How hypergraph-to-graph conversion affects cooperative working visualization: A Multi-metric evaluation

Anonymous authors

Abstract. It is commonplace to use hypergraphs to represent cooperative work, since hypergraphs explicitly capture complex interactions and connections, enabling researchers to analyze with ease. Nonetheless, hypergraphs are usually chaotic due to sophisticated relations between vertices. It is vital to find out a decent layout for a hypergraph in order to mitigate the problem. The layout of a hypergraph is usually done by converting the hypergraph into a simple graph and then using the layout algorithm of a normal graph to complete it. A plethora of conversion methods exist; nonetheless, they have drawbacks depending on the context, which may affect the interpretation of the hypergraph. Therefore, it is necessary to investigate which method prevails over other methods in specific circumstances. In our study, we propose an appraisal framework in which we use qualitative and quantitative metrics including 1) concavity, 2) planarity, 3) coverage rate of the plane, 4) regularity, and 5) cognitive capacity of data to assess the performance of each conversion method in terms of layout quality and effectiveness. The results show that while no method is ideal for all situations, certain methods, such as Centroid-single, perform well in general. Researchers can use our experiment results to select the optimal method tailored to their specific dataset and circumstances. This paper serves as researchers and practitioners choose the most suitable conversion method for their research.

Keywords: Hypergraph evaluation · Hypergraph conversion · Cooperative working · Multi metric.

1 Introduction

Cooperation involves multiple individuals or organizations working together towards a shared goal. Visualization plays a vital role in enhancing cooperation by providing a clearer understanding of complex relationships and processes. Hypergraphs, as a high-dimensional visualization technique, excel in representing data with intricate relationships. They map high-dimensional data to two or three dimensions, allowing for intuitive comprehension of data characteristics and relationships. Unlike regular graphs, where edges connect only two vertices, hypergraphs can connect multiple vertices with each hyperedge.According to research, in addition to the field of cooperative relationships, hypergraphs have practical applications in various fields such as biochemistry, circuit representation [14], databases [17], and machine learning [13]. However, as cooperative work scales up, challenges like visual clutter and limited comprehensibility in hypergraphs become more pronounced. Finding optimal visualization solutions for hypergraphs remains an ongoing endeavor.

Table 1. 12 Conversion Methods (Categorized in 4 Types)

Split methods	Aggregate methods	Centroid methods	Expansion methods
Split-clique	Aggregate-collapse	Centroid-single	Line-expansion
Split-path	Aggregate-summarize	Centroid-aggregate-summarize	Line-expansion-aggregate-summarize
Split-cycle	Aggregate-relationship-summarize	Centroid-aggregate-relationship-summarize	Line-expansion-aggregate-relationship-summarize

To overcome the visual clutter issue in hypergraphs, the hypergraph's layout has been well studied in visualization [22,29] . The majority of proposed hypergraph layout techniques rely on conversing the hypergraph into a graph [1], and in this paper, the term "conversion" is used to refer to the process of creating a graph $G1 = (V_1, E)$ from a hypergraph G = (V, H) . **Figure 1** illustrates how this conversion optimizes the process of hypergraph visualization. Based on the evaluation metrics we defined, subjective experiments were conducted to explore the merits and drawbacks of each conversion method. This paper focuses on exploring each conversion method, and the impact of each method on the performance and quality of the final hypergraph visualization. We analyzed twelve methods as listed in Table 1.

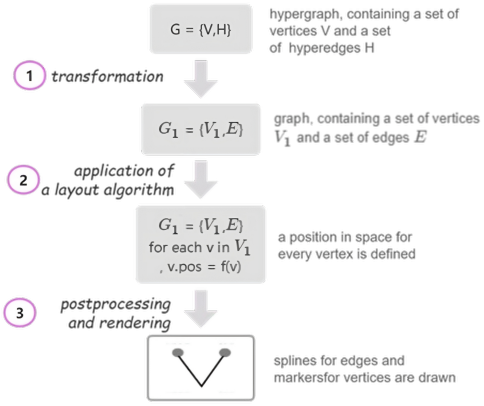


Fig. 1. Steps to be taken from hypergraph definition to drawing

This paper introduces three datasets that exhibit different types and levels of visual clutter. Prior to the main study, a pre-study involving 40 participants was conducted to establish the color stimuli and layout algorithm. The formal study involved 18 participants and encompassed five experiments aimed at examining the impact of conversion methods on hypergraph visualization. Additionally, a subjective questionnaire was devised to capture participants' subjective experiences. The code and documentation resources can be accessed at https://github.com/Hypergraph-to-graph/Hypergraph-to-graph, referred to as the 'appendix' in subsequent sections.

2 Related work

2.1 visualization of hypergraphs

According to our research, various visualization methods for hypergraphs have been proposed. The most commonly used methods are two-fold: one is based on subsets, for example, Kritz and Perlin [19]proposed the QUAD scheme that resembled a matrix encoding of set relations: each hyperedge is a column represented by a rectangle and each vertex is a point along a particular row. Riche and Dwyer [23] attempted to improve the readability of the set intersections based on untangling Euler diagrams, by using compact rectangular shapes or duplicating set elements; the other is node-link-based approach, such as the MetroSets method proposed by Jacobsen et al [14]. for representing geographic hypergraph data, which has a forced rectilinear layout and inevitable crossings that can lead

to confusion; the force-directed 3D layout with spherical and directed links proposed by Kapec et al. [16], which uses color to distinguish callers and called parties;and the radial layout proposed by Kerren et al. [18], in which hyperedges are circular dotted lines around the central node, eliminating overlaps and slightly increasing scalability. In addition, there are other methods such as timeline-based approach (using PAOHvis V aldivia et al.'s [27] ordered timeline layout for hyperedges as an example, which provides rich filtering and interactive controls), and matrix-based approach (using Streeb et al.'s [24] prototype for time hypergraph analysis using a symbol-based matrix view as an example, but it may require interaction to retrieve information, and its scalability is limited).

When visualizing hypergraphs, our emphasis is on differentiating data classifications. Therefore, the discussion and examples in this pape primarily employ Euler diagrams based on the subset approach. Utilizing Euler diagrams to represent hypergraphs offers several advantages. Firstly, Euler diagrams are visually appealing and facilitate easier comprehension and visualization of hypergraphs. Secondly, they provide enhanced spatial efficiency, particularly for larger hypergraphs, as the memory requirements for Euler diagram edges, resembling undirected edges, are relatively minimal. In summary, Euler diagrams serve as an effective, visual, and comprehensible framework for the analysis and exploration of hypergraphs.

2.2 Evaluation Studies of Methods for Hypergraphs' Conversion

There is a scarcity of evaluation papers specifically focused on hypergraphs, particularly those pertaining to conversion methods. However, we came across a paper that presented six conversion methods and carried out a comprehensive evaluation. [19] .

In our paper, we extended seven conversion methods based on it. To eliminate the difference between objective and subjective cognition, we invited participants to perform five experiments designed by us, corresponding to five evaluation metrics. After the experiments, we also conducted a survey of users (see the appendix) to analyze the answers obtained and the results of the experiments.

3 EVALUATION LANDSCAPE

3.1 Selection of Conversion Methods

In order to provide a comprehensive overview of conversion methods employed in the visualization community, we conducted a thorough survey of papers pub-

lished in the Journal of IEEE TVCG, as well as three prominent visualization conferences (IEEE VIS, PacificVis, and EuroVis) over the past decade. Utilizing Google Scholar, we conducted keyword searches for "hypergraph", "hypergraph transform" and "hypergraph conversion" within the aforementioned sources. Additionally, we utilized ChatGPT to inquire about papers specifically related to hypergraph conversion. Initially, we obtained a total of 997 papers. After manually filtering out irrelevant papers, we identified 5 papers that were directly relevant, allowing us to further analyze and summarize the conversion methods presented within them.

