




Structure-Aware Simplification for Hypergraph Visualization

Peter Oliver , Eugene Zhang , Senior Member, IEEE, and Yue Zhang , Member, IEEE

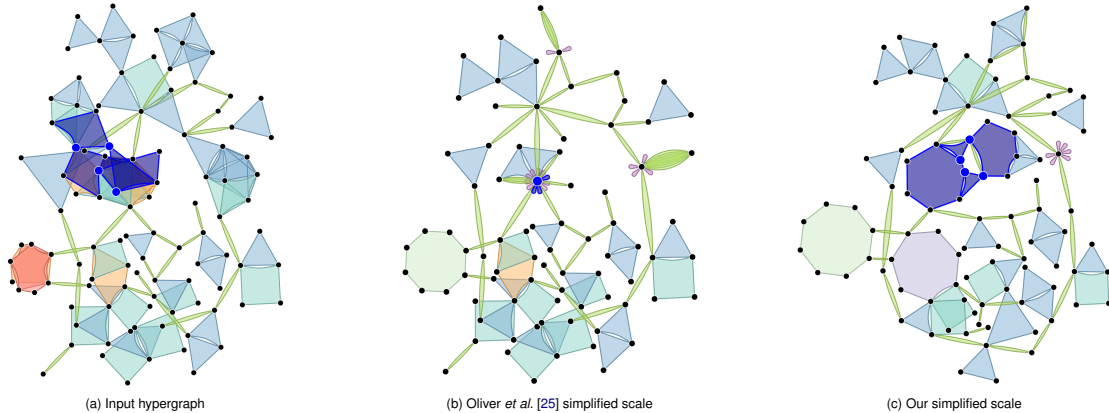


Fig. 1: A hypergraph network representing friendships between 92 high school students in Marseilles [23] is visualized using the polygon visualization metaphor. Visual clutter caused by overlapping polygons in (a) makes it difficult to identify cycle structures such as the one highlighted in blue. The simplification method of Oliver *et al.* [25] shown in (b) reduces the polygon overlaps, but collapses the highlighted cycle into a single hypergraph vertex. This failure to preserve the cycle makes the interpretation of this intermediate scale less reliable. Our new structure-aware simplification method shown in (c) efficiently reduces polygon overlaps while preserving structures in the data such as the cycle. Our method enables multi-scale hypergraph visualizations where the simplified scales can be used to more easily identify meaningful structures in the data.

Abstract—Hypergraphs provide a natural way to represent polyadic relationships in network data. For large hypergraphs, it is often difficult to visually detect structures within the data. Recently, a scalable polygon-based visualization approach was developed allowing hypergraphs with thousands of hyperedges to be simplified and examined at different levels of detail. However, this approach is not guaranteed to eliminate all of the visual clutter caused by unavoidable overlaps. Furthermore, meaningful structures can be lost at simplified scales, making their interpretation unreliable. In this paper, we define hypergraph structures using the bipartite graph representation, allowing us to decompose the hypergraph into a union of structures including topological blocks, bridges, and branches, and to identify exactly where unavoidable overlaps must occur. We also introduce a set of topology preserving and topology altering atomic operations, enabling the preservation of important structures while reducing unavoidable overlaps to improve visual clarity and interpretability in simplified scales. We demonstrate our approach in several real-world applications.

Index Terms—Hypergraph Visualization, Hypergraph Simplification, Hypergraph Topology, Bipartite Representation

1 INTRODUCTION

Polyadic relationships are omnipresent in network data with applications ranging from social networks, to paper authorship, and biology. Hypergraphs, as extensions to graphs, provide an ideal model for polyadic relationships where each relationship is represented as a *hyperedge*. The incident *vertices* of the hyperedge represent the entities in the underlying relationship. Any analysis of polyadic relationship data requires efficient visualization of the corresponding hypergraphs.

There have been recent advances in hypergraph visualization [3], with a focus on identifying an appropriate visual metaphor for hyper-

edges. Qu *et al.* [26] propose the use of an N -sided polygon for each N -ary polyadic relationship in the data. In this *polygon visualization metaphor*, the vertices of the polygons coincide with the entities in the N -ary relationship. In their follow-up research, Qu *et al.* [27] develop an optimization framework that generates high-quality polygon layouts for hypergraphs and their dual hypergraphs, where the roles of the vertices and hyperedges are switched. However, overlaps between polygons can make it challenging to differentiate individual polygons and correctly identify their shared vertices, especially for large datasets. Figure 1 (a) contains several examples. In addition, their optimization often becomes trapped at local minima in its objective function for datasets with more than a few hundred elements, leading to suboptimal layouts. To address this, Oliver *et al.* [25] introduce a layout optimization framework in which the complexity of a hypergraph is iteratively reduced through a set of atomic simplification operations. Once the reduced hypergraph is sufficiently simple, an optimized layout is generated. From there, a sequence of inverse simplification operations is executed to gradually bring the complexity back to that of the original hypergraph. With each inverse operation, the layout is locally optimized and updated before continuing with the next inverse operation. This leads to a gain in layout quality and a reduction in the time required to produce such a layout. However, hypergraphs have structural features that can be lost at intermediate simplified scales, limiting their usefulness for multi-scale visualizations where significant structures should be recognizable in each scale. Figure 1 (b) shows an

- Peter Oliver is with the School of Electrical Engineering and Computer Science, Oregon State University. E-mail: oliverpe@oregonstate.edu.
- Eugene Zhang is a Professor with the School of Electrical Engineering and Computer Science, Oregon State University. E-mail: zhang@eecs.oregonstate.edu.
- Yue Zhang is an Associate Professor with the School of Electrical Engineering and Computer Science, Oregon State University. E-mail: zhangyue@oregonstate.edu.

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

intermediate hypergraph scale created using Oliver et al.’s approach. Notice that the highlighted cycle structure in (a) is no longer present in (b). This cycle represents separation between friendship groups which is not easily visible in (a) due to polygon overlaps, and is completely removed in (b) where the friendship groups and the separation between them are collapsed into a single vertex. In addition, some visual clutter can persist in simplified scales, as shown in (b) by the remaining overlapping polygons, raising a fundamental question: is the visual clutter an artifact of the layout algorithm or is it unavoidable?

In this paper, we introduce a structure-aware approach to hypergraph simplification. We propose a novel decomposition of a hypergraph into an edge disjoint union of topological blocks, branches, and bridges. These structures can play an important role in interpreting hypergraph data. For example, the quantity and size of interweaving cycles in a social group hypergraph dataset can tell us how close-knit a particular group is. Alternatively, the presence of cycles in a contact tracing dataset indicates multiple possible transmission vectors for an infectious disease, namely, along either side of the cycle. Bridges and branches in a paper-author dataset can indicate links between research groups and help identify work that is on the outskirts of a research community. Our decomposition method also enables exact identification of areas in the hypergraph where unavoidable overlaps occur, which are a primary source of visual clutter in polygon visualizations.

At the core of our decomposition and identification of unavoidable overlaps is the use of bipartite graphs as the foundation of hypergraph analysis. The bipartite graph representation replaces both the vertices and hyperedges of the hypergraph with graph nodes, using graph edges to indicate incidence relationships in the hypergraph. We define a *topological block* as a maximal biconnected component in this bipartite graph. We present an efficient algorithm for computing a cycle basis for the bipartite graph, leading to a fast implementation of our decomposition. We also develop new theory showing how the bipartite cycles can be used to identify unavoidable overlaps in polygon visualizations. In particular, we define the *entanglement index* as the ratio between the first Betti number (number of basis cycles) and the total number of elements in a topological block. We further show that unavoidable overlaps occur only in entangled topological blocks which we call *forbidden clusters*. Powered by this analysis, we propose a new set of atomic simplification operations, inspired by the operations from [25], using the nodes and edges of the bipartite graph representation as the fundamental units. We include both topology preserving and topology altering operations. A topology preserving operation does not affect the number of linearly independent bipartite cycles in the *cycle basis*, while a topology altering operation can decrease the number of independent cycles by one. Our hypergraph decomposition paired with the identified unavoidable overlaps allows us to determine exactly how and where the different types of operations should be applied to reduce visual clutter while preserving the most salient structures in the hypergraph. This also leads to a more efficient technique for producing planar simplified hypergraphs compared to [25]. We include three use cases with real-world datasets and provide our interpretations of the results.

2 PREVIOUS WORK

Here we review recent work in network analysis and visualization that is most relevant to our paper.

Hypergraph Visualization. Much of the recent work on hypergraph visualization has focused on identifying effective visual representations of hyperedges using matrices [19, 21, 30, 36], regions [2, 4, 24, 28, 29, 31–34], metro lines [11, 17, 37], and bipartite graphs [2, 8, 35]. Matrix based methods enable spectral methods for analysis but do not as easily show structures in the data. Metro line visualizations can effectively display branching structures but are not ideal for representing topological structures like cycles. We use a bipartite graph representation for analysis, but for visualization, we adopt a region based approach of Qu et al. [27]. In their approach, each hyperedge is represented as a polygon drawn between the corresponding vertices embedded in the plane. By optimizing the positions of the vertices so that each polygon is as near to regular as possible, they generate high quality polygon layouts where the cardinality of a hyperedge

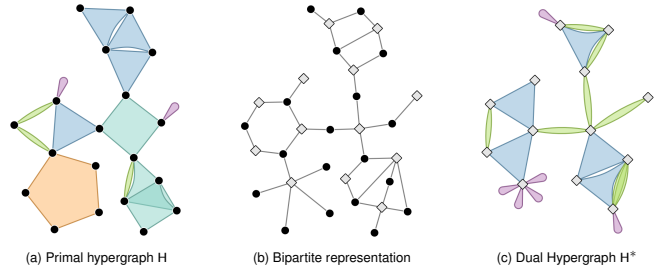


Fig. 2: The bipartite representations (b) of the primal hypergraph (a) and dual hypergraph (b) are identical. We use a black dot to represent a node in the primal hypergraph and a gray diamond to represent a node in the dual hypergraph.

can be identified through easily recognizable shapes.

