Metro Maps on Flexible Base Grids

Hannah Bast
University of Freiburg
Freiburg, Germany
bast@cs.uni-freiburg.de

Patrick Brosi
University of Freiburg
Freiburg, Germany
brosi@cs.uni-freiburg.de

Sabine Storandt
University of Konstanz
Konstanz, Germany
sabine.storandt@uni-konstanz.de

ABSTRACT

We present new generic methods to efficiently draw schematized metro maps for a wide variety of layouts, including octilinear, hexalinear, and orthoradial maps. The maps are drawn by mapping the input graph to a suitable grid graph. Previous work was restricted to regular octilinear grids. In this work, we investigate a variety of grids, including triangular grids and orthoradial grids. In particular, we also construct sparse grids where the local node density adapts to the input graph (e.g. octilinear Hanan grids, which we introduce in this work). For octilinear maps, this reduces the grid size by a factor of up to 5 compared to previous work, while still achieving close-to-optimal layouts. For many maps, this reduction also leads to up to 5 times faster solution times of the underlying optimization problem. We evaluate our approach on five maps. All octilinear maps can be computed in under 0.5 seconds, all hexalinear and orthoradial maps can be computed in under 2.5 seconds.

CCS CONCEPTS

• **Human-centered computing** → **Graph drawings**; • **Mathematics of computing** → **Approximation algorithms**; **Integer programming**.

KEYWORDS

Metro Maps, Graph Drawing, Map Schematization

ACM Reference Format:
Hannah Bast, Patrick Brosi, and Sabine Storandt. 2021. Metro Maps on Flexible Base Grids. In *17th International Symposium on Spatial and Temporal Databases (SSTD '21), August 23–25, 2021, virtual, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3469830.3470899

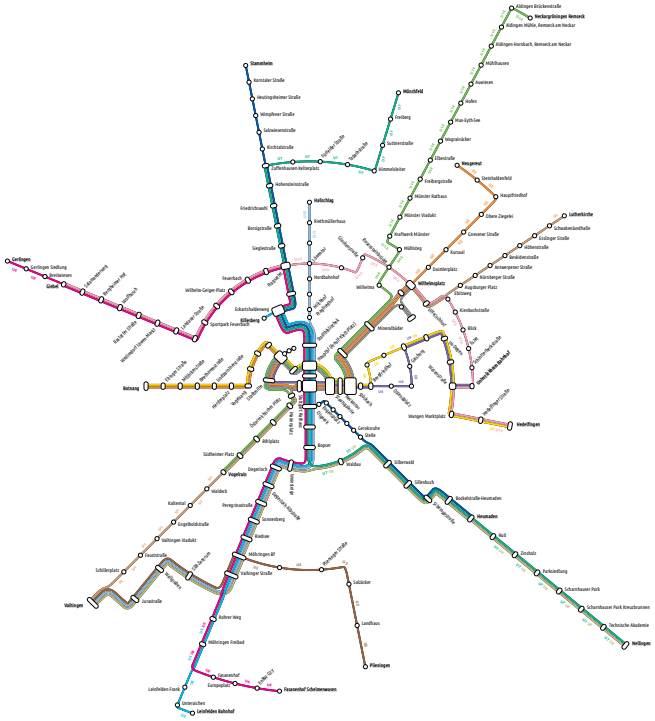

Figure 1: The Stuttgart light rail network rendered in an orthoradial fashion by our approach in 1.9 seconds.

1 INTRODUCTION

Drawing schematized metro maps is a well-studied topic [37]. The prevalent layout is octilinear, where segment orientations are multiples of 45 degrees. Previous work computed such maps via an optimal topological embedding (called *image* in this work) of the input graph G in a regular octilinear grid graph Γ (called the base grid graph or just base grid) that covers the padded bounding box of G [8]. Two approaches to compute that image were presented: an integer linear program (ILP) and an approximation algorithm based on the computation of shortest paths in the grid graph. This work had the following limitations:

(1) *Grid Graph Size.* For sparse areas of the input graph, the octilinear grid graph Γ was unnecessarily dense, which negatively impacted solution times, in particular for the ILP approach.

(2) *No Layout Flexibility.* The approach was restricted to octilinear metro maps, although other layout types have recently found some research interest.

(3) *Various Imperfections.* The approximation algorithm did not always find a feasible solution. The proposed methods only worked for input graphs with a node degree of up to 8.

1.1 Contributions

We fix all of the problems mentioned above:

(1) We investigate a variety of grid graphs suitable for octilinear maps: grids cropped to the convex hull of the input graph, quadtree-based octilinear grids, and a new kind of grid graph which we call *octilinear Hanan grid*; see Figure 2 for examples. The latter is inspired by the well-known fact that an ordinary Hanan grid, constructed by drawing horizontal and vertical lines through an

SSTD '21, August 23–25, 2021, virtual, USA

ACM ISBN 978-1-4503-8425-4/21/08…$15.00
https://doi.org/10.1145/3469830.3470899

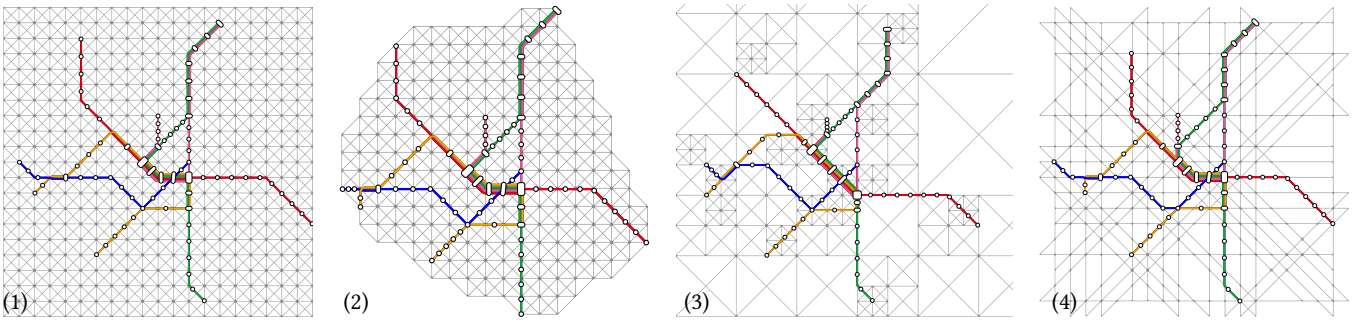

Figure 2: The tram network of Freiburg as an octilinear drawing, rendered using an integer linear program (ILP) on four base grid graphs (depicted in gray): (1) a full octilinear grid graph covering the padded bounding box of the input graph, (2) an octilinear grid graph covering the convex hull of the input graph, (3) an octilinear grid graph built from a quadtree populated with the input nodes, (4) a grid graph built from the octilinear Hanan grid of the input nodes, cropped to its padded bounding box. For (2) and (4), the impact of the sparse grid graph on the solution quality was minimal (see Table 2 for details).

input point set P, contains a rectilinear minimum Steiner Tree for P [18]. Our grids are *sparse* in the sense that the local node densities correspond to those of the input graph. We show that, on average, our best-performing sparse grids achieve solutions within 2% of the optimum.

(2) To compute hexalinear and orthoradial maps, we use triangular and orthoradial grid graphs, respectively. The orthoradial grid consists of concentric rings and rays, where the number of nodes per ring depends on the ring radius; see Figure 12. We are able to compute all our maps in under 2.5 seconds. This is the first method to produce orthoradial metro maps that respect *all* the constraints originally established for octilinear embeddings in [29].[1]

(3) We show how constraint relaxation can be used to ensure that a feasible solution is always found. We describe a node-splitting technique that enables our approach to handle arbitrary node degrees in the input graph.

1.2 Related Work

Our work is strongly related to previous work on map schematization. Prior to metro map drawing, the focus was on schematizing road networks. In [13, 14], general steps for this task were described. In [24], the schematization of a polyline into segments following a predefined set of orientations was studied. Local-search techniques, sometimes combined with probabilistic techniques (e.g. simulated annealing) were described in [4], [36], and [1]. Mathematical properties of topology-preserving octilinear road or railroad maps were studied in [11], and an $O(n \log^3 n)$ algorithm was presented (which guaranteed feasible, but not necessarily optimal solutions). Schematization based on the concept of *strokes* (a decomposition of the input graph into simple paths) was investigated in [11] and [33].