Table 2. References, features, worst-case complexities of conversion, and post-processing of our conversion methods. VC refers to the conversion in which the vertices are changed after the conversion, |I| indicates the maximum number of vertices incident to a single hyperedge. |H| indicates the number of hyperedges. |V| indicates the number of vertices.

Conversion method	References	VC	Converse	Post-processing
Split-clique	[3]		$O(\|H\| * \|I\|^2)$	$O(\|H\| * \|I\|)$
Split-path	[3]		$O(\|H\| * \|I\|)$	$O(\|H\| * \|I\|)$
Split-cycle	[1], [2]		$O(\|H\| * \|I\|)$	$O(\|H\| * \|I\|)$
Aggregate-collapse	[3]	✓	$(\|H\|^2 * \|I\|^2)$	$O(\|H\|^2 * \|I\|^2)$
Aggregate-summarize	[3]	✓	$O(\|H\|^2 * \|I\|^2)$	$O(\|V\|^2)$
Aggregate-relationship-summarize	[2], [5]	✓	$O(\|H\| * \|I\| * \|V\|^2)$	$O(\|V\|^2)$
Centroid-single	[3]	✓	$O(\|H\| * \|I\|)$	$O(\|H\| * \|I\|)$
Centroid-aggregate-summarize	[3]	✓	$O(\|H\|^2 * \|I\|^2)$	$O(\|H\| * \|I\| * \|V\|)$
Centroid-aggregate-relationship-summarize	[3], [2], [5]	✓	$O(\|H\|^2 * \|I\|^2)$	$O(\|H\| * \|I\| * \|V\|)$
Line-expansion	[28]	✓	$O(\|H\|^2 * \|I\|^2)$	$O(\|H\| * \|I\| * \|V\|)$
Line-expansion-aggregate-summarize	[3], [28]	✓	$O(\|H\|^2 * \|I\|^2)$	$O(\|H\| * \|I\| * \|V\|)$
Line-expansion-aggregate-relationship-summarize	[28], [2] , [5]	✓	$O(\|H\| * \|I\| * \|V\|^2)$	$O(\|V\|^2)$

As relatively few methods have been proposed in this field, we have developed and tested several new methods in addition to the original methods as listed in Table 2. The specific definitions, formulas, diagrams, and post-processing steps for each conversion method are included in the appendix.

3.2 Selection of Datasets

To ensure the reliability of the evaluation results, we collected datasets that were used in the works in our survey, and therefore, the obtained datasets were suitable for generating hypergraphs directly. Observing these hypergraphs, we found that the number of vertices and edges, the distribution of density, and the number of subgraphs varied widely. Furthermore, to explore the application of cooperative work in hypergraphs, we selected three representative datasets

with different features from the cooperative relationships domain, as shown in **Figure 2**: Southern Women, which recorded the social interactions of a group of 18 women during 14 informal social activities over 9 months. The dataset contains a total of 18 vertices and 14 hyperedges. [7] ; As for DBLP, we selected papers published on IEEE VIS from the DBLP dataset from 1990 to 2018, and filtered out papers that contained the keyword "visualization". We sorted the papers based on PubsCited, and selected the top 15 papers. In order to make the produced data have the characteristics of a hypergraph, we added other papers in the same field published by the authors corresponding to these 15 papers to the dataset. In the end, there are a total of 68 vertices and 30 hyperedges in the dataset. [21] ; and Coauthor Network, which explored the cooperation network of authors in academic publications. We use the topic-coauthor dataset from ArnetMiner [25] and focus specifically on authors in the information-retrieval research field and an H-index of 40 or higher. The dataset contains a total of 114 vertices and 120 hyperedges.

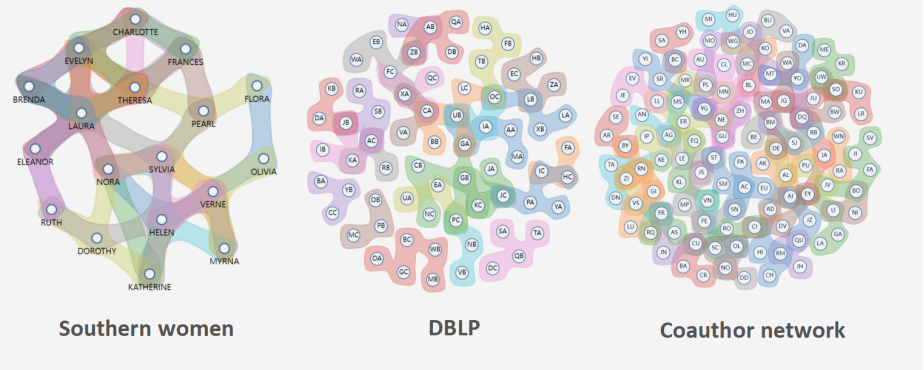


Fig. 2. Datasets select for our evaluation. Here are the output hypergraphs of three datasets using the Centroid-single method

3.3 Selection of Evaluation Metrics

Through comprehensive research, we defined evaluation metrics from both quantitative and qualitative aspects.

For the quantitative metrics, in addition to the time and accuracy of completing trials, we introduced five metrics with reference to literature: (1) concavity, which refers to the number of nonconvex hypergraph edges drawn, the smaller the concavity, the more aesthetically pleasing the hypergraph is [1] ; (2)

planarity, which refers to the number of non-adjacent hypergraph edges intersections, the smaller the planarity, the more helpful it is in avoiding clutter, thus avoiding ambiguity in the relationships represented by the hypergraph [1], [20] ; (3) coverage rate of the plane, whose formula is in the appendix. To control for variables, the canvas size was standardized to 860 x 860 pixels. The larger the ratio, the more appropriately the drawing canvas is utilized [1] ; (4) regularity, which refers to the evenness of the distance between the connections of vertices, is quantified by the variance of the lengths of the edges connecting vertices. The larger the evenness, the more uniform the distribution of vertices [1] , [20]; (5) cognitive capacity of data, to quantify this metric, we used questions related to data categorization. These metrics often appear under different names in papers evaluating graph layouts [9]. **Figure 3** illustrates the impact of each evaluated metric on hypergraph visualization very well.

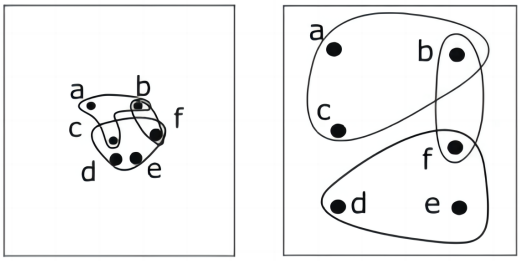


Fig. 3. Two drawings of hyperedgesa,b,c,b,f,d,e,f.The drawing on the left has a concave shape, poor Coverage and non-uniform distribution of vertices(clutter), The drawing on the right is aesthetically superior to the one on the left as it has no concavity, less crossings, comparatively better Coverage and Regularity

In terms of qualitative metrics, we used a questionnaire to ask users questions related to the following five metrics: (1) difficulty rating of each experimental trial; (2) preference rating of 12 output hypergraphs; (3) self-assessment of trial completion speed; (4) self-assessment of trial completion accuracy; (5) visual background. The specific questionnaire is included in the appendix. Therefore, in the study, we will evaluate the conversion methods from both six quantitative and five qualitative metrics in order to find the different applicable scenarios for each conversion method. The metric system for evaluation is in **Figure 4**.

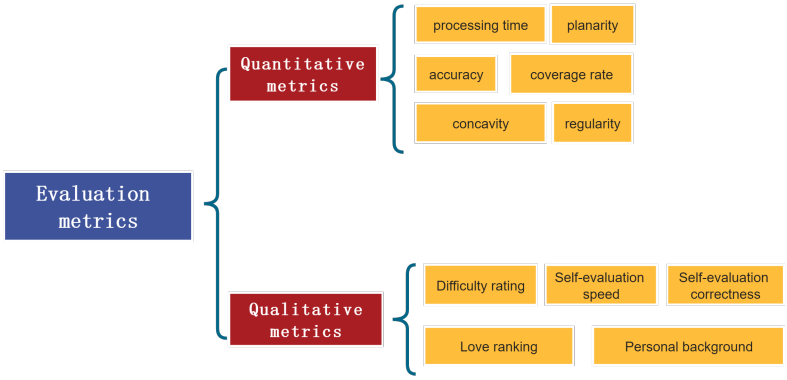


Fig. 4. The metric system for evaluation.

