DOI: 10.1111/cgf.14497

COMPUTER GRAPHICS *forum*
Volume 0 (2022), number 0 pp. 1–20

Computing Schematic Layouts for Spatial Hypergraphs on Concentric Circles and Grids

M.A. Bekos,[1] D.J.C. Dekker,[2] F. Frank,[3] W. Meulemans,[2] P. Rodgers,[4] A. Schulz[5] and S. Wessel[2]

[1]University of Ioannina, Ioannina, Greece
[2]TU Eindhoven, Eindhoven, the Netherlands
[3]Technische Universität München, München, Germany
[4]University of Kent, Canterbury, United Kingdom
[5]FernUniversität in Hagen, Hagen, Germany

Abstract

Set systems can be visualized in various ways. An important distinction between techniques is 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. We study layouts for set systems (or hypergraphs) in which spatial locations are displaced onto concentric circles or a grid, to obtain schematic set visualizations. We investigate the tractability of the underlying algorithmic problems adopting different optimization criteria (e.g. crossings or bends) for the layout structure, also known as the support of the hypergraph. Furthermore, we describe a simulated-annealing approach to heuristically optimize a combination of such criteria. Using this method in computational experiments, we explore the trade-offs and dependencies between criteria for computing high-quality schematic set visualizations.

Keywords: information visualization, hypergraph drawing, visualization, computational geometry, modelling

CCS Concepts: • Human-centred computing → Graph drawings; Geographic visualization; • Theory of computation → Computational geometry

1. Introduction

Various types of data or analysis results can be modelled through a system of potentially overlapping sets over a collection of elements. To facilitate understanding of the structures in such set systems, various techniques to visualize sets have been developed and evaluated by the visualization community. Complementing this, the graph-drawing community has investigated the underlying mathematical structures for drawing a set system, which is also called a *hypergraph* in this context: a generalization of a graph in which a hyperedge (set) is a non-empty set of vertices (elements), rather than a pair. In this paper, we shall adopt this terminology.

We may roughly categorize the techniques for drawing hypergraphs into two types: those that place every vertex at an associated (geo)spatial location (e.g. [ARRC11, CPC09, DvKSW12, MRS*13]; see Figure 1(left)) and those that draw hypergraphs without (using) spatial information (e.g. [EHKP15, AAMH13, LGS*14, SMDS14, MR14, RD10]; see Figure 1(right)). Whereas the former type allows to relate the set structures to the spatial dimension and hence to find spatial patterns, the latter may place elements freely and can leverage this freedom to greatly reduce the visual complexity of the resulting drawing.

Our goal is to break this apparent dichotomy and explore the possibility of finding a hybrid between these two extremes. That is, we want to find a *schematic* drawing of the hypergraph, in which vertices do not need to be positioned precisely at their spatial location. Instead, we allow for displacing the vertices to clarify set structure and reduce visual complexity, while still having a drawing that is spatially informative, albeit not fully accurate; see Figure 1(middle) for an example. This follows the general principles of schematization, which are often applied to visualize, for example, transit maps (e.g. [Rob12]) or the results of geographic analysis (e.g. [Rei15]).

The driving principle is that full detail is not necessary to understand the main or aggregate patterns in the data (e.g. distinguishing patterns between Europe and South America, in a country-level

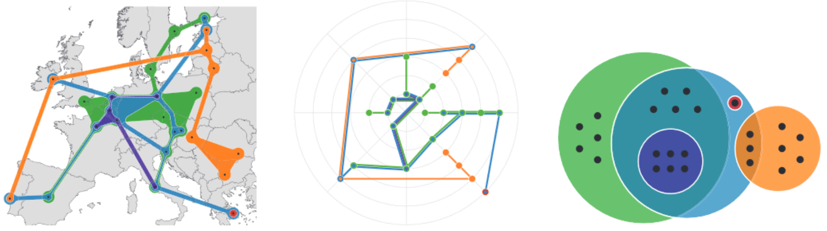

Figure 1: *Three set visualizations of the same dataset. (left) KelpFusion set visualization [MRS*13] on a geographically accurate base map. (right) An Euler diagram without spatial information. (middle) Result of our prototype implementation of a schematic drawing.*

dataset). It may even hinder a clear view of the data, as the spatial dimension often suffers from an uneven distribution, for example: the city center is more dense than the suburbs, the municipalities are smaller in densely populated areas, the density of countries in one continent is higher than in another. Deformation helps to create space in otherwise cluttered areas to ensure good legibility. See Figure 9 in Section 5.3 for example. With precisely positioned elements, Europe is close to unreadable; yet, once it is clear that the schematic displays a world map, one can identify general clusters of, e.g. Europe and South America. An appropriately schematized background map may also reinforce this perception.

Indeed, the mental map [Tve93]—a person's recollection of spatial entities and their relations—is conjectured to be schematic [Tve81] and a map that matches their mental map may facilitate recognition despite the deformation. Though schematic set visualizations may push beyond a threshold for immediate recognition, they should strive to maintain important spatial relations as to retain the benefit of (geo)spatial embedding; the degree of deformation that is applied must be balanced with the intended task of the resulting visualization. Yet, interactions or multiple views can help to arrive at the details, should this be desired. As such, schematic set visualizations can support general overview tasks in the Information-Seeking Mantra [Shn96].

1.1. Contributions and organization

We briefly explore the design space for schematic spatial hypergraphs in Section 2; we introduce the necessary notation, terminology and models that are used throughout the paper here as well. Specifically, we focus on drawings that place vertices on grids, including rectangular grids as well as concentric grids that are defined by the intersection of concentric circles and rays starting at their common center. The degree of schematization is controlled by restricting the allowed placement of a vertex to a neighbourhood surrounding its spatial location. To the best of our knowledge, spatial schematic set visualizations are novel and the concentric style specifically presents the opportunity to create iconic schematic depictions of set data.

We then turn towards the computational problem underlying simple variants of our model in Section 3. Specifically, we consider minimizing the common graph-drawing criteria of crossings, bends and total length. We show that the first two are NP-hard. Though minimizing total length is feasible for simple models, this criterion in isolation does not help to achieve the goal of clarifying structure.

In Section 4, we study the general problem of finding a good drawing, combining various criteria. We describe an implementation based on simulated annealing. We use this prototype to showcase the potential of schematic hypergraphs and present results of computational experiments that investigate its performance in Section 5. Finally, we conclude in Section 6 with a discussion of how our results can be used to support visual analysis of set systems.

1.2. Related work

Set visualization is a diverse area of research; various set-visualization methods can be seen in Alsallakh *et al*. [AMA*16]. Of particular relevance here are methods that link the vertices of each set by overlaying a line or tree-like structure based on a support [ARRC11, DvKSW12, MRS*13]. For an illustration see Figure 1(left): as is typical for these overlay methods, the vertices are fixed and the set structures are routed around them. In terms of automated layout of these visualizations, most attention has been afforded to hypergraph supports [JP87] for both fixed and free vertex locations: though some specific restricted problems are tractable [BCPS12, HKvK*18, BvKM*11], many problems are NP-hard, including general crossing minimization [BvKM*11], total edge length minimization [ALT20] or their combination for just two hyperedges [CvGM*19]. In graph drawing, there is specifically also consideration for drawing a hypergraph together with a regular graph—so called clustered drawings [WNV20]. Wu *et al*. [WNV20] allow vertex duplication for better structure, a technique also used in (non-spatial) set visualization [RD10]. However, for the spatial setting, requiring the vertices to be close to the original location counteracts the usefulness of such duplication.

Schematic maps have been successful in various applications by abstracting spatial relations to a minimum functional level [Rei15], thereby clarifying and emphasizing structure in data while not disregarding (geographic) space; see Wu *et al.* [WNT*20] for a recent survey of the design, human and computational aspects. The automated layout of schematic maps has concentrated on the layout of metro maps, for instance, by multi-criteria optimization [SRMW11], by linear programming [NW11] and force-based methods [CR14]. There is work on drawing metro maps using curves (e.g. [FHN*12, vDvGH*14]) following the argumentation of Roberts to consider alternatives to the dominating octilinear designs [Rob12]. Barth [Bar16] describes a concentric-circles style layout algorithm for metro maps, where simulated annealing is used to minimize area. Subsequent work further investigates the use of this 'ortho-radial' style [NRW19, NR20].

Another strand of research investigates schematization of (geographic) outlines such as countries, e.g. [vG16, Meu14]. A grid map [MSS21, Sli18] can be seen as an extreme form of schematization, where every region (or element) is turned into a square in a grid; theoretical considerations for such systems in connecting cells can be found [vGKvK*17], though the results have not (yet) been shown to be efficacious to visualize sets. It has been shown that such grid maps allow better perception of local details, though other tasks may be supported less well [Sch21].

An alternative is to distort the base map (and the spatial hypergraph) before computing a drawing for the hypergraph. Such focus-context maps can be computed automatically [vDH14] and have been used to support schematization [vDvGH*13]. However, such techniques are driven by map structure, rather than hypergraph structure, or focus on interactivity to steer the distortion [JM03]. Given a fixed support, these techniques may be applicable directly—however, distorting the space may imply that other supports are better used. While this is possible with our method, this is not readily supported by such focus-context techniques currently.

There is also a body of work on using a metro-map metaphor, such as the MetroSets system [JWKN21], where elements are mapped to stations and sets to metro lines. To emphasize, such work leverages a metro-map appearance to visualize *non-spatial* hypergraphs, and thus allows free vertex placement; the main distinction to actual metro-map algorithms is the choice in (often path-based) supports for each set. However, this is done in a pipelined manner: first, a support is computed, which is subsequently drawn using metro-map drawing techniques. Regular metro-map approaches create for spatially informative layouts but are restricted to the given metro lines (the 'support'). Hence, our work can also be interpreted as trying to bridge these two settings, one in which the support remains flexible but the result must be spatially informative as well. Moreover, such work predominantly targets octilinear designs as the 'prototypical metro map'. In contrast, we focus on (not necessarily octilinear) concentric-circles layouts.