The problem of finding metro map embeddings was formally introduced by Hong et al. [19, 20] and a spring layout algorithm was described. An often-used preprocessing step used in later works is to contract input nodes of degree 2 prior to schematization. The contracted nodes are then later re-inserted equidistantly into the final drawing. We call this the degree-2 heuristic. In [22], a polyline simplification method for metro map generation was proposed. To consider networks, an input edge ordering was determined, and the corresponding polylines then simplified in this order. Stott and Rodgers used a local search to find optimal metro map embeddings by iteratively moving vertices on a grid [31]. This was later refined by moving entire clusters of nodes [32]. Nöllenburg and Wolff [29] defined a set of hard and soft constraints for visually pleasing octilinear metro map embeddings.[1] They presented a mixed-integer linear program (MIP) to find optimal maps adhering to these constraints. Several works then built on this MIP: In [30], it was extended to consider labeling. In [23], it was altered to preserve shortest paths in the input line graph (to avoid the distortion of travel distances). Wu et al. [39] added the ability to select a single focus line, which is then rendered horizontally. In [38], the MIP was extended to keep enough space for large pictorial station labels. The spring layout algorithm first described in [19] was reconsidered in [12], with greatly improved results. In [34], a fast octilinearization approach using least-square optimization was described, but did not guarantee octilinearity. Our approach was first described in [8] and will be described in greater detail in Section 1.3. The main difference to previous work is that we search for octilinear *drawings* (allowing an arbitrary number of bends per segment), not octilinear *embeddings* (allowing only straight segments).

There has also been some interest in drawing schematic maps with non-octilinear layouts. A force-based approach to move the control points of cubic Bèzier curves to arrive at curvilinear layouts was described in [15]. In [35], a stroke-based approach to draw maps consisting of circular arcs was presented. In [25], the MIP from [30] was modified to render k-linear layouts (where edge orientations are multiples of $360/2k$). Recently, orthoradial metro maps have also received research attention [37]. In [5] it was noticed that orthoradial maps may be interpreted as orthogonal maps on a cylinder and that a bend-free planar orthoradial drawing has a combinatorial representation based on the angles of adjacent vertices. In [27], polynomial-time algorithms to obtain and to decide whether such a representation is valid were described. Finding a

[1] In a nutshell: no crossings, preserve the (circular) edge order at nodes, adhere to the given layout, nodes at least a certain distance apart, as small as possible: number of bends, node displacement, segment length.

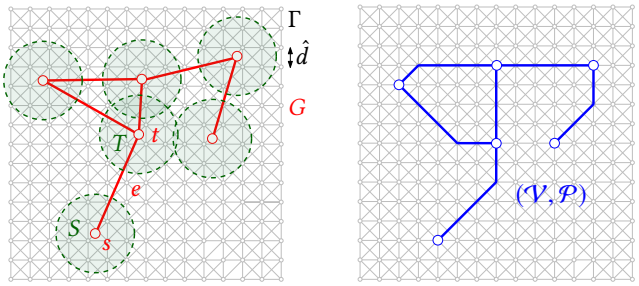

Figure 3: Finding an octilinear drawing for an input graph G using the approach from [8]: an octilinear grid graph Γ is build to cover the (padded) bounding box of G. Each input node is assigned a set of candidate grid nodes. Line bends in Γ are penalized by special turn edges, and grid node displacement are penalized by special sink edges (not depicted). We then search for the optimal set of non-intersecting image paths, resulting in the octilinear metro map image $(\mathcal{V},\mathcal{P})$.

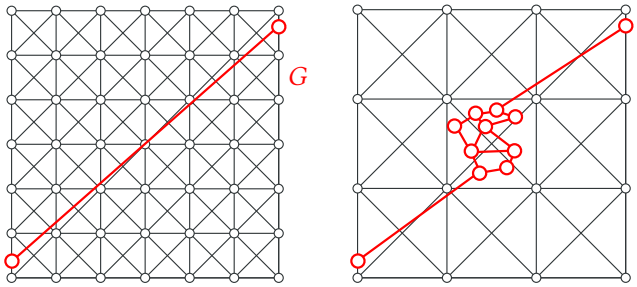

Figure 4: Left: Grid redundancy problem. Although the optimal metro map image of the input graph G is obvious, we still require a full grid graph with 49 nodes and 156 edges. Right: Node density problem. The grid graph is not dense enough to render the input graph without extreme displacement.

representation with minimized edge bends was left as an open question. In [26], an ILP was given to render orthoradial drawings with a minimum number of edge bends (other aspects were not optimized, and a minimum distance between nodes was not guaranteed). In this work, we will show that our own method is able to both render orthoradial and hexalinear maps.

A strongly related problem is that of finding optimal line orderings on segments of the (not necessarily schematized) network. There are several slightly different concepts of what constitutes optimality. In [10], lines were allowed to cross on edges, but not on nodes, and the number of crossings was minimized. This was later called the Metro Line Crossing Minimization (MLCM) problem [9], and an ILP for this (NP-hard) problem was given in [3]. Polynomial algorithms for restricted cases were developed in [2, 9, 16, 28]. An alternative formulation (the Metro Line Node Crossing Minimization problem, MLNCM), where lines cross at nodes, but not on edges, was given in [6]. The concept of additionally minimizing *line separations* was also introduced and an ILP for a weighted version of MLNCM both with and without line separations was described. In [7], this was extended by several simplification rules to speed up the optimization. Finally, [17] considered a variant in which the number of nodes where crossings occur was minimized. We rely on previous work in [6, 7] to optimize line orderings.

For an extensive recent overview over the current state of research regarding metro map drawing we refer to [37].

1.3 Definitions and Overview

We use the following basic notation: As input, we get an undirected labeled graph $G = (V, E, L)$. V may be understood as stations but may also contain non-station intersections. The edges E are connections between nodes, and each edge e is labeled with lines $L(e)$ traveling through them. We then search for a metro map drawing $\mathcal{D}(G)$.

The approach originally described in [8] renders octilinear metro maps by searching for the optimal topological embedding of an input graph G in a grid graph Γ covering the padded bounding box of G. A topological embedding (often called a subgraph homeomorphism) is a pair of injective mappings $(\mathcal{V},\mathcal{P})$, where $\mathcal{V}$ maps vertices of G to image vertices of Γ, and $\mathcal{P}$ maps edges of G to image paths in Γ [21]. In this work, we call $(\mathcal{V},\mathcal{P})$ the image of G in Γ. The image path $\mathcal{P}(\{u,v\})$ of an input edge $e = \{u,v\}$ must start at $\mathcal{V}(u)$ and end at $\mathcal{V}(v)$. To preserve the input topology, we require image paths to be vertex-disjoint. Additionally, the circular orderings of edges at input nodes must be preserved. A drawing $\mathcal{D}(G)$ can be trivially extracted from $(\mathcal{V},\mathcal{P})$ by converting the image paths $\mathcal{P}$ into polyline segments. The assignment of input nodes to grid nodes is unrestricted, but the geographical distance to the original position is penalized. While the grid cell size may be chosen freely, it was observed in [8] that choosing the average distance between adjacent input nodes works well in practice. Each grid node is extended by 8 port nodes, connected to the original grid node by weighted sink edges which model displacement costs (see Figure 12 for an example). To be able to penalize line bends in the final drawing, the port nodes are pairwise connected by bend edges, making it possible to add penalty costs to bends. The modeling of these line bend costs is similar to the modeling of turn costs in transportation networks, but does not require a directed graph. Each grid edge may additionally be assigned an offset weight, for example to prefer diagonal, horizontal or vertical edges, to prefer edges close to the original line course, or to avoid obstacles like rivers, lakes, or coastlines.

