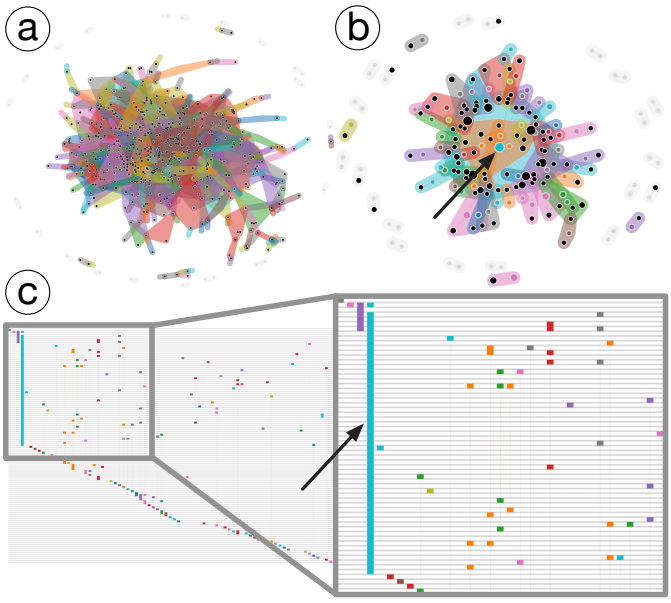


Fig. 24: A hypergraph of the *Coauthor Network*. (a) Original hypergraph. (b) A simplified hypergraph: the central hyperedge in blue (filled arrow) contains a number of prolific researchers with numerous papers and citations. (c) The rainbow box-based visualization of (b) highlighting the researchers that belong to the blue hyperedge/column (filled arrow).

Contrast Fig. 24a with the simplified hypergraph (at $s = 1$, $\varepsilon = 5.5$) in Fig. 24b, where the centrality of authorship is clearly evident. The central node (filled arrow) is the result of merging hyperedges under our framework. The central node contains prolific researchers in information retrieval with numerous citations. In particular, as illustrated by the rainbow box-based visualization of the simplified hypergraph in Fig. 24c, the central node consists of a hyperedge corresponding to **David A. Grossman** and **Ophir Frieder** who are the authors of the book "Information Retrieval: Algorithms and Heuristics", an important introductory textbook of the field.

7 Evaluation

The main contribution of this paper is a framework for topology-based simplifications of hypergraphs, which can be integrated with *any* hypergraph visualization technique. In other words, our hypergraph simplification framework is *agnostic* to the hypergraph visual encoding. On the other hand, to demonstrate the effectiveness of such simplifications, we evaluate the quality of hypergraph visualizations from Sect. 6 using four aesthetic criteria.

Contour intersections. Given *an Euler* diagram-based visualization of a hypergraph H, a contour intersection is a crossing of the boundaries of two convex hulls representing two hyperedges. To compute the number of contour intersections, we approximate the boundary of each convex hull using piecewise linear segments. The *approximated contour intersections*, denoted as m_i, is defined as the number of intersections among these line segments. Similarly, for a bubble sets-based visualization, the boundary of each bubble set is approximated by a set of linear segments, and m_i captures the intersections among them.

Number of edge crossings. Given a bipartite graph-based visualization of a hypergraph H, we compute its *number of edge crossings*, denoted as m_c, which is an aesthetic criterion first proposed by Purchase [42]. For the bipartite graph representation of H with vertex set V and edge set E, m_c is defined as

$$m_c = \begin{cases} 1 - \dfrac{c}{c_{max}}, & \text{if } c_{max} > 0 \\ 1, & \text{otherwise} \end{cases}$$

where c is the number of edge crossings, and c_{max} is the approximation of the upper bound of the number of edge crossings. c_{max} is defined as

$$c_{max} = \frac{|E|(|E|-1)}{2} - \frac{1}{2}\sum_{v \in V}(deg(v)(deg(v)-1))$$

where $deg(v)$ is the degree of a vertex v. Therefore, $0 \le m_c \le 1$, and $m_c = 1$ when there are no edge crossings in the bipartite graph.

