Hypergraph Drawing by Force-Directed Placement

Naheed Anjum Arafat(✉) and Stéphane Bressan

School of Computing, National University of Singapore, 13 Computing Drive, Singapore 117417, Singapore
e0001887@u.nus.edu, steph@nus.edu.sg

Abstract. We propose a family of algorithms that transform a hypergraph drawing problem into a graph drawing problem and leverage force-directed graph drawing algorithms in order to draw hypergraphs. We propose and discuss a number of criteria to evaluate the quality of the drawings from the points of view of aesthetics and of visualization and analytics. We empirically and comparatively evaluate the quality of the drawings based on these criteria on both synthetic and real data sets. Experiments reveal that the algorithms are generally effective and the drawings generated are aesthetically pleasing.

Keywords: Hypergraph · Visualization · Force-directed graph drawing

1 Introduction

We address the problem of drawing hypergraphs. A *Hypergraph* is a generalization of a graph where multi-ary relations exists among the objects including binary relations. Edges of a hypergraph are called *hyperedges*. Hypergraphs have found application in many areas of science [5,8,9,12,15] and a key challenge in visualizing these hypergraphs is the exponential increase in the number of potential relations with the number of objects [2]. The need to perform visual analytics on data with multi-ary, complex relations calls for practically effective and visually pleasing drawings of hypergraphs.

Fruchterman and Reingold, in their 1991 seminal paper [11], presented the popular force-based graph layout algorithm that bears their names. In this algorithm, the graph is modeled as a physical system where vertices are attracted and/or repelled according to some force function, eventually resulting in an equilibrium configuration. The family of algorithms proposed in this paper transform a hypergraph into a graph and leverage force-directed drawing algorithms to draw the graph. The final positions of the vertices of the graph are then used to draw the hyperedges of the hypergraph.

Inspired by the desirable properties of hypergraph drawing proposed by Mäkinen [13], we propose and discuss some criteria for good hypergraph drawing. We also devise several metrics to empirically and comparatively evaluate our algorithms based on both synthetic and real data sets. The experiments

D. Benslimane et al. (Eds.): DEXA 2017, Part II, LNCS 10439, pp. 387–394, 2017.
DOI: 10.1007/978-3-319-64471-4_31

reveal that the approach is practical, efficient and generally effective, and, more importantly, the drawings generated are not only aesthetically pleasing but also readable for the purpose of visualization and analytics.

2 Related Work

Two classes of hypergraph drawing have been frequently used in the literature as mentioned by Mäkinen [13] namely, Subset based and Edge based. In both of them, vertices of the hypergraphs are drawn as points in the plane. In Edge based drawings, hyperedges are drawn as smooth curves connecting their vertices. In Subset based drawings, hyperedges are drawn as closed curves enveloping their vertices.

Mäkinen [13] formulates the desirable properties of a subset based hypergraph drawing. Bertault F. and Eades P. [4] demonstrate a method for drawing hypergraph in subset standard. The drawing algorithm constructs a Euclidean Steiner tree from the position of the vertices in a hyperedge, uses force-directed graph drawing algorithm to get the location of the vertices and draws a contour around the edges of the tree.

Since hyperedges are sets, visualizing hypergraphs is closely related to the approaches for visualizing sets. Euler diagram is amongst the earliest set visualization methods. Closed curves such as ellipses, circles or polygons represent sets and curve overlaps represent set relations. Many algorithms exist for drawing Euler diagrams. They differ from each other by their definitions of Euler diagram and drawable instances. Flower and Howse [10] define concrete Euler diagram and propose an algorithm to generate concrete diagrams automatically up to three sets. An extended definition of Euler diagram is given in [18] and the problem of generating diagrams up to eight sets is addressed. It is worth to mention that, the Subset based drawings differ from Euler diagrams in that the former does not impose regional constraints (e.g. not allowing empty zones, allowing exactly two points of intersections between contours representing sets) as the latter [16].

Recently, Simonetto et al. [17] propose an approach for generating Euler-like diagrams which are Euler diagrams with no regional restrictions. The sets are drawn as closed bezier curves. Subset based drawings such as the one presented in this paper are closely related to Euler-like drawings such as the one proposed by Simonetto et al. [17]. Both of these methods draw sets spatially repositioning the set elements and thus do not cater to the cases where preserving the semantics of the layout is important (e.g. scatter plots, geographical maps). Other approaches such as Bubble Sets [6], LineSets [1], Kelp diagrams [7] and KelpFusion [14] are designed for those cases. Simonetto et al. [17] applies a force-directed algorithm that preserves edge-crossing properties on the intersection graph of the hypergraph whereas our algorithm is able to apply any force-directed algorithm on a class of graphs derived from the hypergraph. Simonetto et al. [17] approximates the set boundaries by computing polygons whereas our algorithm computes convex polygons of the set of points. Thus the resulting diagram might be concave [1] violating one of the aesthetics we propose in this paper.