1.3.1 Edge Routing. The optimal metro map image is then constructed by routing all input edges through Γ. Figure 3 gives an example. An image path $\mathcal{P}(e)$ for an edge $e = \{s,t\}$ is then the shortest path through Γ connecting two sets S and T of grid node candidates which does not cross any other path, and thus optimizes the path length, the node displacements and the number and acuteness of bends simultaneously. The start and end nodes of $\mathcal{P}(e)$ are the image nodes for s and t. The sets of grid node candidates S and T may be restricted to a fixed radius around the input node positions to speed up the computation.

1.3.2 Finding Optimal Metro Map Images. In [8], an integer linear program (ILP) was described to find optimal images. As this

ILP turned out to be too slow (solution times ranged from several minutes to 19 hours), an approximate approach was developed in which image paths were routed iteratively through the grid graph using Dijkstra's algorithm. The routing followed a predetermined ordering of the input edges which was based on a heuristic value of their importance. If no feasible solution could be found using this ordering, random orderings were tested until a feasible solution was found. To preserve the input topology, grid edges already used by an image path, as well as crossing diagonal grid edges, were outfitted with infinite weights for later iterations. Edges which would violate the edge orderings at nodes in the input graph were also outfitted with infinite weights for later iterations. As a final polishing step, a local search (steepest ascent hill-climbing) was added. Station labels were added after an optimal octilinear layout was found. The quality of the approximate approach was found to be comparable to the optimal solutions, with drastically improved solution times (for medium-sized networks, a map was typically found in under one second). If the degree-2 heuristic was used, the resulting segments were sometimes too short to hold all contracted nodes with a minimum distance. For the approximate approach, it was therefore experimented with an additional spring-force based *density penalty* added to segments which were too short.

1.3.3 Grid Density And Redundancy Problems. A problem with the original approach is that the octilinear base grid graph may not be dense enough to render certain parts of the input graph without extreme displacement. The grid density may also be highly redundant. Figure 4 gives two examples. If the input graph G is very simple (e.g. just a single diagonal edge), building a full grid for the padded bounding box of G is unnecessary, as the optimal octilinear representation of G is obvious (Figure 4, left). On the other hand, if node densities in G are heterogeneously distributed, we might chose a grid cell size too small (Figure 4, right). We are therefore interested in techniques that produce adaptive base grids, with grid densities correlating to the node density of the input graph. Octilinearity should still be guaranteed.

1.3.4 Alternative Layouts. The original approach only considered octilinear maps. In principle, it works on arbitrary base graph, and could therefore be used to generate schematic transit maps following other base grid types. In this work, we experiment with orthoradial and hexalinear base grids.

1.3.5 Outline. The remainder of this work is organized as follows: In Section 2, we describe how to handle cases where the input graph has a higher maximum node degree than the base grid graph. Section 3 explains how we apply constraint relaxation to prevent stalling of the approximate approach on sparse base grid graphs, or base grid graphs with a low maximum node degree. Section 4.1 describes how to reduce the size of the base grid graph by cropping not to the padded bounding box of G, but the padded convex hull. We will then discuss grid graphs based on quadtrees and octilinear Hanan grids in Sections 4.2 and 4.3. In Sections 5.1 and 5.2, we will describe two alternative base grid types. All base grids will finally be experimentally evaluated in Section 6.

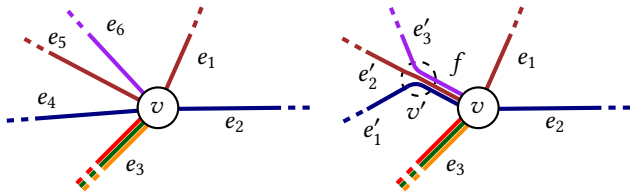

Figure 5: Left: Input node v has a degree of 6, prohibiting an orthoradial layout. Right: We keep the first (in clockwise order) 3 adjacent edges of v, combine the lines of the remaining edges e_4, e_5 and e_6 into a single new edge f and connect it to a new non-station node v'.

2 NODE SPLITTING

Naturally, octilinear metro maps can only be drawn for input graphs with a maximum node degree less than 9. Real-world transportation networks typically satisfy this requirement. However, hexalinear maps require a maximum node degree of 6, and orthoradial metro maps only allow a maximum node degree of 4. As larger node degrees are common in real-world transit networks, this limitation has to be addressed prior to rendering such maps.

We use the approach depicted in Figure 5. Given a maximum node degree D, an input node v with $\deg(v) > D$ and incident edges $e_1 \dots e_{\deg(v)}$, we detach edges $e_{D+1} \dots e_{\deg(v)}$ and add them to a new non-station node v', placed at the same position as v (note that Figure 5 depicts v' with a different position than v' for better readability). An additional edge f is added between v' and v with $L(f) = \bigcup_{e_{D+1} \dots e_{\deg(v)}} L(e)$. If we still have $\deg(v') > D$, we repeat the process for v'.

3 CONSTRAINT RELAXATION

Even with prior node splitting, the greedy approximate approach from [8] might not find a feasible solution, because the algorithm might stall in a situation where an edge cannot be routed crossing-free. To overcome this problem, we relax the constraints that image paths must be crossing free and that edge orderings must be preserved. Instead of setting the weight of edges that would violate these constraints to infinity, we introduce a weight w_∞ which is set high enough so that *any* path through G using only compliant edges is cheaper than w_∞. This enables the subsequent local search to overcome these topology violations. We also note that even if the local search does not fix the violation, a formal topology violation will not necessarily result in a final map drawing with perceived incorrect topology. Figure 6 provides an example.

4 SPARSE GRID GRAPHS

In this section, we describe three techniques to reduce the number of nodes and edges in the octilinear base grid graph: convex-hull grids (Section 4.1), octilinear quadtree grids (Section 4.2), and octilinear Hanan grids (Section 4.3). The first technique is relatively simple, the latter two are adaptive and reflect input node densities.

4.1 Convex-Hull Grids

A simple technique to reduce the size of the base grid graph is to crop the full grid to the padded convex hull of the input graph.

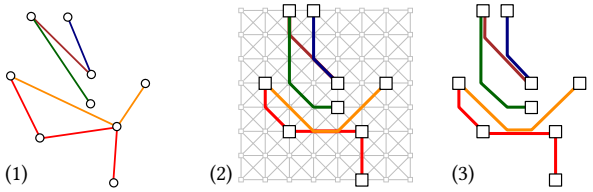

Figure 6: With constraint relaxation, the line graph (1) results in a metro map image (2) which does not have vertex-disjoint paths, violating topology constraints. By inserting non-station nodes where paths cross, the final map (3) allows for a line ordering on the overlapping segments which preserves the input topology.

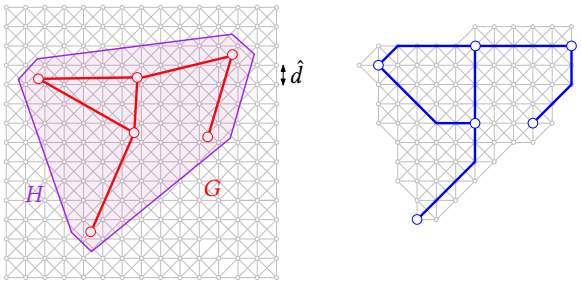

Figure 7: A simple approach for reducing the base grid graph size: instead of building the base grid graph from the padded bounding box of G, we use the padded convex hull H.

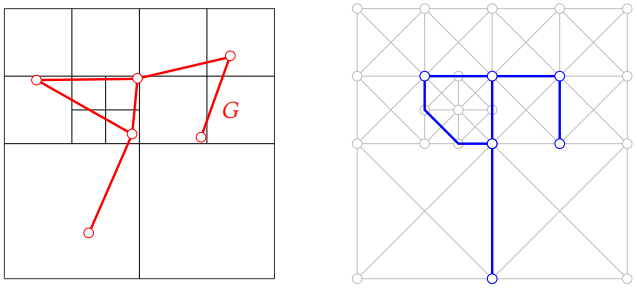

Figure 8: Left: an input graph G and the corresponding quadtree. Each quadtree cell contains exactly one input node. Right: an octilinear grid graph built from the quadtree, with a metro map image of G.