Normalized edge length variation. We also evaluate the bipartite graph visualization of a hypergraph using the *normalized edge length variation* [43], denoted as m_l. For a graph with a vertex set V and an edge set E, Hachul and Jünger [69] proposed a *normalized standard deviation of the edge length* σ_l,

$$\sigma_l = \sqrt{\frac{\sum_{e \in E}(l_e - l_\mu)^2}{|E| \cdot l_\mu^2}}$$

where l_e is the length of an edge e, and l_μ is the mean of the edge length. Kwon *et al.* [43] then proposed a normalized version of σ_l, which is to divide σ_l by its upper bound $\sqrt{|E|-1}$, and the normalized edge length variation m_l is then defined as

$$m_l = \frac{\sigma_l}{\sqrt{|E|-1}}.$$

By definition, $0 \le m_l \le 1$. $m_l = 0$ when all edge lengths are equal.

Minimum angle. Finally, we work with the the *minimum angle between adjacent edges leaving a vertex* [42], denoted as m_a. Given a vertex v, m_a is defined based on the deviation of the minimum angle $\theta_{min}(v)$ between the adjacent incident edges from the ideal minimum angle $\theta(v)$, where $\theta(v) = \frac{360^\circ}{deg(v)}$. The average absolute deviation of minimum angles is defined as

$$d_\theta = \frac{1}{|V|}\sum_{v \in V}\left|\frac{\theta(v) - \theta_{min}(v)}{\theta(v)}\right|,$$

and the minimum angle metric m_a is defined as

$$m_a = 1 - d_\theta.$$

By definition, $0 \le m_a \le 1$. $m_a = 1$ when all the vertices have equal angles between all adjacent incident edges.

Evaluation results. We compute the above four criteria to evaluate the simplified hypergraphs in Sect. 6 across three visual encodings and their hybrids: the Euler diagram, the bipartite graph, and the bubble sets. For each hypergraph, we compute a given criterion

Data	m_i contour intersections			m_c edge crossings			m_l edge length			m_a minimum angle			size (B)		size (A)		bipartite #e	
	B	A	B/A	B	A	B/A	B	A	B/A	B	A	B/A	#v	#e	#v	#e	B	A
SW	164	139	**1.18**	0.94	0.95	**0.99**	0.05	0.08	0.625	0.32	0.46	**0.70**	18	14	10	12	89	46
LM	1246	962	**1.30**	0.98	0.94	1.04	0.04	0.06	0.67	0.56	0.54	1.04	80	45	18	36	276	99
HBP	76	57	**1.33**	0.99	0.86	1.15	0.02	0.05	0.40	0.92	0.27	3.41	1316	10	58	7	1858	152
DNS	656	51	**12.86**	0.96	0.93	1.03	0.04	0.13	0.31	0.45	0.58	**0.78**	57	30	15	9	246	36
CN	26162	278	**94.11**	0.98	0.99	**0.99**	0.02	0.05	0.4	0.67	0.86	**0.78**	506	491	128	81	1907	231

TABLE 1: For Euler diagram and bipartite graph-based visualizations, aesthetic metric values before (B) and after (A) simplification, and the ratio (R) between them. The simplification parameters are as follows. For *Southern Women* (SW) dataset, vertex simplification, $s = 1$, $\varepsilon = 1.6$. For Les Misérables (LM) dataset, vertex simplification, $s = 1$, $\varepsilon = 2.93$. For *Hallmark Biological Pathways* (HBP) dataset, edge simplification, $s = 1$, $\varepsilon = 7.86$. For *ActiveDNS* (DNS) dataset, edge simplification, $s = 3$, $\varepsilon = 2.4$. For *Coauthor Network* (CN) dataset, edge simplification, $s = 1$, $\varepsilon = 5.5$. The bold numbers in the ratio mean that the simplified visualization is better than the original visualization under the corresponding criterion.