4 PRE-STUDY

4.1 Experiment 1: Understanding Layout Algorithm Effect on Visualization

The purpose is to specify a choice for the formal study: which layout algorithm should be used for the data layout of graphs in the formal study. Several layout algorithms have been proposed for common graphs in previous research, and we selected four algorithms from the most commonly used force-directed layout algorithms: forceAltas2 [15],Fruchterman [6], [8], [1], CompoundSpringEmbedderGraph(CoSE) [4], [12], [11], and a force algorithm based on prefuse(force) [10], [26], [30].

Trial and Procedure. We used three datasets with the same number of vertices as the formal study dataset, but with different content. For each dataset, we used four layout algorithms. In each of the three experiments, four graphs representing a layout algorithm were shown to participants. They were asked to rate the visual effectiveness of each layout algorithm using Likert's 5-point scale, with two perspectives for evaluation: 1) the degree of aggregation of points within the same category, and 2) the degree of dispersion of points between different categories. Each participant takes about 10 minutes to complete the experiment.

Participants and Apparatus. We recruited 40 participants (28 males, 12 females, aged 11-30) for the experiment. All participants had a computer science background. 25 participants reported being very familiar with visualization, 10 were moderately familiar, and the remaining 5 were unfamiliar. The experiment was conducted through a web prototype(see appendix), and participants

were asked to conduct the experiment on a screen with a resolution of 1,920 x 1,080. **Results.** Considering that each layout algorithm performs differently on datasets of different sizes, we chose the layout algorithm with the highest average score, which was Fruchterman. As shown in **Figure 5**, the average score of Fruchterman was significantly higher than the other three algorithms, and it performed well on both small and medium-sized graphs. One participant said, *"The graph generated by Fruchterman is more helpful for me to distinguish different classes."* Therefore, we decided to use the Fruchterman layout algorithm in the formal study to optimize the visualization effects.

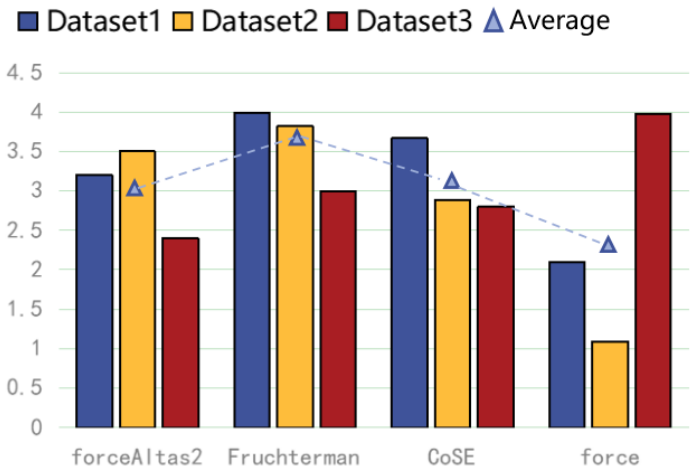


Fig. 5. Results of pre-study.

5 FORMAL STUDY

5.1 Hypotheses

Formal study aims to evaluate the performance of 12 conversion methods based on whether they can optimize the following five visual metrics: concavity, planarity, the coverage rate of the plane, regularity, and the cognitive capacity of data. Thus, five hypotheses were formulated to guide the experimental design.
H1: In terms of reducing concavity, the user perception of the Centroid-aggregate-relationship-summarize method outperforms other conversion methods.

H2: In terms of improving planarity, the user perception of Split-path and Split-cycle methods perform better.
H3: In terms of improving coverage, the user perception of Line-expansion method performs better compared to other methods.
H4: In terms of improving regularity, the user perception of the performance of other conversion methods is worse than that of the Line-expansion-aggregate-relationship-summarize method.
H5: In terms of improving the cognitive capacity of data, users experience the shortest completion time and the highest accuracy in the Line-expansion and Centroid-aggregate-relationship-summarize methods.

5.2 Experiments

Guided by the five hypotheses (H1-H5), we designed five experiments (E1-E5). Note that E1-E5 were controlled experiments.
E1: Perception of concavity reduction. This experiment is used to evaluate the ability of different conversion methods to reduce concavity in terms of visual perception. Participants were asked to rank 12 graphs in order of decreasing concavity, i.e., in order of decreasing number of non-convex superedges.

For three datasets, one question, coupled with 12 graphs corresponding to 12 conversion methods, is generated. Overall, 3 trials have been prepared for each participant.
E2: Perception of planarity increases. This experiment is aimed to evaluate the ability of different conversion methods to improve planarity in terms of visual perception. Participants were asked to rank 12 graphs in order of declining planarity, i.e., in order of decreasing number of superedge crossings. Similar to E1, one question and 12 graphs for each dataset were generated for each question. In total, there are 3 (datasets) x 1 (question) = 3 trials for each participant.
E3: Perception of the coverage rate of the plane increase. This experiment is designed to evaluate the ability of different conversion methods to improve coverage in terms of visual perception. Participants were asked to rank 12 graphs in order of decreasing coverage rate. Similarly, one question for each dataset was created, and 12 graphs for each question. In total, it is 3 (datasets) x 1 (question) = 3 trials waited for participants to solve .
E4: Perception of regularity increases. This experiment is designed to evaluate the ability of different conversion methods in improving the uniformity of distance between vertex connections. Participants were asked to complete a trial: rank the 12 graphs in order of increasing evenness of vertice distances. one question was generated for each dataset and 12 graphs for each question. In total, 3

(datasets) x 1(question) = 3 trials wait for each participant to answer.

E5: Perception of cognitive capacity of data improvement. This experiment is designed to evaluate the ability of different conversion methods to enhance users' cognitive abilities in terms of data categorization in visual perception. Participants are asked to answer questions related to data categorization based on hypergraphs. In each experiment, one question was designed for the output hypergraph of each conversion method. In general, 12 (conversion methods) x 3 (datasets) x 1 (question) = 36 trials have been prepared for participants.

5.3 Participants, Apparatus and Testing Data

Participants. 18 participants(14 males, and 4 females, aged 11-30) engaged in our formal study. All participants have a background related to computers. They consist of graduate students from different years and undergraduate students from different years. Among those participants, 10 reported being familiar with visualization, 5 were moderately familiar, and the remaining 3 were unfamiliar. Each participant possessed either normal vision or achieved normal vision through correction, with no reported instances of color blindness or color deficiency among them.

Apparatus. The experiments were conducted online through a web prototype (Figure 6). The test website URL is http://47.113.196.28/. Participants were required to visit it remotely on the Chrome browser and finish the experiment on a screen with a resolution of 1,536 x 864. They were asked to share their screen with the instructor during the experiments to enable remote monitoring.