3 Aesthetics

We propose the following criteria of a good hypergraph drawing.

Firstly, The *Concavity* metric refers to the number of non-convex hyperedge drawings drawn by an algorithm. Since convex shapes are visually simpler, minimizing non-convex shapes results in good hypergraph drawing. It is to be noted that, minimizing non-convex shapes i.e. minimizing the *Concavity* metric helps ensure the first criterion as well.

Secondly, The *Planarity* metric is defined as the number of non-adjacent hyperedge crossings of a drawing. Drawing of a hypergraph, ideally, should have no crossing between any pair of non-adjacent hyperedges. In practice, however, having a crossing-free drawing of a hypergraph is rather difficult. Thus it is desirable to have as little non-adjacent hyperedge crossing (*Planarity*) as possible. Drawings with better *Planarity* help avoid clutter and thus ambiguity of the relations being represented.

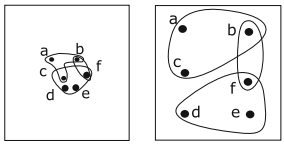


Fig. 1. Two drawings of the hyperedges $\{a, b, c\}, \{b, f\}, \{d, e, f\}$. The drawing on the left has a concave shape, crossing between a non-adjacent hyperedge pair, poor Coverage and non-uniform distribution of vertices (clutter). The drawing on the right is aesthetically superior to the one on the left as it has no concavity, no crossing, comparatively better Coverage and Regularity.

Thirdly, The *Coverage* metric refers to the ratio of the 'mean area per vertex' of the drawing to the 'mean area per vertex' of the entire drawing canvas. The 'mean area per vertex' of a drawing of a hypergraph $H(X, E)$, Mean-APV$_{drawing}(H)$ is defined as

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

where $Area(E_i)$ 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 -

$$\text{Mean-APV}_{canvas}(H) = \frac{Area_{canvas}}{|X|} \tag{2}$$

where, $Area_{canvas}$ is the area of the drawing canvas. Thus, the *Coverage* of the hypergraph $H(X, E)$ is defined as

$$Coverage(H) = \frac{\text{Mean-APV}_{drawing}(H)}{\text{Mean-APV}_{canvas}(H)}. \tag{3}$$

Maximizing the *Coverage* metric implies that the drawing canvas is utilized properly by the drawing, in other words, the drawing is sparse in some way. A *Coverage* value close to 1 implies almost 100% utilization of the drawing area.

Fourthly, The *Regularity* metric is a measure of uniformity of the vertices over the drawing canvas. Maximizing the *Regularity* criterion helps ensure less clutter of the vertices over the drawing canvas. Since the *Coverage* criteria itself fails to capture uniformity of vertices in cases of drawings with crossings, *Regularity* gives insights into the distribution of vertices over the drawing area in those cases.

Figure 1 illustrates how the proposed metrics encapsulate the aesthetics of a drawing. Interested readers may refer to [3] for implementation of these metrics.

4 Algorithms

The family of algorithms we propose initializes the coordinates of the position of the vertices of the hypergraph in a certain way, transforms the input hypergraph into a graph termed as the *associated graph* of the input hypergraph, draw the graph using force-directed graph layout algorithm to find the coordinates of the vertices of the input hypergraph and envelop the vertices each hyperedge inside a closed curve.

It is well-known that the initial position of vertices influences the performance of the force-based graph layout algorithm. Vertices of the hypergraph can be initialized randomly (*Random initialization*), in a circular fashion or uniformly over the drawing canvas. In *Circular initialization*, for each vertex x_i, its position $x_i.pos$ is initialized to the coordinate of a randomly generated point on a circle of radius $k|E_j|$ where $k \in \mathbb{N}$ and $x_i \in E_j$. In *Grid based initialization*, the drawing canvas is divided into grids, the vertices are associated with grids sequentially and $x_i.pos$ is initialized to a random point inside the grid x_i is placed in.

From Hypergraph to Graph. We propose four different ways of constructing associated graphs of a hypergraph- namely, complete associated graph, star associated graph, cycle associated graph and wheel associated graph. Each of these constructions gives rise to an algorithm.

