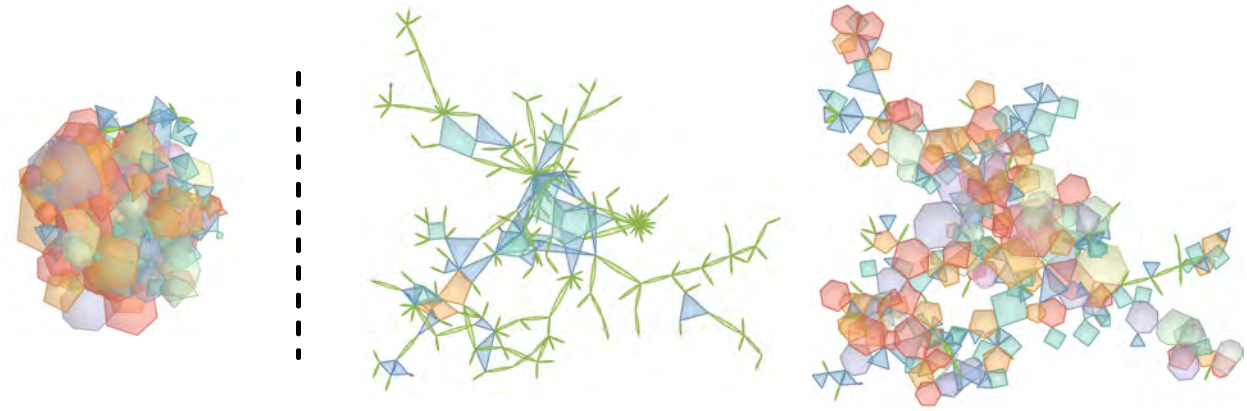
# Scalable Hypergraph Visualization

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



(a) Qu et al. [41] (b) Our method: intermediate scale (c) Our method: final optimized layout

Fig. 1: A paper-author hypergraph network with 786 vertices and 318 hyperedges. We first show the result of Qu et al. [41] in (a). Using our framework, the same hypergraph is simplified before the layout optimization begins (b). Then the simplification is iteratively reversed and the layout is refined until an optimized layout for the original hypergraph is recovered in (c).

**Abstract**—Hypergraph visualization has many applications in network data analysis. Recently, a polygon-based representation for hypergraphs has been proposed with demonstrated benefits. However, the polygon-based layout often suffers from excessive self-intersections when the input dataset is relatively large. In this paper, we propose a framework in which the hypergraph is iteratively simplified through a set of atomic operations. Then, the layout of the simplest hypergraph is optimized and used as the foundation for a reverse process that brings the simplest hypergraph back to the original one, but with an improved layout. At the core of our approach is the set of atomic simplification operations and an operation priority measure to guide the simplification process. In addition, we introduce necessary definitions and conditions for hypergraph planarity within the polygon representation. We extend our approach to handle simultaneous simplification and layout optimization for both the hypergraph and its dual. We demonstrate the utility of our approach with datasets from a number of real-world applications.

**Index Terms**—Hypergraph visualization, scalable visualization, polygon layout, hypergraph embedding, primal-dual visualization

## 1 Introduction

Hypergraphs are a generalization of graph data structures consisting of a set of vertices and a family of hyperedges. A hyperedge joins any number of $n \geq 1$ vertices and provides a natural way to represent *polyadic* (multi-sided) relationships [40]. Hypergraphs can be thought of as networks of polyadic relationships and have many applications in social sciences, biology, computer science, and engineering where such relationships are prevalent [4].

Hypergraph visualization has seen many advances in recent decades [4] with a focus on finding a proper visual metaphor for representing hyperedges (polyadic relationships) to facilitate a number of common analysis tasks. Qu et al. [40] introduce a visual metaphor in which each hyperedge takes the form of a 2D polygon in the plane, whose vertices encode the members of the underlying polyadic relationship. This representation allows the cardinality of a hyperedge (number of vertices) to be easily understood. For example, a paper-author hypergraph dataset, in which each vertex represents an author and each hyperedge a research paper, can be visualized with the polygon metaphor to easily communicate the number of co-authors for each publication (Figure 1). Qu et al. [41] also develop an optimization framework that can automatically generate a high-quality polygon layout for a hypergraph with tens of hyperedges based on a set of visualization design principles that they identify. Recognizing the duality between the vertices and the hyperedges in a hypergraph, they augment their optimization framework to simultaneously generate high-quality layouts for the input hypergraph and its *dual hypergraph* in which the roles of the vertices and hyperedges are reversed.

However, scalability presents a major challenge to their approach for large hypergraph datasets, which can have hundreds or thousands of vertices and hyperedges. With a relatively large dataset, their optimization process can be trapped at local minima, producing suboptimal layouts with excessive overlaps between polygons (Figure 1 (a)). To address this challenge, we introduce a new polygon-based layout optimization framework in which a complex hypergraph is automatically simplified by iteratively applying a set of atomic simplification operations that we have identified. The simplification process terminates when one or more user-specified criteria are met. Next, we generate a polygon layout for the simplified hypergraph using a version of the optimization

- *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: zhange@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.xxxx/TVCG.201x.xxxxxxx*

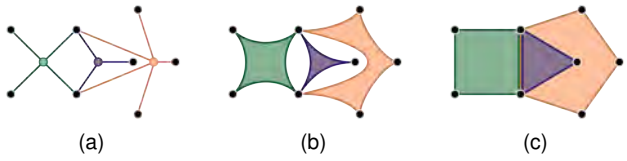



Fig. 2: A hypergraph with a planar König representation (a) is plane embeddable when the regions can be represented by arbitrary shapes as in Zykov's representation (b). However, with the additional requirement of polygon convexity (c), the hypergraph has unavoidable overlaps using the polygon representation [41].

method of Qu et al. [41] which we modify to make efficient use of their optimization energy terms. From there, our framework iteratively inverts the simplification operations until the input hypergraph is recovered. Each time a simplification is inverted, the layout is locally optimized over a neighborhood immediately surrounding the location of the operation. Our framework (Figure 1 (c)) leads to an improved final layout of the original hypergraph compared to the framework of Qu et al. [41] (Figure 1 (a)). We extend our framework to handle simultaneous layout optimization of the hypergraph and its dual hypergraph, taking advantage of the fact that our atomic operations naturally simplify both hypergraphs. This is demonstrated in one of our case study examples in Section 6 (Figure 9). During the simplification step, the order of atomic operations is determined by a priority measure which we design to reduce the amount of overlap among polygons and preserve local structures such as high-degree and centrally located elements in the visualization.

As the polygon-based metaphor requires all polygons to be convex, a hypergraph may not be plane-embeddable even when it is planar (Figure 2). That is, the requirement of convexity greatly reduces the set of hypergraphs that can be mapped to the plane without self-overlap in the regions representing the hyperedges. We investigate this issue and introduce a new notion of hypergraph planarity with convex polygons. Being able to detect subsets of non-planar hyperedges allows us to save time on attempting to remove overlaps between such hyperedges.

Our framework exceeds the performance of [41] for hypergraph datasets with more than 1000 elements in terms of reducing polygon overlaps which is crucial to visual clarity. To enable our priority-guided simplification, we also introduce a new vertex and hyperedge based statistic called *adjacency factor* which correlates to non-planar sub-hypergraphs.

We demonstrate the utility of our framework with two applications: (1) a paper-author collaboration network and (2) a network of international trade agreements. To evaluate the effectiveness of our layout framework, we conduct a user survey where participants have completed analysis tasks using our final optimized layouts as well as a few scales of simplification. We utilize eye-tracking technology to study participants' exploration of our visualizations while they answered task-driven questions. The preliminary results suggest that our new layout method allowed the survey participants to perform the tasks with relatively high accuracy.

We make the following contributions to hypergraph visualization:

1. A novel multi-scale optimization framework for generating high-quality polygon-based visualizations of hypergraphs with thousands of vertices and hyperedges.

2. A novel priority-guided hypergraph simplification method which is the first to operate on both vertices and hyperedges.

3. A set of atomic simplification operations which can simplify a hypergraph and its dual hypergraph simultaneously.

4. A new definition for hypergraph planarity within the polygon visualization metaphor.

## 2 Related Work

In this section, we review past research in graph and hypergraph visualization that is most relevant to our work.

### 2.1 Hypergraph Visualization

Hypergraph visualization has been well explored during recent decades [4]. Much of this research has focused on identifying the visual representation of hyperedges, such as matrices [32, 35, 45, 55], bipartite graphs [3, 17, 52], and metro lines [18, 27, 56]. Region-based visual metaphors derived from Euler and Venn diagrams [39, 43, 48, 51], represent sets (hyperedges) as closed regions whose overlaps indicate the intersections of their corresponding sets. The vertices in the hypergraph are often not explicitly shown, such as [43]. More recent approaches explicitly represent set elements (vertices) by drawing them as points inside the corresponding regions [3, 5, 42, 46, 47]. As pointed out in [41], placing the vertices inside the regions can make it difficult to identify the cardinality of the hyperedges. Instead, Zykov [57] restricts vertex placement to the boundaries of the regions. Qu et al. [40] represent each hyperedge as a polygon so the vertices of the hyperedge are also the vertices of the polygon. Unlike Zykoy's approach where the region can take arbitrary shapes, Qu et al. [40] require the polygons to be as close to regular as possible and thus convex. With this representation, identifying the cardinality of a hyperedge is the same as recognizing the cardinality of the corresponding polygon. Qu et al. [41] identify a number of design principles for polygon-based hypergraph drawings and develop an automatic layout optimization system based on these principles.