Data	m_i contour intersections			m_c edge crossings			m_l edge length			m_a minimum angle			size (S)		bipartite #e
	B	S	B/S	B	S	B/S	B	S	B/S	B	S	B/S	#v	#e	S
SW	164	123	**1.33**	0.94	0.93	1.01	0.05	0.05	1.00	0.32	0.19	1.68	17	13	84
LM	1246	1211	**1.03**	0.98	0.97	1.01	0.04	0.04	1.00	0.56	0.44	1.27	56	44	239
HBP	76	82	0.93	0.99	0.94	1.05	0.02	0.03	0.67	0.92	0.46	2.00	112	10	314
DNS	656	669	0.98	0.96	0.95	1.01	0.04	0.04	1.00	0.45	0.32	1.41	34	30	191
CN	26162	25114	**1.04**	0.98	0.97	1.01	0.02	0.02	1.00	0.67	0.65	1.03	459	446	1811

TABLE 2: Aesthetic metric values of the original hypergraph (B) and the hypergraph after (strict) edge and vertex collapse (S).

Data	m_i contour intersections				
	B	S	A	B/S	B/A
SW	213	169	158	**1.26**	**1.35**
LM	2048	1924	1482	**1.06**	**1.38**
HBP	674	209	210	**3.22**	**3.21**
DNS	1233	1060	85	**1.16**	**14.51**
CN	58658	48534	260	**1.21**	**225.61**

TABLE 3: Bubble sets-based visualizations: aesthetic metrics.

Data	m_i contour intersections				
	B	S	A	B/S	B/A
SW	180	145	256	**1.24**	0.70
LM	2195	1991	1806	**1.10**	**1.22**
HBP	477	177	132	**2.69**	**3.61**
DNS	1073	1067	80	**1.01**	**13.41**
CN	68514	79440	338	0.86	**202.70**

TABLE 4: Bubble-bipartite visualizations: aesthetic metrics.

before (B) and after (A) simplification using the visualizations generated automatically by our tool without any modification. We also compute the ratio of the reported values before and after simplification for comparison (B/A). The criteria m_c, m_l, and m_a are computed using *GLAM* [43], [70].

The evaluation results for the Euler diagram and bipartite-based visualizations are shown in Table 1. For the Euler diagram-based visualization, our hypergraph simplification greatly improves the approximate number of contour intersections (m_i), in particular, for large and complex hypergraphs (e.g., the *Coauthor Networks*), which is indicated by a large ratio B/A that ranges between $1.18\times$ and $94.11\times$ in the table. The result is not surprising since, by design, our simplification framework reduces vertex and hyperedge density in a mathematically justifiable way. Meanwhile, we obtain comparable numbers of edge crossings m_c for the bipartite graph visualization before and after simplification, which is indicated by a ratio ≈ 1 before and after simplification. Such a ratio is only slightly elevated for three of the datasets. On the other hand, a bipartite graph visualization of the simplified hypergraph shows slightly higher normalized edge length variation (m_l). However, we consider the result to be acceptable since all values are very close to 0 (ranging from 0.01 to 0.13 across all datasets). Finally, we observe that our simplification method improves upon the minimum angle criterion m_a for three of the datasets after simplification, which is likely due to the fact that our simplification generally reduces the degree of a vertex in the bipartite visualization.

To further demonstrate the strength of our framework, we also compare against the results of strict hyperedge collapse and vertex collapse in Table 2. Strict vertex collapse for the *Southern Women* dataset has a slightly better ratio (i.e., 1.33) for contour intersections. However, our simplification framework is shown to be much more effective for the remaining four large datasets, improving m_i significantly in comparison, e.g., we obtain a $94.11\times$ improvement with our hyperedge simplification v.s. $1.04\times$ improvement with strict hyperedge collapse for the *Coauthor Network*.

