Eurographics Conference on Visualization (EuroVis) (2013)
M. Hlawitschka and T. Weinkauf (Editors)

A Novel Radial Visualization Approach for Undirected Hypergraphs

Andreas Kerren and Ilir Jusufi

Linnaeus University, Department of Computer Science, Vejdes Plats 7, SE-35195 Växjö, Sweden

Abstract

Hypergraphs are a more generalized concept of graphs where an edge typically connects multiple vertices. They are applicable to many different domains such as the representation of complex biochemical pathways or classification problems with non-empty intersections between different groups, for instance, in social network analysis. There is a need to visualize those relational data structures in such a way that a better understanding of the relationships between vertices as well as their interactive exploration is supported. This paper describes a new radial visualization technique to layout undirected hypergraphs without clutter and to provide methods of interaction and data analysis.

Categories and Subject Descriptors (according to ACM CCS): G.2.2 [Discrete Mathematics]: Graph Theory—Graph Algorithms; H.5.2 [Information Interfaces and Presentation]: User Interfaces—; I.3.6 [Computer Graphics]: Methodology and Techniques—Interaction Techniques;

1. Introduction

The visualization of graphs and networks is in the focus of many researchers in fields such as Information Visualization and Graph Drawing [GPQX07]. During the past years, a variety of many different approaches and techniques for their visualization and analysis were developed. One of many good reasons to do research in this field is the importance of graphs in many application areas. But in contrast to traditional graph visualization, the development of visual metaphors for hypergraphs was not given the same significance. This is quite surprising as they have many interesting and important application domains, such as Software Engineering [Jun08], Biochemistry [AKK*10] or Social Sciences [LBM10].

Hypergraphs are a generalization of ordinal graphs (see for instance [DBETT99]) that allow edges to be incident to more than two vertices [Ber89]. Such edges are called hyperedges which connect a set of vertices. Thus, they can be treated as non-empty subsets of the vertices in the hypergraph. More formally, a *hypergraph* is defined as $H = (V, E)$ with V is the set of vertices and E is the set of hyperedges. A *hyperedge* e is a subset of the Cartesian product of V, i.e., $e = (v_1, v_2, \ldots, v_n)$. The *cardinality* γ of a hyperedge is denoted with $n \geq 1$. In consequence, an ordinal graph is a hypergraph with a constant cardinality of 2 (also called 2-uniform hypergraph). Here and in the following, we only consider undirected/unordered hyperedges. A vertex which is part of a hyperedge is called *hypernode* or *hyperedge node* in order to be able to distinguish it from other vertices in V.

Depending of their application domain, hypergraphs can be interpreted as a *set system* (or a family of sets) drawn from the vertices in V. Consequently, they might be also used for the analysis of categorical data sets, such as customer data. Then the customers are becoming the nodes, and any category of an attribute (age range or similar) is becoming a hyperedge that connects the nodes, i.e., hyperedges represent specific grouping criteria of the vertices. In Software Engineering, for example, this could be useful for the representation of subsystem abstraction: software components might use a common subsystem and thus the components build a group sharing this subsystem which could be represented as a hyperedge [Jun08].

The main problem of the visual representation of hypergraphs is that current tools mostly introduce clutter, overlaps or many edge crossings depending on the used visualization metaphor, cf. Sect. 2. Often, such visualizations are restricted to the pure layout and not the visual analysis of hypergraphs that supports navigation and filtering. In this paper, we present a novel, projection-based visual metaphor for

In Proceedings of the 17th Eurographics Conference on Visualization (EuroVis '13), short paper track, Leipzig, Germany, 2013. Eurographics.

representing hypergraphs. The prototypical implementation of our approach was guided by the following requirements:

- user-friendly metaphor that is intuitively understandable
- input hypergraphs should be specified by GraphML files
- add standard interaction, such as zooming, filtering, or reordering of hypergraph elements
- hyperedges should not overlap (no crossings of hyperedges)
- hyperedges and ordinary edges (hyperedges of cardinality two) should be treated and shown separately