However, the objective functions used in [41] are not convex, thus leading to local minimums that make the final hypergraph layouts suboptimal, especially for large datasets. In addition, some of the hypergraphs cannot be embedded in the plane without overlaps using the polygon representation, even when they are plane embeddable if the hyperedges are represented by arbitrary (possibly non-convex) shapes. In this paper, we introduce a new multi-scale optimization framework that can lead to improved hypergraph layouts compared to those from [41]. Furthermore, we introduce the notion of polygon planarity, which can save on computation attempting to remove overlaps among hyperedges that are inevitable due to the polygon convexity requirement.

### 2.2 Graph and Hypergraph Simplification

Techniques for reducing complexity in graphs have been well studied and provide numerous advantages for improving graph-based algorithm efficiency and graph visualization. Depending on the application, it may be more valuable to reduce the number of graph vertices (coarsening), or the number of edges (sparsification) [9].

Graph sparsification algorithms have been studied extensively and two main categories of graph sparsifiers have arisen: *cut sparsifiers* and *spectral sparsifiers*. We review only the most relevant works here. Benczúr and Karger [6] introduce cut sparsifiers which approximate every cut in a weighted graph to an arbitrarily small multiplicative error. Spielman and Teng [50] introduce the stronger notion of spectral sparsifiers which approximate the Laplacian quadratic form of the graph to an arbitrarily small multiplicative error.

Graph coarsening has been primarily used to construct multi-level graph frameworks for graph partitioning problems. Such frameworks transform an input graph $G_0$ into a sequence of smaller graphs $G_1, G_2, ..., G_n$ such that each level in the sequence contains fewer vertices than the previous graph. This is usually accomplished through a graph coarsening scheme in which a set of vertices in $G_i$ is merged into a single *multi-node* in the next coarser level $G_{i+1}$. Identifying appropriate vertex sets for merging has followed two main approaches: vertex pair matching [11, 24, 31] and vertex grouping based on some graph-based statistics such as high connectivity or affinity [14, 19, 21, 22, 25, 44].

Several frameworks combine graph sparsification with multi-level coarsening to reduce the number of vertices and edges in a graph. Imre et al. [25] perform sparsification and coarsening in two separate phases of their algorithm while Bravo-Hermsdorff and Gunderson [9] present a

Fig. 3: A primal hypergraph (left) and its dual (right). The neighborhood of a vertex in the primal hypergraph (left: the orange dot) corresponds to the neighborhood of its dual hyperedge in the dual hypergraph (right: the orange triangle).

unified framework that incorporates vertex deletion, vertex contraction, edge deletion, and edge contraction.

Hypergraph sparsification is less studied but has been gaining traction in recent years. Cut sparsifier algorithms have been extended to hypergraphs with near-linear time complexities [12,33], and most recently with a sub-linear time complexity [13]. The notion of the Laplacian for undirected hypergraphs is introduced by Louis [38], which has recently been used to design algorithms for producing linear hypergraph spectral sparcifiers [29,49].

Multi-level coarsening has also been extended to hypergraph partitioning [2, 15, 23, 30]. Alpert et al. [2] first convert the hypergraph to a graph by replacing each hyperedge with a graph clique and applying existing graph coarsening schemes. Karypis et al. [30] develop a coarsening scheme that acts directly on the hypergraph in which vertices belonging to selected hyperedges are merged together. More recent applications of hypergraph coarsening have presented distributed hypergraph partitioners using parallel versions of the multi-level technique [16, 28, 54].

To our knowledge, we are the first to present a unified hypergraph simplification framework that operates on both hypergraph vertices and hyperedges. In addition, we introduce the notion of polygon planarity for hypergraphs and develop a simplification priority function that aims to preserve structures in the hypergraphs as well as reduce unnecessary polygon overlaps.

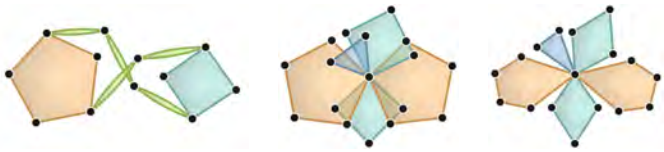
## 3 Background and Notations

Following the terminology of Berge [7] and Bretto [10], a hypergraph $H = \langle V, E \rangle$ on a finite set of $n$ vertices $V$ is defined by a family of $m$ hyperedges $E$. A hyperedge $e \in E$ contains a non-empty subset of vertices $V_e \subseteq V$ which we say 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 we say are *incident* to $v$ and *adjacent* to each other. Let $E_e$ denote the set of hyperedges adjacent to $e$ and $V_v$ the set of vertices adjacent to $v$. $H$ is *complete* if all vertices in $V$ are adjacent to each other and *linear* if $|V_e \cap V_f| \leq 1$ for all $e \neq f \in E$. $H$ is *connected* if there exists an alternating sequence of vertices and hyperedges connecting each pair of distinct vertices in $H$. The hypergraphs we consider in this paper are assumed to be connected unless otherwise specified. Consistent with [41], we define the *degree* of a vertex $v$ as $\deg(v) = |E_v|$ and the *cardinality* of a hyperedge $e$ as $\text{card}(e) = |V_e|$. Notice that the traditional notion of a graph is simply a hypergraph where every hyperedge has cardinality two.

The *dual hypergraph* $H' = \langle V', E' \rangle$ of $H$ is obtained by swapping the roles of vertices and hyperedges in $H$. For convenience, we call the original hypergraph $H$ 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 $e' \in V'$. Furthermore, the incidence and adjacency relationships of corresponding elements in the primal and dual hypergraphs are identical. This means that the degree of a vertex $v \in V$ is the same as the cardinality of the corresponding hyperedge $v' \in E'$ and vice versa. Thus, the dual of a linear hypergraph is also linear [7].

For a set of vertices $A \subseteq V$ and a set of hyperedges $J \subseteq E$, Berge [7] defines the *sub-hypergraph induced by A* and the *partial hypergraph generated by J* as respectively,

$$H_A = \langle A, \{e \cap A \,|\, e \in E, e \cap A \neq \varnothing\} \rangle \quad \text{and} \quad H_J = \langle V_J \subseteq V, J \rangle.$$



(a) (b) (c)

Fig. 4: Examples of avoidable overlaps in layouts of hypergraphs that have a convex polygon representation. In (a), un-twisting the layout would require making one of the polygons temporarily irregular, so the optimization halts before overlap can be resolved. In (b) and (c), there is not enough angular space around the central vertex for all the hyperedges to be drawn as regular polygons. In (b), the overlap minimization objective is compromised. In (c), the regularity maximization objective is compromised.

For a vertex $v \in V$, we call the partial hypergraph generated by the hyperedges incident to $v$, $H_{E_v}$, with vertex set $V_v \cup \{v\}$ the *neighborhood of v*. We define the neighborhood of a hyperedge $e \in E$ to be the sub-hypergraph induced by the vertices incident to $e$, $H_{V_e}$. In other words, the neighborhood of a vertex or hyperedge simply consists of all its incident and adjacent elements (Figure 3).

For a hypergraph $H = \langle V, E \rangle$, the König representation $K(H) = (X \cup Y, D)$ is a bipartite graph with vertices $x \in X$ for every hypergraph vertex $v \in V$ and vertices $y \in Y$ for every hyperedge $e \in E$. Vertices $x \in X$ and $y \in Y$ form an edge $(x, y) \in D$ only if the hypergraph vertex corresponding to $x$ and the hyperedge corresponding to $y$ are incident in $H$. The hypergraph $H$ is *Zykov planar* if $K(H)$ is a planar graph.

## 4 Hypergraph Simplification

In this section, we describe the building blocks of our multi-scale hypergraph layout optimization framework: (1) the set of atomic hypergraph simplification operations and (2) a number of terms used in our simplification objectives.