In conclusion, to the best of our knowledge, there is no prior work in computing schematic layouts for spatial hypergraphs. The one exception here is our own preliminary work [BFM*19]. In that paper, we sketch two results: the linear program for length minimization (here, mentioned in Section 3) and an argument that an overly simple case is in fact tractable: checking whether a given support can be drawn without crossings when there are but two concentric circles. In this work, we instead show that the general problem of crossing minimization, even with some relaxation, is NP-complete (Section 3).

The lack of prior work also implies that we cannot readily compare our results to other techniques: our algorithm focuses on achieving a good layout with limited movement of the hypergraph vertices, while also allowing for a good support to show the hyperedges. This combination requires us to make at least some concession on the complexity of the optimized measures, compared to more involved techniques for fixed vertices such as [DvKSW12, MRS*13]. Of course, once good locations have been decided, alternative rendering techniques or even fine-tuning the chosen support for the now-fixed locations can in principle be done easily in a post-processing step. We focus our work on the efficacy of the schematization step itself in providing a trade-off for visualizing the spatial dimension and the structure of the hypergraph; as such, we do not consider this post-processing in our work.

2. Problem Exploration and Definitions

2.1. Hypergraphs

A *hypergraph* $H = (V, S)$ models a system of intersecting sets and is defined by a set of vertices V and hyperedges $S \subseteq \{X \mid X \subseteq V, X \neq \emptyset\}$. In a *spatial* hypergraph, each vertex $v \in V$ has a (geo)spatial position in the plane. We identify v with this location. In this paper, hypergraph always refers to a spatial hypergraph.

2.2. Supports

Contrasting the drawing of a regular graph, in which every edge is visualized by some linear geometry connecting its endpoints, hypergraphs admit more flexibility. As hyperedges are unordered, there are many ways to connect the vertices of the hyperedge. The *support* [JP87] describes the connectivity using a regular graph. Specifically, a graph $G = (V, E)$ is called a *support* of $H = (V, S)$ if the induced subgraph $G[h]$ of each hyperedge $h \in S$ is connected. We refer to a connected graph on the vertices of $h \in S$ as a *support* of h. Taking the union over a support for each hyperedge gives a support G for H. Note, however, that the induced subgraph $G[h]$ in this union may include more edges than in the original support for hyperedge h: the support of other hyperedges may include edges that were not part of this original support, but do connect vertices in h. In other words, a support of h may be a connected subgraph of $G[h]$: there may be different ways in which to connect the vertices of h, but the induced subgraph is unique. A drawing of a support can be used to structure the visualization of a hypergraph: for example, both [DvKSW12] and [MRS*13] implicitly compute a support using different criteria.

In our prototype, we allow slightly more general supports. Specifically, we allow the addition of new vertices called *anchor points* by the algorithm. The support is then a graph $G = (V \cup A, E)$ on the vertices V and anchor points A, in which $G[h \cup A']$ is connected for every hyperedge h for some subset A' of A. That is, they may connect through these anchor points. These anchor points may have a degree greater than two, allowing them to act as Steiner points to reduce the total length of the drawing. Furthermore, these anchor points can also represent bends in the drawing.

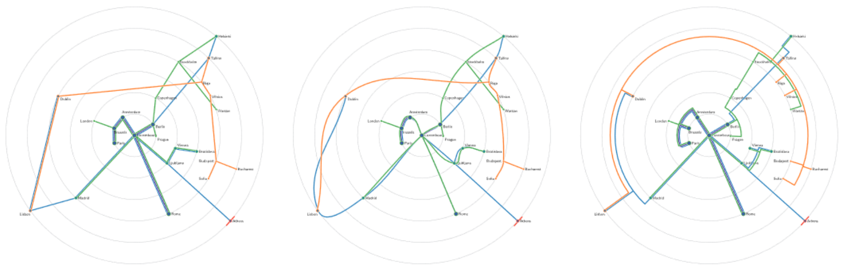

Figure 2: *Various ways to route the edges of the same support: straight lines (left); smooth curves (middle); concentric (right).*

2.3. Deformation models

To arrive at a schematized drawing of a hypergraph, we allow the drawing of its support to displace the vertex positions. To still maintain a spatially informative visualization, we do not want to displace arbitrarily. Here, we focus on placing the vertices on a structured pre-defined set of *admissible positions*, to help the schematic look and feel, as well as to ensure a minimal distance between vertices. We call an assignment from vertices to positions *valid* if no two vertices are assigned the same position.

A simple structure would be a *rectangular grid*, with admissible positions defined by $X \times Y$ where X and Y define allowed x- and y-coordinates respectively. We specifically also consider a *concentric-circles* layout. This is defined by a set C of circles with a common center, and a set R of rays emanating from the common center. The admissible positions are the intersections between the circles and the rays. We define the rays based on the input positions or using evenly spaced rays; the latter we refer to as a *concentric grid*. Grids provide a simple control over distances between nodes and thus naturally create space for drawing the support.

To control displacement, each vertex can be assigned a subset of the admissible locations, such as those within a threshold distance from its original position. We specifically also consider concentric-grid layouts in which each vertex defines a ray in R and the vertex must be assigned to one of the admissible positions defined by its ray (or a subset thereof). In this *radial displacement model* vertices move only radially with respect to the center. The main idea is that such a model may allow an observer to easily find vertices, assuming their relation to the chosen common center is known. That is, the more structure exists in the displacement, the easier it may be to use the spatial dimension. In a grid layout, this would translate to allowing vertices to move only vertically.

2.4. Drawing styles

The deformation model determines where we can place the vertices, and the support prescribes which connections to draw. To fully define the schematic drawing of the spatial hypergraph, we need to decide on how to draw these connections. A simple technique would be to use straight lines, or smooth curves by decomposing the support of each hyperedge into a set of paths. For a grid layout, we can aim to compute, for example, octilinear connections; for a concentric-circles layout, we can utilize geometric form (e.g. concentric arcs and spokes) also in the drawing of the connections (ortho-radial drawing). See Figure 2 for some examples. We place the hyperedges using the same edge of the support side-by-side, but these connections can be drawn according to various styles; see e.g. [DvKSW12] for alternatives drawing styles. As our focus lies with determining the structure of the drawing, we do not further investigate the effect of drawing styles.

2.5. Quality criteria

Perhaps the main question is how to assess the quality of a schematic drawing of a spatial hypergraph H. By using supports, this effectively reduces to assessing the quality of the support drawing. As such, we may identify many criteria that are readily inspired by the graph-drawing literature; see e.g. [DETT98, KW01].

P.1 *Crossings*: The number of intersections between the edges of support G should be small.
P.2 *Bends and anchor points*: A drawing should have few bends and anchor points, to make connections easy to follow.
P.3 *Displacement*: each vertex of H should be assigned to a position that is close to its spatial location. We measure the Euclidean distance, but other distances can be considered, especially for a concentric-circles layout.
P.4 *Vertex-edge resolution*: The distance between a vertex and its non-incident edges should be sufficiently large as to avoid the impression that the vertex is actually incident to the edge.
P.5 *Total edge length*: Following Tufte's rule of minimal ink [Tuf01], the support should be drawn with small total edge length. As we measure this on the support of H, an edge is counted only once, even if it occurs in the support of multiple hyperedges. Consequently, we would like hyperedges to use the same edges, if possible.

P.6 *Detour*: We want to prevent bends and anchor points from significantly increasing the geometric path length between the connected vertices.
P.7 *Octilinearity*: Ensuring that the edges are drawn using octilinear segments gives a schematic appearance and visually conveys that the representation is not a spatially accurate one. Further, it may improve the continuity of edges at a vertex.

These criteria generally conflict, in the sense that improving one may require deterioration of another. As such, a good schematic drawing of a hypergraph must make a trade-off between these.

There are various other potential criteria to measure the quality of the support. For example, KelpDiagrams [DvKSW12] aim to compute a support for a hyperedge that has low dilation, whereas KelpFusion [MRS*13] aims to capture the 'shape' of the locations and allow for more hull-like representations. We leave the integration of such concerns for schematic hypergraphs to future work, as we focus on using a tree support for each hyperedge.

In our above exposition, we effectively equate the 'spatially informative' nature of the schematic drawing to displacement P.3. However, space is generally multi-faceted: beyond location, we can for example also consider maintaining distances or directions between vertices, or neighbourhoods. In the context of grid maps, displacement has been shown to be a good proxy for such other considerations [EvKSS15, MDS*17, MSS21]. We assume a similar relationship here, using displacement as a general proxy for other facets of the spatial dimension as well, in order to simplify algorithmic consideration.

3. Computational Complexity

Here we consider the complexity of computing a layout under the radial displacement model and different optimization criteria. Even in this simplified form, the problems tends to be computationally complex—the more general form of grids with admissible positions nearby are not expected to make the problem considerably easier: in the most general case, the grid and distances can likely be configured such that the complexity results below are still applicable. Though we primarily focus on concentric-circles layouts, the results equally apply to rectangular cases, where vertices are allowed to move only vertically. We refer to Appendix A for proofs.

3.1. Crossing minimization

We first consider the problem of minimizing P.1, the number of crossings using a straight-line drawing of the edges of the support. This problem bears resemblance to layered graph drawing, though we can select edges for the support. However, when every hyperedge contains two elements, we have actually a regular graph. In fact, crossing minimization in two-layered bipartite graphs is an NP-hard problem that can be phrased in our model [EW94]. If we require validity, this correspondence is immediate, even for intervals of admissible positions along the rays.

Even if we allow vertices to map to the same admissible position but do not restrict admissible positions to intervals, this ICM (invalid crossing minimization) problem is NP-complete.