In addition, we considered several tasks that should be supported by the visualization and associated interaction techniques:

- find the hyperedge nodes of a selected hyperedge (or set of hyperedges)
- determine all hyperedges that share a specific node (or set of nodes)
- filter nodes/hyperedges by means of topological features, such as node degree
- estimate the cardinality of hyperedges
- support editing of nodes and hyperedges if needed

The remainder of this short paper is organized as follows. The next section provides a brief overview of related works and highlights their drawbacks which are partly addressed by our own approach. In Section 3, we introduce our visualization tool including the interaction features. Section 4 shortly summarizes the results of a small and simple evaluation to get a first impression of the user acceptance. The conclusion and future work section deals with possible improvements of the tool.

2. Related Works

This section highlights known approaches and tools for the visual representation of hypergraphs. We can distinguish between traditional approaches that are mainly used in the graph theory literature and more recent approaches adopted in a variety of visualization tools. Note that we only focus on graphs with undirected hyperedges.

Traditional Approaches Methods within this category are pure diagrams, i.e., they are typically not interactive, but complex and do not scale well. Mostly, subset representations are used which is based on the hypergraph's interpretation as set system [SAA09]. Here, the vertices are represented as points in the plane, and a hyperedge is shown as closed curve (contour) that only contains those nodes that are part of the edge. The idea is conceptually similar to Venn or Euler diagrams, but without any regional constraints that have to be taken into account. This metaphor is also used in modern tools as described below. Another technique is the use of node-link diagrams with Steiner trees as edges. The so-called Steiner tree problem looks for a minimal-weight tree which connects a specific set of vertices (terminals) in an undirected, weighted graph [HRW92]. Possible non-terminals in the trees are called Steiner points. The problem is NP-complete, but there exist heuristics that run in polynomial time. Another way to build a diagram representing a hypergraph is its visualization as a bipartite graph $G_b = (U, V, E)$. Vertices in V correspond to the vertices of the hypergraph, but those in U represent the hyperedges. As as result of this design, the edges of G_b in E indicate vertex-hyperedge incidences. Disadvantages of this approach are the linear structure of the bipartite graph and the massive line intersections in worst case scenarios.

Recent Approaches Many current network visualization tools simply handle hypergraphs by extending standard graph layouts with colored edges, i.e., each hyperedge is replaced by a set of binary edges with the same color assigned. According to Ware [War04], this idea only works efficiently up to 12 colors (hyperedges); no interaction is usually provided. A simple implementation can be found in Wolfram Mathematica [Wol13]. Sometimes, this edge coloring approach is combined with a redundant node coloring (similar to a pie chart) to strengthen the visual perception. More advanced techniques use tight colored edge hulls as described in [DvKSW12, LQB12, CC12]. Finally, some tools use closed contours for the display of hyperedges as already described above. To distinguish them, every hyperedge region is assigned an individual color, thus each hyperedge node lies within a colored region which may overlap if nodes are part of different hyperedges. This leads to mixed colors inside the intersecting regions. Prominent examples can be found in the Jung library [OFN13] or in similar approaches such as [CPC09, HRD10, BT09, BT06]. Even if there are very advanced layout algorithms for Venn and Euler style diagrams that produce aesthetic results, the visual complexity quickly becomes very high if the data sets are getting larger. Moreover, they usually lack advanced interaction techniques.