In recent years, simplification has been used to reduce clutter and enhance readability in hypergraph visualizations. Oliver et al. [25] build on the layout optimization framework of [27], developing a multi-scale optimization method that is more effective in reducing polygon overlaps in the layout. Recognizing the potential of using simplified scales to study hypergraph structures, we build on the simplification scheme of [25], augmenting their statistic based approach with a new structure-aware implementation that can precisely identify sources of unavoidable overlap. Zhou et al. [38] apply persistent homology techniques to compute a *barcode* for a chosen graph representation of the input hypergraph, which is used to guide merging operations. Their aim is to reduce the size of the data to create more compact and readable visualizations. In contrast, Oliver et al. [25] apply simplifications directly to the hypergraph vertices and hyperedges, guided by a customizable priority measure. They aim to provide scalable visualization techniques for very large datasets. Neither approach directly addresses visual clutter caused by unavoidable overlaps. To achieve simultaneous vertex and hyperedge simplification, we apply simplifications to the bipartite graph representation.

Hypergraph Structures. Recent work from Fan et al. [10] focuses on cycles in graph data with the aim to improve quantification on important nodes. However, such analysis is not focused on bipartite graphs which correspond to hypergraphs. Considerable effort has been made to describe connectedness in various mathematical representations. In addition to prominent work by Erdős [9] and Berge [5] on hypergraph combinatorics, Aksoy et al. [1] introduce concepts of high-order s -walks and s -paths which have concepts of both length and width. Such structures are powerful tools to analyze the connectivity of a hypergraph. To our knowledge, we are the first to explore the decomposition of a hypergraph into its constituent structures to guide structure-aware simplification and identify unavoidable visual clutter.

3 BACKGROUND

A hypergraph $H = \langle V, E \rangle$ on a finite set of vertices V is defined by a family of hyperedges E . A hyperedge $e \in E$ contains a non-empty subset of vertices $V_e \subseteq V$ which are *incident* to e and *adjacent* to each other. Similarly, a vertex $v \in V$ is contained by a subset of hyperedges $E_v \subseteq E$ which are *incident* to v and *adjacent* to each other. Following Oliver et al. [25], let E_e denote the set of hyperedges adjacent to e and V_v the set of vertices adjacent to v . Consistent with [27], we define the *degree* of a vertex v as $\deg(v) = |E_v|$ and the *cardinality* of a hyperedge e as $\text{card}(e) = |V_e|$.

The *dual hypergraph* $H^* = \langle V^*, E^* \rangle$ of H is obtained by switching the roles of vertices and hyperedges in H (Figure 2 (c)). We refer to the original hypergraph H as the *primal hypergraph*. More precisely, each element $v \in V$ corresponds to a unique element $v^* \in E^*$ and each element $e \in E$ corresponds to a unique element $e^* \in V^*$. The incidence and adjacency relationships of corresponding elements in the primal and dual hypergraphs are identical. This means that the primal and dual hypergraphs share combinatorial features and properties such as linearity and planarity [5, 25].

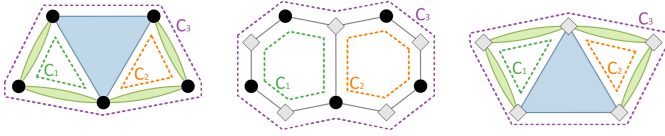


Fig. 3: The three simple cycles C_1, C_2, C_3 of a bipartite graph (center) are highlighted by the green, orange, and purple dotted lines. The same cycles are highlighted in the matching primal hypergraph (left) and dual hypergraph (right). Cycles C_1 and C_2 can be combined to form C_3 .

For $H = (V, E)$, the bipartite representation (also called the König representation) $G(H) = (X \cup Y, D)$ is a bipartite graph with nodes $x \in X$ for every vertex in V and nodes $y \in Y$ for every hyperedge in E (Figure 2 (b)). There is a one-to-one correspondence between V and X , so without ambiguity, we write $X = V$. Similarly, there is a one-to-one correspondence between Y and E , i.e. every node in Y corresponds precisely to one hyperedge in H . Again, we write $Y = E$. Vertices $x \in X$ and $y \in Y$ form an edge $(x, y) \in D$ if and only if x is part of the hyperedge associated with y . We can also define the bipartite representation in terms of the dual hypergraph H^* , resulting in an equivalent graph G^* . Thus, we identify G and G^* as one graph. A node in G that is a vertex in H we refer to as a *primal node* and is drawn as a black dot in our visualizations (Figure 2). A node in G that corresponds to a hyperedge in H (or equivalently a vertex in the dual hypergraph) is referred to as a *dual node* and is drawn as a grey diamond. Notice that an edge in D must link a primal node to a dual node, thus G is bipartite.

Bretto [6] defines a *path* P in H from u to v , $u, v \in V$, as an alternating sequence $P = \langle u = x_1, e_1, x_2, e_2, \dots, x_n, e_n, x_{n+1} = v \rangle$ where

- $x_1, x_2, \dots, x_n \in V$ are distinct vertices,
- $e_1, e_2, \dots, e_n \in E$ are distinct hyperedges,
- $x_i, x_{i+1} \in e_i$ for $1 \leq i \leq n$.

H is *connected* if every distinct pair of vertices is linked by some path P . If $u = x_1 = x_{n+1} = v$, then P is a *cycle*. Figure 3 outlines three such cycles in the primal, bipartite, and dual visualizations. This definition of a hypergraph cycle is consistent with Berge [5], and is analogous to a simple cycle in a graph where none of the vertices is repeated. We assume that all graph and hypergraph cycles discussed in the remainder of the paper are simple.

The simple cycles of the bipartite representation G live within *2-connected components*. Any vertex in G whose removal makes G disconnected, or increases the number of connected components, is called an *articulation vertex*. A connected graph that has no articulation vertices is called *2-connected*. A graph can contain multiple *2-connected* subgraphs. A maximal connected subgraph in G containing no articulation vertices is also called a *block*. A block on more than two vertices is called a *2-connected* (or *biconnected*) component. Blocks can consist of an isolated node or a single edge, while the smallest *2-connected* component is a triangle. Cycles cannot exist outside of a *2-connected* component without containing duplicate nodes or edges.

4 TOPOLOGY GUIDED DECOMPOSITION

While past analysis of hypergraphs has focused on local behaviors, such as the cardinality of a hyperedge or the intersections between hyperedges, hypergraphs have intricate structures that are global in nature. For example, cycles involving multiple vertices and hyperedges can exist, and a pair of cycles can be connected by a single path without which they would be disconnected. In this section, we define a number of hypergraph structures and develop a decomposition for hypergraphs into a union of these features. We also provide an efficient algorithm to compute the decomposition. The structures lead to a number of atomic simplification operations, enabling multi-scale representations of hypergraphs in which important features are preserved at simplified scales, providing meaningful visual interpretations.

4.1 Topological Blocks, Bridges, and Branches

Notice that a path P in H induces a subgraph in the bipartite representation which is a graph path between alternating primal and dual nodes in

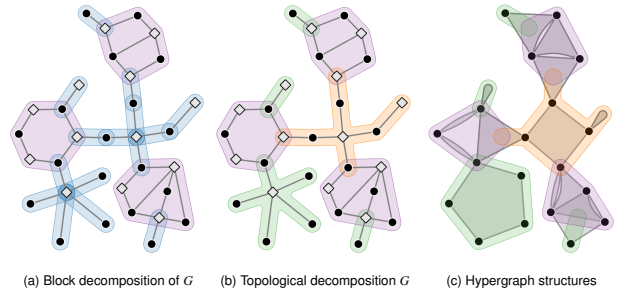


Fig. 4: We use our block decomposition (a) to generate a topological decomposition (b) of the bipartite graph representation for the hypergraph in Figure 2. In (a), the blue bubbles indicate single edge blocks and the purple bubbles multi-edge blocks. This leads to a number of extracted structures in (c) including topological blocks (purple bubbles), bridges (orange bubbles), and branches (green bubbles).

G . If P is a cycle, the subgraph in G induced by P is also a cycle. The reverse is also true if we relax the definition of a hypergraph cycle to allow the starting and ending points to be hyperedges, i.e., the vertices of a path in G induce hypergraph paths in H and H^* . Thus, we can identify cycles in H and H^* using the cycles of G and vice versa (Figure 3). In this paper, we treat all hypergraphs, and by extension their bipartite representations, as unweighted. Thus, the weight of any path in G is simply the length of the path. We use the bipartite path length to describe the lengths of cycles in the primal and dual hypergraphs as well, i.e., a hypergraph cycle containing n vertices has an overall length of $2n$ since it must also contain n hyperedges. The smallest possible cycle contains two vertices and two hyperedges, has a length of 4, and is represented in G as the complete bipartite graph $K_{2,2}$. We refer to these as *minimal cycles*.