Theorem 1. *ICM is NP-complete.*

3.2. Bend minimization

We now turn to bend minimization. Specifically, we consider an edge-drawing style that follows the geometry defining the admissible positions: every piece of a drawn edge is either a circular arc on one of the defining circles, or a straight segment coinciding with some ray emanating from the common center. We prove that bend minimization in a concentric-circles layout under radial displacement is NP-hard, even if each vertex has a consecutive interval of admissible positions along its ray. We refer to this problem as CCCL (consecutive concentric-circles layout).

We are given a hypergraph $H = (V, S)$, a non-negative integer $b \in \mathbb{N}$, and a set $C = \{c_1, c_2, \dots, c_t\}$ of concentric circles, numbered by increasing radius such that c_{i+1} contains c_i for all $i < t$. Moreover, each vertex $v \in V$ has an associated interval $I(v) = [i, j]$, which defines the admissible positions for v by the intersections of its ray with the circles with indices in the given interval. We ask for a drawing of H with at most b bends. The drawing γ_h of each hyperedge h is a curve following the above style: each bend of γ_h must share a ray or a circle with the previous and next bend or vertex along γ_h. Further, we disallow γ_h to pass through vertices that are not in h, to ensure that the vertex-edge resolution is not zero. Note that we choose paths as the support of the hyperedges. However, as we will see, our reduction uses only hyperedges of two or three elements. In this case, any support minimizing the number of bends must be a path: thus our result generalized to arbitrary supports. The above CCCL problem is computationally complex.

Theorem 2. *CCCL is NP-complete.*

3.3. Length minimization

We now consider minimizing the total edge length. For fixed positions, finding a support with small total length is NP-hard [ALT20, CvGM*19]. Allowing some flexibility in vertex placement only generalizes this problem and thus is NP-hard as well. However, we can solve this problem efficiently, if we assume that a support $G = (V, E)$ is given—designed or computed by an algorithm beforehand. 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. Using a (Euclidean) minimum spanning tree for each hyperedge yields an approximation of total edge length for fixed positions [ALT20].

We seek to decide an admissible position for each vertex such that the total edge length is minimized. Considering the concentric design, we measure edge length not in Euclidean distance, but rather using a Manhattan distance interpreted in radial coordinates: absolute difference in radius plus the absolute difference in angle. Generally, this requires some meaningful conversion to combine angles and radial distances. However, since we consider the radial displacement model, the angular components are fixed and we can focus on minimizing the difference in radius (of the assigned circles for the endpoints of an edge). This choice ensures that we do not implicitly favour placement on smaller circles over larger circles: placing vertices closer towards the center may reduce distances but also counteracts the schematic idea of spreading out the vertices for better visibility (see also the discussion below). As such, it may be a more effective measure of quality than Euclidean in this setting, though it may result in very long edges in extreme cases.

A simple linear program suffices to solve this RDM (radial difference minimization) problem, as captured by the theorem below.

Theorem 3. *RDM is solvable in polynomial time.*

However, minimizing total edge length in isolation for a fixed support does not readily give good results. First, we must assume that vertices on the same ray have disjoint intervals, otherwise the result may in fact be invalid. For points in general position, this does not occur, but vertices may be placed close to each other. Furthermore, observe that the implied drawing using concentric arcs and radial lines does not ensure a minimal distance between vertices and non-incident edges: the vertex-edge resolution may also be zero, even if the drawing is valid. But primarily, this type of approach would have a tendency to simply move vertices towards each other: suppose all intervals overlap, then the optimal solution is to place all vertices on the same circle, yielding zero radial difference. This removes most spatial information and place vertices very close to each other. If such strong reduction is desired, it may be more beneficial to directly compute a Necklace map [SV10].

3.4. Other criteria

Optimizing displacement can be done efficiently, as it is a weighted point-set matching problem, as also found in computing grid maps [EvKSS15]. However, this does not consider the hypergraph nor its support at all and as such is not immediately useful in schematizing hypergraphs. The other criteria—vertex-edge resolution, detour and octilinearity—are of little relevance in isolation. For example, a perfect drawing in terms of detour is easily possible by not using any anchors or bends (and any assignment to admissible positions). If we are to constrain other aspects, such as the number of crossings, this effectively means that the computational complexity of that criteria is immediately restrictive.

4. A Simulated-Annealing Approach

The previous section establishes that even focusing on a single quality criterion already tends to yield problems that are computationally complex and the results may still not be quite satisfactory, since such an approach does not make a trade-off between the various criteria. Thus, we describe here a proof-of-concept implementation that automates the process of schematizing a hypergraph while allowing for such trade-offs. Our prototype is open source, and can be found online (https://github.com/stenwessel/setschematics).

The core of our implementation is a simulated-annealing algorithm, a common metaheuristic approach for optimization problems which can escape local optima; note that this technique has also been applied for drawing regular graphs, e.g. [DH96]. The main idea is that the simulated-annealing algorithm iteratively tries a modification to the current solution. If this improves the quality, then it is accepted; if it does not improve, then may still be accepted with some random chance, which depends on the 'temperature' T and the quality of the new and old solution. A cooling schedule reduces the temperature while the algorithm runs, thereby decreasing the chances of moving to a worse solution. It has been shown that the simulated-annealing algorithm has a theoretical stochastic convergence to a global optimum state, provided with an infinite cooling schedule of infinitely small cooling steps [DCM19]. This is of course not feasible in practice, but indicates that performing more iterations tends to lead to a better solution.

The problem that our simulated-annealing technique aims to solve is to find a schematic drawing of a hypergraph $H = (V, S)$ on a concentric (or rectangular) grid, where each vertex (and anchor point) is allowed to be assigned to any admissible position in the grid. The only hard requirement is that the drawing is valid. Furthermore, we assume a straight-line drawing of edges, but note that we can introduce bends via anchor points. Below, we describe the necessary components of our simulated-annealing implementation.

4.1. Representation

A solution to our problem consists of three components: (i) a set A of anchor points; (ii) a mapping of vertices V and anchor points A to admissible locations of the concentric grid; (ii) a support per hyperedge h possibly including anchor points, this support is always a tree. The union over all supports per hyperedge gives the general support $G = (V \cup A, E)$ for H.

Simulated annealing requires an initial solution; we compute this as follows. We start with an empty set of anchor points and we greedily map each vertex of H to the nearest admissible position that does not have an assigned vertex yet. Then, we determine the support of each hyperedge h by computing its Euclidean minimum spanning tree on its vertices at their assigned positions.

4.2. Measuring quality

We measure the quality as a weighted sum of the quality criteria P.1–P.7 (see Section 2). The weights of the criteria can be altered to facilitate a trade-off; with our experiments in Section 5 we determine good default values. Hence, we obtain the following measures, where d denotes the diagonal of the bounding box of the input locations:

P.1 The number of crossings divided by $|E|^2$.
P.2 The number of anchor points divided by $|V|$.
P.3 The total displacement is the Euclidean distance between a vertex's spatial location and its assigned position, summed over all vertices. This is divided by $|V| \cdot d$.
P.4 For a vertex $u \in V$ and an edge $(v, w) \in E$ that is not incident to u, we compute the shortest distance δ between u and the segment vw. The resolution penalty of this combination is $\max\{0, \Delta - \delta\}$, where Δ is a fixed threshold. We sum the penalties over all combinations and divide it by $|V| \cdot |E| \cdot d$.
P.5 We divide the total edge length by $|E| \cdot d$.

P.6 We measure this as the total edge length of the support as drawn, divided by the total edge length if we were to draw that same support but with every bend and anchor point removed. As anchor points may connect more than two vertices of H, we use the length of the Euclidean minimum spanning tree as the replacement length. We divide this measure by the number of anchor points; if there are no anchor points, this measure is 0.

P.7 For each edge $(u, v) \in E$, we compute

$$\left|\sin\left(4\arctan\frac{|y_u - y_v|}{|x_u - x_v|}\right)\right|,$$

which expresses how far (u, v) is from being octilinear [SRMW11]. We sum all these values and divide it by $|E|$.

4.3. Modification

To modify a current solution to a new solution, we choose one of the following actions, uniformly at random.

Change anchors. We choose with equal probability to add or remove an anchor point. If there are no anchor points to remove, this operation always adds an anchor point. To add an anchor point, we pick an edge with crossings (if one exists) with probability 0.5, as such edges are often useful to reroute; otherwise, we pick an arbitrary edge (which may or may not have crossings). All hyperedges using this edge in their support will be rerouted via the new anchor point. To determine the position of the anchor point, we select one at random. All locations within an ellipse through the endpoints that is twice as wide as the edge is long have a five times higher chance of being selected than those outside of this ellipse. This helps to steer selection to a lower detour without eliminating possibilities. We remove only anchor points of degree 2, by adding an edge connecting its two neighbours to all supports using this anchor point.

Change positions. We select a vertex or anchor point uniformly at random, such that it has at least one free neighbouring admissible position in the grid. Of these free neighbouring positions, one is selected at random.

Change support. For a random hyperedge h, we choose two vertices in h that are not adjacent in its support. We add this edge to its support and remove a random edge along the path between these vertices in the old support. This ensures that the support remains a tree. Note that this changes only the support for h.

By maintaining the number of crossings for each edge, we implement the above operations in $O(n + a + k + p)$ time, on a hypergraph with n vertices, a anchor points, k hyperedges and p positions in the grid. The point of efficiency hence lies in ensuring that the quality measures are updated efficiently. Our support as $O(N)$ edges at any time, where $N = O(nk)$ is the sum over the cardinalities of all hyperedges. Hence, computing the measure for P.4 takes $O(nN)$ time, using the trivial algorithm. Initializing the measure for P.1, takes $O(N \log N + C)$ time where $C = O(N^2)$ is the actual number of crossings. To update P.1, we compute the change of the number of crossings for the $O(n)$ edges that change with an operation of the drawing (a vertex of degree $O(n)$ is moved); this gives an update time of $O(nN)$. In practice, we expect much fewer edges to change with a single operation. All other measures are easily updated in time that is linear in the number of changing edges.