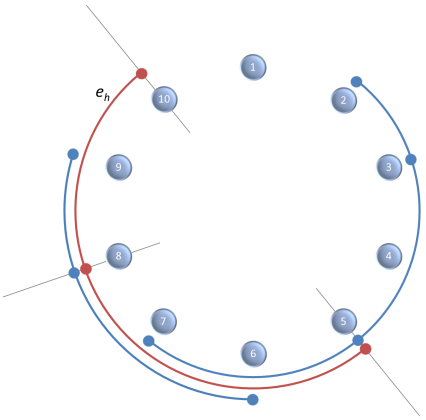

Figure 1: *Visual metaphor used in our approach. The red marked hyperedge e_h connects the nodes 5, 8 and 10.*

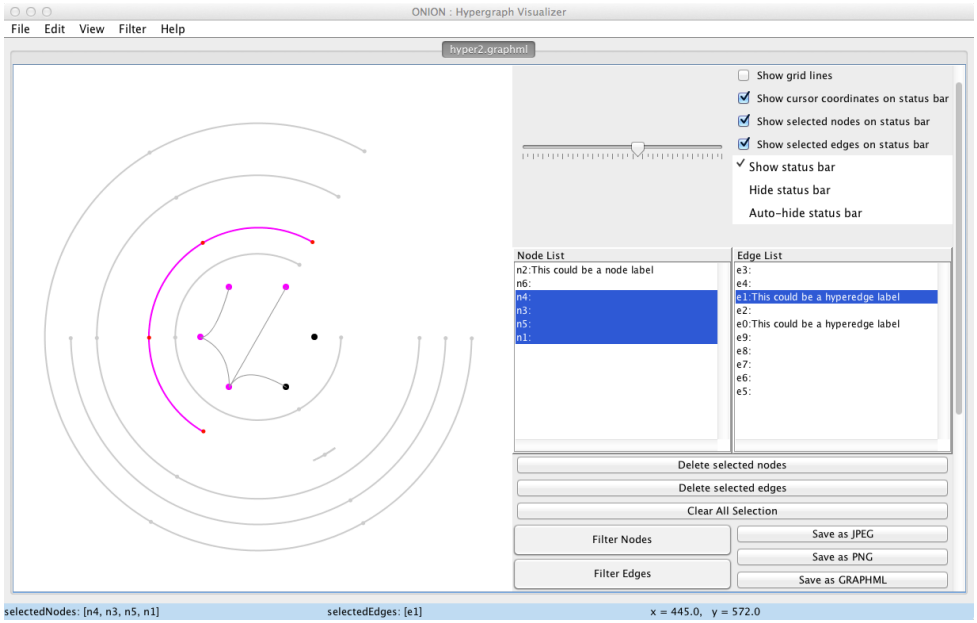

Figure 2: *Main window of our tool with the hypergraph view on the left hand side and the control panel on the right. The small sample hypergraph has six nodes and ten hyperedges (four of them with a cardinality of two). Edge e_1 was selected by the user.*

3. Visualization Approach

To address the design requirements discussed in the introduction, we decided to use a radial layout approach for our first prototype implementation (inspired by [GW03]). As shown in Fig. 1, the nodes of the hypergraph are evenly distributed on a virtual circle (ten nodes in the presented example). In the current implementation their order is arbitrary. Hyperedges with a cardinality of at least three are represented as arcs that enclose the circle. Thick points on the arcs indicate which nodes are part of them. Thus, to interpret the red highlighted hyperedge in the figure, the user has to project the points to the center of the overall layout and is thus able to identify the hyperedge nodes. Here, the highlighted hyperedge $e_h = (v_5, v_8, v_{10})$. By doing so, we can guarantee that no hyperedges with a cardinality larger than two can overlap.

One potential drawback of this design is the wasted space in the middle. In order to make use of it, we decided to draw binary edges (hyperedges e with $\gamma(e) = 2$) inside of the virtual circle. Simultaneously, we fulfill another requirement that hyperedges and ordinary edges should be treated and shown separately. In order to avoid clutter, we bundle inner ordinary edges together in case they are incident to the same node. Note that self-loops ($\gamma(e) = 1$) are allowed; they are represented as short arcs with just one point.

Based on these design decisions, we have implemented a first prototype in Java which accepts hypergraphs specified in GraphML [BEH*02] as input. Fig. 2 shows a screenshot of the current prototype.