Let S be the set of coordinates of the input line nodes V. We extend each $p = (x, y) \in S$ by four additional coordinates: $p_1 = (x + r, y)$, $p_2 = (x - r, y)$, $p_3 = (x, y + r)$ and $p_4 = (x, y - r)$, where r is the padding. We then calculate the convex hull of the resulting coordinate set S' and only keep grid nodes within that hull. Figure 7 gives an example.

4.2 Octilinear Quadtree Grids

A quadtree is a grid which can be constructed for a set S of input points in the following way: start with the (padded) bounding box of S as a grid with a single cell. For a given threshold value t, while there is a cell that has more than t points from S in it, split this cell into 4 rectangular cells. The resulting grid has a tree-like structure, where each grid cell of size l may have 4 direct child vertices of size $l/2$. To again maintain a minimum segment length in the final drawing, we additionally add a minimum grid cell size $d \leq l$: if a cell contains more than 1 point, but a split would result in a cell size $l < d$, we do not split.

The resulting grid (see Figure 8, left for an example) is rectilinear, but can easily be transformed into a sparse octilinear grid graph by adding two diagonal edges to leaf cells (Figure 8, right).

We construct the quadtree in the following way: the lower left corner of the quadtree bounding box is aligned to the lower left corner of the full grid bounding box. We then search for the smallest i such that a bounding box B with side length $2^i \cdot d$ contains the entire full grid, and build the quadtree from this box. Additionally, we require the quadtree cell size to be at least d. Then any cell size is a multiple of d. This maintains comparability to the original approach: the nodes in the quadtree grid are a subset of a full octilinear grid with grid cell size d covering B.

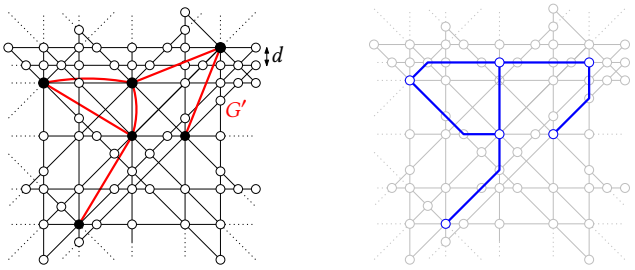

Figure 9: Left: a bounding-box restricted octilinear Hanan grid for an input graph G. The black nodes were part of the input graph G and have been snapped to a base grid with cell size d. Edges of G are depicted in red. Right: a metro map image of G on $O(G)$.

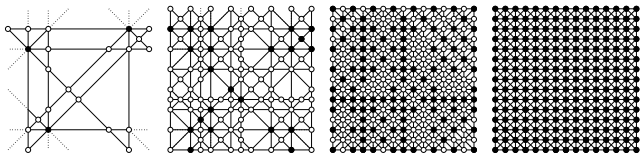

Figure 10: Hanan iterations for an octilinear Hanan grid on 3 input points S. After 4 iterations, the octilinear Hanan grid is the full octilinear grid.

4.3 Octilinear Hanan Grids

Given a set of two-dimensional points S, the Hanan grid $H(S)$ is a graph constructed as follows: (1) Draw vertical and horizontal lines through each $p \in S$. (2) Where these lines intersect, add vertices. A Hanan grid $H(S)$ contains a rectilinear minimum Steiner tree for S [18]. As a rectilinear Steiner tree may be understood as a primitive metro map, it seems worthwhile to evaluate their applicability to grid-based metro map drawing.

Since rectilinear metro maps are of only little interest, we slightly extend the definition of a classic Hanan grid and introduce the concept of octilinear Hanan grids. Again given a point set S on the plane, the construction of an octilinear Hanan grid $O(S)$ is similar to

a classic Hanan grid, but we also draw diagonal lines and consider them for possible intersection points.

4.3.1 Base Grid Snapping. As we would like to maintain a minimum segment length in the final maps, we do not construct the octilinear Hanan grid directly on the input graph G, but first snap the input node positions to a regular grid with cell size d. We denote this graph by G'. The base grid is restricted by the (padded) bounding box of the input graph. We additionally restrict the extent of $O(S)$ to the boundaries of the original grid. See Figure 13 for an example.

4.3.2 Hanan Iterations. An octilinear Hanan grid $O(S)$ with nodes V can be made increasingly more fine-grained if we take the nodes V as points S' and construct an octilinear Hanan grid $O(S')$ from them. We call such a step a Hanan iteration. Figure 10 gives an example.

4.4 Path Cost Preserving Edge Weights

For both the quadtree and the octilinear Hanan grid graph we would like the costs of the image paths to be comparable to a full octilinear grid graph. Consider such a full octilinear grid graph $\Gamma = (\Psi, \Omega)$ covering the entire bounding box of a sparse grid graph S, built either using the quadtree approach, or the octilinear Hanan grid approach. Because of the way we constructed S, the following properties hold: (1) All paths in S will only consist of octilinear segments. (2) The nodes of S are a subset of Ψ. (3) Each path p through S has a corresponding path in Γ.

We additionally would like a path in S to have the same cost as the corresponding path in Γ. Observe that an edge e in S might "simulate" n full grid edges $\omega \in \Omega$. This might also be understood as a virtual contraction of $n-1$ grid nodes $\psi \in \Psi$. If $n = 1$, we can use the original edge weight $w(\omega)$ directly. If $n > 1$, e contains $n-1$ contracted nodes which are passed with a 180° bend. We therefore use $w(e) = (n-1) \cdot c_{180} + \sum_{i=1}^{n} w(\omega_i)$ as an edge weight, where c_{180} is the cost of 180 degree bend edges.

5 GRIDS FOR ALTERNATIVE LAYOUTS

To render both hexalinear and orthoradial maps, we describe two kinds of base grid graphs in this section: triangular grids (Section 5.1) and pseudo orthoradial grids (Section 5.2).

5.1 Triangular Grids

A triangular grid partitions the plane into equilateral triangles. At the intersection points, exactly 6 triangle sides meet at 60° angles. If the intersection points are interpreted as nodes, and the triangle sides as edges, any path on such a triangular grid graph is hexalinear. See Figure 11 for an example of a triangular grid graph and hexalinear paths through it.

In [8], we extended each grid node of an octilinear grid graph by 8 port nodes and connected them via sink edges to the original grid node, and via bend edges to each other (Figure 12.1). Bend edge costs reflected the cost of a 45°, 90° or 135° bend at the grid node. We designed them in such a way that shortcuts always had higher or equal costs - for example, in Figure 12.1, the blue 45° bend edge may be circumvented by first taking a 180° edge, and then a 135° edge. Such shortcuts are still possible in triangular grid graphs, see Figure 12.2 for an example. To avoid them, we use the same technique as in [8]. For the grid edges, we used unit weights.

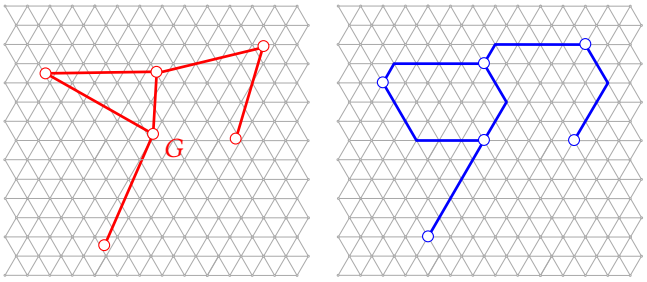

Figure 11: Left: A triangular grid graph covering the padded bounding box of an input graph G. Right: A hexalinear image of G on the triangular grid graph.

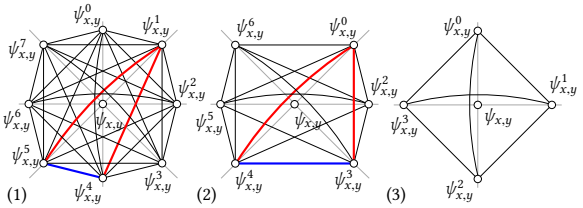

Figure 12: Extended grid node $\psi_{x,y}$ in an octilinear grid graph (1), a triangular grid graph (2) and an orthoradial grid graph (3). Paths may perform bends at $\psi_{x,y}$ by using bend edges, weighted by the respective bend penalty. In (1) and (2), shortcuts may undermine the bend penalties.