4.4. Cooling schedule

We use a linear cooling schedule derived from the iteration count. The probability of accepting a new solution with quality q' from a current solution with quality q is 1 if $q' < q$ (note that lower values for q are better) and $\exp(-(q' - q)/T)$ otherwise. This follows standard practice for such probabilities.

5. Evaluation

To evaluate our simulated-annealing method, we ran quantitative experiments to understand how the weights influence the results and how the quality measures interact. At the end of this section, we also discuss example schematic drawings qualitatively.

5.1. Data

We use the following datasets for this evaluation, which are also available from the repository of our implementation.

Toronto (full). Ninety-four restaurants in Toronto where the 23 hyperedges represent some categories of food, price and rating.
Toronto (filtered). Seventy vertices and three hyperedges, filtered from the full version.
Europe. Dataset of 25 countries of the European Union represented by their capitals with five hyperedges indicating the Eurozone, the EEC and three levels of economic welfare.
MLB Cities. Twenty-five US cities with one or more Major League Baseball teams, with eight hyperedges indicating the leagues and divisions.
World Cup. Twenty-nine countries that participated in three soccer world-cup tournaments, with six hyperedges representing the hosting countries, top 16 per year, finalists and champions.

5.2. Quantitative evaluation

For each dataset except World Cup, we construct a 30-by-30 rectangular grid constructed from the bounding box of the input vertices. We opt for the rectangular grid, as this makes the octilinearity criterion more natural and matches more intuitively with the Euclidean distance used in our implementation. Using this grid, we ran the simulated-annealing algorithm for 100, 000 iterations, with different weight settings. As randomization is involved, we repeat each trial (dataset-weights combination) 10 times. Though computational efficiency is not our primary concern, we mention that for these datasets, running 100, 000 iterations takes roughly 7–15 s on a standard laptop with a 2.6 GHz Intel i7-9750H CPU.

5.2.1. *Weights*

To understand the influence of the weights for the various criteria, we wish to vary these weights and investigate the resulting quality. To do so, we first need reasonable weights as a baseline, since the entire parameter space is too large to efficaciously explore experimentally. Based on informal trials with visual inspection, we arrive at the following rationale for the highest weights and set the default weights as displayed in Table 1. We observe that, though crossings are not the ultimate goal, clearly avoidable crossings are undesirable

Table 1: *Default weights of the quality measures.*

Measure	Weight	Measure	Weight
P.1: Crossings	100,000.00	P.5: Total length	123.25
P.2: Anchor points	2.50	P.6: Detour	10.00
P.3: Displacement	5312.00	P.7: Octilinearity	29.00
P.4: Resolution	0.85		

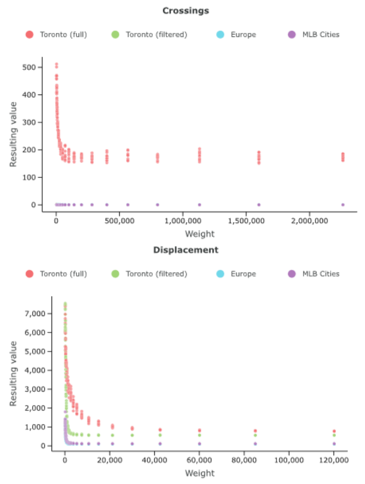

Figure 3: *Varying the weight of crossings or displacement, showing the resulting value of the same (unweighted) criterion.*

and we would like to encourage the use of anchor points to route edges to avoid such if reasonably possible. As such, P.1 achieves a very high weight. This is followed by a high weight for displacement (P.3) to ensure that the drawing is indeed spatially informative. We do not want needlessly long edges and thus the next weight is total edge length.

To validate these weights, we ran the experimental setup, varying each weight exponentially in an interval around the default value, while keeping the other weights fixed. The results are displayed in Figure 3, which plots the unweighted value of the quality measure as a function of the corresponding weight; see Figure B.1 in Appendix B for the full figure. As to be expected, the criteria decrease (improve) as the weight becomes higher, suggesting that the simulated-annealing approach can adequately deal with this criteria. Furthermore, we observe that, for crossings, displacement and detour, the values converge readily, and are effectively stable at the chosen default weights. As increasing the weights further will have little influence, this implies that our aim of prioritizing crossings and placement has been achieved. The variation in the other weights still communicates that these criteria are not ignored altogether, at the expense of these high-weight criteria.

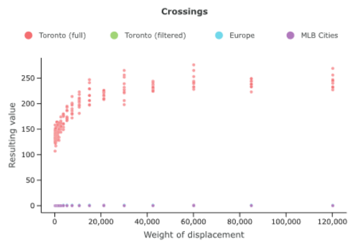

Figure 4: *Effect of displacement weight on crossings.*

5.2.2. *Quality criteria*

We now consider potential dependencies between quality criteria included in the objective function. For many quality-measure pairs, changing the weight of one has little to no effect on the other. In a few cases, however, we see that a dependency exists between quality measures. Refer to Figure 4 where varying weights of one criterion are plotted against the resulting value of the other measures; see Figure B.2 in Appendix B for the full figure.

In the topmost charts, we see that a higher weight for vertex displacement has an adverse effect on the crossings and octilinearity measures. As to be expected, this signals a trade-off between displacement on the one hand and minimizing crossings and octilinearity on the other hand. The bottom chart shows a clear dependency between total edge length and detour, suggesting that total edge length readily improves detour. This also explains why detour converges quickly in the results described earlier.

To further detect whether criteria are redundant, we set one or more weights to zero, keeping the others at the default values. Figure 5 shows the impact on the original objective function (i.e. *all* measures at their default weight). We see very little effect, except for the two highest-weight criteria: crossings and displacement. By disabling multiple criteria, we do see an effect, suggesting that these measures do contribute to finding a good solution.

We also look at how setting a weight to zero affects that specific quality criteria. Figure 6 shows the results; Figure B.3 in Appendix B is the full figure. For all quality measures except for detour and anchors, we see a clear improvement (decrease of the objective function) when the quality measure is enabled. For vertex-edge resolution, this effect is rather small—we attribute this to the small weight that this measure has in the default values. For detour, however, the behaviour is erratic: some instances improve, others deteriorate. For anchors, we even see that the measure increases slightly for some instances. Both effects are likely caused by their

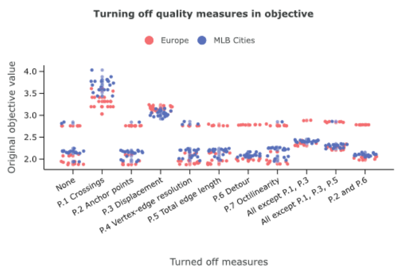

Figure 5: *The effect on the objective function when turning off optimization of specific measures in the simulated-annealing process.*

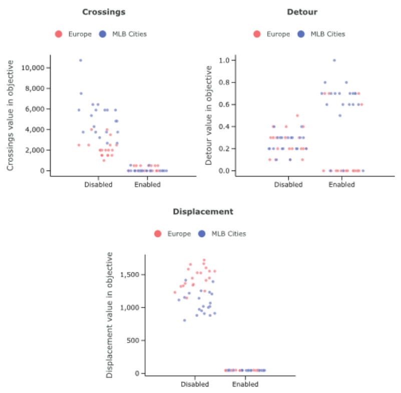

Figure 6: *The effect of enabling and disabling optimizing a quality measure on the quality measure itself.*

correlation with total edge length. As such, it may be beneficial to remove these criteria from the objective function to obtain more reliable results. The influence of disabling both criteria on the overall objective (with all default weights) is in Figure 5: the results slightly improve.

5.3. Qualitative evaluation

Figure 1(middle) shows our result on the Europe dataset. Given the model we are using, this seems to be quite an effective solution: the spatial structure is largely maintained, but the overall drawing has fairly low visual complexity and a schematic appearance. We do observe that the vertex resolution is quite poor—the grid is too coarse to alleviate it without sacrificing other criteria too much. This

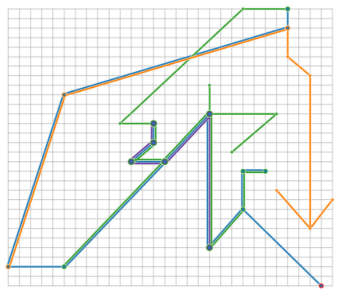

Figure 7: *Result of the Europe dataset on a rectangular grid.*

problem may hence be resolved by refining the grid, but also by applying other styles of drawing the edges. Figure 7 shows this dataset in a rectangular grid; the stronger octilinearity may help in achieving a schematic look and feel, and we see a clear trade-off being made to improve octilinearity in favour of the total edge length (e.g. the vertical orange edge on the right), but not for crossings (consider the more octilinear blue-orange connection between the left-bottom and the righttop). Though this map looks more like a 'standard' schematic metro map, establishing whether this is truly advantageous for users in our setting is an open question.

Figure 8 shows the initial layout of our simulated-annealing method as well as the eventual result for the MLB Cities dataset, both on a circular and rectangular grid. We observe in both cases that our approach quite readily improves the drawing, effectively reducing crossings. We do observe that there are some alternative choices to be made for the edges of the support. For example, in the circular result, the two blue edges on the left connect to the leftmost vertex, whereas we could improve the drawing possibly by connecting them to the vertex immediately to its right. This reduces total edge length slightly, though it may reduce octilinearity. However, the importance of such measure depends on the visual style.

For our third example, we consider the World Cup dataset; see Figure 9. This dataset is particularly challenging for spatially accurate methods as shown by the KelpFusion visualization. The culprit is the high number of countries in Europe—with a fairly small spatial extent on a global scale. Here, schematization can be particularly useful, and our result nicely pulls apart the counties in Europe and provides a far more readable diagram, one that is still spatially informative. In the rectangular grid result, we do see another instance of poor vertex resolution. This suggests that this may need to receive a slightly higher way to be avoided, or that routing of edges may need to be done in a final post-processing step.

