Appendix A. Questionnaire (Pre-study)

(1) Please select the range of your age.

(A) 11~ 20

(B) 21~ 30

(C) 31~ 40

(D) 41~ 50

(E) 51~ 60

(F) 60+

(2) Please select your gender.

(A) Male

(B) Female

(3) Please specify your major or your research field.

(4) Are you familiar with visualization?

(A) Very unfamiliar

(B) Unfamiliar

(C) Moderately familiar

(D) Familiar

(E) Very familiar

(5) Are you familiar with hypergraphs?

(A) Very unfamiliar

(B) Unfamiliar

(C) Moderately familiar

(D) Familiar

(E) Very familiar

(6) For the current dataset, please score each layout algorithm from two perspectives: 1. The degree of clustering of points in the same class; 2. The degree of dispersion of different classes. Please follow the 5-point Likert scale, 1 means "neither 1 nor 2 has been achieved", 2 means "1 has been achieved, 2 has not", 3 means "1 has not been achieved, 2 has achieved", 4 means "both 1 and 2 have achieved", 5 means "both 1 and 2 have been achieved, and the performance is very good", giving your score.

(7) Please detail the reasons why you give scores to different layouts.

Appendix B. Questionnaire (Formal study)

Section 1: Basic information

(1) Please select the range of your age.

(A) 11~ 20

(B) 21~ 30

(C) 31~ 40

(D) 41~ 50

(E) 51~ 60

(F) 60+

(2) Please select your gender.

(A) Male

(B) Female

(3) Please specify your major or your research field.

Section 2: Visualization background

(1) Are you familiar with visualization?

(A) Very unfamiliar

(B) Unfamiliar

(C) Moderately familiar

(D) Familiar

(E) Very familiar

(2) Have you learned about hypergraphs or used them in your work or research? If yes, please share us with the usage, the usage scenario, the purpose of usage, as well as the difficulties and dissatisfaction that you have confronted.

Section 3: Please answer the questions in Experiment 1.

Please sort the 12 images in descending order of concavity. (Concavity: refers to the number of non-convex hyperedges)

Section 4: Please answer the questions in Experiment 2.

Please sort the 12 pictures in descending order of Planarity. (Planarity: Refers to the number of hyperedge intersections)

Section 5: Please answer the questions in Experiment 3

Please sort the 12 graphs in descending order of coverage. (Coverage: Refers to the ratio of the hypergraph to the canvas)

Section 6: Please answer the questions in Experiment 4.

Please sort the 12 graphs from high to low according to the uniformity of the connection distance between nodes.

Section 7: Please answer the questions in Experiment 5.

In the displayed hypergraph, whether OLIVIA and LAURA are in the same class in the framed area?[Here is one question as an example]

(A) Yes

(B) No

Section 8: Please rate the difficulty of each task.

What do think of the difficulty of each task?[Experiment 2-6 as well]

(A) Very easy

(B) Easy

(C) Moderate

(D) Difficult

(E) Very difficult

Section 9: Please rate yourself on how quickly and correctly you completed each experiment

(1) What do you think of your speed when completing Experiment 1?[Experiment 2-6 as well]

(A) Very slow

(B) slow

(C) Moderate

(D) Fast

(E) Very fast

(2) What do you think of the accuracy of your task completion?[Experiment 2-6 as well]

(A) Very low correctness

(B) Low correctness

(C) Moderate correctness

(D) High correctness

(E) Very high correctness

Section 10: Please rank according to your liking for the 12 output hypergraphs.

Please rank according to your liking for the 12 output hypergraphs .

Section 11: Please rate the importance of the visual factors in hypergraph visualization.

(1) Do you think it is important to lower concavity in hypergraph visualization?

(A) Not at all

(B) Low Importance

(C) Neutral

(D) Important

(E) Very important

(2) Do you think it is important to improve planarity in hypergraph visualization?