We also consider the topology of H in terms of the topology of the bipartite graph G . Using the language of homology [18], the topology of $G(H)$ can be measured in terms of its *Betti numbers*. The zeroth Betti number, B_0 , is the number of connected components in G . In our datasets, all of the hypergraphs are connected, so $B_0 = 1$. On the other hand, the first Betti number, B_1 , is the number of independent cycles in the graph. A cycle can be considered as the combination of two or more other cycles as shown in Figure 3. In this case, only two of the three cycles are mutually independent, i.e., $B_1 = 2$.

In fact, there exists a *cycle basis* $\mathcal{C} = \{C_1, \dots, C_p\}$ where each C_i ($1 \leq i \leq p$) is an independent cycle in G , and any cycle of G not in \mathcal{C} can be written as a combination of two or more cycles in \mathcal{C} . Here, cycles are combined using the symmetric difference operator. Such a basis \mathcal{C} spans the *cycle space* of G . Furthermore, $\mathcal{C} - \{C_i\}$ for any index i no longer spans the cycle space so it is not a cycle basis. There are a number of choices for a cycle basis of G , however, the number of cycles p in any basis is given by the first Betti number B_1 . A common problem in graph theory is to find a cycle basis such that the sum of all basis cycle weights is minimum. Such a basis is called a *minimal cycle basis*. Horton [14] prove that a minimum cycle basis consists only of *tight* cycles. A cycle C is *tight* if the shortest path between every pair of nodes in C is a sub-sequence of C . In Fig. 3, C_1 and C_2 are tight, but C_3 is not tight because it does not contain the shortest path between the middle primal and dual nodes. Furthermore, each of the pairings $\{C_1, C_2\}$, $\{C_1, C_3\}$, and $\{C_2, C_3\}$ define a cycle basis, but $\{C_1, C_2\}$ is the only minimum cycle basis and it contains tight cycles.

The topological structure of H naturally leads to a decomposition of both the hypergraph and the bipartite representation G . Let \mathcal{C} be a cycle basis for H . Within \mathcal{C} , we consider two basis cycles C_1 and C_2 to be connected if they share a common edge in G , i.e., they share at least one primal node and one dual node. Suppose that \mathcal{C} has q connected components $\{T_1, \dots, T_q\}$. We refer to each connected component T_i as a *topological block*. Let $K = G - \mathcal{C}$. Then $K = \bigcup_{i=1}^q K_i$ is the disjoint union of K_i 's where each subgraph K_i is a tree. This is illustrated in Figure 4 (a) where the purple bubbles indicate extracted topological

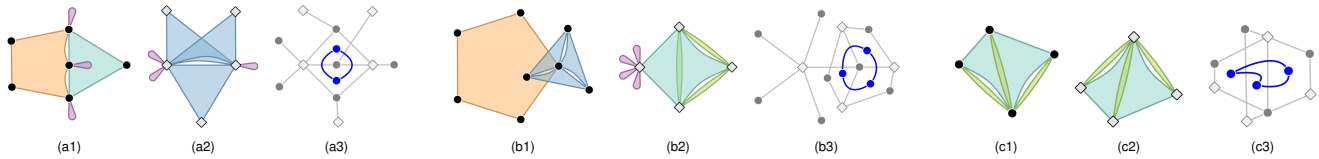


Fig. 5: The forbidden sub-hypergraphs of polygon hypergraph drawings: (a1) 3-adjacent hyperedge bundle of 2 hyperedges, (a2) 2-adjacent hyperedge bundle of 3 hyperedges, (b1) strangled vertex cycle variant, (b2) strangled hyperedge cycle variant, and (c1, c2) strangled vertex and hyperedge star variant. Notice that (a2) is the dual of (a1), (b2) is the dual of (b1), and (c2) is the dual of (c1). The cycle adjacency graph for each primal-dual pair is drawn in blue over the corresponding bipartite graph representation in (a3), (b3), and (c3).

blocks and the blue bubbles indicate the remaining tree subgraphs. Each K_i can be connected to one or more topological blocks. For each block T_j connected to K_i , we call the node $x_j \in T_j$ a *root* of K_i . Note that K_i cannot have more than one root in a single topological block, otherwise, it would define a new cycle. Similarly, a pair of topological blocks cannot be connected by more than one K_i . If a tree K_i is connected to only one topological block, having only one root node, we refer to it as a *branch*. Figure 4 (b,c) show that tree structures can be rooted at either primal or dual bipartite nodes. On the other hand, if K_i is connected to two or more topological blocks, thus having two or more roots, we refer to it as a *bridge*. Figure 4 (b,c) shows a bridge highlighted with an orange bubble that is connected to three topological blocks, having one primal node root and two dual node roots. This example also shows that a bridge, like a branch, can contain leaf nodes in G . We use the roots of a tree K_i to distinguish between bridges and branches instead of using leaf nodes, although we note that a branch necessarily contains one or more leaf nodes while a bridge can have no leaf nodes. Thus, our bipartite graph is an edge-disjoint union of basis cycles in \mathcal{C} which form the topological blocks, the set of all branches, and the set of all bridges (Figure 4 (b,c)). We refer to this as the *topological decomposition* D of the bipartite graph G and corresponding hypergraph H .

4.2 Decomposition Algorithm

To find the topological decomposition D of the bipartite graph $G(H)$, we first compute a more granular decomposition D' based on the blocks of G . Recall that a *block* in G is a maximal connected subgraph that has no articulation nodes. Following the classic algorithm of Hopcroft and Tarjan [13], we use a depth first search to extract all the blocks of G . The result is an edge-disjoint decomposition D' of G into a set of blocks. Within D' , we have two types of blocks: those consisting of a single edge in G , and those consisting of multiple edges and nodes in G . These are shown using blue and purple bubbles respectively in Figure 4 (a). Notice that the multi-edge blocks are exactly the topological blocks from our cycle-based decomposition D . That is, each multi-edge block consists of a subset of connected cycles in a cycle basis \mathcal{C} of G . Furthermore, a single-edge block cannot belong to any cycle in G since both of its endpoints are articulation nodes in G . This means that every single-edge block in D' is contained within some bridge or branch structure in D . We construct the bridges and branches of D by finding connected components of the single-edge blocks in D' .

In practice, computing the decomposition D' and using it to extract the topological blocks, bridges, and branches of D can be performed within a single depth-first search. We augment the algorithm of Hopcroft and Tarjan [13] to keep track of the multi-edge blocks in D' as well as the connected components of single-edge blocks in D' . For an arbitrary hypergraph H and its bipartite representation G , our algorithm produces a topological decomposition in $O(|V(G)| + |E(G)|)$ time.

As an additional step, we also extract a cycle basis \mathcal{C} associated with the topological decomposition D . Instead of searching all of G for cycles, we compute a cycle basis for each topological block $T \in D$ individually. Since G is unweighted, we use a pair of nested breadth-first searches to find a basis of linearly independent tight cycles in T . Our algorithm is inspired by Gashler and Martinez [12] who use a similar algorithm to find topological holes in manifold learning datasets. The first breadth-first search builds up a subgraph S of traversed edges in T .

When the first search discovers a back edge $(x, y) \in E(T)$ connecting to a previously visited node y , a new breadth-first search on S is started at x . Once this second breadth-first search reaches y , the search path from x to y along with the edge (x, y) is saved as a new basis cycle. Implementing the second search as a breadth-first search ensures that we find the shortest path in T from x to y apart from the edge (x, y) . This means that every extracted cycle is a tight cycle. We provide pseudocode for this algorithm in the supplementary material Appendix A.

5 PLANARITY

In a topological decomposition of hypergraph H , the set of bridges and branches are planar both in the bipartite representation G and in the polygon representation of H . This is because the bridges and branches all have a tree structure. Consequently, any non-planarity in G and any unavoidable overlaps in H must occur within the topological blocks. Note that unavoidable overlaps are a major source of visual clutter in hypergraph visualizations. However, a topological block T is not guaranteed to contain unavoidable overlaps. A set of connected cycles can have a planar representation if they are not entangled.

Consider one such topological block T , which is connected by definition and contains a subset of the basis cycles in \mathcal{C} . Thus, the Betti numbers $B_0(T) = 1$ and $B_1(T) > 0$. Let $\chi(T) = |V(T)| - |E(T)|$ denote the Euler characteristic of T . We also know that $\chi(T) = B_0(T) - B_1(T)$. Since $B_0(T) = 1$, we have $B_1(T) = 1 + |E(T)| - |V(T)|$. That is, the number of independent cycles in a topological block is directly related to the difference in the number of edges and the number of vertices in the block. The more cycles in the block, the more likely that T is entangled. We thus define the *entanglement index* of T as $\eta(T) = \frac{B_1(T)}{|V(T)|}$. Subgraphs that are trees, such as our bridge and branch structures, do not contain any cycles and have an entanglement index of zero.

The entanglement index can be used to determine which topological blocks likely contain unavoidable overlaps. However, within an entangled topological block, the unavoidable overlaps may only occur among a small subset of the cycles in the block. In this section, we develop theory on the configurations that cause unavoidable overlaps using the language of our topological decomposition.

