Concentric Set Schematization

Michael A. Bekos[*] Universität Tübingen
Fabian Frank University of Arizona
Wouter Meulemans[†] TU Eindhoven
Peter Rodgers University of Kent
André Schulz[‡] FernUniversität in Hagen

Abstract

Sets can be visualized in various ways and settings. An important distinction between techniques is based on whether the elements have a spatial location that is to be used for the visualization; for example, the elements are cities on a map. Strictly adhering to such location may severely limit the visualization and force overlay, intersections and other forms of clutter. On the other hand, completely ignoring the spatial dimension omits information and may hide spatial patterns in the data. In this abstract, we present ongoing research on a method that is in between spatial and nonspatial visualizations. The main idea is to schematize (move) the spatial locations onto concentric circles, to improve the visualization of the set system while roughly maintaining spatial structure.

Index Terms: Human-centered computing—Visualization—Visualization techniques—Graph drawings

1 Introduction

Set systems (or hypergraphs) are useful for modeling various types of data. To analyze such systems, various techniques have been developed to visualize sets, either on a geographically accurate base map or in an abstract nonspatial layout. In the former locations of the elements are prescribed, whereas in the latter the elements can be placed freely by a visualization algorithm. A recent survey [3] shows variety of visualization techniques; various of these methods target spatial data, e.g., [2,6,7,10]. For an illustration see Figure 1, left. In the graph-drawing community, most attention has been afforded to hypergraph supports [9] for both fixed and free vertex locations, e.g. [1,4,5,8]. However, geographic accuracy is often not necessary for overview tasks or tasks focusing on set structures, even for spatial data. Yet, completely discarding spatial context may also hide structure or patterns. Schematic maps have been successful in various applications, such as metro maps, by simplifying and abstracting spatial relations to a minimum functional level, thereby clarifying and emphasizing structure in data while not disregarding (geographic) space.

In this abstract, we explore the possibility of computing schematic set visualizations. That is, we are given a set of points in a geographic space, each associated with one or more sets. We want to shift the points to new locations such that we can provide a clear representation for each of the sets; this representation is a geometry connecting (e.g. a tree, cycle or path) or encompassing (e.g. a simple polygon) exactly the points that belong to the set. The main considerations are the extent of the changes we allow to the points, such that we can control for geographic distortion, and the criteria and measures to assess the quality of the resulting set representations.

2 Concentric Set Schematization

We define a generic concentric set schematization problem as follows. We are given a hypergraph $H = (V,S)$ with vertices V and hyperedges (sets) S, where each vertex has a (geo)spatial position in the plane. Moreover, we have a set of concentric circles C, roughly spanning the range of locations in V. The goal is to displace all vertices of H onto the circles of C, such that the resulting placement admits a good drawing of the set visualization; such as sketched in Figure 1, middle. This results in two aspects of quality: spatial quality and hyperedge quality.

Spatial quality aims at capturing how well the spatial locations of the vertices are preserved. That is, less displacement or less structural change results in a visualization that more accurately reflects the input. A standard measure is the maximum or total distance between original and displaced location of each vertex, but other measures could be used as well.

Hyperedge quality aims at capturing how well we can render the various hyperedges. We assume that we use a rendering style compatible with supports: that is, we structure our visualization based on a graph $G = (V,E)$ with the same vertices, such that each hyperedge of H induces a connected subgraph in G. For supports, typical graph drawing measures can be used, such as the total edge length and the number of intersections.

Both aspects are to be considered simultaneously in order to obtain high-quality schematic set visualizations. Though various models can be considered, we focus on allowing vertices to move only along the ray through it, originating from the circles' center. For every vertex an interval of possible circle locations is given with the input. The sets are to be represented as connecting geometries (trees, paths, or cycles). That is, we aim to compute a vertex placement together with a support of the hypergraph, such that their combination has good quality.

3 Crossing minimization

The problem of finding a support with minimal crossings bears resemblance to layered graph drawing and (multispine) book embeddings, yet is distinct with different challenges. Hence, we conjecture that the general problem is NP-hard.