(A) Not at all

(B) Low Importance

(C) Neutral

(D) Important

(E) Very important

(3) Do you think it is important to improve coverage in hypergraph visualization?

(A) Not at all

(B) Low Importance

(C) Neutral

(D) Important

(E) Very important

(4) Do you think it is important to the uniformity of distance between vertexes in hypergraph visualization?

(A) Not at all

(B) Low Importance

(C) Neutral

(D) Important

(E) Very important

(5) Do you think “improving the cognitive ability related to data category” is vital to hypergraph visualization?

(A) Not at all

(B) Low Importance

(C) Neutral

(D) Important

(E) Very important

(6) Please list other visual factors you think are also important in hypergraph visualization.

Appendix C. Methods for converting hypergraphs to graphs

In this section, we analyze 12 conversion methods (the first step in Figure 1) and categorize them into 4 types, comparing their impact on computational performance and visualization readability (defined by the metrics discussed in Section 3.3). Each method includes a description and a visualization example. For methods that require additional explanations (such as their impact on performance), a discussion section is also added. The goal of each method is to transform a hypergraph into a graph. After sorting the graphs using a layout algorithm, we may want to recover the representation back to the original form for display through a post-processing step [2]. At the end of each method type, we briefly describe how to perform post-processing on graphs. The implementation code of the different methods and post-processing steps can be found at https://github.com/Hypergraph-to-graph/Hypergraph-to-graph.

Table 3 outlines the complexity of the conversions and the post-processing cost. Table 4 contains definitions for the symbols used in the equations.

Table 3 : Definitions and notation used throughout the paper

G = (V, H)	A hypergraph containing a set of vertices $v_i \in V$ and a set of hyperedges $h_i \in H$.
$G_1 = (V_1, E)$	A graph resulting from one of our proposed conversions, containing a set of vertices vi $\in V$ and a set of edges $e_i \in E$.
$I(h_i)$	The set of vertices incident to hyperedge h_i. Can be used for edges as well.
$a(h_i)$	Aggregate vertex for hyperedge h_i.
$c(h_i)$	Centroid of hyperedge h_i.

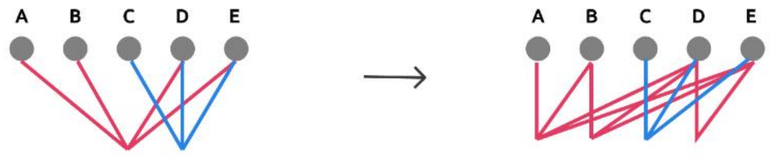
Table 4: Worst-case complexities of all the transformation methods and their impact on the complexity of the barycentric method. In this table, |I| indicates the maximum number of vertices incident to a single hyperedge. |H| indicates the number of hyperedges. |V| indicates the number of vertices.

	Transform	Post-processing
Split-complete	$O(\|H\| * \|I\|^2)$	$O(\|H\| * \|I\|)$
Split-path	$O(\|H\| * \|I\|)$	$O(\|H\| * \|I\|)$
Split-cycle	$O(\|H\| * \|I\|)$	$O(\|H\| * \|I\|)$
Aggregate-collapse	$(\|H\|^2 * \|I\|^2)$	$O(\|H\|^2 * \|I\|^2)$
Aggregate-summarize	$O(\|H\|^2 * \|I\|^2)$	$O(\|V\|^2)$
Aggregate-relationship-summarize	$O(\|H\| * \|I\| * \|V\|^2)$	$O(\|V\|^2)$
Centroid-within-layer	$O(\|H\| * \|I\|)$	$O(\|H\| * \|I\|)$
Centroid-aggregate-summarize	$O(\|H\|^2 * \|I\|^2)$	$O(\|H\| * \|I\| * \|V\|)$
Centroid-aggregate-relationship-summarize	$O(\|H\|^2 * \|I\|^2)$	$O(\|H\| * \|I\| * \|V\|)$
Line-expansion	$O(\|H\|^2 * \|I\|^2)$	$O(\|H\| * \|I\| * \|V\|)$
Line-expansion-aggregate-summarize	$O(\|H\|^2 * \|I\|^2)$	$O(\|H\| * \|I\| * \|V\|)$
Line-expansion-aggregate-relationship-summarize	$O(\|H\| * \|I\| * \|V\|^2)$	$O(\|V\|^2)$