Finally, we consider a large dataset to test the scalability of our method. To this end, we use census data of the 388 Dutch municipalities in 2017 to define seven sets on population density, water surface and dominant age groups. The result is shown in Figure 10, computed using 50,000 iterations in about 45 s. The

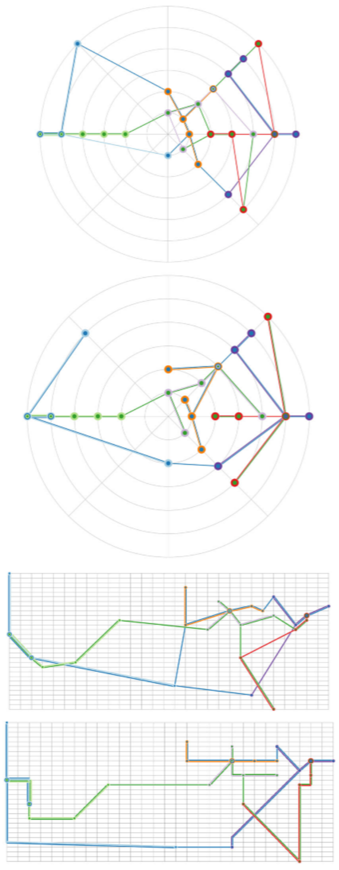

Figure 8: *MLB Cities. (top) Initial layout and result on a concentric grid. (bottom) Initial layout and result on a rectangular grid.*

octilinear, schematic nature features quite strongly, and we can see patterns here, such as the blue set (municipalities with a large percentage of water surface) following the coastline. As we focus on computing the structural connectivity of the diagram, there is room for improving this visualization, for example by using enclosed ar-

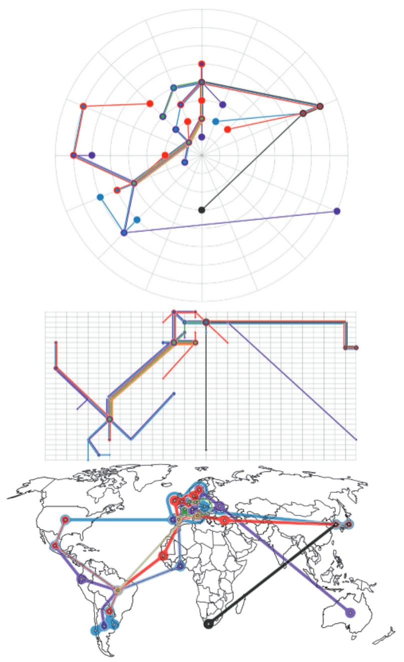

Figure 9: *World Cup. (top) Result on a concentric grid. (middle) Result on a rectangular grid. (bottom) KelpFusion visualization [MRS*13] with the original locations.*

eas for dense areas of a set (e.g. [MRS*13]) or possibly by option to not draw all connections to reduce crossings (e.g. [GCH*21]).

6. Discussion and Future Work

We studied the problem of drawing spatial hypergraphs, while allowing vertices to be displaced to clarify the set structure. The results of our evaluation shows that our simulated-annealing approach can indeed handle this adequately, in spite of the theoretical computational complexity of simplified versions of this problem. We have focused on the structure that underlies a visualization of the set system, but these can be readily used with rendering styles such as that employed by KelpFusion (nesting edges) or by metro maps (side-by-side rendering).

Complemented by adequate rendering and interaction, our algorithm supports the analysis of set systems in such a way that the

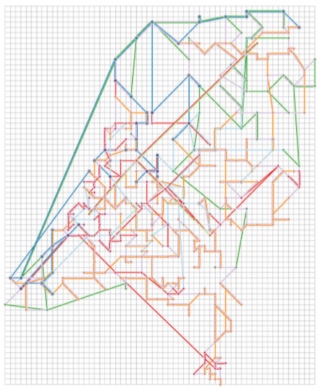

Figure 10: *Result of a municipalities dataset on a rectangular grid.*

spatial dimension does not cause unnecessary clutter, but is not fully discarded either. Zooming into the graphic may also, for example, refine the grid, thereby allowing a more accurate visualization. The simulated-annealing algorithm could be refined to repeatedly modify the current visualization to allow for this type of interaction. This would save on computation time as well as increase the stability between the visualizations at different zoom levels.

This leaves us with several avenues for interesting future work.

6.1. Design

We would like to emphasize that our work concentrates on the support, the structure underpinning various set-visualization methods. The question is which kind of rendering styles are effective, especially for the concentric circles layout. We have only briefly shown some options in Section 2, but not treated these further. A broader consideration of the design space here is out of scope for this work, but would be interesting and necessary to move to a fully fledged set-visualization technique, based on schematic hypergraphs.

Moreover, we may want to visually communicate the deformation of the spatial dimension. The primary question is how to do this most effectively. We may again take inspiration from metro maps where rivers or fare-zone boundaries give some spatial reference.

6.2. Evaluation

We focused on validating our simulated-annealing algorithm using computational experiments. In future work, we would like to complement this with a case study and human-participants study, that delve into the questions of what kind of patterns schematic drawings of hypergraphs help to uncover, to what degree the set structure is actually clarified, and to what degree the spatial dimension can still be used to assess spatial patterns. This, however, moves beyond the scope of this paper where we study the underlying structure – such studies would rely specifically also on answering the design questions above.

6.3. Computation

Alternative graph drawing methods could be applied to produce spatial hypergraphs. Whilst our simulated annealing works adequately on many datasets, applying other techniques may prove a interesting avenue of research. For example, spring embedder methods [Ead84] or more general randomized algorithms based on the vertex movement paradigm [RRRW18], could be adjusted to produce schematic drawings of spatial hypergraphs.

Our method also assumes that the admissible positions are specified beforehand. For full automation, it would be useful to automatically find an effective grid for a given hypergraph, by deciding whether a concentric or rectangular grid is more suitable, and by estimating the parameters of such a grid. It is reasonable to expect that this choice depends on the (spatial) structure of the data: a dense center with sparse periphery may naturally lean towards a concentric grid, whereas a hypergraph with different dense areas may better be schematized on a rectangular grid. We may even consider more adaptive forms where grids are forced onto dense areas but not on those areas that are sufficiently sparse as to allow an effective no-deformation rendering.

Depending on the eventual design of how to render the actual support, we also obtain different considerations for the computation. For example, how do we effectively include the bends that a concentric-line routing style induces? If we go for a side-by-side rendering of the hyperedges that use the same edge of the support, how do we integrate choosing a good order with the computation of the support and its drawing?

Finally, the interactivity alluded to above may warrant attention, in determining how effective such refinements indeed are. That is, how much computation time is necessary to update a drawing for a finer grid? Transitioning between drawings of set visualizations [MWTI19] may be useful to understand the layout changes; but in this use case, the algorithm controls the resulting drawing and thereby also the necessary changes—this control may alleviate the need for transitions, or make them easier to compute. To understand this control, we need to understand the stability of our method: does the algorithm inherently maintain the main structures of the support, or should this be controlled more explicitly? Finally, seeing only a portion of the map may break some of the desired properties: for example, are all nodes in the viewport in fact still connected through the support, or does this rely on connections outside of the viewport? Does this need to be addressed through the local structure or would a design of the off-viewport information be more effective?

Acknowledgements

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.

Appendix A: Proofs

Theorem A.1. *ICM is NP-complete.*

Proof. Assume that $G = (U \cup V, E)$ is bipartite graph with partitions $U = \{u_1, \ldots, u_m\}$ and $V = \{v_1, \ldots, v_n\}$. Assume w.l.o.g. that $m > n$. We create m^2 circles centered at the origin, denoted by c_j in increasing radius. We position U and V such that each partition defines precisely one ray. As G is bipartite, each edge has one endpoint on each ray. A vertex v_i or u_i can be placed only on one of the m circles c_j with j mod $m = i - 1$. This implies that no two vertices of one partition can be assigned to the same circle. We have enough circles such that every permutation of the vertices on each of the rays is realizable. Every bipartite two-layered layout of G corresponds to a concentric-circles layout of G under the radial position model; see also Figure A.1. Hence, minimizing crossings is NP-hard.

Since we can easily count in polynomial time the number of crossings in a given assignment of vertices to admissible positions, the problem is also in NP and therefore it is NP-complete. □

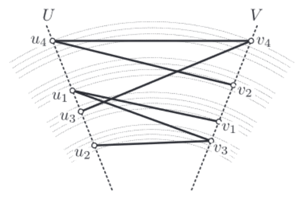

Figure A.1: *A two-layered bipartite layout as an instance of our concentric-circles layout for $m = 4$. The concentric circles hence are shown as four groups of four circles each.*

Theorem A.2. *CCCL is NP-complete.*

Proof. The problem is in NP, as for a given vertex-circle assignment (which would be the certificate), we can minimize the number of bends of every hyperedge independently of the others in polynomial time.

The NP-hardness is shown via a reduction from Vertex Cover [GJ79]. Let an instance of vertex cover be given by a graph $G = (V, E)$ and an integer k. For convenience we denote the vertices of G by $v_1, v_2, \ldots v_n$ and its edges by $e_1, e_2, \ldots, e_m$. The reduction generates an instance of CCCL given by a hypergraph $H = (X, S)$, a set $\mathcal{C} = \{c_1, \ldots, c_{3n+1}\}$ of $3n + 1$ circles and a number b, which upper bounds the bends. To distinguish the vertices from G and H, we name the vertices of H with the letter x and an appropriate index. To simplify the presentation we think of every circle as a horizontal line in the projective plane. The angles $\alpha(\cdot)$ in $\mathcal{I}_c$ are then specifying the x-coordinates of the vertices.