3.1. Interaction

Our implementation supports several interaction features which help the user to discover the structure of the input hypergraph and to analyze it further. Of course standard techniques are offered, such as zooming/panning or changing a set of layout parameters (distance between arcs, color of graphical elements, thickness of arcs, etc.). It is also possible to export the displayed graph in different image formats, but also as GraphML file if the user edited the graph itself. In the following, we briefly discuss the most important features that facilitate the analysis process.

Selecting, Highlighting, and Grouping The selection (incl. hovering) of a node by mouse click has two effects. First, all hyperedges that include the selected node (called hyperedge group) as well as all other nodes of the group are highlighted in the selection color. The node itself is marked with a black halo to distinguish it from the others. Second, the corresponding hyperedge group is moved inwards, similar to the Bring & Go approach [MCH*09]. As a result, all related hyperedge arcs are very close to the selected node; unrelated arcs are moved outwards and faded out a bit (cf. Fig. 3). However, it is also possible to move individual arcs outwards or inwards manually.

Vice versa if an hyperedge arc is selected (or hovered), then the related hyperedge nodes are highlighted only. Multiple selection is also provided. Both, nodes and hyperedges can be alternatively selected by using the lists in the control panel as shown on the right hand side of the screenshot in Fig. 2. Here, hyperedge e_1 was selected by the user.

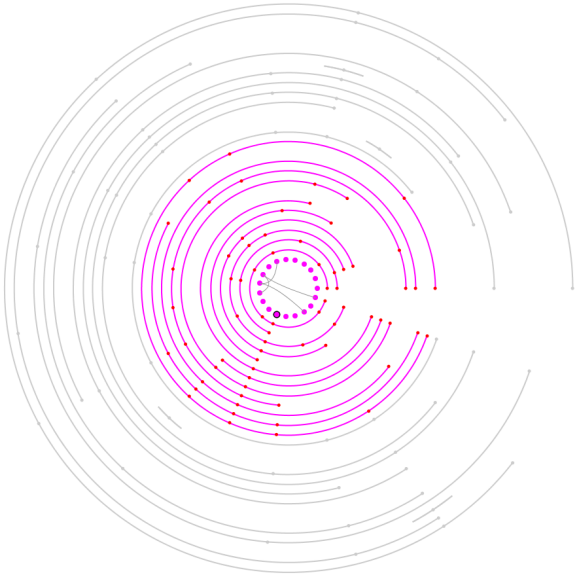

Figure 3: *Hypergraph with 20 nodes and 30 hyperedges. The black circled node was manually selected, and the ten hyperedges that contain the selected node were highlighted and moved inwards.*

Filtering The user also has the possibility to filter nodes and edges based on specific topological features. If the node filtering option was chosen in the menu, then a dialog box is opened where the user is able to specify conditions, such as that a node must have a specific degree or that a node must be incident to all selected hyperedges. Similar conditions can be expressed with the help of an hyperedge filtering dialog box. After the filtering was applied, nodes and hyperedges fulfilling the conditions are highlighted with the selection color whereas others are faded out.

Editing Our tool permits the deletion of selected nodes/ hyperedges. Users may save a modified version of the original hypergraph in GraphML. Adding of hypergraph elements is not supported yet, but planned for future versions.

4. First Evaluation

We performed a small evaluation to learn more about the user acceptance, current problems and pros/cons of our idea compared to edge coloring visualization (cf. Section 2). For this, we asked 32 test persons ($\frac{1}{3}$ female and $\frac{2}{3}$ male) to complete a questionnaire based on experiences they made with the two visualization approaches applied on a sample hypergraph. This sample hypergraph represented groupings of movie data, i.e., 50 actors/actresses were considered as nodes and 17 movie directors as hyperedges. Our participants were mainly undergraduate students in a software development study program. We are aware that there are learning effects and other issues in this evaluation design, but at this stage of the development, we just wanted to get a first impression if our ideas make sense for users.

First results support our impression that the new projection-based approach might outperform standard edge coloring methods: 80% of the participants think that our idea is more powerful, and 93% believe that it is more appropriate for larger hypergraphs. With respect to hypergraph topology, 70% are of the opinion that our approach is better in revealing connections between nodes/hyperedges. In addition, they made proposals how to improve our tool. These are partly discussed in the next section.