1. Split methods
   (1) Split-clique: The first method replaces each hyperedge h_i with edges between each pair of vertices associated with h_i. The result of hyperedge conversion effectively creates a clique among all vertices associated with h_i. For example, hyperedge ABC becomes edges AB, BC, and AC, while hyperedge CDE becomes edges CD, DE, and CE.

$$E = \{e : v_i, v_i \in I(e) \forall v_i, v_j \in I(h_k), \forall h_k \in H\} \qquad (1)$$

$$V_i = V \qquad (2)$$

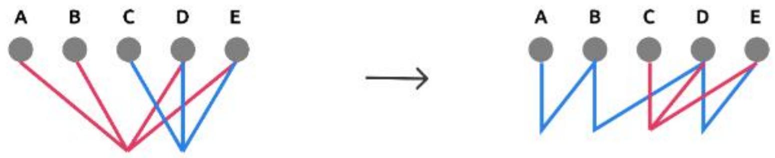


Discussion: Split-clique does not add vertices but instead adds many edges, and the number of edges grows rapidly with the number of vertices associated with the hyperedge, becoming $|I(h_i)| * (|I(h_i)| - 1) / 2$ edges for a hyperedge h_i, which naturally reduces computational performance.

(2) Split-path: Each hyperedge h_i is split into $|I(h_i)|-1$ edges, forming a path between all related vertices. In the example below, hyperedge ABC becomes edges AB and BC, while hyperedge CDE becomes edges CD and DE.

$$E = \{e : v_i,\ v_{i+1}\ \in I(e), \forall v_i,\ v_j\ \in\ I(h_k), \forall h_k \in H\} \quad (3)$$

$$V_i = V \quad (4)$$

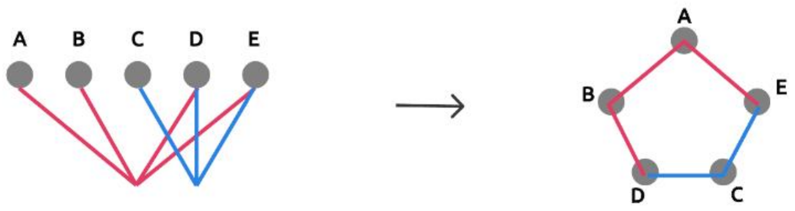


Discussion: This is done in an attempt to mitigate the performance drawbacks of the previous method. However, this method does not specify a unique set of edges to create: hyperedge ABC can become AB and BC, or AC and BC. For more information on how the choice of which edges to create impacts the output, please refer to the appendix at https://github.com/Hypergraph-to-graph/Hypergraph-to-graph.

(3) Split-cycle: Each hyperedge h_i is split into $|I(h_i)|$ edges, forming a closed circular path between all related vertices. In the example below, every two different vertices A, B, C, D, and E are connected by exactly one edge, so it can be simplified to 5 edges, which are AB, BD, DC, CE, and EA.

$$E = \left\{ \begin{matrix} e : v_i, v_{i+1} \in I(e), v_{max}, v_{min} \in I(e), \\ i \leq \max - 2, \forall h_k \in H \end{matrix} \right\} \quad (5)$$

$$V_i = V \quad (6)$$



Discussion: This method can also reduce the performance drawbacks of the first method. Additionally, due to its closed circular nature, we speculate that it might perform better in force-directed layout algorithms. For more information on how the order of vertices impacts

the output, please refer to the appendix at
https://github.com/Hypergraph-to-graph/Hypergraph-to-graph.