5.1 Forbidden Sub-Hypergraphs

A hypergraph H is called *Zykov planar* if its bipartite representation $G(H)$ is a planar graph [39]. A graph is planar, i.e. an edge crossing-free plane embedding can be found, if and only if it does not contain a subdivision of the complete graph K_5 or complete bipartite graph $K_{3,3}$ [20]. Zykov's definition for planarity assumes that hyperedges are drawn as arbitrary closed regions. However, Oliver *et al.* [25] find this definition to be insufficient when drawing hyperedges as convex polygons, leading to a new definition for planarity within the polygon visualization metaphor:

Definition 1 (Oliver *et al.* [25]). *A convex polygon representation is a drawing of a hypergraph in the plane where each hyperedge is represented as a strictly convex polygon such that the area of intersection between each pair of polygons is zero.*

A hypergraph that admits a convex polygon representation is called *convex polygon planar*. Oliver *et al.* [25] identify four *forbidden sub-hypergraphs* that are Zykov planar but lack a convex polygon

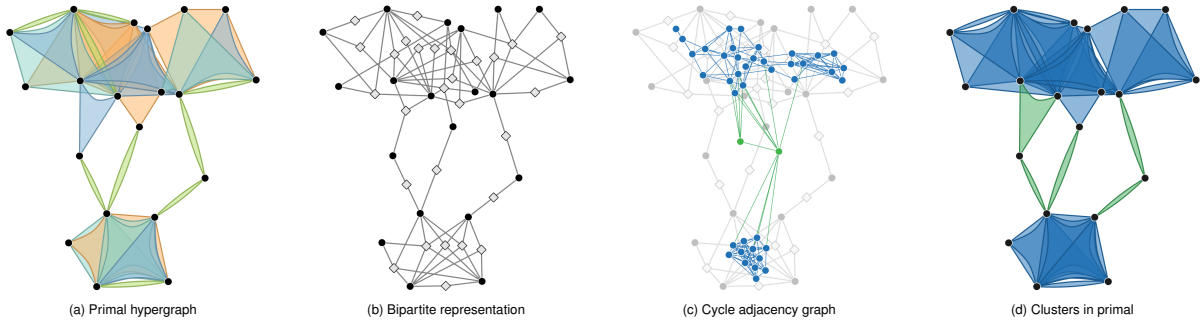


Fig. 6: An example of a topological block (a) containing multiple forbidden clusters. In (c), the cycle adjacency graph is superimposed over the bipartite representation (b), with the minimal basis cycles drawn as blue nodes, and the long basis cycles drawn as green nodes. Removing the green cycles leaves two 2-connected components of blue nodes, corresponding to the forbidden clusters highlighted in blue in (d).

representation. They define an n -adjacent cluster as the partial hypergraph induced by hyperedges $J \subseteq E$ containing a common set of vertices $X \subseteq V$ where $|X| = n \geq 2$. That is, set X is contained within each hyperedge $e \in J$. To avoid confusion with other cluster definitions, we refer to this as an n -adjacent bundle of hyperedges. The first forbidden sub-hypergraphs are a 3-adjacent bundle of 2 hyperedges (Figure 5 (a1)) and a 2-adjacent bundle of 3 hyperedges (Figure 5 (a2)). These two cases create a primal-dual pair. The next two forbidden sub-hypergraphs involve a *strangled vertex* or *strangled hyperedge* whose set of incident and adjacent elements have a particular structure. Oliver *et al.* [25] describe a variant where a proper subset of the incident and adjacent elements form a cycle of length $n \geq 3$ (Figure 5 (b1,b2)). We introduce another variant where a subset of the incident and adjacent elements form a star structure around a central element with $n \geq 3$ points (Figure 5 (c1,c2)). The strangled vertex and strangled hyperedge cases also create primal-dual pairs.

Theorem 2 (Oliver *et al.* [25]). *Let H be a Zykov planar hypergraph. Then H has a convex polygon representation if and only if it does not contain any of the following as a sub-hypergraph:*

- (a) A 3-adjacent bundle of 2 hyperedges,
- (b) A 2-adjacent bundle of 3 hyperedges,
- (c) A strangled vertex,
- (d) A strangled hyperedge.

In a polygon drawing of T , we observe that unavoidable overlaps have two causes: the forbidden sub-hypergraphs, which occur inside a local neighborhood of mutually incident and adjacent elements, and subdivisions of the Kuratowski subgraphs K_5 and $K_{3,3}$ in the bipartite representation which can occur among multiple long cycles. The K_5 and $K_{3,3}$ subdivisions can also occur inside a small radius of elements, but such instances often contain a forbidden sub-hypergraph within them. The forbidden sub-hypergraphs, on the other hand, are local structures by definition. We use *local entanglement* and *global entanglement* to describe these two categories of unavoidable overlap.

5.2 Forbidden Clusters

Let \mathcal{C} be a cycle basis for the topological block T containing only tight cycles, and let G_T be the bipartite representation of T . We define the *cycle adjacency graph* $A(\mathcal{C})$ as a graph containing vertices for each basis cycle $C \in \mathcal{C}$ and edges (C_i, C_j) for every bipartite edge in G_T which the cycles C_i and C_j have in common. Note that $A(\mathcal{C})$ is a non-simple graph since a pair of basis cycles can share multiple bipartite edges. The cycle adjacency graphs for each forbidden sub-hypergraph are shown in Figure 5, and an example for a larger topological block is shown in Figure 6. Oliver *et al.* [25] observe that the presence of forbidden sub-hypergraphs is correlated with the size of the intersection between pairs of adjacent vertices and pairs of adjacent hyperedges. We notice that every pair of shared vertices between hyperedges $e, f \in E$ corresponds to a minimal cycle. This motivated us to study the subgraph of the cycle adjacency graph induced by the set of minimal basis cycles which we denote as $A(\mathcal{C}_4)$.

Theorem 3. *A cycle in $A(\mathcal{C}_4)$ defined by the sequence $F = \langle C_1, C_2, \dots, C_k \rangle \subseteq \mathcal{C}_4$ for some tight cycle basis \mathcal{C} of T contains a common primal or dual node $x \in V(G_T)$ within each of the basis cycles $C_i \in F$ iff F corresponds to a forbidden sub-hypergraph in H .*

We prove Theorem 3 in our supplementary material Appendix B. Theorem 3 implies that forbidden sub-hypergraphs appear as cycles within $A(\mathcal{C}_4)$. We define a *forbidden cluster* inside a topological block T as a 2-connected component of the elements in $A(\mathcal{C}_4)$. This is a stronger notion than our definition of a topological block which includes any 1-connected component of elements in $A(\mathcal{C})$. Figure 6 shows multiple forbidden clusters extracted from a single topological block. Under this definition, any forbidden sub-hypergraphs in T must occur within a forbidden cluster. To extract the forbidden clusters of T , we re-use Hopcroft and Tarjan’s algorithm to detect blocks of elements in $A(\mathcal{C}_4)$. Each of the non-trivial blocks extracted by this algorithm constitutes a new forbidden cluster in T .

6 STRUCTURE-AWARE SIMPLIFICATION

To achieve a meaningful multi-scale representation of H , we present a simplification method using operations specifically designed for the individual structures in our topological decomposition. We use three atomic simplifications that operate on the bipartite representation G : a minimal cycle collapse operation, a cycle edge cut operation, and a leaf pruning operation. The goal of our simplification is first, to eliminate non-planarity caused by local and global entanglement, and second, to reduce the space required by each structure in a polygon layout to facilitate high quality visualizations. A major difference in Oliver *et al.*’s [25] approach is that they initialize candidate operations for every vertex and hyperedge in H , while we use the topological decomposition D to generate candidate operations only where they are needed.

Another difference is the explicit use of both topology preserving and topology altering simplification operations. Oliver *et al.* use element removal and merger operations, both of which have the potential to maintain or alter the topology of H depending on the configuration of the operand elements. For example, a series of merger operations could be used to collapse a cycle into a single vertex or hyperedge, reducing the first Betti number B_1 by one. On the other hand, a series of merger operations on a bridge or branch structure would not affect B_1 or the number of connected components B_0 . In this paper, we use operations that are either always topology preserving, or always topology altering, allowing us to simplify H in a more controlled way.

The topological decomposition D of H is an edge-disjoint union of topological blocks T_i , bridges K_j , and branches K_k . The topological blocks are defined by a tight cycle basis $\mathcal{C} = \{C_1, \dots, C_p\}$. While each of the structures in D can be simplified in parallel, our implementation simplifies each structure sequentially in order of decreasing entanglement index. As a result, the bridges and branches, which contain no entanglement are simplified last. Our reasoning is based on experimental observations that non-planarity, which only occurs within the topological blocks, is the primary source of visual clutter in polygon

visualizations. Of course, the sources of visual clutter can be data-dependent, and our simplification framework allows for any measure to be used in place of the entanglement index.

6.1 Topological Blocks

A topological block T can contain both local entanglement in the form of forbidden sub-hypergraphs, and global entanglement in the form of Kuratowski subgraphs. We design a pair of topology altering operations to specifically target and reduce both types of entanglement, thereby reducing the amount of unavoidable overlaps in T .

The first operation we call a *minimal cycle collapse*. Let $C = \langle u, e, v, f \rangle$ be a minimal cycle in T where $u, v \in V(G)$ are primal nodes and $e, f \in E(G)$ are dual nodes. Collapsing C into a single node has the potential to make G non-bipartite and the hypergraph H invalid. Instead, we collapse C by merging together either the primal nodes u, v or the dual nodes e, f . Figure 7 (a) shows an example of a minimal cycle collapse using primal nodes. This operation alters the topology of H by removing one or more minimal cycles depending on the direction of merging, reducing the value of B_1 accordingly. If C is collapsed by merging primal nodes v, u , any of the other minimal cycle C_i containing u and v are also collapsed and are removed from the cycle basis \mathcal{C} . A longer cycle C_j containing u and v , where $|C_j| > 4$, is not at risk of being collapsed, but its length is reduced by 2.