Finally, for the bubble sets-based and bubble-bipartite hybrid visualizations, we demonstrate in Table 3 and Table 4 that our simplification framework drastically improves the contour intersections for larger datasets. In particular, for the *Coauthor Network*, the improvement in m_i is $202\times$ fold with the bubble-bipartite hybrid visualization, and $225\times$ fold with the bubble sets-based visualization, in comparison to the $94\times$ fold improvement using the Euler diagram. In summary, our simplification framework is shown to generally improve aesthetic metrics across four types of hypergraph visualizations for majority of our datasets. It introduces drastic improvement in comparison with the strict vertex/hyperedge collapse.

8 Discussion

In this paper, we introduce a framework that supports topological simplification of hypergraphs via both vertex and hyperedge simplifications. We exploit the duality between hypergraphs, line graphs, and clique expansions, and apply barcode-guided simplification of a hypergraph across multiple scales. We expect our framework to be applicable for general hypernetwork science (e.g., to be integrated with HyperNetX [6]). There are several interesting venues for future research.

We map a hypergraph to a metric space representation, where we use a shortest path metric in our current framework. Other notions of metrics, in particular, resistance distance [71], commute time distance [72], and diffusion distance [73], that are applicable in our unifying framework (see Sect. 4). These metrics can

be particularly interesting as they capture structural or physical properties of the underlying data.

Since the barcode used to guide the simplification can be obtained by computing the minimum spanning tree, the simplification process is equivalent to applying a single-linkage clustering with a threshold. Some other clustering approaches, such as average-linkage clustering and complete-linkage clustering, or more complex methods using hypergraph Laplacians [74] or modularity [75], might also be used to guide hypergraph simplifications. A comparison among these clustering approaches will be interesting. Finally, it will be interesting to explore hypergraph simplification using a higher dimensional barcode.

ACKNOWLEDGMENTS

This project was partially supported by NSF IIS 1513616 and DOE DE-SC0021015. This work was also partially funded under the High Performance Data Analytics (HPDA) program at the Department of Energy's Pacific Northwest National Laboratory. Pacific Northwest National Laboratory is operated by Battelle Memorial Institute under Contract DE-ACO6-76RL01830.

REFERENCES

[1] C. A. Joslyn, S. Aksoy, D. Arendt, J. Firoz, L. Jenkins, B. Praggastis, E. A. Purvine, and M. Zalewski, "Hypergraph analytics of domain name system relationships," in *Procceedings of the 17th Workshop on Algorithms and Models for the Web Graph, Lecture Notes in Computer Science*, 2020.

[2] M. Ashburner, C. A. Ball, J. A. Blake, D. Botstein, H. Butler, J. M. Cherry, A. P. Davis, K. Dolinski, S. S. Dwight, J. T. Eppig, M. A. Harris, D. P. Hill, L. Issel-Tarver, A. Kasarskis, S. Lewis, J. C. Matese, J. E. Richardson, M. Ringwald, G. M. Rubin, and G. Sherlock, "Gene ontology: tool for the unification of biology," *Nature Genetics*, vol. 25, no. 1, pp. 25–9, 2000.

[3] The Gene Ontology Consortium, "The gene ontology resource: 20 years and still GOing strong," *Nucleic Acids Research*, vol. 47, no. D1, pp. D330–D338, 2019.

[4] A. Liberzon, C. Birger, H. Thorvaldsdóttir, M. Ghandi, J. P. Mesirov, and P. Tamayo, "The molecular signatures database Hallmark gene set collection," *Cell systems*, vol. 1, no. 6, pp. 417–425, 2015.

[5] A. Subramanian, P. Tamayo, V. K. Mootha, S. Mukherjee, B. L. Ebert, M. A. Gillette, A. Paulovich, S. L. Pomeroy, T. R. Golub, E. S. Lander, and J. P. Mesirov, "Gene set enrichment analysis: A knowledge-based approach for interpreting genome-wide expression profiles," *Proceedings of the National Academy of Sciences of the United States of America (PNAS)*, vol. 102, no. 43, pp. 15 545–15 550, 2005.