Post-processing for split methods: In the splitting method, it is sufficient to keep the vertices in the same positions obtained through the sorting algorithm and replace the newly added edges with hyperedges.

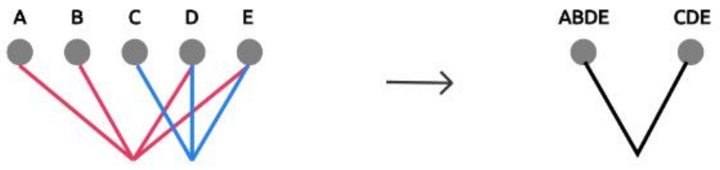
2. Aggregate methods

The aggregation method relies on creating aggregation vertices (also known as meta nodes) to remove hyperedges. These methods transfer the weight onto the preprocessing step of computing how to aggregate the graph, resulting in a much smaller graph that is faster to compute for layout purposes, but at the cost of a more expensive preprocessing step. Additionally, the post-processing step of conversing the aggregated graph back to the original hypergraph is more complicated.

(1)Aggregate-collapse: Each hyperedge h_i is transformed into an aggregation vertex $a(h_i)$, which aggregates all vertices related to h_i. The original vertices in the hypergraph are removed and replaced with aggregation vertices. Each vertex that is stored in an aggregation vertex becomes a member of that aggregation. Therefore, $I(h_i)$ is the set of members of the aggregation vertex $a(h_i)$ and corresponds to the vertices related to h_i. Then, if there is a shared member vertex between two aggregation vertices $a(h_i)$ and $a(h_j)$, i.e., $I(h_i) \cap I(h_j) \neq \varnothing$, then an edge is added between $a(h_i)$ and $a(h_j)$. This method allows for the same vertex to be a member of multiple aggregation vertices. In the example below, the aggregation vertices ABD and CDE are connected by an edge because they share vertex D.

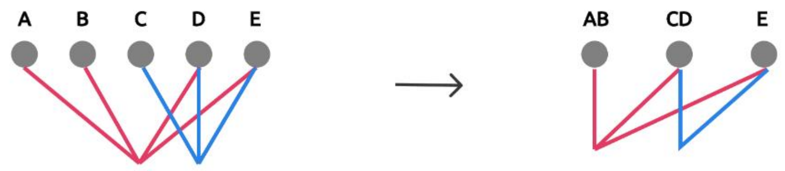
$$E = \left\{ \begin{matrix} e : a(h_i), a(h_j) \in I(c), \\ I(h_i) \cap I(h_j) \neq 0, \\ \forall h_i, h_j \in H \end{matrix} \right\} \tag{7}$$

$$V_1 = \{a(h_i), \forall h_i \in H\} \tag{8}$$



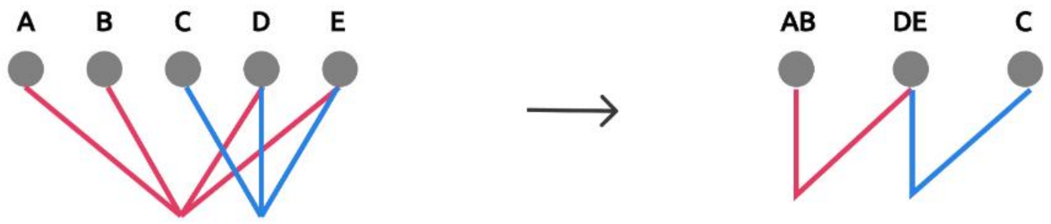
Discussion: Aggregate-collapse replaces all the vertices in the original hypergraph. The size of the conversed graph depends on the connectivity of the hypergraph: a highly connected hypergraph might introduce a large number of new vertices and edges, while a sparsely connected hypergraph might effectively reduce the number of elements and result in a much smaller graph compared to the initial one.