The purpose of the minimal cycle collapse operation is to eliminate forbidden sub-hypergraphs. To identify candidate collapse operations, we first extract the set of forbidden clusters from T . If T does not contain any forbidden clusters, it cannot contain any forbidden sub-hypergraphs, and we do not need any cycle collapse operations. Otherwise, for each forbidden cluster in T , we search for tight cycles within $A(\mathcal{C}_4)$, reusing our tight cycle basis algorithm from Section 4.2. Let $F = \langle C_1, \dots, C_n \rangle$ be a tight cycle in $A(\mathcal{C}_4)$. If the minimal cycles $C_i \in F$ share any common bipartite nodes, we save F as a forbidden sub-hypergraph and classify it according to the number and types of the shared bipartite nodes, as in the proof of Theorem 3. If F corresponds to a forbidden sub-hypergraph, we identify a candidate cycle collapse operation for every $C_i \in F$.

The second operation we call a *cycle edge cut*. Given a basis cycle C_i , we cut one of the edges $(v, e) \in E(C_i)$ in the bipartite graph, breaking the cycle C_i , and removing the connection between the corresponding vertex and hyperedge in H , as shown in Figure 7 (b). This operation also alters the topology of H by breaking one of the basis cycles, reducing the value of B_1 accordingly. Any basis cycles $C_j \neq C_i$ containing the edge (v, e) are also affected and must be updated by replacing (v, e) with the new shortest path between v and e .

The purpose of the cycle edge cut operation is to eliminate instances of K_5 and $K_{3,3}$ occurring between non-minimal basis cycles. Thus, to identify edge cut candidates, we first construct a modified copy T' by replacing each forbidden cluster in T with a single node, retaining the external connections of the cluster. We note that T' may no longer be a bipartite graph since the forbidden clusters contain both primal and dual nodes. For this reason, we only use B' to search for instances of K_5 and $K_{3,3}$ and not as a representation of the original hypergraph. We then build an embedding of T' with minimized edge crossings using a subgraph planarization algorithm from the Open Graph Drawing Framework [7]. We repeat the edge insertion phase of the planarization algorithm with ten permutations of the edge insertion order, taking only the best result with the fewest edge crossings. For each crossing between edges $(v_i, e_i), (v_j, e_j) \in E(T')$, we identify a candidate cycle edge cut operation for both (v_i, e_i) and (v_j, e_j) .

Once all of the candidate minimal cycle collapse and cycle edge cut operations for topological block T have been identified, we apply the priority ranking system of Oliver *et al.* [25] to guide the order of application for each operation. Their system uses a priority measure containing multiple terms for per-element statistics including degree and cardinality, and betweenness centrality, each controlled by tuning parameters α, β and γ . We add to this an additional term and tuning parameter δ to evaluate how much each operation alters the topology of the input hypergraph H .

Let $O(x_1, x_2)$ be a candidate minimal cycle collapse or cycle edge cut operation. In the case that O is a cycle collapse operation, let x_1 and x_2 represent the pair of primal or dual nodes to be merged in the collapse of minimal basis cycle C . In the case that O is a cycle cut operation, let x_1 be the primal type node and x_2 the dual type node of the edge $(x_1, x_2) \in T$ to be cut. Let s be the potential change in the first Betti number B_0 caused by the application of O . If O is a minimal cycle collapse, s is the number of minimal basis cycles in \mathcal{C} containing x_1 and x_2 . If O is a cycle edge cut, s is the total number of basis cycles in \mathcal{C} containing the edge (x_1, x_2) . Now let l be the average length of all basis cycles containing x_1 and x_2 . We use $\delta \left(\frac{1}{s} + \frac{1}{l} \right)$ as the additional term in our operation priority measure. This term gives higher priority to operations that eliminate the fewest cycles, and lower priority to operations that affect long cycles. Our reasoning for prioritizing operations that eliminate the fewest cycles is straightforward, we aim to alter the topology of H as little as possible. We prioritize preserving long basis cycles over short basis cycles for two reasons. Firstly, the long basis cycles can be considered more topologically significant while shorter basis cycles can be considered topological noise. Secondly, the long basis cycles are less likely to constrict the layout in a polygon drawing of H , and tend to pose fewer challenges for layout optimization algorithms.

After an operation O has been applied, we update any remaining candidate operations that may have been affected. If O is a cycle edge cut operation, we find the other edge in the same crossing identified by the planarization algorithm and remove it from consideration since the crossing has been resolved. If O is a cycle collapse operation on a minimal cycle C_i belonging to some forbidden sub-hypergraph F_i , we remove from consideration any collapse operations on the other minimal cycles $C_j \in F$, unless C_j also belongs to another forbidden sub-hypergraph F_j that has not yet been simplified. The simplification of T terminates whenever the entanglement index $\eta(T)$ drops below a user defined threshold, or when T is free from both forbidden sub-hypergraphs and subdivisions of the Kuratowski subgraphs, i.e., when T has a convex planar representation.

6.2 Bridges and Branches

Let K be a bridge or branch structure with $r \geq 1$ roots. Since K does not contain any entanglement, our primary goal is to facilitate high-quality polygon layouts by reducing the amount of space required by the individual hypergraph elements in K . We do this using a *leaf node pruning* operation. This is similar to the cycle edge cut operation in that it breaks the incidence relationship between a vertex $v_i \in V(K)$ and a hyperedge $e_i \in E(K)$. However, the leaf pruning operation requires that either v_i or e_i correspond to a leaf node in G , and deletes the leaf node after cutting the bipartite edge (v_i, e_i) . Furthermore, the leaf pruning operation is topology preserving while the edge cut operation is topology altering. A leaf node cannot belong to a cycle, so pruning does not affect B_1 . Since the leaf node is deleted after the edge is cut, pruning also has no impact on the number of connected components B_0 . An example of pruning a dual leaf node from a bridge is shown by Figure 7 (c), and an example of pruning a primal leaf node from a branch structure is shown by Figure 7 (d).

We identify leaf nodes in K using a multi-source breadth-first search starting from the root nodes of K . For each leaf node discovered, we identify a new pruning operation candidate. Within the search, we also keep track of the depth $d(x)$ of each node x with respect to the roots of K as well as the furthest depth $d_{low}(x)$ reached by any of the children of x . A root node r of K has $d(r) = 0$ and $d_{low}(r)$ equal to the total height of the tree while a leaf node x has $d_{low}(x) = d(x)$.

Once the candidate pruning operations have been identified, we again use the operation priority ranking system of Oliver *et al.* to determine the order of simplifications. For bridge and branch structures we add the term $\delta \left(1 - \frac{d_{low}(x)}{d_{low}(r)} \right)$ to the operation priority measure where x is the leaf node in a candidate operation and r is a root node of K . This term promotes preserving the deepest elements in K as well as the elements on the connecting paths between r and the deepest elements. We use this to help preserve the longest paths in each bridge and branch which

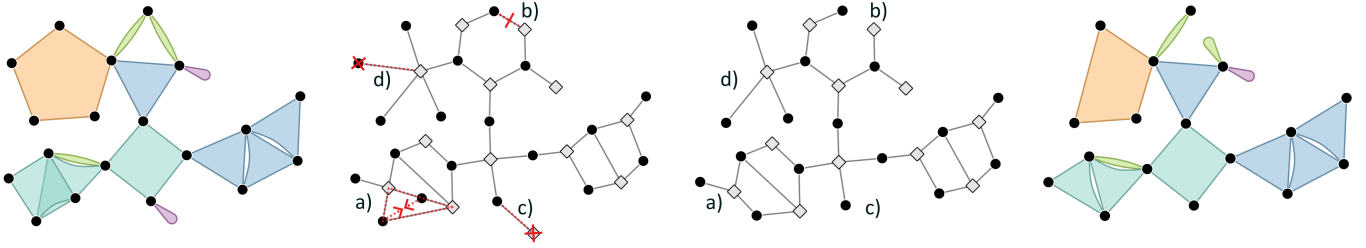


Fig. 7: Example simplification operations in the hypergraph from Figures 2 and 4. On the far left and far right, we have the primal hypergraph before and after simplification. In the middle figures, we have four labeled operations (a), (b), (c), and (d). Operation (a) is a minimal cycle collapse, operation (b) is a cycle edge cut, operation (c) is a leaf pruning on a bridge, and operation (d) is a leaf pruning on a branch.

Table 1: Comparison of hypergraph simplification methods for reducing forbidden sub-hypergraphs in paper-author datasets. Each dataset is collected from the DBLP database [22] and consists of a maximal connected subset of publications in the specified year range from the IEEE journals Transactions on Pattern Analysis and Machine Intelligence (TPAMI) and Transactions on Visualization and Computer Graphics (TVCG). Each simplification was run until the input no longer contained forbidden sub-hypergraphs. Note that our method is more efficient and uses fewer operations to achieve this.

description	Dataset				Oliver et al. [25]			Ours		
	$ V $	$ E $	B_1	$\eta(H)$	num ops.	initialization (s)	simplification (s)	num ops.	decomp. (s)	simplification (s)
TVCG (2015-2017)	1008	429	334	0.234	275	0.305	0.556	83	0.012	0.006
TPAMI (2013-2020)	2054	947	805	0.268	581	1.298	3.902	180	0.045	0.025
TVCG (2013-2020)	3460	1570	1898	0.346	1452	4.190	845.0	429	0.101	0.094

allows them to be easily identified in visualizations of simplified scales.