### 4.1 Atomic Operations

Two guiding principles for generating high-quality polygon layouts of hypergraphs are to maximize the regularity of each polygon and minimize overlap between the polygons [41]. As such, non-planar hypergraphs are difficult to handle through optimization since their polygon layouts contain necessary overlap leading to local minima in the optimization space. Non-planar sub-hypergraphs tend to appear in clusters of hyperedges that share common vertices and are also caused by structures analogous to $K_5$ and $K_{3,3}$ from graph theory. Certain structures in planar hypergraphs are also prone to overlaps, such as high-degree vertices that do not have enough angular space around them to draw each of their incident hyperedges as non-overlapping regular polygons. In these situations, a conflict between overlap minimization and regularity maximization makes optimization more difficult and requires that one or both of these objectives be compromised in the final results (Figure 4). A core idea behind our multi-scale layout optimization approach is to use simplification to reduce the challenging configurations in both planar and non-planar portions of the input hypergraph and avoid optimization that terminates prematurely. To achieve this, we present a set of hypergraph simplification operations specifically designed to eliminate these challenging configurations.

We identify four atomic operations for simplifying a hypergraph $H$:

1. *Vertex removal:* a vertex is removed from $H$.
2. *Hyperedge removal:* a hyperedge is removed from $H$.
3. *Vertex merger:* a pair of adjacent vertices are combined into a single vertex whose set of incident hyperedges is the union of the two inputs.
4. *Hyperedge merger:* a pair of adjacent hyperedges are merged into a single hyperedge whose set of incident vertices is the union of the two inputs.

The vertex removal and hyperedge removal operations form a primal-dual pair in the sense that applying one to the primal hypergraph is equivalent to applying the other to the dual hypergraph (Figure 5 (a,b)). The vertex merger and hyperedge merger operations similarly form a primal-dual pair (Figure 5 (c,d)). We define the *footprint* of an operation $O$ to be the union of the neighborhoods of its operand elements. For example, if $O$ merges two vertices $u, v \in V(H)$, the footprint of $O$ is given by $H_O = \langle V_u \cup V_v, E_u \cup E_v \rangle$.

Each atomic simplification operation has a corresponding inverse operation:

1. *Vertex addition:* a removed vertex is added back into $H$.
2. *Hyperedge addition:* a removed hyperedge is added back into $H$.
3. *Vertex split:* a merged vertex is split into two vertices with one or more common hyperedges.
4. *Hyperedge split:* a merged hyperedge is split into two hyperedges that contain one or more common vertices.

These inverse operations are used to reverse simplification and similarly form primal-dual pairs.

A sequence of atomic simplification operations $\{O_1, O_2, \ldots, O_n\}$ on a hypergraph $H$ defines a sequence of simplified scales $\{H_0, H_1, H_2, \ldots, H_n\}$ where $H_0 = H$ and $H_i = O_i(H_{i-1})$. Here $O_i(*)$ denotes applying operation $O_i$ to a hypergraph. In this multi-scale representation, we call $H_0$ the *input* or *original scale*, each $H_i = \langle V_i, E_i \rangle$ $(0 < i \leq n)$ the *i-th simplified scale*, and $H_n$ the *coarsest simplified scale*. Given the nature of our atomic operations, each simplified scale is smaller than the previous scale, i.e., $|V(H_i)| + |E(H_i)| > |V(H_{i+1})| + |E(H_{i+1})|$. By defining a prioritized sequence of atomic operations, we can construct a multi-scale representation where the size or number of non-planar sub-hypergraphs is reduced at each scale. In such a representation, it is generally easier to optimize the polygon layouts of successive simplified scales. This observation is central to our multi-scale layout optimization framework where we start by optimizing the coarsest simplified scale and handle the non-planar sub-hypergraphs on a localized basis while reversing simplification.

### 4.2 Simplification Objectives

Numerous vertex and hyperedge based statistics could be used to guide the prioritization of atomic operations to achieve a variety of simplification objectives. We consider three statistics for this purpose: *vertex degree* and *hyperedge cardinality, betweenness centrality,* and *adjacency factor*. Vertex degree and hyperedge cardinality are straightforward to compute based on the incidence relationships present in the hypergraph. Since the incidence relationships are identical between corresponding primal and dual hypergraph elements, the degree of a primal vertex is the same as the cardinality of its dual hyperedge and vice versa. By computing both vertex degree and hyperedge cardinality, we account for the primal and dual hypergraphs simultaneously. Vertex degree gives us an estimate of how much angular space is needed around the vertex for its incident polygons, and hyperedge cardinality gives us an estimate of how much area each hyperedge requires in an optimized polygon layout. As such, simplifying high-degree vertices and high-cardinality hyperedges can leave more space in the layout for neighboring elements and potentially reduce avoidable polygon overlaps. However, simplifying high-degree vertices and high-cardinality hyperedges may not be appropriate if they have an important semantic meaning in the underlying dataset.

*Betweenness centrality* quantifies the proportion of shortest paths passing through a given vertex or hyperedge. Let $\sigma_{st} = \sigma_{ts}$ denote the number of shortest paths between $s, t \in V$, where $\sigma_{ss} = 1$ by convention. Let $\sigma_{st}(v)$ denote the number of shortest paths from $s$ to $t$ passing through $v \in V$. Then the betweenness centrality for $v$ is given by

$$C_B(v) = \sum_{s \neq v \neq t \in V} \frac{\sigma_{st}(v)}{\sigma_{st}}.$$

We compute the betweenness centrality of vertices and hyperedges simultaneously by applying the algorithm of Brandes [8] to the König graph $K(H)$. To avoid an explicit summation in the betweenness centrality computation of each element, Brandes' algorithm leverages a recursive relationship for partial sums. Their algorithm is able to accumulate partial sums over a single depth-first search and return the betweenness centralities of each vertex. Avoiding simplification of elements with high betweenness centrality can be used to preserve the path structure in the input hypergraph and to preserve path-related features such as hypergraph cycles.

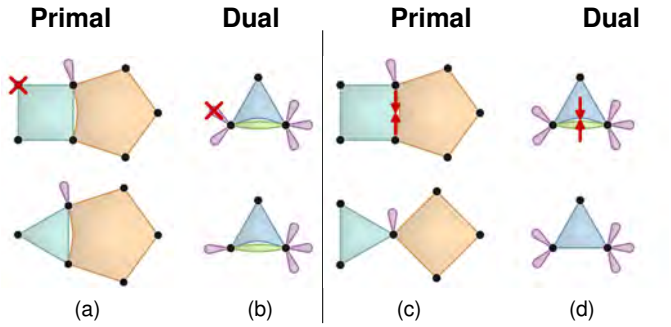



Fig. 5: The four atomic simplification operations. (a) A vertex removal in the primal hypergraph corresponds to (b) a hyperedge removal in the dual hypergraph. (c) A vertex merger in the primal hypergraph corresponds to (d) a hyperedge merger in the dual hypergraph.

We define a new statistic, adjacency factor, to measure the volume of connections between a given vertex or hyperedge and its adjacent elements. It is an extension of adjacency as defined by Bretto [10] for their construction of a hypergraph adjacency matrix. Bretto defines the *adjacency* between a pair of vertices $u, v \in V$, $u \neq v$ as $a_{uv} = |\{e \in E : u, v \in V_e\}|$. Adjacency relationships are identical between corresponding primal and dual elements, so it is natural to consider the adjacency between a pair of hyperedges $e, f \in E$ as being equal to the adjacency of their dual vertices $e', f' \in V'$, i.e., $a_{ef} = a_{e'f'}$. We define the *adjacency factor* of a vertex $v \in V$ and a hyperedge $e \in E$ as

$$\mathrm{Adj}(v) = \sum_{u \in V, u \neq v} a_{uv}^t, \qquad \mathrm{Adj}(e) = \sum_{f \in E, f \neq e} a_{ef}^t,$$

where $t \geq 0$ is used to adjust the influence of vertex pairs with multiple shared hyperedges and hyperedge pairs with multiple shared vertices. Notice that setting $t = 0$ simply gives the number of vertices adjacent to $v$, i.e. $|V_v|$. Setting $t > 0$ results in a larger adjacency factor for vertices having high adjacency with their neighbors. We discuss the ideal value for $t$ in the next section, in conjugation of the planarity issue of polygon representations of hypergraphs.

#### 4.2.1 Polygon Planarity

A graph is planar, i.e. an edge crossing-free embedding can be found, if and only if it does not contain a subdivision of the complete graph $K_5$ or complete bipartite graph $K_{3,3}$ [34].

Recall that a hypergraph $H$ is Zykov planar if its König representation $K(H)$ is a planar graph (Section 3). This definition of planarity assumes that hyperedges can be represented as arbitrary closed regions. However, when requiring that the regions be drawn as near-regular polygons, as in [40, 41], Zykov's definition is insufficient. This motivates a new definition for hypergraph planarity for the (near-regular) polygon representation:

**Definition 1.** *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.*

We say that a hypergraph is *convex polygon planar* if it admits a convex polygon representation. We have identified four *forbidden sub-hypergraphs* that are Zykov planar but lack a convex polygon representation. We begin by defining an *n-adjacent cluster* as the partial hypergraph induced by a set of hyperedges $J \subseteq E$ which contain a set of vertices $X \subseteq V$, $|X| = n \geq 2$, where each hyperedge in $J$ contains all of the vertices in $X$, that is, $v_i \in e_j$ for all $v_i \in X$ and $e_j \in J$. Our first

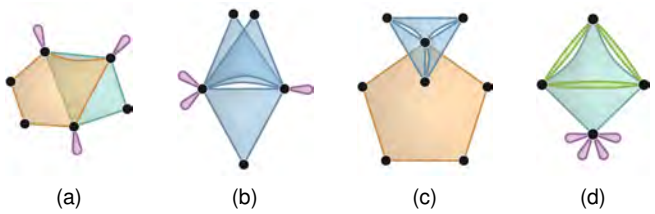



Fig. 6: Examples of the four forbidden sub-hypergraphs in the polygon visualization metaphor: (a) 3-adjacent hyperedge cluster of 2 hyperedges, (b) 2-adjacent hyperedge cluster of 3 hyperedges, (c) strangled vertex, (d) strangled hyperedge. Notice that (b) is the dual of (a) and (d) is the dual of (c).