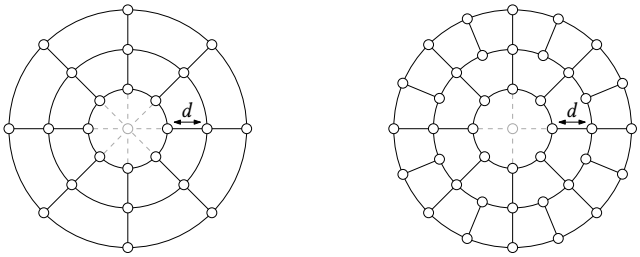

Figure 13: Two kinds of orthoradial grid graphs, both with (optional) center nodes (depicted in gray). d is the distance between rings. Left: Orthoradial grid graph with $b = 8$. Right: Pseudo orthoradial grid graph where b is doubled each time the radius doubles.

5.2 Orthoradial Grids

As mentioned above, recent work discussed the problem of drawing metro maps in an orthoradial fashion [26, 27, 37] and it was observed that an orthoradial may be interpreted as orthogonal maps on a cylinder [27]. Figure 13, left depicts an orthoradial grid produced in this fashion from an ortholinear grid. For the distance between rings (and for the radius of the inner ring) we use the cell grid size parameter d. As the grid density decreases with greater radius, an obvious esthetic problem are the large white-space areas.

Table 1: Effects of sparse base grids on base graph size, given as number of nodes $|\Psi|$ and number of edges $|\Omega|$. The first column gives the abbreviated dataset name. Under 'red.' we give the average reduction when compared to the full grid.

	Full				Convex Hull				Quadtree				OHG-1				OHG-2			
	$\|\Psi\|$	$\|\Omega\|$	rows	cols	$\|\Psi\|$	$\|\Omega\|$	rows	cols	$\|\Psi\|$	$\|\Omega\|$	rows	cols	$\|\Psi\|$	$\|\Omega\|$	rows	cols	$\|\Psi\|$	$\|\Omega\|$	rows	cols
F	3.7k	15.8k	0.1M	0.6M	1.8k	7.8k	46.5k	280.1k	1.2k	4.6k	31.6k	0.2M	1.7k	5k	43k	164.1k	3.7k	15.8k	0.1M	0.6M
V	5.5k	23.7k	0.1M	0.9M	2.3k	9.8k	64.9k	0.4M	1.3k	4.5k	36.1k	0.2M	2.1k	6.2k	59.6k	232.1k	5.5k	23.7k	0.1M	0.9M
ST	12.2k	52.7k	0.7M	4.3M	7k	30.3k	388.3k	2.5M	2.8k	10.4k	160.8k	0.8M	6.2k	19.9k	346.4k	1.5M	12.2k	52.7k	0.7M	4.3M
B	10.5k	45.6k	0.5M	3.5M	5.9k	25.2k	304.1k	1.9M	2.1k	7.9k	115.9k	0.6M	4.8k	15.1k	250.1k	1.1M	10.5k	45.6k	0.5M	3.5M
SD	14.9k	64.7k	0.6M	3.8M	9.4k	40.7k	370.4k	2.4M	2.3k	8.5k	97.6k	0.5M	5.7k	16.9k	228.4k	0.9M	14.9k	64.6k	0.6M	3.8M
red.					46%	47%	45%	46%	77%	80%	76%	80%	57%	69%	55%	71%	0%	0%	0%	0%

To overcome this problem, we use an altered orthoradial grid graph, as depicted in Figure 13, right. The innermost ring starts with a number $b = 8$ of nodes. This number is doubled each time the ring radius doubles. So at ring 1, $b_1 = 8$, at ring 2, $b_2 = 16$, $b_4 = 32$, $b_8 = 64$ and so on. This will give our maps a denser look while still maintaining an orthoradial layout and guaranteeing a minimum distance between nodes. We call this graph a *pseudo orthoradial* grid graph. Note that the length of a segment between adjacent nodes is always at least $\frac{\pi}{4}d$ in this configuration, so a minimum distance of d between adjacent nodes in the final drawing is not guaranteed. To guarantee a minimum distance of d, the inner ring radius would have to be set to $\frac{4}{\pi}d$. In practice, however, we found the difference of the resulting maps to be marginal.

Also in [27] it was observed that real-world orthoradial metro maps often feature a center node (Figure 13, gray). For our experiments in Section 6.3 we also added such a center node to our pseudo orthoradial grids. The center node was connected to 4 nodes on the first ring in an ortholinear fashion to guarantee a consistent maximum grid node degree of 4. The grid was always centered at the input node of highest degree (see for example Figure 15, where the grid center for Sydney is at the far right). The radius of the pseudo orthoradial grid was chosen such that it completely covered the padded bounding box of the input graph.

We chose grid edge weights which respect the length of the corresponding segment. Edges which connect two rings always have the same length and are weighted uniformly. Edges on circular segments received a uniform base weight on the inner ring. On all other rings, the weights were additionally multiplied by the ratio between the circular segment length on the respective ring and the circular segment length on the first ring.

6 EXPERIMENTAL EVALUATION

We implemented all grid types described above and tested them on five datasets of increasing complexity: the tram network of Freiburg, the subway network of Vienna, the light rail network of Stuttgart, the subway network of Berlin, and the light rail network of Sydney. We evaluated all grids using the ILP and the approximate approach described in Section 1.3.2, both with and without density penalty. The results can be found online.[2] For the sparse octilinear base grids, we compared their dimensions to the original full octilinear grid in Section 6.1. Their effect on the speed and quality of the schematization process will be measured in Section 6.2. In Section 6.3 we will present results following non-octilinear base grids. We consider the following as our main results:

(1) The sparse grids greatly reduce the base grid graph dimensions, up to a factor of 5. If an ILP is used to find the optimal metro map, the ILP dimensions are also greatly reduced.

(2) The convex hull grid and the octilinear Hanan grid have little impact on optimality, while the quadtree base grid produces results which are up to 53% worse.

(3) The base grid size reduction may speed up solution times, but does not do so consistently. If an ILP is used, some maps are generated much faster, but others take up to 6 times longer. If our approximate approach is used, the performance gains are slightly better on average, but still not consistently so.

(4) Our approach can render orthoradial and hexalinear metro maps in under 2.5 seconds for all datasets.

All experiments were performed on an AMD FX-8150 machine with 8 cores and 32 GB of memory. The ILPs were optimized using Gurobi 9.1.1 (we also experimented with GLPK and COIN-OR, but found Gurobi to be superior). The base grid size was set to the average distance between adjacent stations in the input graph and the degree 2 heuristic was employed.

6.1 Sparse Grid Graph Dimensions

Table 1 gives the effects of our sparse base grids Convex Hull, Quadtree, Octilinear Hanan Grid with 1 iteration (OHG-1) and Octilinear Hanan Grid with 2 iterations (OHG-2) on the dimensions of the base grid graph and the corresponding ILP.

In general, all sparse grids greatly reduced these dimensions. For the relatively simple Convex Hull approach, the number of nodes decreased by 46% on average, and the number of edges by 47%. The quadtree based grid even reduced the number of edges by 80% on average, while OHG-1 produced base graphs with 69% fewer edges than the full grid graph. The ILP sizes directly reflected this reduction. An octilinear Hanan grid with 2 iterations (OHG-2) already resulted in a nearly full grid for all test datasets. The size reductions were therefore negligible.

6.2 Performance of Sparse Grid Graphs

To measure the effect of the sparse grids on the quality of the ILP based solutions, we first determined the optimal target values on the full grid and then measured the approximation error introduced

[2] https://octi.informatik.uni-freiburg.de/flexmaps

Table 2: Effects of sparse base grids on ILP optimality. Θ^* is the optimal target value using a full grid, Θ is the target value for the respective simplified grid, $\delta = \Theta/\Theta^* - 1$ is the approximation error.