5. Conclusions and Future Work

In this paper, we have presented a new radial visualization method which is able to display undirected hypergraphs without clutter including a set of possibilities to interact with them. Our own experiences and the small evaluation provide reasons to believe that it scales well up to 100 nodes and 150 hyperedges. In our opinion, this is better as many other layout approaches, especially with respect to the number of hyperedges (other techniques usually scale up to 20-30 hyperedges only). Here, the maximal number of nodes is strongly restricted by the (variable) size of the inner circle. On the other hand, the number of crossing-free hyperedge arcs is restricted by the size/resolution of the display, but an arc should always be completely visible. Otherwise, the user loses his/her mental map [MELS95]. The interaction possibilities, particularly the filtering or the arc grouping and bringing features, improve this situation in practice. However, there is more work to do in order to offer an efficient tool for the visual analysis of hypergraphs. As future work, we plan to extend and improve the current implementation according to the following list of modifications:

- A better edge bundling in the center should be implemented that considers the hyperedges outside. The current arrangement of nodes on the circle is arbitrary which has to be also improved in future versions.
- Another drawback are the missing node/edge labels within the main view (they are only shown in the status bar). It would be more intuitive if they would be available inside the graph view (for example as tooltips or in form of a magic lens approach).
- Comparison of at least two hypergraphs should be supported.
- It is not possible to add nodes or edges in the current prototype, because this is usually not needed in the analysis process. Adding those elements would turn the tool into a hypergraph editor which also might be used for data curation [KHP*11].
- A real usability study is needed to get significant evaluation results.

Acknowledgments The authors wish to thank Alev and Deniz Cakici for implementing the first prototype and for performing the evaluation [CC12].

References

[AKK*10] ALBRECHT M., KERREN A., KLEIN K., KOHLBACHER O., MUTZEL P., PAUL W., SCHREIBER F., WYBROW M.: On open problems in biological network visualization. In *Proceedings of the International Symposium on Graph Drawing (GD '09)* (2010), vol. 5849 of *LNCS*, Springer, pp. 256–267. doi:10.1007/978-3-642-11805-0_25. 1

[BEH*02] BRANDES U., EIGLSPERGER M., HERMAN I., HIMSOLT M., MARSHALL M.: GraphML progress report structural layer proposal. In *Graph Drawing*, Mutzel P., Jünger M., Leipert S., (Eds.), vol. 2265 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2002, pp. 501–512. doi:10.1007/3-540-45848-4_59. 3

[Ber89] BERGE C.: *Hypergraphs: The Theory of Finite Sets*. North-Holland, Amsterdam, The Netherlands, 1989. 1

[BT06] BYELAS H., TELEA A.: Visualization of areas of interest in software architecture diagrams. In *Proceedings of the ACM Symposium on Software Visualization (SoftVis '06)* (2006), ACM, pp. 105–114. doi:10.1145/1148493.1148509. 2

[BT09] BYELAS H., TELEA A.: Visualizing metrics on areas of interest in software architecture diagrams. In *Proceedings of the IEEE Pacific Visualization Symposium (PacificVis '09)* (2009), pp. 33–40. doi:10.1109/PACIFICVIS.2009.4906835. 2

[CC12] CAKICI A., CAKICI D.: *Visualization of Hypergraphs*. Master's thesis, Linnaeus University, School of Computer Science, Physics and Mathematics, Växjö, Sweden, 2012. 2, 4

[CPC09] COLLINS C., PENN G., CARPENDALE S.: Bubble sets: Revealing set relations with isocontours over existing visualizations. *IEEE Transactions on Visualization and Computer Graphics 15*, 6 (2009), 1009–1016. doi:http://doi.ieeecomputersociety.org/10.1109/TVCG.2009.122. 2