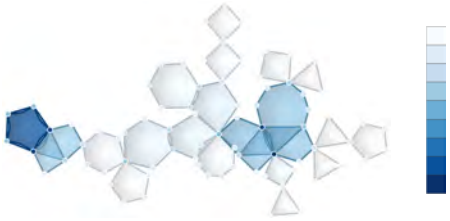


Fig. 7: Hypergraph elements colored according to adjacency factor. The two regions drawn in dark blue indicate elements with high adjacency factor and both correspond to forbidden sub-hypergraphs.

forbidden sub-hypergraph is a 3-adjacent cluster of two hyperedges (Figure 6 (a)), and the second is a 2-adjacent cluster of three hyperedges (Figure 6 (b)). Notice that these sub-hypergraphs are a primal-dual pair: if one appears in the primal view, the other appears in the dual view among the corresponding dual elements. Our third forbidden sub-hypergraph is the neighborhood of a vertex $v$ where a proper subset of its incident hyperedges and adjacent vertices form a cycle of length $n \geq 3$ (Figure 6 (c)). The fourth is the neighborhood of a hyperedge $e$ where a proper subset of its incident vertices and adjacent hyperedges form a cycle of size $n \geq 3$ (Figure 6 (d)). These sub-hypergraphs also form a primal-dual pair. We refer to these forbidden sub-hypergraphs as containing a strangled vertex or strangled hyperedge respectively.

**Theorem 2.** *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 cluster of 2 hyperedges,*
*(b) A 2-adjacent cluster of 3 hyperedges,*
*(c) A strangled vertex,*
*(d) A strangled hyperedge.*

We refer the reader to Appendix A for a proof of Theorem 2. Since the forbidden sub-hypergraphs form primal-dual pairs, we further claim that a hypergraph $H$ has a convex polygon representation if and only if its dual hypergraph $H'$ has a convex polygon representation.

We refer to polygon overlaps occurring in a polygon layout of a hypergraph that has a convex polygon representation as *avoidable overlaps* (Figure 4). Otherwise, such overlaps are *unavoidable overlaps*. Note that forbidden sub-hypergraphs are the simplest examples of unavoidable polygon overlaps. While a 3-adjacent cluster of 5 hyperedges clearly involves more hyperedge overlaps, it also necessarily contains a 3-adjacent cluster of 2 hyperedges.

Our atomic operations are specifically designed to enable eliminating forbidden sub-hypergraphs. Notice that each of the examples in Figure 6 can be converted to a hypergraph with a convex polygon representation using a single vertex or hyperedge operation. We find a good correlation between forbidden sub-hypergraphs (Figure 7) and our adjacency factor when $t = 2$. Given this correlation, simplifying elements with high adjacency factor can reduce the number and size of non-convex polygon planar sub-hypergraphs.

## 5 Scalable Optimization Framework

In this section, we detail our multi-scale polygon layout optimization framework which consists of two iterative processes: iterative simplification, and iterative layout refinement. The goal of the simplification process is to construct a sequence of simplified scales from an input hypergraph $H$ such that each successive scale contains fewer areas of potential polygon overlap, either unavoidable overlaps caused by forbidden sub-hypergraphs or avoidable overlaps caused by a lack of space around high-degree vertices and high-cardinality hyperedges. Either type of polygon overlap can lead to challenges in layout optimization and significant visual clutter in the layouts of large hypergraphs, so we address both simultaneously during simplification. Once the sequence of simplified scales $\{H_0, H_1, \ldots, H_n\}$ is generated, the goal of the iterative layout refinement process is to produce a high-quality polygon layout for each scale. We achieve this by first optimizing the layout of the coarsest scale $H_n$, then iteratively inverting simplification operations and locally refining the layout of intermediate scales until the original scale is recovered.

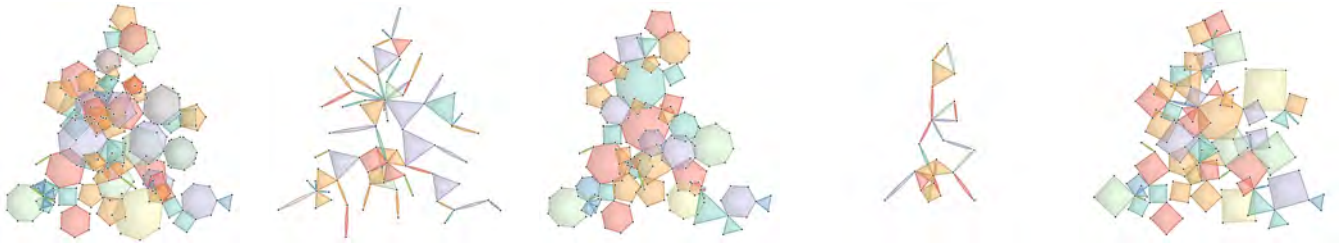
### 5.1 Simplification Operation Generation

We initially generate removal operations for every vertex and hyperedge in $H_0$, and merger operations between every pair of adjacent vertices and hyperedges. Notice that removing a vertex or hyperedge arbitrarily has the potential to make a hypergraph disconnected. To avoid this, we constrain the *legality* of removal operations. For a hyperedge $e$ at the current hypergraph scale, we mark its removal operation as *illegal* if it would make a pair of vertices $u, v \in e$ non-adjacent. Otherwise, we mark it as *legal*. Similarly, a vertex removal operation is marked illegal if it makes any pair of incident hyperedges non-adjacent. By allowing only legal removal operations, our simplification ensures that the hypergraph remains connected in each simplified scale. We also constrain the legality of merger operations to avoid simplifying portions of the hypergraph that are already linear. For a pair of hyperedges $e, f$ at the current hypergraph scale, we mark their merger operation as legal if their adjacency $a_{ef} \geq 2$ and illegal otherwise. For a pair of vertices $u, v$ in the current hypergraph scale, we mark their merger operation as legal if their adjacency $a_{uv} \geq 2$ and illegal otherwise. After the legality of each generated operation has been determined, we place the legal operations in a priority queue keyed on a simplification priority measure.

Our operation priority measure consists of three terms based on the following statistics: vertex degree (hyperedge cardinality), adjacency factor, and betweenness centrality. As discussed in Section 4.2.1, adjacency factor is correlated to the presence of a forbidden sub-hypergraph. Instead of trivially deleting or collapsing forbidden sub-hypergraphs, we use a term based on adjacency factor to promote simplifying sub-hypergraphs until they have a convex polygon representation (Figure 8 (c)). We use the term based on vertex degree and hyperedge cardinality to promote reducing the space required by high-degree vertices and large hyperedges in the polygon layouts of simplified scales. This can help to reduce avoidable polygon overlaps (Figure 8 (b)). Finally, we use the term based on betweenness centrality to promote preserving centrally located elements that are relevant to the path structure of the input hypergraph (Figure 8 (d)). By combining these terms, we are able to generate simplified scales with reduced visual clutter in areas with the most polygon overlaps, while also retaining the relative polygon sizes and core connectivity of the hypergraph (Figure 8 (e))

To normalize the distributions of each statistic, we also require the global minimum and maximum values of the input hypergraph $H_0$ (and its dual $H_0'$) for vertex degree and hyperedge cardinality, $d_{min}, d_{max}$, hyperedge adjacency factor, $a_{min}, a_{max}$, and betweenness centrality, $b_{max}, b_{min}$. Given an atomic simplification operation $O$, the final priority measure is a weighted sum of our three terms given by

$$\mathrm{P}(O) = \alpha\left(\frac{\hat{d}_O - d_{min}}{d_{max} - d_{min}}\right) + \beta\left(\frac{\bar{a}_O - a_{min}}{a_{max} - a_{min}}\right) + \gamma\left(\frac{b_{max} - \bar{b}_O}{b_{max} - b_{min}}\right). \quad (1)$$



(a) Input hypergraph (b) $\alpha = 1, \beta = 0, \gamma = 0$ (c) $\alpha = 0, \beta = 1, \gamma = 0$ (d) $\alpha = 0, \beta = 0, \gamma = 1$ (e) $\alpha = 0.4, \beta = 0.4, \gamma = 0.2$

Fig. 8: A paper-author network with 260 vertices and 83 hyperedges is simplified with our system using different values for the weight parameters $\alpha, \beta, \gamma$ in Equation (1). In (b), only the term targeting high-degree vertices and high-cardinality hyperedges is used, requiring 220 atomic operations to make the hypergraph linear. Many of the polygons in this layout have been simplified to digons, making it difficult to determine the relative sizes of the hyperedges in the original data. In (c), only the term targeting elements with high adjacency factor is used, requiring 103 atomic operations to make the hypergraph linear. The relative sizes of the hyperedges are more accurately maintained, but the visualization still contains many avoidable polygon overlaps, especially around high-degree vertices. In (d), only the term preserving elements with high betweenness centrality is used, requiring 304 atomic operations to make the hypergraph linear. Most of the information on vertex degree and hyperedge cardinality is lost in this visualization. We can however see evidence of three hypergraph cycles which are not apparent in the other simplifications. In (e), all three terms are used, requiring 135 simplification operations to make the hypergraph linear. This priority weighting scheme and visualization preserves some information on relative hyperedge sizes and also reduces visual clutter around high-degree vertices.