Consider a hypergraph H denoted by the tuple $(X, E = \{E_1, E_2, \ldots, E_n\})$. For practical purposes, also consider $x_i.pos$ denotes the position vector of an arbitrary vertex $x_i \in X$ on the drawing canvas. Furthermore, $\{x_1, x_2, \ldots, x_{|E_i|}\}$ denotes the set of vertices in the arbitrary hyperedge E_i.

A *complete associated graph* $C(H)$ of the hypergraph H is a graph whose set of vertices is X and for each hyperedge E_i, any pair of distinct vertices in E_i are connected by a unique edge in $C(H)$. To illustrate, the complete associated graph of the hypergraph $H_1 = (\{a, b, c, d\}, \{\{a, b, c, d\}, \{c, d\}, \{a\}\})$ consists of $\{a, b, c, d\}$ as the set of vertices and $\{(a, b), (b, c), (c, d), (a, c), (b, d), (a, d)\}$ as the set of edges. The drawing algorithm which transforms a hypergraph into its

complete associated graph is termed as the *Complete algorithm*. The motivation behind the *Complete* algorithm stems explicitly from the underlying principle of the force-directed graph layout algorithms and implicitly from the *Planar* characteristic of good drawing. Intuitively, fewer crossings are expected to occur among a set of hyperedges if the constituent vertices of a hyperedge are spatially closer to each other than to the vertices of the other hyperedges.

A *cycle associated graph* $Cy(H)$ of the hypergraph H is a graph whose set of vertices is X and for each hyperedge E_i, a cycle is formed by adding edges $(x_1, x_2), (x_2, x_3), \ldots, (x_{|E_i|}, x_1)$ in $Cy(H)$. The cycle formed is unique if we consider the vertices in E_i sorted clockwise according to their position in the drawing. The drawing algorithm which transforms a hypergraph into its cycle associated graph is named the *Cycle algorithm*. To illustrate, the cycle associated graph of the hypergraph H_1 mentioned before consists of $\{a, b, c, d\}$ as the set of vertices and $\{(a, b), (b, c), (c, d), (a, d)\}$ as the set of edges. Sometimes attractive forces between the vertices in the complete subgraphs of $C(H)$ are too strong which in turn results in a cluttered drawing. The desire to have a sparse drawing with good *Coverage* and *Regularity* is the driving force of the *Cycle* algorithm since $Cy(H)$ is a subgraph of $C(H)$.

If we allow vertices other than X into our associated graph, transformations of other kinds emerge. Given the position vectors of the vertices in X and an arbitrary hyperedge E_i in the hypergraph H, the barycenter of the vertices in E_i denoted as b_i is the unique vertex located at the position $\sum_{i=1}^{k} \frac{x_i.pos}{k}$. The set of barycenters of H denoted as B is $\{b_1, b_2, \ldots, b_n\}$.

A *star associated graph* $S(H)$ of a hypergraph $H = (X, E)$ is a graph whose set of vertices is $H \cup B$ and for each hyperedge E_i and its barycenter b_i, a star is formed by adding edges $(b_i, x_1), (b_i, x_2), \ldots, (b_i, x_{|E_i|})$. To illustrate, $S(H_1)$ of the hypergraph H_1 mentioned before consists of $\{a, b, c, d, b_1, b_2\}$ as the set of vertices and $\{(b_1, a), (b_1, b), (b_1, c), (b_1, d), (b_2, c), (b_2, d)\}$ as the set of edges. The drawing algorithm which transforms a hypergraph into its star associated graph is named the *Star algorithm*. The design principle of the star algorithm follows from the fact that, spatial nearness among the vertices from the same hyperedge and remoteness among the vertices from distinct hyperedge can be achieved if all the vertices feel the attractive force towards the barycenter.

A *wheel associated graph* $W(H)$ of the hypergraph H is a graph whose set of vertices is $H \cup B$ and for each hyperedge E_i and its barycenter b_i, a wheel is formed by adding edges $(b_i, x_1), (b_i, x_2), \ldots, (b_i, x_{|E_i|}), (x_1, x_2), (x_2, x_3), \ldots, (x_{|E_i|}, x_1)$. To illustrate, $W(H_1)$ of the hypergraph H_1 mentioned above consists of ($\{a, b, c, d, b_1, b_2\}$ as the set of vertices and $\{(b_1, a), (b_1, b), (b_1, c), (b_1, d), (b_2, c), (b_2, d), (a, b), (b, c), (c, d), (a, d)\}$) as the set of edges. The drawing algorithm which transforms a hypergraph into its wheel associated graph is named the *Wheel algorithm*. Note that, the set of hyperedges in the wheel associated graph is the union of the set of hyperedges of the cycle and the star associated graphs i.e. $W(H) = S(H) \cup Cy(H)$.