Let us turn towards a simple case. We assume C contains exactly two circles, each vertex may map to either of the two circles, the hyperedges of H are pairwise disjoint, and the support is given. Moreover, we assume edges are always routed between the two circles: that is, the space where we may draw the edges is effectively the annulus enclosed by the two circles.

We define two configurations: one implies that a support edge must have its endpoints on the same circle, and one implies that a support edge must have its endpoints on different circles. To define these, consider all vertices to be ordered clockwise around the center of C, denoted as $v_1, \dots v_n$. If an edge $e = (v_i, v_j)$ of the support is "contained" within another edge $f = (v_h, v_k)$ – that is, $h < i < j < k$ – then e must have its endpoints on the same circle, to avoid intersecting f. If there is an even-length sequence of edges $e_1, \dots, e_k$, such that the first vertex of e_i is in between the vertices of e_{i-1} and the second vertex is in between the vertices of e_{i+1}, and there is an edge f with its first vertex between those of e_1 and its second vertex between those of e_k, there f must have its endpoints on different circles.

If there are no contradicting configurations, then a planar support exists, by appropriately choosing sides for each vertex. The above works on a strip rather than an annulus. To test for an annulus, we

[*]e-mail: bekos@informatik.uni-tuebingen.de
[†]e-mail: w.meulemans@tue.nl
[‡]e-mail: andre.schulz@fernuni-hagen.de

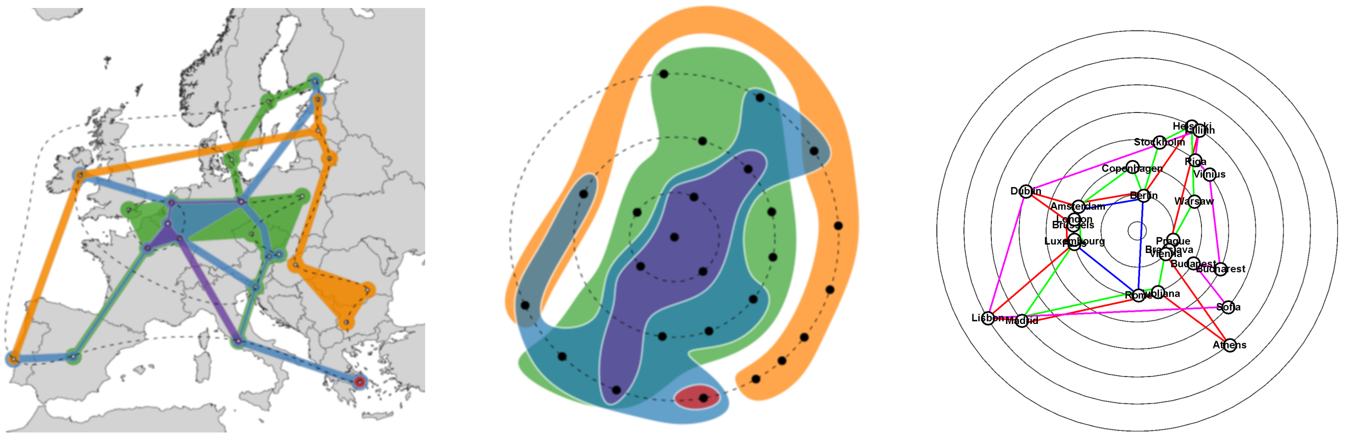

Figure 1: Three set visualizations of the same data set. Left: KelpFusion set visualization [10] on a geographically accurate base map. Middle: Manual sketch of a concentric schematic representation of the sets. Right: Preliminary result of our prototype implementation.

test every edge to have its endpoints on different circles: this cuts the annulus into a strip and we define the order based on the cut.

4 Length minimization

Let us now turn to minimizing the total edge length. We assume that the desired support $G = (V, E)$ is given: designed or already computed by another algorithm. Considering the concentric design, a support that is a union of paths or cycles obtained by connecting each hyperedge clockwise around the center may for example yield a reasonable support. We need to decide only on which circle to place each vertex. Rather than the Euclidean length, we measure the length of an edge as the difference between circle radii of its endpoints. We may ignore the angular change, as it is constant in our setting constrained by rays. Hence, we do not implicitly favor placement on smaller circles over larger circles. As such it may be a more effective measure of quality than edge length in this setting, though it may result in very long edges in extreme cases.