Here we use $\bar{a}_O$ to denote the adjacency factor of the removed element in the case of a removal operation, and the average adjacency factor of the merged elements in the case of a merger operation. Similarly, $\bar{b}_O$ denotes the average betweenness centrality of the operand elements. We use $\hat{d}_O$ to denote the maximum vertex degree or hyperedge cardinality in the footprint of $O$ (Section 5.1). That is,

$$\hat{d}_O = \max\left\{ \max_{v \in V(H_O)} \{deg(v)\}, \max_{e \in E(H_O)} \{card(e)\} \right\}. \quad (2)$$

This helps to simplify elements surrounding a high-degree vertex instead of removing or merging the high-degree vertex itself as demonstrated in Figure 8 (b).

We compute betweenness centralities once for the input hypergraph $H_0$ and use these values for the entire simplification process. This is because we aim to preserve the path structure of the input hypergraph, not the path structure of the previous simplified scale. This also avoids having to recompute betweenness centrality which can be costly. We do however update vertex degrees, hyperedge cardinalities, and adjacency factors after the application of each operation.

### 5.2 Iterative Simplification

At this stage, our priority queue only contains operations that can legally be applied to the input hypergraph $H_0$ with the highest priority operations at the front. Our framework proceeds to iteratively simplify $H_0$ by popping operations from the priority queue and applying them. We apply simplification iteratively to accommodate changes in legality or priority that any operation can incur. With each iteration, the priority queue is re-sorted to keep the highest priority operations at the front. When a legal vertex removal is performed, the vertex is removed from each of its containing hyperedges, reducing their cardinality by one, and subsequently deleted. The process is similar for a hyperedge removal. When a pair of vertices $u, v \in V$ are merged through a legal operation, we first update $v$ to include all the hyperedges incident to $u$ and then remove $u$ from the hypergraph. We call $u$ the *removed* vertex and $v$ the *retained* vertex. The process for hyperedge-based mergers is identical except that the roles of vertices and hyperedges are reversed.

When an operation $O$ is applied, the operands, legality, and priority of operations on elements in the footprint of $O$ may need to be updated. In addition, merger operations have the potential of making previously nonadjacent elements adjacent, making them eligible for merging. After $O$ is applied and the appropriate updates made, a record of the operation is added to a stack data structure containing applied operations eligible for future reversal. When the footprint of each operation is relatively small, the necessary updates can be completed efficiently. If $O$ removes an element $r \in V \cup E$, any merger operations involving $r$ become invalid and are removed from the priority queue. For each element $s \neq r$ in the footprint of $O$, any operation involving $s$ must have its priority updated since the incidence and adjacency relationships within its footprint will have changed. In addition, the legality of the removal operation on $s$ must be revisited since the connectivity within its footprint will have changed. If any previously illegal operation is found to be legal its priority is recalculated and the operation is added to the queue. If any previously legal operation is found to be illegal, it is removed from the priority queue.

If $O$ merges an element $r \in V \cup E$ into $t \in V \cup E$, the removal operation on $r$ becomes invalid and is removed from the priority queue. Similar to a removal operation, for each element $s \notin \{r,t\}$ in the footprint of $O$, any operation involving $s$ must have its priority updated, and if it is a removal operation, its legality must be revisited. Furthermore, if a merger operation exists between $s$ and $r$, the reference to $r$ is replaced with a reference to $t$. Finally, if a pair of elements $s_1, s_2 \notin \{r,t\}$ in the footprint of $O$ become newly adjacent, we initialize a new merger operation between them, calculate the priority of the operation, and add it to the priority queue.

We provide several options for defining the coarsest simplified scale $H_n$. The user can set a target number for the vertices or hyperedges in $H_n$, or specify that simplification terminates as soon as the hypergraph becomes linear or is free of forbidden sub-hypergraphs. Once a termination criterion has been met, the stack of applied operations $\{O_n, O_{n-1}, \ldots, O_1\}$ records the operations in the reverse order of how they were applied.

### 5.3 Simplification Reversal and Layout Optimization

Once the coarsest hypergraph scale $H_n$ has been reached, we construct the dual hypergraph $H_n'$ (if used), and apply a modified version of the automatic polygon layout framework of Qu et al. [41]. Their objective function for layout optimization includes five energy terms: polygon regularity energy, polygon area energy, polygon separation energy, polygon intersection regularity energy, and primal-dual coordination energy. They minimize this objective function by iteratively adjusting vertex locations in the primal and dual hypergraphs using an L-BFGS quasi-Newton solver [37]. Instead of minimizing each of these energies simultaneously, our modified version starts with a *separation* phase which uses only the polygon separation energy and primal-dual coordination energy, followed by a *regularity* phase that incorporates all five energy terms including the polygon regularity, area, and intersection regularity energies.

The purpose of the separation phase is to unravel any avoidable overlaps present in the initial layout of $H_n$ (i.e. to separate crossing paths or twisted cycles). Qu et al. [41] define the separation energy

between a pair of polygons as a function of the difference between their current separation and the minimum acceptable separation if the polygons are assumed to be regular. That is, given two polygons $\Gamma_1, \Gamma_2$, the separation energy between them is given by $E_{PS}(\Gamma_1,\Gamma_2) = f(d(\Gamma_1,\Gamma_2) - d_0(|\Gamma_1|,|\Gamma_2|))$ [41]. Here $d(\Gamma_1,\Gamma_2)$ is the current distance between polygon centroids and $d_0(|\Gamma_1|,|\Gamma_2|) = \rho_{|\Gamma_1|} + \rho_{|\Gamma_2|} + d_b$ is determined by the circumradii of regular polygons with cardinalities $|\Gamma_1|$ and $|\Gamma_2|$, and a constant buffer distance $d_b$. Since we do not optimize polygon regularity during the separation phase, we alter $d_0$ to be determined by the buffer distance and current radii of $\Gamma_1$ and $\Gamma_2$:

$$d_0(|\Gamma_1|,|\Gamma_2|) = \frac{1}{2}\left(\max_{u,v\in\Gamma_1} d(u,v) + \max_{u,v\in\Gamma_2} d(u,v)\right) + d_b. \quad (3)$$

In the regularity phase of the layout optimization for $H_n$, we reuse the objective function of [41]. However, instead of using the cardinalities of the hyperedges in $H_n$ for polygon area and separation energy calculations, we use the corresponding cardinalities saved from the original scale $H_0$. This helps to ensure that enough space is reserved for elements that are reintroduced during iterative layout refinement.

Once the layout of the coarsest scale $H_n$ (and $H'_n$) has been optimized, we enter an iterative process in which the applied simplification operations are reversed and the layout is refined. During each iteration, the operation at the top of the stack of recorded operations $O_i$ is popped, and its inverse operation $O_i^{-1}$ is applied to the current hypergraph scale $H_i$. The corresponding inverse operation is applied to the dual of the current hypergraph scale, $H'_i$. Whenever a vertex addition is applied to the primal hypergraph, the new vertex is positioned so that it aligns with the center of the corresponding new hyperedge in the dual hypergraph. The same is true when a vertex split is applied to the primal hypergraph: the split vertices are aligned with the centers of their counterparts in the dual hypergraph. We then optimize the positions of vertices inside the footprint of $O$ (in both primal and dual) while keeping all other vertices fixed. The polygon locations in the fixed portion of the layout are still used in the separation energy computation, but the remaining energies are only computed over the footprint of $O$. Confining the layout optimization to the local operation footprint in this way has two benefits: it speeds up the gradient and line search computations used with the L-BFGS solver and helps promote consistent vertex locations between simplified scales.

## 6 Case Studies

We apply our framework to two real-world cases: a network of international trade agreements, and a paper-author collaboration network.