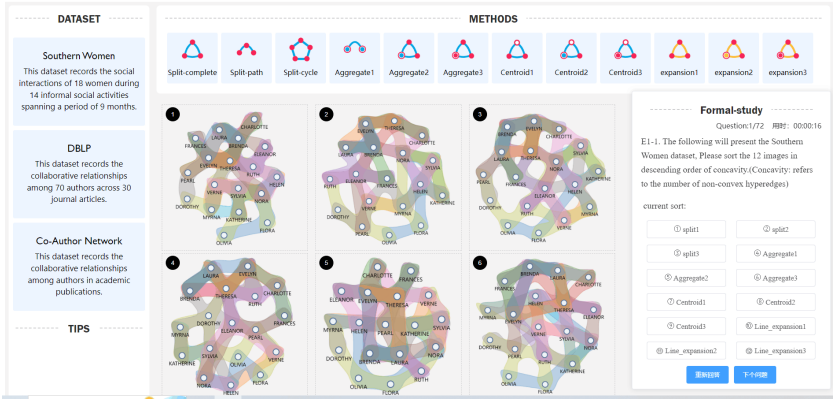


Fig. 6. Formal Study Website

5.4 Procedure

Each experiment consisted of a training phase and an actual testing phase. At the beginning of the training course, the experiments and related concepts (such as concavity) were explained. Participants were encouraged to ask questions during the training course to facilitate their understanding of the experiment. Once they reported that they fully understood the trials, the actual testing phase would start. Participants were allowed a 5-minute break before each experiment. Before starting the experiment, they were asked to provide background and basic information. After completing the experiment, the participants were asked to rate the importance of several evaluation metrics using the Likert 5-point scale and provide other visual metrics they deemed important. The whole process lasted for about one hour for each participant.

6 EXPERIMENTAL RESULTS

6.1 Analysis Approach

Objective measurements were recorded for E1-E5 and the accuracy and completion time for each trial were documented.

For the sorting problem in E1-E4, the user-selected conversion method that has the best performance in theory and the method that has performed best in real experiments were compared. If they are consistent, it could indicate that users subjectively perceived the significant impact of this metric on hypergraph visualization. In the event of inconsistency, it would imply a diminished influence of this metric, thereby allowing for a potential reduction in its significance based on other analysis findings. The importance of this metric through user-rated problem difficulty, self-completion speed, and self-completion accuracy was also analyzed. For the judgment problem in E5, meticulous calculations and analysis were conducted to determine the accuracy and completion time associated with each conversion method, with the aim of identifying a method that excels in both aspects, displaying superior performance. After completing the questionnaire, the results were comprehensively analyzed in conjunction with the participants' backgrounds, in order to discern the underlying relationship between them and gain a clearer understanding of the specific scenarios in which the conversion method would be most suitable and applicable.

6.2 Results Analysis

Results and analysis for the Southern Women dataset and DBLP dataset in E1-E4 are included in the appendix.

H1 is partially confirmed because the Centroid-aggregate-summarize method is the best conversion method for medium-sized datasets based on actual results, while users consider the Aggregate-relationship-summarize method to be the best. For small datasets, both the actual results and the user perception consider the Line-expansion method to be the best. For large datasets, the Aggregate-collapse method is the best according to actual results, while the users' perception is the Split-clique method. For completion time in E1, which is positively proportional to dataset size, completion time in E2-E4 is inversely proportional to dataset size. It is hypothesized that this phenomenon can be attributed to the fact that the indicators associated with the last three experiments become more readily observable when a larger volume of data is made available for analysis.

Figure 7 shows the average completion time for each dataset in **E1**. For the Coauthor Network dataset, the average completion time was 205,459ms, and 27.8% of users thought that the Split-clique method had the lowest concavity, 27.8% of users chose Line-expansion-aggregate-relationship-summarize method, and 16.7% of users thought that the Aggregate-collapse method performed the best. Not only was the completion time lengthy, but the accuracy was also disappointingly low, measuring only 16.7%. In addition, 38.9% of users rated their completion speed as moderate, and 44.4% of users rated their accuracy as low, indicating that the performance differences between different conversion methods in terms of concavity were not apparent, and users had difficulty observing the impact of this factor on hypergraph visualization. Therefore, the importance of concavity could be reduced.

H2 is partially confirmed because, for small datasets, the Split-path method performs the best, while the Centroid-single method is more commonly known among users; for medium-sized datasets, the Centroid-single method is both the best in actual results and user perception; for large datasets, the Aggregate-relationship-summarize method has the best actual results, while the Centroid-aggregate-relationship-summarize method is more commonly known among users.

Figure 7 shows the average completion time for each dataset in **E2**. For the Coauthor Network dataset, the average completion time was 83,202ms, and 33.3% of users thought that the Centroid-aggregate-relationship-summarize method had the lowest planarity, and 22.2% of users thought that the Aggregate-

relationship-summarize method performed the best. In addition, 38.9% of users rated their completion speed as slow, 38.9% of users rated their completion speed as fast, and the rest of the users rated their speed as moderate. Upon thorough investigation, a significant revelation emerged: participants with a research background in computer vision self-rated themselves as possessing a higher degree of speed, suggesting that this particular expertise might play a crucial role in effectively discerning planarity within hypergraphs.55.6% of users rated their accuracy as moderate, much higher than the actual average accuracy of 25.9% for the three datasets. Our initial suspicion arose from the observation that users exhibited a limited understanding of planarity, strongly indicating that this concept inherently possesses a relatively complex nature, making it challenging to comprehend.

H3 is rejected because for small datasets, the Aggregate-collapse method has the best actual results, but most users are familiar with the Centroid-aggregate-summarize method. For medium-sized datasets, both the Centroid-single method and user perception are the best. The results and user perception are best with the Split-path method for large datasets.

Figure 7 shows the average completion time for each dataset in **E3**. For the Coauthor Network dataset, the average completion time was 63,499ms, and 22.2% of users thought that the Split-path method had the highest coverage rate. In addition, 33.3% of users rated the experiment's difficulty as moderate, indicating that the experiment was relatively reliable. 44.4% of users rated their completion speed as fast, and 44.4% of users rated their accuracy as moderate, indicating that the performance differences between different conversion methods in terms of coverage rate were apparent.

H4 is rejected because for small datasets, the Centroid-single method has the best actual results, but most users are familiar with the Line-expansion method. For medium-sized datasets, both the actual results and user perception are the best with the Aggregate-relationship-summarize method. For large datasets, both the actual results and user perception are best with the Centroid-aggregate-relationship-summarize method.

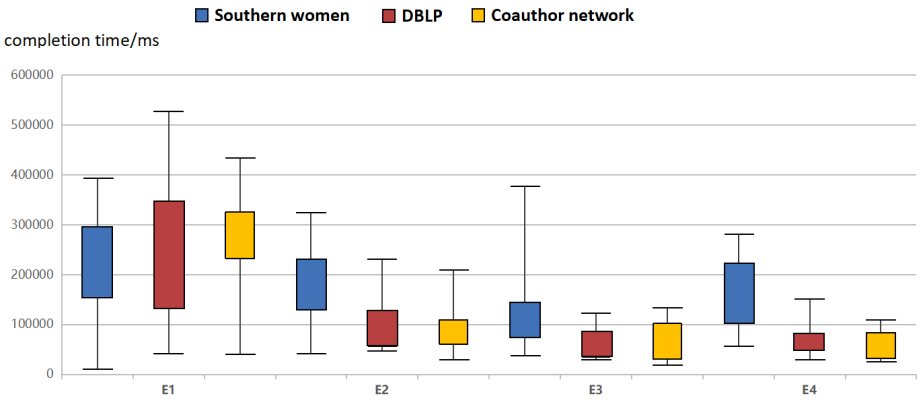


Fig. 7. The average completion time of the three datasets in Experiments 1 to 4.

Figure 7 shows the average completion time for each dataset in **E4**. For the Coauthor Network dataset, the average completion time was 59,953ms, and 27.8% of users thought that the Centroid-aggregate-relationship-summarize method had the greatest uniformity in vertice-to-vertice connection distance. In addition, 38.9% of users rated their completion speed as fast, and 33.3% of users rated their accuracy as moderate, indicating that the performance differences between different conversion methods in terms of regularity were apparent.