After pruning a leaf node x , we check whether the parent of y of x has become a leaf node. If so, we identify a new candidate pruning operation for y and add it to the priority list of candidate operations. The simplification terminates once the priority of the next pruning operation drops below a user-defined threshold.

7 RESULTS

To evaluate the performance of our structure-guided simplification, we compare it to the priority guided method of Oliver *et al.* [25]. They take a statistical approach where iterative atomic simplifications are guided by a priority measure containing terms for the distributions of different per-element statistics. In particular, they use terms for the distribution of vertex degrees and hyperedge cardinalities, vertex and hyperedge adjacency factors, and betweenness centrality. The influence of each distribution on the order of simplifications is controlled by tuning parameters α, β, γ . The purpose of their adjacency factor term, which measures the volume of shared vertices between a hyperedge and all of its adjacent hyperedges, is to guide the simplification toward eliminating forbidden sub-hypergraphs.

We compare the efficiency of both approaches in Table 1 for the task of removing forbidden sub-hypergraphs from three different hypergraph datasets. For each method, we list the number of operations required to remove all forbidden sub-hypergraphs from the input, the time required to initialize the operations, or in our case, perform the decomposition, and the time required to apply the operations and perform any necessary updates. We observe that both approaches increase in the number of operations and time required as the datasets increase in size as well as entanglement index $\eta(H)$. However, our approach requires significantly fewer operations and significantly less execution time. The smaller number of operations can be attributed to our decomposition and extraction of forbidden clusters which tell us exactly where the forbidden sub-hypergraphs live. The difference in execution time, especially for the largest dataset, is even greater. This can be due to the fact that we simplify each decomposition structure independently, maintaining separate operation priority queues for the operations on each structure, while Oliver *et al.* maintain a much larger global priority queue that needs to be updated and potentially re-sorted after every atomic simplification. Finally, we observe that their implementation is iterative in nature relying on an updating scheme that does not have an obvious parallel implementation. Ours has the potential to be sped up even further by simplifying the decomposition structures in parallel.

To evaluate the utility of our decomposition and simplification framework, we apply it to three real-world datasets. For a simplified result H' , we generate high quality polygon layouts by first applying a force directed layout to the simplified bipartite graph G' . We align the vertices of H' with the corresponding primal nodes in the layout of G' , and draw the hyperedges as polygons between their contained vertices using the starrization technique of Qu *et al.* [27]. We then apply the automatic primal-dual polygon layout optimization system of Qu *et al.* to H' . Their optimization uses a multi-term objective function that promotes polygon regularity and uniform side lengths while avoiding unnecessary polygon overlaps.

Friendship Dataset. This dataset involves friendships between high school students in Marseilles, France, recorded in 2013 [23]. The original dataset is a directed network of reported friendships among the students. We construct an undirected graph from this dataset including only edges where both students reported being friends with each other. We then construct a hypergraph H by creating a hyperedge for every maximal clique in this graph. A connected subset of H is visualized in Figure 1 (a). Each hyperedge represents a friendship group where every student reported being friends with every other student in the group.

The topology can tell something about the different communities in H . For example, in relation to the zeroth Betti number B_0 , the connected components of H suggest larger communities among the students, possibly representing the individual classes from the dataset described in [23]. In relation to the first Betti number B_1 , the cycles within each topological block can tell us how *close-knit* a particular community is. For instance, a community consisting of many small cycles is more close-knit than a community with only long cycles, which is in turn more close-knit than a community that has a tree structure. Thus, being able to identify the existence and length of cycles in H , related to the entanglement index of its topological blocks, can be a valuable tool for studying the different communities.

Figure 1 (a) highlights in blue a length 4 cycle of H that occurs between two forbidden clusters. In (b), a simplified scale H' is produced using the method of Oliver *et al.* [25] with tuning parameters $\alpha = 0.0, \beta = 0.9, \gamma = 0.4$. Notice that the vertices in the highlighted cycle have been collapsed into a single vertex, combining the forbidden clusters on either side. This gives the impression of a single close-knit group of students in H' where there is actually some separation between the clusters in H . In (c), the highlighted cycle is preserved after applying our structure-aware simplification method, and the original forbidden

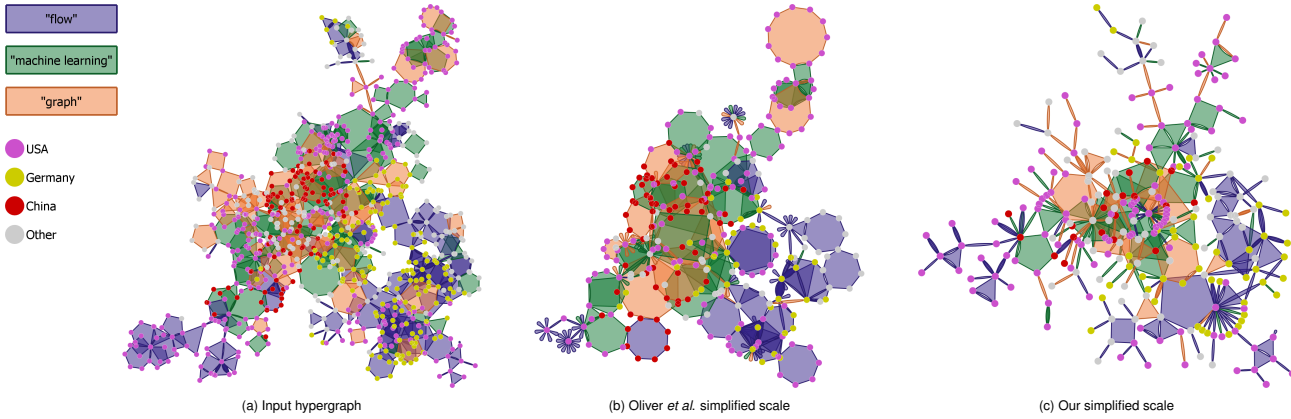


Fig. 8: A paper-author hypergraph network with 786 vertices and 318 hyperedges, (a) is simplified using the priority guided approach of Oliver *et al.* [25] (b), and our topological decomposition guided approach (c). Notice that some of the branches in (b) have been reduced and others have been eliminated entirely. Our hypergraph decomposition extracts structures ahead of time, allowing us to preserve the skeleton of each branch. In each visualization, the hyperedges are colored according to the keywords of their papers: blue for the keyword “flow”, green for the keywords “machine learning”, and orange for the keyword “graph”. Additionally, the vertices are colored according to the geographic location of the affiliated research institution for each author: magenta for institutions based in the United States, yellow for institutions based in Germany, red for institutions based in China, and gray for institutions based elsewhere. Enlarged versions of these visualizations are provided in our supplementary material [Appendix D](#).

clusters from H remain visually distinct. Our simplification H'' was produced using the same number of simplification operations as H' , the same values for α, β, γ , and a value of $\delta = 1.0$ for our new priority terms. We also observe that the bottom half of H' is nearly identical to H , and still contains a few forbidden sub-hypergraphs. Our result H'' on the other hand does not contain any forbidden sub-hypergraphs. This indicates that our decomposition-based approach is a more targeted and efficient way to reach a planar result.

The result in (c) was reached after applying 25 minimal cycle collapse operations and 1 cycle edge cut operation. This indicates that the original hypergraph contains more local entanglement in the form of forbidden clusters than global entanglement in the form of K_5 or $K_{3,3}$ subdivisions. These forbidden clusters can indicate broader friendship circles than the individual hyperedges would imply. While it would be possible to cut open these friendship circles to reach planarity, it seems more appropriate to eliminate the forbidden sub-hypergraphs by merging some of their elements. This way, some distinction between individual elements is lost, but the connections within the friendship circle are emphasized. For example, the overlapping red and orange polygons on the left side of (a) are merged into a single octagon in (c), filling in the few missing connections among the friendship circle. On the other hand, two vertices being merged together can be interpreted as a pair of students who have very similar friendship connections. This led to the appearance of *monogons* (purple teardrop shapes) in (c) which indicate tighter friendship groups within a merged “super-vertex”.

The cycle cut operation that is applied in (c) removes one friend from a friendship group. Our simplification system is designed such that the cut friendship occurs along a non-minimal cycle, and does not cut open a close-knit forbidden cluster. Furthermore, we prioritize cutting bipartite edges that occur along the fewest and shortest basis cycles. Thus, while the immediate connection is broken, the increase in the shortest path between the removed friend and their friendship group (along the other side of the broken cycle) is minimized. In this way, our simplification framework balances maintaining both path length and path structure by using both collapse and cut operations, but the acceptability of artifacts from one or the other may depend on the specific dataset and required analysis tasks.