Figure 9 shows a network of international regional trade agreements (RTAs) in force as of May 2022 retrieved from the World Trade Organization Regional Trade Agreements Database [1]. The dataset excludes bilateral trade agreements as well as trade agreements where one of the parties is itself an RTA. The largest RTA contains 36 participating nations and the largest number of trade agreements that a single nation participates in is 8. In the visualizations, we show both the primal and dual polygon layouts for the original scale (Figure 9 (a,c)) and one of the simplified scales (Figure 9 (b,d)). In the primal layouts, RTAs are drawn as polygons with incident vertices representing their participating nations. In the dual layouts, the nations are drawn as polygons and trade agreements as vertices. In the bottom left of the original scale primal layout (Figure 9 (a)), we see a large pink polygon with many degree-1 vertices. This polygon represents a trade agreement between island nations in the Caribbean with small services-based economies. We observe that many of the other nations participating in only one RTA have relatively small economies. These nations are easier to see in the dual visualization where they are drawn as monogons with a distinctive water-drop shape. In the original scale primal layout (Figure 9 (a)), the overlapping polygons in the center of the visualization make it difficult to tell which trade agreement has the most participants. In the dual layout (Figure 9 (c)), we can more clearly see a vertex in the center of the visualization with a particularly high degree. This vertex represents the Global System of Trade Preferences among Developing Countries (GSTP) which is the largest RTA in our dataset. In the simplified scale

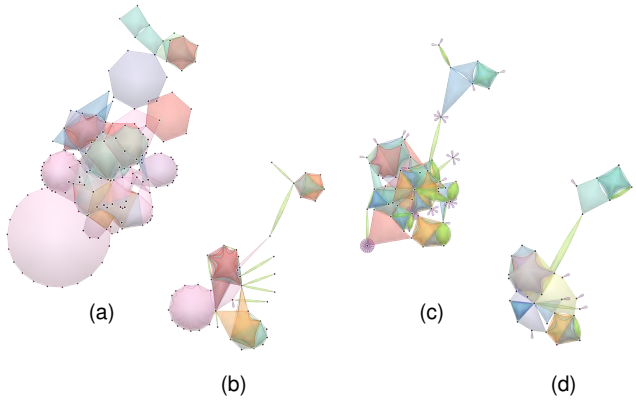



Fig. 9: Primal and dual visualizations of regional trade agreements and their participating nations. (a) Input primal layout. (b) Simplified primal layout. (c) Input dual layout. (d) Simplified dual layout.

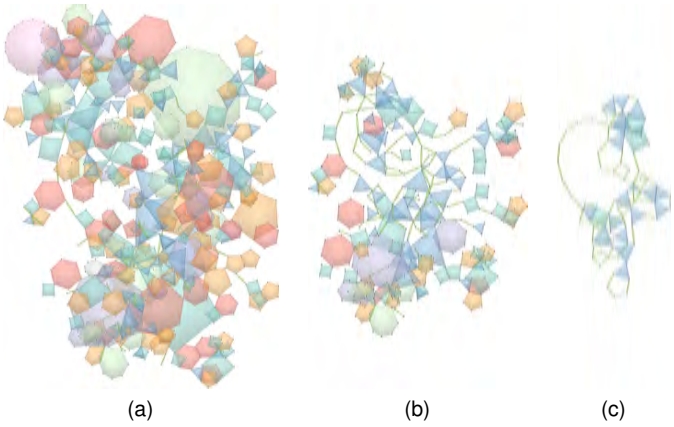



Fig. 10: A paper-author hypergraph dataset containing 1008 vertices and 429 hyperedges (a) is simplified with our framework down to the coarsest allowable scale $H_{1214}$ (c) and the layout is optimized. Then the simplification is iteratively reversed, and the layout is refined at each intermediate scale, an example of which is shown in (b), until the original scale $H_0$ is reached.

(Figure 9 (b,d)), we can see clusters of overlapping polygons in both the primal and dual layouts which indicate different trade blocs. In the upper right of the simplified layouts, we can see a cluster containing nations in Eastern Europe. In both the simplified and original scales, we can observe that this cluster is connected to the rest of the hypergraph through a single path, indicating that the trade bloc in Eastern Europe is somewhat isolated in the global economy.

Figure 10 shows the largest connected subset of publications in IEEE Transactions on Visualization and Computer Graphics (TVCG) from 2015 to 2017 retrieved from the DBLP database [36]. In these visualizations, each polygon represents a published paper whose incident vertices represent its authors. Appendix B shows an enlarged version of this figure. The dataset contains a total of 1008 vertices and 429 hyperedges with a maximum vertex degree of 17 and a maximum hyperedge cardinality of 20. Figure 10 shows an optimized layout of the original scale (a) as well as the coarsest simplified scale (c) and one intermediate scale (b) generated by our framework. Each of these scales can be used to inspect different aspects of data. In the original scale (Figure 10 (a)), we can easily see which papers have the largest number of authors by looking for the largest polygons. We can also see that these papers have numerous authors with only one publication in the dataset. Many such papers concern domain-specific visualization techniques, so we speculate that they may include domain experts as

coauthors. For example, the large pink polygon in the top left corner represents a paper on visualizations of concepts from special and general relativity and includes a team of both computer graphics and astrophysics researchers. Another polygon in the bottom left corner represents a paper on interactive visualizations for prostate cancer health risk communication and includes coauthors from computer graphics as well as medical professionals. With the optimized layout of the original scale, we can see that areas with the most polygon overlap contain a forbidden sub-hypergraph or a particularly high degree vertex. In the context of the paper-author network, forbidden sub-hypergraphs like $n$-adjacent clusters indicate a set of authors who collaborate on multiple papers. Such clusters could represent organized research groups that collaborate frequently on a series of papers. High-degree vertices represent authors with many publications who are likely experienced researchers and academic advisors.

In the coarsest simplified scale (Figure 10 (c)), we can see several cycle structures among the remaining authors and publications. Where the cycles are small and tangled, we can infer that there is a significant amount of inter-collaboration between the corresponding researchers in the community. Where the cycles are larger and more spread out, there may be less inter-collaboration and more inclusion of experts from other disciplines. In the intermediate simplified scale (Figure 10 (b)), we can clearly see branching sub-tree structures along the right side of the visualization. The ends of such branching structures may represent unique subtopics or specific domain applications that are somewhat removed from the main research topics in the journal. For example, the small grouping of polygons at the center-right of the intermediate simplified scale contains the only papers in the dataset studying visualizations of cerebral blood flow in aneurysms.

## 7 Eye Tracking Survey

We conducted a preliminary user survey among 12 graduate and undergraduate university students to evaluate the usability of our visualizations. We designed the survey to analyze how participants interact with our visualizations when presented with both primal and dual layouts as well as the layouts of several simplified scales. The survey was conducted in person and involved the use of eye-tracking hardware to monitor participants' exploration of the visualizations. The survey included visualizations of our international trade agreement dataset from Section 6, as well as two paper-author collaboration networks retrieved from Isenberg et al.'s openly available Vispubdata dataset [26]. Their dataset contains information on IEEE Visualization publications from 1990-2021. The first of these datasets, shown in Figure 1, consists of a connected subset of publications containing the keywords "flow", "graph", and "machine learning", while the second consists of the largest connected subset of papers from the InfoVis and SciVis tracks of the IEEE Visualization journal. For each dataset, we asked two questions requiring participants to analyze different properties of specific hypergraph elements. In addition to recording participant responses, we also tracked participant gaze fixation points as they completed each question. We used a Gazepoint GP3 HD 150Hz eye tracking system with a reported accuracy of 0.5-1.0° [20]. The eye tracking for each survey trial was conducted in a controlled lab environment with a 24-inch 1920x1200p 144Hz monitor.

For the trade agreement dataset, participants were presented with the original scale visualizations of the primal and dual hypergraph layouts generated by our framework (Figure 9 (a,c)). The first question asked participants to count the trade agreements involving five or more countries that only participate in one trade agreement. The second question asked participants to count the number of trade agreements that a specific country participates in. Participants answered the first question with 66.7% accuracy and the second question with 83.3% accuracy. Participants who answered the questions correctly spent an average of 58.1% viewing the dual while participants who answered incorrectly spent an average of 22.3% of their time viewing the dual (Figure 11). This points to the value of including both the primal and dual layouts for simple analysis tasks.

For the first paper-author dataset (containing 786 vertices and 318 hyperedges), participants were presented with both primal and dual layouts of the original hypergraph scale and two simplified scales generated by our framework. The first question for this dataset asked participants to count polygons of a particular color incident to a specific high-degree vertex. The second question asked them to count the number of distinct paths between a pair of vertices. Participants answered the first question with 91.7% accuracy and the second question with 50.0% accuracy. However, not all participants used the simplified scale layouts. Participants who used the simplified scales answered the first question with 100% accuracy and the second question with 66.7% accuracy. Participants who did not use the simplified scales answered the first question with 75.0% accuracy and the second question with 0% accuracy. On average, participants spent less time viewing the simplified scales than the original scale, however, the participants did not necessarily find the original scale visualization more useful. For example, Figure 12 shows a participant who spent most of their time studying the original scale layouts but was able to arrive at the correct answer soon after viewing the second simplified scale.