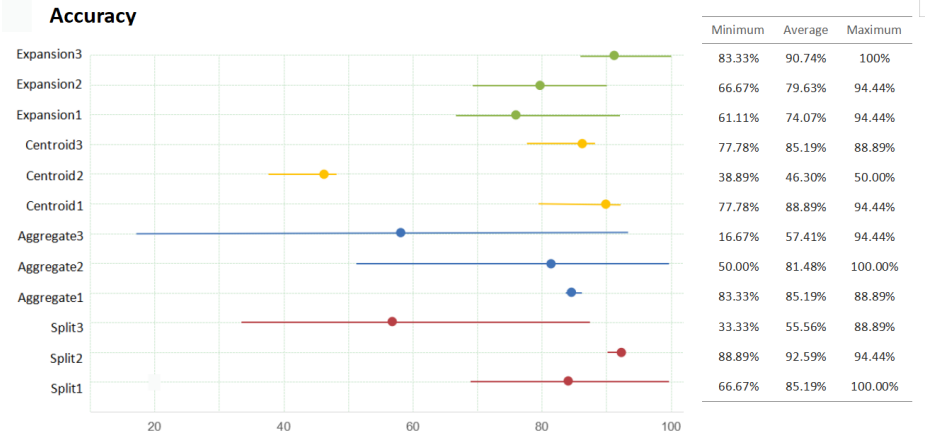


Fig. 8. Accuracy for 12 conversion methods

H5 is rejected because Split-path method has the highest accuracy and Aggregate-relationship-summarize method has the shortest time.**Figure 8, 9** shows the results for each conversion method in E5.

Split-path method has the highest accuracy (92.59%). Aggregate-relationship-summarize method has the shortest time (13,954ms) but a low accuracy (57.41%). Aggregate-relationship-summarize(57.41%), Split-cycle(55.56%), and Centroid-aggregate-summarize(46.30%) methods perform significantly worse than others in terms of accuracy. Aggregate-relationship-summarize, Aggregate-collapse methods perform significantly better than others in terms of average completion time. Overall, Line-expansion-aggregate-relationship-summarize performed well in terms of accuracy (90.74%) and average completion time (16,485 ms).

Takeaways: When confronted with datasets of medium size, it is advisable to opt for Centroid-single as it exhibits strong performance in both actual results and user perception in E2 and E3, particularly when the emphasis lies on planarity or coverage rate. However, if the research focus centers more on regularity, a competitive option would be Aggregate-relationship-summarize. For larger datasets, selecting Split-path is recommended when researchers prioritize achieving higher coverage. Conversely, Centroid-aggregate-relationship-summarize presents a substantial advantage when the objective is to attain higher regularity. In terms of enhancing users' cognitive abilities for data classification, Line-expansion demonstrates commendable accuracy and completion time.

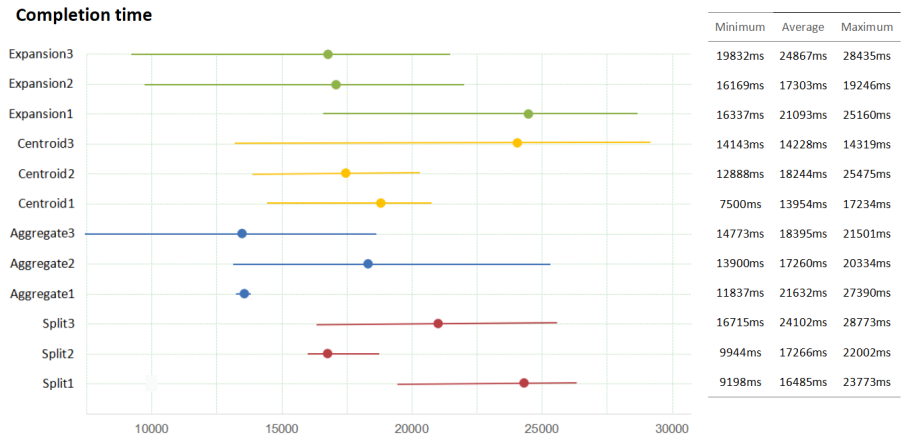


Fig. 9. Completion time for 12 conversion methods

6.3 Important Visual Metrics

The subjective questionnaire on important evaluation metrics provided insight into the choice of conversion methods in different scenarios. The calculation results are shown in **Figure *10*** , with average ratings for regularity, planarity, and concavity of 4.32, 4.1, and 4.03, respectively. The high ratings confirm that the evaluated metrics are generally important issues in hypergraph visualization.

Through analyzing the variations among participants with diverse backgrounds, intriguing insights emerged. Participants categorized as "Familiar" or more experienced in visualization gave a rating of 3.70 for regularity, while those less acquainted with visualization scored an average of 4.27. This discrepancy may stem from the inclination of individuals in their daily lives to prefer even distributions, whereas visualization researchers prioritize conveying information intuitively over visual aesthetics. Furthermore, participants with moderate to more familiarity with visualization demonstrated a heightened interest in enhancing their cognitive abilities for data classification compared to other participants. Moreover, divergent research directions among participants yielded disparate outcomes. Those with a focus on computer vision generally accorded less importance to concavity, in contrast to participants specializing in visualization or software engineering. This distinction might be attributed to the infrequent involvement of the concavity concept in the realm of artificial intelligence. In the interviews conducted after the experiment, participants stated that *"knowledge related to data structures is very helpful for hypergraph visual analysis."* In addition, participants proposed new evaluation metrics, such as color, the degree of cohesion within the same class, and the degree of dispersion between different classes. One participant said, *"During the experiment, some hyperedges were difficult to distinguish due to color issues."* Meanwhile, other participants also expressed that *"the overlap and intersection of hyperedges could lead to color overlap and visual misguidance."*

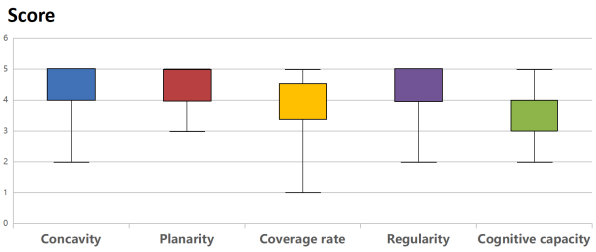


Fig. 10. The score of evaluation metrics.

7 Conclusion and Future Work

This paper proposes an empirical evaluation of hypergraph conversion methods from a perceptual perspective. After a comprehensive investigation, 12 methods, 7 quantitative metrics, and 5 qualitative metrics were identified. Based on this, we proposed five hypotheses and designed five experiments. Before the formal study, a pre-study was conducted to determine which layout algorithm to use after converting hypergraphs to graphs. The results of the formal study provide practical guidance for the selection of conversion methods in different application scenarios and also provide guiding suggestions for visualization methods in cooperative work.

Nonetheless, it is imperative to acknowledge certain limitations and propose potential avenues for improvement within this paper. Specifically, the consideration of the dataset's vertex and hyperedge quantities and their influence on hypergraph visualization remains unexplored. In addition, the issue of color overlap leading to user bias in data classification remains. Future endeavors should concentrate on delving into these aspects, encompassing discussions on how dataset variations impact each conversion method, as well as devising strategies to assign distinct colors to intersecting hyperedges. Furthermore, it is crucial to note that the dataset employed in this study is confined in scope, necessitating future research to expand the scale and diversity of datasets employed for comprehensive analysis.