Every vertex $v_i \in V$ of G contributes three consecutive circles c_{3i-2}, c_{3i-1}, c_{3i} in set $\mathcal{C}$ of the CCCL-instance. Set $\mathcal{C}$ contains one additional circle with index $z = 3n + 1$. For a better understanding, we give a rough idea what the use of the circles are. The circles c_{3i} will be used to indicate that an edge is covered from the vertex cover by the endpoint v_i (when we place a vertex here). We call the circle c_{3i} the *selector circle* of v_i. The circles c_{3i-1} have the purpose to give room for routing the hyperedges. The circles c_{3i-2} will be used to add constraints to the drawing of the hyperedges by placing vertices on them, which block certain paths. We use the short-hand notation $s(i) = 3i$ and $b(i) = 3i - 2$. The cycle c_z will be the place for a special vertex that is incident to many hyperedges.

Each edge $e_p = v_i v_j \in E$ of G contributes three vertices x_p^l, x_p and x_p^r in hypergraph H such that $I(x_p) = [s(i), s(j)]$, $I(x_p^l) = s(j)$, $I(x_p^r) = s(i)$, $\alpha(x_p^l) = 3p - 1$, $\alpha(x_p) = 3p$ and $\alpha(x_p^r) = 3p + 1$. Vertices x_p^l, x_p and x_p^r are joined with a hyperedge $\{x_p^l, x_p, x_p^r\}$, called *v-selector*, of multiplicity 5 in H. Note that for now, we allow some hyperedges with multiplicity, i.e. to appear multiple times. This makes the reduction easier. We discuss in the end, how to avoid hyperedges with multiplicity. It is not difficult to see that for any two edges e_p and e_q of G with $p < q$, the corresponding triple of vertices x_p^l, x_p and x_p^r due to edge e_p are all to the left of the triple of vertices x_q^l, x_q and x_q^r due to edge e_q. □

Observation A.1. A v-selector for $e_p = v_i v_j$ has at least two bends if x_p is placed on a circle other than $c_{s(i)}$ or $c_{s(j)}$. If it is placed either on $c_{s(i)}$ or on $c_{s(j)}$ then one bend suffices; see Figure A.2.

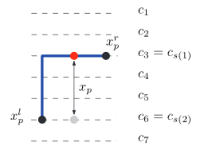

Figure A.2: *Handling of the edge $e_p = v_1 v_2$ in the reduction.*

Observation A.1 implies that, if x_p is placed on a circle other than the selector circles of the endpoints of e_p, then we get a ‘penalty’ of five additional bends (due to the multiplicity).

We continue the construction of hypergraph H by introducing $3n + 1 - k$ vertices, called *selection vertices*, which we denote by x_i^s for $1 \leq i \leq 3n + 1 - k$. Each of these vertices has α-value $3m + 3$, which guarantees that all selection vertices are placed to the right of all previously introduced vertices. For each $1 \leq i < 3n + 1 - k$, we set $I(x_i^s) = [1, 3n]$, while for $i = 3n + 1 - k$, we set $I(x_i^s) = 3n + 1$. In other words, each selection vertex can be placed on any circle of $\mathcal{C} \setminus \{c_z\}$, except one that is inevitably placed on c_z. The fact that all selection vertices have the same α-value further implies that no two selection vertices can be placed on the same circle; this in turn implies that there exist exactly k circles in $\mathcal{C} \setminus \{c_z\}$ that do not contain a selection vertex.

The construction of hypergraph H continues by introducing a special vertex x^z, such that $I(x^z) = 3n + 1$ and $\alpha(x^z) = 3m + 4$, that is, x^z lies on c_z and appears to the right of all previously introduced vertices. Further, for every edge $e_p \in E$ of G, hypergraph H contains a hyperedge $\{x_p, x^z\}$. We refer to these hyperedges as *e-connectors*.

We conclude the construction of hypergraph H by introducing a set of $n(m+1)$ *blocking* vertices, which restrict the possible routings of the e-connectors. In particular, every vertex $v_i \in V$ contributes a blocking vertex x_i^b in H, such that $I(x_i^b) = s(i)$ and $\alpha(x_i^b) = 3m + 2$, that is, x_i^b lies on $c_{s(i)}$ and is restricted between the selection vertices and the remaining vertices of H. Further, every pair consisting of a vertex $v_i \in V$ and an edge $e_p \in E$ of G also contributes a blocking vertex $x_{p,i}^b$ in H, such that $I(x_{p,i}^b) = b(i)$ and $\alpha(x_{p,i}^b) = \alpha(x_p)$, that is, $x_{p,i}^b$ has the same α value as x_p and lies on $c_b(i)$. We are now ready to make a crucial observation regarding the routing of the e-connectors in the presence of blocking vertices.

Observation A.2. Each e-connector $\{x_p, x^z\}$ with $e_p = v_i v_j$ has at least two bends if x_p lies on a selector circle c_{3k} due to the blocking vertices x_i^b, and x_j^b and the selection vertex on c_z. If there is no selection vertex on c_{3k-1} then two bends suffice.

An example of the construction of the CCCL-instance can be seen in Figure A.3. Since the size of $\mathcal{C}$ is $O(n)$ while the size of H is $O(nm)$, the construction of the CCCL-instance can be done in time polynomial with respect to the size of the Vertex Cover instance, which is $O(n+m)$. We now claim the following: vertex-cover instance G has a solution with k vertices, if and only if there exists a drawing of the constructed CCCL-instance H with $b = 7m$ bends.

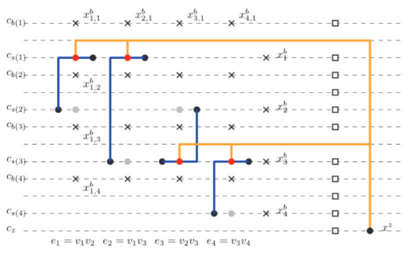

Figure A.3: *Generated CCCL instance for a graph on four vertices (a triangle $v_1v_2v_3$ with edge v_3v_4). Vertices for edges are shown as a red dot (•), alternative spots as a grey dot (•), blocking vertices as a cross (×) and selection vertices as a square (□). The drawing encodes the vertex cover $\{v_1, v_3\}$.*

We first prove the 'only if direction'. Assume that there exists a vertex cover U with $|U| = k$. For each edge $e_p = v_i v_j$ of G, we place vertex x_p of H on the circle $c_{s(i)}$ if $v_i \in U$, and on the circle $c_{s(j)}$ otherwise. Furthermore, we distribute the selection vertices on the circles of $\mathcal{C}$, such that there is no selection vertex on the circle c_{3i-1}, if vertex v_i belongs to U. This ensures that we can route all e-connectors with two bends by Observation A.2. By Observation A.1, a total of $5m + 2m = 7m$ bends suffices.

We are left with the 'if' direction. Assume that there exists a drawing with $7m$ bends. We now count the number of bends for each of the six hyperedges that contain x_p with $e_p = v_i v_j$. By Observation A.1, it follows that if x_p is not placed on $c_{s(i)}$ or $c_{s(j)}$ then we get at least 10 bends for the v-selectors of e_p. If x_p *is* placed on $c_{s(i)}$ or $c_{s(j)}$, then we can have five bends instead. Assume that x_p is placed on $c_{s(i)}$. If there is no selection vertex on the circle c_{3i-1}, then we need two bends for the e-connector of x_p. Otherwise we need at least three bends. This shows that seven bends for these hyperedges are possible only if there is no selection vertex either on c_{3i-1} or c_{3j-1}. Since we assumed that the drawing has at most $7m$ bends in total, we know that there is no selection vertex on either c_{3i-1} or c_{3j-1} for all edges $v_i v_j$. Since there are $3n + 1 - k$ such selection points, we have found a set of k vertices that forms a vertex cover of the graph G.

In the remainder of the proof, we discuss a few technical details that have been currently left out. Since we have used the projective plane in our construction, we have to rule out that a curve leaves to the right and enters from the left. This can be easily achieved by placing an additional blocking vertex b_i on every circle c_i such that $I(b_i) = i$ and $\alpha(b_i) = 3m + 5$, i.e. b_i is to the right of all vertices in the construction (under the projective plane assumption). We also have added the v-selectors with multiplicity. To avoid this, we have to adapt the reduction and implement a different construction to have an equivalent of Observation A.1. In particular, we need to guarantee that any routing of a v-selector other than ⌜/⌟-shapes, leads to a negative CCCL-instance. Whenever a v-selector is drawn as a zigzag path (which would allow to place x_p on a circle that is not a selector circle), an additional bend is needed. This could currently be saved by drawing the e-connector with only one bend joining x_p straight from the right. To prevent such routing, for each $1 \leq i \leq n$, we introduce a vertex x_i^m with $I(x_i^m) = 3i - 1$ and $\alpha(x_i^m) = -2$ (that is, x_i^m is placed on circle c_{3i-1} and at the left of all vertices in the construction) and a hyperedge $\{x_i^m, x_i^b\}$; see Figure A.4. We can route $\{x_i^m, x_i^b\}$ with one bend (and one bend is always needed), only if it contains a large piece of c_{3i-1}. It is now impossible to place any x_p on a circle c_{3i-1} unless more bends are added. Thus, we cannot 'repair' any zigzag routing of the v-selectors anymore. Consequently, if we require $m + 2m + n$ total bends, every v-selector needs one bend, every e-connector needs two bends and every of the n hyperedges that we added last needs one bend. This shows that the only possible positive instances have to behave as it was required for the case with multi-hyperedges.

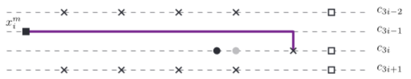

Figure A.4: *The additional hyperedges necessary to avoid multi-hyperedges in the reduction.*

Theorem A.3. *RDM is solvable in polynomial time.*

Proof. 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 may be placed on. We use d_e to capture the radial change of edge e. Specifically, the following linear program (LP) describes our problem:

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

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 (analogous to the Simplex algorithm [Dan90]). □

Appendix B: Experimental results

This appendix shows, on the next few pages, extended figures of those presented in the main text. That is, it shows all trial results for all measures.