[6] B. Praggastis, D. Arendt, E. Purvine, C. Joslyn, M. Raugas, S. Aksoy, and K. Monson, "HyperNetX," https://github.com/pnnl/HyperNetX, 2018.

[7] G. Carlsson, A. J. Zomorodian, A. Collins, and L. J. Guibas, "Persistence barcodes for shapes," *Proceedings Eurographs/ACM SIGGRAPH Symposium on Geometry Processing*, pp. 124–135, 2004.

[8] D. Cohen-Steiner, H. Edelsbrunner, and J. Harer, "Stability of persistence diagrams," *Discrete & Computational Geometry*, vol. 37, pp. 103–120, 2007.

[9] J.-C. Bermond, M.-C. Heydemann, and D. Sotteau, "Line graphs of hypergraphs I," *Discrete Mathematics*, vol. 18, no. 3, pp. 235–241, 1977.

[10] J. Zien, M. Schlag, and P. K. Chan, "Multi-level spectral hypergraph partitioning with arbitrary vertex sizes," *IEEE Transactions on Computer-Aided Designof Integrated Circuits and Systems*, vol. 18, pp. 1389–1399, 1999.

[11] I. Herman, G. Melancon, and M. S. Marshall, "Graph visualization and navigation in information visualization: A survey," *IEEE Transactions of Visualization and Computer Graphics*, vol. 6, no. 1, pp. 24–43, 2000.

[12] C. Wang and J. Tao, "Graphs in scientific visualization: A survey," *Computer Graphic Forum*, vol. 36, no. 1, pp. 263–287, 2017.

[13] 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*, vol. 30, no. 6, pp. 1719–1749, 2011.

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

[15] P. Eades and R. Tamassia, "Algorithms for drawing graphs: An annotated bibliography," *Computational Geometry: Theory and Applications*, vol. 4, no. 5, pp. 235–282, 1994.

[16] E. Mäkinen, "How to draw a hypergraph," *International Journal of Computer Mathematics*, vol. 34, no. 3-4, pp. 177–185, 1990.

[17] N. A. Arafat and S. Bressan, "Hypergraph drawing by force-directed placement," in *International Conference on Database and Expert Systems Applications*, 2017, pp. 387–394.

[18] J. Paquette and T. Tokuyasu, "Hypergraph visualization and enrichment statistics: How the EGAN paradigm facilitates organic discovery from big data," in *Human Vision and Electronic Imaging XVI*, vol. 7865, no. 78650E. International Society for Optics and Photonics, 2011.

[19] B. Alsallakh, L. Micallef, W. Aigner, H. Hauser, S. Miksch, and P. Rodgers, "The state-of-the-art of set visualization," *Computer Graphics Forum*, vol. 35, no. 1, pp. 234–260, 2016.

[20] L. Euler, "Lettres á une princesse d'allemagne," *Letters 102-105*, 1761.

[21] M. Kritz and K. Perlin, "A new scheme for drawing hypergraphs," *International journal of computer mathematics*, vol. 50, no. 3-4, pp. 131–134, 1994.

[22] N. H. Riche and T. Dwyer, "Untangling Euler diagrams," *IEEE Transactions on Visualization and Computer Graphics*, vol. 16, no. 6, pp. 1090–1099, 2010.

[23] P. Simonetto, D. Auber, and D. Archambault, "Fully automatic visualisation of overlapping sets," *Computer Graphic Forum*, vol. 28, no. 3, pp. 967–974, 2009.

[24] P. Simonetto, "Visualisation of overlapping sets and clusters with Euler diagrams," Ph.D. dissertation, University of Bordeaux, 2011.

[25] B. Alper, N. Riche, G. Ramos, and M. Czerwinski, "Design study of linesets, a novel set visualization technique," *IEEE transactions on visualization and computer graphics*, vol. 17, no. 12, pp. 2259–2267, 2011.