References

1. Arafat, N.A., Bressan, S.: Hypergraph drawing by force-directed placement. pp. 387–394 (08 2017). https://doi.org/10.1007/978-3-319-64471-4_31
2. Bannister, M.J., Eppstein, D., Goodrich, M.T., Trott, L.: Force-directed graph drawing using social gravity and scaling. CoRR **abs/1209.0748** (2012), http://arxiv.org/abs/1209.0748
3. Di Bartolomeo, S., Pister, A., Buono, P., Plaisant, C., Dunne, C., Fekete, J.D.: Six methods for transforming layered hypergraphs to apply layered graph layout algorithms. Computer Graphics Forum **41** (06 2022). https://doi.org/10.1111/cgf.14538
4. Dogrusoz, U., Belviranli, M.E., Dilek, A.: Cise: A circular spring embedder layout algorithm. IEEE Transactions on Visualization and Computer Graphics **19**, 953–966 (2013)
5. Ducournau, A., Bretto, A., Rital, S., Laget, B.: A reductive approach to hypergraph clustering: An application to image segmentation. Pattern Recognition **45**, 2788–2803 (07 2012). https://doi.org/10.1016/j.patcog.2012.01.005

6. Frickey, T., Lupas, A.: Clans: A java application for visualizing protein families based on pairwise similarity. Bioinformatics (Oxford, England) **20**, 3702–4 (01 2005). https://doi.org/10.1093/bioinformatics/bth444
7. FULLER, C.: Caste, race, and hierarchy in the american south. Journal of the Royal Anthropological Institute **17**, 604 – 621 (08 2011). https://doi.org/10.1111/j.1467-9655.2011.01709.x
8. Gajdo, P., Ježowicz, T., Uher, V., Dohnalek, P.: A parallel fruchterman–reingold algorithm optimized for fast visualization of large graphs and swarms of data. Swarm and Evolutionary Computation **26** (08 2015). https://doi.org/10.1016/j.swevo.2015.07.006
9. Haleem, H., Wang, Y., Puri, A., Wadhwa, S., Qu, H.: Evaluating the readability of force directed graph layouts: A deep learning approach. CoRR **abs/1808.00703** (2018), http://arxiv.org/abs/1808.00703
10. Heer, J., Card, S., Landay, J.: Prefuse: A toolkit for interactive information visualization. pp. 421–430 (04 2005). https://doi.org/10.1145/1054972.1055031
11. Herman, I., Melançon, G., Marshall, M.: Graph visualization and navigation in information visualization: A survey. Visualization and Computer Graphics, IEEE Transactions on **6**, 24 – 43 (02 2000). https://doi.org/10.1109/2945.841119
12. Hu, Y.: Efficient and high quality force-directed graph drawing. Mathematica Journal **10**, 37–71 (01 2005)
13. Huang, J., Zhang, R., Yu, J.: Scalable hypergraph learning and processing. pp. 775–780 (11 2015). https://doi.org/10.1109/ICDM.2015.33
14. Jacobsen, B., Wallinger, M., Kobourov, S., Nöllenburg, M.: Metrosets: Visualizing sets as metro maps. IEEE Transactions on Visualization and Computer Graphics **PP**, 1–1 (10 2020). https://doi.org/10.1109/TVCG.2020.3030475
15. Jacomy, M., Venturini, T., Heymann, S., Bastian, M.: Forceatlas2, a continuous graph layout algorithm for handy network visualization designed for the gephi software. PloS one **9**, e98679 (06 2014). https://doi.org/10.1371/journal.pone.0098679
16. Kapec, P.: Hypergraph-based software visualization. International Workshop on Computer Graphics, Computer Vision and Mathematics, GraVisMa 2009 - Workshop Proceedings pp. 149–153 (01 2009)
17. Kenig, B., Weinberger, N.: Quantifying the loss of acyclic join dependencies (10 2022)
18. Kerren, A., Jusufi, I.: A novel radial visualization approach for undirected hypergraphs (01 2013). https://doi.org/10.2312/PE.EuroVisShort.EuroVisShort2013.025-029
19. Kritz, M., Perlin, K.: A new scheme for drawing hypergraphs. International Journal of Computer Mathematics **50**(3-4), 131–134 (Jan 1994). https://doi.org/10.1080/00207169408804250, copyright: Copyright 2015 Elsevier B.V., All rights reserved.

20. Kwon, O.H., Crnovrsanin, T., Ma, K.L.: What would a graph look like in this layout? a machine learning approach to large graph visualization. IEEE Transactions on Visualization and Computer Graphics **24**, 477–488 (01 2018). https://doi.org/10.1109/TVCG.2017.2743858
21. Ley, M.: Dblp.uni-trier.de: Computer science bibliography. http://dblp.uni-trier.de/ (1993), http://dblp.uni-trier.de/
22. Qu, B., Zhang, E., Zhang, Y.: Automatic polygon layout for primal-dual visualization of hypergraphs. CoRR **abs/2108.00671** (2021), https://arxiv.org/abs/2108.00671
23. Riche, N., Dwyer, T.: Untangling euler diagrams. IEEE Transactions on Visualization and Computer Graphics **16**(6), 1090 – 1099 (2010). https://doi.org/10.1109/TVCG.2010.210
24. Streeb, D., Arya, D., Keim, D.A., Worring, M.: Visual analytics framework for the assessment of temporal hypergraph prediction models (2019)
25. Tang, J., Zhang, J., Yao, L., Li, J., Zhang, l., Su, Z.: Arnetminer: Extraction and mining of academic social networks. pp. 990–998 (08 2008). https://doi.org/10.1145/1401890.1402008
26. Thawonmas, R., Kurashige, M., Chen, K.T.: Detection of landmarks for clustering of online-game players. IJVR **6**, 11–16 (01 2007)
27. Valdivia, P., Buono, P., Plaisant, C., Dufournaud, N., Fekete, J.D.: Analyzing dynamic hypergraphs with parallel aggregated ordered hypergraph visualization. IEEE Transactions on Visualization and Computer Graphics **PP**, 1–1 (08 2019). https://doi.org/10.1109/TVCG.2019.2933196
28. Yang, C., Wang, R., Yao, S., Abdelzaher, T.F.: Hypergraph learning with line expansion. CoRR **abs/2005.04843** (2020), https://arxiv.org/abs/2005.04843
29. Zhou, Y., Rathore, A., Purvine, E., Wang, B.: Topological simplifications of hypergraphs. CoRR **abs/2104.11214** (2021), https://arxiv.org/abs/2104.11214
30. Ágg, B., Császár, A., Szalay-Bekő, M., Veres, D., Mizsei, R., Ferdinandy, P., Csermely, P., Kovacs, I.: The entoptlayout cytoscape plug-in for the efficient visualization of major protein complexes in protein-protein interaction and signalling networks. Bioinformatics (Oxford, England) **35**, 4490–4492 (11 2019). https://doi.org/10.1093/bioinformatics/btz257

How hypergraph-to-graph conversion affects hypergraph visualization: A Multi-indicator evaluation

Anonymous authors

June 16, 2023

Contents

1 Paper List

ID	Paper Title
1	Six methods for transforming layered hypergraphs to apply layered
2	Hypergraph Learning with Line Expansion
3	Hypergraph Learning Methods and Practices
4	Multilevel hypergraph partitioning applications in VLSI domain
5	Hypergraph Drawing by Force-Directed

2 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 hypergraph?
(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 reasons why you give scores to different layouts

3 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 hypergraph 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.

4 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 [4]. 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 in https://github.com/Hypergraph-to-graph/Hypergraph-to-graph.

Table 3 outlines the complexity of the conversions, and the post-processing cost. Table 2 contains definitions for the symbols used in the equations.

Table 1: 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 2: 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	$O(\|H\| * \|I\| * \|V\|^2)$	$O(\|V\|^2)$

4.1 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.

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

$$V_i = V \tag{2}$$

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

$$|I(h_i)| * (|(I(h_i)| - 1)/2$$

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.

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

$$V_i = V \tag{4}$$

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 on https://github.com/Hypergraph-to-graph/Hypergraph-to-graph.

(3)Split-cycle: In the example below, every pair of distinct vertices A, B, D, and E in the same hyperedge is connected by exactly one edge, so it can be simplified to 7 edges, which are AB, BD, DC, CE, EA and DE.

$$\begin{aligned} E = \{e : v_i, v_{i+1} \in I(e), & v_{max}, v_{min} \in I(e), i <= max - 2, \\ & \forall v_i \in I(h_k), \forall h_k \in H\} \cup \{(\mathbf{v}_{\min}, \mathbf{v}_{\max})\} \end{aligned} \tag{5}$$

$$V_1 = V \tag{6}$$

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.