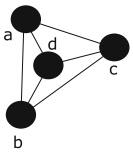


Fig. 2. The vertices of a hyperedge $\{a, b, c, d\}$ after drawing its Complete associated graph.

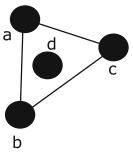


Fig. 3. Convex hull of the vertices a, b, c, d and its bordering vertices a, b, c.

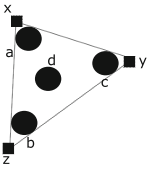


Fig. 4. Pair-wise outtangents of the bordering vertices and points x, y, z as their intersections.

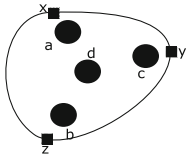


Fig. 5. The hyperedge drawn as a closed Catmull-Rom Spline going through x, y, z.

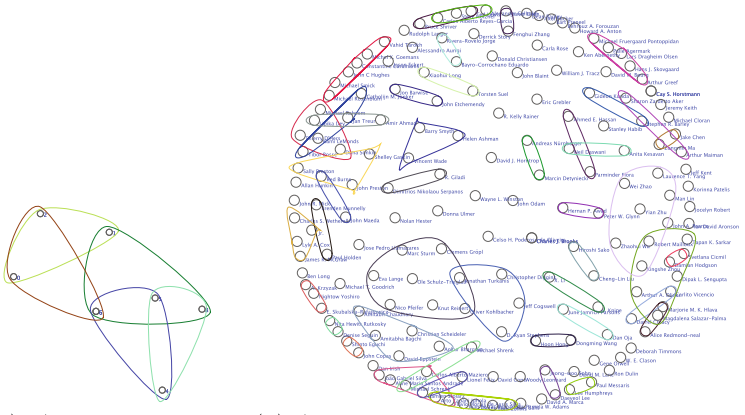
Drawing Hypergraph from Associated Graph. The Force-directed graph drawing algorithm applied to an associated graph results in an embedding of the vertices of its corresponding hypergraph. Each hyperedge of that hypergraph is then drawn as a closed curve enveloping its vertices. Figures 2, 3, 4 and 5 illustrate this process in sequence.

5 Experiments, Results, and Analysis

We empirically and comparatively evaluate the effectiveness and scalability of the algorithms on both synthetic and real dataset. We generate random hypergraphs with 2000 vertices and varying number of hyperedges and use them as our synthetic dataset. We use the DBLP [1] co-authorship network as our real dataset. Readers may refer to [3] for details about our experimental results.

The *Complete* and the *Wheel* algorithms have better Planarity than the rest. The reason is the attractive forces among the vertices in the Complete and the Wheel associated graphs are higher than the other associated graphs since a wheel and complete graph has more edges than the star or the cycle graph over the same number of vertices. The *Cycle* and the *Star* algorithm have better Coverage than the others due to the dominance of repulsive forces among the vertices in the associated graphs. In terms of Regularity, the performance varies depending on the granularity parameter in the experiment. We also observe that the Grid based initialization has the same effect on the performance as the circular initialization. Random initialization results in better Regularity than the Circular initialization. In Scalability experiment, the performance of the algorithms on the metrics are consistent and similar as in the effectiveness experiment. Figure 6 illustrates some drawings generated by one of our algorithms.

[1] http://dblp.uni-trier.de/xml/.



(a) A synthetic 3-uniform hypergraph with 5 hyperedge.

(b) A hypergraph with 100 hyperedges sampled from the DBLP dataset

Fig. 6. Example of some drawings by the *Wheel* algorithm

6 Conclusion

We propose a family of algorithms for drawing hypergraphs. We also propose a set of measurable criteria for evaluating the performance of the algorithms. We empirically evaluate the effectiveness and scalability of our proposed algorithms with different initial positioning of the vertices. The drawings by our algorithms are not only aesthetically pleasing in a qualitative way but also follow a set of quantitative criteria of good drawing. However, the drawings generated have lesser uniformity than expected stemming from scattering of disconnected components by the underlying force-directed graph layout algorithm. In future, we would like to extend Fruchterman-Reingold's algorithm from two-dimensional canvas to higher dimensions by modeling the hyperedges as elastic manifolds with the hope of having better drawings.