[26] C. Collins, G. Penn, and S. Carpendale, "Bubble Sets: Revealing set relations with isocontours over existing visualizations," *IEEE Transactions on Visualization and Computer Graphics*, vol. 15, no. 6, pp. 1009–1016, 2009.

[27] A. Efrat, Y. Hu, S. G. Kobourov, and S. Pupyrev, "MapSets: Visualizing embedded and clustered graphs," *Journal of Graph Algorithms and Applications*, vol. 19, no. 2, pp. 571–593, 2015.

[28] A. Lex, N. Gehlenborg, H. Strobelt, R. Vuillemot, and H. Pfister, "UpSet: visualization of intersecting sets," *IEEE Transactions on Visualization and Computer Graphics*, vol. 20, no. 12, pp. 1983–1992, 2014.

[29] P. Rodgers, G. Stapleton, and P. Chapman, "Visualizing sets with linear diagrams," *ACM Transactions on Computer-Human Interaction*, vol. 22, no. 6, pp. 1–39, 2015.

[30] B. Jacobsen, M. Wallinger, S. Kobourov, and M. Nöllenburg, "MetroSets: Visualizing sets as metro maps," *IEEE Transactions on Visualization and Computer Graphics*, vol. 27, no. 2, pp. 1257–1267, 2020.

[31] J.-B. Lamy, "Visualizing undirected graphs and symmetric square matrices as overlapping sets," *Multimedia Tools and Applications*, vol. 78, no. 23, pp. 33 091–33 112, 2019.

[32] P. Valdivia, P. Buono, C. Plaisant, N. Dufournaud, and J.-D. Fekete, "Analyzing dynamic hypergraphs with Parallel Aggregated Ordered Hypergraph visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 27, no. 1, pp. 1–13, 2021.

[33] A. Kerren and I. Jusufi, "A novel radial visualization approach for undirected hypergraphs," in *Proceedings of the 17th Eurographics Conference on Visualization, Short paper track*, 2013.

[34] U. Brandes, S. Cornelsen, B. Pampel, and A. Sallaberry, "Path-based supports for hypergraphs," *Journal of Discrete Algorithms*, vol. 14, pp. 248–261, 2012.

[35] K. Buchin, M. van Kreveld, H. Meijer, B. Speckmann, and K. Verbeek, "On planar supports for hypergraphs," *International Symposium on Graph Drawing*, pp. 345–356, 2009.

[36] A. A. Zykov, "Hypergraphs," *Russian Mathematical Surveys*, vol. 29, no. 6, 1974.

[37] H. Gropp, "The drawing of configurations," *International Symposium on Graph Drawing*, pp. 267–276, 1995.

[38] W. Evans, P. Rzążewski, N. Saeedi, C.-S. Shin, and A. Wolff, "Representing graphs and hypergraphs by touching polygons in 3D," in *International Symposium on Graph Drawing and Network Visualization*, 2019, pp. 18–32.

[39] B. Qu, E. Zhang, and Y. Zhang, "Automatic polygon layout for primal-dual visualization of hypergraphs," *IEEE Transactions on Visualization and Computer Graphics*, vol. 28, no. 1, pp. 633–642, 2021.

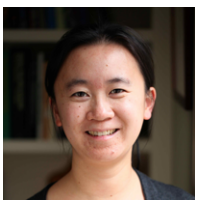
[40] A. Meidiana, S.-H. Hong, P. Eades, and D. Keim, "A quality metric for visualization of clusters in graphs," in *International Symposium on Graph Drawing and Network Visualization*. Springer, 2019, pp. 125–138.