(2)Aggregate-summarize: This method uses the graph summarization algorithm to aggregate vertices based on the amount of graph-structural information lost when aggregating two vertices. This reduces the number of vertices that need to be sorted and eliminates hyperedges. This aggregation technique does not allow for vertices to be members of multiple aggregation vertices, unlike Aggregate-collapse, and the conversed graph is always smaller.



Discussion: Here, [1] can be replaced with different algorithms with the same purpose, which will give different results and different output graph sizes. The complexity of the transformation stated in Table 4 corresponds to the complexity of [1].

(3)Aggregate-relationship-summarize: By observing the relationships between vertices, vertices with the same relationships are aggregated, and edges are drawn between them according to hyperedges. This also leads to the fact that the number of vertices being aggregated in this method is not necessarily two, it can be multiple. In the following example, since A is related to B and C is related to E, they are separately aggregated into AB and CE, and because there is a hyperedge between A, B, and D, there is an edge connecting the new vertex AB and D.



Discussion: Compared to the previous method, Aggregate-relationship-summarize adds a condition filtering when aggregating vertices, which leads to a smaller conversed graph.

Post-processing for aggregate methods: In the aggregation method, each vertex represented is stored in an aggregated vertex a_i. After the layout calculation, these aggregated vertices need to be removed. Then we traverse each point in each vertex in turn. If the point no longer appears in other vertices, we record its degree as its weight. If it appears, we wait until we reach the last vertex where it appears before recording its degree. Finally, we sort the degrees of all points in this vertex, and then unpack this aggregated vertex. The next step is to unpack the next vertex until there are no more aggregated vertices. Finally, replacing the newly added edges with hyperedges is enough. In any case, there will be collisions in the positions of the vertices: multiple vertices may eventually have the same weight. This problem can be solved in a post-processing step (code in the appendix on https://github.com/Hypergraph-to-graph/Hypergraph-to-graph): we collect vertices with the same weight and then test each permutation of the vertex set to find the one that produces the least number of crossings. The cost of this process depends on the number of vertices with the same weight (Table 4).

3. Centroid methods

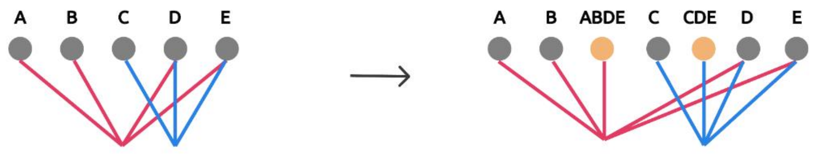
If we consider hyperedges as a specific type of vertex that connects original vertices in the hypergraph, this type of graph can be seen as another representation of the hypergraph.

(1)Centroid-single: Hyperedges are seen as individual vertices, with edges linking the vertices they are associated with. In the example below, ABC is a hyperedge, and as such, is treated as a special vertex. Because this hyperedge connects vertices A, B, and C, the special vertex ABC is linked with edges to vertices A, B, and C.

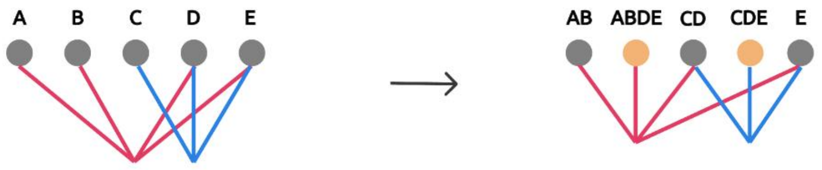
In the Centroid-single method, we create a centroid vertex c(h_i) for each hyperedge $h_i \in H$ and then replace h_i with edges that link each vertex associated with h_i to the newly created centroid vertex c(h_i).