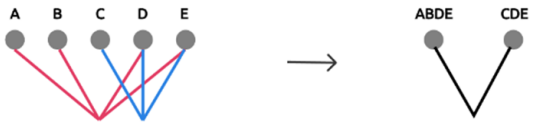
4.2 Aggregate methods

The aggregation method relies on creating aggregation vertices (also known as metanodes) 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 conversed 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.,$I(h_i) \cap I(h_j) \neq \emptyset$, 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.

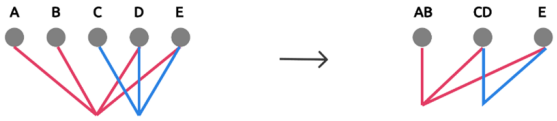
$$V_1 = \{a\left(h_i\right) \quad \forall h_i \in H\} \tag{7}$$

$$\begin{aligned} E = \{e : a\left(h_i\right), a\left(h_j\right) \in I(c), \\ I\left(h_i\right) \cap I\left(h_j\right) \neq 0. \qquad \forall h_i, h_j \in H\} \end{aligned} \tag{8}$$

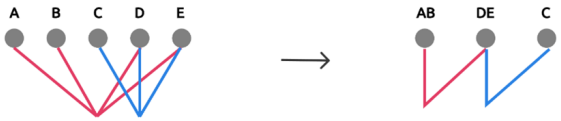


Discussion: Aggregate-collapse replaces all the vertices in the original hypergraph. The size of the conversed graph depends on the connectivity of the hypergraph: a highly connected hypergraph might introduce a large number of new vertices and edges, while a sparsely connected hypergraph might effectively 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[2] 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,[2] can be replaced with different algorithms with the same purpose, which will give



different results and different output graph sizes. The complexity of the conversion stated in Table 3 corresponds to the complexity of [2] .
(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 3).

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 Centroid-single method, we create a centroid vertex $c(h_i)$for each hyperedge

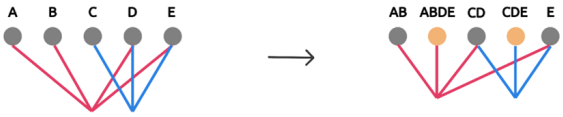
$$h_i \in H$$

, and then replace h_i with edges that link each vertex associated with h_i to the newly created centroid vertex $c(h_i)$.

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

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

(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, 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.

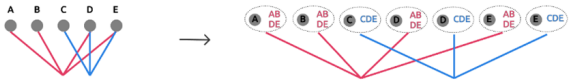
Post-processing for centroid methods: As with split methods, we keep the order of the vertices as

the same as obtained through the layout algorithm, then remove the centroids, and replace the newly introduced edges with their respective hyperedges.

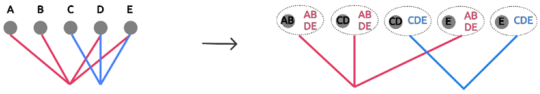
4.4 Expansion methods

(1)Line-expansion:In this method, we treat the association between vertices and hyperedges as a whole, and define that two line points 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.

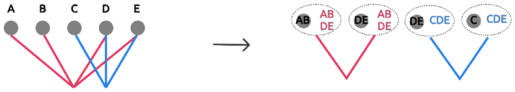
$$\begin{aligned} \mathrm{E} &= \{e : \mathrm{V_i}, \mathrm{V_{i+1}} \in \mathrm{I(e)} \quad \forall \mathrm{V_i}, \mathrm{V_{i+1}} \in \mathrm{V_1}, \forall \mathrm{v_i}, \mathrm{v_{i+1}} \in \mathrm{I}\left(\mathrm{h_k}\right), \forall \mathrm{h_k} \in \mathrm{H}\} \\ \mathrm{V_1} &= \{(\mathrm{v_{i,}}, \mathrm{h_i}) \mid \forall \mathrm{v_i} \in \mathrm{l(hi)}, \quad \forall \mathrm{h_i} \in \mathrm{H}\} \end{aligned} \tag{11}$$



(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, and then treats the association between vertices and hyperedges as a whole for line expansion.



(3)Line-expansion-aggregate-relationship-summarize: 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, and then treats the association between vertices and hyperedges as a whole for line expansion.

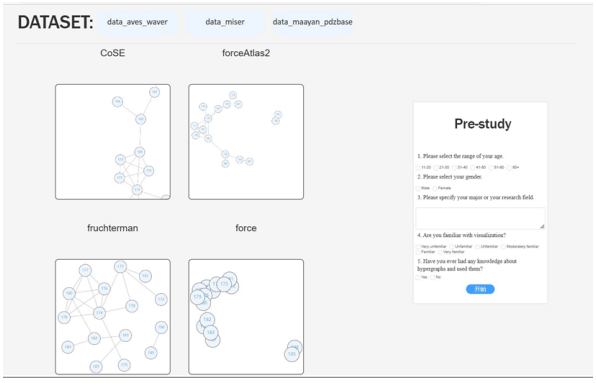


Post-processing for expansion methods: We first undo the line expansion by replacing each point in the "vertex-hyperedge" form with just the "vertex" form. Then, for each point that appears in multiple positions, we take the average of those positions to get a new coordinate. Finally, we replace the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

5 Example interfaces

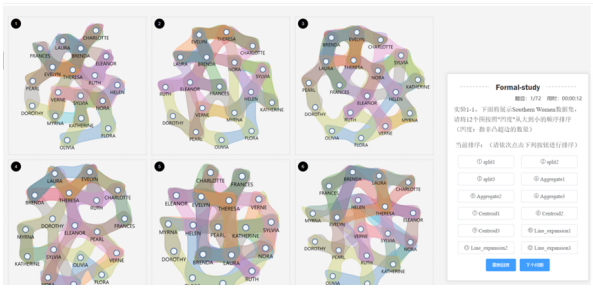
5.1 Example interface of the pre-study

Experiment 1: Understanding Layout Algorithm Effect on Visualization Effect.

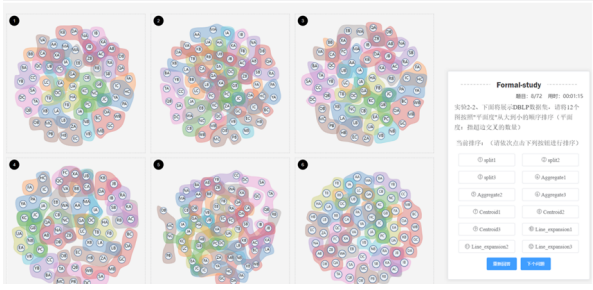


5.2 Example interface of the formal study

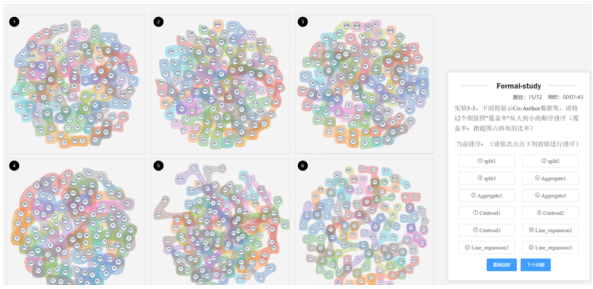
Experiment 1: Perception of concavity reduction.



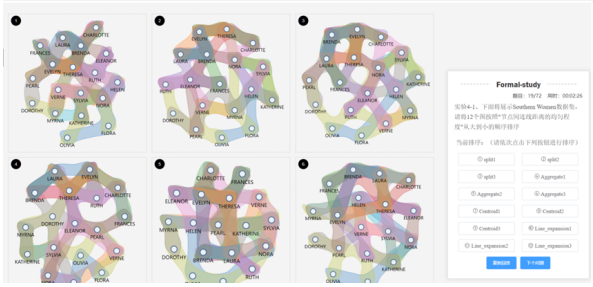
Experiment 2: Perception of planarity increases.