[41] Q. Nguyen, P. Eades, and S.-H. Hong, "On the faithfulness of graph visualizations," in *International Symposium on Graph Drawing*. Springer, 2012, pp. 566–568.
[42] H. C. Purchase, "Metrics for graph drawing aesthetics," *Journal of Visual Languages & Computing*, vol. 13, no. 5, pp. 501–516, 2002.
[43] O.-H. Kwon, T. Crnovrsanin, and K.-L. Ma, "What would a graph look like in this layout? a machine learning approach to large graph visualization," *IEEE transactions on visualization and computer graphics*, vol. 24, no. 1, pp. 478–488, 2017.
[44] M. Purohit, B. A. Prakash, C. Kang, Y. Zhang, and V. Subrahmanian, "Fast influence-based coarsening for large networks," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014, pp. 1296–1305.
[45] K. Shin, A. Ghoting, M. Kim, and H. Raghavan, "SWeG: Lossless and lossy summarization of web-scale graphs," in *The World Wide Web Conference*, 2019, pp. 1679–1690.
[46] M. A. Beg, M. Ahmad, A. Zaman, and I. Khan, "Scalable approximation algorithm for graph summarization," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2018, pp. 502–514.
[47] Z. Shen, K.-L. Ma, and T. Eliassi-Rad, "Visual analysis of large heterogeneous social networks by semantic and structural abstraction," *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 6, pp. 1427–1439, 2006.
[48] K. Lee, H. Jo, J. Ko, S. Lim, and K. Shin, "SSumM: Sparse summarization of massive graphs," *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 144–154, 2020.
[49] D. Koutra, U. Kang, J. Vreeken, and C. Faloutsos, "VoG: Summarizing and understanding large graphs," *Proceedings of the 2014 SIAM international conference on data mining*, pp. 91–99, 2014.
[50] N. Shah, D. Koutra, T. Zou, B. Gallagher, and C. Faloutsos, "TimeCrunch: Interpretable dynamic graph summarization," *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1055–1064, 2015.
[51] C. Dunne and B. Shneiderman, "Motif simplification: improving network visualization readability with fan, connector, and clique glyphs," *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 3247–3256, 2013.
[52] 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*, vol. 26, no. 1, pp. 697–707, 2020.
[53] L. Lemonnier, J. van de Wetering, and A. Kissinger, "Hypergraph simplification: Linking the path-sum approach to the ZH-calculus," *arXiv preprint arXiv:2003.13564*, 2020.
[54] S. G. Aksoy, C. Joslyn, C. O. Marrero, B. Praggastis, and E. Purvine, "Hypernetwork science via high-order hypergraph walks," *EPJ Data Science*, vol. 9, no. 1, p. 16, 2020.
[55] R. Ghrist, "Barcodes: The persistent topology of data," *Bullentin of the American Mathematical Society*, vol. 45, pp. 61–75, 2008.
[56] H. Edelsbrunner, D. Letscher, and A. Zomorodian, "Topological persistence and simplification," *Discrete & Computational Geometry*, vol. 28, pp. 511–533, 2002.
[57] M. Hajij, B. Wang, C. Scheidegger, and P. Rosen, "Visual detection of structural changes in time-varying graphs using persistent homology," *IEEE Pacific Visualization Symposium*, 2018.
[58] H. Edelsbrunner and J. Harer, "Persistent homology - a survey," *Contemporary Mathematics*, vol. 453, pp. 257–282, 2008.
[59] J. C. Gower and G. J. Ross, "Minimum spanning trees and single linkage cluster analysis," *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, vol. 18, no. 1, pp. 54–64, 1969.
[60] S. Gerber, P.-T. Bremer, V. Pascucci, and R. Whitaker, "Visual exploration of high dimensional scalar functions," *IEEE Transactions on Visualization and Computer Graphics*, vol. 16, pp. 1271–1280, 2010.
[61] A. Davis, B. B. Gardner, and M. R. Gardner, *Deep South: A social anthropological study of caste and class*. University of South Carolina Press, 2009.
[62] L. C. Freeman, "Finding social groups: A meta-analysis of the southern women data," http://moreno.ss.uci.edu/86.pdf, 2003.
[63] D. E. Knuth, *The Stanford GraphBase: a platform for combinatorial computing*. New York: ACM Press and Addison-Wesley Publishing Company, 1994.