		Conv. Hull		Quadtree		OHG-1		OHG-2	
	Θ^*	Θ	δ	Θ	δ	Θ	δ	Θ	δ
F	115.9	116.8	0.8%	122.1	5.3%	117.8	1.6%	115.9	0%
V	127.6	128.5	0.7%	144.7	13.4%	128.1	0.4%	127.6	0%
ST	305.8	307.1	0.4%	334.3	9.3%	312.6	2.2%	310.7	1.6%
B	234.2	237.0	1.2%	268.4	14.6%	240.3	2.6%	234.2	0%
SD	291.4	291.4	0%	306.2	5.1%	295.0	1.2%	291.4	0%
avg.			0.6%		9.5%		1.6%		0.5%

by the sparse grids. The quality loss was small (< 2% on average) for all sparse grids except the Quadtree grid (Table 2). For our approximate approach, we compared the target values of all sparse grids with the target value of the *approximate* approach on the full grid (not the optimal target value found by our ILP). We also evaluated two variants of our approximate approach: the standard version which optimizes the same target function as the ILP, and the standard version with the additional density penalty described in Section 1.3.2. Additionally, we counted the number of topology violations introduced by our approximate approach due to the constraint relaxation described in Section 3. Only a single topology violation was introduced on the Quadtree grid for the Berlin dataset without the density penalty.

Using the standard version, the Convex Hull approach yielded slightly better results than the baseline, most likely because the cropped areas contained local optima. The error introduced by the other sparse grid types was comparable to the ILP approach (Table 3).

With the additional density penalty, the quality of the Quadtree approach greatly deteriorated (with an approximation error of up to 53.2%). The approximation error of the Convex Hull and OHG-1 approach also increased, but not as drastically (Table 4).

The effect of our sparse base grids on the solution times remained below our expectations. While the Convex Hull and OHG-1 grids were able to speed up the ILP solution time for the Freiburg, Vienna and Sydney datasets (by up to 84%), the solution times for the Stuttgart and Berlin dataset were often several times longer (Table 5). A similar effect was measured for our approximate approach, both with and without the density penalty (Tables 6 and 7).

For the approximate approach, we suspect several reasons for this: (1) The sparse grid graphs take longer to construct than a simple full grid. This is especially noticeable if we use multiple Hanan iterations, as is reflected in the increased running time for the OHG-2 approach in Table 6. (2) Simple look-ups like neighboring grid nodes are easy in a full grid graph, but not so on quadtree grids or octilinear Hanan grids. (3) Aspects of locality may slow down the shortest path calculations on the quadtree grid and the octilinear Hanan grid. In a full grid, we add grid nodes from left to right, which already leads to neighboring grid nodes being close together in memory. For both OHG variants and the quadtree grid, the memory layout is much more irregular. (4) More generally, a sparse base grid makes it more difficult to find a first feasible solution. This is especially true for input graphs with high node densities, which is reflected by the increased solution times for Stuttgart and Berlin. As mentioned above, if our approximate approach stalls with the initial routing ordering of the input edges, it tries random edge orderings until a feasible solution is found.

For the ILP, we also suspect (4) to be the main reason for the increased solution times. It may also be that the irregularity of the sparse grids might compromise some heuristics of the ILP solver.

We note that there is practical value in sparse base grids apart from faster solution times: as the sparse base grids produce much smaller ILPs, they require less memory to solve. For example, to optimize the ILP of the Sydney dataset, we required 28.9 GB of memory using a full octilinear grid, and only 8.3 GB using an octilinear Hanan grid.

6.3 Hexalinear and Orthoradial Experiments

We evaluated all our testing datasets on a triangular base grid, producing hexalinear maps, and on a pseudo orthoradial base grid. Figure 14 shows the hexalinear results of our approximate approach with density penalty for Berlin, Vienna, and Sydney, as rendered by LOOM [7], with a simple a-posteriori labeling which was not part of the approach described in this work. Figure 15 shows the orthoradial results using the same setup. The solution times as well as the approximation errors introduced by our approximate approach can be found in Table 8. We were able to render all maps in under 2.5 seconds, with zero topology violations for all datasets except Berlin, where a single violation was introduced in the orthoradial map (highlighted in red in Figure 15). The slower solution times when compared to octilinear maps can partly be explained by the lower maximum node degree of the grid graphs, which makes it harder to find feasible solutions. We also used a carefully designed A^* heuristic to speed up the shortest-path computations in the octilinear setting, but not on the hexalinear or pseudo orthoradial grid. The schematized maps for all datasets can be found online.[3]

7 CONCLUSION AND FUTURE WORK

We have extended an earlier approach of drawing metro maps as images on regular octilinear grid graphs by considering other grid graphs and making them more sparse. We experimented with octilinear base grids cropped to the convex hull of the input graph, octilinear base grids constructed in a quadtree fashion, octilinear Hanan grids, pseudo orthoradial grids, and triangular grids.

These sparse octilinear base grids were able to greatly reduce the problem size with only minimal impact on the solution optimality for the convex hull and the octilinear Hanan grid approach. This problem size reduction did not always result in the expected performance gains. We discussed possible reasons for this in Section 6.2.

Using pseudo orthoradial and triangular grids, we were able to render esthetically pleasing orthoradial and hexalinear drawings of our test datasets in under 2.5 seconds. We see further speed-up potential by combining the shortest-path computations with an A^* heuristic.

[3] https://octi.informatik.uni-freiburg.de/flexmaps

Table 3: Effects of sparse base grids on the optimality of our approximate approach. δ gives the approximation error compared to the approximate approach on the full grid. A negative δ means the result was better than the result on the full grid. Under 'vio.' we give the number of topology violations introduced by our approximate approach due to constraint relaxation.

	Full		Conv. Hull			Quadtree			OHG-1			OHG-2		
	Θ	vio.	Θ	vio.	δ	Θ	vio.	δ	Θ	vio.	δ	Θ	vio.	δ
F	117.0	0	117.9	0	0.7%	125.5	0	7.3%	120.2	0	2.8%	117.0	0	0%
V	128.6	0	129.6	0	0.8%	153.6	0	19.5%	132.1	0	2.7%	128.6	0	0%
ST	328.4	0	328.7	0	0.1%	364.7	0	11.1%	338.3	0	3.1%	330.1	0	0.5%
B	258.5	0	257.9	0	-0.2%	$> w_\infty$	1	—	260.2	0	0.6%	256.3	0	-0.8%
SD	318.0	0	298.6	0	-6.1%	316.0	0	0.6%	314.9	0	0.9%	318.0	0	0%
avg.					-0.9%			9.3%			1.6%			0%

Table 4: Effects of sparse base grids on the optimality of our approximate approach with additional density penalty.

	Full		Conv. Hull			Quadtree			OHG-1			OHG-2		
	Θ	vio.	Θ	vio.	δ	Θ	vio.	δ	Θ	vio.	δ	Θ	vio.	δ
F	144.8	0	158.5	0	9.5%	183.3	0	26.7%	160.6	0	10.9%	144.8	0	0%
V	174.5	0	177.8	0	1.8%	267.4	0	53.2%	206.1	0	18.1%	168.7	0	-3.3%
ST	401.2	0	405.8	0	1.1%	557.3	0	38.9%	422.9	0	5.4%	401.4	0	-0.1%
B	314.4	0	318.5	0	1.3%	456.2	0	45.1%	333.3	0	6.1%	311.5	0	0%
SD	385.7	0	387.7	0	0.5%	540.8	0	40.2%	387.3	0	0.4%	385.7	0	0%
avg.					2.9%			39.7%			8.1%			-0.1%

Table 5: Effects of sparse base grids on ILP solution times. Under 'red.' we give the reduction when compared to the original solution time on the full grid in percent.

	Full	Conv. Hull		Quadtree		OHG-1		OHG-2	
	t	t	red.	t	red.	t	red.	t	red.
F	3m	2m	35%	1m	63%	1.3m	57%	3m	-1%
V	1.3h	54m	33%	18m	77%	18m	77%	1.8h	-37%
ST	46m	3.2h	-312%	4.5h	-480%	1h	-29%	1h	-30%
B	2h	14.5h	-637%	5.2h	-161%	19.2h	-870%	2.6h	-36%
SD	3.5h	2.3h	35%	1h	72%	34m	84%	2.6h	21%
avg.	1.5h	3.7h	-169%	2.2h	-66%	4.2h	-136%	1.7h	-16%

Table 6: Effects of sparse base grids on solution times of our approximate approach, when compared to the full grid.