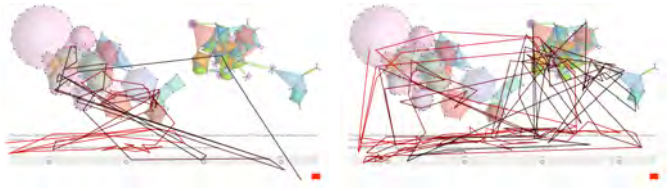


Fig. 11: Gaze fixation paths of two participants answering the same question for the trade agreement dataset in our user survey. The participant with the gaze path on the left did not study the dual hypergraph and answered the question incorrectly. The participant with the gaze path on the right studied both the primal and dual hypergraph visualizations and answered the question correctly. Enlarged versions of these plots are available in Appendix C.

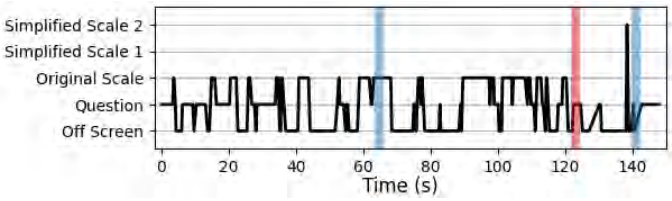



Fig. 12: Gaze fixation timeline of a participant answering a question in our user survey. The vertical axis indicates different regions on the participant's screen, including the question text and visualization scales. The horizontal axis represents the time in seconds that a participant spent on the question. The vertical lines in the plot indicate when the participant selected an answer. The blue lines indicate a correct answer and the red lines indicate an incorrect answer. An enlarged version of this plot is available in Appendix D.

For the second paper-author dataset (containing 1878 vertices and 966 hyperedges), participants were presented with the primal layout of the original hypergraph scale and two simplified scales generated by our framework. We did not include the dual layouts because the primal layout required the entirety of the user's screen to be viewed clearly. The first question asked participants to identify the highest degree vertex in the visualization, and the second question asked them to count the length of the shortest path between two hyperedges. Participants answered the first question with 91.7% accuracy and the second question with 66.7% accuracy. All participants used the simplified scale layouts for both questions. On average, participants spent 21.3% of their time viewing the original scale layout, 27.3% of their time viewing the first simplified scale, and 51.4% of their time viewing the second simplified scale. The increased time spent viewing the simplified scales along with a combined accuracy of 79.2% suggests that participants were able to effectively use the simplified scale visualizations.

Table 1: Comparison of polygon layout optimization methods on paper-author datasets. Each dataset is collected from the DBLP database [36] and consists of a maximal connected subset of publications in the specified year range from the given IEEE journals: Transactions on Robotics (TOR), Transactions on Pattern Analysis and Machine Intelligence (TPAMI), Transactions on Visualization and Computer Graphics (TVCG), Transactions on Human-Machine Systems (THMS), Transactions on Learning Technology (TLT), Transactions on Education (TOE), Transactions on Haptics (TOH), Transactions on Cybernetics (TOC). The first six datasets are the same as those used in [41]. Note that our method leads to much improvement in terms of fewer avoidable overlaps than the method of Qu et al. [41].

| **Dataset** | | | | **Qu et al.** [41] | | | **Ours** | | |
|---|---|---|---|---|---|---|---|---|---|
| description | $\|V\|$ | $\|H\|$ | forbidden sub-hypergraphs | execution time (s) | pairwise overlap count | sum pairwise overlap area | execution time (s) | pairwise overlap count | sum pairwise overlap area |
| TOR (2015-2020) | 22 | 11 | 1 | 0.05 | 4 | 1.48 | 0.02 | 1 | 0.42 |
| TPAMI (2015) | 77 | 24 | 6 | 0.42 | 60 | 29.11 | 0.43 | 15 | 5.63 |
| TPAMI (2013-2014) | 93 | 42 | 2 | 0.93 | 61 | 7.70 | 0.56 | 13 | 2.12 |
| TOR (2015-2020) | 146 | 56 | 8 | 2.48 | 92 | 25.29 | 2.09 | 30 | 7.88 |
| TPAMI (2013-2015) | 314 | 126 | 12 | 20.15 | 156 | 37.01 | 14.17 | 62 | 15.18 |
| TOR (2013-2020) | 527 | 232 | 37 | 61.08 | 422 | 98.12 | 63.07 | 154 | 37.12 |
| TVCG (2015-2017) | 1008 | 429 | 113† | 45.19* | 6366 | 3806.36 | 687.14 | 3119 | 1297.50 |
| THMS, TLT, TOE, TOH (2013-2023) | 1754 | 635 | 265† | 1917.49* | 6125 | 4598.78 | 4570.77 | 2895 | 1154.67 |
| TPAMI (2013-2020) | 2054 | 947 | 364† | 219.92* | 22113 | 17331.60 | 8583.15 | 8877 | 3666.70 |
| TOC (2022) | 3047 | 1000 | 172† | 4518.35 | 5566 | 3497.09 | 13156.90 | 2510 | 863.64 |

† Datasets also contain unavoidable overlaps due to $K_5$ and $K_{3,3}$ sub-hypergraphs.
* Layout optimization did not converge and terminated early.

## 8 Performance Evaluation

Our framework leverages simplification operations that generally have small, localized footprints. However, if the input hypergraph is complete, where all the vertices are adjacent to each other, the local footprint of an operation can include the entire hypergraph. This represents the worst-case scenario for the computational complexity of our framework. For a hypergraph $H = \langle V, E \rangle$, recall that $n = |V|$ and $m = |E|$. Our framework first identifies all possible removal operations for vertices and hyperedges, and all merger operations between pairs of adjacent elements, requiring $O((n+m)^2)$ time and generating $O((n+m)^2)$ operations. Our operation priority function (Equation (1)) computes a maximum over the footprint of each operation. In the worst case, the footprint of an operation contains $n+m$ elements, so ranking the operations by priority requires $O((n+m)^3)$ time for computation and $O((n+m)^2)\log((n+m)^2))$ time for sorting. Iterative simplification also requires updating the local footprint of each applied operation, adding $O((n+m)^3)$ complexity. Our notation here uses $n$ and $m$ from the original hypergraph scale even though the number of vertices and hyperedges is reduced by a constant factor in each iteration. Altogether, our framework's simplification process requires $O((n+m)^3)$ time. However, the running time of our framework is dominated by the iterative optimization process. Qu et al. [41] report that their layout optimization method has a lower bound $\omega((n+m)^4)$. Since our iterative optimization process uses a modified version of their method, we also have a lower bound of $\omega((n+m)^4)$. Each iterative layout refinement then has a lower bound of this form relative to the size of the corresponding operation footprint. Hypergraphs for most practical use cases are far from complete, so operation footprints are smaller than $n+m$. Our testing on real datasets, shown in Table 1, indicates that the execution time of our full framework is less than the theoretical bounds. Table 1 compares our method to that of [41] on datasets with tens to thousands of elements. These datasets also vary in complexity with respect to the number of unavoidable overlaps they contain. We measure this complexity using the number of forbidden sub-hypergraphs in each dataset. For each method, we display execution time as well as the number of pairwise polygon overlaps and sum of pairwise overlap areas in the optimized layouts.

## 9 Conclusion and Future Work

The main contribution of our work is a multi-scale framework for producing high-quality polygon visualizations of hypergraphs. Our framework features a novel top-down iterative simplification process followed by a bottom-up layout optimization process. To our knowledge, it is the first time that hypergraph simplification has been used specifically for layout optimization. Unlike previous work which focuses on either hyperedge-based sparsification or vertex-based coarsening, we introduce a set of atomic hypergraph simplification operations including both hyperedge and vertex-based operations. Our simplification process is guided by a custom operation priority measure which includes terms for multiple objectives: reducing visual clutter around high-degree vertices, eliminating non-planar sub-hypergraphs, and preserving hypergraph path structures. Additionally, our system is designed to handle primal and dual hypergraphs simultaneously and maintain consistency between them throughout the layout optimization process. We also introduce new and necessary theory on planarity for convex polygon drawings of hypergraphs. This includes a new criterion for convex polygon representations akin to Kuratowski's Theorem for planar graphs [34].

A major challenge that remains for our layout optimization framework is its time complexity. While our framework can handle large datasets, the execution time is constrained by an $\omega((n+m)^4)$ lower bound from the quasi-Newton optimization solver. We plan to continue exploring techniques to enhance the speed of our technique including different possibilities for layout initialization, pre-processing, and employing more efficient optimization solvers.

Our simplification system is designed for hypergraphs that are Zykov planar but lack a convex polygon representation according to our definition. We do not directly address the large class of hypergraphs that are non-planar because they contain structures analogous to $K_5$ and $K_{3,3}$. Furthermore, while our atomic simplification operations can be constrained to ensure the preservation of local connections, they do not consider global topological structures in the hypergraph. We plan to investigate topology-aware simplification methods to handle a larger class of non-planar hypergraphs and create multi-scale representations of hypergraphs that preserve their topological properties.