To minimize the radial change for each support edge, a simple linear program (LP) suffices. Assume the circles in C are indexed, sorted by their radius and that the difference in radius between consecutive circles is constant. The input specifies a range $[v_{\min}, v_{\max}]$ of indices that each vertex $v \in V$ may be placed on. We use d_e to capture the radial change of Specifically, the LP is as follows:

$$\begin{array}{rl} \text{minimize} & \sum_{e \in E} d_e \\ v_{\min} \leq c_v \leq v_{\max} & \text{for } v \in V \\ d_e \geq c_v - c_u & \text{for } e = (u,v) \in E \\ d_e \geq c_u - c_v & \text{for } e = (u,v) \in E \end{array}$$

The LP requires that the c_v variables are integer. However, we prove that the relaxation has an optimal integer solution and thus the problem can be solved efficiently. In particular, every solution to the LP has a set of constraints that are not tight (one of the two constraints for each edge $e \in E$): removing these yields an LP with the same solution, for which the underlying matrix is totally unimodular. In fact, all vertices of the feasible region induced by the original LP are integral and in bijection to the layer assignments. As a consequence the optimization problem can be solved by greedily improving a layer assignment.

Using the cycle-based support as described above, an example assignment, where each vertex is allowed to move to its two closest circles, is given in Figure 1, right.

5 Outlook

Our initial findings leave us with several interesting questions, both algorithmic in nature and on visual design of concentric set schematization. Our solutions assume that the support is given or decided previously – algorithms to decide on good supports using the flexible, yet constrained vertex placement are useful to further improve the resulting visualization. We also plan to investigate different models of spatial distortion and criteria for layout quality. Finally, We need to determine how to best route the connecting geometries to obtain an effective design of schematic set representations.

Acknowledgments

This work was initiated at Dagstuhl Seminar 19192 "Visual Analytics for Sets over Time and Space". The authors wish to thank the organizers for the inspiring workshop.

References

[1] H. A. Akitaya, M. Löffler, and C. D. Tóth. Multi-colored spanning graphs. In *Graph Drawing and Network Visualization*, LNCS 9801, pp. 81–93, 2016.

[2] B. Alper, N. Henry Riche, G. Ramos, and M. Czerwinski. Design study of LineSets, a novel set visualization technique. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2259–2267, 2011.

[3] B. Alsallakh, L. Micallef, W. Aigner, H. Hauser, S. Miksch, and P. Rodgers. The state of the art of set visualization. *Computer Graphics Forum*, 3(1):234–260, 2016.

[4] U. Brandes, B. P. S. Cornelsen, and A. Sallaberry. Path-based supports for hypergraphs. *Journal of Discrete Algorithms*, 14:248–261, 2012.

[5] T. Castermans, M. van Garderen, W. Meulemans, M. Nöllenburg, and X. Yuan. Short plane supports for spatial hypergraphs. In *Graph Drawing and Network Visualization*, LNCS 11282, pp. 1–14, 2018.

[6] C. Collins, G. Penn, and S. Carpendale. Bubble Sets: Revealing set relations with isocontours over existing visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1009–1016, 2009.

[7] K. Dinkla, M. van Kreveld, B. Speckmann, and M. Westenberg. Kelp Diagrams: Point set membership visualization. *Computer Graphics Forum*, 31(3pt1):875–884, 2012.

[8] F. Hurtado, M. Korman, M. van Kreveld, M. Löffler, V. Sacristán, A. Shioura, R. I. Silveira, B. Speckmann, and T. Tokuyama. Colored spanning graphs for set visualization. *Computational Geometry: Theory and Applications*, 68:262–276, 2018.

[9] D. S. Johnson and H. O. Pollak. Hypergraph planarity and the complexity of drawing venn diagrams. *Journal of Graph Theory*, 11(3):309–325, 1987.

[10] W. Meulemans, N. Henry Riche, B. Speckmann, B. Alper, and T. Dwyer. KelpFusion: A hybrid set visualization technique. *IEEE Transactions on Visualization and Computer Graphics*, 19(11):1846–1858, 2013.