Paper-Author Dataset. Isenberg et al.’s Vispubdata dataset [16] contains publication information for IEEE Visualization papers from 1990-2021. Figure 8 (a) shows a connected subset of publications containing the keywords “flow”, “graph”, and “machine learning”, where each author is a vertex and each paper a hyperedge. The vertices are colored according to the geographic location of each author’s affiliated research

institution: magenta for institutions based in the United States, yellow for institutions based in Germany, red for institutions based in China, and gray for institutions based elsewhere. The hyperedges are colored according to the keywords of the corresponding papers: blue for the keyword “flow”, green for the keywords “machine learning”, and orange for the keyword “graph”. For papers containing multiple of these keywords, we choose the color based on which keyword appears first. This dataset contains numerous bridge and branch structures as well as several large cycles. Entangled topological blocks can be interpreted as topic areas with a high level of cross-collaboration within or between research groups. Based on the density of overlapping polygons in Figure 8 (a), there appears to be much cross-collaboration between researchers based in Germany on the topic of flow visualization and cross-collaboration between researchers based in China on machine learning and graph visualization topics. However, it is difficult to tell whether there are actually cycles present among the overlapping hyperedges or if this is just a result of the layout. Our simplified result in (c) was produced using only minimal cycle collapse and leaf pruning operations. In this layout, all of the forbidden sub-hypergraphs are eliminated, but we still see a number of cycles and overlapping hyperedges among the Germany and China based researchers. This confirms our observations of cross-collaboration among these research communities. The result in (b) was produced by running the simplification algorithm of [25] with the same tuning parameters until reaching a scale with the same number of elements as (c). Their method does not guarantee that large cycles are preserved, so we cannot reach the same conclusion based on their visualization result.

We also see in Figure 8 (a) that many of the US based researchers appear to be connected to the rest of the hypergraph through long bridge or branch structures. These structures can represent new research directions or new connections between existing topic areas. The large bridge and branch structures are even more apparent in (c), but we also discover many short branches that are hidden among the overlapping hyperedges in (a) and (b). This indicates that in addition to the cross-collaboration seen in the topological blocks, there are many papers connected to the rest of the community through a single author, who may be an advisor with numerous students, or an influential publication.

In Figure 8 (a), we can see two rabbit ear structures near the top extending from the central part of the hypergraph. In (c), we find that the rabbit ear structures include small topological blocks near the ends. In the context of our topological decomposition, the rabbit ears are the union of a bridge, topological block, and several small branches. The small topological blocks in each ear are important since they suggest a research group with frequent collaborations rather than an individual

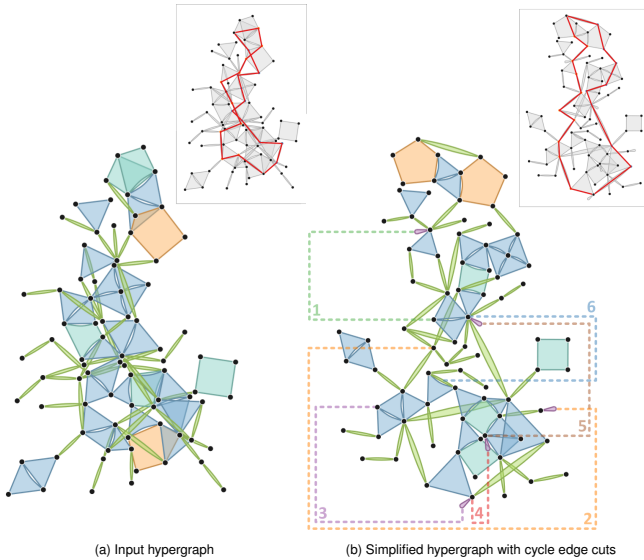


Fig. 9: A hypergraph representing face-to-face interactions between museum visitors [15] is visualized before, (a), and after simplifying the topological blocks, (b). Notice the numerous triangles and digons in (a) that overlap despite being non-adjacent. On the right, six cycles have been cut to make the hypergraph planar. The deleted incidence relationships between vertices and hyperedges are represented by dashed annotation lines along the boundary of the visualization. In the upper right corners, we highlight the same two paths for each layout from a vertex near the top to a vertex near the bottom.

researcher with many one-time collaborators. In fact, the topological block in the left ear is formed by a group of three researchers, each pair of whom collaborated on a different topic. In (b), the left ear is reduced to a short branch with a high degree vertex at the end, giving the opposite impression of a sole prolific researcher. Furthermore, because the simplification method of Oliver *et al.* [25] is iterative, the cycle in the left ear was collapsed in a previous simplified scale, changing the structure of the ear from a union of a bridge, cycle, and branches into a single branch. Thus, the simplified result in (b) involves both a false negative case in the collapsed cycle, and a false positive in the new branch structure replacing the original rabbit ear from (a).

The interpretation of minimal cycle collapse operations in this dataset is similar to the previous friendship dataset: papers or authors from the same close-knit research group are merged together to eliminate visual clutter inside forbidden clusters. The leaf nodes pruned in (c) could be considered less significant in terms of connectivity in the research community because they represent authors with only one paper in the field, possibly student research assistants, or papers with only one author. Iterative leaf pruning has the potential to eliminate a branch structure entirely, removing important connecting authors, which is why we prioritize preserving the deepest elements in each branch. However, for certain visualization tasks, such as comparing the number of authors on each paper or the number of publications for each author, leaf pruning may not be appropriate.

Museum Interactions Dataset. Our final dataset from Isella *et al.* [15] tracks social contact patterns between visitors in a museum gallery. Face-to-face interactions between gallery visitors were measured using electronic badges. Qu *et al.* [27] construct hypergraphs from this data by creating hyperedges between maximal cliques of participants who spent more than 40 seconds interacting face-to-face with each other. Figure 9 (a) shows the hypergraph H for gallery visitors on May 5th, 2009. In (b), we show a planar simplified scale obtained using our topology altering simplification operations. We include dashed annotation lines to indicate the incidence relationships from the original hypergraph that are removed in the simplified scale. Such annotations in simplified results can provide a visual indication of the

global entanglement present in the original hypergraph. We provide a comparison of our simplifications here to Zhou *et al.* [38] in our supplementary material Appendix C.

We observe that reaching a planar simplified scale H' in this case requires 6 edge cycle cut operations and 2 minimal cycle collapse operations, meaning H contains more subdivisions of the Kuratowski graphs than forbidden sub-hypergraphs. This indicates that although there are many overlapping polygons in the layout of H , the average adjacency between pairs of elements is relatively low. From Figure 9 (a) we can already see that a majority of the face-to-face interactions only involve two or three people. The visualization in (b) further shows that there are not many common participants between separate interactions. This makes sense in a gallery setting where small groups of visitors who know each other have only a few interactions outside that group where they may be listening to a tour guide or speaking with a docent.

In the context of tracking infectious diseases, the disentangled visualization in (b) gives a clearer view of the possible transmission vectors between visitors. The close proximity of several vertices in (a) combined with the overlapping polygons makes it difficult to tell how many distinct paths connect the uppermost elements in the layout to the bottommost elements. In the upper right corner of (a), two such distinct paths are highlighted in red, but there are several places in the layout where these paths appear to share a vertex. This could be improved by refining the layout, but the underlying issue remains that some amount of overlap is unavoidable since the hypergraph does not have a convex polygon representation. In (b), the vertices and hyperedges are more spread out, and it is easier to distinguish distinct paths between elements. In the upper right corner of (b), the same two distinct paths are highlighted in red, and we can clearly see that they do not intersect except at the uppermost and bottommost vertices.

8 CONCLUSIONS AND FUTURE WORK

In this paper, we present a novel structure-based hypergraph decomposition using the topology of the bipartite graph representation. Within our decomposition, the use of the bipartite graph representation also allows us to identify entangled minimal cycles as the main source of unavoidable overlaps in hypergraph visualizations. This leads to a new definition of the entanglement index which is based on the ratio of the first Betti number and the number of vertices in a topological block. We use our decomposition to augment the atomic simplification framework of Oliver *et al.* [25] making use of structure preserving and structure altering operations. We also provide efficient algorithms to compute the decomposition, as well as a framework for implementing structure-aware simplification. Compared with [25], our work makes the interpretation of simplified hypergraph visualizations more reliable because the main structures in the data are preserved. An implementation of our decomposition, simplification, and visualization tools are available on GitHub: <https://github.com/peterdanieloliver/HGPolyVis>.

The goal of our work is to provide the theoretical groundwork for structure-based hypergraph simplification upon which future works can build. A thorough evaluation is needed to fully explore the potential visualization benefits, such as identifying disjoint paths in simplified hypergraphs. We consider our evaluations preliminary and plan to conduct additional user studies as future work.

We will collaborate with domain scientists in future work to assess the usefulness of our approaches and explore possible improvements for application dependent visualization problems. We also plan to more rigorously investigate the use of glyphs and annotations for communicating artifacts produced by our simplifications. While our work identifies entangled cycles as the source of unavoidable overlaps in polygon visualizations, requiring the polygons to be near-regular can also cause overlaps in the bridge and branch structures. We plan to investigate the use of hyperbolic spaces to address this issue. We also plan to explore structure-based hierarchical layout methods to make polygon visualization of hypergraphs with thousands to tens of thousands of hyperedges more tractable. Finally, we plan to study extensions of our work to time-varying hypergraphs as well as AR/VR.

ACKNOWLEDGMENTS

The authors wish to thank the anonymous reviewers and paper chairs for their constructive feedback.