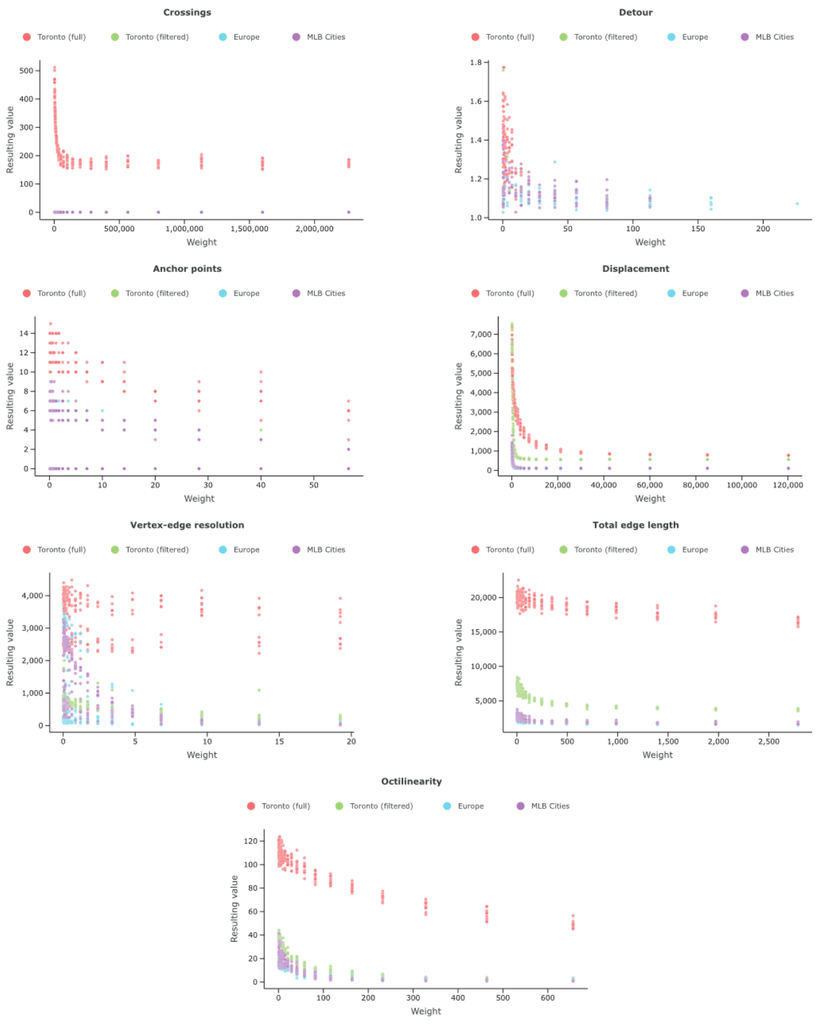

Figure B.1: *Varying the weight of a single quality measure in the objective function, showing the resulting value of the (unweighted) quality measure for which the weight is varied.*

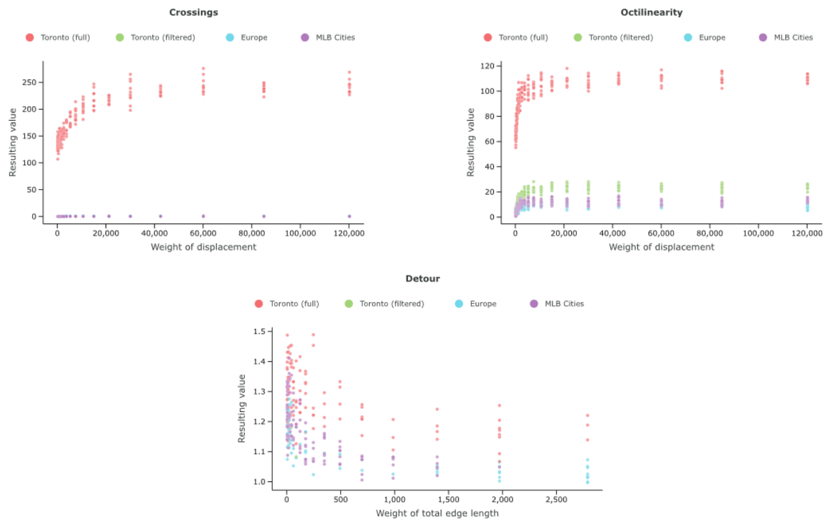

Figure B.2: *Relations between quality measures. (topleft) The effect on crossings when varying the weight of displacement. (topright) The effect on octilinearity when varying the weight of displacement. (bottom) The effect on detour when varying the weight of total edge length.*

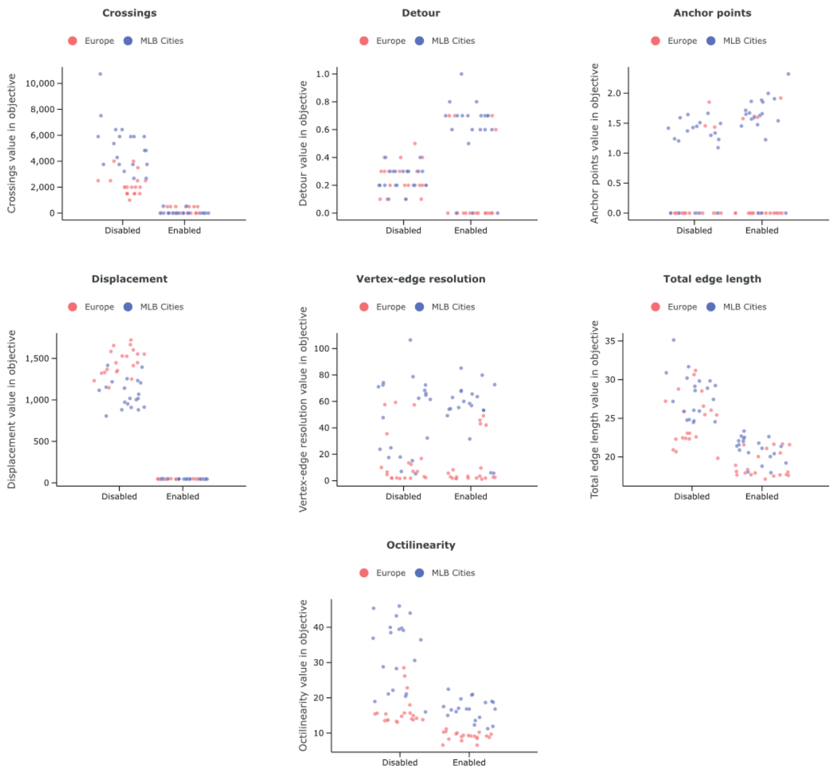

Figure B.3: *The effect of enabling and disabling optimizing a quality measure on the quality measure itself.*

References

[AAMH13] Alsallakh B., Aigner W., Miksch S., Hauser H.: Radial sets: Interactive visual analysis of large overlapping sets. *IEEE Transactions on Visualization and Computer Graphics 19*, 12 (2013), 2496–2505.

[ALT20] Akitaya H. A., Löffler M., Tóth C. D.: Multi-colored spanning graphs. *Theoretical Computer Science 833* (2020), 11–25.

[AMA*16] Alsallakh B., Micallef L., Aigner W., Hauser H., Miksch S., Rodgers P. J.: The state-of-the-art of set visualization. *Computer Graphics Forum 35*, 1 (2016), 234–260.

[ARRC11] Alper B., Riche N. H., Ramos G. A., Czerwinski M.: Design study of linesets, a novel set visualization technique. *IEEE Transactions on Visualization and Computer Graphics 17*, 12 (2011), 2259–2267.

[Bar16] Barth L.: Drawing Metro Maps on Concentric Circles. Master's thesis, Karlsruher Institut für Informatik, 2016.

[BCPS12] Brandes U., Cornelsen S., Pampel B., Sallaberry A.: Path-based supports for hypergraphs. *Journal of Discrete Algorithms 14* (2012), 248–261.

[BFM*19] Bekos M. A., Frank F., Meulemans W., Rodgers P. & Schulz A.: Concentric set schematization. In Proceedings of the Abstract of the Set Visual Analytics (SetVA) Workshop at IEEE VIS 2019 (2019), pp. 1–2.

[BvKM*11] Buchin K., van Kreveld M. J., Meijer H., Speckmann B., Verbeek K.: On planar supports for hypergraphs. *Journal of Graph Algorithms and Applications 15*, 4 (2011), 533–549.

[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.

[CR14] Chivers D., Rodgers P.: Octilinear force-directed layout with mental map preservation for schematic diagrams. In *Proceedings of the 8th International Conference on Diagrammatic Representation and Inference* (2014), T. Dwyer, H. C. Purchase and A. J. Delaney (Eds.), LNCS 8578, pp. 1–8. https://doi.org/10.1007/978-3-662-44043-8_1.

[CvGM*19] Castermans T., van Garderen M., Meulemans W., Nöllenburg M., Yuan X.: Short plane supports for spatial hypergraphs. *Journal of Graph Algorithms and Applications 23*, 3 (2019), 463–498.

[Dan90] Dantzig G. B.: Origins of the simplex method. In *A History of Scientific Computing*. S. G. Nash (Ed.). ACM, New York, NY, United States, 1990, pp. 141–151. https://doi.org/10.1145/87252.88081.

[DCM19] Delahaye D., Chaimatanan S., Mongeau M.: Simulated annealing: From basics to applications. In *Handbook of Metaheuristics* (3rd edition). M. Gendreau M. and J.-Y. Potvin (Eds.). Springer, Cham, Switzerland, 2019, pp. 1–35. https://doi.org/10.1007/978-3-319-91086-4.

[DETT98] Di Battista G., Eades P., Tamassia R., Tollis I. G.: *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice Hall, Upper Saddle River, NJ, United States, 1998.