$$E = \{e: c(h_i), \forall n_j \in I(h_i), \forall h_i \in H\} \tag{9}$$

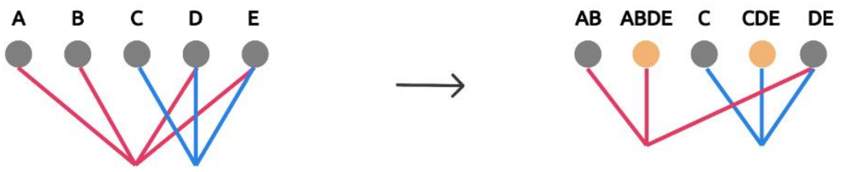
$$V_1 = V \cup \{c(h_i), \forall h_i \in H\} \tag{10}$$



(2)Centroid-aggregate-summarize: Unlike Centroid-single, this method aggregates two vertices in sequence to form a graph that is the same as the Aggregate-summarize method in Section 2.2.2, and then draws hyperedges as special vertices on the graph. At the same time, an edge is drawn to link the vertex representing the hyperedge with the vertices containing its related vertices. In the example below, as hyperedge ABC is associated with vertices A, B, and D, and the vertices AB and CD respectively contain A, B, and D, vertex ABD is linked by edges to vertices AB and CD.



(3)Centroid-aggregate-relationship-summarize: Unlike Centroid-aggregate-summarize, this method strictly specifies which vertices can be combined based on the relationships between them. After forming the same graph as the Aggregate-relationship-summarize method, hyperedges are drawn as special vertices on the graph. Additionally, an edge is drawn to link the vertex representing the hyperedge with the vertices containing its related vertices. In the example below, since hyperedge ABD is related to vertices A, B, and D, and the vertices AB and D respectively contain A, B, and D, vertex ABD is linked by edges to vertices AB and D.



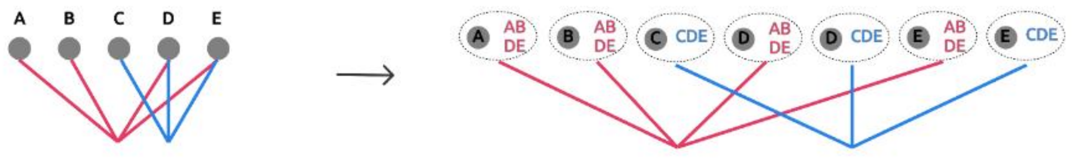
Post-processing for centroid methods: As with split methods, we keep the order of the vertices the same as obtained through the layout algorithm, then remove the centroids, and replace the newly introduced edges with their respective hyperedges.

4. Expansion methods

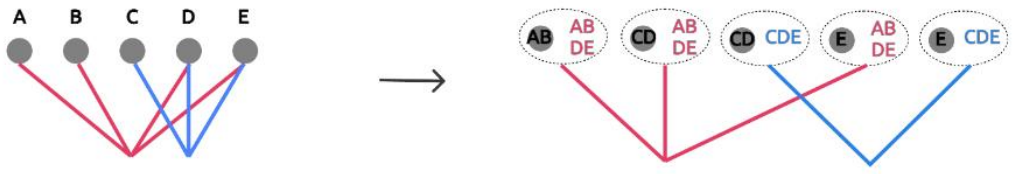
(1)Line-expansion: In this method, we treat the association between vertices and hyperedges as a whole, and define that two line nodes are neighbors if they contain the same vertices (vertex similarity) or the same hyperedges (edge similarity). Therefore, the expansion of the lines preserves high-order associations.

$$E = \{e : V_i, V_{i+1} \in I(e) \quad \forall V_i, V_{i+1} \in V_l,\ \forall v_i, v_{i+1} \in I(h_k),\ \forall h_k \in H\} \qquad (11)$$