	Full	Conv. Hull		Quadtree		OHG-1		OHG-2	
	t	t	red.	t	red.	t	red.	t	red.
F	56ms	39ms	30%	56ms	-1%	62ms	-11%	63ms	-13%
V	78ms	53ms	32%	313ms	-301%	61ms	21%	107ms	-37%
ST	266ms	115ms	57%	386ms	-45%	176ms	34%	330ms	-24%
B	158ms	213ms	-34%	723ms	-356%	205ms	-30%	375ms	-137%
SD	215ms	323ms	-50%	239ms	-11%	148ms	31%	256ms	-19%
avg.	155ms	149ms	7%	344ms	-90%	130ms	9%	226ms	-46%

We are also convinced that octilinear Hanan grids should be further studied in the context of octilinear metro maps. For example, it would be interesting to evaluate their application to restrict the search space of other approaches.

Table 7: Effects of sparse base grids on solution times of our approximate approach with additional density penalty, when compared to the full grid.

	Full	Conv. Hull		Quadtree		OHG-1		OHG-2	
	t	t	red.	t	red.	t	red.	t	red.
F	116ms	102ms	12%	146ms	-26%	85ms	26%	131ms	-13%
V	244ms	105ms	57%	427ms	-75%	147ms	40%	307ms	-26%
ST	440ms	395ms	10%	542ms	-23%	676ms	-54%	552ms	-25%
B	301ms	302ms	-0.4%	931ms	-209%	457ms	-51%	510ms	-66%
SD	340ms	362ms	-7%	349ms	-2%	432ms	-27%	409ms	-20%
avg.	288ms	253ms	14%	479ms	-32%	360ms	-13%	382ms	-31%

Table 8: Solution times using an ILP, our approximate approach (A), and our approximate approach with density penalty (A+D) on hexalinear and pseudo orthoradial grids.

	Hexalinear Grid				Pseudo Orthoradial Grid			
	ILP	A	δ	A+D	ILP	A	δ	A+D
F	3.8m	138ms	7.1%	313ms	50s	234ms	17.5%	283ms
V	6.5m	146ms	1.2%	654ms	18.2m	145ms	7.6%	351ms
ST	43.6m	616ms	15.3%	1.3s	28.7m	706ms	14.8%	1.9s
B	1.8h	470ms	17.3%	1.4s	8.4h	2.4s	17.8%	2.5s
SD	23.5m	1.6s	8.5%	2.2s	23.2m	468ms	6.0%	1.7s
avg.	0.6h	594ms	9.9%	1.2s	1.9h	791ms	12.7%	1.3s

ACKNOWLEDGMENTS

This work was partially funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) - Project-ID 50974019 - TRR 161.

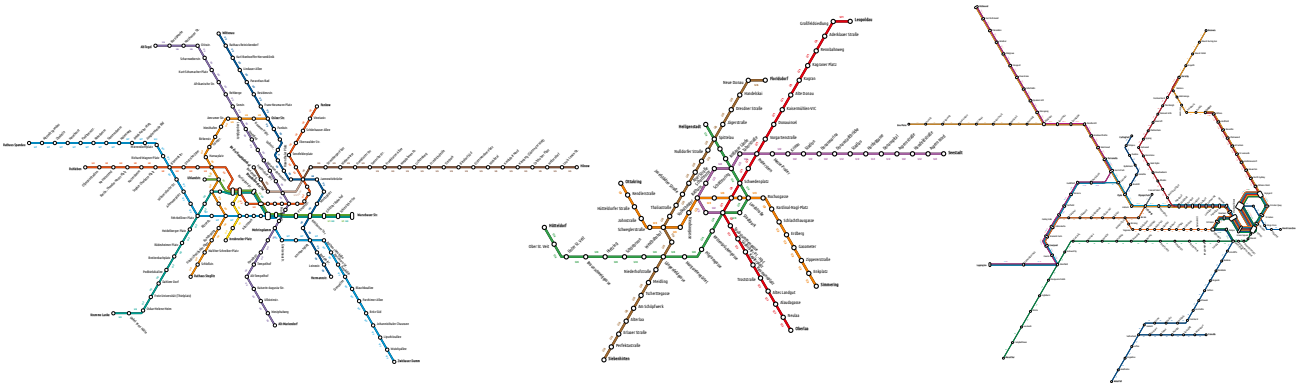

Figure 14: Hexalinear drawings of the Berlin, Vienna, and the Sydney network. Produced by our approximate approach with density penalty.

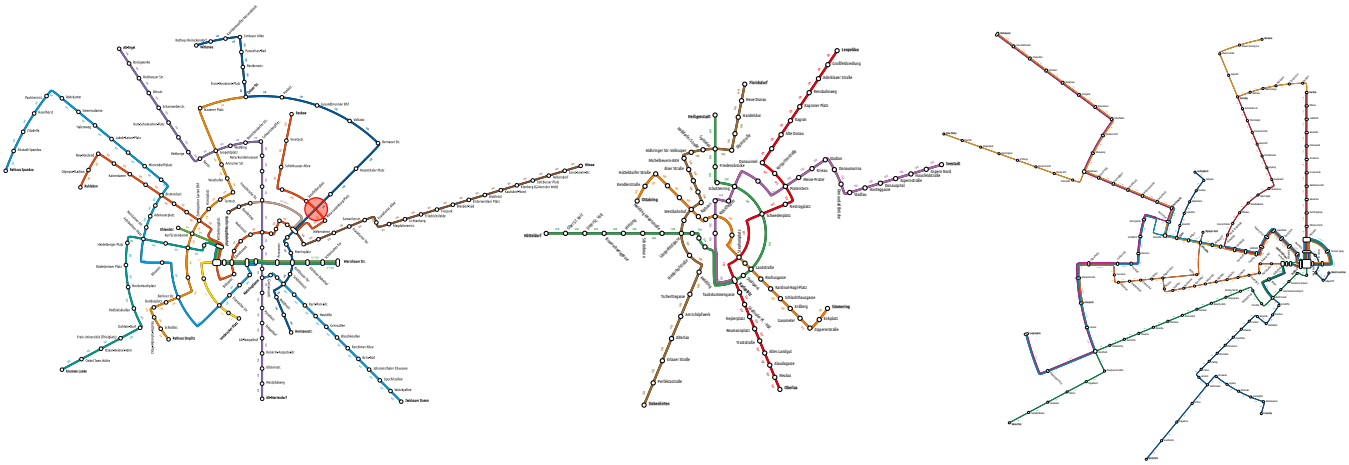

Figure 15: Orthoradial drawings of the Berlin, Vienna, and the Sydney network. Produced by our approximate approach with density penalty. For Berlin, a single topology violation was introduced, highlighted in red.

REFERENCES

[1] Suchith Anand, Silvania Avelar, J Mark Ware, and Mike Jackson. 2007. Automated schematic map production using simulated annealing and gradient descent approaches. In *GISRUK*, Vol. 7.

[2] Evmorfia N. Argyriou, Michael A. Bekos, Michael Kaufmann, and Antonios Symvonis. 2010. On Metro-Line Crossing Minimization. *J. Graph Algorithms Appl.* 14, 1 (2010), 75–96.

[3] Matthew Asquith, Joachim Gudmundsson, and Damian Merrick. 2008. An ILP for the metro-line crossing problem. In *CATS 2008, Wollongong, NSW, Australia, January 22-25, 2008*, Vol. 77. 49–56.

[4] Silvania Avelar and Matthias Müller. 2000. *Generating topologically correct schematic maps.* Technical Report. ETH Zurich.

[5] Lukas Barth, Benjamin Niedermann, Ignaz Rutter, and Matthias Wolf. 2017. Towards a Topology-Shape-Metrics Framework for Ortho-Radial Drawings. In *SoCG 2017, July 4-7, 2017, Brisbane, Australia*, Vol. 77. 14:1–14:16.

[6] Hannah Bast, Patrick Brosi, and Sabine Storandt. 2018. Efficient generation of geographically accurate transit maps. In *SIGSPATIAL 2018, Seattle, WA, USA, November 06-09, 2018.* 13–22.