Preliminary results from our user survey and case studies suggest that the simplified scales used in our layout optimization process can also be used for pattern recognition in hypergraph visualizations. As such, we also plan to pursue multi-scale hypergraph representations that are focused on preserving visual structures in simplified scales. We hope to apply such a visualization system to domains like biology and medicine where interaction networks play an important role, and engineering simulation ensembles where numerous intertwined parameters influence simulation results.

## Acknowledgments

The authors appreciate the constructive feedback from our anonymous reviewers.

## REFERENCES


[1] World trade organization regional trade agreements database, May 2022.

[2] C. J. Alpert, L. W. Hagen, and A. B. Kahng. A hybrid multilevel/genetic approach for circuit partitioning. In *Proceedings of APCCAS'96-Asia Pacific Conference on Circuits and Systems*, pp. 298–301. IEEE, 1996. doi: 10.1109/APCAS.1996.569275

[3] 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

[4] B. Alsallakh, L. Micallef, W. Aigner, H. Hauser, S. Miksch, and P. Rodgers. The state-of-the-art of set visualization. *Comput. Graph. Forum*, 35(1):234–260, Feb. 2016. doi: 10.1111/cgf.12722

[5] 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

[6] A. A. Benczúr and D. R. Karger. Approximating st minimum cuts in õ (n 2) time. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pp. 47–55, 1996. doi: 10.1145/237814.237827

[7] C. Berge. *Graphs and hypergraphs*. North-Holland mathematical library, v. 6. North-Holland Pub. Co., Amsterdam, [2d rev. ed.] translated by edward minieka. ed., 1976.

[8] U. Brandes. A faster algorithm for betweenness centrality. *Journal of mathematical sociology*, 25(2):163–177, 2001. doi: 10.1080/0022250X.2001.9990249

[9] G. Bravo-Hermsdorff and L. M. Gunderson. A unifying framework for spectrum-preserving graph sparsification and coarsening. *arXiv preprint arXiv:1902.09702*, 2019. doi: 10.48550/arXiv.1902.09702

[10] A. Bretto. Hypergraph theory. *An introduction. Mathematical Engineering. Cham: Springer*, 2013. doi: 10.1007/978-3-319-00080-0

[11] T. N. Bui and C. Jones. A heuristic for reducing fill-in in sparse matrix factorization. Technical report, 12 1993.

[12] C. Chekuri and C. Xu. Minimum cuts and sparsification in hypergraphs. *SIAM Journal on Computing*, 47(6):2118–2156, 2018. doi: 10.1137/18M1163865

[13] Y. Chen, S. Khanna, and A. Nagda. Sublinear Time Hypergraph Sparsification via Cut and Edge Sampling Queries. In N. Bansal, E. Merelli, and J. Worrell, eds., *48th International Colloquium on Automata, Languages, and Programming (ICALP 2021)*, vol. 198 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pp. 53:1–53:21. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 2021. doi: 10.4230/LIPIcs.ICALP.2021.53

[14] C.-K. Cheng and Y.-C. Wei. An improved two-way partitioning algorithm with stable performance (vlsi). *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 10(12):1502–1511, 1991. doi: 10.1109/43.103500

[15] J. Cong and M. Smith. A parallel bottom-up clustering algorithm with applications to circuit partitioning in vlsi design. In *30th ACM/IEEE Design Automation Conference*, pp. 755–760. IEEE, 1993. doi: 10.1145/157485.165119

[16] K. D. Devine, E. G. Boman, R. T. Heaphy, R. H. Bisseling, and U. V. Catalyurek. Parallel hypergraph partitioning for scientific computing. In *Proceedings 20th IEEE International Parallel & Distributed Processing Symposium*, pp. 10–pp. IEEE, 2006. doi: 10.1109/IPDPS.2006.1639359

[17] 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

[18] 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

[19] J. Garbers, H. J. Promel, and A. Steger. Finding clusters in vlsi circuits. In *1990 IEEE International Conference on Computer-Aided Design*, pp. 520–521. IEEE Computer Society, 1990. doi: 10.1109/ICCAD.1990.129970

[20] Gazepoint. Gp3 eye-tracker. https://www.gazept.com, 2021.

[21] L. Hagen and A. Kahng. Fast spectral methods for ratio cut partitioning and clustering. In *1991 IEEE international conference on computer-aided design digest of technical papers*, pp. 10–11. IEEE Computer Society, 1991. doi: 10.1109/ICCAD.1991.185177

[22] L. Hagen and A. B. Kahng. A new approach to effective circuit clustering. In *ICCAD*, vol. 92, pp. 422–427, 1992. doi: 10.1109/ICCAD.1992.279334

[23] S. Hauck and G. Borriello. An evaluation of bipartitioning techniques. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 16(8):849–866, 1997. doi: 10.1109/43.644609

[24] B. Hendrickson and R. Leland. A multi-level algorithm for partitioning graphs. In *SC Conference*, p. 28. IEEE Computer Society, Los Alamitos, CA, USA, dec 1995. doi: 10.1109/SUPERC.1995.3

[25] M. Imre, J. Tao, Y. Wang, Z. Zhao, Z. Feng, and C. Wang. Spectrum-preserving sparsification for visualization of big graphs. *Computers & Graphics*, 87:89–102, 2020. doi: 10.1016/j.cag.2020.02.004

[26] 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

[27] 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

[28] I. Kabiljo, B. Karrer, M. Pundir, S. Pupyrev, A. Shalita, A. Presta, and Y. Akhremtsev. Social hash partitioner: a scalable distributed hypergraph partitioner. 10(11):1418–1429, aug 2017. doi: 10.14778/3137628.3137650

[29] M. Kapralov, R. Krauthgamer, J. Tardos, and Y. Yoshida. Spectral hypergraph sparsifiers of nearly linear size. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 1159–1170. IEEE, 2022. doi: 10.1109/FOCS52979.2021.00114

[30] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar. Multilevel hypergraph partitioning: Applications in vlsi domain. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 7(1):69–79, 1999. doi: 10.1109/92.748202

[31] G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on scientific Computing*, 20(1):359–392, 1998. doi: 10.1137/S1064827595287997

[32] 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

[33] D. Kogan and R. Krauthgamer. Sketching cuts in graphs and hypergraphs. In *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science*, ITCS '15, p. 367–376. Association for Computing Machinery, New York, NY, USA, 2015. doi: 10.1145/2688073.2688093

[34] 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

[35] A. Lex, N. Gehlenborg, H. Strobelt, 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

[36] M. Ley. DBLP Computer Science Bibliography, 2005.

[37] D. C. Liu and J. Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(1):503–528, 1989. doi: 10.1007/BF01589116

[38] A. Louis. Hypergraph markov operators, eigenvalues and approximation algorithms. In *Proceedings of the Forty-Seventh Annual ACM Symposium on Theory of Computing*, STOC '15, p. 713–722. Association for Computing Machinery, New York, NY, USA, 2015. doi: 10.1145/2746539.2746555

[39] L. Micallef and P. Rodgers. eulerAPE: Drawing area-proportional 3-Venn diagrams using ellipses. *PloS One*, 9:e101717, 07 2014. doi: 10.1371/journal.pone.0101717

[40] 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

[41] 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

[42] 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

[43] 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, p. 13–27. Springer-Verlag, Berlin, Heidelberg, 2008. doi: 10.1007/978-3-540-87730-1_6

[44] D. Ron, I. Safro, and A. Brandt. Relaxation-based coarsening and multiscale graph organization. *Multiscale Modeling & Simulation*, 9(1):407–423, 2011. doi: 10.1137/100791142

[45] 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

[46] 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

[47] 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

[48] 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

[49] T. Soma and Y. Yoshida. Spectral sparsification of hypergraphs. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 2570–2581. SIAM, 2019. doi: 10.1137/1.9781611975482.159

[50] D. A. Spielman and S.-H. Teng. Spectral sparsification of graphs. *SIAM Journal on Computing*, 40(4):981–1025, 2011. doi: 10.1137/08074489X

[51] G. Stapleton, J. Flower, P. Rodgers, and J. Howse. Automatically drawing Euler diagrams with circles. *J. Vis. Lang. Comput.*, 23(3):163–193, June 2012. doi: 10.1016/j.jvlc.2012.02.001

[52] 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

[53] C. Thomassen. Plane representations of graphs. *Progress in graph theory*, 1984.

[54] A. Trifunović and W. J. Knottenbelt. Parallel multilevel algorithms for hypergraph partitioning. *Journal of Parallel and Distributed Computing*, 68(5):563–581, 2008. doi: 10.1016/j.jpdc.2007.11.002

[55] 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

[56] 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

[57] A. A. Zykov. Hypergraphs. *Russian Mathematical Surveys*, 29(6):89, 1974. doi: 10.1070/RM1974v029n06ABEH001303