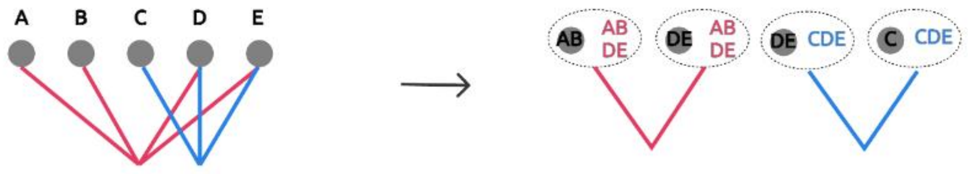
$$V_l = \{(v_i, h_i) | \forall v_i \in I(h_i),\ \forall h_i \in H\} \qquad (12)$$



(2)Line-expansion-aggregate-summarize: Compared to Line-expansion, this method aggregates two vertices in sequence to form a graph that is the same as the Aggregate-summarize method in Section 2.2.2, and then treats the association between vertices and hyperedges as a whole for line expansion.



(3)Line-expansion-aggregate-relationship-summarize: Compared to Line-expansion-aggregate-summarize, this method aggregates multiple vertices based on specific rules (combining vertices when their relationships are the same), forming a graph that is the same as the Aggregate-relationship-summarize method in Section 2.2.2, and then treats the association between vertices and hyperedges as a whole for line expansion.



Post-processing for centroid methods: We first undo the line expansion by replacing each point in the "vertex-hyperedge" form with just the "vertex" form. Then, for each point that appears in multiple positions, we take the average of those positions to get a new coordinate. Finally, we replace the newly-introduced edges with their respective hyperedges.

References

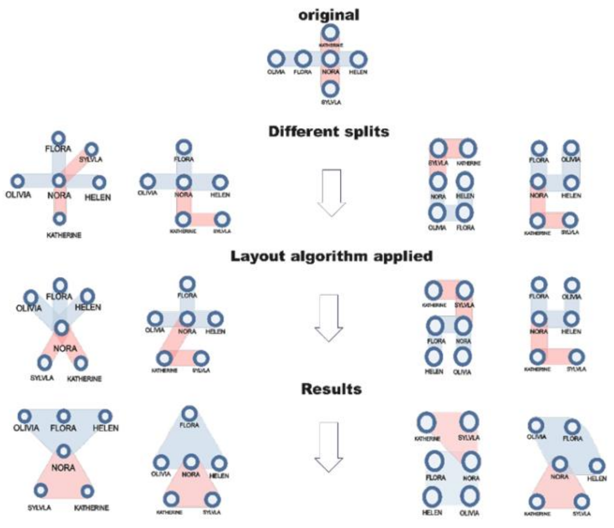
[1] García-Soriano, D., Riondato, M., Bonchi, F.: Graph summarization with quality guarantees. vol. 2015 (12 2014). https://doi.org/10.1109/ICDM.2014.56

[2] Young, J., Petri, G., Peixoto, T.: Hypergraph reconstruction from network data. Communications Physics 4(1) (Jun 2021). https://doi.org/10.1038/s42005-021-00637-w, funding Information: This work was funded, in part, by the James S. McDonnell Foundation (J.-G.Y.), the Sanpaolo Innovation Center (G.P.), and the Compagnia San Paolo via the ADnD project (G.P.).

Appendix D. The discussion of how to choose edges and vertices in conversion methods

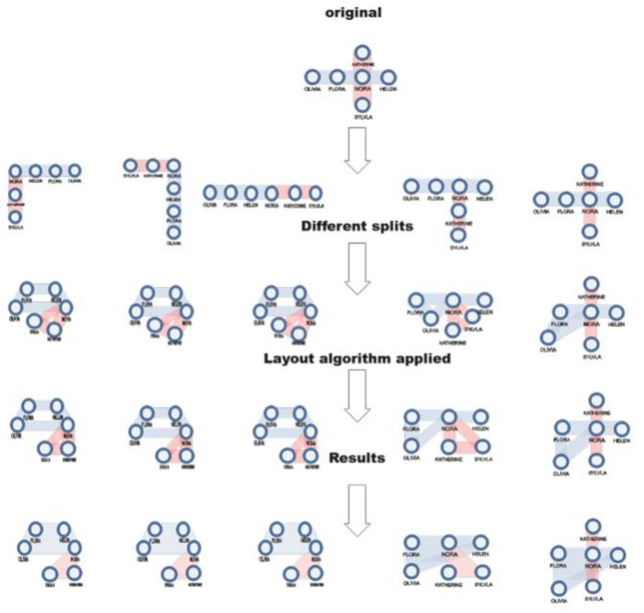
1. The order of vertices when using Split-path