[7] Hannah Bast, Patrick Brosi, and Sabine Storandt. 2019. Efficient Generation of Geographically Accurate Transit Maps. *ACM Trans. Spatial Algorithms Syst.* 5, 4 (2019), 25:1–25:36.

[8] Hannah Bast, Patrick Brosi, and Sabine Storandt. 2020. Metro Maps on Octilinear Grid Graphs. *Comput. Graph. Forum* 39, 3 (2020), 357–367.

[9] Michael A. Bekos, Michael Kaufmann, Katerina Potika, and Antonios Symvonis. 2007. Line Crossing Minimization on Metro Maps. In *GD 2007, Sydney, Australia, September 24-26, 2007*, Vol. 4875. 231–242.

[10] Marc Benkert, Martin Nöllenburg, Takeaki Uno, and Alexander Wolff. 2006. Minimizing Intra-edge Crossings in Wiring Diagrams and Public Transportation Maps. In *GD 2006, Karlsruhe, Germany, September 18-20, 2006*, Vol. 4372. 270–281.

[11] Sergio Cabello, Mark de Berg, Steven van Dijk, Marc J. van Kreveld, and Tycho Strijk. 2001. Schematization of road networks. In *Proceedings of the Seventeenth Annual Symposium on Computational Geometry, Medford, MA, USA, June 3-5, 2001.* 33–39.

[12] Daniel Chivers and Peter Rodgers. 2014. Octilinear Force-Directed Layout with Mental Map Preservation for Schematic Diagrams. In *Diagrammatic Representation and Inference - 8th International Conference, Diagrams 2014, Melbourne, VIC, Australia, July 28 - August 1, 2014*, Vol. 8578. 1–8.

[13] Daniel Elroi. 1988. Designing a network line-map schematization software enhancement package. In *Proc. 8th Ann. ESRI User Conference.*

[14] Daniel Elroi. 1988. GIS and schematic maps: A new symbiotic relationship. In *Proc. GIS/LIS*, Vol. 88.

[15] Martin Fink, Herman J. Haverkort, Martin Nöllenburg, Maxwell J. Roberts, Julian Schuhmann, and Alexander Wolff. 2012. Drawing Metro Maps Using Bézier Curves. In *GD 2012, Redmond, WA, USA, September 19-21, 2012.* 463–474.

[16] Martin Fink and Sergey Pupyrev. 2013. Metro-Line Crossing Minimization: Hardness, Approximations, and Tractable Cases. In *GD 2013, Bordeaux, France, September 23-25, 2013*, Vol. 8242. 328–339.

[17] Fabian Frank, Michael Kaufmann, Stephen G. Kobourov, Tamara Mchedlidze, Sergey Pupyrev, Torsten Ueckerdt, and Alexander Wolff. 2021. Using the Metro-Map Metaphor for Drawing Hypergraphs. In *SOFSEM 2021, Bolzano-Bozen, Italy, January 25-29, 2021*, Vol. 12607. 361–372.

[18] Maurice Hanan. 1966. On Steiner's problem with rectilinear distance. *SIAM J. Appl. Math.* 14, 2 (1966), 255–265.
[19] Seok-Hee Hong, Damian Merrick, and Hugo A. D. do Nascimento. 2004. The Metro Map Layout Problem. In *GD 2004, New York, NY, USA, September 29 - October 2, 2004*, Vol. 3383. 482–491.
[20] Seok-Hee Hong, Damian Merrick, and Hugo AD do Nascimento. 2006. Automatic visualisation of metro maps. *Journal of Visual Languages & Computing* 17, 3 (2006), 203–224.
[21] Andrea S. LaPaugh and Ronald L. Rivest. 1978. The Subgraph Homeomorphism Problem. In *STOC 1978, May 1-3, 1978, San Diego, California, USA*. 40–50.
[22] Damian Merrick and Joachim Gudmundsson. 2006. Path Simplification for Metro Map Layout. In *GD 2006, Karlsruhe, Germany, September 18-20, 2006*, Vol. 4372. 258–269.
[23] Tal Milea, Okke Schrijvers, Kevin Buchin, and Herman J. Haverkort. 2011. Shortest-Paths Preserving Metro Maps. In *GD 2011, Eindhoven, The Netherlands, September 21-23, 2011*, Vol. 7034. 445–446.
[24] Gabriele Neyer. 1999. Line Simplification with Restricted Orientations. In *WADS'99, Vancouver, British Columbia, Canada*, Vol. 1663. 13–24.
[25] Soeren Nickel and Martin Nöllenburg. 2019. Drawing k-linear Metro Maps. *CoRR* abs/1904.03039 (2019).
[26] Benjamin Niedermann and Ignaz Rutter. 2020. An Integer-Linear Program for Bend-Minimization in Ortho-Radial Drawings. In *GD 2020, Vancouver, BC, Canada, September 16-18, 2020*, Vol. 12590. 235–249.
[27] Benjamin Niedermann, Ignaz Rutter, and Matthias Wolf. 2019. Efficient Algorithms for Ortho-Radial Graph Drawing. In *SoCG 2019, June 18-21, 2019, Portland, Oregon, USA*, Vol. 129. 53:1–53:14.
[28] Martin Nöllenburg. 2009. An Improved Algorithm for the Metro-line Crossing Minimization Problem. In *GD 2009, Chicago, IL, USA, September 22-25, 2009*, Vol. 5849. 381–392.
[29] Martin Nöllenburg and Alexander Wolff. 2005. A Mixed-Integer Program for Drawing High-Quality Metro Maps. In *GD 2005, Limerick, Ireland, September 12-14, 2005*. 321–333.
[30] Martin Nöllenburg and Alexander Wolff. 2011. Drawing and Labeling High-Quality Metro Maps by Mixed-Integer Programming. *IEEE Trans. Vis. Comput. Graph.* 17, 5 (2011), 626–641.
[31] Jonathan M. Stott and Peter Rodgers. 2004. Metro Map Layout Using Multicriteria Optimization. In *IV 2004, 14-16 July 2004, London, UK*. 355–362.
[32] Jonathan M. Stott, Peter Rodgers, Juan Carlos Martinez-Ovando, and Stephen G. Walker. 2011. Automatic Metro Map Layout Using Multicriteria Optimization. *IEEE Trans. Vis. Comput. Graph.* 17, 1 (2011), 101–114.
[33] Peng Ti and Zhilin Li. 2014. Generation of schematic network maps with automated detection and enlargement of congested areas. *Int. J. Geogr. Inf. Sci.* 28, 3 (2014), 521–540.
[34] Thomas C. van Dijk and Dieter Lutz. 2018. Realtime linear cartograms and metro maps. In *SIGSPATIAL 2018, Seattle, WA, USA, November 06-09, 2018*. 488–491.
[35] Thomas C. van Dijk, Arthur van Goethem, Jan-Henrik Haunert, Wouter Meulemans, and Bettina Speckmann. 2014. Map Schematization with Circular Arcs. In *GIScience 2014, Vienna, Austria, September 24-26, 2014*, Vol. 8728. 1–17.
[36] J. Mark Ware, George E. Taylor, Suchith Anand, and Nathan Thomas. 2006. Automated Production of Schematic Maps for Mobile Applications. *Trans. GIS* 10, 1 (2006), 25–42.
[37] Hsiang-Yun Wu, Benjamin Niedermann, Shigeo Takahashi, Maxwell J. Roberts, and Martin Nöllenburg. 2020. A Survey on Transit Map Layout - from Design, Machine, and Human Perspectives. *Comput. Graph. Forum* 39, 3 (2020), 619–646.
[38] Hsiang-Yun Wu, Shigeo Takahashi, Daichi Hirono, Masatoshi Arikawa, Chun-Cheng Lin, and Hsu-Chun Yen. 2013. Spatially Efficient Design of Annotated Metro Maps. *Comput. Graph. Forum* 32, 3 (2013), 261–270.
[39] Hsiang-Yun Wu, Shigeo Takahashi, Chun-Cheng Lin, and Hsu-Chun Yen. 2012. Travel-Route-Centered Metro Map Layout and Annotation. *Comput. Graph. Forum* 31, 3 (2012), 925–934.