REFERENCES

- [1] S. G. Aksoy, C. Joslyn, C. O. Marrero, B. Praggastis, and E. Purvine. Hypernetwork science via high-order hypergraph walks. *EPJ Data Science*, 9(1):16, 2020. doi: 10.1140/epjds/s13688-020-00231-0 2
- [2] B. Alsallakh, W. Aigner, S. Miksch, and H. Hauser. Radial sets: Interactive visual analysis of large overlapping sets. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2496–2505, 2013. doi: 10.1109/TVCG.2013.184 2
- [3] B. Alsallakh, L. Micalef, W. Aigner, H. Hauser, S. Miksch, and P. Rodgers. The state-of-the-art of set visualization. *Comput. Graph. Forum*, 35(1):234–260, 27 pages, Feb. 2016. doi: 10.1111/cgf.12722 1
- [4] N. A. Arafat and S. Bressan. Hypergraph drawing by force-directed placement. In *Database and Expert Systems Applications*, pp. 387–394. Springer International Publishing, Cham, 2017. doi: 10.1007/978-3-319-64471-4_31 2
- [5] C. Berge. *Graphs and Hypergraphs*. North-Holland Publishing Company, Amsterdam, 1973. 2, 3
- [6] A. Bretto. Hypergraph theory. *An introduction. Mathematical Engineering. Cham: Springer*, 2013. doi: 10.1007/978-3-319-00080-0 3
- [7] M. Chimani, C. Gutwenger, M. Jünger, G. W. Klau, K. Klein, and P. Mutzel. The open graph drawing framework (OGDF). *Handbook of graph drawing and visualization*, 2011:543–569, 2013. doi: 10.17877/DE290R-7670 6
- [8] M. Dörk, N. Henry Riche, G. Ramos, and S. Dumais. PivotPaths: Strolling through faceted information spaces. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2709–2718, 2012. doi: 10.1109/TVCG.2012.252 2
- [9] P. Erdős and T. Gallai. On maximal paths and circuits of graphs. *Acta Mathematica Academiae Scientiarum Hungaricae*, 10(3-4):337–356, 1959. doi: 10.1007/BF02024498 2
- [10] T. Fan, L. Lü, D. Shi, and T. Zhou. Characterizing cycle structure in complex networks. *Communications Physics*, 4(1):272, Dec 2021. doi: 10.1038/s42005-021-00781-3 2
- [11] F. Frank, M. Kaufmann, S. Kobourov, T. Mchedlidze, S. Pupyrev, T. Ueckerdt, and A. Wolff. Using the metro-map metaphor for drawing hypergraphs. In T. Bureš, R. Dondi, J. Gamper, G. Guerrini, T. Jurdziński, C. Pahl, F. Sikora, and P. W. Wong, eds., *SOFSEM 2021: Theory and Practice of Computer Science*, pp. 361–372. Springer International Publishing, Cham, 2021. doi: 10.1007/978-3-030-67731-2_26 2
- [12] M. Gashler and T. Martinez. Robust manifold learning with CycleCut. *Connection Science*, 24(1):57–69, 2012. doi: 10.1080/09540091.2012.664122 4
- [13] J. Hopcroft and R. Tarjan. Algorithm 447: efficient algorithms for graph manipulation. *Commun. ACM*, 16(6):372–378, 7 pages, jun 1973. doi: 10.1145/362248.362272 4
- [14] J. D. Horton. A polynomial-time algorithm to find the shortest cycle basis of a graph. *SIAM Journal on Computing*, 16(2):358–366, 1987. doi: 10.1137/0216026 3
- [15] L. Isella, J. Stehlé, A. Barrat, C. Cattuto, J.-F. Pinton, and W. Van den Broeck. What’s in a crowd? Analysis of face-to-face behavioral networks. *Journal of Theoretical Biology*, 271(1):166–180, 2011. doi: 10.1016/j.jtbi.2010.11.033 9, 12, 13, 15
- [16] P. Isenberg, F. Heimerl, S. Koch, T. Isenberg, P. Xu, C. Stolper, M. Sedlmair, J. Chen, T. Möller, and J. Stasko. vispubdata.org: A metadata collection about IEEE visualization (VIS) publications. *IEEE Transactions on Visualization and Computer Graphics*, 23(9):2199–2206, Sept. 2017. doi: 10.1109/TVCG.2016.2615308 8
- [17] B. Jacobsen, M. Wallinger, S. Kobourov, and M. Nöllenburg. MetroSets: Visualizing sets as metro maps. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):1257–1267, 2021. doi: 10.1109/TVCG.2020.3030475 2
- [18] T. Kaczynski, K. M. Mischaikow, M. Mrozek, and K. Mischaikow. *Computational homology / Tomasz Kaczynski, Konstantin Mischaikow, Marian Mrozek*. Applied mathematical sciences (Springer-Verlag New York Inc.); v. 157. Springer, New York, 2004. doi: 10.1007/b97315 3
- [19] B. Kim, B. Lee, and J. Seo. Visualizing set concordance with permutation matrices and fan diagrams. *Interacting with Computers*, 19(5-6):630–643, 2007. doi: 10.1016/j.intcom.2007.05.004 2
- [20] C. Kuratowski. Sur le problème des courbes gauches en topologie. *Fundamenta Mathematicae*, 15(1):271–283, 1930. doi: 10.4064/fm-15-1-271-283 4
- [21] A. Lex, N. Gehlenborg, H. Strobel, R. Vuillemot, and H. Pfister. UpSet: Visualization of intersecting sets. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):1983–1992, 2014. doi: 10.1109/TVCG.2014.2346248 2
- [22] M. Ley. DBLP: some lessons learned. *Proc. VLDB Endow.*, 2(2):1493–1500, 8 pages, aug 2009. doi: 10.14778/1687553.1687577 7
- [23] R. Mastrandrea, J. Fournet, and A. Barrat. Contact patterns in a high school: A comparison between data collected using wearable sensors, contact diaries and friendship surveys. *PLOS ONE*, 10(9):1–26, 09 2015. doi: 10.1371/journal.pone.0136497 1, 7
- [24] L. Micalef and P. Rodgers. eulerAPE: Drawing area-proportional 3-Venn diagrams using ellipses. *PLoS One*, 9:e101717, 07 2014. doi: 10.1371/journal.pone.0101717 2
- [25] P. Oliver, E. Zhang, and Y. Zhang. Scalable hypergraph visualization. *IEEE Transactions on Visualization and Computer Graphics*, 30(1):595–605, 2024. doi: 10.1109/TVCG.2023.3326599 1, 2, 4, 5, 6, 7, 8, 9, 14
- [26] B. Qu, P. Kumar, E. Zhang, P. Jaiswal, L. Cooper, J. Elser, and Y. Zhang. Interactive design and visualization of n-ary relationships. In *SIGGRAPH Asia 2017 Symposium on Visualization*, p. 15. ACM, 2017. doi: 10.1145/3139295.3139314 1
- [27] B. Qu, E. Zhang, and Y. Zhang. Automatic polygon layout for primal-dual visualization of hypergraphs. *IEEE Transactions on Visualization and Computer Graphics*, 28(1):633–642, 2022. doi: 10.1109/TVCG.2021.3114759 1, 2, 7, 9
- [28] N. H. Riche and T. Dwyer. Untangling Euler diagrams. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1090–1099, 2010. doi: 10.1109/TVCG.2010.210 2
- [29] P. Rodgers, L. Zhang, and A. Fish. General Euler diagram generation. In *Proceedings of the 5th International Conference on Diagrammatic Representation and Inference*, Diagrams ’08, 15 pages, p. 13–27. Springer-Verlag, Berlin, Heidelberg, 2008. doi: 10.1007/978-3-540-87730-1_6 2
- [30] R. Sadana, T. Major, A. Dove, and J. Stasko. OnSet: A visualization technique for large-scale binary set data. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):1993–2002, 2014. doi: 10.1109/TVCG.2014.2346249 2
- [31] R. Santamaría and R. Therón. Visualization of intersecting groups based on hypergraphs. *IEICE Transactions on Information and Systems*, 93(7):1957–1964, Jan. 2010. doi: 10.1587/transinf.E93.D.1957 2
- [32] P. Simonetto, D. Archambault, and C. Scheidegger. A simple approach for boundary improvement of Euler diagrams. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):678–687, 2015. doi: 10.1109/TVCG.2015.2467992 2
- [33] P. Simonetto, D. Auber, and D. Archambault. Fully automatic visualisation of overlapping sets. In *Computer Graphics Forum*, vol. 28, pp. 967–974. Wiley Online Library, 2009. doi: 10.1111/j.1467-8659.2009.01452.x 2
- [34] G. Stapleton, J. Flower, P. Rodgers, and J. Howse. Automatically drawing Euler diagrams with circles. *J. Vis. Lang. Comput.*, 23(3):163–193, 31 pages, June 2012. doi: 10.1016/j.jvlc.2012.02.001 2
- [35] J. Stasko, C. Gorg, Z. Liu, and K. Singhal. Jigsaw: Supporting investigative analysis through interactive visualization. In *2007 IEEE Symposium on Visual Analytics Science and Technology*, pp. 131–138, 2007. doi: 10.1109/VAST.2007.4389006 2
- [36] 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*, 27(1):1–13, 2019. doi: 10.1109/TVCG.2019.2933196 2
- [37] H.-Y. Wu, B. Niedermann, S. Takahashi, M. J. Roberts, and M. Nöllenburg. A survey on transit map layout – from design, machine, and human perspectives. *Computer Graphics Forum*, 39(3):619–646, 2020. doi: 10.1111/cgf.14030 2
- [38] Y. Zhou, A. Rathore, E. Purvine, and B. Wang. Topological simplifications of hypergraphs. *IEEE Transactions on Visualization and Computer Graphics*, 29(7):3209–3225, 2023. doi: 10.1109/TVCG.2022.3153895 2, 9, 12, 13
- [39] A. A. Zykov. Hypergraphs. *Russian Mathematical Surveys*, 29(6):89, 1974. doi: 10.1070/RM1974v029n06ABEH001303 4