[DBETT99] DI BATTISTA G., EADES P., TAMASSIA R., TOLLIS I. G.: *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice Hall, 1999. 1

[DvKSW12] DINKLA K., VAN KREVELD M. J., SPECKMANN B., WESTENBERG M. A.: Kelp diagrams: Point set membership visualization. *Computer Graphics Forum 31*, 3pt1 (2012), 875–884. doi:10.1111/j.1467-8659.2012.03080.x. 2

[GPQX07] GÖRG C., POHL M., QELI E., XU K.: Visual Representations. In *Human-Centered Visualization Environments* (2007), Kerren A., Ebert A., Meyer J., (Eds.), LNCS Tutorial 4417, Springer, pp. 163–230. 1

[GW03] GU Q.-P., WANG Y.: Efficient algorithm for embedding hypergraphs in a cycle. In *High Performance Computing (HiPC '03)* (2003), Pinkston T., Prasanna V., (Eds.), vol. 2913 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp. 85–94. doi:10.1007/978-3-540-24596-4_10. 3

[HRD10] HENRY RICHE N., DWYER T.: Untangling euler diagrams. *IEEE Transactions on Visualization and Computer Graphics 16*, 6 (nov.-dec. 2010), 1090–1099. doi:10.1109/TVCG.2010.210. 2

[HRW92] HWANG F. K., RICHARDS D. S., WINTER P.: *The Steiner Tree Problem*. No. 53 in Annals of Discrete Mathematics. North-Holland, 1992. 2

[Jun08] JUNGHANS M.: *Visualization of Hyperedges in Fixed Graph Layouts*. Master's thesis, Brandenburg University of Technology Cottbus, Computer Science Department, Cottbus, Germany, 2008. 1

[KHP*11] KANDEL S., HEER J., PLAISANT C., KENNEDY J., VAN HAM F., HENRY RICHE N., WEAVER C., LEE B., BRODBECK D., BUONO P.: Research directions in data wrangling: Visualizations and transformations for usable and credible data. *Information Visualization 10*, 4 (2011), 271–288. doi:10.1177/1473871611415994. 4

[LBM10] LIU D., BLENN N., MIEGHEM P. V.: Modeling social networks with overlapping communities using hypergraphs and their line graphs. *CoRR abs/1012.2774* (2010). 1

[LQB12] LAMBERT A., QUEYROI F., BOURQUI R.: Visualizing patterns in node-link diagrams. In *Proceedings of the 16th International Conference on Information Visualisation (IV '12)* (2012), pp. 48–53. doi:10.1109/IV.2012.19. 2

[MCH*09] MOSCOVICH T., CHEVALIER F., HENRY N., PIETRIGA E., FEKETE J.-D.: Topology-aware navigation in large networks. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '09)* (2009), ACM, pp. 2319–2328. doi:10.1145/1518701.1519056. 3

[MELS95] MISUE K., EADES P., LAI W., SUGIYAMA K.: Layout adjustment and the mental map. *Journal of Visual Languages and Computing 6* (1995), 183–210. 4

[OFN13] O'MADADHAIN J., FISHER D., NELSON T.: JUNG - Java Universal Network/Graph Framework, last accessed: 07-03-2013. http://jung.sourceforge.net/. 2

[SAA09] SIMONETTO P., AUBER D., ARCHAMBAULT D.: Fully automatic visualisation of overlapping sets. In *Proceedings of the 11th Eurographics/IEEE – VGTC Conference on Visualization (EuroVis'09)* (2009), Eurographics Association, pp. 967–974. doi:10.1111/j.1467-8659.2009.01452.x. 2

[War04] WARE C.: *Information Visualization: Perception for Design*, 2nd ed. Morgan Kaufmann, 2004. 2

[Wol13] WOLFRAM DEMONSTRATIONS PROJECT (CONTRIBUTED BY MICHAEL SOLLAMI): Three-Uniform Hypergraphs, last accessed: 27-02-2013. http://demonstrations.wolfram.com/ThreeUniformHypergraphs/. 2