[DH96] Davidson R., Harel D.: Drawing graphs nicely using simulated annealing. *ACM Transactions on Graphics 15*, 4 (1996), 301–331.

[DvKSW12] Dinkla K., van Kreveld M. J., Speckmann B., Westenberg M. A.: Kelp Diagrams: Point set membership visualization. *Computer Graphics Forum 31*, 3 (2012), 875–884.

[Ead84] Eades P.: A heuristic for graph drawing. *Congressus Numerantium 42* (1984), 149–160.

[EHKP15] Efrat A., Hu Y., Kobourov S. G., Pupyrev S.: MapSets: Visualizing embedded and clustered graphs. *Journal of Graph Algorithms and Applications 19*, 2 (2015), 571–593.

[EvKSS15] Eppstein D., van Kreveld M. J., Speckmann B., Staals F.: Improved grid map layout by point set matching. *International Journal of Computational Geometry and Applications 25*, 2 (2015), 101–122. https://doi.org/10.1142/S0218195915500077.

[EW94] Eades P., Wormald N. C.: Edge crossings in drawings of bipartite graphs. *Algorithmica 11*, 4 (1994), 379–403.

[FHN*12] Fink M., Haverkort H. J., Nöllenburg M., Roberts M. J., Schuhmann J., Wolff A.: Drawing metro maps using Bézier curves. In *Proceedings of the 20th International Symposium on Graph Drawing* (2012), W. Didimo and M. Patrignani (Eds.), LNCS 7704, pp. 463–474. https://doi.org/10.1007/978-3-642-36763-2_41.

[GCH*21] Geiger J., Cornelsen S., Haunert J., Kindermann P., Mchedlidze T., Nöllenburg M., Okamoto Y., Wolff A.: ClusterSets: Optimizing planar clusters in categorical point data. *Computer Graphics Forum 40*, 3 (2021), 471–481.

[GJ79] Garey M. R., Johnson D. S.: *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, San Francisco, CA, United States, 1979.

[HKvK*18] Hurtado F., Korman M., van Kreveld M. J., Löffler M., Sacristán V., Shioura A., Silveira R. I., Speckmann B., Tokuyama T.: Colored spanning graphs for set visualization. *Computational Geometry 68* (2018), 262–276.

[JM03] Jankun-Kelly T. J., Ma K.: MoireGraphs: Radial focus+context visualization and interaction for graphs with visual nodes. In *Proceedings of the IEEE Symposium on Information Visualization* (2003). https://doi.org/10.1109/INFVIS.2003.1249009.

[JP87] Johnson D. S., Pollak H. O.: Hypergraph planarity and the complexity of drawing Venn diagrams. *Journal of Graph Theory 11*, 3 (1987), 309–325.

[JWKN21] Jacobsen B., Wallinger M., Kobourov S. G., Nöllenburg M.: MetroSets: Visualizing sets as metro maps. *IEEE Transactions on Visualization and Computer Graphics 27*, 2 (2021), 1257–1267.

[KW01] Kaufmann M., Wagner D. (Eds.):. *Drawing Graphs: Methods and Models*, *LNCS 2025*. Springer, Berlin Heidelberg, Germany, 2001. https://doi.org/10.1007/3-540-44969-8.

[LGS*14] Lex A., Gehlenborg N., Strobelt H., Vuillemot R., Pfister H.: UpSet: Visualization of intersecting sets. *IEEE Transactions on Visualization and Computer Graphics 20*, 12 (2014), 1983–1992.

[MDS*17] Meulemans W., Dykes J., Slingsby A., Turkay C., Wood J.: Small multiples with gaps. *IEEE Transactions on Visualization and Computer Graphics 23*, 1 (2017), 381–390.

[Meu14] Meulemans W.: Similarity Measures and Algorithms for Cartographic Schematization. PhD thesis, Technische Universiteit Eindhoven, 2014.

[MR14] Micallef L., Rodgers P.: EulerForce: Force-directed layout for Euler diagrams. *Journal of Visual Languages and Computing 25*, 6 (2014), 924–934.

[MRS*13] Meulemans W., Riche N. H., Speckmann B., Alper B., Dwyer T.: KelpFusion: A hybrid set visualization technique. *IEEE Transactions on Visualization and Computer Graphics 19*, 11 (2013), 1846–1858.

[MSS21] Meulemans W., Sondag M., Speckmann B.: A simple pipeline for coherent grid maps. *IEEE Transactions on Visualization and Computer Graphics 27*, 2 (2021), 1236–1246.

[MWTI19] Mizuno K., Wu H., Takahashi S., Igarashi T.: Optimizing stepwise animation in dynamic set diagrams. *Computer Graphics Forum 38*, 3 (2019), 13–24.

[NR20] Niedermann B., Rutter I.: An integer-linear program for bend-minimization in ortho-radial drawings. In *Proceedings of the 28th International Symposium on Graph Drawing and Network Visualization* (2020), LNCS 12590, pp. 235–249. https://doi.org/10.1007/978-3-030-68766-3_19.

[NRW19] Niedermann B., Rutter I., Wolf M.: Efficient algorithms for ortho-radial graph drawing. In *Proceedings of the 35th International Symposium on Computational Geometry* (2019), LIPIcs 129, pp. 53:1–53:14. https://doi.org/10.4230/LIPIcs.SoCG.2019.53.

[NW11] Nöllenburg M., Wolff A.: Drawing and labeling high-quality metro maps by mixed-integer programming. *IEEE Transactions on Visualization and Computer Graphics 17*, 5 (2011), 626–641.

[RD10] Riche N. H., Dwyer T.: Untangling euler diagrams. *IEEE Transactions on Visualization and Computer Graphics 16*, 6 (2010), 1090–1099.

[Rei15] Reimer A.: Cartographic Modelling for Automated Map Generation. PhD thesis, Technische Universiteit Eindhoven, 2015.

[Rob12] Roberts M. J.: *Underground Maps Unravelled—Explorations in Information Design*. Self-published, 2012.

[RRRW18] Radermacher M., Reichard K., Rutter I., Wagner D.: A geometric heuristic for rectilinear crossing minimization. In *Proceedings of the Meeting on Algorithm Engineering and Experiments* (2018), R. Pagh and S. Venkatasubramanian (Eds.), pp. 129–138. https://doi.org/10.1137/1.9781611975055.12.

[Sch21] Schiewe J.: Distortion effects in equal area unit maps. *KN-Journal of Cartography and Geographic Information 71* (2021), 71–82.

[Shn96] Shneiderman B.: The eyes have it: A task by data type taxonomy for information visualizations. In *Proceedings of the IEEE Symposium on Visual Languages* (1996), pp. 336–343. https://doi.org/10.1109/VL.1996.545307.

[Sli18] Slingsby A.: Tilemaps for summarising multivariate geographical variation. In *Proceedings of the Workshop on Visual Summarization and Report Generation* (2018).

[SMDS14] Sadana R., Major T., Dove A. D. M., Stasko J. T.: Onset: A visualization technique for large-scale binary set data. *IEEE Transactions on Visualization and Computer Graphics 20*, 12 (2014), 1993–2002.

[SRMW11] Stott J. M., Rodgers P., Martinez-Ovando J. C., Walker S. G.: Automatic metro map layout using multicriteria optimization. *IEEE Transactions on Visualization and Computer Graphics 17*, 1 (2011), 101–114.

[SV10] Speckmann B., Verbeek K.: Necklace maps. *IEEE Transactions on Visualization and Computer Graphics 16*, 6 (2010), 881–889.

[Tuf01] Tufte E. R.: *The Visual Display of Quantitative Information* (2nd edition). Graphics Press, Cheshire, CT, USA, 2001.

[Tve81] Tversky B.: Distortions in memory for maps. *Cognitive Psychology 13*, 3 (1981), 407–433.

[Tve93] Tversky B.: Cognitive maps, cognitive collages, and spatial mental models. In *Proceedings of the European Conference on Spatial Information Theory* (1993), A. U. Frank and I. Campari I. (Eds.), LNCS 716, pp. 14–24. https://doi.org/10.1007/3-540-57207-4_2.

[vDH14] van Dijk T. C., Haunert J.: Interactive focus maps using least-squares optimization. *International Journal of Geographical Information Science 28*, 10 (2014), 2052–2075.

[vDvGH*13] van Dijk T. C., van Goethem A., Haunert J., Meulemans W., Speckmann B.: Accentuating focus maps via partial schematization. In *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems* (2013), C. A. Knoblock, M. Schneider, P. Kröger, J. Krumm and P. Widmayer P. (Eds.), pp. 418–421. https://doi.org/10.1145/2525314.2525452.

[vDvGH*14] van Dijk T. C., van Goethem A., Haunert J., Meulemans W., Speckmann B.: Map schematization with circular arcs. In *Proceedings of the 8th International Conference on Geographic Information Science* (2014), LNCS 8728, pp. 1–17. https://doi.org/10.1007/978-3-319-11593-1_1.

[vG16] van Goethem A.: Algorithms for Curved Schematization. PhD thesis, Technische Universiteit Eindhoven, 2016.

[vGKvK*17] van Goethem A., Kostitsyna I., van Kreveld M. J., Meulemans W., Sondag M., Wulms J.: The painter's problem: Covering a grid with colored connected polygons. In *Proceedings of the 25th International Symposium on Graph Drawing and Network Visualization* (2017), LNCS 10692, pp. 492–505. https://doi.org/10.1007/978-3-319-73915-1_38.

[WNT*20] Wu H., Niedermann B., Takahashi S., Roberts M. J., Nöllenburg M.: A survey on transit map layout—from design, machine, and human perspectives. *Computer Graphics Forum 39*, 3 (2020), 619–646.

[WNV20] Wu H.-Y., Nollenburg M., Viola I.: Multi-level area balancing of clustered graphs. *IEEE Transactions on Visualization and Computer Graphics* (2020). https://doi.org/10.1109/TVCG.2020.3038154.