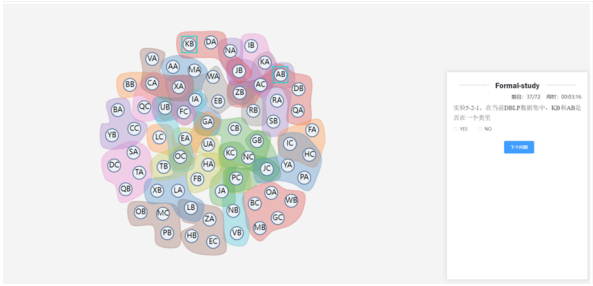
Experiment 3: Perception of the coverage rate of the plane increase.



Experiment 4: Perception of regularity increases.



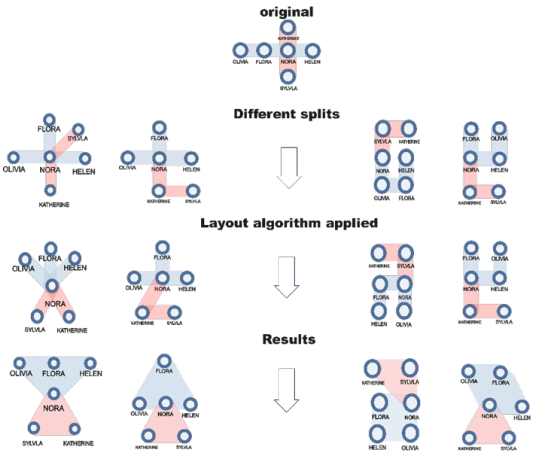
Experiment 5: Perception of the cognitive capacity of data improvement.



6 The discussion of how to choose edges and vertices in conversion methods

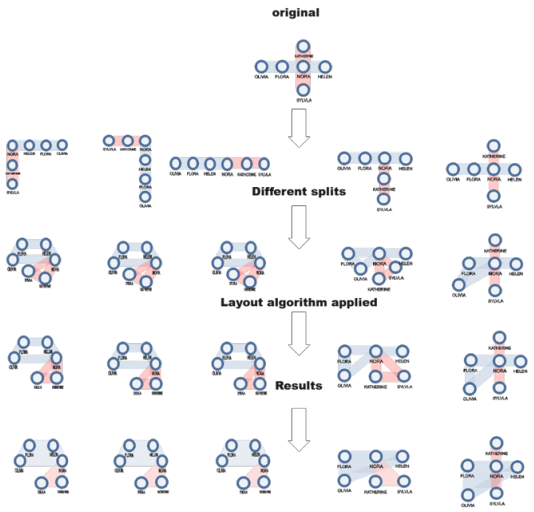
6.1 The order of vertices when using Split-path

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.



6.2 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.



7 Results and analysis of the Southern Women dataset and the DBLP dataset

7.1 Experiment 1

For the Southern Women dataset, the average completion time is 178,648ms, with 33.3% of users considering 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 Aggregate-relationship-summarize method to have the lowest concavity. 22.2% of users chose Split-clique, and 11.1% of users found 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.

7.2 Experiment 2

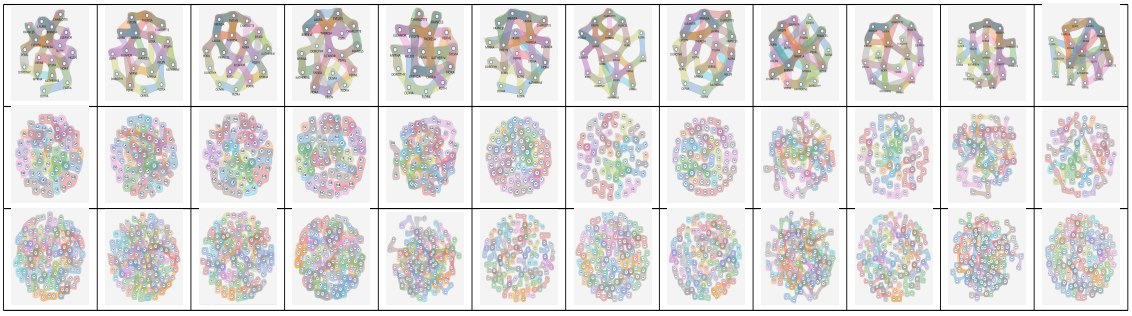
For the Southern Women dataset, the average completion time is 140,637 ms, with 27.8% of users considering Centroid-single method to have the lowest planarity. Only 16.7% of users' choices align with the actual results. The longer completion time and low accuracy of 16.7% suggest that planarity has a 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 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.

7.3 Experiment 3

For the Southern Women dataset, the average completion time is 103,034 ms, with 27.8% of users considering 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 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.

7.4 Experiment 4

For the Southern Women dataset, the average completion time is 124,874 ms, with 22.2% of users considering 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 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.



8 Selection of Evaluation Metrics

Through comprehensive research, we defined evaluation metrics from both quantitative and qualitative aspects. For the quantitative metrics, in addition to the time and accuracy of completing trials, we introduced five metrics with reference to literature: (1) concavity, which refers to the number of nonconvex hypergraph edges drawn, the smaller the concavity, the more aesthetically pleasing the hypergraph is ; (2) planarity, which refers to the number of non-adjacent hypergraph edges intersections, the smaller the planarity, the more helpful it is in avoiding clutter, thus avoiding ambiguity in the relationships represented by the hypergraph ; (3) coverage rate of the plane, whose formula is as follows. To control for variables, the canvas size was standardized to 860 x 860 pixels. The larger the ratio, the more appropriately the drawing canvas is utilized ;

$$\text{Mean}-\text{APV}_{\text{drawing}}\ (H)=\frac{\sum_{i=1}^{|E|}\frac{\text{Area}(E_i)}{|E_i|}}{|E|}$$

where $\text{Area}\,(E_i)$ is the area of the shape representing E_i and $|E_i|$ is the number of vertices in the hyperedge E_i (the cardinality of E_i). The 'Mean area per vertex' of the drawing canvas, Mean-APV canvas is computed as

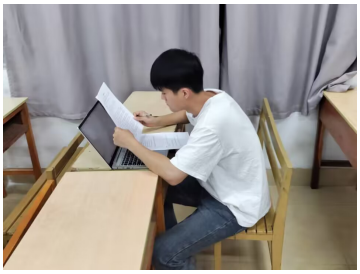
$$\text{Mean-}\,\text{APV}_{\text{canvas}}\ (H)=\frac{\text{Area}_{\text{ canvas}}}{|X|}$$

hypergraph $H(X,E)$ is defined as

$$\text{Coverage}(H)=\frac{\text{Mean}-\text{APV}_{\text{drawing}}\ (H)}{\text{Mean}-APV_{\text{canvas}}\ (H)}.$$

(4) regularity, which refers to the evenness of the distance between the connections of vertices, is quantified by the variance of the lengths of the edges connecting vertices. The larger the evenness, the more uniform the distribution of vertices[1] ; (5) cognitive capacity of data, to quantify this metric, we used questions related to data categorization. These metrics often appear under different names in papers evaluating graph layouts [3]. In terms of qualitative metrics, we used a questionnaire to ask users questions related to the following five metrics: (1) difficulty rating of each experimental trial; (2) preference rating of 12 output hypergraphs; (3) self-assessment of trial completion speed; (4) self-assessment of trial completion accuracy; (5) visual background. The specific questionnaire is included in the appendix on https://github.com/Hypergraph-to-graph/Hypergraph-to-graph. Therefore, in the study, we will evaluate the conversion methods from both seven quantitative and five qualitative metrics in order to find the different applicable scenarios for each conversion method.

9 User Study



References

[1] Arafat, N.A., Bressan, S.: Hypergraph drawing by force-directed placement. In: International Conference on Database and Expert Systems Applications (2017)

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

[3] Haleem, H., Wang, Y., Puri, A., Wadhwa, S., Qu, H.: Evaluating the readability of force directed graph layouts: A deep learning approach. CoRR **abs/1808.00703** (2018), http://arxiv.org/abs/1808.00703

[4] 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.).