The order of vertices of Split-path in our study has a significant impact on the final results, as depicted in the figure. We explored various splitting approaches falling into different categories to determine the most favorable one. Our investigation revealed that starting the path from the vertex with the highest degree or from other edges yielded the most remarkable outcomes. This approach resulted in newly split edges tending to center around vertices with higher degrees, thereby enhancing the overall layout's symmetry. Generally, the sequence of splitting paths has a minimal disruptive effect on the final result.



2. The order of vertices in Split-cycle

In the context of Split-cycle, the order of vertices with different degrees plays a crucial role, as illustrated in the figure. It has been demonstrated that when the vertex with the highest degree is either at the beginning or the end of the hyperedge during sorting, the final results remain the same. However, if the vertex with the highest degree falls in the middle of hyperedges, the results are less optimal.



Appendix E. Results and analysis of the Southern Women dataset and the DBLP dataset

1. Experiment 1

For the Southern Women dataset, the average completion time is 178,648ms, with 33.3% of users considering the Line-expansion method to have the lowest concavity. The shorter completion time and higher accuracy indicate that concavity has a significant impact on the hypergraph generated for small datasets.

For the DBLP dataset, the average completion time is 215,918ms, with 38.9% of users considering the Aggregate-relationship-summarize method to have the lowest concavity. 22.2% of users chose Split- clique and 11.1% of users found the Centroid-aggregate-relationship-summarize method to perform the best. The longer completion time and low accuracy of only 11.1% suggest that concavity has a minor impact on the hypergraph generated for medium-sized datasets, making it difficult for users to observe.

2. Experiment 2

For the Southern Women dataset, the average completion time is 140,637 ms, with 27.8% of users considering the Centroid-single method to have the lowest planarity. Only 16.7% of users' choices align with the actual results. The longer completion time and low accuracy of 16.7% suggest that planarity has a minor impact on the hypergraph generated for small datasets, making it difficult for users to observe.

For the DBLP dataset, the average completion time is 101,771 ms, with 38.9% of users considering the Centroid-single method to have the lowest planarity. The shorter completion time and higher accuracy indicate that planarity has a significant impact on the hypergraph generated for medium-sized datasets.

3. Experiment 3

For the Southern Women dataset, the average completion time is 103,034 ms, with 27.8% of users considering the Centroid-aggregate-summarize method to have the highest coverage. Only 5.6% of users' choices align with the actual results. The longer completion time and low accuracy of 5.6% indicate that coverage has a minor impact on the hypergraph generated for small datasets, making it difficult for users to observe with a significant deviation.

For the DBLP dataset, the average completion time is 62,888 ms, with 27.8% of users considering the Centroid-single method to have the highest coverage. The shorter completion time and higher accuracy suggest that coverage has a significant impact on the hypergraph generated for medium-sized datasets.

4. Experiment 4

For the Southern Women dataset, the average completion time is 124,874 ms, with 22.2% of users considering the Line-expansion method to have the highest regularity. Only 11.1% of users' choices align with the actual results. The longer completion time and low accuracy of 11.1% indicate that regularity has a minor impact on the hypergraph generated for small datasets, making it difficult for users to observe with a significant deviation.

For the DBLP dataset, the average completion time is 67,763 ms, with 44.4% of users considering the Aggregate-relationship-summarize method to have the highest regularity. The shorter completion time and high accuracy suggest that regularity has a significant impact on the hypergraph generated for medium-sized datasets.