Acknowledgement. This work is supported by the National Research Foundation, Prime Minister's Office, Singapore under its Campus for Research Excellence and Technological Enterprise (CREATE) programme.

References

1. Alper, B., Riche, N., Ramos, G., Czerwinski, M.: Design study of linesets, a novel set visualization technique. IEEE Trans. Vis. Comput. Graph. **17**(12), 2259–2267 (2011)
2. Alsallakh, B., Micallef, L., Aigner, W., Hauser, H., Miksch, S., Rodgers, P.: Visualizing sets and set-typed data: State-of-the-art and future challenges. In: Eurographics conference on Visualization (EuroVis)-State of The Art Reports, pp. 1–21 (2014)

3. Arafat, N.A., Bressan, S.: Hypergraph drawing by force-directed placement. School of Computing, National University of Singapore, Technical report number TRC6/17 (2017). http://www.comp.nus.edu.sg/naheed/files/hypergraphdrawing.pdf
4. Bertault, F., Eades, P.: Drawing hypergraphs in the subset standard (short demo paper). In: Marks, J. (ed.) GD 2000. LNCS, vol. 1984, pp. 164–169. Springer, Heidelberg (2001). doi:10.1007/3-540-44541-2_15
5. Brinkmeier, M., Werner, J., Recknagel, S.: Communities in graphs and hypergraphs. In: Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management, pp. 869–872. ACM (2007)
6. Collins, C., Penn, G., Carpendale, S.: Bubble sets: revealing set relations with isocontours over existing visualizations. IEEE Trans. Vis. Comput. Graph. **15**(6), 1009–1016 (2009)
7. Dinkla, K., van Kreveld, M.J., Speckmann, B., Westenberg, M.A.: Kelp diagrams: point set membership visualization. In: Computer Graphics Forum, pp. 875–884. Wiley Online Library (2012)
8. Eschbach, T., Günther, W., Becker, B.: Orthogonal hypergraph drawing for improved visibility. J. Graph Algorithms Appl. **10**(2), 141–157 (2006)
9. Fagin, R.: Degrees of acyclicity for hypergraphs and relational database schemes. J. ACM (JACM) **30**(3), 514–550 (1983)
10. Flower, J., Howse, J.: Generating Euler diagrams. In: Hegarty, M., Meyer, B., Narayanan, N.H. (eds.) Diagrams 2002. LNCS (LNAI), vol. 2317, pp. 61–75. Springer, Heidelberg (2002). doi:10.1007/3-540-46037-3_6
11. Fruchterman, T.M., Reingold, E.M.: Graph drawing by force-directed placement. Softw. Pract. Exp. **21**(11), 1129–1164 (1991)
12. Lundgren, J.R.: Food webs, competition graphs, competition-common enemy graphs, and niche graphs. In: Roberts, F. (ed.) Applications of Combinatorics and Graph Theory to the Biological and Social Sciences. The IMA Volumes in Mathematics and its Applications, vol. 17, pp. 221–243. Springer, New York (1989)
13. Mäkinen, E.: How to draw a hypergraph. Int. J. Comput. Math. **34**(3–4), 177–185 (1990)
14. Meulemans, W., Riche, N.H., Speckmann, B., Alper, B., Dwyer, T.: Kelpfusion: a hybrid set visualization technique. IEEE Trans. Vis. Comput. Graph. **19**(11), 1846–1858 (2013)
15. Ramadan, E., Tarafdar, A., Pothen, A.: A hypergraph model for the yeast protein complex network. In: Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International. p. 189. IEEE (2004)
16. Santamaría, R., Therón, R.: Visualization of intersecting groups based on hypergraphs. IEICE Trans. Inf. Syst. **93**(7), 1957–1964 (2010)
17. Simonetto, P., Auber, D., Archambault, D.: Fully automatic visualisation of overlapping sets. In: Computer Graphics Forum, pp. 967–974. Wiley Online Library (2009)
18. Verroust, A., Viaud, M.-L.: Ensuring the drawability of extended Euler diagrams for up to 8 sets. In: Blackwell, A.F., Marriott, K., Shimojima, A. (eds.) Diagrams 2004. LNCS (LNAI), vol. 2980, pp. 128–141. Springer, Heidelberg (2004). doi:10.1007/978-3-540-25931-2_13