[64] A. Kountouras, P. Kintis, C. Lever, Y. Chen, Y. Nadji, D. Dagon, M. Antonakakis, and R. Joffe, "Enabling network security through active dns datasets," in *International Symposium on Research in Attacks, Intrusions, and Defenses*. Springer, 2016, pp. 188–208.
[65] S. Aksoy, S. Harun, L. Jenkins, C. Joslyn, C. Lightsey, H. Medal, D. Mentgen, T. Stavenger, T. Bhuiyan, and M. Zalewski, "Chapel Hypergraph Library," https://github.com/pnnl/CHGL, 2018.
[66] L. P. Jenkins, T. Bhuiyan, S. Harun, C. Lightsey, S. Aksoy, T. Stavenger, M. Zalewski, H. Medal, and C. Joslyn, "Chapel Hypergraph Library (CHGL)," in *IEEE High Performance Extreme Computing Conference*, 2018.
[67] "Hurricane Electric BGP Toolkit," https://bgp.he.net/.
[68] J. Tang, J. Zhang, L. Yao, J. Li, L. Zhang, and Z. Su, "ArnetMiner: Extraction and miningof academic social networks," in *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2008, pp. 990–998.
[69] S. Hachul and M. Jünger, "Large-graph layout algorithms at work: An experimental study," *Journal of Graph Algorithms and Applications*, vol. 11, no. 2, pp. 345–369, 2007.
[70] O.-H. Kwon, T. Crnovrsanin, and K.-L. Ma, "GLAM: Graph layout aesthetic metrics," https://github.com/VIDILabs/glam.
[71] D. J. Klein and M. Randic, "Resistance distance," *Journal of Mathematical Chemistry*, vol. 12, pp. 81–95, 1993.
[72] F. Fouss, A. Pirotte, J. michel Renders, and M. Saerens, "Random-walk computation of similarities between nodes of a graph, with application to collaborative recommendation," *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, no. 3, pp. 355–369, 2007.
[73] R. R. Coifman, S. Lafon, A. B. Lee, M. Maggioni, B. Nadler, F. Warner, and S. W. Zucker, "Geometric diffusions as a tool for harmonic analysisand structure definition of data: Diffusion maps," *Proceedings of the National Academy of Sciences of the United States of America (PNAS)*, vol. 102, no. 21, pp. 7426–7431, 2005.
[74] K. Hayashi, S. G. Aksoy, C. H. Park, and H. Park, "Hypergraph random walks, Laplacians, and clustering," in *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, 2020, pp. 495–504.
[75] B. Kamiński, V. Poulin, P. Prałat, P. Szufel, and F. Théberge, "Clustering via hypergraph modularity," *PloS one*, vol. 14, no. 11, p. e0224307, 2019.

Youjia Zhou is a PhD student at the School of Computing and the Scientific Computing and Imaging (SCI) Institute, University of Utah. Her research focuses on developing visual analytics systems for large and complex data, in particular, networks, high-dimensional point clouds, and vector/tensor fields, using topological techniques.

Archit Rathore is a PhD student at the School of Computing and the Scientific Computing and Imaging (SCI) Institute, University of Utah. His current research focuses on probing machine learning models through visualization techniques to improve interpretability.

Emilie Purvine is a Senior Data Scientist at Pacific Northwest National Laboratory (PNNL). She received a Ph.D. in Mathematics from Rutgers University in May 2011 with a focus on experimental mathematics and nonlinear recurrence relations. At PNNL she is now focused on applications of combinatorics and computational topology together with theoretical advances needed to support the applications, ranging from computational chemistry and biology to cyber security and power grid modeling.



Bei Wang is an assistant professor at the School of Computing and a faculty member at the Scientific Computing and Imaging (SCI) Institute, University of Utah. She received her Ph.D. in Computer Science from Duke University. She is interested in the analysis and visualization of large and complex data. Her research interests include topological data analysis, data visualization, computational topology, computational geometry, machine learning and data mining.